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

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 elemenet 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 measuremnt methods, metrics or 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 existing 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 partial 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 Management Library* (NVML)), 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.

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

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

TABLE 2.1: Comparison of power collection methods for cloud 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, focussing on different measurement approches: Direct hardware measurements, In-band measurement techniques and model-based estimation. The following sections are organized by measurement approach, foregoing organization by server component. For this reason, section 2.8 provides a brief summary of component-specific energy consumption measurement techniques.

2.3   Direct Hardware Measurement

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 et al used their 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

2.3.2.1   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 Management System, first released in 2015 explicitly 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].

2.4 In-Band Measurement Techniques

In-band measurement techniques refer to methods of power consumption monitoring that utilize built-in telemetry capabilities of system components to collect energy usage data directly from within the host system. Unlike external power meters or BMCs like IPMI, which operate independently of the main system, in-band techniques leverage on-die sensors and software interfaces to gather power metrics in real-time. These techniques provide fine-grained data with minimal additional hardware, making them well-suited for scalable environments like Kubernetes clusters. However, their accuracy and granularity are often dependent on the hardware's internal estimation algorithms, which may introduce uncertainties compared to direct measurement methods.

2.4.1 ACPI

The *Advanced Configuration and Power Interface (ACPI)* is a standardized interface that facilitates power management and hardware configuration by allowing the operating system to control hardware states such as processor sleep, throttling, and performance modes [20]. It plays a significant role in processor performance tuning by exposing C-states (idle), P-states (performance), and T-states (throttling) which the OS can leverage to adjust the processor's activity, frequency, and voltage.

Although ACPI defines these power states, their actual implementation is processor-specific, and the interface does not provide real-time telemetry. As such, ACPI does not expose instantaneous power consumption values. Any attempt to estimate power based on ACPI would require detailed knowledge of processor-specific behavior, including the mapping between frequency, voltage, and power—information that is not exposed through ACPI. As a result, limited research was conducted on this topic.

In theory, one could attempt to use ACPI's ⎇PSS (Performance Supported States) table, which lists available P-states along with nominal voltage, frequency, and optionally estimated maximum power dissipation, to perform rough CPU power estimation. This method would involve tracking CPU residency in each performance state and applying simple integration models to estimate total energy. However, due to the static nature of ⎇PSS entries and the lack of temporal precision, such estimates would be inherently coarse-grained and typically inaccurate for modern processors with dynamic voltage and frequency scaling or turbo modes.

Consequently, ACPI is rarely used in contemporary power estimation contexts. Its primary role remains in system configuration and power state control rather than accurate energy quantification. In modern Intel processors, the Running Average Power Limit (RAPL) interface provides a more appropriate solution for in-band power measurement. This makes RAPL the preferred tool for energy-aware computing research and production environments alike.

2.4.2 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[21]. An adaptation of RAPL for AMD processors uses largely the same mechansms and the same interface[22], although it provides less information than Intel's RAPL, providing no DRAM energy consumption[23]. Unfortunately RAPL does not have a detailed low-level implementation documentation, and the exact methodology of the RAPL calculations remain unknown[24].

Intel RAPL has been used extensively in research to measure energy consumption[25] despite some objections about its accuracy, which will be discussed sections 2.4.2.2 and 2.4.2.3. 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[26] 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 2.4.2.1 presents an overview of the RAPL fundamentals.Finally, section **??** discusses the currently available RAPL-based tools.

2.4.2.1   RAPL measurement methods

This subsection provides an overview of how RAPL works and is used. It is based on the Intel rchitectures Software Developer's Manual[27, Section 16.10] and the works of Raffin et al [26] (2024) and Schöne et al [28] (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[29], it now uses physical measurements. The processor is divided into different power domains or "planes", representing specific components, seen in figure 2.1. Notably, not all domains are present in all systems: Both client-grade systems feature the *Package* and *PP0 core* domains, server grade processors typically

