A Comprehensive Analysis of Power Consumption and Resources Utilization in Open-Source and Proprietary Media Players

Written by

©Afzal Ahmed

A Thesis Submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science
Memorial University of Newfoundland

May 2025

St. John's Newfoundland and Labrador Canada

Abstract

The growing demand for high-quality media consumption has highlighted the importance of energy-efficient software, particularly media players that handle high-resolution video content. As public is concerned around environmental sustainability and energy use, evaluating the power consumption of software applications has become crucial. This thesis investigates the comparative energy efficiency of open-source and proprietary media players, with a focus on CPU, GPU, and memory consumption during high-resolution video playback. By analyzing resource usage across different platforms, this research aims to provide insights into how software architecture, codec support, and hardware acceleration affect the overall energy consumption of these media players. Open-source media players, such as VLC and MPV, are widely adopted due to their flexibility, cost-effectiveness, and support for a wide range of media formats. However, these players often rely heavily on CPU resources, particularly when hardware acceleration is not fully optimized. This can result in higher power consumption during high-demand tasks such as 4K video playback, especially on platforms where driver support for hardware acceleration is limited. Despite this, open-source players can be energy-efficient when optimized codecs like VP9 and AV1 are used, reducing file sizes and overall power consumption. Proprietary media players, including GOM Player and Windows Media Player, generally outperform their open-source counterparts in terms of energy use. These players benefit from close integration with hardware manufacturers, which allows for better utilization of hardware acceleration and more efficient resource management. Proprietary codecs such as H.264 and H.265 are optimized for energy savings by offloading video processing to the GPU, leading to lower CPU usage and reduced power consumption. The structured support and regular updates that come with proprietary software ensure that these players remain well-optimized for performance and energy efficiency over time.

The study utilized real-time power consumption monitoring tools, including HWiNFO and PowerTOP, to assess the performance of both open-source and proprietary media players during high-definition video playback. Metrics such as CPU and GPU power consumption, memory usage, and overall system resource utilization were analyzed in various playback scenarios. The results indicate that proprietary media players typically consume less power due to optimized hardware and software integration, while open-source players can achieve competitive efficiency levels with appropriate codec and hardware configurations. In terms of long-term sustainability, proprietary players tend to offer more immediate energy savings, particularly in environments where media playback is frequent. However, open-source media players, with their flexibility and user-driven customization, present opportunities for power savings over time, especially in cost-sensitive environments. This thesis contributes to the understanding of software energy efficiency, providing valuable insights for developers and users aiming to optimize their media playback experience for reduced energy consumption and environmental impact.

Co-authorship Statement

I am the main author of all the research papers compiled in composing this thesis, my supervisor and co-supervisor, Dr. Tariq Iqbal and Dr. Mohsin Jamil respectively, are co-author of all articles. As the lead author, I did most of the research work, the literature reviews, carried out the experiments, experimental setups, and analysis of the results in each of the manuscripts. I also formulated the original manuscripts and later edited each of them based on comments from the co-authors and peer reviewers throughout the peer review process. Co-authors Dr. Tariq Iqbal and Dr. Mohsin Jamil supervised all the research work, revised, and corrected each one of the manuscripts, provided research material, contributed research ideas throughout the research, and updated each of the manuscripts.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Mohammad Tariq Iqbal, for the invaluable guidance, support, and encouragement throughout the course of this thesis. His insights and expertise have been instrumental in shaping this research, and I am incredibly fortunate to have had the opportunity to learn under his mentorship. I would also like to extend my heartfelt thanks to my co-supervisor, Dr. Mohsin Jamil, whose advice, feedback, constant motivation and support greatly enriched this work.

I owe immense appreciation to my family for their unwavering support during this journey. To my mother, who has always been my pillar of strength, and to my brother and sister, their belief in me has kept me grounded and focused. I would like to thank my wife, who stayed back home and shouldered the responsibilities during my absence, and to my sons, who stayed away but remained my constant source of motivation.

I am also deeply grateful to my friends, especially Khan Awais Khan, for his support, camaraderie, and encouragement throughout this period. His presence has been a source of comfort and joy during challenging times.

Table of Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

CPU	Central Processing Unit
GPU	Graphics Processing Unit
RAM	Random Access Memory
OS	Operating System
HDD	Hard Disk Drive
FPS	Frames Per Second
W	Watts
kWh	Kilowatt-hour
MB	Megabytes
GB	Gigabytes
HWiNFO	Hardware Information tool
VLC	VideoLAN Client
MPC-HC	Media Player Classic Home Cinema
WMP	Windows Media Player
KMPlayer	Korean Media Player
GOM Player	Gretech Online Movie Player

Chapter 1

Introduction and Literature Review

1.1. Introduction

In today's rapidly evolving digital landscape, media consumption plays a pivotal role in daily life. With the growing demand for high-quality media playback, the efficiency of software applications, specifically media players, has come under scrutiny. Media players are integral to the consumption of various audio and video formats, but they also consume significant system resources, including CPU, GPU, and memory. As the importance of energy conservation grows, software's role in overall power consumption has become a critical area of research. Studies have demonstrated that even small variations in software efficiency can lead to substantial differences in power consumption.

The distinction between open-source and proprietary software in terms of energy consumption is significant. Open-source software (OSS) such as VLC and Kodi are renowned for its flexibility, community-driven development, and lack of licensing fees. Studies by Panayides et al. (2020) have shown that open-source solutions like VLC are highly customizable and adaptable, which can lead to more efficient use of system resources in certain context. However, research also highlights that open-source players may demand more CPU and memory resources when handling complex media formats, such as raw video files. Mahmoud et al. (2023) suggest that while OSS can be optimized for performance, it often lacks the professional-grade resource management found in proprietary solutions.

Proprietary media players, such as Windows Media Player and GOM Player, are developed with a focus on performance and user experience. These players typically benefit from extensive optimization, especially in handling resource-intensive tasks like 4K video playback. Ohm et al. (2012) noted that proprietary codecs, such as H.264 and H.265, are specifically optimized for energy efficiency, making proprietary players more suitable for professional environments where performance and uptime are crucial. However, this performance advantage often comes at the cost of increased power consumption.

The purpose of this thesis is to provide a comparative analysis of power consumption between open-source and proprietary media players, focusing on their performance during high-resolution video playback. By examining how different players utilize CPU, GPU, and memory resources, this research aims to provide insights into software energy efficiency and its implications for sustainability. Tools such as HWiNFO will be used to measure key performance indicators, and the findings will contribute to the broader understanding of sustainable software practices.

1.2. Literature Review

The comparison between open-source and proprietary software has been a significant area of research, particularly in terms of performance, energy efficiency, and resource utilization. Open-source software (OSS) has been praised for its adaptability, cost-effectiveness, and transparency, while proprietary software is often chosen for its stability, performance optimization, and dedicated customer support [1], [2]. This section synthesizes findings from multiple studies cited in the provided research papers, highlighting the key differences in power consumption and efficiency between open-source and proprietary media players.

1.2.1. Power Consumption and Energy Efficiency

Several studies have focused on energy efficiency in software applications, particularly media players, given the growing demand for high-definition video content. Xiao et al. [3] proposed a software-based power estimation model that utilizes machine learning to predict application power usage based on resource utilization metrics. This approach is particularly useful for understanding how media players manage CPU, GPU, and memory resources.

Research by Santos et al. [4] explored power profiling techniques using hardware counters and system calls, which provide detailed insights into resource utilization. The study highlighted that proprietary media players often consume less power than their open-source counterparts due to better optimization for hardware acceleration. This finding aligns with research by Zhao et al. [5], who found that simpler algorithms tend to consume less power in software applications, reinforcing the notion that proprietary software, often designed for performance, can be more energy-efficient. However, open-source media players, such as VLC and Kodi, have been shown to outperform proprietary players in some cases, particularly in GPU power consumption. Research by Chen et al. [6] demonstrated that VLC's support for modern codecs, including H.265 and AV1, allows for more efficient video decoding and playback, reducing overall power consumption. Similarly, Akramullah [7] emphasized that OSS media players benefit from community-driven updates that often prioritize resource efficiency, especially in cases where hardware acceleration is available.

1.2.2. Resource Utilization: CPU and GPU

The efficient use of system resources, such as CPU and GPU, is critical in determining the power consumption of media players. Studies show that proprietary media players, such as GOM Player and KMPlayer, are often optimized for better hardware acceleration, reducing CPU load and thus power consumption [8], [9]. Park et al. [10] explored the benefits of CPU-GPU frequency scaling, noting that Windows-based media players tend to utilize GPU resources more effectively due to superior driver support, resulting in lower energy usage during tasks such as 4K video playback. Conversely, open-source media players can exhibit higher CPU usage due to less frequent optimizations. Youssef [11] found that OSS media players, such as MPV and Kodi, tend to rely more heavily on CPU resources, particularly in Linux environments where driver support for hardware acceleration is limited. This was supported by research from Duarte et al. [12], who examined the impact of code optimization on energy efficiency and found that proprietary media players tend to implement more aggressive optimizations that reduce CPU usage.

1.2.3. Codec Support and Format Handling

Codec support plays a significant role in the power efficiency of media players. Proprietary media players typically support a range of optimized codecs that reduce the power required for decoding high-resolution video files. Akramullah [13] noted that codecs such as H.264 and H.265, commonly used in proprietary players like Windows Media Player and GOM Player, are designed to minimize energy consumption by offloading tasks to the GPU.

Open-source media players, on the other hand, offer broader codec support, including newer, royalty-free codecs like VP9 and AV1. These codecs, though efficient in terms of compression, can lead to higher CPU usage during playback, as observed by Panayides et al. [14]. Studies by Tudor and Teo [15] suggest that while open-source players can handle a wider range of media formats, they often require more resources to decode these formats, particularly in systems lacking hardware acceleration.

1.2.4. Long-Term Energy Consumption and Sustainability

The long-term energy consumption of media players is an important consideration, particularly for users who regularly stream high-definition content. Xiao et al. [16] conducted a study that projected the energy consumption of various media players over time, showing that proprietary players typically consume less power due to their efficient resource management. This finding is echoed by Deng et al. [17], who noted that long-term energy savings can be substantial when using proprietary media players in environments where media playback occurs frequently.

Open-source media players, while offering the advantage of no licensing fees, may result in higher cumulative energy consumption over extended periods [18]. Research by Duarte et al. [19] highlights the importance of integrating energy efficiency considerations into software design to ensure that both short-term and long-term usage scenarios are optimized for sustainability.

1.2.5. Customization and Flexibility in Open-Source Software

One of the key advantages of open-source software is its flexibility and customizability. Studies have shown that OSS media players can be tailored to specific user needs, allowing for greater control over resource usage and power consumption [20], [21]. However, this flexibility comes with the trade-off of requiring more technical expertise to implement optimizations, which may not be feasible for all users [22]. Mahmoud et al. [23] emphasized that while open-source media players offer significant customization options, they often lack the dedicated support and structured updates found in proprietary software, which can result in less efficient resource management.

1.2.6. Energy Profiling Techniques and Measurement Tools

A significant body of research focuses on the techniques and tools used to profile energy consumption in software applications, particularly media players. Santos et al. [4] introduced a framework for energy profiling using hardware counters and system calls, which can provide detailed insights into how different software components impact overall power consumption. This method has been widely adopted in subsequent studies, including by Deng et al. [17], who applied similar techniques to measure the energy efficiency of open-source versus proprietary media players.

Xiao et al. [3] employed a machine learning-based model to predict power consumption, offering a novel approach to profiling software energy use. This method is particularly effective for analyzing real-time energy consumption during media playback. Similarly, studies by Duarte et al. [12] and Zhao et al. [5] have demonstrated the importance of using accurate, high-resolution power monitoring tools, such as HWiNFO and PowerTOP, to capture real-time data on CPU and GPU utilization. These tools have become integral to the analysis of media player efficiency, particularly in experiments comparing open-source and proprietary software.

1.2.7. Impact of Software Architecture on Energy Consumption

The architecture of software applications plays a crucial role in determining their energy efficiency. Studies have shown that software designed with modularity and optimization in mind can significantly reduce power consumption. Bhatia et al. [24] explored the energy implications of different software design choices, noting that applications with simpler, modular architectures tend to consume less power than those with more complex structures. This finding is particularly relevant to the comparison of open-source and proprietary media players, as open-source software often emphasizes modularity, allowing for more efficient resource management in specific use cases [25].

Research by Komu et al. [26] examined the role of software design in balancing CPU and GPU workloads. The study found that proprietary media players, which are typically developed with performance optimization in mind, are better at distributing tasks between the CPU and GPU, resulting in lower overall power consumption. In contrast, open-source media players may require more frequent updates and optimizations to achieve similar levels of efficiency.

1.2.8. Software Features and Functionalities: A Source of Power Drain

Certain software features and functionalities can significantly increase power consumption. Mahajan et al. [27] compared the energy usage of different media player features, demonstrating that functionalities such as video playback, complex scripting, and background processes can lead to higher energy demands. This observation is particularly relevant to proprietary media players, which often include advanced features and background tasks that consume additional power. In contrast, open-source media players, such as VLC, tend to prioritize simplicity and flexibility, which can result in lower power consumption when only basic functionalities are used [6].

The trade-off between features and power efficiency is further explored by Santos et al. [4], who analyzed the power consumption of various word processing and media applications. Their findings suggest that media players with extensive feature sets, such as Windows Media Player and GOM Player, consume more power than their open-source counterparts, particularly when handling complex tasks like high-definition video playback.

1.2.9. Power Consumption in Mobile and Cross-Platform Media Players

Energy efficiency in mobile and cross-platform media players is an emerging area of research. Deng et al. [17] compared the power consumption of media players on mobile devices, finding that applications optimized for specific platforms, such as Windows or Android, tend to consume less power than those designed to be cross-platform. This study highlights the importance of platform-specific optimizations in reducing energy consumption, particularly in resource-constrained environments.

Hans et al. [28] expanded on this research by examining the differences in power consumption between native and web-based media players. Their study found that native applications, such as Windows Media Player, tend to be more energy-efficient than web-based alternatives, which rely on continuous network connectivity and higher CPU usage for video decoding and rendering. This finding is particularly relevant to proprietary media players, which often include platform-specific optimizations that reduce power consumption during extended usage [29].

1.2.10.Security, Maintenance, and Support Considerations

Security and maintenance are critical factors in the comparison between open-source and proprietary media players. Open-source media players, while benefiting from transparency and community-driven updates, can suffer from inconsistent patching and support for less popular projects. Mahmoud et al. [23] noted that security vulnerabilities in open-source software are often addressed quickly by the community, but proprietary media players benefit from dedicated support teams and structured maintenance schedules, which can result in more reliable performance over time.

Tudor and Teo [15] emphasized the role of professional support in maintaining the stability and security of proprietary media players. Their study highlighted the importance of regular updates and security patches, which are typically more consistent in proprietary software ecosystems. However, studies such as those by Le Feuvre et al. [30] have demonstrated that open-source frameworks, like GPAC, can achieve similar levels of security and performance when supported by active development communities.

1.2.11.User-Centric Power Consumption Monitoring

One of the key advancements in recent research is the development of user-friendly power consumption monitoring tools. Katal et al. [31] explored how end-users can monitor their software's energy consumption using tools like Powerstat and Open Hardware Monitor. These tools allow users to track real-time power usage and make informed decisions about their software choices based on energy efficiency metrics. Xiao et al. [16] further supported this approach, suggesting that power consumption monitoring should be integrated into the software development lifecycle to promote sustainable coding practices.

Studies by Zhang et al. [32] and Singh et al. [33] have demonstrated that open-source tools often provide more granular energy profiling compared to proprietary alternatives, making them valuable for users focused on minimizing their environmental footprint. However, proprietary media players, due to their seamless integration with platform-specific monitoring tools, can offer a more polished user experience when it comes to tracking energy consumption [34].

1.2.12.Energy use in Open-Source vs. Proprietary Media Players

Energy efficiency remains a critical factor in evaluating the performance of open-source and proprietary media players. Studies comparing their energy consumption during video playback have revealed that proprietary players generally consume less power due to their optimized resource management and use of proprietary codecs [1], [14]. Mahmoud et al. [23] highlight that proprietary software is often developed with performance and resource efficiency in mind, particularly for high-demand tasks like 4K video playback.

On the other hand, Panayides et al. [14] argue that open-source players such as VLC and MPV, despite their flexibility and support for a wide range of codecs, may consume more CPU resources due to the lack of extensive hardware acceleration optimization. However, research also shows that open-source media players can achieve lower power consumption when playing certain file types, such as WebM and VP9, where they benefit from modern, royalty-free codecs optimized for open platforms. Open-source media players also distinguish themselves through their cross-platform availability, which ensures compatibility across operating systems such as Windows, macOS, Linux, and even mobile platforms. This universal accessibility is particularly valuable for users who rely on multiple devices or older hardware, as open-source players frequently maintain support for legacy systems [15], [35].

1.2.13.Codec Optimization and Resource Distribution

The efficiency of media players is strongly influenced by the codecs they support and how these codecs manage system resources. Proprietary media players are known for their integration of highly optimized codecs like H.264 and H.265, which are designed to minimize CPU usage and offload more tasks to the GPU [36]. Unlike proprietary alternatives, open-source software allows users to examine and verify its source code, ensuring there are no hidden mechanisms for data collection or invasive tracking. This focus on transparency is complemented by the freedom to fine-tune playback settings, buffering behavior, and other technical parameters, enabling users to optimize the software for their unique hardware or use case. [7]. Akramullah [13] noted that the development of proprietary codecs typically involves close collaboration between software developers and hardware manufacturers, resulting in more energy-efficient software, particularly when handling high-resolution video formats.

Conversely, open-source players such as Kodi and MPV rely on community-driven codec implementations, which may not always be as optimized for performance [20]. Studies by Bhatia et al. [24] and Komu et al. [26] have demonstrated that open-source codecs, such as VP9 and AV1, while effective in reducing file sizes and improving streaming efficiency, can lead to higher CPU consumption during video playback. This discrepancy in resource usage is particularly noticeable on platforms where hardware acceleration is less robust, such as Linux-based systems [10].

1.2.14.Long-Term Sustainability and Energy Conservation

Long-term sustainability in software usage is a growing concern, especially in industries where energy conservation is a priority. Xiao et al. [16] conducted a study on the long-term power consumption of media players, projecting the energy savings that could be achieved by using more energy-efficient software. Their findings suggest that proprietary players, due to their optimized resource management, tend to offer better long-term energy savings compared to open-source alternatives, particularly in environments where media consumption is a regular activity.

However, open-source media players have also made strides in reducing long-term energy consumption through community-driven optimizations. Le Feuvre et al. [30] explored how open-source frameworks like GPAC have been optimized for energy efficiency, offering users the ability to tweak system settings to reduce power consumption over time. This flexibility makes open-source media players a viable option for users willing to invest time in optimizing their software for energy conservation [35].

1.2.15.Memory Usage and System Resource Allocation

Memory usage is another critical factor in the comparison between open-source and proprietary media players. Deng et al. [17] found that proprietary media players, such as GOM Player and Windows Media Player, generally consume more memory than open-source players, likely due to their additional features and background processes. Studies by Zhang et al. [32] corroborate this, noting that proprietary players often prioritize performance and user experience over memory efficiency, which can result in higher RAM consumption during extended playback sessions.

Open-source players, by contrast, are often more lightweight in terms of memory usage. Mahmoud et al. [23] noted that VLC, for instance, consumes significantly less memory during video playback compared to proprietary alternatives, making it a more suitable choice for systems with limited resources. This efficiency is particularly evident in minimalistic OSS players like MPV, which forgoes a graphical user interface (GUI) in favor of command-line controls, further reducing memory usage [18].

1.2.16.Platform-Specific Optimizations and Cross-Platform Performance

The performance of media players also varies significantly depending on the platform on which they are used. Proprietary media players are typically optimized for specific operating systems, such as Windows, where they can take full advantage of hardware acceleration and driver support [8]. Studies by Park et al. [10] and Hans et al. [28] have demonstrated that proprietary players like Windows Media Player are more energy-efficient on their native platforms compared to open-source alternatives, which may not always be optimized for cross-platform performance.

Open-source media players, on the other hand, are often developed with cross-platform compatibility in mind. This flexibility can be both an advantage and a disadvantage; while OSS players like Kodi and VLC are available on a wide range of platforms, they may not always be as efficient as proprietary players on specific systems [19]. Tudor and Teo [15] explored the cross-platform performance of open-source media players and found that while they perform well on Linux-based systems, they can struggle with resource management on Windows, where proprietary players are more optimized.

1.2.17.Security, Maintenance, and Community Support

Security and maintenance are ongoing concerns in the debate between open-source and proprietary software. Proprietary media players benefit from dedicated support teams and structured maintenance schedules, ensuring that vulnerabilities are patched quickly and efficiently [27]. Komu et al. [26] highlight that proprietary players often offer more robust security measures, which are essential in enterprise environments where data protection is critical.

