Zurich University of Applied Sciences

Department School of Engineering

Institute of Computer Science

SPECIALIZATION PROJECT 2

Title

Author:
Caspar Wackerle

Supervisors:
Prof. Dr. Thomas Bohnert
Christof Marti

Submitted on
July 31, 2025

Study program:
Computer Science, M.Sc.

Imprint

Project:	Specialization Project 2
Title:	Title
Author:	Caspar Wackerle
Date:	July 31, 2025
Keywords:	energy efficiency, cloud, kubernetes

Study program:
Computer Science, M.Sc.
Zurich University of Applied Sciences

Supervisor 1:
Prof. Dr. Thomas Bohnert
Zurich University of Applied Sciences
Email: thomas.michael.bohnert@zhaw.ch
Web: Link

Supervisor 2:
Christof Marti
Zurich University of Applied Sciences
Email: christof.marti@zhaw.ch
Web: Link

Abstract

Abstract

The accompanying source code for this thesis, including all deployment and automation scripts, is available in the **PowerStack**[1] repository on GitHub.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Context

1.1 Introduction and Context

1.1.1 Cloud Computing and its impact on the global energy challenge

Global energy consumption is rising at an alarming pace, driven in part by the accelerating digital transformation of society. A significant share of this growth comes from data centers, which form the physical backbone of cloud computing. While the cloud offers substantial efficiency gains through resource sharing and dynamic scaling, its aggregate energy footprint is growing rapidly. While data center accounted for around 1.5% (around 415 TWh) of the worlds electricity consumption in 2024, they are set to more than double by 2030[2]. That is slightly more than Japans's current electricity consumption today.

This increase is fueled by the rising demand for compute-heavy workloads such as artificial intelligence, large-scale data processing, and real-time services. Meanwhile, traditional drivers of efficiency—such as Moore's law and Dennard scaling—are slowing down[3][4]. Improvements in data center infrastructure, like cooling and power delivery, have helped reduce energy intensity per operation[5], but these gains are approaching diminishing returns. As a result, total data center energy use is expected to grow faster than before, even as efficiency per unit of compute continues to improve more slowly[6].

1.1.2 Rise of the Container

Containers have become a core abstraction in modern computing, enabling lightweight, fast, and scalable deployment of applications. Compared to virtual machines, containers impose less overhead, start faster, and support finer-grained resource control. As such, they are widely used in microservice architectures and cloud-native environments[7].

This trend is amplified by the growing popularity of Container-as-a-Service (CaaS) platforms, where containerized workloads are scheduled and managed at high density on shared infrastructure. Kubernetes has become the de facto orchestration tool for managing such workloads at scale. While containers are inherently more energy-efficient than virtual machines in many scenarios[8], their widespread use presents a new challenge: understanding and attributing their energy consumption accurately.

1.1.3 Container Energy Consumption Measurement Challenges

Knowing the energy consumed by a container on a server is the essential elemenent to a container-level energy efficiency assessment of both the container itself, as well as the environment surrounding it. An accurate energy consumption estimation is therefore required to validate and improve any potential energy efficiency improvements of a container enviromnent, from kubernetes system components (e.g. Kubernetes Schedulers) to the containers themselves.

Energy consumption in containerized systems is inherently hard to measure due to the abstraction layers involved. Tools like RAPL (Running Average Power Limit) expose component-level energy metrics on modern Intel and AMD CPUs, but this information is not accessible from within containers or virtual machines. In public cloud environments, such telemetry is either not exposed or aggregated at coarse granularity, making direct measurement infeasible.

Containers further complicate attribution: because they share the kernel and hardware resources, it is difficult to isolate the energy impact of one container from another. Only indirect metrics—such as CPU time, memory usage, or performance counters—are available, and even these may be incomplete or noisy depending on system configuration and workload behavior. Various tools exist that attempt to model container power usage based on these inputs, but rarely are their produced metrics transistent and verified.

1.1.4 Problem Definition

The growing importance of containers in cloud environments, combined with the difficulty of directly measuring their energy usage, motivates this work. In particular, this thesis investigates the questions:

Question 1: Which metrics and models allow for reliable container-level power estimation?

Question 2: How should a software-based container energy consumption estimation tool be implemented?

Question 3: How can existng container energy consumption estimation tools be validated?

To answer these questions, this study explores methods of measuring server energy consumption, analyzes container workload metrics, and evaluates modeling techniques that aim to bridge the gap between raw energy data and container-level attribution. **The focus is on bare-metal Kubernetes environments, where full system observability allows for deeper analysis and model validation, serving as a foundation for future energy-aware cloud architectures.**

1.1.5 Context of this thesis

This thesis is part of the Master's program in Computer Science at the Zurich University of Applied Sciences (ZHAW) and represents the second of two specialization projects ("VTs"). The preceding project (VT1) focused on the practical implementation of a test environment for energy efficiency research in Kubernetes clusters. This thesis (VT2) is meant to explore theoretical and methodolocical aspects of container energy consumption measurements in detail.