don't show the *PP1 uncore*-domain typically used for integrated GPUs, and client-grade processors don't show the *DRAM* domain. The *PSYS* domain for the "whole machine" is ill defined and only exists on client-grade systems. In an experiment with recent Lenovo and Alienware laptops, Raffin et al found that the *PSYS* domain reported the total consumption of the laptop, including display, dedicated GPU and other domains. Regardless, this thesis will focus on the RAPL power domains available to server-grade processors.
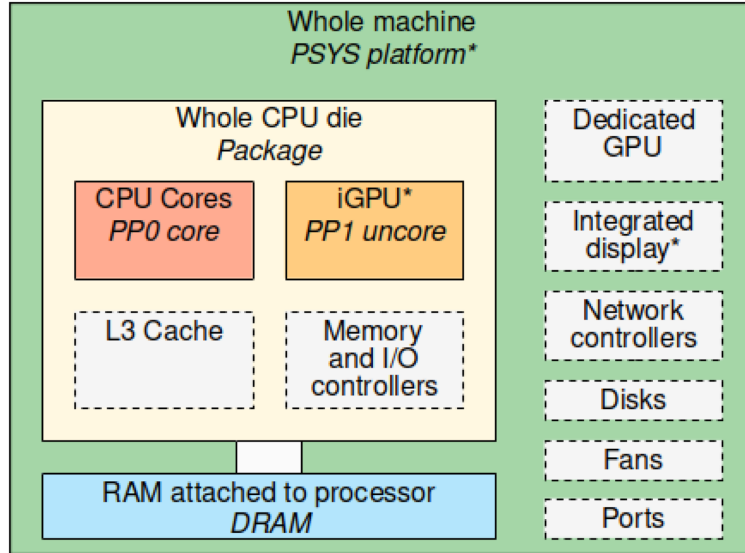
FIGURE 2.1: Hierarchy of possible RAPL domains and their corresponding hardware components. Domain names are in italic, and grayed items do not form a domain on their own, items with an asterisk are not present on servers[26].

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. $61\mu J$ 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 measurement methods can be used to extract RAPL measurements. In a detailed comparison, Raffin et al[26] outline their individual features and tradeoffs, which are summarize in figure 2.2b:

- **Lacking documentation:** Since there is no publicly available documentation of the low-level RAPL implementation, implementations are bound to suffer inaccuracies and inconsistencies due to a lack of understanding.

- The **Model-Specific Register (MSR)** interface provides low-level access to RAPL energy counters but is complex and hardware-dependent. Developers must manually determine register offsets and unit conversions based on processor model and vendor documentation. This method lacks safeguards, requires deep processor knowledge, and is error-prone, with incorrect readings difficult to detect. Although read-only access poses no risk to system stability,

MSRs expose sensitive data and are thus restricted to privileged users (e.g.,
`root` or `CAP_SYS_RAWIO`). Fine-grained access control is not supported na-
tively, though the `msr-safe` module offers limited mitigation.

- The **Power Capping (powercap)** framework is a high-level Linux kernel in-
  terface that exposes RAPL energy data through the sysfs filesystem, making it
  accessible from userspace. It simplifies energy measurements by automatically
  handling unit conversions and domain discovery, requiring minimal hard-
  ware knowledge. Though domain hierarchy can be confusing (especially with
  DRAM domains appearing nested under the package domain) powercap re-
  mains user-friendly and scriptable. It supports fine-grained access control via
  file permissions and offers good adaptability to hardware changes, provided
  the measurement tool doesn't rely on hard-coded domain structures.

- The **perf-events** subsystem provides a higher-level Linux interface for access-
  ing RAPL energy counters as counting events. It supports overflow correction
  and requires less hardware-specific knowledge than MSR. Each RAPL domain
  must be opened per CPU socket using `perf_event_open`, and values are
  polled from userspace. While it lacks a hierarchical structure like powercap
  and may be harder to use in certain languages or scripts, it remains adaptable
  and robust across different architectures. Fine-grained access control is possi-
  ble via kernel capabilities or `perf_event_paranoid` settings.