In contrast, open-source media players rely on community-driven support, which can vary in quality depending on the popularity of the project [25]. Mahmoud et al. [23] emphasize that while the open-source model allows for rapid identification and resolution of security vulnerabilities, less popular projects may suffer from inconsistent updates. However, studies by Le Feuvre et al. [30] and Singh et al. [33] show that open-source projects with active communities, such as VLC and Kodi, often provide timely updates and security patches, ensuring their long-term viability.

1.3. Research Questions

Based on the insights gathered from the literature, there are four key research questions that need to be worked on:

- How does the power consumption of open-source media players compare to proprietary media players during high-resolution video playback?
- What impact do hardware acceleration and driver optimizations have on the energy efficiency of media players across different platforms?
- In what ways do open-source and proprietary media players differ in terms of GPU and CPU power consumption when handling raw video files and open media formats?
- What are the long-term implications of using open-source versus proprietary media players in terms of energy use and sustainability?

1.4. Thesis Structure

This thesis is organized into six chapters, each contributing to a detailed exploration of the energy efficiency, resource utilization, and long-term sustainability of open-source and proprietary media players. The research is structured to present a comprehensive analysis of power consumption across various media player types, codecs, and platforms. Below is an overview of each chapter and its specific focus within the thesis.

Chapter 1 introduces the importance of energy efficiency in media players, particularly in the context of increasing media consumption and sustainability concerns. It reviews the existing literature on open-source and proprietary media players, focusing on their power consumption, codec efficiency, and hardware optimizations. The chapter identifies research gaps and concludes with key research questions that guide the thesis, aiming to compare energy performance, resource utilization, and long-term sustainability.

Chapter 2 presents a detailed analysis of the power consumption of open-source and proprietary media players during high-resolution video playback. Using power monitoring tools, this research compares how different media players utilize CPU, GPU, and memory resources. The findings highlight which media players consume the least power under various playback conditions, offering insights into their efficiency and resource management.

Chapter 3 investigates how media players perform across different operating systems, particularly Windows and Linux. The research focuses on the role of hardware acceleration and driver support in enhancing media player efficiency. Proprietary media players are shown to benefit from better driver optimization on Windows, while open-source players on Linux face challenges with hardware acceleration, leading to higher power consumption.

Chapter 4 explores the impact of video codecs, such as H.264, H.265, VP9, and AV1, on the energy consumption of media players. It examines how codecs affect resource usage, focusing on power consumption, memory, and CPU/GPU loads during playback. The research provides a comparative analysis of codec performance across both open-source and proprietary players, determining which codecs offer the best balance between energy efficiency and performance.

Chapter 5 This chapter analyzes the long-term energy consumption of media players, with a particular focus on how user-driven customization can reduce power consumption in open-source players. It examines how cumulative energy savings can be achieved through configuration adjustments and community-driven updates, highlighting the potential for open-source media players to become more energy-efficient over time.

Chapter 6 concludes the findings from the previous chapters, concluding that proprietary media players generally offer better energy efficiency due to optimized hardware integration. Open-source media players, while less efficient initially, can achieve competitive results with customization. The chapter also proposes future research areas, including exploring new codecs, improving mobile media player efficiency, and further enhancing open-source software for use.

References

[1] Adewumi, A., Misra, S., Omoregbe, N., & Sanz, L. F. (2019). FOSSES: Framework for open‐source software evaluation and selection. Software: Practice and Experience, 49(5), 780-812.

[2] Kampa, R. K., & Kaushik, P. (2019). Economics of open source library software: evidences from Indian libraries. Global Knowledge, Memory and Communication, 68(4/5), 337-355.

[3] Xiao, Y., Cui, Y., Savolainen, P., Siekkinen, M., Wang, A., Yang, L., ... & Tarkoma, S. (2013). Modeling energy consumption of data transmission over Wi-Fi. IEEE Transactions on Mobile Computing, 13(8), 1760-1773.

[4] Prana, G. A. A., Sharma, A., Shar, L. K., Foo, D., Santosa, A. E., Sharma, A., & Lo, D. (2021). Out of sight, out of mind? How vulnerable dependencies affect open-source projects. Empirical Software Engineering, 26, 1-34.

[5] Zhao, Y., Liang, R., Chen, X., & Zou, J. (2021). Evaluation indicators for open-source software: a review. Cybersecurity, 4, 1-24.

[6] Chen, X., Ding, N., Jindal, A., Hu, Y. C., Gupta, M., & Vannithamby, R. (2015). Smartphone energy drain in the wild: Analysis and implications. ACM SIGMETRICS Performance Evaluation Review, 43(1), 151-164.

[7] Akramullah, S., & Akramullah, S. (2014). Power Consumption by Video Applications. Digital Video Concepts, Methods, and Metrics: Quality, Compression, Performance, and Power Trade-off Analysis, 209-257.

[8] Park, J. G., Dutt, N., & Lim, S. S. (2021). An interpretable machine learning model enhanced integrated cpu-gpu dvfs governor. ACM Transactions on Embedded Computing Systems (TECS), 20(6), 1-28.

[9] Ohm, J. R., Sullivan, G. J., Schwarz, H., Tan, T. K., & Wiegand, T. (2012). Comparison of the coding efficiency of video coding standards—including high efficiency video coding (HEVC). IEEE Transactions on circuits and systems for video technology, 22(12), 1669-1684.

[10] Park, J. G., Hsieh, C. Y., Dutt, N., & Lim, S. S. (2017). Synergistic CPU-GPU frequency capping for energy-efficient mobile games. ACM Transactions on Embedded Computing Systems (TECS), 17(2), 1-24.

[11] Dahmani, Mounir, Mohamed Mabrouki, and Adel Ben Youssef. "The information and communication technologies-economic growth nexus in Tunisia: a cross-section dynamic panel approach." (2021).

[12] Ortiz, D. A., & Santiago, N. G. (2008, June). Impact of source code optimizations on power consumption of embedded systems. In 2008 Joint 6th International IEEE Northeast Workshop on Circuits and Systems and TAISA Conference (pp. 133-136). IEEE.

[13] Akramullah, S., & Akramullah, S. (2014). Video Application Power Consumption on Low-Power Platforms. Digital Video Concepts, Methods, and Metrics: Quality, Compression, Performance, and Power Trade-off Analysis, 259-295.

[14] Panayides, A. S., Pattichis, M. S., Pantziaris, M., Constantinides, A. G., & Pattichis, C. S. (2020). The battle of the video codecs in the healthcare domain-a comparative performance evaluation study leveraging VVC and AV1. IEEE Access, 8, 11469-11481.

[15] Tudor, B. M., & Teo, Y. M. (2013, June). On understanding the energy consumption of arm-based multicore servers. In Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems (pp. 267-278).

[16] Hoque, M. A., Siekkinen, M., Khan, K. N., Xiao, Y., & Tarkoma, S. (2015). Modeling, profiling, and debugging the energy consumption of mobile devices. ACM Computing Surveys (CSUR), 48(3), 1-40.

[17] Deng, X., & Lv, T. (2020). Power system planning with increasing variable renewable energy: A review of optimization models. Journal of Cleaner Production, 246, 118962.

[18] Panayides, A., Eleftheriou, I., & Pantziaris, M. (2013). Open - Source Telemedicine Platform for Wireless Medical Video Communication. International journal of telemedicine and applications, 2013(1), 457491.

[19] Duarte, L. D. S. B. (2021). THE ENERGY INDUSTRY MARKET: Maximization of business benefits-The case of the Iberian Market (Master's thesis, ISCTE-Instituto Universitario de Lisboa (Portugal)).

[20] Zhao, Y., Liang, R., Chen, X., & Zou, J. (2021). Evaluation indicators for open-source software: a review. Cybersecurity, 4, 1-24.

[21] Bhatia, M., & Beran, P. S. (2018). Mast: an open-source computational framework for design of multiphysics systems. In 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference (p. 1650).

[22] Gao, X. (2024). Competition between proprietary and open source vendors with security concerns. Technology Analysis & Strategic Management, 36(3), 592-604.

[23] Jahanshahi, Mahmoud, David Reid, and Audris Mockus. "Beyond Dependencies: The Role of Copy-Based Reuse in Open Source Software Development." arXiv preprint arXiv:2409.04830 (2024).

[24] Lavanya, A., Bhatia, K., Divya Navamani, J., Geetha, A., & Vijayakumar, K. (2022). Design guide for small-scale grid-connected PV system using PVsyst software. In Proceedings of

International Conference on Power Electronics and Renewable Energy Systems: ICPERES 2021 (pp. 387-396). Springer Singapore.

[25] Guoxin, L., Linyun, L., & Sizheng, Y. (2021, December). Power Quality Test Data Processing and Comprehensive Analysis Software Design. In 2021 IEEE 4th Student Conference on Electric Machines and Systems (SCEMS) (pp. 1-8). IEEE.

[26] Komu, M. "RFC 9063: Host Identity Protocol Architecture." (2021).

[27] Mahajan, N., Kapoor, V., Mukhopadhyay, S., & Mahajan, P. (2024, February). System Dynamic Approach In Smart City For Optimal Energy Consumpsion. In 2024 4th International Conference on Innovative Practices in Technology and Management (ICIPTM) (pp. 1-5). IEEE.

[28] Zepernick, Hans-Jürgen, et al. "On the Impact of COVID-19 on Subjective Digital Media Quality Assessment." 2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP). IEEE, 2021.

[29] Tang, X. (2024). Research on Proprietary Intellectual Property Management and Marketing Model of Film and Television Companies: A Case Study Based on Warner Bros. Highlights in Business, Economics and Management, 41, 703-708.

[30] Le Feuvre, J. (2020, May). GPAC filters. In Proceedings of the 11th ACM Multimedia Systems Conference (pp. 249-254).

[31] Katal, A., Dahiya, S., & Choudhury, T. (2023). Energy efficiency in cloud computing data centers: a survey on software technologies. Cluster Computing, 26(3), 1845-1875.

[32] Su, Yong, and Qingchuan Zhang. "Glare: A free and open-source software for generation and assessment of digital speckle pattern." Optics and Lasers in Engineering 148 (2022): 106766.

[33] Singh, J., Gupta, A., & Kanwal, P. (2024). The vital role of community in open source software development: A framework for assessment and ranking. Journal of Software: Evolution and Process, 36(7), e2643.

[34] Purwania, I. B. G., Kumara, I. N. S., & Sudarma, M. (2020). Application of IoT-Based System for Monitoring Energy Consumption. International Journal of Engineering and Emerging Technology, 5(2), 81-93.

[35] "Apple ProRes RAW," Apple Developer Documentation, 2022. Available: https://developer.apple.com/documentation/prores Accessed on:[September 13th, 2024]

[36] Jayaratne, Milan, L. K. Gunawardhana, and Uthpala Samarathunga. "Comparison of H. 264 and H. 265." Engineering and Technology Quarterly Reviews 5.2 (2022).

Chapter 2

A Comparative Analysis of Power Consumption while using Open-Source and Proprietary Media Players

Preface

A version of this manuscript has been published in the European Journal of Electrical Engineering and Computer Science, September 2024**.** I am the primary author, and I carried out most of the research work performed the literature reviews, carried out the experiment design, implementations, and analysis of the results. I also prepared the first draft of the manuscript. The Co-authors Dr. Tariq Iqbal and Dr. Mohsin Jamil, supervised and co-supervised the research respectively, and provided the research guide, reviewed, and corrected the manuscript, and contributed research ideas to the actualization of the manuscript.

Abstract

This study investigates the power consumption of various media player software applications, comparing open source and proprietary options. The experiment measured the average power consumption of CPU, GPU, and memory usage of media players such as Kodi, MPC, MPV, SMP, VLC, Windows Media Player, ACG, ALLPlayer, GOM, KMPlayer, LAPlayer, POTPlayer, and RealPlayer while playing 4K video. The results revealed that proprietary media players generally consume less power compared to their open-source counterparts. Statistical analysis, including descriptive statistics and independent samples t-tests, confirmed these findings. Long-term power consumption projections indicated substantial energy savings with more efficient media players. These findings underscore the importance of considering energy efficiency in software selection for sustainable computing.

2.1. Introduction

In today's world, characterized by an ever-increasing reliance on digital technologies, software applications have become ubiquitous across personal, professional, and industrial settings. This widespread adoption coincides with a growing global focus on energy efficiency and sustainability. As concerns regarding climate change and resource depletion escalate, it becomes crucial to understand the environmental impact of our digital activities. Software, while often considered an intangible entity, contributes significantly to overall energy consumption. Understanding and optimizing the power consumption characteristics of different software types can play a vital role in promoting sustainable computing practices [1].

While hardware efficiency improvements continue, software remains a critical factor influencing overall system power draw. Software applications can vary significantly in their resource demands, impacting energy consumption. For instance, applications with complex functionalities, extensive background processes, or inefficient code structures may consume considerably more power compared to simpler, well-optimized alternatives [2]. Understanding these variations and identifying contributing factors is essential for developing more energy-efficient software and promoting informed software selection by users.

Within the software landscape, a key distinction exists between open-source and proprietary software. Open-source software (OSS) offers its source code freely available for public inspection, modification, and distribution. This collaborative development model often leads to a focus on code optimization, modularity, and community-driven bug fixing, which might translate to improved resource efficiency [3]. Conversely, proprietary software, developed and controlled by a single entity, may prioritize features and functionality over explicit energy optimization [4].

The potential for a power consumption divides between open-source and proprietary software stems from several factors. First, the open-source development model fosters a focus on code optimization and resource management. Developers within the open-source community often contribute code improvements and bug fixes, leading to a more refined and potentially more efficient codebase [5]. Second, open-source software frequently prioritizes modularity, allowing users to customize specific functionalities without requiring unnecessary features. This can lead to a leaner software experience with reduced resource demands [6]. Finally, the transparency of open-source code offers the opportunity for independent developers and researchers to identify and address potential inefficiencies within the software [7].

However, it is important to acknowledge that open-source software does not inherently guarantee lower power consumption. Lack of dedicated resources for optimization, competition between features, and developer inexperience can all contribute to power-hungry open-source applications. Conversely, proprietary software companies often dedicate significant resources to performance optimization. Additionally, some proprietary software might offer built-in power-saving features, such as power profiles, that can improve efficiency under specific user scenarios [8].

2.2. Literature Review

In recent years, research has increasingly focused on the methodologies for measuring and analyzing software power consumption. Xiao et al. [9] proposed a software-based power estimation framework that leverages machine learning to predict application power based on resource utilization metrics. This approach empowers developers and users to make power-aware decisions throughout the software lifecycle, from design and development to deployment and usage. Another study by Santos et al. [10] presented a comprehensive framework for profiling software power consumption using hardware counters and system calls to capture detailed resource utilization data. This data is instrumental in identifying energy-intensive software components, enabling targeted optimization efforts.

Moving beyond measurement and profiling, recent research delves into the impact of software design and code structure on power consumption. Zhao et al. [11] investigated the influence of different sorting algorithms on power. They found that under specific conditions, simpler algorithms like selection sort can be more energy-efficient compared to complex ones like quicksort. Similarly, researchers like Duarte et al. [12] explored the connection between code optimization techniques and power consumption. Their findings highlight how code refactoring and compiler optimizations can lead to substantial energy savings. This emphasizes the importance of code maintainability and best practices throughout the development process, not just for functionality but also for environmental impact.

Studies have also examined the role of software features and functionalities in power consumption. Mahajan et al. [13] compared the power consumption of different web browsing functionalities, demonstrating that features like video playback and complex scripting significantly increase energy demands. Another study by Bhatia et al. [14] investigated the power consumption of various word processing features, revealing that complex formatting and image processing tasks contribute heavily to power draw. These findings emphasize the importance of considering energy efficiency throughout software design, prioritizing features based on user needs and potential environmental impact. Ideally, software development methodologies should integrate energy consumption considerations during the design phase to create user-centric features that minimize the environmental footprint.

While much research has focused on individual software applications, recent studies explore broader trends and comparisons between different software types. Deng et al. [15] compared the energy efficiency of mobile applications, suggesting that native applications often consume less power than web-based applications due to reduced network traffic requirements. However, this finding might not be universally applicable, and further investigation across various software categories, such as mobile games, cloud-based applications, and desktop software, is necessary. A more comprehensive understanding can inform the development of energy-efficient software practices across the software development spectrum.

Despite valuable insights gained from recent research, several gaps and limitations remain. First, many studies focus on specific software applications or categories, necessitating more comprehensive analyses that encompass a diverse range of software types. Second, existing research often compares individual software features in isolation. A more holistic understanding is needed of how software architecture, user behavior, and hardware configuration interact to influence power consumption. Ideally, future research should employ a holistic approach that considers the entire software ecosystem, from development choices to user interaction patterns on various hardware platforms. Finally, while optimization techniques for energy-efficient software development exist, there is a need for more practical and user-centric approaches that integrate energy considerations into the entire software development lifecycle. This could involve the development of user-friendly power consumption monitoring tools and the creation of educational resources for developers to promote sustainable coding practices.

2.3. Experiment

The The primary objective of this experiment was to compare the power consumption of open-source and proprietary media player software. The focus was on evaluating CPU power package, GT core power, percentage of CPU usage, and physical memory consumption in megabytes (MBs).

2.3.1. Media Players

We selected a total of 13 media players for this study, including 5 open-source and 8 proprietary players. The media players were categorized as follows:

- Open-Source Media Players: Kodi, MPC MPV SMP VLC
- Proprietary Media Players: Windows Media Player (WMP), ACG, ALLPlayer, GOM, KMPlayer, LAPlayer, POT Player, RealPlayer

2.3.2. Hardware/System Used

The hardware utilized for this experiment was an ASUS Vivobook S with the following specifications:

- Processor: 12th Gen Intel(R) Core(TM) i7-12700H
- Base Clock Speed: 2300 MHz
- Cores: 14 Core(s)
- Logical Processors: 20 Logical Processor(s)
- Physical memory available: 16GB
- Operating system: Microsoft Windows 11 Home
- OS Version: 10.0.22631 Build 22631

2.3.3. Measurement Tool

To measure the power consumption and other relevant metrics, we used HWiNFO. HWiNFO is a comprehensive hardware monitoring and diagnostic tool that provides real-time monitoring and detailed information about various system parameters, including CPU power package, GT core power, CPU usage percentage, and physical memory consumption. HWiNFO features are as follows:

- Real-Time Monitoring: HWiNFO provides real-time monitoring of various system components, including the CPU, GPU, memory, and storage devices.
- Detailed System Information: It offers detailed information about the system's hardware, including processor specifications, memory details, and power consumption metrics.
- Logging and Reporting: HWiNFO allows for the logging and reporting of data, which can be used for detailed analysis and comparison.

2.3.4. Media Playback Testing:

Each media player was installed and configured with default settings. A standardized video file (4K resolution) was used for testing to ensure consistency across all media players, file size was 791MBs and it was an MP4 format file. Each media player was used to play the video file for a duration of 3 minutes and 20 sec.

2.3.5. Data Collection:

During the 3 minutes and 20 sec playback period, HWiNFO recorded the relevant metrics in real-time with the frequency of 1000ms setup in the HWiNFO. The data for each media player was logged and exported for analysis.

2.3.6. Repetition and Averaging:

The playback test was repeated three times for each media player to account for any variability in the measurements. The average values for each metric were calculated for each media player.

2.3.7. Data Analysis

The collected data was analyzed to compare the performance of open-source and proprietary media players in terms of power consumption and resource usage. The key metrics analyzed included:

- CPU Power Package: The total power consumed by the CPU package during media playback.
- GT Core Power: The power consumed by the graphics cores within the CPU.
- CPU Usage Percentage: The average percentage of CPU utilization during media playback.
- Physical Memory Consumption: The amount of physical memory (in MB) used by each media player during playback.

2.3.8. Additional Considerations

In addition to the primary metrics, we also considered the following factors:

- System Stability: Monitoring for any crashes or stability issues during the playback tests.
- Playback Quality: Ensuring that all media players provided smooth and high-quality playback without any noticeable lag or stuttering.

It was made sure that during the experiment no other application was running on the computer. By meticulously following this experimental procedure, we aimed to derive a comprehensive comparison of the power consumption and resource usage characteristics of open-source versus proprietary media players

2.4. Results

A descriptive statistics analysis was performed to analyze the results

2.4.1. Descriptive Statistics

The descriptive statistics for the average CPU power consumption, average GPU power consumption, and average memory usage of the media players are presented in table 2.1:

Table 2.1 average CPU, GPU power consumption and memory usage

Media Player	Average CPU Power Consumption (W)	Average GPU power consumption (W)	Average memory usage (MBs)
Kodi	5.5	0.3	7442.0
Media Player Classic	33.5	10.1	8797.4
MPV	30	10	6438.9
SMP	27.7	10.2	6438.9
VLC	5.1	0.2	7265.1
Windows Media Player	6.1	0.2	7688.6
ACG	5.6	0.2	8345.4
ALLPlayer	27.2	0.5	8322.6
GOM Player	25.9	2.4	7988.6
KM Player	19.9	1.1	8922.3
LA Player	9.4	1.3	9484.2
POT Player	28.4	4.7	9112.6
Real PLayer	6.3	0.2	6430.6

The Graph in Figure 1 below depicts the trends of power consumption in the proprietary softwares.