Furhtermore, this thesis builds upon prior works focused on performance optimization and energy measurement. EVA1 covered topics such as operating system tools, statistics, and eBPF, while EVA2 explored energy measurement in computer systems, covering hardware, firmware, and software aspects. These foundational topics provide the basis for the current thesis but will not be revisited in detail.

1.1.6 Use of AI Tools

During the writing of this thesis, *ChatGPT*[9] (Version 4o, OpenAI, 2025) was used as an auxiliary tool to enhance efficiency in documentation and technical writing. Specifically, it assisted in:

- Structuring and improving documentation clarity.
- Beautifying and formatting smaller code snippets.
- Assisting in LaTeX syntax corrections and debugging.

All AI-generated content was critically reviewed, edited, and adapted to fit the specific context of this thesis. **ChatGPT was not used for literature research, conceptual development, methodology design, or analytical reasoning.** The core ideas, analysis, and implementation details were developed independently.

1.1.7 Project Repository

All code, configurations, and automation scripts developed for this thesis are publicly available in the PowerStack[1] repository on GitHub. The repository contains Ansible playbooks for automated deployment, Kubernetes configurations, monitoring stack setups, and benchmarking scripts. This allows for full reproducibility of the test environment and facilitates further research or adaptation for similar projects.

Chapter 2

State of the Art and Related Research

2.1 Energy consumption measurement and efficiency on data center level

Energy consumption and efficiency on a data center level has been well-studied to the point where various Literature reviews were published[10][11]. The bigger part of this research is focused on the data center infrasturture (cooling and power), and with good reason, as the data center infrastructure is responsible for a large part of the energy consumption. While a large number of coarse-, medium- and fine-grained metrics for data center energy consumption exist, most data center operators have focused on improving coarse-grained mertrics (especially the *Power Utilization Effectiveness, PUE*) with improvements to infrastructure. This has resulted in a PUE of 1.1 or lower in some cases[5]. Meanwhile, server energy efficiency has substantially improved, especially for parial load and idle power[12]. This has allowed data center operators to improve energy efficiency by simply installing more efficient cooling and power systems and servers. Fine-grained metrics such as server component utilization rates or speed were generally not used in the context of energy efficiency, but rather as performance metrics to ensure customer satisfaction.

2.2 Energy consumption measurement on a server level

As a result of the energy efficiency improvements of both data center infrastructure and server hardware mentioned in the previous section, a shift has started towards evaluating the actual server load energy efficiency. Efficiency gains on this level compound into further gains at the data center level. The method of resource-sharing of modern cloud computing (and especially the use of containers) have created great opportunities for server workload optimitation for energy efficiency, which in turn require power consumption measurements for evaluation. In the context of containers on multi-core processors, measuring the energy consumption of the entire server is insufficient, since it does not allow the attribution of consumped energy to specific containers or processes. While component-level power measurements provide finer measurements that could theoretically be modelled to display container energy consumption, they drastically raise the complexity for a number of reasons:

- Component-level energy consumption measurement without external tools is far from easy. While some components provide estimation models (e.g. Intel RAPL or Nvidia SMI), others can only be estimated using their performance metrics. This will invariably lead to large measurement uncertainties, especially with the component hardware differences between generations and manufacturers.
- The problem of attributing measured or estimated energy consumption to individual containers is in itself non-trivial: It not only requires a fine-grained time synchronization of energy consumption and used container resources due to the fast-switching nature most server components during any sort of multitasking.
- A deep understanding of dynamic or static energy consumption is required: Depending on the energy consumption attribution model, a container might not only account the energy it actively used, but potentially also account for a fraction of the energy consumed for any shared overhead such as shared hardware components, or system resources (such as the Kubernetes system architecure). This idea can be further extended: containers could potentially be penalized for any unused server resources, as these unused capacity still consume energy. These different attribution models lead to a larger debate about the goals of the measurements.
- Any server-level power models used to estimate the relation of individual component energy consumption suffers from the varity of different server configurations due to server specialization, such as Storage-, GPU-, or Memory-optimized servers.
-

The following sections of this chapter aim to present the current state-of-the-art in the various fields of research of the problem domains listed above.

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx here, a sentence will introduce the subsections:

- Hardware components: CPU / RAM / SoC / GPU / ...
-

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Finally, section xxxxxxxxxxxxxxxxxxxxxxxxxxxx aims to give an overview of the currently existing implementations of software-based container-level energy consumption estimation.

2.3 Overview of Power Data collection

In a systematic review cloud servers power models, Lin et al[13] categorize power collection methods into 4 categories:

Key	Value	Description	Data Granularity	Data Credibility
Based on instruments	Installation of extra devices	Bare-metal machines	Machine Level	Very high
Based on dedicated aquisition system	Specialized systems	Specified models of machines	Machine or component-level	High
Based on software monitoring	Build-in power models	Bare-metal and virtual servers	Machine, component, or VM level	Fair
Based on simulation	System simulation	Machine, component or VM level	Machine, component, or VM level	Low

TABLE 2.1: Comparison of power collection methods for cloud servers

2.3.1 Instrument-based power data aquisition

Instrument-based Data collection aquisition produces the hightest data credibility at a low granularity: These devices, installed externally (measuring the power supplied to the PDU) or internally (measuring the power flow between the PDU and motherboard) have been the source of information for a number of studies. The approach to simply measure electric power at convenient hardware locations using dedicated equipment can of course be extended to provide additional granularity: For example, Desrocher et al[14] custom-created a DIMM extender custom-fitted with Hall-sensor resistors and a linux measurement utility to measure power consumed by a DIMM memory module at 1kHz sampling rate using a *WattsAppPro?* power meter and a *Measurment Computing USB.1208FS-Plus* data aquisition board.

