Tracking CPU Energy Consumption of Applications: Intel RAPL for Process-Level Power Monitoring in Linux

Bryan Jonay Liegsalz
University of Applied Sciences Munich
Munich, Germany
bryan.liegsalz@hm.edu

***Abstract*—The rapid expansion of digital services has led to a marked increase in the energy consumption of data centers, with projections indicating a near doubling by 2026. This paper addresses the challenge of accurately measuring and managing the energy usage of individual software processes. Instead of assessing the total energy consumption of entire servers, we focus on converting the CPU usage of specific processes into precise measurements in watts and kilowatt-hours. Utilizing Intel's Running Average Power Limit (RAPL) technology, we present a method that enables real-time, process-level monitoring of energy consumption. This detailed approach provides valuable insights into how individual software activities contribute to overall energy usage, facilitating targeted optimizations that can significantly reduce operational costs and environmental impact.**

***Index Terms*—Power Monitoring; Intel RAPL; Green IT**

I. Introduction

The surge in demand for digital services has significantly escalated the energy consumption of data centers. According to a report by the International Energy Agency (IEA) [1], data centers consumed an estimated 460 terawatt-hours (TWh) in 2022. This figure is expected to nearly double by 2026. For context, the electricity consumption of Germany in 2023 was 457 TWh, as noted by the Fraunhofer Institute [2]. To accurately estimate the energy consumption produced by a company, it is crucial to understand the energy usage generated by its servers.

This paper addresses the critical challenge of accurately tracking and managing the energy consumption of specific software processes within CPU usage. Instead of calculating the total energy consumption of an entire server, our focus is on quantifying and converting the CPU usage of individual processes into precise measurements in watts and kilowatt-hours. By examining the behavior of these processes over time, we gain valuable insights into how software activities impact overall energy usage. This detailed analysis allows us to explore strategies for optimizing and reducing energy consumption.

Ultimately, our goal is to support the transition towards a greener and more efficient digital future, fostering improvements that benefit both the environment and operational costs.

II. Understanding CPU Power Consumption

A. Definitions and Concepts: Power, Energy and Efficiency

To measure energy consumption in software processes, it is essential to understand the fundamental concepts of power, energy and efficiency.

Power is the rate at which energy is consumed or produced, measured in watts (W). In computing, power consumption refers to the amount of electrical power used by components such as the CPU. Tools such as RAPL deliver the power in Joules per second (J/s). 1 Joule per Second equals 1 Watt [3].

Energy is the total amount of power consumed over time, measured in joules (J) or kilowatt-hours (kWh). One kilowatt-hour equals 3.6 million joules [3]. For software applications, energy consumption represents the cumulative power used during their execution:

$$E = \int_{t_1}^{t_2} P(t)\,dt$$

where:

- E is the energy consumed over the time interval $[t_1, t_2]$.
- $P(t)$ is the power consumption function for a time point t,
- The integral represents the cumulative power used over the period from t_1 to t_2, which results in energy.

Efficiency in computing refers to how effectively a system performs its tasks while minimizing energy usage. This can involve optimizing software algorithms, reducing the power draw of hardware, or both.

Understanding these concepts is needed in order to evaluate and improve the energy efficiency of software processes. A process that consumes less energy to perform the same task is considered more energy-efficient.

III. Current approaches

A review of the literature reveals several existing implementations designed to track and optimize energy consumption at the software process level. These approaches employ diverse methods and technologies to address the challenges of accurately measuring and managing power usage within computing environments. Each method offers distinct advantages and limitations, contributing valuable insights into the ongoing efforts to enhance energy efficiency in digital systems.

A. PowerScope

- **Approach**: Samples energy consumption by process using a multimeter and stores data on a separate computer for offline analysis [4].
- **Advantages**: High precision in mapping energy consumption to specific processes and procedures.
- **Disadvantages**: Requires additional hardware (multimeter). Does not provide real-time energy consumption values.