- **eBPF** enables running custom programs in the Linux kernel, and in this con-
  text, it is used to directly read RAPL energy counters from within kernel space,
  potentially reducing measurement overhead by avoiding user-kernel context
  switches. The implementation attaches an eBPF program to a CPU clock event,
  using `perf_event_open` to access energy counters and buffering results for
  userspace polling (is visualized in figure 2.2a).While offering the same over-
  flow protection as regular `perf-events`, this approach is significantly more
  complex, prone to low-level errors (especially in C), and requires elevated priv-
  ileges (`CAP_BPF` or `root`). It also lacks portability, as it demands manual adap-
  tation to kernel features and domain counts, limiting its maintainability across
  systems.
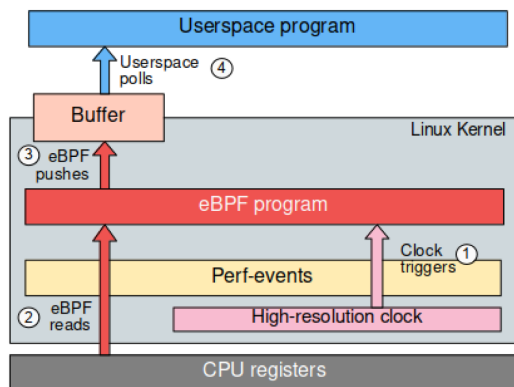
(A) RAPL perf-event eBPF mechanism

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) RAPL measurement mechanisms comparison

FIGURE 2.2: RAPL measurements: eBPF and comparison[26]

In their research, Raffin et al conclude that all four mechanisms have small or negligible impact on the running time of their benchmarks. They formulate the following recommendations for future energy monitoring implementations:

- Measuring frequencies should be adapted to the state of the node, preventing high measurement overhead, due to a reduction in time spent in low-power states. Under heavy load, a highfrequencay can be used in order to capture more information.

- `perf-events` is the overall recommeded measurement method with good efficiency, latency and overflow protection. Powercap is less efficient, but provides a simpler sysfs API.

- Even though `perf-events` and eBPF-measurement method seems to be the most energy-efficient, it is not recommended in light of its complexity. For the same reason, the MSR method is not recommeded, as it raises complexity while counter-intuitively being slower than `perf-events`

RAPL MSRs can be read on some cloud computing resources (e.g. some Amazon EC2-instances), although the hypervisor traps the MSR reads, which can add to the polling delay. in EC2, the performance overhead also significantly increases to <2.5% (as compared to <1% on standalone systems)[24].

2.4.2.2 RAPL 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"[26]. Notable works are Hackenberg et al, that in 2013 found RAPL accurate but missing timestamps[30], and in 2015 noticed a major improvement to RAPL accuracy, after Intel switched from a modeling approach to actual measurements for their Haswell architecture[29]. 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. Later, Alt et al[31] tested DRAM accuracy of heterogeneous memory systems of the more recent Ice Lake-SP architecture and concluded that DRAM estimates behaved differently than on older architectures. They noted that the RAPL overestimates DRAM energy consumption by a constant offset, which they attribute to the off-DIMM voltage regulators of the memory 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 Protection.

Schöne et al[28] 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, albeit tested on a conusmer-grade Intel Core i9-12900K processor measured at the base frequency of 0.8GHz.
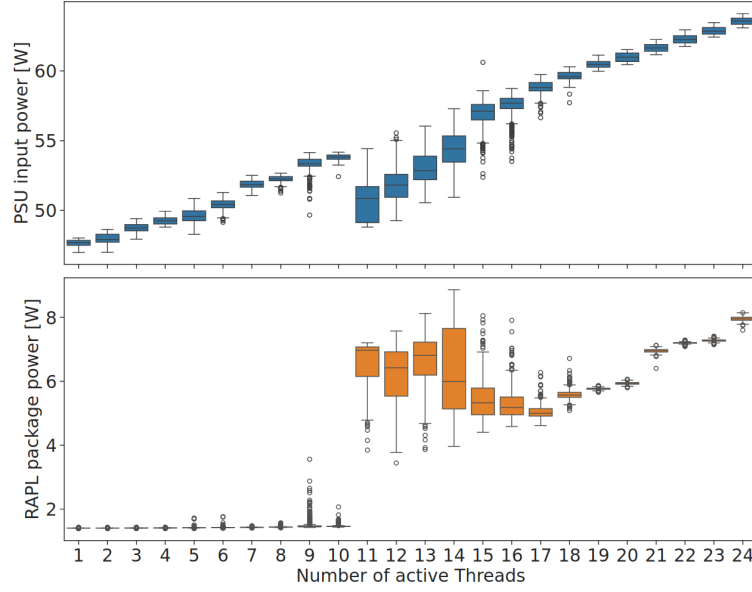
FIGURE 2.3: RAPL and reference power consumption sampled at 100
ms / 50 ms intervals respectively. Double precision matrix multipli-
cation kernel at 0.8GHz running for 60s each at increasing number of
active threads[28].

