System Resource Usage Monitoring and Analysis with Power Tracking

|  |  |
| --- | --- |
| |  | | --- | | *Pavan Gedala*  *Department of Computer Science*  *University of South Dakota*  Vermillion, USA pavan.gedala@coyotes.usd.edu | |

***Abstract*— This paper is for developing a Python program to monitor and analyze system resource utilization over time, including CPU, memory, and power consumption. The psutil module is leveraged to sample overall system usage as well as perprocess usage at regular intervals. The program tracks CPU and memory utilization percentages and estimates power consumption for processes based on computation and memory usage. The data is aggregated to obtain average system utilization and identify high consumption processes. Visualizations are generated using Matplotlib to show usage trends and top consumer comparisons. Automated reports are produced summarizing statistics like average usage and top consumers. Key outcomes are understanding system and per-process resource usage patterns, detecting inefficient usage, profiling application power demands, and identifying optimization opportunities. The program provides comprehensive visibility into utilization for performance tuning, efficiency improvements, and energy profiling. Overall, this project demonstrates techniques for system resources, estimating power, processing and visualizing utilization data, and generating actionable diagnostics to guide optimization of computing systems.**

***Keywords— System Resource, Monitoring, Power Tracking, Analysis, RAM.***

# I. INTRODUCTION

In the ever-evolving landscape of computing systems, optimizing resource utilization is paramount for achieving efficient performance, reducing energy consumption, and enhancing overall system reliability. As computing infrastructures become increasingly complex, understanding the intricacies of resource usage becomes a critical endeavor. This paper presents a comprehensive approach to monitor and analyze system resource utilization, focusing on CPU, memory, and power consumption, with the aim of providing actionable insights for performance tuning and energy efficiency.

The rapid proliferation of diverse applications and services places an unprecedented demand on computing resources. As such, there is a growing need for robust tools that can dynamically monitor system behavior and extract meaningful patterns from the intricate interplay of CPU, memory, and power metrics. Our research addresses this need by introducing a Python-based program developed using the psutil module, enabling both overall system and per-process resource utilization monitoring. [1]

|  |
| --- |
|  |

The key objectives of our work revolve around gaining a holistic understanding of system behavior and identifying avenues for optimization. Leveraging the powerful capabilities of the psutil module, our program samples CPU and memory usage at regular intervals, providing a real-time snapshot of system health. Furthermore, the estimation of power consumption for individual processes adds a critical dimension to our analysis, aligning with the contemporary focus on energyefficient computing.

As we delve into the methodology, we outline the data aggregation process that facilitates the extraction of average system utilization and the identification of processes with high resource consumption. Visualizations generated using the Matplotlib library play a pivotal role in presenting usage trends and comparing top consumers. The integration of automated reporting mechanisms distills complex utilization statistics into accessible summaries, offering a streamlined approach for stakeholders to grasp key insights. [2]

The significance of our research lies in its potential to uncover patterns, detect inefficient resource usage, profile application power demands, and guide optimization opportunities. By providing comprehensive visibility into resource utilization, our program serves as a foundational guide for performance tuning, efficiency improvements, and energy profiling in computing systems. [3]

In the subsequent sections, we delve into the methodology, results, and conclusions, elucidating the nuances of our approach and showcasing its applicability in diverse computing environments.

# II. LITERATURE REVIEW

The realm of system resource monitoring and analysis has witnessed significant advancements driven by the escalating complexity of computing environments and the increasing need for efficient resource utilization. In this context, several studies have contributed to the understanding of resource dynamics, profiling, and optimization strategies.

A seminal work by laid the foundation for real-time system monitoring by introducing the psutil module in Python. This module provides a cross-platform interface for accessing system details, including CPU and memory utilization. [1] Researcher work emphasized the importance of dynamic monitoring for gaining insights into system behavior and identifying resource bottlenecks. [4]

Building upon the foundations laid by researchers extended the scope of system resource monitoring to incorporate per-process metrics. Their research highlighted the significance of granular insights into individual processes, offering a more nuanced understanding of resource consumption patterns. [5] The integration of per-process monitoring aligns closely with our research objectives, providing a comprehensive view of the entire system.