B. Joulemeter

- **Approach**: Monitors resource usage such as CPU, disk I/O and display brightness to estimate energy consumption [5].
- **Advantages**: No need for additional hardware. Provides estimates based on machine-specific power models.
- **Disadvantages**: Requires calibration on each hardware configuration, limiting flexibility. Estimates are based on model parameters which may not be precise for all workloads.

C. PowerAPI

- **Approach**: PowerAPI monitors and analyzes power consumption of computer components using sensors, providing detailed reports to help optimize energy use [6].
- **Advantages**: Can estimate energy consumption at both process and method levels.
- **Disadvantages**: In environments with diverse hardware architectures, PowerAPI might struggle to provide consistent and accurate monitoring.

IV. Tracking Energy of Processes

To better analyze software and see real-time energy usage, a custom solution will be created. This new system will track the power used by each process and show live data. By building our own solution, we can avoid the problems of existing methods and get a more accurate and flexible way to monitor energy use.

However, by using Intel RAPL, we face some disadvantages which will be discussed in the next chapter.

A. Mathematical Approach

To measure the energy consumption of a specific process, we need to understand its power usage over time. This can be done by approximating the total energy consumed using the concept of integration. As explained in chapter II, in order to get the energy consumption, the power must be integrated over time. By using the definition of an integral, the consumption can be estimated by summing the power usage over discrete time intervals, as described by the following equation:

$$E \approx \sum_{i=1}^{N} P(t_i)\Delta t$$

where:

- E is the approximate energy consumed over the period of interest.
- $P(t_i)$ is the power consumption at discrete time t_i, measured in watts (W).
- Δt is the duration of each discrete time interval, measured in seconds (s).
- N is the total number of intervals over the period of observation.

With this approach, it is possible to create a sampler that tracks the consumption over an interval Δt, applying the formula for each new data point.

To implement this method, we follow these steps:

1. **Data Collection**: Set up a system to measure power consumption $P(t_i)$ at regular intervals.
2. **Interval Selection**: Choose a suitable time interval Δt for sampling. This interval should be short enough to capture significant variations in power consumption but long enough to avoid generating excessive data and computational load.
3. **Summation**: For each interval Δt, compute the product $P(t_i)\Delta t$. Summing these values over all intervals N will provide an estimate of the total energy consumed E.

This method allows us to monitor power usage in a detailed manner, gaining insights into the energy consumption patterns of the process under study.

B. Practical Implementation

In the repository (see appendix), a class has been implemented that addresses the problem using the mathematical approach explained above.

For data collection: Intel's **R**unning **A**verage **P**ower **L**imit (RAPL) technology, integrated into modern Intel processors, allows for precise monitoring and control of power consumption [7]. RAPL provides real-time power consumption data by using hardware counters that directly measure

the energy usage of specific domains. These counters record energy consumption in microjoules μJ and can be read through the system's interface.

RAPL is renowned for its high accuracy in measuring power consumption. Numerous studies and validations have confirmed RAPL's power measurements' accuracy. For instance, research from the University of Maine [8] found that RAPL's measurements are within 2.3% of actual power usage, making it a reliable tool for power monitoring.

Regarding the time interval selection: for our tests, an interval of 1 second was used, which was granular enough for smaller sample periods. For longer trace durations it is recommended to use a slightly bigger interval in order to not produce too much data.

To calculate the energy increase for a given Δt, `start_energy`, `start_total_cpu_time` and `start_process_cpu_time` are determined using the operating system libraries and RAPL. After waiting for our time interval Δt, the current `energy`, `total_cpu_time` and `process_cpu_time` are measured. The process energy used equals the `system_energy_diff` multiplied by the percentage of CPU time used by our process, to get the percentage of energy used:

$$E_p = \Delta E_t * \frac{\Delta CT_p}{\Delta CT_t}$$

where:

- E_p is the energy consumed by the process in the time interval.
- ΔE_t is the change in system energy.
- ΔCT_p and ΔCT_t are the CPU times for the process and total system during that interval, respectively.