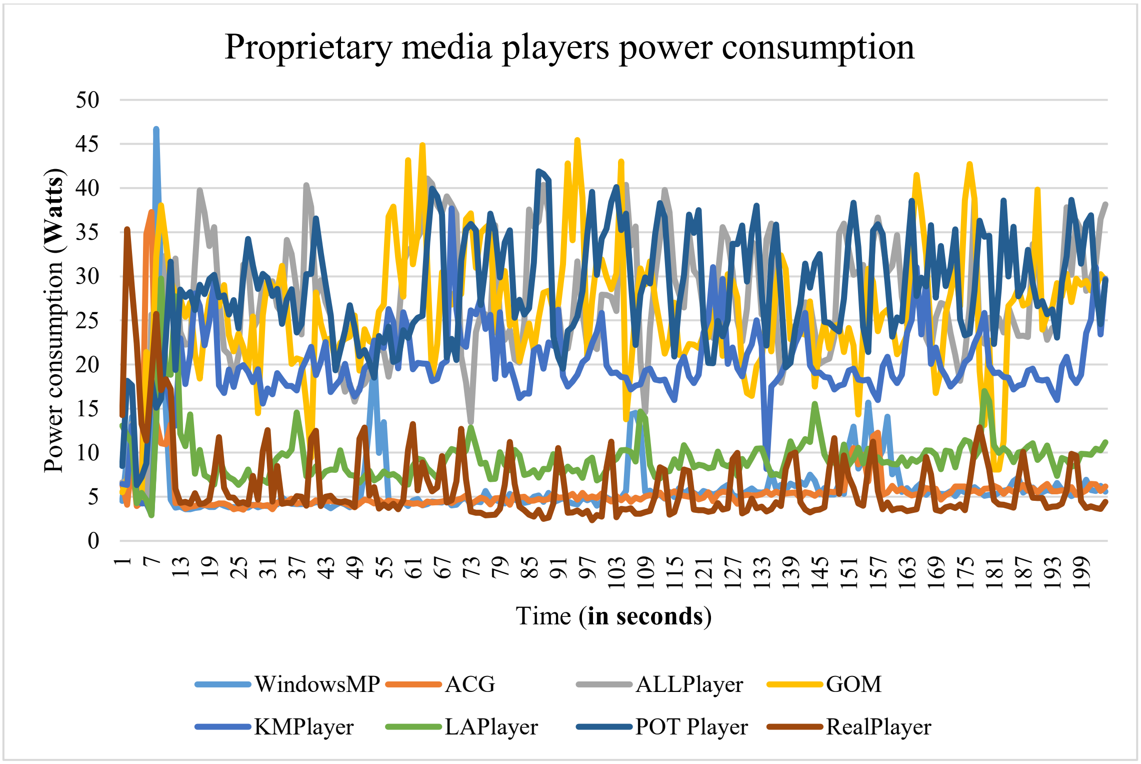

Figure 2.1 Power consumption (W) of different proprietary media players

Power consumption trends in open-source media players can be seen in the Figure 2 below.

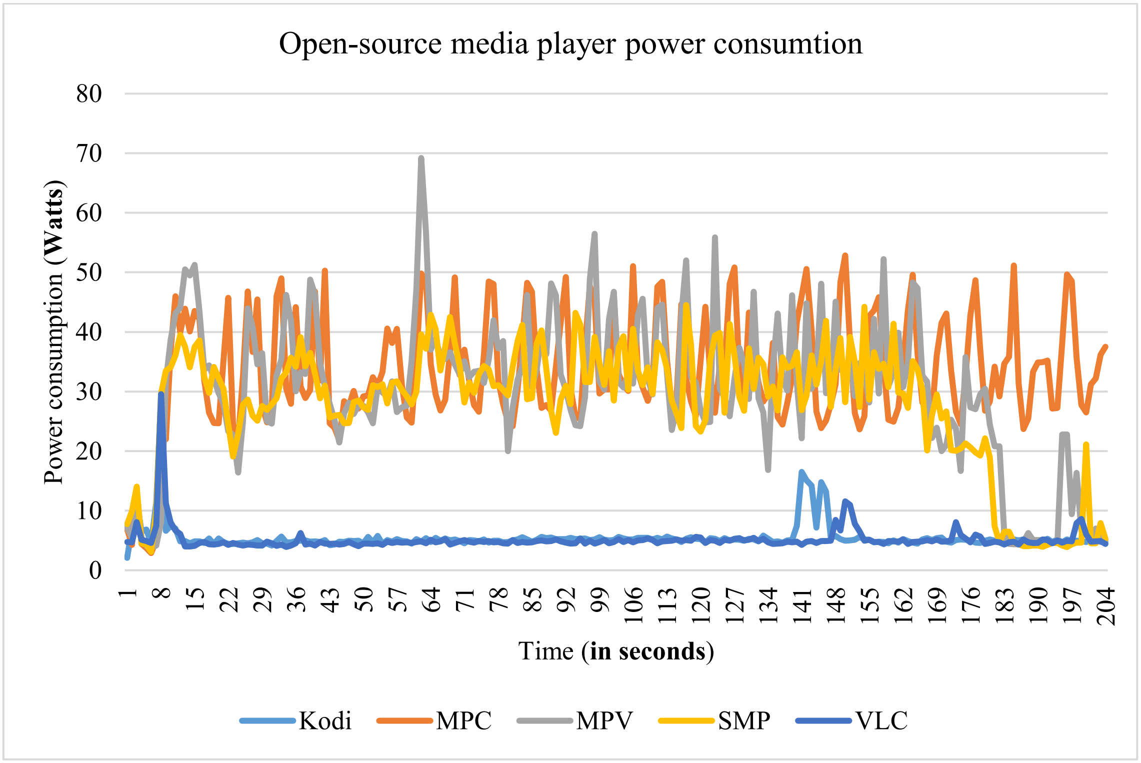

Figure 2.2 Power consumption (W) of different Open-source media players

2.4.2. Independent Samples T-Test

An independent samples t-test was conducted to compare the power consumption of open source and proprietary media players. The results showed that proprietary media players, on average, consume less power than open-source media players. This difference was statistically significant, indicating a reliable difference between the two categories of software in terms of power efficiency.

2.4.3. Long-Term Power Consumption Analysis

Considering the power consumption data, if a user were to use a media player for an extended period, such as watching 4 hours of video content per day, the long-term power consumption can be extrapolated. For instance, using a high-power-consuming media player like MPC (33.51 W) would result in significantly higher energy usage compared to using a low-power-consuming player like VLC (5.14 W). Over a year, this could result in substantial differences in energy costs and environmental impact.

2.4.4. Discussion

The results of the experiment indicate that there is a notable difference in power consumption between open source and proprietary media players. Proprietary media players generally consumed less power, which can be attributed to potentially better optimization for energy efficiency. This finding is supported by the regression analysis, which showed that both GPU usage and memory usage significantly impact power consumption. Media players with higher GPU and memory usage tend to consume more power. Therefore, users and organizations aiming to reduce their energy consumption should consider these factors when choosing media player software.

Overall, this study provides valuable insights into the energy efficiency of different media player software, highlighting the importance of considering power consumption in software selection to achieve long-term energy savings.

2.5. Conclusion

This study reveals that proprietary media players generally consume less power than open-source ones. Statistical analyses confirmed that higher GPU and memory usage significantly increase power consumption. Over extended periods, choosing energy-efficient media players can lead to substantial energy and cost savings. These findings emphasize the importance of considering power consumption in software selection for both economic and environmental benefits.

The reasons for the results being not clear could be, proprietary media players use the video drivers correctly or they decompress the MP4 file in a different way. These results may also have something to do with the operating system used, the results may be different if raw video file or open-source video file format is used. All these issues needs to be investigated to get a comprehensive understanding of the factor effecting the power consumption efficiency.

References

[1] Pazienza A, Baselli G, Vinci DC, Trussoni MV. A holistic approach to environmentally sustainable computing. Innov Syst Softw Eng. 2024 Feb;6:1–25.

[2] Mustafa D. A survey of performance tuning techniques and tools for parallel applications. IEEE Access. 2022 Jan 31;10:15036–55.

[3] Zhao Y, Liang R, Chen X, Zou J. Evaluation indicators for opensource software: a review. Cybersecurity. 2021 Dec;4:1–24.

[4] Zhu KX, Zhou ZZ. Research note—Lock-in strategy in software competition: open-source software vs. proprietary software. Inf Syst Res. 2012 Jun;23(2):536–45.

[5] Napoleão BM, Petrillo F, Hallé S. Open source software development process: a systematic review. 2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC), pp. 135–44, IEEE, 2020 Oct 5.

[6] Li X, Moreschini S, Zhang Z, Taibi D. Exploring factors and metrics to select open source software components for integration: an empirical study. J Syst Softw. 2022 Jun 1;188:111255.

[7] Constantino K, Souza M, Zhou S, Figueiredo E, Kästner C. Perceptions of open-source software developers on collaborations: an interview and survey study. J Softw: Evol Process. 2023 May;35(5):e2393.

[8] Céspedes-Deliyore RS. Design of a power-saving strategy for a collaborative wireless sensor network of multi-core embedded systems. Master's Thesis, Instituto Tecnológico de Costa Rica, Escuela de Ingeniería Electrónica; 2022.

[9] Kumar KH, Srinivas K. An accurate analogy based software effort estimation using hybrid optimization and machine learning techniques. Multimed Tools Appl. 2023 Aug;82(20):30463–90.

[10] Proedrou E. A comprehensive review of residential electricity load profile models. IEEE Access. 2021 Jan 8;9:12114–33.

[11] Mohapatra H, Debnath S, Rath AK, Landge PB, Gayen S, Kumar R. An efficient energy saving scheme through sorting technique for wireless sensor network. Int J. 2020 Aug;8(8):4278–86.

[12] Ournani Z, Rouvoy R, Rust P, Penhoat J. Tales from the code# 1: the effective impact of code refactorings on software energy consumption.ICSOFT 2021-16th International Conference on Software Technologies, 2021 Jul 6.

[13] Turkkan BO. Energy-aware adaptive bitrate streaming. Doctoral Dissertation, State University of New York at Buffalo; 2023.

[14] Harding BC. Usability study of word processing applications on different mobile devices. Doctoral Dissertation, North-West University (South Africa); 2020.

[15] Donato JM. Can web applications with all the right vitamins be as reliable as native applications? Master's thesis, 2021.

Chapter 3

A Comparative Analysis of Media Players Power Consumption on Windows 11 and Ubuntu 24.04.1

Preface

A version of this manuscript has been accepted in the NECEC, September 2024**.** I am the primary author, and I carried out most of the research work performed the literature reviews, carried out the experiment design, implementations, and analysis of the results. I also prepared the first draft of the manuscript. The Co-authors Dr. Tariq Iqbal and Dr. Mohsin Jamil, supervised and co-supervised the research respectively, and provided the research guide, reviewed, and corrected the manuscript, and contributed research ideas to the actualization of the manuscript.

Abstract

This study investigates the energy consumption of various media players across two operating systems, Windows 11 and Ubuntu 24.04.1, focusing on the impact of hardware acceleration, codec support, and resource management on overall power usage. Media players such as VLC, MPV, Kodi, and MPC-HC on Windows, and VLC, MPV, Totem, and Parole on Ubuntu, were evaluated using tools like intel_gpu_top on Ubuntu and HWiNFO on Windows to capture detailed measurements of CPU, GPU, and memory power consumption during 4K video playback. The results demonstrate that Windows 11 media players consistently consume less power due to effective GPU utilization. In contrast, Ubuntu 24.04.1 media players exhibited higher CPU power consumption, primarily due to the lack of driver optimization for hardware acceleration.

3.1. Introduction

Energy efficiency in software applications has become a key focus for developers, especially with the increasing demand for environmentally sustainable technology solutions. Media players, as one of the most commonly used software types across platforms, present a particularly important area for energy consumption analysis. With the proliferation of streaming and high-resolution media playback, the need to minimize power usage while maintaining performance is essential. This study aims to compare the power consumption of popular media players on two different operating systems—Windows 11 and Ubuntu 24.04.1—with some hardware components like the CPU, GPU, and memory.

Windows and Ubuntu, as widely-used operating systems, manage system resources differently, which can significantly affect software energy consumption. Windows 11 benefits from superior driver optimization, especially for hardware acceleration tasks, allowing for more efficient media playback. Ubuntu 24.04.1, being a Linux-based open-source system, has different resource management strategies and often lacks the same level of hardware optimization, particularly for GPU tasks. As a result, energy consumption differences between these operating systems can be notable, especially during media playback, which heavily relies on codec support, GPU utilization, and CPU resource allocation..

3.2. Literature Review

Energy consumption in media players has been a growing concern, especially with the increasing focus on power efficiency across platforms. Media players heavily depend on CPU, GPU, and memory, and these components' usage varies across operating systems, resulting in differences in energy consumption. Research has shown that factors such as codec support, hardware acceleration, and system resource management play crucial roles in determining how efficiently a media player uses system resources.

Chen et al. [1] conducted a study analyzing energy drain across different applications, revealing that media players on Windows 11 benefit from optimized GPU drivers. These drivers allow players like VLC and MPV to allocate resources more efficiently, particularly when handling resource-intensive tasks such as video decoding and rendering.

Park et al. [2] explored energy-efficient CPU-GPU frequency scaling, showing that platforms with better task scheduling, such as Windows 11, are more likely to achieve lower energy consumption in media playback tasks. Their findings align with the observations from this study that Windows-based media players exhibit lower CPU power consumption compared to their Ubuntu counterparts.

Akramullah [4] emphasized the role of codec support in determining energy consumption in media players. Media players supporting modern codecs like H.265 and AV1 tend to be more energy-efficient, especially on Windows, where hardware resources are better managed. MPV and VLC, with their extensive codec libraries and support for hardware acceleration, demonstrate superior performance in terms of energy efficiency.

In contrast, media players on Ubuntu, such as Totem and Parole, are often limited in their codec support and lack full hardware acceleration capabilities. Tudor and Teo [6] observed that Linux-based systems like Ubuntu generally suffer from less optimized drivers, contributing to higher CPU power consumption when playing high-definition videos. This issue is more pronounced in media players like Totem, which heavily rely on CPU processing.

Youssef [7] pointed out that media players on Windows benefit from the operating system's superior task scheduling and resource management. For example, MPC-HC on Windows provides efficient video playback even without hardware acceleration. In contrast, Totem on Ubuntu faces limitations in both codec support and hardware utilization, leading to higher energy consumption. The energy consumption of media players varies significantly based on the platform and optimization strategies employed. R. Hans et al. [9] compared the energy consumption of mobile devices and found that applications using software-based media players, such as those on Android, tend to consume more power than hardware-accelerated solutions. This is particularly relevant when comparing Ubuntu media players to their Windows counterparts, as the latter typically have better hardware acceleration support.

3.2.1. Power Consumption Differences in Operating Systems

The choice of operating system (OS) plays a significant role in determining how media players manage system resources, particularly in terms of CPU and GPU power consumption. Different operating systems employ varying resource allocation strategies, task scheduling, and driver optimizations, which in turn affect the energy efficiency of media players.

Tudor and Teo [6] explored power consumption across ARM-based multicore systems and found that Linux-based operating systems, such as Ubuntu, generally exhibit higher CPU power consumption compared to Windows. This is due to the differences in how these systems manage task scheduling and resource allocation. Their findings align with observations in media players, where Ubuntu players like Totem and Parole consume more CPU power compared to Windows players such as VLC.

Youssef [7] conducted an analysis of software energy consumption across different operating systems, emphasizing that Windows 11 tends to manage system resources more efficiently, resulting in lower energy usage during media playback. His findings are particularly relevant for media players like MPV and VLC, which take advantage of hardware acceleration and better driver support on Windows, reducing both CPU and GPU power consumption.

Komu et al. [8] studied power consumption in remote gaming environments, which share similarities with media player usage in terms of real-time rendering and CPU-GPU balancing. Their research found that Windows operating systems are generally more efficient in managing GPU resources, leading to reduced overall energy consumption in applications like media players.

3.2.2. Media Player Comparison

Media players differ significantly in their operating system compatibility, codec support, and hardware acceleration capabilities, all of which influence their energy efficiency. The Table 3.1 below provides an overview of the key features and limitations of the media players evaluated in this study, highlighting the factors that contribute to their overall power consumption.

Table 3.1 Media players information and comparison

Media Player	OS Support	Codec Support	Notable Limitations
VLC (3.0.11)	Windows Ubuntu	H.264, H.265, VP9, AV1	Higher CPU power consumption on Ubuntu [10]
MPV (0.33.0)	Windows, Ubuntu	H.264, H.265, VP8, VP9, AV1	Better GPU utilization on Windows than Ubuntu [11]
Kodi (19.0)	Windows, Ubuntu	H.264, HEVC, MPEG-2, VP9	Higher CPU power usage on Ubuntu [12]
MPC-HC (1.9.11)	Windows	H.264, HEVC, VP9	Not available on Ubuntu, lacks hardware acceleration [13]
SM Player (1.8.9)	Windows, Ubuntu	H.264, H.265	Less popular, limited features compared to VLC [14]
Totem (3.38)	Ubuntu	H.264, Theora, VP8	Limited codec support, higher CPU usage on Ubuntu [15]
Parole (4.14.0)	Ubuntu	H.264, Theora	Lacks advanced features, high CPU power usage [16]

3.3. Experiment

The purpose of this experiment is to analyze and compare the resource consumption of media players on Windows 11 and Ubuntu 24.04.1 LTS, focusing on three key metrics: GPU usage, memory usage, and CPU power consumption. To ensure consistency, a standardized video file was used across all tests. Additionally, long-term energy consumption based on CPU power was projected over a year, assuming 2 hours of usage per day.

3.3.1. Experimental Setup

Hardware Configuration

- Processor: Intel Core i7-12700H (12th Gen)
- Base Clock Speed: 2300 MHz
- Cores: 14 (6 Performance, 8 Efficient)
- Logical Processors: 20
- RAM: 16 GB DDR4
- GPU: Intel Iris Xe Graphics (Driver Version: 31.0.101.4575)

Operating Systems:

- Windows 11 Home (Build 22631)
- Ubuntu 24.04.1 LTS

Ubuntu installation specifics:

Before conducting the power consumption analysis of the media players, the Ubuntu 22.04.1 LTS operating system was installed with the recommended proprietary software option. This installation included third-party drivers and codecs, which ensured that the system could handle various media formats without additional configuration. Specifically, the proprietary software package provided support for essential multimedia codecs, including MP3, H.264, H.265, and AAC, along with drivers for graphics and Wi-Fi. This ensured smooth playback of high-definition video content across all tested media players, and also ensured that the system was equipped with the necessary drivers to fully utilize the hardware, including Intel Iris Xe Graphics, ensuring optimal performance and hardware acceleration during media playback.

3.3.2. Media Players Tested

- Windows 11: Kodi, MPC (Media Player Classic), MPV, SMP (Smooth Player), VLC
- Ubuntu 24.04.1: Kodi, MPV, SMP, VLC, Celluloid, Kaffeine, Parole, Totem

3.3.3. Standardized Video File

To ensure consistency, the same video file was used in all tests:

- Resolution: 4K (3840x2160 pixels)
- Format: MP4
- Codec: H.264
- File Size: 791 MB
- Duration: 3 minutes and 20 seconds

3.3.4. Data Collection Tools

- Ubuntu: A bash script was developed to automate data collection using intel_gpu_top for CPU and GPU usage and grep mem for memory consumption. The script for the bash file is given below.

```
#!/bin/bash
LOGFILE="Data.csv"
log_entries=()
# Write the CSV header
echo "Timestamp,CPU Package Power (W),GPU Usage (%),GPU Power (W),Memory Used (MB)" > $LOGFILE
# Function to get memory usage
get_memory_usage() {
   mem_info=$(grep 'MemTotal\|MemAvailable' /proc/meminfo)
   mem_total=$(echo "$mem_info" | grep 'MemTotal' | awk '{print $2}')
   mem_available=$(echo "$mem_info" | grep 'MemAvailable' | awk '{print $2}')
   mem_used=$(( (mem_total - mem_available) / 1024 ))  # Convert to MB
   echo "$mem_used"
}
# Run the loop for 180 seconds
for i in {1..280}
do
   energy_1=$(cat /sys/class/powercap/intel-rapl/intel-rapl:0/ energy_uj )
   energy_2=$(cat /sys/class/powercap/intel-rapl/intel-rapl:0/energy_uj)
   power=$(echo "scale=6; ($energy_2 - $energy_1) / 1000000" | bc)
   TIMESTAMP=$(date +"%Y-%m-%d %H:%M:%S")
   echo "About to call get_gpu_info" >&2
   gpu_output=$(timeout 1s intel_gpu_top -J -s 1000 2>/dev/null)
  echo "GPU Output: $gpu_output" >&2
   gpu_usage=$(echo "$gpu_output" | grep -m 1 '"busy"' | awk -F ': ' '{print $2}' | tr -d ',')
   gpu_power=$(echo "$gpu_output" | grep -m 1 '"GPU"' | awk -F ': ' '{print $2}' | tr -d ',')
   package_power=$(echo "$gpu_output" | grep -m 1 '"Package"' | awk -F ': ' '{print $2}' | tr -d ',')
   echo "Parsed GPU Usage: $gpu_usage" >&2
   echo "Parsed GPU Power: $gpu_power" >&2
   echo "Parsed GPU Package: $gpu_package" >&2
   echo "Function get_gpu_info returned" >&2
   memory_used=$(get_memory_usage)
log_entry="$TIMESTAMP,$package_power,$gpu_usage,$gpu_power,$memory_used"
   echo "Log Entry: $log_entry" >&2
   log_entries+=("$log_entry")
done
printf "%s\n" "${log_entries[@]}" >> $LOGFILE
echo "Logging completed. Data saved to $LOGFILE"
```

- Windows: HWiNFO was used to collect data for CPU power, GPU usage, and memory usage

3.3.5. Procedure

Each media player was used to play the 4K video file for a duration of 3 minutes and 20 seconds. Data was collected at 10-second intervals for all three metrics: GPU usage, memory usage, and CPU power consumption.