2.4.2.3 RAPL 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[26, 32].
  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 con-
  sumption. On a Haswell processor consuming 84W, an overflow would occur
  every 52 minutes. Intel acknowledges this in the official documentation, stat-
  ing that the register has a *wraparound time of around 60 seconds when power con-
  sumption is high*[27] 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

$$\Delta m = \begin{cases} m_{\text{current}} - m_{\text{prev}} + C & \text{if } m_{\text{current}} < m_{\text{prev}} \\ m_{\text{current}} - m_{\text{prev}} & \text{otherwise} \end{cases} \tag{2.2}$$

  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, 31, 32], and may still exibit a constant power offset (like at-
  tributed to the voltage regulator power loss of the memory system).

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[32]. 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[33]), *supersampling* (lowering the sampling interval, increasing overhead and occasionaly creating duplicates that need to be filtered[32]), or *high frequency sampling* (*lowering* the sampling rate when the resulting data is still sufficient[34]). Another solution is to use a *low sampling frequency* to smoothe out the relative error due to spikes, with the only drawback of loss of temporal precision. At sampling rates slower than 50Hz, the relative error is less than 0.5% [24].

- **Non-atomic register updates:** RAPL register updates are nont atomic[32], meaning that the different RAPL values show a delay between individual updates. This may introduce errors when sampling multiple counters at a high sampling rate, making it possible to read both fresh and stale values of different counters.

- **Lower idle power accuracy:** When measuring an idling server, RAPL tends to be less accurate[14, 28].

- **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[28]. This can be used to retrieve processed data in a side channel attack (coined "Platypus")[28, 35].