This of course highlights a fundamental truth of instrument-based data collection: While it is possible to implement a measuring solution that provides high-granular and high-sampling rate power data, it is paired with an immense effort since solutions like this are not provided off-the-shelf. Unsuprisingly, this is most valuable for benchmarking or validation (Desrochers used this setup to validate Intel RAPL DRAM power estimations on three different systems). However, this methodology is (currently) unsuitable for deployment to data center servers due to its bad scalability and prohibitive costs. Hence, the primary role of instrument-based power data aquisiton is as a benchmarking and validation tool for research and development.

2.3.2 Dedicated Aquisition systems

BMC Devices, IPMI and Redfish

Some manufacturers have developed specialized power data aquisition systems for their own server products. The baseboard management controller (BMC) is a typical dedicated aquisition system usually integrated with the motherboard, usually as part of the intelligent platform management interface (IPMI)[13]. It can be connected to the system bus, sensors and a number of components to provide power and temparature information about the CPU, memory, LAN port, fan, and the BMC itself.

Some comprehensive management systems such as Dell iDRAC or Lenovo xClarity have been further developed to provide high-quality, fine-grained power data due to their close interoperation between system software and underlying hardware. BMC devices on modern servers often offer IPMI- or Redfish interfaces. While these interfaces use the same physical servers, their implementation differ significantly, where Redfish generally offers higher accuracy (e.g through the use of higher-bit formats, whereas IPMI often uses 8-bit raw numbers).

In the context of container power consumption estimation, IPMI-implementations occupy an interesting role. In 2016, Kavanagh et al[15] found the accuracy of IMPI power data to be relatively inaccurate when compared with an external power meter, mainly due to the large measurement window size of 120 to 180 seconds and the inaccurate assessment of the idle power. They concluded that IMPI power data was still useful when a longer averaging window was used, and the initial datapoints discounted. In a later study, they suggest combining the measurements of IPMI and Intel RAPL (which they find to underestimate the power consumption) for a reasonable approximation of true measurement[16]. Kavanagh's findings have been cited in various studies, often to negate the use of IPMI for power measurement. When used, it sometimes is chosed because it was the "simplest power metric to read"[17] in the context of entire data centers.

Redfish is a modern Out-of-band System Management, first released in 2015 explicitely to replace IPMI [18]. It uses a RESTful API and JSON data format, making queries with code easier. In 2019, Wang et al[19] directly compared IPMI and redfish power data to a reading of a high accuracy power analyzer, and found Redfish to be more accurate than IPMI, with a MAPE of 2.9%, while also finding a measurement latency of about 200ms. They also found measurements to be more accurate in higher power ranges, which they attribute to the improved latency.

In conclusion, BMC power data aquired over Redfish provides a simple simple and comparatively easy way to measure system power based on various physical system sensors. Its biggest strenght lies in easy implementation and general availability. In the context of container energy consumption, BMC power data lacks the short sampling rates necessary to measure a a highly dynamcic container setup, but can prove useful as a validation or cross-reference dataset for longer intervals exceeding 120 seconds. Unfortunately, the data quality of BMC power data depends on the actual system, and power models can be significantly improved by initial calibration with an external power measurement device[15].

2.3.3 Power Monitoring Counters: Intel RAPL

Intel Running Average Power Level (RAPL) is a Power Monitoring Counter (PMC)-based feature introduced by Intel and provides a way to monitor and control the energy consumption of various components within their processor package[20]. An adaptation of RAPL for AMD processors uses largely the same mechansms and the same interface[21], although it provides less information than Intel's RAPL[22], providing no DRAM energy consumption.

Intel RAPL has been used extensively in research to measure energy consumption[23] despite some objections about its accuracy, which will be discussed sections 2.3.3 and 2.3.3. The general concencus is that RAPL is *good enough* for most scientific work in the field of server energy consumption and efficiency. As Raffin et at[24] point out, it is mostly used *like a black box without deep knowledge of its behavior*, resulting in implementation mistakes. For this reason, the next section 2.3.3 presents an overview of the RAPL fundamentals. Finally, section 2.6.1 discusses the currently available RAPL-based tools.

How RAPL works, and how it's used

[25], [24] This subsection provides an overview of how RAPL works and is used. It is based on the official Intel documentation[26, Section 16.10] and the works of Raffin et al [24] (2024) and Schöne et al [25] (2024).