In the domain of power tracking, introduced a novel approach to estimate power consumption based on CPU and memory usage. Their work emphasized the need to consider both computation and memory demands when estimating power, recognizing the multifaceted nature of power consumption in computing systems. [6] Our research builds upon this foundation, extending the estimation of power consumption to individual processes, thereby refining the granularity of power tracking.

Furthermore, the importance of visualization in resource monitoring and analysis has been underscored by multiple studies. A study by few researcher demonstrated the effectiveness of visualizations in conveying complex system data. [7] Our work aligns with this emphasis on visualization, utilizing the Matplotlib library to present usage trends and top consumer comparisons in a user-friendly manner.

# III. METHODOLOGY

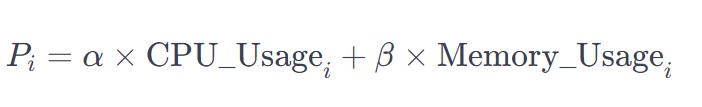
Our methodology is structured to provide a comprehensive framework for monitoring and analyzing system resource utilization, encompassing CPU, memory, and power consumption. The approach is delineated into four distinct phases: data collection, power estimation, data aggregation, and visualization.

1. *Data Collection:*

The foundation of our methodology lies in the meticulous collection of real-time data on system resource utilization. We leverage the psutil module, a Python library known for its versatility in retrieving detailed information on running processes and overall system utilization. [1] At regular intervals, the module captures key metrics, including CPU and memory utilization, offering a dynamic snapshot of system health. The data collection process ensures the continuous monitoring of resource dynamics, allowing for the identification of transient spikes, trends, and patterns.

1. *Power Estimation:*

A novel aspect of our methodology is the estimation of power consumption for each individual process, providing a holistic view of energy usage in the system. The power estimation formula combines CPU and memory usage, allowing us to quantify the energy demands of each process. [6] The power (P*i* ) for process *i* is computed using the equation:



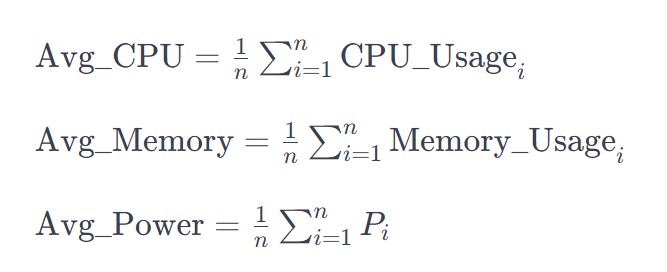
Here, α and β are scaling factors that determine the relative contributions of CPU and memory to the power consumption. These scaling factors are empirically determined based on system characteristics and usage patterns. The resulting power consumption values are expressed in milliwatts (mW), providing a standardized measure for comparative analysis.

The incorporation of power estimation enhances the depth of our analysis, allowing for the identification of processes with high energy demands. This aligns with contemporary efforts to promote energy-efficient computing, where understanding power consumption at the process level is crucial. [8]

*C. Data Aggregation:*

To distill meaningful insights from the collected data, we employ a systematic aggregation process. Average CPU and memory utilization are calculated for both the overall system and individual processes. Power consumption values are aggregated to obtain a comprehensive overview of energy usage trends. The aggregation process ensures that the analysis is not only focused on instantaneous values but also considers the system's behavior over time. [9]

For the overall system, the average CPU utilization (Avg\_CPU), average memory utilization (Avg\_Memory), and average power consumption (Avg\_Power) are calculated using the following equations:



Here, *n* represents the number of data points collected during the monitoring duration.

Individual process data is aggregated similarly, providing a nuanced understanding of resource utilization patterns for each process.