To mitigate this issue while retaining RAPL functionality, Intel implements a filtering technique via the `ENERGY_FILTERING_ENABLE`[36, Table 2-2] entry, or when *Software Guard Extension (SGX)* is activated in the BIOS. This filter adds random noise to the reported values (vizualized in Figure 2.4a). This can be seen 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 2.4 shows the effect of the filter, clearly indicating the loss granularity resulting from the activation of the filter. In a 2022 article, Tamara[37] found a surprising higher mean with the filter activated and deemed filtered RAPL energy data unusable. In a more elaborate experiment in 2024, Schöne et al did not encounter these inaccuracies anymore.
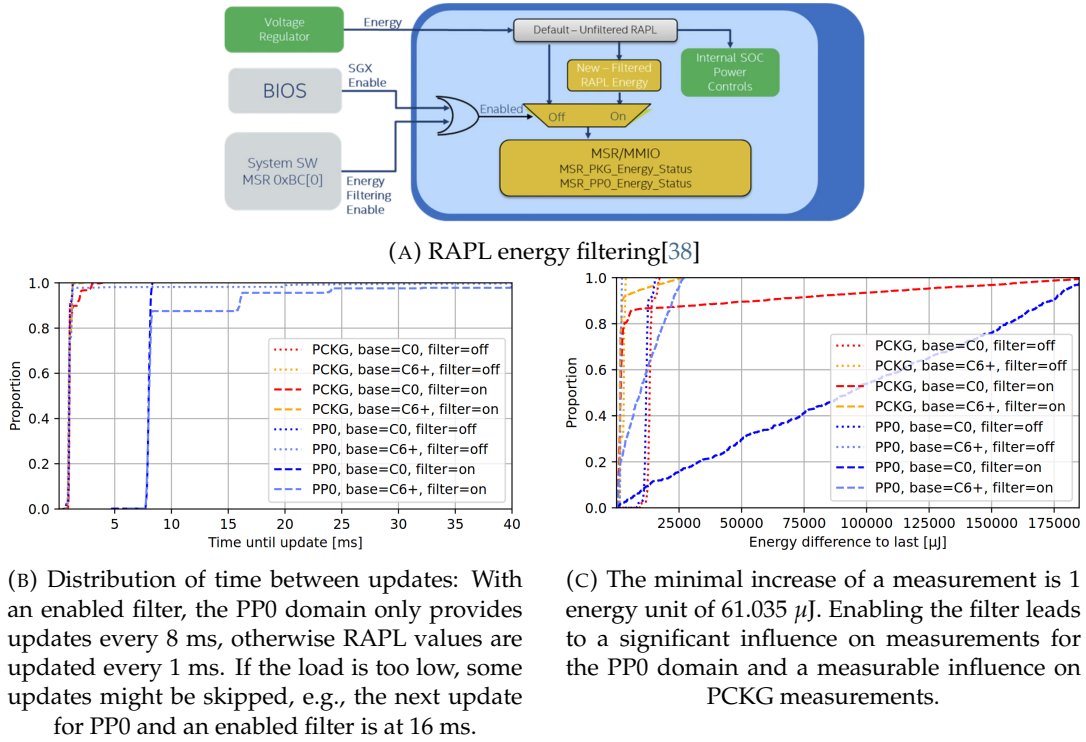
(A) RAPL energy filtering[38]

(B) 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.

(C) 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 measurable influence on PCKG measurements.

FIGURE 2.4: Observable loss of granularity caused by the activation of `ENERGY_FILTERING_ENABLE`[28]

2.4.2.4 RAPL conclusions

The energy measurement accuracy of RAPl has significantly improved since its inception and provides a generally accepted way to measure system energy consumption. It is well-validated and accepted as the most accurate fine-granular energy measurement tool. Some known limitations have historically created inaccuracies in developed measurement tools, but corrections of these limitations exist.

2.4.3 Graphical Processing Units (GPU)

In recent years, the utilization of GPUs in cloud computing environments has grown significantly, driven primarily by the increasing demand for high-performance computations in machine learning, artificial intelligence, and large-scale data processing [39]. Kubernetes now includes mechanisms for GPU provisioning, enabling containerized workloads to leverage GPU acceleration [40].

Although GPUs remain less common than traditional CPU-based workloads in typical Kubernetes clusters, their adoption is rapidly accelerating. Industry reports indicate that GPU usage in Kubernetes has seen a growth rate of nearly 58% year-over-year, outpacing general cloud computing growth rates [41]. This increase is largely attributed to ML workloads and real-time processing tasks that benefit from the parallel processing capabilities of GPUs [42]. Furthermore, hyperscalers have integrated GPU support directly into their managed Kubernetes services, reflecting the growing demand for GPU-powered workloads in containerized environments.

Despite this growth, GPU deployments are still not as pervasive as CPU-based workloads in Kubernetes-managed clusters. The primary focus of this thesis is on the

measurement and analysis of energy consumption in more common, CPU- and memory-centric Kubernetes workloads. Nevertheless, due to the rising significance of GPUs, their energy measurement techniques and potential integration within Kubernetes environments are briefly examined.

Ultimately, the inclusion of GPU energy measurements remains outside the primary scope of this thesis but is acknowledged as an important area for future research. This structured exploration serves to highlight current limitations and opportunities for enhancing energy efficiency in Kubernetes-managed GPU workloads.