In order to retrieve the necessary information, following methods where implemented. The full implementation is included in the repository[1]:

```
def measure_power_for_pid(...):
    ...
    start_energy = read_rapl_energy()
    start_total_cpu_time = get_total_cpu_time()
    start_process_cpu_time = get_process_time()
    for _ in range(int(duration/interval))
        ...
        wait(interval)
        energy, total_cpu_time, process_cpu_time
             = new_data()
        store()
        swap_vars()
```

This method starts the tracing for a given process. In case the process finishes before, it automatically will abort it self. This method makes use of other helper methods, which must be implemented to match the underlying system.

The algorithm reads the data and stores it in the `start_*` variables. After waiting the interval Δt, the new set of data gets recolleted and calculated with the just mentioned mathematical approach. The data then gets swapped in order to calculate the next data point, so that the last measured data is now the `start`-data.

C. Example of tracing

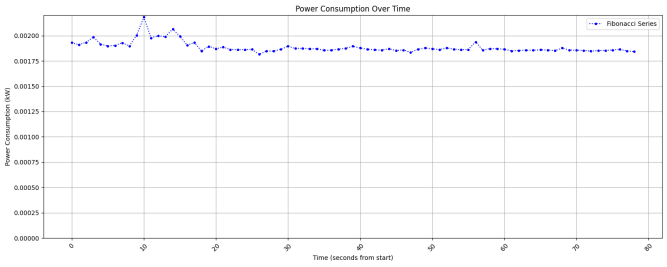

Figure 1. Energy Consumption of Fibonacci series

Figure 1 shows the tracing[2] of a program running a function called `fib(50)`, where the Fibonacci series [9] was implemented recursively. The complexity of the implemented algorithm is $O(2^n)$. Regarding power, the program used an average of $1.9^{-3}kW$ and a total energy consumption of $4^{-5}kWh$.

When talking about energy efficiency, it is possible to optimize the fibonacci algorithm to use $\approx 0kWh$. For this, a caching system has to be implemented, that stores the values produced before as a lookup table [10]. While the optimization is not directly part of the paper, the realization that by optimizing an algorithm we could save $\approx 4^{-5}kWh$ each execution, is.

V. Application Scenarios

As demonstrated in the constructed example, understanding which parts of a program consume the most energy allows us to implement optimization strategies that can reduce both costs and environmental impact.

The implementation presented in this paper for tracking the energy consumption of individual software processes offers significant benefits in various contextst. However, there are also certain limitations that need to be considered. Here, we explore both the advantages and constraints of this approach in practical usage scenarios.

A. Benefits

The use of Intel's RAPL technology enables highly accurate monitoring of power consumption at the granularity of individual processes. This precision is crucial for applications where even minor variations in energy usage are critical,

[1]The code is shortened and simplified to just explain the keypoints of the code.

[2]The benchmark was done on an Intel Core i7-13700k (Linux Manjaro).

such as in scientific computing or financial services, where energy costs and performance trade-offs must be carefully managed.

This method also facilitates real-time tracking of energy consumption, providing immediate insights into the power usage patterns of running processes. This capability is particularly beneficial for developers and system administrators who need to optimize software performance dynamically or troubleshoot energy-intensive operations.

By focusing on individual processes rather than overall server consumption, our approach allows for a granular analysis of software behavior. This is invaluable in scenarios such as cloud computing and data center management, where understanding the energy profile of specific applications can lead to more efficient resource allocation and reduced operational costs.

Furthermore, the method's ability to measure energy usage at various time intervals (Δt) makes it adaptable to a wide range of workloads, from long-running background tasks to short-lived high-intensity computations. This versatility supports diverse applications, from batch processing to real-time analytics.

Last, by identifying energy-intensive processes, our approach can drive efforts to reduce carbon footprints and promote sustainable computing practices. This is increasingly important in industries committed to environmental sustainability and regulatory compliance, such as technology firms with large-scale data operations.

B. Downsides

The implementation relies on Intel's RAPL technology, limiting its applicability to systems equipped with compatible Intel processors. This restricts its use in environments that employ a diverse set of hardware architectures, such as mixed-processor data centers or companies with heterogeneous computing environments. If this is not given, the user has to find a way to accurately track the consumption and implement it in the framework.