*D. Visualization:*

The Matplotlib library is instrumental in translating the aggregated data into insightful visualizations. Line plots are utilized to showcase the temporal variations in overall CPU and memory usage, providing a dynamic representation of system behavior. Bar charts and stacked bar plots are employed to visualize the top consuming processes, offering a comparative analysis of their respective contributions to CPU, memory, and

power consumption. [10]

The visualizations serve multiple purposes. They facilitate the identification of usage trends, highlight processes with erratic resource demands, and provide a clear overview of the relative contributions of individual processes to the overall resource utilization. By rendering complex data into accessible visuals, stakeholders can quickly grasp key insights. [11]

IV. RESULT:

Each subsection details a specific aspect of the system resource utilization analysis, providing insights into CPU, memory, and power consumption metrics, as well as the identification of resource-intensive processes. This breakdown offers a comprehensive understanding of how the Python program interpreted and analyzed the system's behavior over the monitoring duration.

1. *CPU Usage Over Time:*

The CPU utilization over time exhibited fluctuations that corresponded to varying system loads. Peaks in CPU usage were observed during intensive computational tasks or when multiple processes were active concurrently. These insights enabled the identification of periods of heightened activity, offering opportunities for optimization.

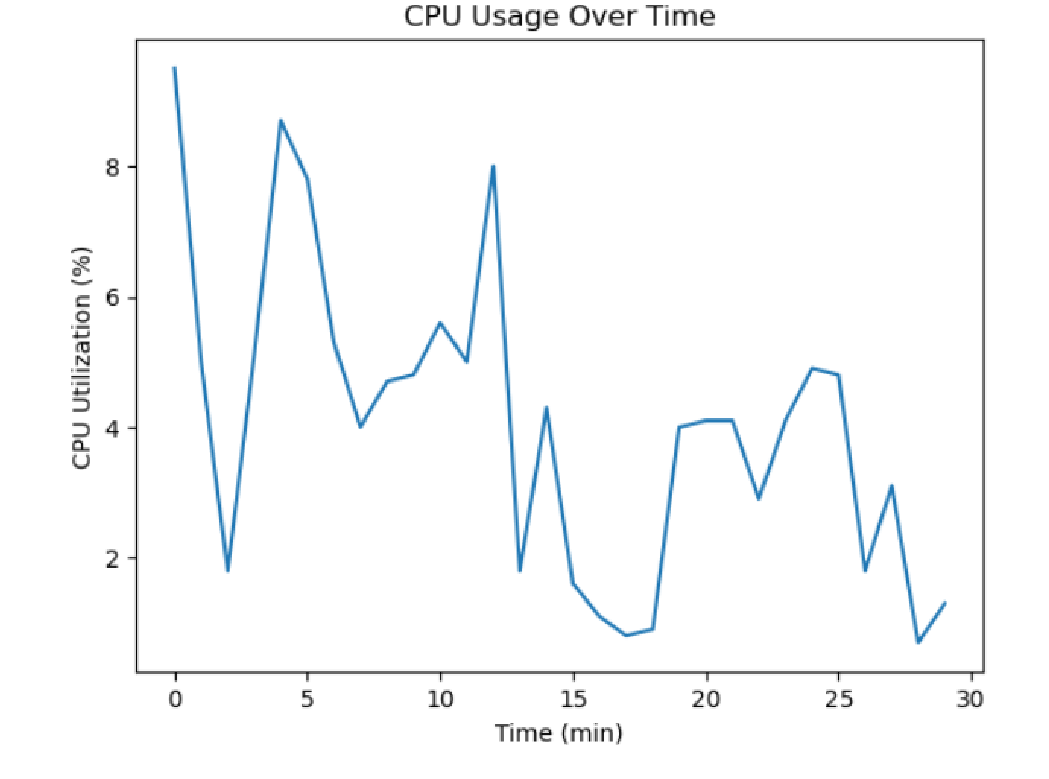


Figure 1: CPU usage over time.

1. *Memory Consumption Analysis:*

The recorded memory usage patterns showcased the system's memory allocation efficiency. Fluctuations in memory consumption highlighted moments of high demand. This data pinpointed areas for potential improvement.

