European Journal of Electrical Engineering and Computer Science
Vol 8 | Issue 5 | September 2024
ISSN 2736-5751

RESEARCH ARTICLE

A Comparative Analysis of Power Consumption While Using Open-Source and Proprietary Media Players

Afzal Ahmed*, Mohammad Tariq Iqbal, and Mohsin Jamil

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.

Keywords: Media players, open source, power consumption, proprietary.

Submitted: August 23, 2024
Published: September 18, 2024

10.24018/ejece.2024.8.5.649

Department of Electrical and Computer Engineering, Faculty of Engineering and Applied Science, Memorial University of Newfoundland, Canada.

****Corresponding Author:***
e-mail: afzala@mun.ca

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. Literature Review

In recent years, research has increasingly focused on the methodologies for measuring and analyzing software power consumption. Kumar and Srinivas [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 Proedrou [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. Mohapatra *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 Ournani *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. Turkkan [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 Harding [14] investigated the power consumption of various word-processing features, revealing that complex formatting and image-processing tasks contribute heavily to power drawing. 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. Donato [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.

3. Experiment

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 the CPU power package, GT core power, percentage of CPU usage, and physical memory consumption in megabytes (MBs).

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.

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*: 16 GB
- *Operating system*: Microsoft Windows 11 Home
- *OS Version*: 10.0.22631 Build 22631

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.

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, the file size was 791 MBs 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 seconds.

3.5. *Data Collection*

During the 3-minute and 20-second playback period, HWiNFO recorded the relevant metrics in real-time with the frequency of 1000 ms setup in the HWiNFO.

The data for each media player was logged and exported for analysis.

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.

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.

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.

4. Results

A descriptive statistics analysis was performed to analyze the results.

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 I.

The Graph in Fig. 1 depicts the trends of power consumption in the proprietary software.

Power consumption trends in open-source media players can be seen in Fig. 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.

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.

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

TABLE I: 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
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
ALL player	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

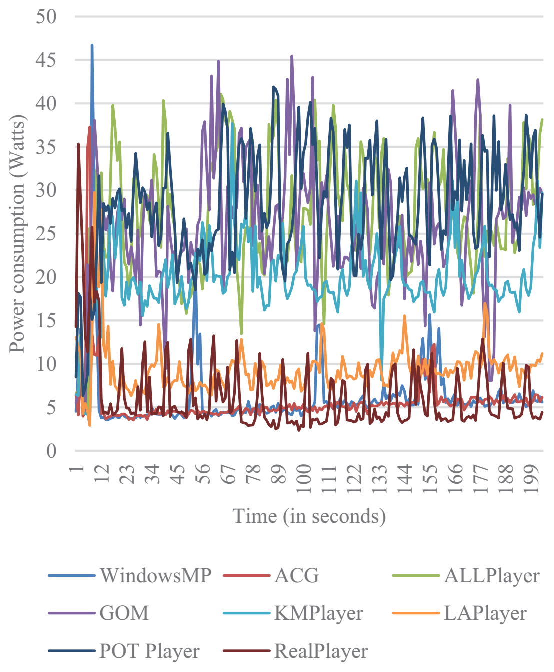

Fig. 1. Comparison of power consumption (W) between different proprietary media players.

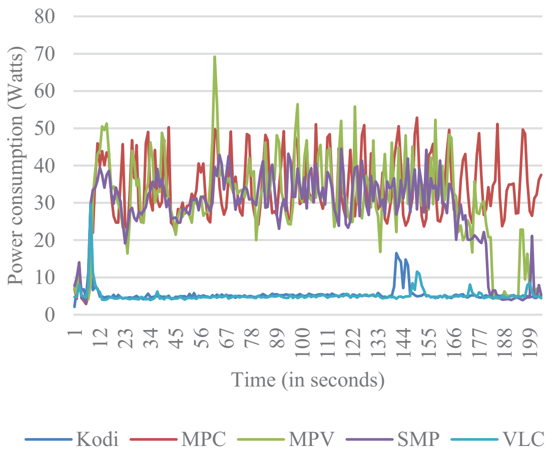

Fig. 2. Comparison of power consumption (W) between different open-source media players.

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.

5. Conclusions and Future Work

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 that 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 a raw video file or open-source video file format is used. All these issues need to be investigated to get a comprehensive understanding of the factors affecting power consumption efficiency.

Conflict of Interest

Authors declare that they do not have any conflict of interest.

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 open-source 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.