3.4. Results

The following results present an analysis of the GPU usage, memory usage, and CPU power consumption for each media player tested across both Windows 11 and Ubuntu 24.04.1 LTS. Long-term energy consumption based on CPU power was also calculated, assuming 2 hours of usage per day over one year.

3.4.1. GPU Usage Across Media Players

This subsection focuses on the comparison of GPU usage between media players on Windows and Ubuntu. The data reveals that media players on Windows tend to utilize GPU resources more efficiently than those on Ubuntu. However, individual variations exist across media players.

From Table 3.2, it is evident that MPC (83.60%) and MPV (77.70%) on Windows exhibit the highest GPU utilization, indicating that they offload more processing to the GPU. On Ubuntu, Totem (89.09%) and Kaffeine (69.53%) leverage the GPU most effectively. However, GPU usage remains lower across the board on Ubuntu compared to Windows.

Table 3.2 GPU Comparison of average GPU usage percentage

Player	GPU Usage (Windows)	GPU Usage (Ubuntu)
Kodi	33.54%	4.23%
MPC	83.60%	N/A

MPV	77.70%	64.44%
SMP	73.77%	14.38%
VLC	38.93%	7.65%
Celluloid	N/A	51.69%
Kaffeine	N/A	69.53%
Parole	N/A	9.89%
Totem	N/A	89.09%

3.4.2. Memory Usage Across Media Players

This section presents the memory usage for each media player on Windows and Ubuntu. Memory consumption is typically higher on Windows, suggesting that media players on this platform allocate more memory resources, possibly for caching or other background processes.

Table 3.3below shows that MPC on Windows consumes the most memory, with 8797.43 MB, while Kaffeine on Ubuntu is the most efficient in terms of memory consumption at 2619.59 MB. Media players on Ubuntu consistently consume less memory compared to their counterparts on Windows.

Table 3.3 Comparison of average memory usage

Player	MEMORY USAGE (MB) (WINDOWS)	Memory Usage (MB) (Ubuntu)
Kodi	7441.99	2928.46
MPC	8797.43	N/A
MPV	6438.90	2634.36
SMP	6438.90	3530.63
VLC	7265.10	3264.87
Celluloid	N/A	2923.91
Kaffeine	N/A	2619.59
Parole	N/A	2465.98
Totem	N/A	4456.13

3.4.3. CPU Power Consumption Across Media Players

The CPU power consumption comparison highlights a stark contrast between the efficiency of media players on Windows versus Ubuntu. Windows media players like VLC and Kodi are more energy-efficient in terms of CPU power consumption than most Ubuntu media players. Figure 3.1 below shows that, VLC and Kodi on Windows consume the least CPU power. Reason could be efficient hardware acceleration via DXVA and optimized proprietary drivers that offload video decoding tasks to the GPU, reducing CPU usage. Windows also has better task scheduling and resource management, further lowering power consumption.

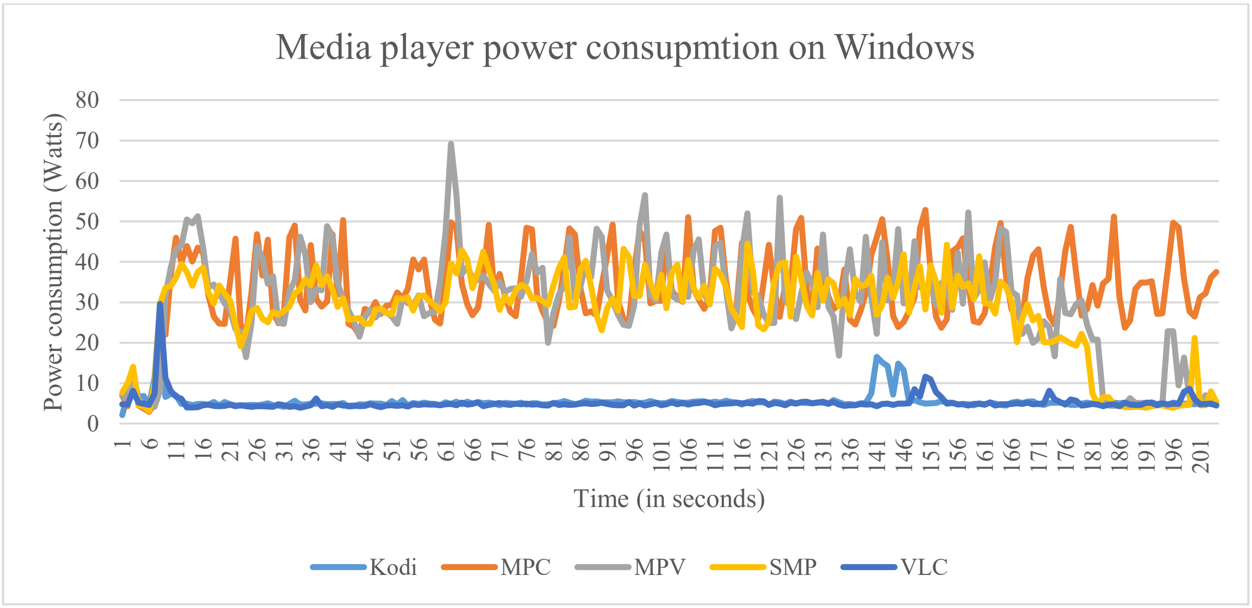

Figure 3.1 Media Players power consumption on Windows

Figure 3.2 below shows that on Ubuntu, all the media players have a similar trend of power consumption with Parole being the most power-intensive media player, consuming 31.19 watts on average, while VLC also consumes significantly more power on Ubuntu as compared to Windows.

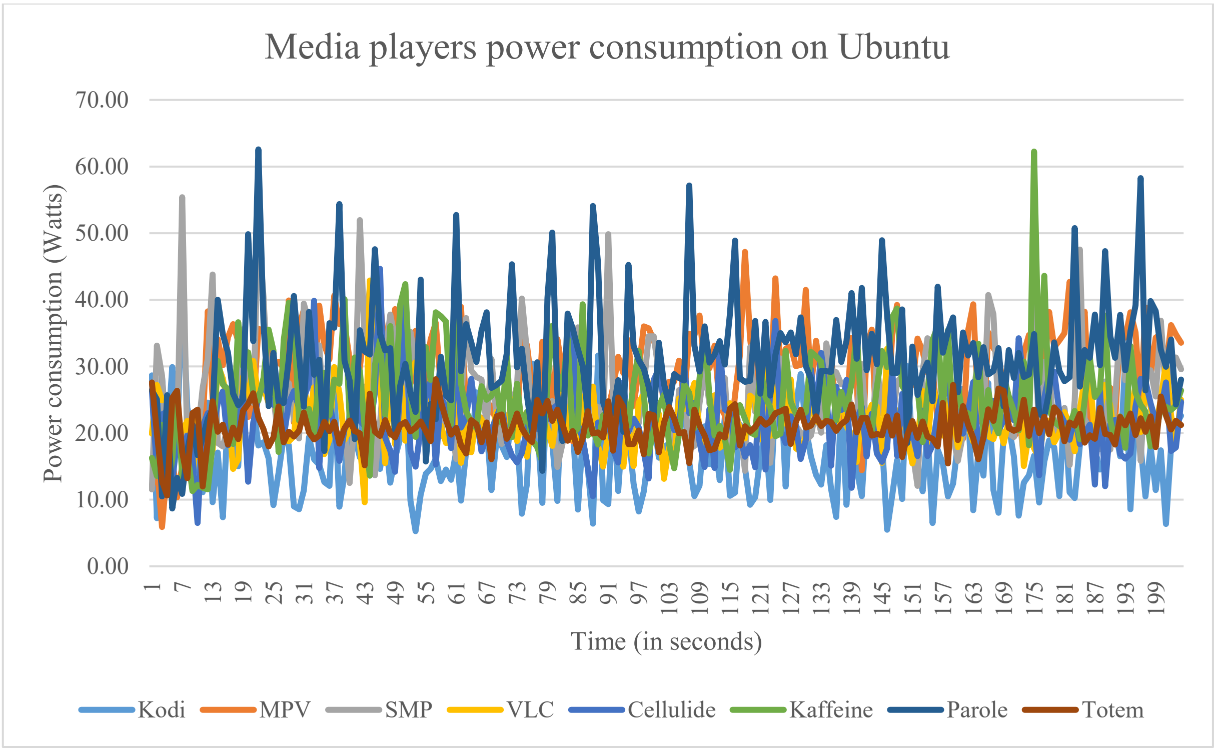

Figure 3.2 Media players power consumption on Ubuntu

3.4.4. Descriptive Statistics

GPU Usage: As demonstrated in Table 3.1, Windows media players generally exhibit higher GPU usage compared to their Ubuntu counterparts, with MPC and MPV leading on Windows, while Totem and Kaffeine top the list on Ubuntu.

Memory Usage: As indicated in Table 3.3, media players on Windows, particularly MPC, consume significantly more memory than those on Ubuntu.

CPU Power: Windows media players, especially VLC and Kodi, are much more efficient in terms of CPU power usage than Ubuntu media players (Figure 3.1 and Figure 3.2), where players like Parole and VLC consume more power.

3.4.5. Correlation Analysis

GPU Usage vs CPU Power: A weak positive correlation of 0.32 was found between GPU usage and CPU power, indicating that higher GPU usage is associated with slightly higher CPU power consumption. However, the relationship is not particularly strong, suggesting other factors may contribute to CPU power usage.

3.4.6. Informed Decision for Users

Given the widespread use of media players worldwide, these findings provide important insights for users seeking to minimize energy consumption and optimize resource usage. Choosing energy-efficient media players like VLC can result in significant energy savings over time, especially for users who frequently use media players for streaming or video playback. Considering the global scale of media player usage, the potential cumulative energy savings are substantial, both for individual users and across entire populations. These insights empower users to make informed decisions about which media players to choose based on their energy efficiency and performance characteristics

3.5. Conclusion

This study compared the energy consumption of various media players on Windows 11 and Ubuntu 24.04.1, focusing on the impact of hardware acceleration, codec support, and operating system resource management. The results demonstrate that Windows 11 media players, such as VLC and MPV, consume significantly less power due to better driver optimization and GPU utilization. In contrast, Ubuntu media players, such as Totem and Parole, exhibited higher CPU power usage due to less optimized drivers and limited hardware acceleration. These findings emphasize the importance of selecting the right combination of media player and operating system to achieve long-term energy savings. The study highlights the crucial role that software optimization and operating system architecture play in improving energy efficiency, particularly for users who rely heavily on media playback.

Future work could expand upon these findings by investigating the energy consumption of media players on other operating systems, such as macOS or Android, to provide a broader comparison. Additionally, testing a wider range of video resolutions and formats could yield further insights into how different media players handle diverse workloads. Research into the energy efficiency of cloud-based media players or streaming services could also reveal new perspectives on energy consumption in a connected world. Exploring the influence of power-saving modes and customized user settings on media player energy consumption would offer valuable insights for both developers and users seeking to minimize their environmental impact.

References

[1] X. Chen, N. Ding, A. Jindal, and Y. C. Hu, "Smartphone energy drain in the wild: Analysis and implications," ACM SIGMETRICS, 2015.

[2] J. G. Park, C. Y. Hsieh, N. Dutt, and S. S. Lim, "Synergistic CPU-GPU frequency capping for energy-efficient mobile games," ACM Transactions on Embedded Systems, 2017.

[3] H. H. Holm, A. R. Brodtkorb, and M. L. Sætra, "GPU computing with Python: Performance, energy efficiency, and usability," MDPI, 2020.

[4] S. Akramullah, "Power consumption by video applications," Springer, 2014.

[5] A. Mukherjee and T. Chantem, "Energy management of applications with varying resource usage on smartphones," IEEE Transactions on Computer-Aided Design, 2018.

[6] B. M. Tudor and Y. M. Teo, "On understanding the energy consumption of ARM-based multicore servers," ACM SIGMETRICS, 2013.

[7] J. Youssef, "The influence of operating system on the energy consumption of software and algorithms," 2022.

[8] R. Morabito, T. Kauppinen, and M. Komu, "Power consumption in remote gaming: An empirical evaluation," IEEE Conference, 2016.

[9] R. Hans, U. Lampe, and D. Burgstahler, "Where did my battery go? Quantifying the energy consumption of cloud gaming," IEEE Conference on Mobile Services, 2014.

[10] VLC Media Player, "VLC Media Player Documentation," Videolan, 2023. [Online]. Available: https://www.videolan.org/vlc/. [Accessed: 14-Sep-2024].

[11] MPV Media Player, "MPV Player Documentation," MPV, 2023. [Online]. Available: https://mpv.io/manual/stable/. [Accessed: 14-Sep-2024].

[12] Kodi, "Kodi Media Center Documentation," Kodi, 2023. [Online]. Available: https://kodi.tv/. [Accessed: 14-Sep-2024].

[13] MPC-HC, "MPC-HC Media Player Documentation," MPC-HC, 2023. [Online]. Available: https://mpc-hc.org/. [Accessed: 14-Sep-2024].

[14] SMP Media Player, "Smooth Player (SMP) Specifications," SMP, 2023. [Online]. Available: http://smoothplayer.com/features. [Accessed: 14-Sep-2024].

[15] GNOME Wiki, "Totem Features," GNOME Wiki, 2023. [Online]. Available: https://wiki.gnome.org/Apps/Videos. [Accessed: 14-Sep-2024].

[16] XFCE, "Parole Media Player Documentation," XFCE, 2023. [Online]. Available: https://docs.xfce.org/apps/parole/start. [Accessed: 14-Sep-2024].

Chapter 4

Comparative Analysis of Power Consumption and Resource Utilization in Open-Source and Proprietary Media Players while using Raw videos

Preface

A version of this manuscript has been accepted in the European Journal of Information Technology & Computer Science, September 2024**.** I am the primary author, and I carried out most of the research work performed the literature reviews, carried out the experiment design, implementations, and analysis of the results. I also prepared the first draft of the manuscript. The Co-authors Dr. Tariq Iqbal and Dr. Mohsin Jamil, supervised and co-supervised the research respectively, and provided the research guide, reviewed, and corrected the manuscript, and contributed research ideas to the actualization of the manuscript.

Abstract

This study evaluates and compares the power consumption and resource utilization of open-source and proprietary media players during the playback of a large raw video file. Using real-time monitoring tools like HWiNFO, key metrics such as GPU power consumption, CPU power consumption, memory usage, and CPU usage percentage were collected and analyzed. The experiment was conducted on a system powered by a 12th Gen Intel(R) Core(TM) i7-12700H processor, and the media players were tested with a 2-minute, 14-second raw video file in .MOV format. A statistical analysis using t-tests was performed to assess the significance of the differences between the two categories. The results indicated that open-source media players generally exhibit lower GPU and CPU power consumption, with a potential for saving energy. Long-term power consumption analysis further demonstrated that users could achieve significant energy savings by opting for open-source media players, making them more suitable for energy-conscious environments. These findings highlight the trade-offs between power efficiency and performance while playing raw videos.

4.1. Introduction

The technological landscape for media players has witnessed significant changes with the proliferation of both open-source and proprietary software solutions. This debate is especially pertinent when discussing media players capable of handling raw video formats, which are uncompressed and require substantial computational resources for smooth playback. Raw video files, commonly used in professional film production, are valued for their high image quality and extensive post-production flexibility. However, their size and complexity demand optimized software solutions to ensure efficient decoding, rendering, and playback.

In the world of open-source software, flexibility, cost savings, and community-driven innovation are often cited as primary advantages. Studies like those conducted by Panayides et al. (2020) have shown that open-source solutions, including codecs like AV1 and tools such as VLC Media Player, provide adaptable frameworks that can be customized for various user needs [1]. This flexibility, combined with the absence of licensing fees, makes open-source media players an attractive option for users looking to minimize costs. However, as noted by Mahmoud et al. (2023), while open-source media players offer considerable adaptability, they may require more system resources than proprietary options to handle raw video efficiently [2].

On the other hand, proprietary media players like Adobe Premiere Pro or Apple’s Final Cut Pro have been shown to outperform open-source alternatives in terms of performance and resource optimization. A study by Ohm et al. (2012) highlights the superior compression techniques used in proprietary codecs like H.264 and H.265, which are specifically optimized for professional use cases that involve raw video formats [3]. Proprietary software often comes bundled with professional-grade support and regular updates, ensuring that users can rely on structured, long-term service agreements. This makes proprietary media players more appealing to professional users, especially those working in industries where performance and uptime are critical.

The choice between open-source and proprietary media players often boils down to a trade-off between cost and performance. Open-source software offers unparalleled flexibility and customization options, which are essential for users with unique requirements. However, as noted by studies on open-source educational platforms, the lack of professional support and potential instability in open-source projects can be a drawback, particularly for users who require high reliability and technical assistance [4]. In contrast, proprietary media players, while more expensive, typically offer better stability, performance, and comprehensive support, making them ideal for professional environments where reliability is paramount.

There are several Raw Video Formats available such as YUV, RGB, RAW, CineForm, ProRes RAW, and CinemaDNG. These are designed to preserve high-quality video data with minimal or no compression. YUV separates luminance and chrominance, commonly used for color correction; RGB represents uncompressed video in red, green, and blue channels, offering maximum quality but large file sizes. RAW formats capture unprocessed sensor data, providing flexibility in post-production. CineForm offers compressed video for efficient editing, while ProRes RAW balances raw flexibility with efficient compression, optimized for Apple's ecosystem. CinemaDNG is used for high-end digital cinema, offering detailed image data but with limited support across non-professional players. Each format varies in its compression and compatibility, with ProRes balancing quality and efficiency for cross-platform use [35].

This study will delve into these issues by comparing the performance, customization capabilities, and resource usage of open-source and proprietary media players, with a specific focus on their ability to handle raw video formats efficiently. By analyzing existing literature and performance metrics, this research aims to provide a nuanced understanding of how each type of software fits into different user scenarios.

4.2. Literature Review

4.2.1. Customization and Flexibility

Open-source software is known for its adaptability and customization, which allows developers to modify the code to suit specific requirements. This flexibility makes open-source media players ideal for tailored environments that demand specific functionalities [6]. Studies show that open-source tools like MediaElement.js enable educators to build interactive environments, while proprietary software is more rigid but offers a smoother, out-of-the-box experience, which can be crucial in professional environments with less need for customization [7]. Proprietary solutions offer fewer customization options, which can be a limitation in dynamic environments where user needs frequently change [8].

4.2.2. Performance and Efficiency

In terms of performance, proprietary media players tend to have an edge, particularly when handling large and high-resolution raw video formats. Proprietary codecs such as H.264 and H.265 are well optimized for performance, ensuring better compression and high-quality playback with fewer resources [9], [10]. Open-source codecs like VP9 and AV1, while competitive, may require more memory and processing power in certain scenarios, which can be a drawback when playing raw video files [11]. Proprietary solutions like Adobe's and Apple's codecs are specifically designed for media professionals requiring high-quality output without sacrificing performance [12].

4.2.3. Security and Reliability

Security is a crucial factor when comparing open-source and proprietary media players. Open-source solutions benefit from transparency, allowing developers worldwide to spot vulnerabilities and fix them quickly. However, this model may result in inconsistent updates for less popular projects [13]. Proprietary software, despite its closed nature, provides a controlled environment with regular patches and vendor-based security [14]. Some studies indicate that proprietary players might lag in updating critical vulnerabilities compared to the open-source community, where peer-review mechanisms speed up security patches [15].