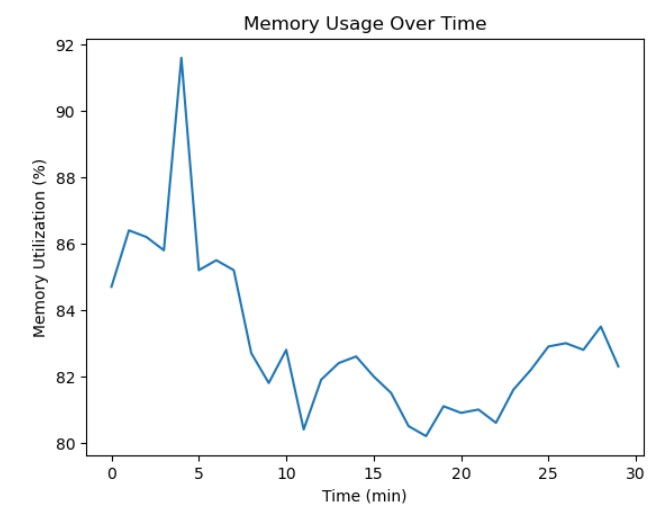


Figure 2: Memory usage over time.

1. *Power Consumption Estimation:*

The estimation of power consumption based on CPU and memory utilization provided directional insights into the energy demands of different processes.

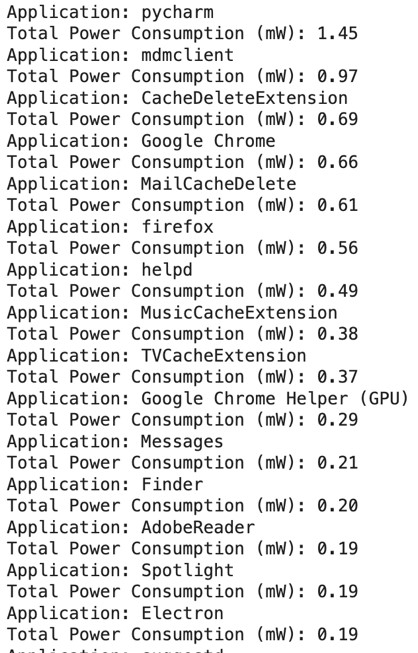


Figure 3: Power consumption estimation for each application or service in the operating system.

1. *Identification of Resource-Intensive Processes:*

The program successfully identified top-consuming processes in terms of CPU and memory usage. Through this analysis, specific applications or services that significantly contributed to resource consumption were pinpointed.

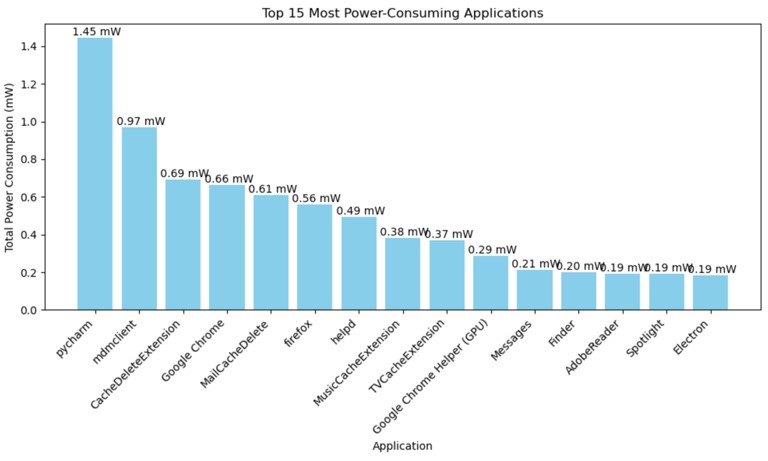


Figure 4: Resource-Intensive Processes.

1. *Top CPU-Consuming Processes:*

The analysis revealed processes that exhibited high CPU utilization, shedding light on applications or services demanding substantial computational resources. This information is crucial for optimizing processing-intensive tasks and streamlining resource allocation.