Running Average Power Limit (RAPL) is a power management interface in Intel CPUs. Apart from power limiting and thermal management, it also allows to measure the energy consumed by various components (or *domains*). These domains individual CPU cores, integrated graphics (in non-server CPUs) and DRAM, as well as *package*, refering to the whole CPU die. While it initially used models to estimate energy use[27], it now uses physical measurements. The processor is divided into different power domains or "planes", representing specific components, seen in figure 2.1

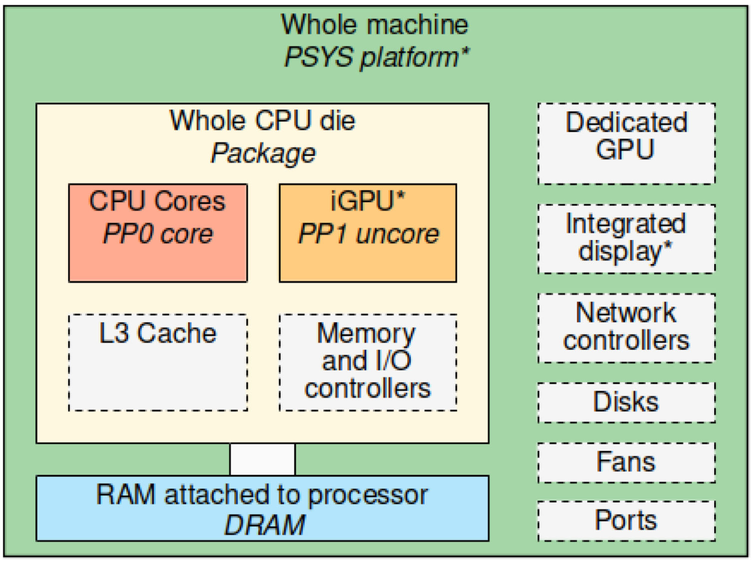
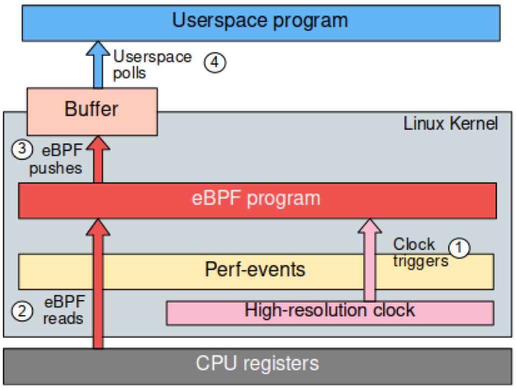


FIGURE 2.1: Hierarchy of possible RAPL domains and their corresponding hardware components. Domain names are in italic, and grayed items do not form a domain on their own, items with an asterisk are not present on servers[24].

RAPL provides hardware counters to read the energy consumption (and set power limits) for each domain. The energy consumption is measured in terms of processor-specific "energy units" (e.g. 61μJ for Haswell and Skylake processors). The counters are exposed to the operating system through model-specific registers (MSRs) and are updated approximately every millisecond. The main advantages of RAPL are that no external powermeters are required, nor a privileged access to the BMC (which could be used to power off the server). RAPL is more accurate than any untuned statistical estimation model.

Various interfaces can be used to extract measurements: One can use the low-level MSR directly, or choose a higher-level interface provided by the operating system.

Linux provides both the Power Capping framework (powercap) and the Performance Counters subsystem (perf-events). The perf-events interface can be read either from user space or from kernel space using eBPF. eBPF (*extended Berkeley Packet filter*) allows the injection of code into a kernel. Triggering the an eBPF program by a call to the `rdmsr x86` instruction would prevent a mode switch and theoretically lower measurement overhead. This method is visualized in figure 2.2a



(A) Measurement mechanism based on perf-events and eBPF[24].

mechanism	technical difficulty	required knowledge	safeguards	privileges	resiliency
MSR	medium	CPU knowledge	none	SYS_RAWIO cap. + msr module	poor
perf-events + eBPF	high (long, complicated code)	limited	overflows unlikely, many other possible mistakes	PERFMON and BPF capabilities	manual tweaks necessary for adaptation
perf-events	low	limited	good, overflows unlikely	PERFMON capability	good
powercap	low	limited	beware of overflows	read access to one dir	good, very flexible

(B) Comparison of RAPL measurement mechanisms[24].

In a detailed comparison, Raffin et al conclude that while the different interface involve different features and tradeoffs, which are summarize in figure 2.2b

Validation

Since its inception, RAPL has been subject of various validation studies, with the general concensus that it's accuracy could be considered "good enough"[24]. Notable works are Hackenberg et al, that in 2013 found RAPL accurate but missing timestamps[28], and in 2015 noticed a major improvement to RAPL accuracy, after Intel switched from a modeling approach to actual measurements for their Haswell architecture[27]. Desrochers et al concluded in a 2016 RAPL DRAM validation study[14] that DRAM power measurement was reasonably accurate, especially on server-grade CPUs. They also found measurement quality to drop when measuring and idling system.