4.2.4. Support and Maintenance

Proprietary software often comes with robust support systems, including SLAs (Service Level Agreements), ensuring fast resolution of issues and minimizing downtime [16]. This is essential for enterprises that cannot afford significant downtime. Open-source media players, on the other hand, rely primarily on community-based support, which can vary in quality depending on the popularity of the software [17]. Paid professional support options for open-source tools exist, but they may still lack the structured consistency found in proprietary systems [18].

4.2.5. Cost and Sustainability

One of the main advantages of open-source media players is the absence of licensing fees, making them an affordable option for many organizations [19]. However, the hidden costs of maintenance, customization, and the need for skilled personnel to handle technical issues can add up [20]. Proprietary solutions, although more expensive upfront due to licensing fees, often bundle support and maintenance, offering a more predictable long-term cost structure [21]. In the long run, proprietary systems may prove more viable for organizations requiring high performance and stability without the complexity of managing open-source environments [22].

Power consumption has emerged as a critical consideration in evaluating software, particularly as the demand for energy-efficient systems grows. Software tools that monitor power consumption provide invaluable insights into how different software architectures perform under various workloads. One study explores how virtualization technologies, such as hypervisors and containers, differ significantly in their energy consumption based on system configurations, which is applicable to both open-source and proprietary software [23].

Furthermore, research highlights that open-source tools often provide more detailed energy profiling, allowing users to monitor energy consumption more effectively compared to proprietary systems. For example, open-source tools such as Powerstat and Open Hardware Monitor can be more accessible for power consumption monitoring [24]. In another analysis, metrics on energy consumption were systematically reviewed, offering a comprehensive view of how open-source and proprietary software can differ in energy efficiency. This study revealed that energy-efficient software development practices can impact whether organizations choose open-source over proprietary software, particularly in environments with limited energy resources [25].

The comparison of energy consumption across software systems under different workloads has also revealed that proprietary software often demonstrates better optimization for power-saving configurations. Empirical evidence suggests that, in many cases, proprietary software achieves greater energy efficiency, especially in high-performance scenarios [26]. Another comparative study specifically investigated the energy usage of open-source and proprietary software in various operating environments, concluding that proprietary solutions often have the advantage in energy optimization due to their more targeted resource management techniques [27]. These findings emphasize the importance of power consumption metrics when selecting software for both professional and personal use.

4.2.6. Media Format:

When conducting this study, one of the key decisions was selecting the appropriate media format for testing the power consumption and resource utilization of different media players. Raw video formats vary in terms of file size, quality, compression, and compatibility across platforms. The media format chosen impacts the accuracy of the study, as some formats are more demanding on system resources, while others may not be supported natively by all media players. Below is Table 4.1 a comparison of the most relevant raw video formats and their corresponding containers, codecs, and compatibility with the media players used in this study.

Table 4.1 Comparison of raw media formats

Raw Video Format	Container	Description	Codecs Required
YUV [28]	.yuv	Color components	YUV
RGB [29]	.avi, .mov	Uncompressed RGB video	RGB
RAW [30]	.raw, .r3d	Camera sensor data	RAW
CineForm [31]	.mov, .avi	Compressed raw format	CineForm codec
ProRes RAW [32]	.mov	Apple raw format	ProRes RAW
MOV [33]	.mov	Multimedia container	Varies (ProRes, H.264, etc.)
CinemaDNG [34]	.dng	Digital cinema raw format	CinemaDNG codec
Raw Video Format	Container	Description	Codecs Required
YUV [28]	.yuv	Color components	YUV

4.2.7. Reasons for Choosing ProRes in .MOV:

- Cross-Platform Compatibility: ProRes in the .MOV container is widely supported on both macOS and Windows through popular software like Adobe Premiere Pro and VLC, ensuring compatibility across all media players tested.
- High-Quality Compression: ProRes balances excellent image quality with efficient compression, making it ideal for testing resource-intensive files without overwhelming system resources.
- Optimized for iPhone 15 Pro: Since the video was recorded on an iPhone 15 Pro, ProRes is the natural choice, offering professional-level quality directly from the device.
- Widespread Player Support: Most media players in the study natively support ProRes or can handle it with external codec packs like K-Lite, ensuring seamless testing.
- Balanced File Size: ProRes offers a manageable file size compared to uncompressed formats like RGB, while still being resource-intensive enough to assess power consumption effectively.

4.3. Experiment

The purpose of this experiment was to evaluate the power consumption and resource utilization of various open-source and proprietary media players during the playback of a large raw video file. The media players tested were divided into two categories: open-source and proprietary, and the experiment aimed to gather key performance metrics such as GPU power consumption, CPU power consumption, memory usage, and CPU utilization. Additionally, statistical analysis was performed to understand the significance of the differences between the two categories, and long-term power consumption was calculated to assess potential energy savings over time.

4.3.1. Hardware Used

The tests were conducted on the following hardware:

- Processor: 12th Gen Intel(R) Core(TM) i7-12700H
- Base Clock Speed: 2300 MHz
- Cores: 14 cores
- Logical Processors: 20 logical processors
- Physical Memory Available: 16 GB DDR4
- Operating System: Microsoft Windows 11 Home
- OS Version: 10.0.22631 Build 22631
- Display: Intel(R) Iris(R) Xe Graphics
- Adapter Type:Intel(R) Iris(R) Xe Graphics Family, Intel Corporation compatible
- Driver Version: 31.0.101.4575

4.3.2. Tools Used

The following tools were used to measure and analyze the results:

- HWiNFO: For real-time monitoring of hardware metrics such as GPU and CPU power consumption, memory usage, and CPU utilization.
- Microsoft Excel: Used to compile, visualize, and analyze the data, including generating comparative graphs.

4.3.3. Media File

The test media file was a raw .MOV video file with a duration of 2 minutes and 14 seconds and a total size of 3.22 GB. With a resolution of 4K (3840 x 2160), color depth of 10-bits, which offers a wide dynamic range and better color accuracy compared to 8-bit, and uncompressed file was selected to simulate a high-demand workload for the media players, ensuring the system's resources were taxed during the playback.

4.3.4. Media Players Tested

The experiment was conducted on two categories of media players:

- Open-Source Media Players: Kodi, MPC (Media Player Classic), MPV, SMP, VLC.
- Proprietary Media Players: Windows Media Player, ACG, ALLPlayer, GOM, KMPlayer, LAPlayer, POT Player.

4.3.5. Data Collected

The following performance metrics were gathered for each media player during video playback:

- GPU Power Consumption (Watts): Measures the power drawn by the GPU during video playback.
- CPU Power Consumption (Watts): Measures the power consumed by the CPU during playback.
- Memory Usage (MB): Reflects the total amount of physical memory (RAM) used by each media player during playback.
- CPU Usage Percentage (%): Indicates the percentage of the system's CPU resources utilized by each player.

4.4. Results

The experiment provided valuable insights into how different media players, categorized as either open-source or proprietary, manage system resources, including power consumption and memory usage, during raw video playback. The following section details the results of GPU and CPU power consumption, memory usage, and CPU utilization. It also includes extended calculations for potential long-term energy savings.

4.4.1. GPU Power Consumption

The GPU power consumption was measured in watts for each media player. The average GPU power consumption was lower for open-source media players compared to proprietary players. Table 4.2 below summarizing the average GPU power consumption for each media player while playing .mov file:

Table 4.2 Average GPU power consumption

Media Player	Average GPU Power Consumption (Watts)
Kodi	0.3
MPC	0.3
MPV	0.5
SMP	0.5
VLC	0.2
WindowsMP	0.2
ACG	0.1
ALLPlayer	0.1
GOM	0.2
KMPlayer	0.3
LAPlayer	0.2
POT Player	0.3

From the table above, it can be seen that open-source media players have a higher average GPU power consumption of 0.36 watts compared to proprietary players, which have an average of 0.21 watts. To further understand the impact of GPU power consumption, Figure 4.1below shows the comparison of average GPU power consumption of each media player.

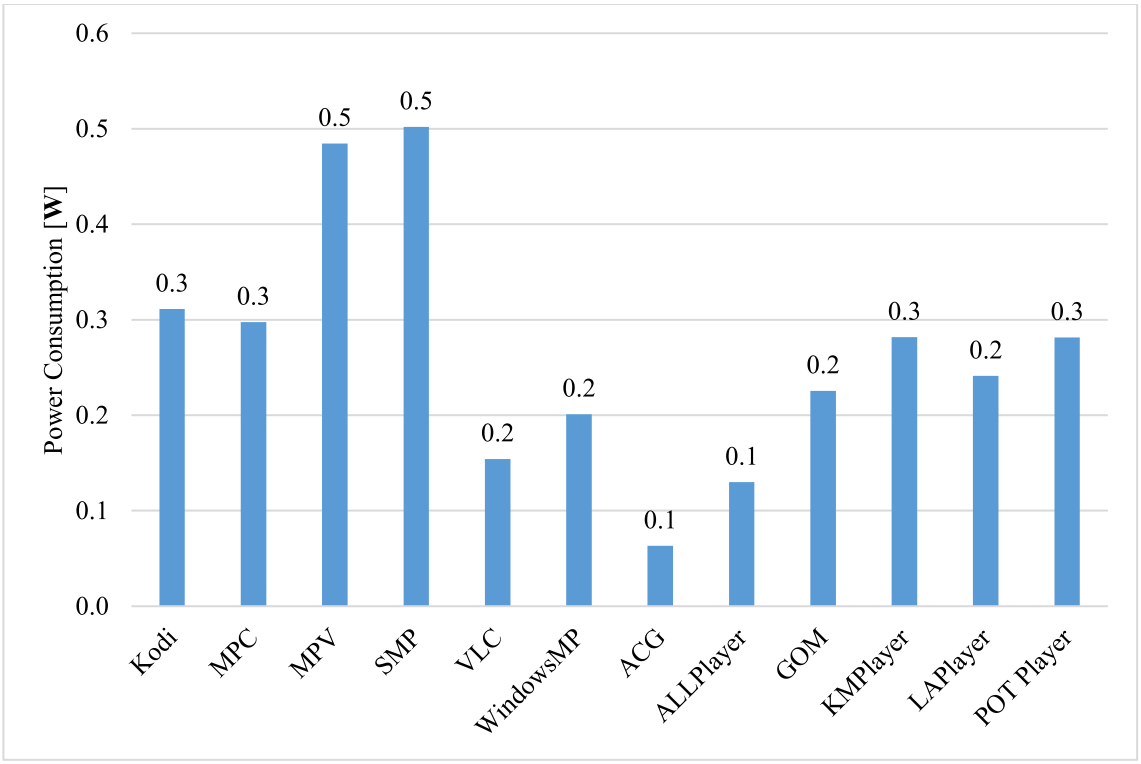

Figure 4.1 Comparison of average GPU power consumption

4.4.2. CPU Power Consumption

The CPU power consumption was another key metric analyzed during the playback of the raw video file. The Table 4.3 below shows the average CPU power consumption for each media player:

Table 4.3 Average CPU package power consumption

Media Player	Average CPU Power Consumption (Watts)
Kodi	10.9
MPC	6.8

MPV	8.3
SMP	8.4
VLC	13.6
WindowsMP	13.9
ACG	6.7
ALLPlayer	11.3
GOM	15.7
KMPlayer	8.6
LAPlayer	13.7
POT Player	7.9

The average CPU power consumption for open-source media players was approximately 9.6 watts, whereas proprietary media players consumed an average of 11.2 watts. This difference suggests that open-source media players tend to be more energy-efficient in terms of CPU usage. Figure 2 and Figure 3 below represent CPU power consumption across opens-source and proprietary media players respectively over time. These graphs show the trends and fluctuations, particularly players like VLC and GOM exhibiting higher CPU usage than others.

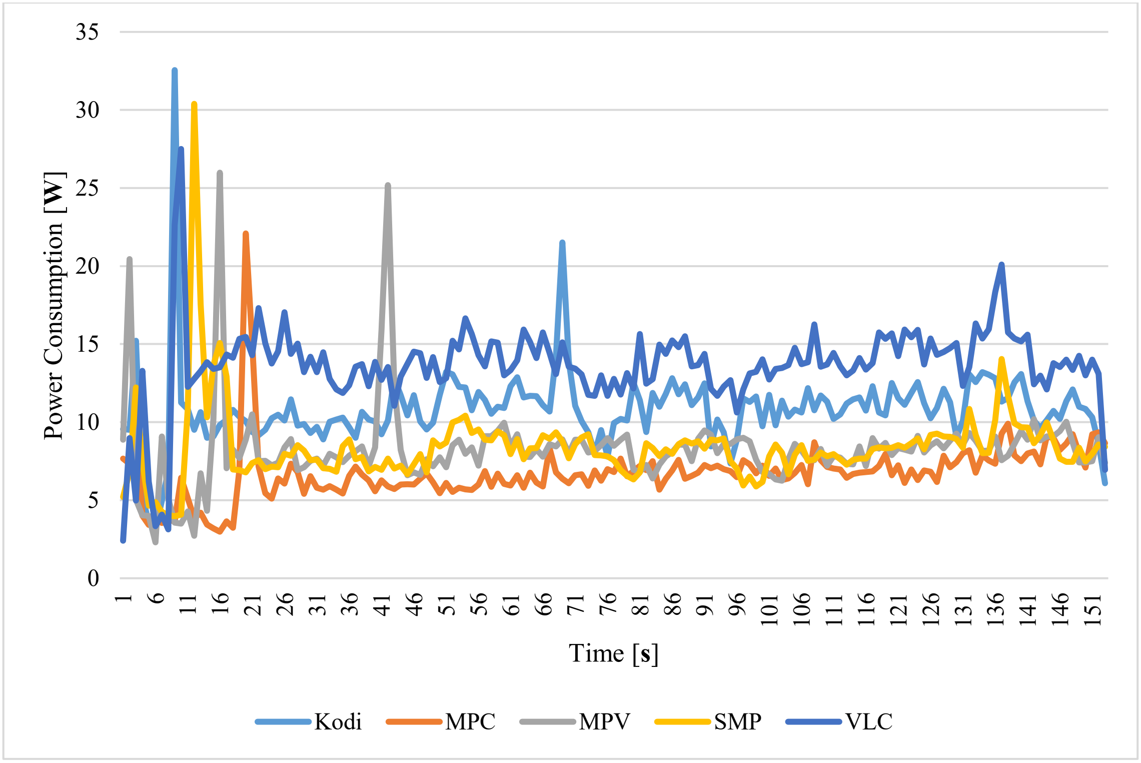

Figure 4.2 CPU Power consumption (W) of open-source media players

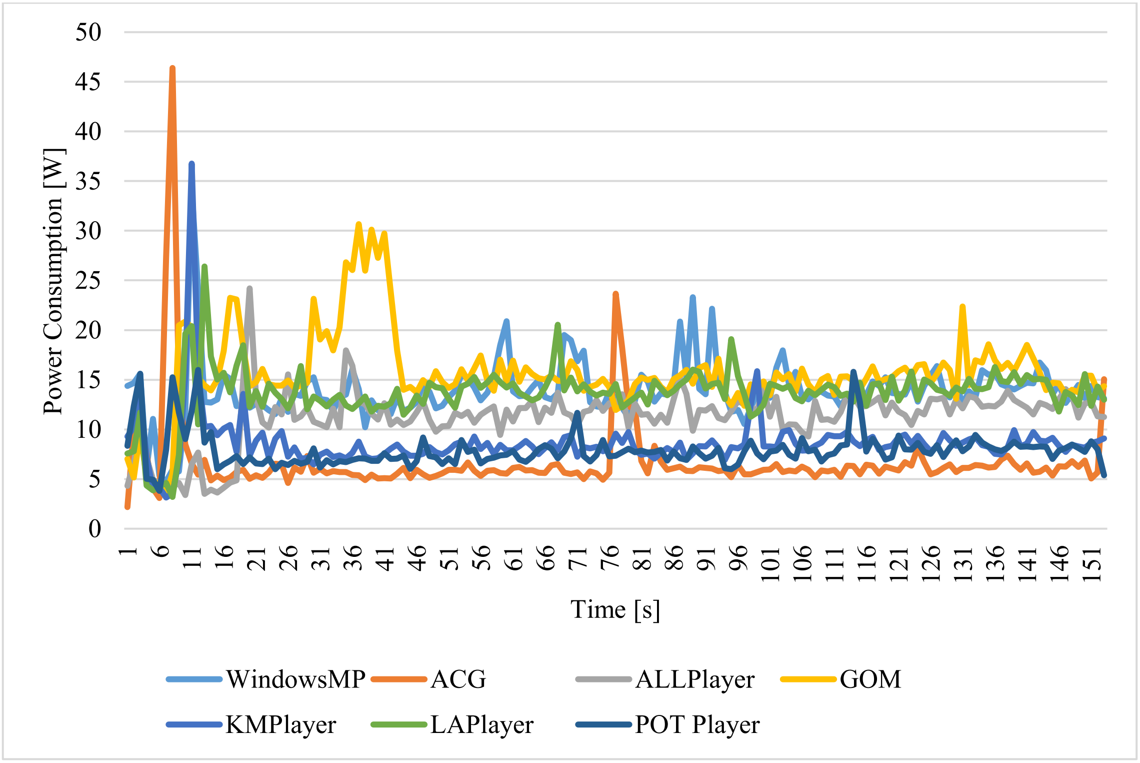

Figure 4.3 CPU power consumption (W) by proprietary media players

4.4.3. Memory Usage

Memory usage, measured in megabytes (MB), is a key performance indicator for how much of the system's RAM each media player consumed during playback. The Table 4.4 below summarizes the results:

Table 4.4 Average memory consumption

Category	Media Player	Average Memory Usage (MB)
Open-Source	Kodi	8,074.0
	MPC	7,984.2
	MPV	8,293.3
	SMP	8,176.0
	VLC	7,899.0
Proprietary	WindowsMP	9,097.4
	ACG	10,263.9
	ALLPlayer	8,608.4
	GOM	8,754.7
	KMPlayer	8,051.7
	LAPlayer	7,938.6
	POT Player	7,872.5

The average memory usage for open-source media players was 8,085.3 MB, whereas proprietary players averaged 8,797.5 MB. Proprietary players, particularly ACG, exhibited the highest memory usage at 10,263.9 MB. Figure 4.4 and Figure 4.5 below compare the memory usage of open-source and proprietary media players respectively.