2.4.3.1 GPU virtualization technologies

Full GPU Virtualization   Full GPU virtualization provides isolated instances of a single physical GPU to multiple virtual machines. This is achieved using technologies such as NVIDIA's *vGPU* or AMD's *MxGPU (Multiuser GPU)*. These technologies allow a VM to see a complete GPU, while the underlying hypervisor manages resource partitioning and scheduling [43, 44] eigher though the use of partitioning or time-slicing. In a Kubernetes environment, full GPU virtualization is commonly utilized through:

- **vGPU on VMware or OpenStack:** Kubernetes clusters running on VMware vSphere or OpenStack can request vGPU instances as if they were physical GPUs. These instances are shared among containers while maintaining memory and compute isolation.

- **Device Plugin Integration:** NVIDIA, AMD and Intel provide a Device Plugin for Kubernetes, enabling seamless GPU discovery and allocation across pods [40].

Multi-Instance GPU (MIG)   Introduced with the NVIDIA A100 architecture, Multi-Instance GPU (MIG) allows a single GPU to be partitioned into up to seven independent instances, each with its own dedicated compute, memory, and cache resources [45]. Unlike traditional vGPU, MIG provides true hardware-level isolation, preventing noisy-neighbor effects and enabling finer resource allocation. MIG instances are exposed to Kubernetes as individual GPUs. For example, a single A100 GPU partitioned into seven MIG instances appears as seven separate GPU resources, each assignable to different containers. MIG-aware device plugins ensure proper scheduling and isolation. Hence, MIG technology is particularly useful for multi-tenant environments and supports finer granularity in resource allocation compared to traditional vGPU models.

GPU Passthrough   GPU passthrough allows a physical GPU to be exclusively assigned to a single VM or container. Unlike virtualization, where resources are shared, passthrough dedicates the full GPU to one environment, offering near-native performance [46]. GPU passthrough is configured at the hypervisor level (e.g., KVM or VMware ESXi) and can be exposed to Kubernetes nodes. Pods scheduled on nodes with GPU passthrough access gain complete control of the GPU, enabling direct memory access and high-performance computation.

GPU virtualization technologies enable efficient multi-tenant use of GPU resources, enhancing performance and cost-effectiveness in cloud-native environments. For

the purposes of energy measurement, understanding these virtualization layers is essential for accurate per-container energy attribution.

2.4.3.2   GPU nvidia-NVML energy measurements and validation

Modern GPUs are equipped with **built-in power sensors** that enable real-time energy measurement. For instance, Nvidia GPUs expose power metrics through the *Nvidia System Management Interface (nvidia-smi)*, which reports instantaneous power draw, temperature, and memory usage [47]. This interface allows for programmatic access to GPU power consumption, making it a common choice for monitoring and energy profiling in both standalone and containerized environments [45].

In 2024, Yang et al. conducted a comprehensive study on the accuracy and reliability of NVIDIA's built-in power sensors, examining over 70 different models[48]. He concludes that previous research placed excessive trust in nvidia-NVML, overlooking the importance of measurement methodology. The study revealed several critical findings:

- **Sampling Limitations:** Nvidia NVML gives the option to specify a sampling frequency in units of milliseconds. However, on certain models, such as the A100 and H100, power is sampled only around 25% of the time, introducing potential inaccuracies in total energy consumption estimations.

- **Transient Response Issues:** While measured power reacted instantly to a suddenly applied workload, nvidia-NVML would report values with a delay of several hunderd milliseconds on some devices. Also, a slower rise (with linear growth) was discovered, taking over a second to catch up to correct power figures in some instances. Generally, server-grade GPUs were shown to provide more instantaneous power measurements.

- **Measurement Inaccuracies:** The average error rate in reported power draw was found to be approximately 5%, deviating from NVIDIA's claimed fixed error margin of 5W. This error would remain consistent when the GPU reached a constant power draw.

- **Averaging Effects:** Reported power consumption values are averaged over time, masking short-term fluctuations and potentially underreporting peak consumption.

To address these limitations, the study proposed best practices such as running multiple or longer iterations of workloads to average out sampling errors, introducing controlled phase shifts to capture different execution states, and applying data corrections to account for transient lags [48]. These adjustments reduced measurement errors by up to 65%, demonstrating the importance of refining raw sensor data for more accurate energy profiling.