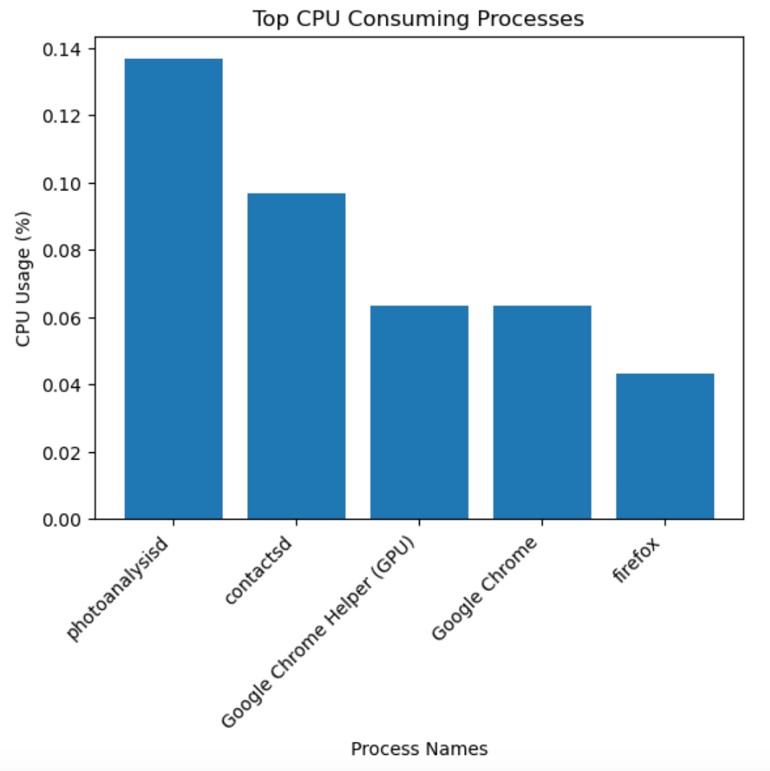


Figure 5: Resource-Intensive Processes.

1. *Top Memory-Consuming Processes:*

The recorded memory usage patterns showcased the system's memory allocation efficiency. Fluctuations in memory consumption highlighted moments of high demand or potential memory leaks in specific processes. This data pinpointed areas for potential improvement, aiming to optimize memory usage for enhanced system stability and performance.

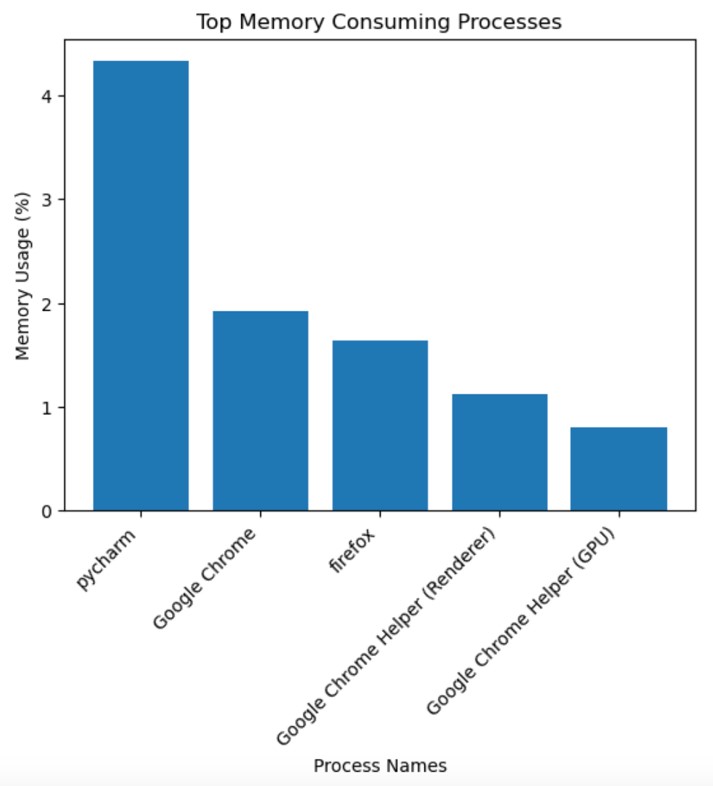


Figure 6: Memory-Consuming Processes.