At the same time, tracing virtual machines or containerized applications may present a challenge, as the current implementation only supports tracing processes.

VI. Next Steps

To gain better insights into energy usage, it is necessary to expand the framework to monitor power consumption at method level. This can be achieved using **E**xtended **B**erkeley **P**acket **F**ilter (eBPF), a powerful tool that enables low-overhead monitoring within the Linux kernel [11].

eBPF programs can be hooked into process-related events like start, stop and context switches, capturing detailed real-time data on resource usage.

A. Advantages

- **Low Overhead**: eBPF is efficient and adds minimal performance impact.
- **High Precision**: Offers accurate and detailed data collection.
- **Flexibility**: Allows dynamic monitoring without changing the kernel or applications.

By leveraging eBPF, we can monitor individual processes in real-time and optimize their energy usage, advancing towards a more efficient and sustainable system.

B. Alternative

An alternative would be to integrate the built solution onto an application. In order to keep the overhead a mininum it would be needed to finetune the parameters and calibrate the system to get usefull data. For implementing this, the built software could integrate an API which allows other programs to start and stop traces. The data for each specific trace could then be labeled and stored to later be displayed, gaining valuable insight at method level.

VII. Conclusion

Tracking energy consumption is vital for understanding the behavior of programs. By accurately measuring the energy usage of individual software processes, we can gain valuable insights into their efficiency. This understanding is crucial for developing strategies to optimize energy consumption, leading to more sustainable and cost-effective operations.

Precise energy tracking enables us to identify inefficiencies and target areas for improvement. By focusing on the energy consumption of specific processes rather than the entire server, we can implement more effective optimizations. These optimizations not only reduce operational costs but also contribute to significant reductions in CO2 emissions.

Our approach highlights the importance of integrating detailed energy measurement techniques to support the development of greener digital solutions. With the ability to monitor and manage energy usage at a granular level, we can drive improvements that benefit both the environment and the bottom line.

In conclusion, understanding and optimizing energy consumption is key to fostering a more efficient and sustainable digital future. By leveraging precise energy tracking, we can enhance program efficiency and motivate substantial CO2 reduction, contributing to global environmental sustainability efforts.

VIII. Appendix

A. Repository URL

Repository containing the code and explanations.: https://github.com/liegsalz/intel-rapl-energy-tracing