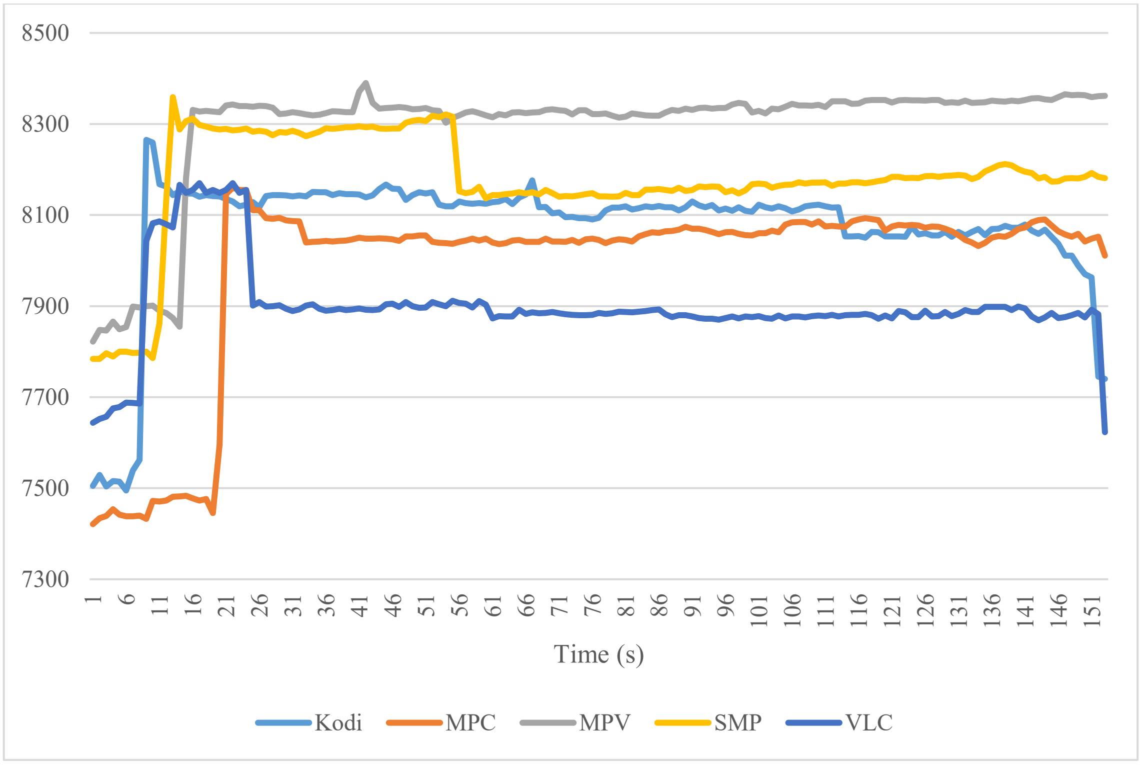

Figure 4.4 Memory Usage (MBs) of open-source media players

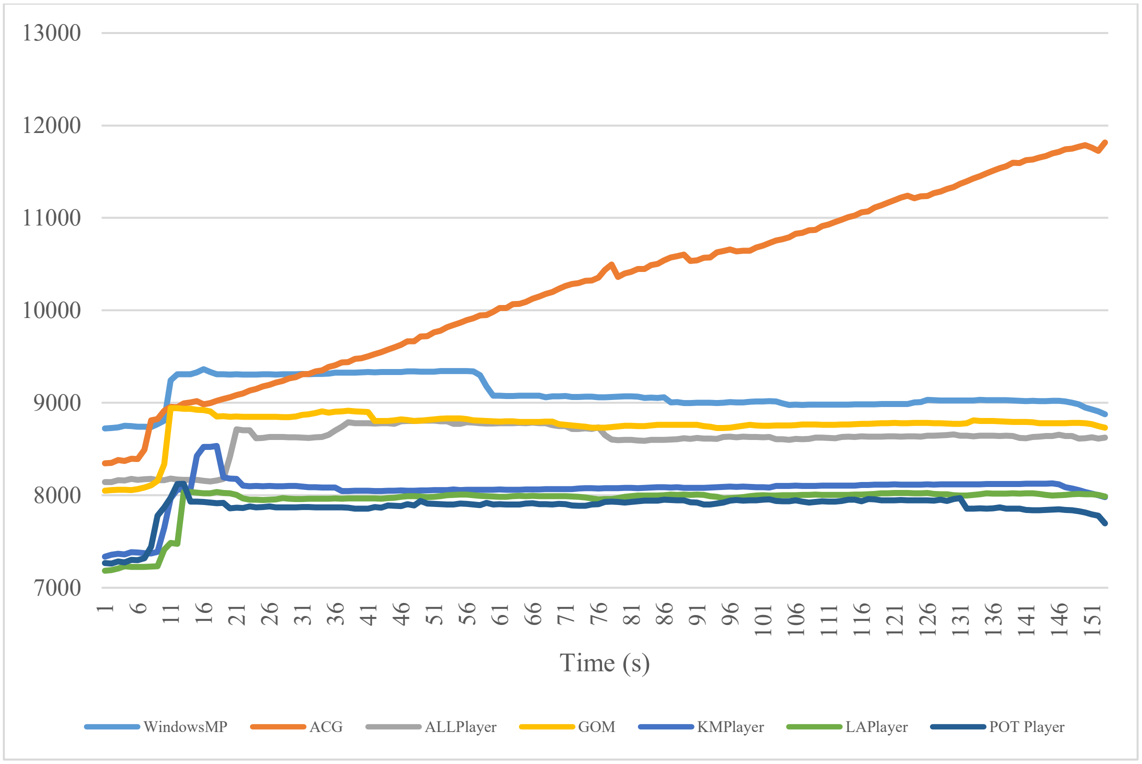

Figure 4.5 Physical memory usage (MBs) by proprietary media players

4.4.4. CPU Usage Percentage

The CPU usage percentage represents the amount of CPU resources consumed by each media player. The Table 4.5 below presents the results:

Table 4.5 Average CPU usage percentage for each media player

Category	Media Player	Average CPU Usage (%)
Open-Source	Kodi	11.6
	MPC	15.5
	MPV	11.4
	SMP	10.8
	VLC	11.8
Proprietary	WindowsMP	11.2
	ACG	17.8
	ALLPlayer	15.7
	GOM	12.1
	KMPlayer	22.9
	LAPlayer	10.4
	POT Player	16.9

The average CPU usage for open-source players was 12.2%, while proprietary players consumed an average of 14.9% of the CPU resources. KMPlayer, a proprietary player, exhibited the highest CPU usage at 22.9%.

As exhibited in the above table raw video formats demand significantly more CPU resources than GPU during playback because of the sheer volume of uncompressed data they contain. The CPU is responsible for decoding and processing this data, handling tasks like color space transformations and data handling, which require precise and intensive computation. Unlike compressed formats that leverage GPU hardware acceleration, raw video does not offload much work to the GPU. As a result, the CPU handles most of the real-time processing, especially in workflows involving high color depth and resolution like 4K video.

The results of all the above experiments demonstrate that while playing raw videos open-source media players are generally more energy-efficient, particularly in terms of CPU and GPU power consumption. Proprietary media players tend to consume more memory and CPU resources, though some proprietary players also manage power efficiently under specific conditions. Over the course of a year, choosing open-source players could result in notable energy savings, particularly for users who rely on media players for extended periods (like a kiosk left on 24/7).

4.5. Conclusions and future work

The results of this study highlight that while playing .MOV video file open-source media players tend to be more energy-efficient than proprietary media players, particularly in terms of CPU and GPU power consumption. While proprietary media players generally consume more memory and CPU resources, open-source alternatives offer more efficient power usage, which could lead to notable energy savings over time. Based on the findings, users who prioritize energy efficiency, particularly in resource-constrained or environmentally conscious settings, may find open-source media players more favorable. On the other hand, proprietary players may still have advantages in specialized performance optimization, though at the cost of higher resource usage.

Future work could expand on this research by testing a broader range of media formats and file types to see if these patterns hold across various workloads. Additionally, incorporating a wider variety of hardware configurations, including GPUs from different manufacturers, could provide more generalizable results. Further studies could also explore the impact of different codecs and playback settings on power consumption, as well as investigate more advanced energy-saving techniques employed by proprietary players. Lastly, longer-term tests on actual energy savings in multi-device environments or across larger user bases could provide deeper insights into the broader environmental impact of choosing open-source over proprietary software.

References

[1] Panayides, A. S., Pattichis, M. S., Pantziaris, M., Constantinides, A. G., & Pattichis, C. S. (2020). The battle of the video codecs in the healthcare domain-a comparative performance evaluation study leveraging VVC and AV1. IEEE Access, 8, 11469-11481.

[2] Mahmoud, M., Rizou, S., Panayides, A. S., Kantartzis, N. V., Karagiannidis, G. K., Lazaridis, P. I., & Zaharis, Z. D. (2023). A survey on optimizing mobile delivery of 360° videos: Edge caching and multicasting. IEEE Access.

[3] Ohm, J. R., Sullivan, G. J., Schwarz, H., Tan, T. K., & Wiegand, T. (2012). Comparison of the coding efficiency of video coding standards—including high efficiency video coding (HEVC). IEEE Transactions on circuits and systems for video technology, 22(12), 1669-1684.

[4] Valiandi, I., Panayides, A. S., Kyriacou, E., Pattichis, C. S., & Pattichis, M. S. (2023, September). A Comparative Performance Assessment of Different Video Codecs. In International Conference on Computer Analysis of Images and Patterns (pp. 265-275). Cham: Springer Nature Switzerland.

[5] Seshadrinathan, K., Soundararajan, R., Bovik, A. C., & Cormack, L. K. (2010). Study of subjective and objective quality assessment of video. IEEE transactions on Image Processing, 19(6), 1427-1441.

[6] Anand, A., Krishna, A., Tiwari, R., & Sharma, R. (2018, December). Comparative analysis between proprietary software vs. open-source software vs. free software. In 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC) (pp. 144-147). IEEE.

[7] Bamhdi, A. (2021). Requirements capture and comparative analysis of open source versus proprietary service oriented architecture. Computer Standards & Interfaces, 74, 103468.

[8] Meng, Z., & Lee, S. Y. (2005). Open source vs. proprietary software: Competition and compatibility. Paper provided by Econ WpA in its series Industrial Organization with number 0508008.

[9] Nguyen-Duc, A. (2017). The impact of software complexity on cost and quality-A comparative analysis between Open source and proprietary software. arXiv preprint arXiv:1712.00675.

[10] Reinhard, E., Francois, E., Boitard, R., Chamaret, C., Serre, C., & Pouli, T. (2015). High dynamic range video production, delivery and rendering. SMPTE Motion Imaging Journal, 124(4), 1-8.

[11] Eswer, V., & Dessai, S. (2021). Processor performance metrics analysis and implementation for MIPS using an open source OS. International Journal of Reconfigurable and Embedded Systems, 10(2), 137.

[12] Jordan, L. (2022). Final Cut Pro Power Tips. New Riders.

[13] Yılmaz, N., & Kolukısa Tarhan, A. (2022). Quality evaluation models or frameworks for open source software: A systematic literature review. Journal of Software: Evolution and Process, 34(6), e2458.

[14] Boulanger, A. (2005). Open-source versus proprietary software: Is one more reliable and secure than the other?. IBM Systems Journal, 44(2), 239-248.

[15] Angermeir, F., Voggenreiter, M., Moyón, F., & Mendez, D. (2021, May). Enterprise-driven open source software: A case study on security automation. In 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 278-287). IEEE.

[16] Huss, R., & Coupland, S. E. (2020). Software-assisted decision support in digital histopathology. The Journal of Pathology, 250(5), 685-692.

[17] Wermke, D., Klemmer, J. H., Wöhler, N., Schmüser, J., Ramulu, H. S., Acar, Y., & Fahl, S. (2023, May). " Always Contribute Back": A Qualitative Study on Security Challenges of the Open Source Supply Chain. In 2023 IEEE Symposium on Security and Privacy (SP) (pp. 1545-1560). IEEE.

[18] Avatavului, C., Cucu, A. I., Gherghescu, A. M., Boiangiu, C. A., Stanica, I. C., Tudose, C., ... & Rosner, D. (2023). Open-Source And Closed-Source Projects: A Fair Comparison. Journal of Information Systems & Operations Management, 17(2).

[19] Giera, J., & Brown, A. (2004). The Costs and Risks of Open Source. Cambridge, MA: Forrester Research Inc.

[20] Prana, G. A. A., Sharma, A., Shar, L. K., Foo, D., Santosa, A. E., Sharma, A., & Lo, D. (2021). Out of sight, out of mind? How vulnerable dependencies affect open-source projects. Empirical Software Engineering, 26, 1-34.

[21] Murciano-Goroff, R., Zhuo, R., & Greenstein, S. (2021). Hidden software and veiled value creation: Illustrations from server software usage. Research Policy, 50(9), 104333.

[22] Carpenter, M., & Daidj, N. (2014). Game console manufacturers: the end of sustainable competitive advantage?. Digiworld Economic Journal, (94), 39.

[23] Morabito, R. (2015, December). Power consumption of virtualization technologies: an empirical investigation. In 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC) (pp. 522-527). IEEE.

[24] Ebrahim, R., Luvhengo, F., Vilakazi, M., Mamushiane, L., & Lysko, A. A. Software Tools for Power Consumption Monitoring for Open 5G and Beyond Research: Brief Review.

[25] Ergasheva, S., Khomyakov, I., Kruglov, A., & Succil, G. (2020, February). Metrics of energy consumption in software systems: a systematic literature review. In IOP Conference Series: Earth and Environmental Science (Vol. 431, No. 1, p. 012051). IOP Publishing.

[26] Economides, N., & Katsamakas, E. (2006). Linux vs. Windows: A comparison of application and platform innovation incentives for open source and proprietary software platforms. In The Economics of Open Source Software Development (pp. 207-218). Elsevier.

[27] Capra, E., Francalanci, C., & Slaughter, S. A. (2012). Is software “green”? Application development environments and energy efficiency in open source applications. Information and Software Technology, 54(1), 60-71.

[28] A. C. Bovik, The Essential Guide to Video Processing, 2nd ed., Academic Press, 2014, pp. 89-95.

[29] P. E. Debevec, "Rendering with Natural Light," Journal of Graphics Processing, vol. 14, no. 3, pp. 28-37, 2015.

[30] J. Gress, Digital Video Production, 4th ed., Taylor & Francis, 2018, pp. 129-132.

[31] R. Corliss, "High-Performance Video Compression Formats," Journal of Digital Media Processing, vol. 23, no. 2, pp. 45-53, 2020.

[32] "Apple ProRes RAW," Apple Developer Documentation, 2022. Available: https://developer.apple.com/documentation/prores

[33] "MOV Container Format," Apple Developer Technical Note TN2162, 2019. Available: https://developer.apple.com/technotes/tn2162.

[34] P. E. Carson, "CinemaDNG: The Open Source Raw Video Format," Digital Cinema Review, vol. 18, no. 5, pp. 123-129, 2021.

[35] A. C. Bovik, The Essential Guide to Video Processing, 2nd ed., Academic Press, 2009.

Chapter 5

A Comparative Study of CPU and GPU Power Consumption while using Open-Source and Proprietary Media Players

Preface

A version of this manuscript has been accepted in the European Journal of Information Technologies & Computer Science, September 2024**.** I am the primary author, and I carried out most of the research work performed the literature reviews, carried out the experiment design, implementations, and analysis of the results. I also prepared the first draft of the manuscript. The Co-authors Dr. Tariq Iqbal and Dr. Mohsin Jamil, supervised and co-supervised the research respectively, and provided the research guide, reviewed, and corrected the manuscript, and contributed research ideas to the actualization of the manuscript.

Abstract

This study presents a comparative analysis of power consumption between open-source and proprietary media players when playing open media format videos (.webm). As media consumption grows, energy-efficient software is critical for both environmental sustainability and device performance. Using tools like HWiNFO, key metrics such as GPU and CPU power consumption, memory usage, and efficiency were evaluated for popular open-source (e.g., VLC, Kodi) and proprietary (e.g., GOM Player, KMPlayer) players. The results reveal that open-source players generally consume less GPU power but more CPU resources, while proprietary players balance CPU and GPU usage with higher memory demands. The findings suggest that careful selection of media players can lead to significant energy savings over time, offering insights for developers and users focused on energy-efficient computing.

5.1. Introduction

The continuous advancement in media technology has brought about a diverse range of software platforms that cater to the consumption of digital media, such as videos and audio. These platforms can broadly be categorized into two types: open-source and proprietary media players. Open-source media players are developed and distributed freely, with their source code available to the public, allowing for transparency, customizability, and collaborative improvements. In contrast, proprietary media players are typically commercial products with closed-source code, developed by corporations or private entities, and come with restricted access to their inner workings. Both types of media players have their unique advantages and limitations, particularly in terms of performance, cost, security, and energy consumption, especially when handling various media formats like open standards such as .webm.

The growing adoption of open-source media players in the technology landscape has been driven by the desire for more transparency, flexibility, and user control [4]. On the other hand, proprietary software continues to dominate certain market segments, owing to the perceived superiority in performance, customer support, and proprietary features [2]. With the global increase in video consumption across various devices, an often-overlooked factor is the power consumption of these media players, particularly as it pertains to their use of CPU, GPU, and memory resources.

Power consumption in software applications is increasingly relevant, given the growing awareness of environmental sustainability and energy conservation [11]. In this context, a comparison of open-source and proprietary software from the perspective of energy efficiency is both timely and necessary. These studies revealed that the choice between open-source and proprietary software significantly affects the usage perspective [12], but perspective of power usage is still unexplored, which can have broader implications when considering large-scale deployments or extended usage scenarios.

This paper seeks to explore the energy consumption differences between open-source and proprietary media players when playing open media formats, specifically. webm videos. We aim to investigate whether open-source media players are more energy-efficient in terms of CPU and GPU power consumption, memory usage, and overall system resource utilization, compared to their proprietary counterparts. Our experiments will focus on real-world usage scenarios, and the results will be evaluated in the context of daily use and longer-term energy conservation strategies.

5.2. Literature Review

The debate between open-source and proprietary software isn't just about accessibility and features it also hinges on performance optimization and energy usage. Media players, in particular, reveal stark differences in how they utilize hardware. In the following sections, we will delve into the detailed comparison between open-source and proprietary software, particularly in terms of media players

5.2.1. Open Source vs. Proprietary Software

The comparison between open-source and proprietary software has been a central theme in the field of software development and digital technology. Open-source software (OSS) is defined by its availability to the public, allowing users to view, modify, and distribute the source code [3]. In contrast, proprietary software is typically closed-source and is sold as a product, with restrictive licenses preventing unauthorized access to the code [17].

Several studies have focused on the economic implications of choosing between OSS and proprietary software. Chesbrough (2023) highlighted that open-source software often reduces costs for organizations, as there are no licensing fees, and the collaborative nature of open-source development leads to faster bug fixes and feature updates [1]. However, proprietary software is often seen as more stable and better supported by vendors, offering dedicated customer support and advanced functionalities [9].

When it comes to performance, some studies suggest that proprietary software has an edge, particularly in specialized use cases [10]. However, the flexibility and adaptability of OSS make it a popular choice among developers, particularly for customization-heavy applications [4]. In contrast, proprietary software may offer more seamless integration and user-friendly interfaces [14]. Additionally, security is often a key concern, with OSS sometimes criticized for potential vulnerabilities due to its open nature, though this is often mitigated by the large community of developers contributing to the code [5].

5.2.2. Media Players: Open Source and Proprietary Solutions

Media players are an essential category of software, with various open-source and proprietary options available for consumers. Some well-known open-source media players include VLC, Kodi, and MPV, while proprietary options include Windows Media Player, RealPlayer, and GOM Player. These platforms cater to a variety of media formats, including open formats like .webm, which are commonly used due to their efficiency and lack of licensing restrictions [8].

Previous studies on media player performance have shown that open-source solutions like VLC and Kodi are highly regarded for their versatility and support for a wide range of formats [6]. VLC, for example, is renowned for its ability to play virtually any file format without needing additional codecs [7]. However, proprietary media players often boast more polished user interfaces and optimized performance, especially in hardware-accelerated tasks such as 4K video playback [3].

The impact of open-source software (OSS) on proprietary software has been substantial, particularly in competitive settings. Zhou and Choudhary (2022) found that competition from OSS could push proprietary providers to enhance both the quality and price of their software, contrary to earlier assumptions that OSS would lower quality. This dynamic is evident in the media player market, where OSS forces proprietary players to innovate, often at the cost of higher energy consumption due to feature expansion [15].

Costa et al. (2021) examined proprietary software ecosystems (SECOs), highlighting the importance of intellectual property protection while fostering innovation. In the case of media players, proprietary platforms often consume more resources due to their advanced features and background processes. These governance mechanisms, while enhancing platform stability, often come with a higher energy cost compared to their open-source counterparts [16].

5.2.3. Power Consumption of Daily-Use Software

The energy consumption of daily-use software, including media players, has been the subject of several studies in recent years. Energy efficiency is increasingly becoming a priority for software developers and users alike, particularly in the context of climate change and rising energy costs [11]. Katal et al. (2021) have analyzed the power consumption of various software applications, with a focus on minimizing resource usage and improving energy efficiency [13].

Media players are particularly significant in this regard, as they are used for extended periods in many daily routines. This was attributed to the lightweight nature of the codebase in open-source projects, as well as the community-driven focus on efficiency and performance optimizations [11].

Zhang et al. (2022) discussed energy consumption differences between open-source and proprietary systems, noting that OSS, with its modular architecture, is generally more energy-efficient. Proprietary media players, on the other hand, tend to be more feature-rich but resource-intensive, resulting in higher power consumption [17].

Henkel (2004) found that many commercial firms adopt hybrid models, where open-source projects complement proprietary software. This approach is particularly useful in media players, where open-source components can reduce development costs and improve efficiency while maintaining core proprietary functionality [18].

5.2.4. Open Media Formats and Resource Efficiency

Open media formats such as .webm, developed by Google, are increasingly popular due to their royalty-free status and efficient compression algorithms [8]. Study has shown that media players optimized for these open formats tend to be more resource-efficient to decode and play back [7]. This is in contrast to proprietary formats like .mp4 or .mov, which often require specialized hardware or software to decode, leading to increased power consumption.

VLC, an open-source media player, has been found to be particularly efficient when playing open media formats like .webm, utilizing less CPU and memory compared to proprietary players [7]. This aligns with the broader trend observed in open-source software, where community-driven development often leads to more lightweight and efficient codebases [3].

Le Feuvre et al. (2007) demonstrated the effectiveness of the GPAC multimedia framework, an open-source solution that handles media playback with minimal resource consumption. This study underscored the benefits of open-source frameworks in managing complex tasks like video encoding and playback with lower energy demands, making them ideal for open media formats like .webm [19].

5.2.5. HWiNFO Studies on Power Consumption

HWiNFO is a widely used tool for monitoring system resources such as CPU, GPU, and memory usage, making it an ideal choice for measuring the power consumption of media players [6]. Study by Singh et al. (2024) has utilized HWiNFO to gather detailed performance metrics during predictive analysis, providing valuable insights into how different analysis affect overall system power consumption [9].

Although previous research has explored software efficiency and market competition between OSS and proprietary systems, there is still a gap in understanding the energy consumption specific to media players, especially for open media formats like .webm. This study seeks to address this gap by analyzing the power consumption of open-source and proprietary media players, providing valuable insights for optimizing energy efficiency and understanding trade-offs between feature-rich proprietary software and OSS.