2.4.3.3   Related Research

While most reaseach has used nvidia-NVML to measure GPU power consumption, some research was done on alternative measurement tools, ususally to address similar issues as were stated by Yang et al in the previous section. Specifically, the following three tools focussed were proposed to provide higher sampling rates to enable

finer-grained power analysis.

AccellWattch   In 2021, Pan et al proposed *AccelWattch*[49], a configurable GPU power model that provides both a higher accuracy cycle-level power model, and a way to measure constant and static power, utilizing any pure-software software performance mode,nvidia-NVML, or a combination of the two. Notably, their model is DVFS-, power-gating- and divergence-aware. The resulting power model was validated against measured ground truth using an Nvidia Volta GV100, yielding a MAPE error between $7.5 - 9.2 \pm 2.1\% - 3.1\%$, depending on the AccelWattch variant. The Volta model was later validated against Pascal TITAN X and Turing RTX 2060-architectures without retraining, achieving $11 \pm 3.8\% and 13 \pm 4.7\%$ MAPE, respectively. The authors conclude that AccelWattch can reliably predict power consumption of these specific GPU architectures. In the context of Kubernetes energy consumption, AccelWattch contributes a fine-grained temporal granularity

FinGraV   In 2024, Singhania et al propose *FinGraV*[50] (abbreviated from **Fine-Gra**in **V**isibility), a fine-grained power measurements tool capable of sub-millisecond power profiling for GPU executions on an AMD MI300X GPU. They identify these main challenges of high-resolution GPU power analysis (see figure 2.5a):

- **Low sampling frequency:** Standard GPU power loggers operate at intervals too coarse (tens of milliseconds) to capture the sub-millisecond executions of modern kernels.

- **CPU-GPU time Synchronization:** Synchronizing power measurements with kernel start and end times is problematic due to the asynchronous nature of CPU-GPU communication.

- **Execution time variation:** Minor variations in memory allocation or access patterns lead to inconsistent kernel execution times, complicating time-based power profiling.

- **Power variance across executions:** Repeated executions of the same kernel, or interleaved executions with other kernels, manifest in fluctuating power consumption, challenging consistent profiling.

(A) Challenges in fine-grain GPU power analysis

(B) FinGraV strategies to adress challenges

FIGURE 2.5:   FinGraV GPU power measurement challenges and strategies[50]

To overcome these challenges, FinGraV introduces several strategies (see figure 2.5b).

- **On-GPU Power Logger:** FinGraV leverages a high-resolution (1 ms) power logger, capturing the average of multiple instantaneous power readings.

- **High-Resolution Time Synchronization:** GPU timestamps are read from the CPU side before kernel execution, and synchronization is maintained throughout execution to correlate power samples with kernel events.

- **Execution Time Binning:** Kernel executions are grouped into "bins" based on empirical runtime ranges, enabling tighter power profiling while discarding outlier runs.

- **Power Profile Differentiation:** FinGraV distinguishes between Steady-State Execution (SSE) and Steady-State Power (SSP) profiles. SSP represents the stabilized power consumption after initial transients, providing the most accurate depiction of kernel power consumption.

The application of FinGraV to bencharmks reveals several critical obervations: Kernel executions differ significantly between initial runs and steady-state, with deviations up to 80%. Memory-bound kernels and compute-light kernels are found to be highly sensitive to the preceding kernel, impacting their power profile. Furthermore, the authors expose discrepancies in GPU power scaling relative to computational load, particularly for compute-light kernels.

FinGraV introduces promising concepts that could, in theory, enable more granular and accurate GPU power analysis in container-based GPU workloads. Its methodological approach addresses key challenges in sub-millisecond power measurement. However, its current implementation is tightly coupled with the AMD MI300X GPU,

relying on hardware-specific logging capabilities that are not universally available. While the underlying concepts may be extendable to other GPUs, achieving this is far from trivial, requiring significant adaptation and low-level access to power metrics that are often proprietary or limited by driver capabilities.