B. Benchmark data

Timestamp	Power Consumption (Watts)
2024-06-02 21:41:57	1.9301837855060389
2024-06-02 21:41:58	1.9100498125780923
2024-06-02 21:41:59	1.933763064502705
2024-06-02 21:42:00	1.9856304456476468
2024-06-02 21:42:01	1.915621990837151
2024-06-02 21:42:02	1.8977264662781013
2024-06-02 21:42:03	1.9023566666666665
2024-06-02 21:42:04	1.9292044583333334
2024-06-02 21:42:05	1.8968702455264252
2024-06-02 21:42:06	2.003004755944931
2024-06-02 21:42:07	2.182132488139825
2024-06-02 21:42:08	1.9783354314297625
2024-06-02 21:42:09	1.9976358867610324
2024-06-02 21:42:10	1.9895680833333333
2024-06-02 21:42:11	2.0650709166666665
2024-06-02 21:42:12	1.9930055366972477
2024-06-02 21:42:13	1.9051381348875938
2024-06-02 21:42:14	1.931773
2024-06-02 21:42:15	1.8483330833333333
2024-06-02 21:42:16	1.8940007919966653
2024-06-02 21:42:17	1.870170875
2024-06-02 21:42:18	1.8858908796835971
2024-06-02 21:42:19	1.862167430237401
2024-06-02 21:42:20	1.8609421491045397
2024-06-02 21:42:21	1.860769694294044
2024-06-02 21:42:22	1.8635969166666666
2024-06-02 21:42:23	1.814646022490629
2024-06-02 21:42:24	1.8467040399833403
2024-06-02 21:42:25	1.8472497083333332
2024-06-02 21:42:26	1.863408988764045
2024-06-02 21:42:27	1.895883325281133
2024-06-02 21:42:28	1.872171826883063
2024-06-02 21:42:29	1.874376967930029
2024-06-02 21:42:30	1.8680750208159866
2024-06-02 21:42:31	1.8699496249999998
2024-06-02 21:42:32	1.8548453962500002
2024-06-02 21:42:33	1.856674052478134
2024-06-02 21:42:34	1.8660484583333332
2024-06-02 21:42:35	1.8749036666666665
2024-06-02 21:42:36	1.894330003331945
2024-06-02 21:42:37	1.8770741357767595
2024-06-02 21:42:38	1.8653729695960015
2024-06-02 21:42:39	1.8590696919233973
2024-06-02 21:42:40	1.8560669445602334
2024-06-02 21:42:41	1.8696190416666667
2024-06-02 21:42:42	1.8517204999999999
2024-06-02 21:42:43	1.858776531888287
2024-06-02 21:42:44	1.8347098100791337
2024-06-02 21:42:45	1.8635340615896798
2024-06-02 21:42:46	1.878117110741049
2024-06-02 21:42:47	1.8691009571369122
2024-06-02 21:42:48	1.8621984576907045
2024-06-02 21:42:49	1.878970125
2024-06-02 21:42:50	1.8657577740725304
2024-06-02 21:42:51	1.8601052083333331
2024-06-02 21:42:52	1.8630069166666667
2024-06-02 21:42:53	1.9381562499999998
2024-06-02 21:42:54	1.8564409754064195
2024-06-02 21:42:55	1.871469775187344
2024-06-02 21:42:56	1.8695081598667778
2024-06-02 21:42:57	1.8660802335279398
2024-06-02 21:42:58	1.848292375
2024-06-02 21:42:59	1.8525521249999999
2024-06-02 21:43:00	1.8564583749999999
2024-06-02 21:43:01	1.8546149521032902
2024-06-02 21:43:02	1.8598363182007496
2024-06-02 21:43:03	1.8576358333333334
2024-06-02 21:43:04	1.8502299042065804
2024-06-02 21:43:05	1.8765795416666666
2024-06-02 21:43:06	1.857177384423157
2024-06-02 21:43:07	1.8555937473947481
2024-06-02 21:43:08	1.8533642648896294
2024-06-02 21:43:09	1.846594710537276
2024-06-02 21:43:10	1.8519178259058726
2024-06-02 21:43:11	1.8535134166666665
2024-06-02 21:43:12	1.8578281132861307
2024-06-02 21:43:13	1.864404456476468
2024-06-02 21:43:14	1.848397042898792
2024-06-02 21:43:15	1.8427000833333331

References

[1] I. E. A. (IEA), "Electricity market report: Analysis and forecast to 2026," International Energy Agency, Report, 2024.

[2] P. D. B. Burger, "Fraunhofer: Electricity generation in germany in 2023," Fraunhofer, Report, 2024.

[3] B. I. des Poids et Mesures (BIPM), *The international system of units (si)*. Sèvres, France: Bureau International des Poids et Mesures (BIPM), 2019.

[4] J. Flinn and M. Satyanarayanan, "PowerScope: A tool for profiling the energy usage of mobile applications." IEEE, 2002.

[5] Y. Agarwal, S. Savage, and R. Gupta, "Joulemeter: How to measure energy usage in virtualized environments," Microsoft Research, 2009.

[6] PowerAPI Developers, "PowerAPI: A high-level api for monitoring and profiling energy consumption." 2024.

[7] The Linux Kernel Documentation Project, *Power capping framework (powercap)*. 2023.

[8] V. M. W. Spencer Desrochers Chad Paradis, "Using intel rapl to estimate dram power," University of Maine, Technical Report ECE-597, 2015.

[9] O. M. Oyelami, "The fibonacci numbers and its amazing applications," *ResearchGate*, 2018.

[10] I. Chen, “Understanding caching with memoize and fibonacci numbers.” Medium, 2020.

[11] eBPF Community, “EBPF - extended berkeley packet filter.” 2024.