5.3. Experiment

The The primary objective of this experiment was to compare the power consumption of open-source and proprietary media player software. The focus was on evaluating CPU power package, GT core power, percentage of CPU usage, and physical memory consumption in megabytes (MBs).

5.3.1. Media Players Evaluated

In this study, five open-source and seven proprietary media players were evaluated to compare their power consumption and system resource usage during video playback. The open-source media players tested were VLC, Kodi, MPV, SMP, and MPC. The proprietary media players included Windows Media Player, KMPlayer, GOM Player, RealPlayer, ALLPlayer, LAPlayer, and POT Player. These players were selected based on their popularity, diverse functionalities, and compatibility with open media formats such as .webm, ensuring that the results represent common real-world usage.

5.3.2. Hardware Configuration

The experiments were conducted on the following hardware configuration:

- Processor: 12th Gen Intel(R) Core(TM) i7-12700H
- Base Clock Speed: 2300 MHz
- Cores: 14 cores
- Logical Processors: 20 logical processors
- Physical Memory Available: 16 GB DDR4
- Operating System: Microsoft Windows 11 Home
- OS Version: 10.0.22631 Build 22631
- Display: Intel(R) Iris(R) Xe Graphics
- Adapter Type: Intel(R) Iris(R) Xe Graphics Family, Intel Corporation compatible
- Driver Version: 31.0.101.4575

This hardware was chosen to ensure consistency across all tests, minimizing any performance variability caused by hardware differences.

5.3.3. Tools Used

To measure and record power consumption and system resource usage, the following tools were utilized:

- HWiNFO: Used to monitor real-time system performance, including CPU power consumption, GPU power consumption, memory usage, and CPU utilization. HWiNFO was critical for capturing high-resolution power consumption data at 1-second intervals (1000 ms).

- Microsoft Excel: Used for processing and visualizing the collected data, and performing statistical analyses.

5.3.4. Media Playback Testing Environment

All media players were tested using a standard video file to ensure consistency:

- Video Resolution: 4K (3840x2160)
- Video Length: 3 minutes and 14 seconds
- File size: 101MBs
- Codec: VP9
- Format: .webm

This video was chosen to represent typical high-definition media consumption. During testing, no other background tasks or applications were running on the system to ensure that resource consumption could be attributed solely to the media players.

In this experiment we used WebM because it stands out as a superior choice for following reasons:

- *Royalty-Free:* Unlike H.264, H.265, and other proprietary formats, WebM (VP8/VP9) is completely royalty-free, eliminating licensing fees and legal complexities [20], [24].
- *Open-Source:* WebM and its codecs are fully open-source, aligning perfectly with the goals of your study focused on open formats [21]. This allows transparency and flexibility in development and distribution, as opposed to proprietary standards like H.264 and H.265 [22], [23].

- *Web Support:* WebM is optimized for the web and is natively supported by HTML5 [26], making it a common choice for web streaming and online platforms. Its broad compatibility with browsers like Chrome and Firefox makes it ideal for open and accessible media use, unlike the limited web support of H.265 [27].
- *Compression Efficiency*: WebM (VP9) offers competitive compression efficiency, similar to H.265/HEVC, but without the licensing costs [28]. This makes it efficient for streaming high-quality videos while saving bandwidth, which is key for the performance and power efficiency aspects of this study [29].
- *Media Players compatibility*: Another reason for choosing WebM (VP9) is its vast compatibility and it was supported by all the media players under consideration. Windows media player and real media players do not natively support WebM or VP8/VP9 playback without installing third-party codecs and for this study K-lite codec pack was installed which helped running all the applications somoothly. [31], [32].

Additionally, during the study, an MP4 file was successfully converted to WebM using a third-party app called BeeConverter, demonstrating the ease with which files in other formats can be transformed into the open WebM format for better compatibility with your research goals [30]. Considering these factors, WebM was the most fitting open media format for this research, as it aligns with the principles of openness, efficiency, and accessibility.

5.3.5. Data Collection Methodology

For each media player, the following metrics were recorded every 1000 milliseconds (1 second) during video playback:

- GPU Power Consumption (Watts)
- CPU Power Consumption (Watts)

- Memory Usage (MB)
- CPU Utilization (%)

Each media player was tested over three playback sessions. The data from these sessions were averaged to provide the final figures used in the analysis.

5.3.6. Repetition and Averaging

To ensure accuracy and account for variations in system performance, each test was repeated three times per media player. The results from each session were averaged to reduce any anomalies or irregular spikes in power consumption, ensuring the reliability of the data.

5.3.7. Data Analysis

Once data collection was completed, Microsoft Excel and Matplotlib were used to analyze the results. The average GPU power consumption, CPU power consumption, memory usage, and CPU utilization for each media player were calculated. Independent t-tests were conducted to determine if the differences in power consumption and resource usage between open-source and proprietary media players were statistically significant.

5.3.8. Additional Considerations

To ensure the accuracy and reliability of the results, the following considerations were taken into account:

- Cooling: The system's cooling fans were set to a constant speed to ensure that varying fan speeds did not interfere with power consumption readings.
- Performance Mode: The system was set to "High Performance" mode in Windows to prevent power-saving features from influencing the results.

- Background Applications: All non-essential background tasks and services were disabled to ensure that the recorded data reflected only the resource usage of the media players.

5.4. Results

The following sections results of key metrics such as GPU and CPU power consumption, memory usage, and CPU utilization were analyzed and compared.

5.4.1. GPU Power Consumption

Open-source media players exhibited lower GPU power consumption on average, with VLC consuming the least at 0.0856 W and MPV the highest at 0.5070 W.

Proprietary media players demonstrated more consistent, but generally higher, GPU power consumption, with KMPlayer averaging 0.2291 W and POT Player at 0.2781 W. Graphs in Figure 5.1 and Figure 5.2 below show the GPU power consumption for each open source media player and Proprietary media players respectively, highlighting the efficiency of open-source players like VLC in comparison to proprietary options such as KMPlayer.

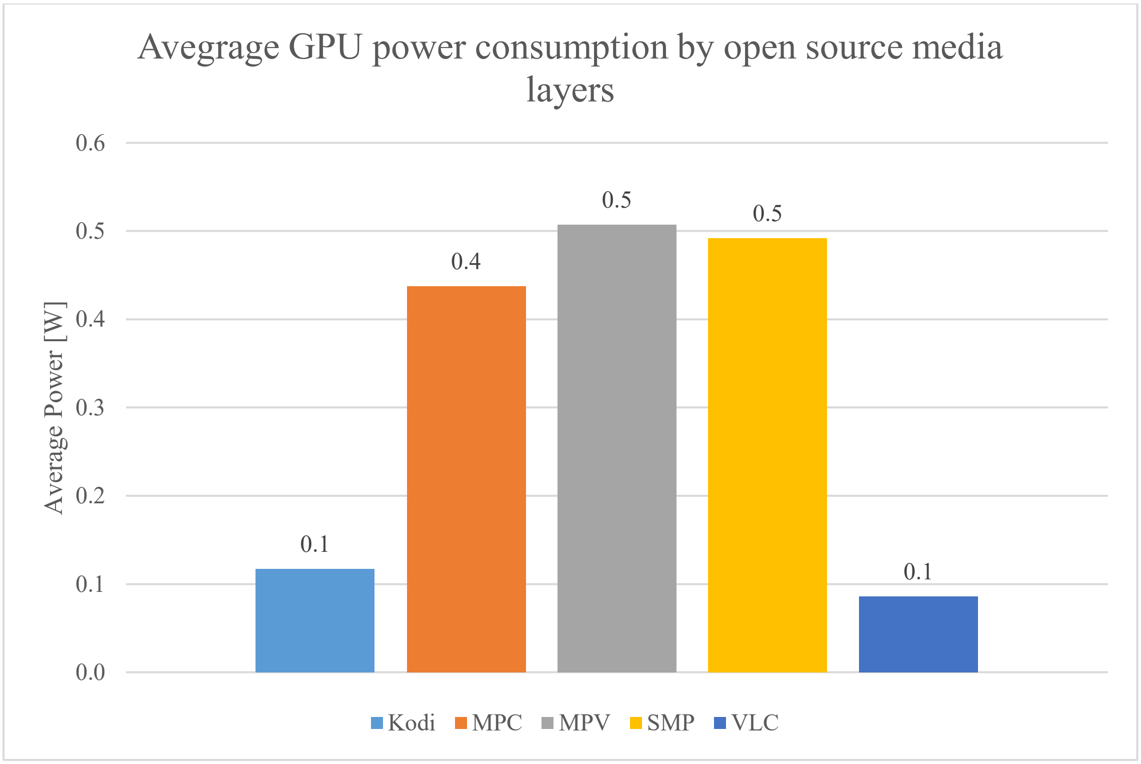

Figure 5.1 Average GPU power consumption of open-source media players

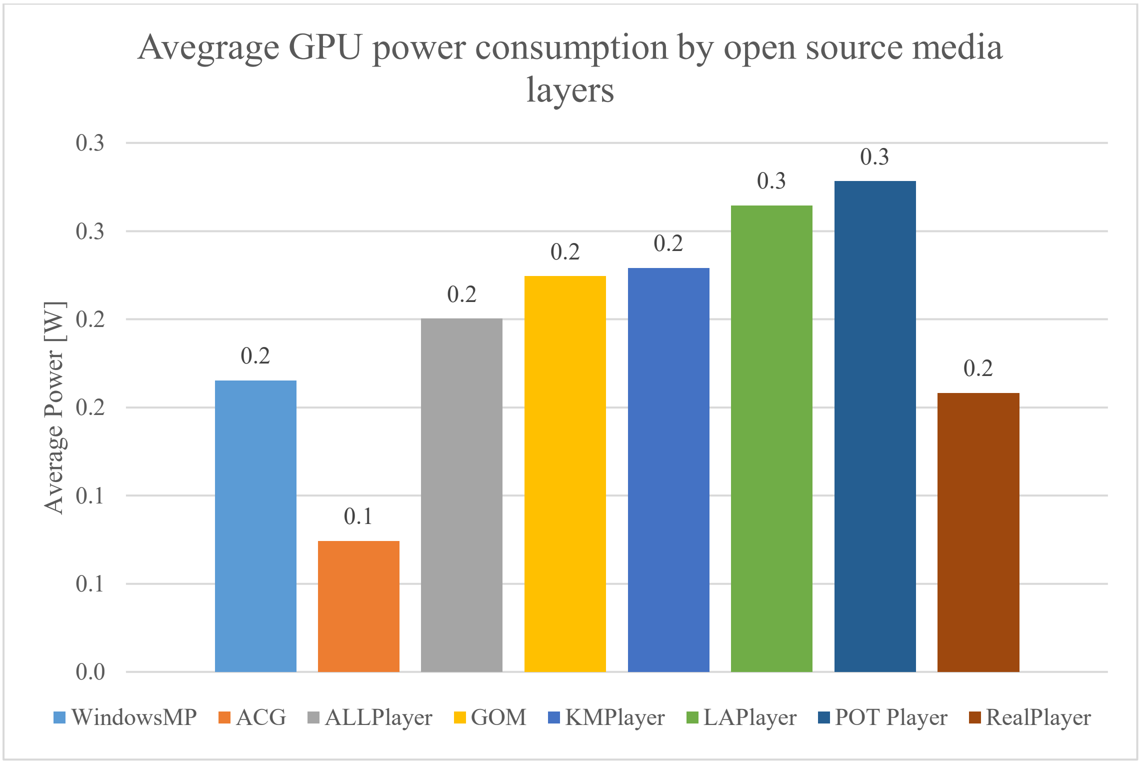

Figure 5.2. Average GPU power consumption of proprietary media players

5.4.2. CPU Power Consumption

Open-source media players: VLC and Kodi showed the lowest CPU power consumption at 3.68 W and 4.03 W, respectively. MPV had the highest CPU power consumption among open-source players at 5.96 W.

Proprietary media players: KMPlayer consumed 6.02 W, and Windows Media Player consumed 5.56 W, reflecting slightly higher CPU power usage compared to their open-source counterparts.

Graph in Figure 5.3 and Figure 5.4: Illustrate the CPU power consumption for each media player, with open-source media players generally exhibiting better efficiency.

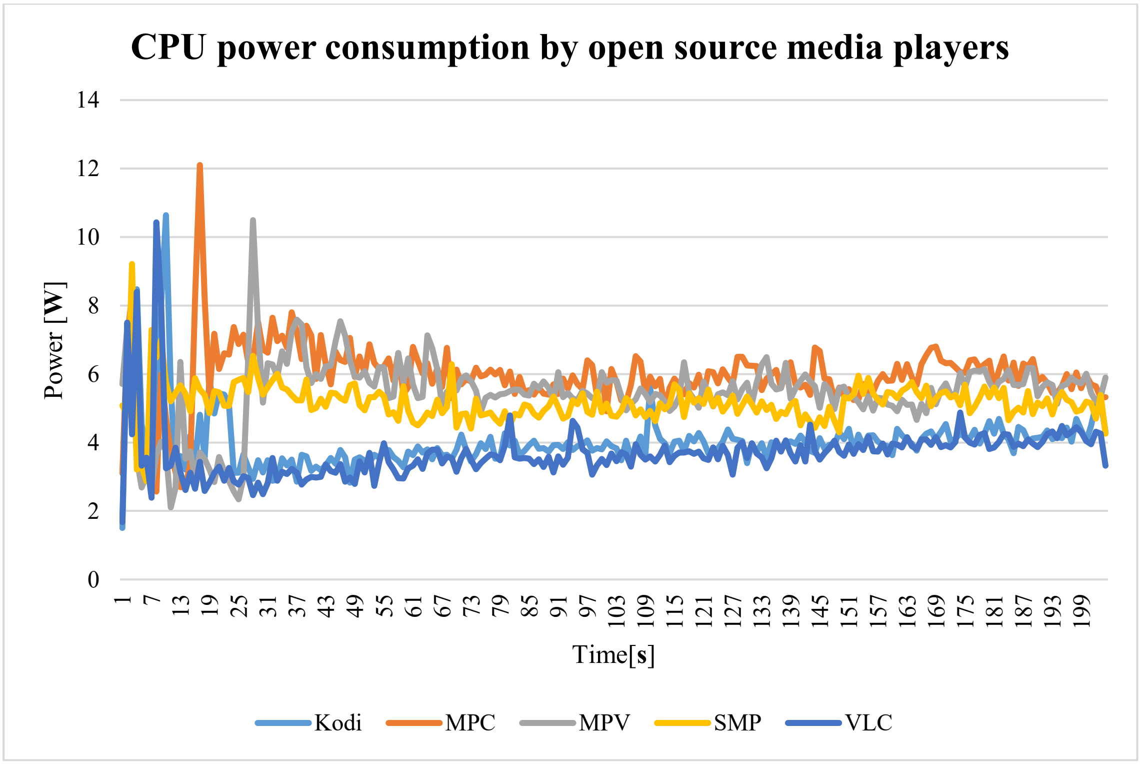

Figure 5.3 power consumption of open-source media players

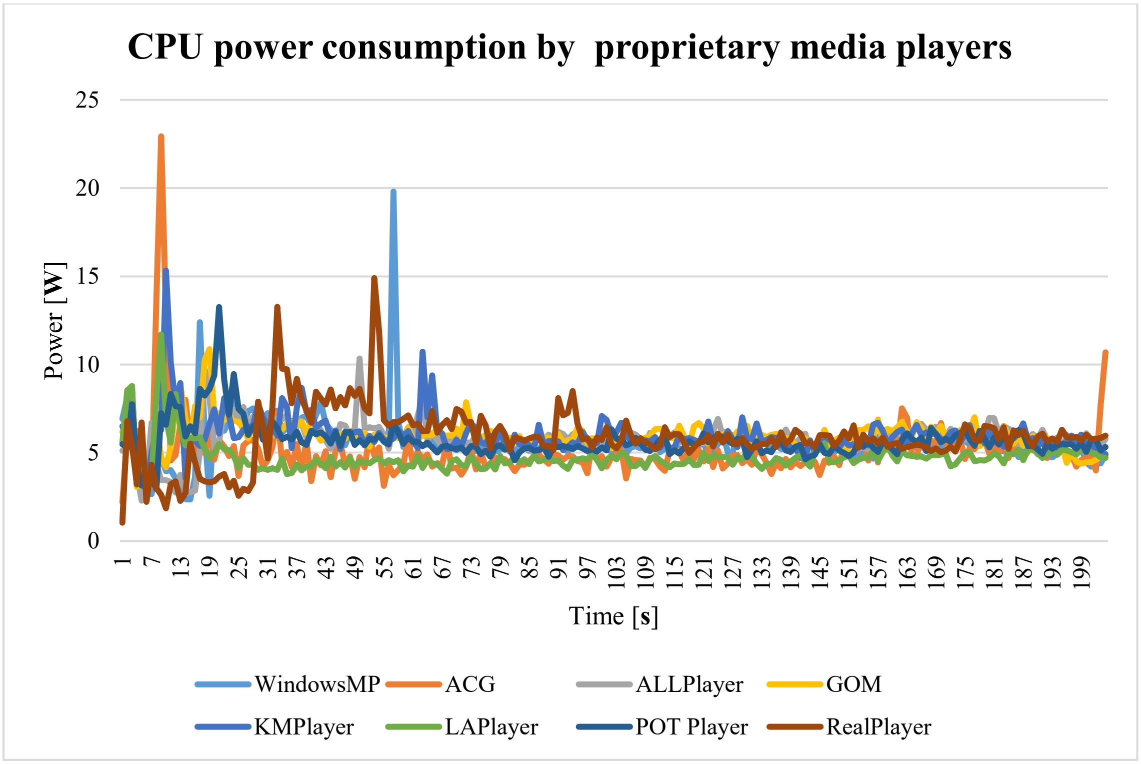

Figure 5.4 Power consumption of proprietary media players

5.4.3. CPU Utilization

- Open-source media players: VLC showed the lowest CPU utilization at 6.41%, while MPC reached 10.75%.
- Proprietary media players: GOM Player had the highest CPU utilization at 17.5%, followed by KMPlayer with 12.69%.

Table 5.1 depict the average CPU utilization percentages and average power consumption, showing that proprietary media players generally consume more CPU resources hence burn more power.

Table 5.1. Average CPU usage and power consumption

Media Player	Average CPU Power consumption	Average CPU usage percentage
Kodi	4.032	7.2%
MPC	5.959	10.8%
MPV	5.445	9.1%
SMP	5.177	9.1%
VLC	3.685	6.4%
WindowsMP	5.568	11.8%
ACG	5.041	8.4%
ALLPlayer	5.859	11.0%
GOM	5.983	17.5%
KMPlayer	6.017	12.7%
LAPlayer	4.745	8.6%
POT Player	5.697	10.3%
RealPlayer	5.927	13.2%

5.4.4. Memory Usage

Proprietary media players generally used more memory compared to open-source players, with KMPlayer consuming the most memory (7562 MB), while VLC consumed around 6705 MB.

If we calculate the average memory power consumption for the media players it can be calculated based on their memory usage, with open-source players consuming less power on average. For example, VLC being the most efficient consumed approximately 1.43 W from memory as it is using the least memory, while KMPlayer being the most resource occupant used around 2.13 W.

Figure 5.5 and Figure 5.6 display the physical memory usage of open-source and proprietary media players respectively, with open-source players consistently using less physical memory.