A critical point in the RAPL validation was the introduction of the Alder Lake architecture, marking Intel's first heterogeneous processor, combining two different core architectures from the Core and Atom families (commonly referred to as P-Cores and E-cores) to improve performance and energy efficiency. While this heterogenity can improve performance and energy efficiency, it also increases complexity of scheduling decisions and power saving mechanisms, adding to the already complex architecture, featuring per-core Dynamic Voltage and frequency Scaling (DVFS), Idle states and Power Limiting / Thermal Proection.

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx THIS NEEDS TO BE PUT INTO THE CONTEXT OF RAFFIN 2024: SCHÖNE ONLY MEASURED AT 0.8GHz ON A CONSUMER i9. ACCORDING TO RAFFIN, ANYTHING FRQUENCY RELATED IS EXTREMELY IMPORTANT.

Schöne et al[25] found RAPL in the Alder Lake architecture to be generally consistent with external measurements, but exhibiting lower accuracy in low power scenarios. The following figure 2.3 shows these inaccuracies drastically, albeit on a conusmer-grade Intel Core i9-12900K processor measured at the base frequency of 0.8GHz. XXXXXXXXXXXXXXXXXXXXXXX MISSING CONTEXT

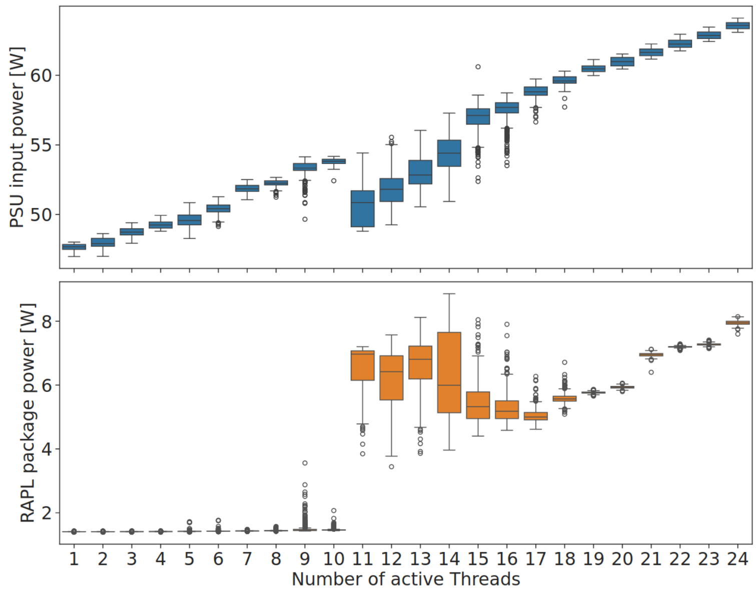


FIGURE 2.3: RAPL and reference power consumption sampled at 100 ms / 50 ms intervals respectively. Double precision matrix multiplication kernel at 0.8GHz running for 60s each at increasing number of active threads[25].

Limitations and issues

Several limitations of RAPL were noticed in various research works. Since RAPL is continually improved by Intel as new Processors are released, some of these issues have since been improved or entirely solved.

- **Register overflow:** The 32-bit register can experience an overflow error[24, 29]. This can be mitigated by sampling more frequently than the register takes to overflow. This interval can be calculated using the following equation:

$$t_{\text{overflow}} = \frac{2^{32} \cdot E_u}{P} \tag{2.1}$$

  Here, E_u is the energy unit used (61μJ for haswell), and P is the power consumption. On a Haswell processor consuming 84W, an overflow would occur every 52 minutes. Intel acknowledges this in the official documentation, stating that the register has a *wraparound time of around 60 seconds when power consumption is high*[26] This is solvable with a simple correction, provided that the measurement interfals are small enough: For two successive measurements m_{prev} and m_{current}, the actual measured difference is given by where C is a correction constant that depends on the chosen mechanism:

- **DRAM Accuracy:** DRAM Accuracy can only reliably be used for the Haswell architecture[14, 29], and may still exibit a constant power offset.

mechanism	constant C
MSR	`u32::MAX` i.e. $2^{32}-1$
perf-events	`u64::MAX` i.e. $2^{64}-1$
perf-events with eBPF	`u64::MAX` i.e. $2^{64}-1$
powercap	value give by the file `max_energy_uj` in the sysfs folder for the RAPL domain

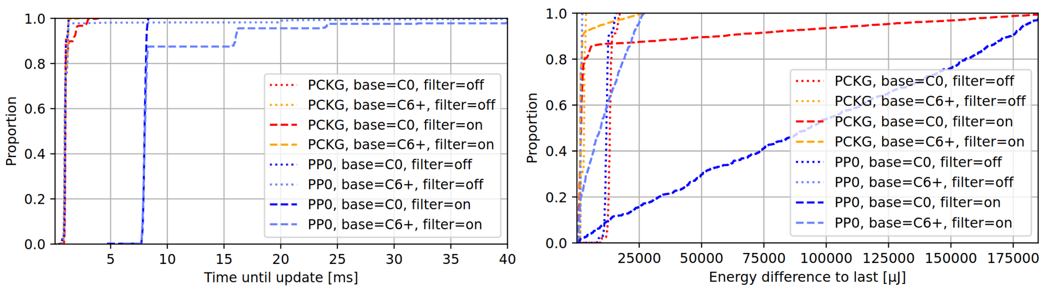
TABLE 2.2: RAPL overflow correction constant