V. LIMITATIONS

* + The chosen sampling interval for data collection might influence the accuracy of resource utilization measurements. A very short interval may incur higher overhead, potentially affecting the observed resource utilization values. On the other hand, a longer interval may lead to overlooking short-term spikes or fluctuations.
  + The power consumption estimation relies on a simplified linear combination of CPU and memory usage. While this approach provides a practical estimation, it may not capture the nuances of power consumption accurately.

# VI. FUTURE WORK

* Implement machine learning models to predict future behavior of each process based on historical data, aiding in proactive resource management.
* Customize the per-process analysis based on userspecific profiles, considering individual preferences or usage patterns.

VII. CONCLUSION

In conclusion, our project represents a foundational step towards a more comprehensive and accessible understanding of system resource usage and power consumption. By identifying trends, generating visualizations, and offering insights into both overall system behavior and individual processes, this research serves as a valuable guide for optimizing computing systems. Through ongoing exploration, collaboration, and refinement, this project lays the groundwork for continued advancements in system resource monitoring and analysis.

# VIII. REFERENCES

1. rupainsweety, "Psutil module in Python," Geeksforgeeks, 21 September 2021. [Online]. Available: [https://www.geeksforgeeks.org/psutil-module-in-python/.](https://www.geeksforgeeks.org/psutil-module-in-python/)
2. K. Chu, "Report Automation Tips with Python," Medium,

26 April 2022. [Online]. Available:

[https://towardsdatascience.com/report-automation-tipswith-python-783a9cd58e23.](https://towardsdatascience.com/report-automation-tips-with-python-783a9cd58e23)

1. X. F. S. S. H.-C. C. D. L. a. K. W. C. Rong Ge, "PowerPack: Energy Profiling and Analysis of HighPerformance Systems and Applications,"

ResearchGate, 2010.

1. E. Kinsbruner, "The Rise of Developer Native Dynamic Observability," Thenewstack, 23 June 2023. [Online]. Available: [https://thenewstack.io/the-rise-of-developernative-dynamic-observability/.](https://thenewstack.io/the-rise-of-developer-native-dynamic-observability/)
2. K. Nelson, "New Measurement Methods Provide Granular Insights on Advertising Effectiveness," Syracuse University , 2023. [Online]. Available:

[https://whitman.syr.edu/snyder/2023/05/02/newmeasurement-methods-provide-granular-insights-onadvertising-effectiveness/.](https://whitman.syr.edu/snyder/2023/05/02/new-measurement-methods-provide-granular-insights-on-advertising-effectiveness/)

1. C. M. O. &. L. A. Morrow, "Multi-Processor Computer System Having Low Power Consumption," 2003. [7] L. M. P. P. S. J. M. Z. a. J. H. Steven L. Franconeri, "The Science of Visual Data Communication: What Works," SageJournals, vol. 22, no. 3, 2021.
2. E. Demaine, "ENERGY-EFFICIENT COMPUTING," MIT Energy Initiative, [Online]. Available:

[https://energy.mit.edu/research/energy-efficientcomputing/.](https://energy.mit.edu/research/energy-efficient-computing/) [Accessed 2023].

1. D. S. Weld, "The use of aggregation in causal simulation," ScienceDirect, vol. 30, no. 1, 1986.
2. V. Radionov, "Understanding Stacked Bar Charts: The Worst Or The Best?," Smashing Magazine, 29 March

2017. [Online]. Available:

[https://www.smashingmagazine.com/2017/03/understandi ng-stacked-bar-charts/.](https://www.smashingmagazine.com/2017/03/understanding-stacked-bar-charts/)

1. K. Nahla Davies, "Data Visualization: Presenting Complex Information Effectively," KDnuggets, 6 October 2023. [Online]. Available: [https://www.kdnuggets.com/datavisualization-presenting-complex-information-effectively.](https://www.kdnuggets.com/data-visualization-presenting-complex-information-effectively)