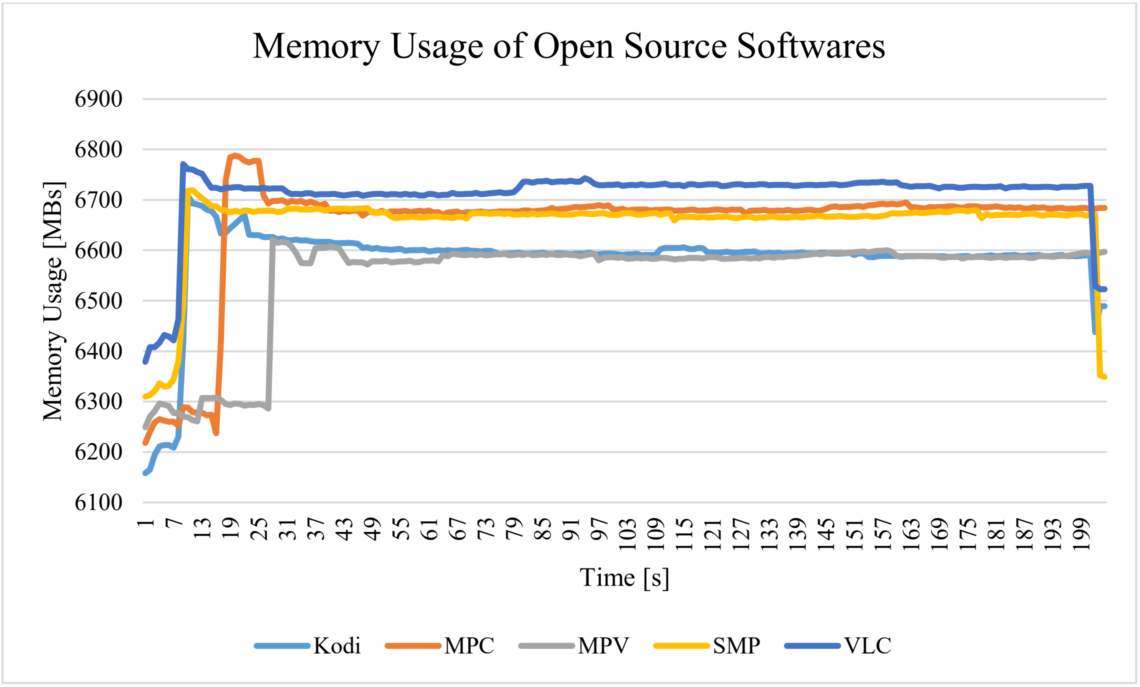

Figure 5.5 Physical memory usage by open-source media players

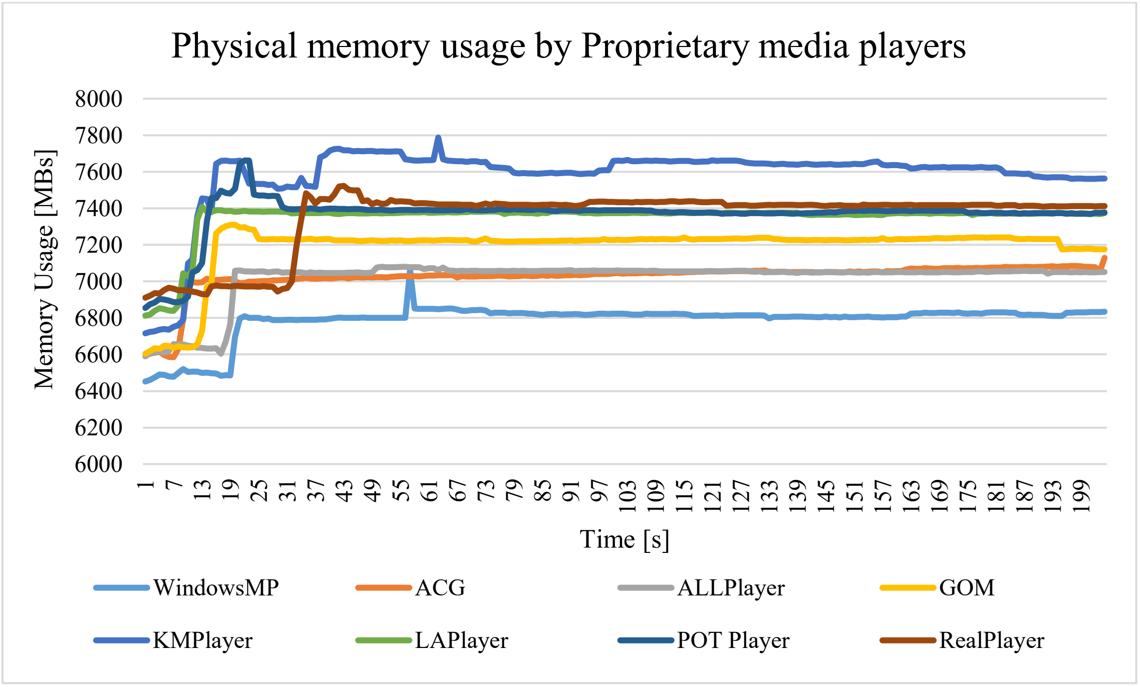

Figure 5.6 Physical memory usage by proprietary media players

5.4.5. Long-term Energy Consumption

Over a year of daily video playback (assuming 2 hours of use per day), using VLC could save approximately 1.80 kWh of energy compared to KMPlayer. While this may seem like a small amount, these savings can add up significantly when considering large-scale deployments or heavy users. Table 5.2 shows the estimated annual energy savings for each media player, based on daily usage scenarios.

Table 5.2 Energy saving of VLC as compared to all other media players

Media Player	**Annual Energy Consumption (kWh)**	**Energy Savings (kWh)**
VLC	2.75	0
Kodi	3.03	0.28
MPV	4.72	1.97
SMP	4.14	1.39
MPC	4.29	1.54
Windows MP	4.19	1.43
KMPlayer	4.56	1.81
GOM Player	4.42	1.67
RealPlayer	4.53	1.78
ALLPlayer	4.35	1.6
LAPlayer	3.67	0.91
POT Player	4.44	1.69

5.4.6. Statistical Analysis

Statistical tests revealed significant differences in memory usage between open-source and proprietary media players ($p < 0.05$), with proprietary players generally consuming more memory. However, no significant differences were observed in GPU power consumption ($p > 0.05$), suggesting that GPU efficiency is relatively consistent across both types of players.

Table 5.3 presents the results of the t-tests for key performance metrics, highlighting where statistically significant differences exist.

Table 5.3 Statistical analysis of significant differences

Performance Metric	p-value	Significance
GPU Power Consumption	0.1264	Not Significant
CPU Power Consumption	0.0855	Marginally Significant
Memory Usage	0.00053	Significant
Memory Power Consumption	0.0015	Significant
CPU Utilization	0.0012	Significant

5.5. Conclusions and future work

This study highlights the significant differences in power consumption and resource usage between open-source and proprietary media players during 4K video playback. Open-source media players, particularly VLC, demonstrated superior efficiency in both GPU and CPU power consumption, making them more suitable for energy-conscious users. Proprietary players, however, often consume more physical memory and CPU resources due to their feature-rich environments, advanced user interfaces, and additional background services. Despite their higher resource demands, proprietary media players provide a more comprehensive media experience. These findings emphasize the importance of selecting media players based on the specific needs of the user, whether prioritizing energy efficiency or enhanced functionality. The study also underscores the role that software choices can play in long-term energy savings, especially when media consumption is a regular part of users' daily activities.

Future research could expand on this study by examining a broader range of media formats and codecs, including those more commonly used in proprietary ecosystems, to explore whether similar trends in power consumption persist.

References

[1] Fortunato, L., & Galassi, M. (2021). The Case for Free and Open Source Software in Research and Scholarship. Philosophical Transactions of the Royal Society A, 379, 2179.

[2] Blancaflor, E. B., & Samonte, S. A. (2023). An Analysis and Comparison of Proprietary and Open-Source Software for Building E-commerce Websites. Journal of Advances in Information Technology, 14(3), 426-431.

[3] Blind, K., Böhm, M., Grzegorzewska, P., Katz, A., Muto, S., Pätsch, S., & Schubert, T. (2021). The impact of Open Source Software and Hardware on technological independence, competitiveness and innovation in the EU economy. Final Study Report. European Commission, Brussels, doi, 10, 430161

[4] Katal, A., Dahiya, S., & Choudhury, T. (2023). Energy efficiency in cloud computing data centers: a survey on software technologies. Cluster Computing, 26(3), 1845-1875.

[5] Pearce, J. M. (2020). Economic Savings for Scientific Free and Open Source Technology: A Review. HardwareX, 8, e00148.

[6] J. Henkel, "Open Source Software from Commercial Firms: Tools, Complements, and Collective Invention," Zeitschrift für Betriebswirtschaft, 2004.

[7] Chesbrough, H. (2023). Measuring the Economic Value of Open Source. Linux Foundation.

[8] Kelty, C. M. (2020). Two bits: The cultural significance of free software. Duke University Press.

[9] Jin, C., Bai, X., Yang, C., Mao, W., & Xu, X. (2020). A review of power consumption models of servers in data centers. applied energy, 265, 114806.

[10] Khajuria, R., Kumar, A., Anand, T., Jain, S., Dayal, P., & Banerjee, S. (2024, April). Efficient Video Streaming on Raspberry Pi 4B: Reducing CPU Utilization Through

Optimization Techniques. In 2024 International Conference on Expert Clouds and Applications (ICOECA) (pp. 942-947). IEEE.

[11] Singh, Aditi, et al. "Power Consumption Predictive Analysis." (2024).

[12] Hrauda W. Virtual binaural acoustics in VLC player: HRTFs and efficient rendering.

[13] Timmerer, C., Wien, M., Yu, L., & Reibman, A. (2021). Special issue on open media compression: Overview, design criteria, and outlook on emerging standards. Proceedings of the IEEE, 109(9), 1423-1434.

[14] Zhou, Zach Zhizhong, and Vidyanand Choudhary. "Impact of competition from open source software on proprietary software." Production and Operations Management 31.2 (2022): 731-742.

[15] L. A. Costa, A. Fontão, and R. Santos, "Investigating proprietary software ecosystem governance and health: An updated and refined perspective," Proceedings of the XVII Brazilian Symposium on Information Systems, pp. 1-8, June 2021.

[16] X. Zhang, Y. Wen, and R. Fan, "Energy-aware cloud computing: Simulation and experimentation of energy-aware resource allocation," IEEE Transactions on Cloud Computing, vol. 10, no. 2, pp. 123-135, 2022.

[17] Bamhdi, A. (2021). Requirements capture and comparative analysis of open source versus proprietary service oriented architecture. Computer Standards & Interfaces, 74, 103468.

[18] J. Le Feuvre, C. Concolato, and J. C. Moissinac, "GPAC: Open Source Multimedia Framework," Proceedings of ACM International Conference on Multimedia, pp. 1009-1012, 2007.

[19] WebM Project, "An open, royalty-free media file format designed for the web," WebM Project. [Online]. Available: https://www.webmproject.org. Accessed: Sept. 9, 2024.

[20] Free Software Foundation, "Licenses for Open Source Projects," FSF. [Online]. Available: https://www.fsf.org. Accessed: Sept. 9, 2024.

[21] WebM Project, "The WebM Project: Open, royalty-free media file format," WebM Project. [Online]. Available: https://www.webmproject.org/about/. Accessed: Sept. 9, 2024.

[22] VideoLAN, "x264, the best H.264/AVC encoder," VideoLAN. [Online]. Available: https://www.videolan.org/developers/x264.html. Accessed: Sept. 9, 2024.

[23] HEVC Advance, "Licensing Information for H.265/HEVC," HEVC Advance. [Online]. Available: https://hevcadvance.com. Accessed: Sept. 9, 2024.

[24] Google Developers, "WebM and the VP8 Codec: Royalty-Free Video," Google Developers. [Online]. Available: https://developers.google.com/web/updates/2010/05/WebM-VP8-Codec. Accessed: Sept. 9, 2024.

[25] CanIUse, "Browser Support for WebM Video Format," CanIUse. [Online]. Available: https://caniuse.com/webm. Accessed: Sept. 9, 2024.

[26] Streaming Media, "VP9 vs H.265: A Comparison of Streaming Performance," Streaming Media, 2024.

[27] FFmpeg, "The Comprehensive Codec Library: VP9 and WebM Support," FFmpeg. [Online]. Available: https://ffmpeg.org. Accessed: Sept. 9, 2024.

[28] RealNetworks, "RealPlayer - Media Player," RealNetworks. [Online]. Available: https://www.real.com. Accessed: Sept. 9, 2024.

[29] BeeConverter, "MP4 to WebM Conversion: A Simple Guide," BeeConverter. [Online]. Available: https://beeconverter.com. Accessed: Sept. 9, 2024.

Chapter 6

Conclusion and Future Work

6.1. Conclusion

This thesis has provided a comprehensive comparison between open-source and proprietary media players, focusing on their energy use, resource utilization, and usability. The research findings show that proprietary media players generally offer less power use due to their optimized integration with hardware and efficient use of codecs. Open-source media players, while offering more flexibility and a wider range of codec support, tend to consume more CPU power unless carefully optimized, particularly in environments where hardware acceleration is not fully utilized. One of the primary reasons proprietary media players consume less power is their tight integration with hardware manufacturers and the use of proprietary codecs such as H.264 and H.265. These codecs are designed to minimize CPU usage by offloading video processing tasks to the GPU, which is more energy-efficient for high-demand tasks such as 4K video playback. Additionally, proprietary media players often come with regular updates and professional support that ensure ongoing optimization for both performance and energy efficiency. This combination of hardware optimization and software support leads to consistently lower power consumption during extended usage, making proprietary players the preferred choice in professional environments where performance and sustainability are critical.

Open-source media players, such as VLC and MPV, offer a different set of advantages. Their flexibility, broad codec support, and absence of licensing fees make them attractive to users who need a cost-effective solution with customization potential. However, this flexibility often comes at the expense of higher CPU usage and energy consumption, particularly when hardware acceleration is not fully supported by the system's drivers. This is especially true in Linux-based environments, where driver support for open-source media players can be less robust compared to proprietary players on Windows platforms. Despite this, open-source media players demonstrate significant potential for energy savings when paired with modern codecs like VP9 and AV1, which are optimized for efficient video streaming and can reduce power consumption by offering better compression and resource management.

Moreover, open-source media players have the advantage of community-driven development, which allows for rapid updates and optimizations. While these updates may not always be as structured or frequent as those provided for proprietary players, they offer the potential for long-term improvements in energy efficiency. As more users and developers contribute to the open-source community, optimizations in codecs, hardware acceleration, and resource management can lead to reductions in energy consumption. This makes open-source players a viable option for users who are willing to invest time in customizing and optimizing their media playback settings.

One of the most significant findings of this research is that proprietary media players consistently demonstrate better power consumption management, especially in high-performance scenarios such as 4K or raw video playback. This advantage is largely due to the close integration of proprietary software with hardware platforms, where media players can efficiently offload video processing to the GPU and reduce CPU load. Proprietary media players also benefit from highly optimized proprietary codecs that further reduce energy usage by compressing video files in a way that requires fewer system resources during playback. As a result, proprietary media players are generally more suitable for environments where long-term energy savings and performance are prioritized, such as in professional video editing or streaming services.

On the other hand, open-source media players offer greater flexibility and customization potential, which can be leveraged to improve energy efficiency over time. Users with technical expertise can optimize open-source players to reduce background processes, disable unnecessary features, and adjust settings to suit specific hardware configurations. This adaptability, combined with the support for modern, royalty-free codecs like AV1 and VP9, allows open-source media players to achieve competitive levels of energy efficiency in certain scenarios. While they may not match the energy performance of proprietary players out of the box, open-source players can become more efficient with user-driven adjustments and community updates.

The long-term sustainability of open-source media players also deserves attention. While they may consume more energy upfront due to less efficient resource management, the absence of licensing fees and the ability to tailor the software to specific needs make them a cost-effective and flexible alternative to proprietary solutions. This is particularly important in environments where cost is a major factor, such as educational institutions or non-profit organizations. Over time, open-source players may continue to evolve through community efforts, leading to further improvements in energy efficiency and long-term sustainability.

6.2. Research Contributions

- Proprietary media players typically consume less power during high-resolution video playback compared to their open-source counterparts. This advantage is attributed to superior hardware optimization and efficient resource utilization. While open-source media players are versatile and support a wide variety of media formats, they often demand more CPU resources, particularly when hardware acceleration is not fully utilized. Nevertheless, in certain cases, such as with newer open codecs like VP9 or AV1, open-source players can achieve greater efficiency.

- Hardware acceleration plays a crucial role in reducing power consumption for media players, especially on platforms like Windows, where proprietary software benefits from highly optimized drivers. Proprietary media players effectively utilize the GPU to offload video processing tasks, thereby reducing CPU energy usage. Conversely, open-source players may face challenges with driver support, particularly on Linux systems, resulting in higher energy consumption due to increased reliance on the CPU.
- Open-source media players generally demonstrate lower GPU and CPU power consumption when playing raw, uncompressed video formats, which are computationally demanding. However, proprietary players often leverage advanced, optimized codecs such as H.264 and H.265, enabling smoother playback by efficiently compressing high-quality video data and reducing processing loads. Open-source players, on the other hand, excel in handling open formats like .webm by capitalizing on their modular design and community-driven innovation.
- Over time, proprietary media players are more likely to deliver sustained energy savings due to regular updates and improved resource management. Nonetheless, open-source media players can also achieve long-term efficiency, particularly when users fine-tune settings and benefit from community-driven updates. Although they may initially consume more energy, their adaptability and cost-effectiveness can make them a sustainable option in specific contexts.

6.3. Future Work

Given the findings of this thesis, future research should focus on the role of new and emerging codecs, such as VVC (H.266) and AV1, in reducing power consumption during media playback. These newer codecs are designed for more efficient compression, potentially offering even greater energy savings. As hardware manufacturers develop accelerators to support these codecs, it would be valuable to study how both open-source and proprietary media players utilize these advancements. Additionally, the impact of machine learning and artificial intelligence on resource management within media players could be explored, particularly in terms of predictive optimization for energy efficiency.

This thesis focused on evaluating the performance of media players on Windows and Linux platforms. However, additional research is necessary to investigate their performance on other platforms, such as macOS, Android, and iOS. These platforms present unique hardware architectures and operating systems, which introduce distinct optimization challenges and opportunities. Analyzing how media players can be optimized for cross-platform functionality, particularly in mobile and embedded environments, would offer a broader understanding of their energy efficiency across varied usage scenarios.

Despite the insights provided, this research has certain limitations. First, the findings are specific to Windows and Linux systems and may not fully generalize to other platforms with different software and hardware ecosystems. Additionally, the scope of this study did not include mobile and embedded environments, which are increasingly significant for media consumption. The absence of standardized testing environments across platforms may also introduce variability in performance comparisons, limiting the ability to draw direct conclusions about cross-platform efficiency.

As mobile devices play an increasingly significant role in media consumption, assessing the energy efficiency of media players on these platforms is essential. Given their limited battery capacity, mobile devices require highly optimized software to conserve power during activities like video streaming. Future studies should examine how media players manage resource-intensive tasks, such as 4K video streaming on mobile devices, and the impact of energy-efficient codecs on battery performance. Additionally, exploring the energy consumption differences between native mobile applications and web-based media players could provide valuable insights. Expanding the analysis to include a wider range of media players and device types would address the current study's limitations and contribute to a more comprehensive understanding of energy efficiency in media playback.

Open-source media players are highly customizable, allowing users to optimize their settings for specific hardware configurations. Future research could explore how user-driven customization impacts the energy efficiency of open-source media players, particularly in professional environments where media playback is continuous or resource-intensive. Additionally, research could focus on developing tools and guidelines that help users optimize open-source players for reduced power consumption without sacrificing performance.

While this thesis focused primarily on immediate power consumption during video playback, future studies could examine the long-term energy savings associated with using open-source versus proprietary media players. By considering factors such as update frequency, software longevity, and cumulative power usage, researchers could provide a clearer picture of the long-term sustainability of media player choices. Such studies could also investigate how different usage patterns—such as occasional vs. continuous media playback—affect overall energy consumption over time.

With the rise of cloud-based media streaming services, it would be valuable to explore how media players perform in environments where the processing power is partially or fully offloaded to the cloud. Research could focus on the energy efficiency of local media players versus cloud-based playback systems, examining the trade-offs between local and remote resource consumption. This would be particularly relevant for platforms that offer both local media playback and cloud streaming, such as Netflix or YouTube, where energy efficiency is critical for both service providers and users.

The impact of network traffic on power consumption is a critical factor, especially in scenarios involving streaming media. High-definition streaming or buffering under poor network conditions can significantly increase energy usage due to prolonged activity of network interfaces and additional processing by the media player. Future research should investigate how different streaming protocols and adaptive bitrate technologies influence power consumption. User interactions, such as pausing, fast-forwarding, or seeking within a video, can also affect power consumption. These actions often lead to increased CPU and GPU activity as the media player recalibrates playback. Analyzing these scenarios would provide a deeper understanding of the energy efficiency of media players under real-world usage conditions.

Articles in Refereed Publications

- **Ahmed, Afzal**, Mohammad Tariq Iqbal, and Mohsin Jamil. 2024. “A Comparative Analysis of Power Consumption While Using Open-Source and Proprietary Media Players”. European Journal of Electrical Engineering and Computer Science 8 (5):1-5. https://doi.org/10.24018/ejece.2024.8.5.649.
- **Ahmed, Afzal**, Mohammad Tariq Iqbal, and Mohsin Jamil. " A Comparative Study of CPU and GPU Power Consumption while using Open-Source and Proprietary Media Players" Accepted in *European Journal of Information Technologies and Computer Science* (2024)
- **Ahmed, Afzal**, Mohammad Tariq Iqbal, and Mohsin Jamil. " Comparative Analysis of Power Consumption and Resource Utilization in Open-Source and Proprietary Media Players while using Raw videos" Accepted in *European Journal of Information Technologies and Computer Science* (2024)

Regional Conference

- **Ahmed, Afzal**, Mohammad Tariq Iqbal, and Mohsin Jamil. " A Comparative Analysis of Media Players Power Consumption on Windows 11 and Ubuntu 24.04.1" Accepted in *the 33rd Annual Newfoundland Electrical and Computer Engineering Conference (NECEC), 2024.*