- **Unpredictable Timings:** While the Intel documentation states that the RAPL time unit is 0.976ms, the actual intervals may vary. This is an issue since the measurements do not come with timestamps, making precise measurements difficult[29]. Several coping mechanisms have been used to mitigate this, notably *busypolling* (busypolling the counter for updates, significantly compromizing overhead in terms of time and energy[30]), *supersampling* (lowering the sampling interval, increasing overhead and occasionaly creating duplicates that need to be filtered[29]), or *high frequency sampling* (*lowering* the sampling rate when the resulting data is still sufficient[31]).

- **Lower idle power accuracy:** When measuring an idling server, RAPL tends to be less accurate[14, 25].

- **Side-channel attacks:** While the update rate of RAPL is usually 1ms, it can get as low as 50 μs for the PP0 domain (processor cores) on desktop processors[25]. This can be used to retrieve processed data in a side channel attack[25, 32]. To mitigate this issue while retaining RAPL functionality, Intel implements a filtering technique via the `ENERGY_FILTERING_ENABLE`[33, Table 2-2] entry. This filter adds random noise to the reported values. For the PP0 domain, this raises the temporal granularity to about 8ms. While this does not affect the average power consumption, point measurement power consumption can be affected. Figure **??** shows the effect of the filter, clearly indicating the loss granularity resulting from the activation of the filter.

FIGURE 2.4: Observable loss of granularity caused by the activation of `ENERGY_FILTERING_ENABLE`[25]



(A) Distribution of time between updates: With an enabled filter, the PP0 domain only provides updates every 8 ms, otherwise RAPL values are updated every 1 ms. If the load is too low, some updates might be skipped, e.g., the next update for PP0 and an enabled filter is at 16 ms.

(B) The minimal increase of a measurement is 1 energy unit of 61.035 μJ. Enabling the filter leads to a significant influence on measurements for the PP0 domain and a measureable influence on PCKG measurements.

RAPL has several limitations.

- counter overflow of the 32 bit register non atomic register updates "lack of individual core-level measurements"?????? ->fixed later?? in virtualised environments like cloud instances, the RAPL readings may be intercepted or modified by the hypervisor, potentially affecting their accuracy https://projectexigence.eu/green-ict-digest/running-average-power-limit-rapl/ (bad source) - Not all measuremnt methods are equally accurate?? -> [24]

Methods of measurement

Good comparison -> [24] - MSR / perf-events + eBPF / perf-events / powercap

2.4 Server Power models

In the absence of actual power data, power consumption models can be formulated that essentially map variables (such as CPU, Memory utilization) related to a server's state to its power consumption. Due to the strong correlation between CPU uzilization and server power, a great number of models use CPU metrics as the only indicator of server power. Fan et al[34] proposed a linear interpolation between idle power and full power, which they further refine into a non-linear form, with a parameter γ to be fitted to minimize mean square error. Similar research was done to further reduce error by introducting more complex non-linear models, such as Hsu and Poole[35], who studied the SPECpower_ssj2008-dataset of systems released between December 2007 and August 2010, and suggested the adaptation of two non-linear terms:

$$P_{\text{server}} = \alpha_0 + \alpha_1 u_{\text{cpu}} + \alpha_2 \left(u_{\text{cpu}}\right)^{\gamma_0} + \alpha_3 \left(1 - u_{\text{cpu}}\right)^{\gamma_1} \tag{2.2}$$

While models like these might work well when custom-fitted to specific, multipurpose servers, they have since been surpassed by the more common approach of modelling server power is to consider it an assembly of its components, such as Song et al[36] propose as:

$$P_{\text{server}} = P_{\text{cpu}} + P_{\text{memory}} + P_{\text{disk}} + P_{\text{NIC}} + C \tag{2.3}$$

, where C denotes the server's base power, which includes the power consumption of other components (regarded as static). This approach can easily be extended to include various other components such as GPUs, FPGAs or other connected components.

....

In a systematic review cloud servers power models, Lin et al[13] state that the common way

2.5 Power data collection

see Lin et al for overview -> instruments / dedicated aquisition system / software monitoring and calculation / simulation

2.5.1 CPU

2.5.2 Memory

2.5.3 Storage

2.5.4 Networking

2.6 Container energy estimation based on hardware power estimation

2.6.1 Tools

RAPL-based tools

- [37] An experimental comparison of software-based power meters (focus on CPU / GPU)
- [16] Rapid and accurate energy models through calibration with IPMI and RAPL
- [38] Scaphandre
- [**feieni2020smartwatts**] Smartwatts: Self-Calibrating Software-Defined Power Meter for containers
- [39] JoularJX: jaba-based agent for power monitoring at the code level
- [40]: KEPLER
- [41]: "AI power meter": Library to measure energy usage of machine learning programs, uses RAPL for CPU and nvidia-smi for GPU
- [42] CodeCarbon: Python package, estimates GPU + CPU + RAM: uses pyn-vml, ram RATIO (3W for 8G) and RAPL
- [43]: powertop
- [44]: Green metrics tool: measuring energy and CO2 consumption of software through a software life cycle anslysis (SLCA): Metric providers: RAPL, IPMI, PSU, Docker, Temperature, CPU, ... (sone external devices) [45]: PowerAPI: Python framework for building software-defined power

Chapter 3

Table templates

cell1 dummy text dummy text dummy text	cell2	cell3
cell1 dummy text dummy text dummy text	cell5	cell6
cell7	cell8	cell9

cell1 dummy text dummy text dummy text	cell2	cell3
cell1 dummy text dummy text dummy text	cell5	cell6
cell7	cell8	cell9

cell1 dummy text dummy text dummy text	cell2	cell3
cell1 dummy text dummy text dummy text	cell5	cell6
cell7	cell8	cell9

cell1 dummy text dummy text dummy text	cell2	cell3
cell1 dummy text dummy text dummy text	cell5	cell6
cell7	cell8	cell9

item 11	item 12	item 13
item 21	item 22	item 23

Appendix A

Appendix Title

Bibliography

[1] Caspar Wackerle. *PowerStack: Automated Kubernetes Deployment for Energy Efficiency Analysis*. GitHub repository. 2025. URL: https://github.com/casparwackerle/PowerStack.

[2] International Energy Agency. *Energy and AI*. Licence: CC BY 4.0. Paris, 2025. URL: https://www.iea.org/reports/energy-and-ai.

[3] Ryan Smith. *Intel's CEO Says Moore's Law Is Slowing to a Three-Year Cadence — But It's Not Dead Yet*. Accessed: 2025-04-14. 2023. URL: https://www.tomshardware.com/tech-industry/semiconductors/intels-ceo-says-moores-law-is-slowing-to-a-three-year-cadence-but-its-not-dead-yet.

[4] Martin Keegan. *The End of Dennard Scaling*. Accessed: 2025-04-14. 2013. URL: https://cartesianproduct.wordpress.com/2013/04/15/the-end-of-dennard-scaling/.

[5] Uptime Institute. *Global PUEs – Are They Going Anywhere?* Accessed: 2025-04-14. 2023. URL: https://journal.uptimeinstitute.com/global-pues-are-they-going-anywhere/.

[6] Eric Masanet et al. "Recalibrating global data center energy-use estimates". In: *Science* 367.6481 (2020), pp. 984–986. DOI: 10.1126/science.aba3758. eprint: https://www.science.org/doi/pdf/10.1126/science.aba3758. URL: https://www.science.org/doi/abs/10.1126/science.aba3758.

[7] Amit M Potdar et al. *Performance Evaluation of Docker Container and Virtual Machine*. 2020. DOI: https://doi.org/10.1016/j.procs.2020.04.152.. URL: https://www.sciencedirect.com/science/article/pii/S1877050920311315.

[8] Roberto Morabito. *Power Consumption of Virtualization Technologies: An Empirical Investigation*. 2015. DOI: 10.1109/UCC.2015.93.

[9] OpenAI. *ChatGPT (Version 4o)*. Used for document generation and formatting. 2025. URL: https://chat.openai.com.

[10] Saiqin Long et al. "A review of energy efficiency evaluation technologies in cloud data centers". In: *Energy and Buildings* 260 (2022), p. 111848.

[11] Chaoqiang Jin et al. "A review of power consumption models of servers in data centers". In: *applied energy* 265 (2020), p. 114806.

[12] Hannes Tröpgen et al. "16 Years of SPEC Power: An Analysis of x86 Energy Efficiency Trends". In: *2024 IEEE International Conference on Cluster Computing Workshops (CLUSTER Workshops)*. IEEE. 2024, pp. 76–80.

[13] Weiwei Lin et al. "A taxonomy and survey of power models and power modeling for cloud servers". In: *ACM Computing Surveys (CSUR)* 53.5 (2020), pp. 1–41.

[14] Spencer Desrochers, Chad Paradis, and Vincent M Weaver. “A validation of DRAM RAPL power measurements”. In: *Proceedings of the Second International Symposium on Memory Systems*. 2016, pp. 455–470.

[15] Richard Kavanagh, Django Armstrong, and Karim Djemame. “Accuracy of energy model calibration with IPMI”. In: *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*. IEEE. 2016, pp. 648–655.

[16] Richard Kavanagh and Karim Djemame. “Rapid and accurate energy models through calibration with IPMI and RAPL”. In: *Concurrency and Computation: Practice and Experience* 31.13 (2019), e5124.

[17] Joseph P White et al. “Monitoring and analysis of power consumption on hpc clusters using xdmod”. In: *Practice and Experience in Advanced Research Computing 2020: Catch the Wave*. 2020, pp. 112–119.

[18] Thomas-Krenn.AG. *Redfish - Thomas-Krenn-Wiki*. Accessed: April 27, 2025. n.d. URL: https://www.thomas-krenn.com/de/wiki/Redfish.

[19] Yewan Wang et al. “An empirical study of power characterization approaches for servers”. In: *ENERGY 2019-The Ninth International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*. 2019, pp. 1–6.

[20] Project Exigence. *Running Average Power Limit (RAPL)*. https://projectexigence.eu/green-ict-digest/running-average-power-limit-rapl/. Accessed April 2025. n.d.

[21] AMD. *amd_energy: AMD Energy Driver*. Accessed: 2025-04-28. 2023. URL: https://github.com/amd/amd_energy.

[22] Robert Schöne et al. “Energy efficiency aspects of the AMD Zen 2 architecture”. In: *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE. 2021, pp. 562–571.

[23] Tom Kennes. “Measuring IT carbon footprint: What is the current status actually?” In: *arXiv preprint arXiv:2306.10049* (2023).

[24] Guillaume Raffin and Denis Trystram. “Dissecting the software-based measurement of CPU energy consumption: a comparative analysis”. In: *IEEE Transactions on Parallel and Distributed Systems* (2024).

[25] Robert Schöne et al. “Energy efficiency features of the intel alder lake architecture”. In: *Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering*. 2024, pp. 95–106.

[26] Intel Corporation. *Intel® 64 and IA-32 Architectures Software Developer's Manual*. Volume 3B, Chapter 16.10: Plaform Specific Power Management Support. Available online: https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html. 2024.

[27] Daniel Hackenberg et al. “An energy efficiency feature survey of the intel haswell processor”. In: *2015 IEEE international parallel and distributed processing symposium workshop*. IEEE. 2015, pp. 896–904.

[28] Daniel Hackenberg et al. “Power measurement techniques on standard compute nodes: A quantitative comparison”. In: *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE. 2013, pp. 194–204.

[29] Kashif Nizam Khan et al. “Rapl in action: Experiences in using rapl for power measurements”. In: *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)* 3.2 (2018), pp. 1–26.

[30] Marcus Hähnel et al. “Measuring energy consumption for short code paths using RAPL”. In: *ACM SIGMETRICS Performance Evaluation Review* 40.3 (2012), pp. 13–17.

[31] Harald Servat et al. "Detailed and simultaneous power and performance analysis". In: *Concurrency and Computation: Practice and Experience* 28.2 (2016), pp. 252–273.

[32] Moritz Lipp et al. "PLATYPUS: Software-based power side-channel attacks on x86". In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2021, pp. 355–371.

[33] Intel Corporation. *Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 4: Model-Specific Registers*. Tech. rep. 335592-081US. Accessed 2025-04-28. Intel Corporation, Sept. 2023. URL: `https://cdrdv2.intel.com/v1/dl/getContent/671098`.

[34] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. "Power provisioning for a warehouse-sized computer". In: *ACM SIGARCH computer architecture news* 35.2 (2007), pp. 13–23.

[35] Chung-Hsing Hsu and Stephen W Poole. "Power signature analysis of the SPECpower_ssj2008 benchmark". In: *(IEEE ISPASS) IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE. 2011, pp. 227–236.

[36] Shuaiwen Leon Song, Kevin Barker, and Darren Kerbyson. "Unified performance and power modeling of scientific workloads". In: *Proceedings of the 1st International Workshop on Energy Efficient Supercomputing*. 2013, pp. 1–8.

[37] Mathilde Jay et al. "An experimental comparison of software-based power meters: focus on CPU and GPU". In: *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE. 2023, pp. 106–118.

[38] Hubblo-org. *Scaphandre Documentation*. Accessed: 2025-04-28. 2024. URL: `https://github.com/hubblo-org/scaphandre-documentation`.

[39] JoularJX Contributors. *JoularJX: Energy profiling agent for Java applications*. Accessed: 2025-04-28. 2023. URL: `https://github.com/joular/joularjx`.

[40] Inc. Meta Platforms. *Kepler: Kubernetes-based power and energy estimation framework*. Accessed: 2025-04-28. 2023. URL: `https://github.com/sustainable-computing-io/kepler`.

[41] GreenAI-UPPA. *AI PowerMeter: A Tool to Estimate the Energy Consumption of AI Workloads*. Accessed: 2025-04-28. 2023. URL: `https://greenai-uppa.github.io/AIPowerMeter/`.

[42] MLCO2. *CodeCarbon: Track emissions from your computing*. Accessed: 2025-04-28. 2023. URL: `https://github.com/mlco2/codecarbon`.

[43] Intel Corporation. *PowerTOP: Linux tool to diagnose issues with power consumption and power management*. Accessed: 2025-04-28. 2023. URL: `https://github.com/fenrus75/powertop`.

[44] Arne Tarara. *Green Coding Documentation*. Accessed: 2025-04-28. 2023. URL: `https://github.com/green-coding-solutions/green-metrics-tool`.

[45] Guillaume Fieni et al. "PowerAPI: A Python framework for building software-defined power meters". In: *Journal of Open Source Software* 9.98 (2024), p. 6670.