Consequently, FinGraV highlights both the challenges and potential solutions for fine-grained GPU power analysis but falls short of providing a general-purpose framework that could be easily integrated into Kubernetes energy measurement tools. It also underscores the broader issue that GPU energy consumption analysis remains relatively immature, with only vendor-specific tools like nvidia-NVML offering practical (but coarse) power metrics. This illustrates that while the methodology is theoretically sound, practical implementation across diverse GPU architectures remains a significant challenge.

PowerSensor3   *PowerSensor3*[51] is an open-source hardware tool introduced in 2025, designed to provide high-resolution power measurements for GPUs, SoC boards, PCIe devices, SSDs, and FPGAs. Unlike software-based power models or vendor-specific tools such as NVIDIA's NVML, PowerSensor3 achieves significantly higher accuracy and granularity through direct voltage and current measurements at a sampling rate of up to 20 kHz. This fine temporal resolution allows it to capture transient power behaviors that are typically missed by software-based methods, which are constrained by lower sampling frequencies and indirect estimations. As expected for a purpose-built hardware solution, PowerSensor3 outperforms NVML in both precision and the ability to detect rapid changes in power consumption.

A particularly valuable feature of PowerSensor3 is its capability to monitor not only GPUs but also other critical components such as SoC boards, PCIe-connected accelerators, and storage devices like SSDs. For Kubernetes-based energy efficiency analysis, this would provide unprecedented visibility into the power usage of individual containers, extending monitoring beyond the CPU and GPU to the broader spectrum of peripherals that contribute to overall energy consumption. Such granularity could enhance resource scheduling and energy optimization in containerized environments.

However, while its technical benefits are evident, the practical deployment of dedicated hardware sensors like PowerSensor3 at scale remains both complex and expensive. Integrating such devices across large Kubernetes clusters would require substantial investment in hardware and reconfiguration of infrastructure, making wide adoption unlikely outside of specialized research environments. Consequently, PowerSensor3 and other hardware-dependent methods are not considered in the scope of this thesis. Furthermore, the very recent introduction of PowerSensor3 in 2025 highlights the ongoing challenges of accurate energy monitoring through software alone, reflecting the current gap in reliable, scalable, software-based power measurement solutions.

2.4.3.4   GPU Limitations in Kubernetes Context

The analysis of GPU power consumption has revealed promising research efforts aimed at achieving fine-grained power visibility and energy optimization. Tools such as FinGraV and PowerSensor3 demonstrate that significant strides are being

made in capturing detailed power metrics with high temporal resolution and sub-component granularity. FinGraV addresses the complexities of short-lived GPU kernel executions through innovative profiling methodologies, while PowerSensor3 delivers hardware-level accuracy for GPUs, SoC boards, and various PCIe-connected peripherals. These solutions underscore the potential for more refined power monitoring in high-performance GPU workloads.

However, the current state of GPU energy consumption measurement presents significant challenges for scalable, container-based energy tracking in Kubernetes environments. Research tools like FinGraV and PowerSensor3, while technically robust, are either hardware-dependent or too tightly coupled to specific architectures (such as AMD's MI300X in the case of FinGraV). Hardware-based solutions like PowerSensor3, though highly accurate, are impractical for widespread deployment due to cost and scalability concerns. Meanwhile, software-based vendor solutions such as NVIDIA's NVML are far more accessible, but suffer from limitations in temporal granularity and measurement accuracy. These tools offer convenient integration and broad support across data center infrastructures but struggle with capturing rapid transients in power consumption, which are crucial for real-time container energy attribution.

In the context of this thesis, GPU energy consumption is acknowledged as an important yet currently impractical aspect of container energy measurement. The relative immaturity of fine-grained, scalable monitoring solutions for GPUs, combined with the relatively small role of GPUs in Kubernetes clusters, justifies this exclusion. Although the utilization of GPU accelerators in Kubernetes environments is expected to grow, current measurement methods do not yet support the level of precision and scalability required for effective implementation. As such, this thesis will focus on more readily measurable server components, with the understanding that future advancements in GPU power analysis may enable their integration into Kubernetes-based energy efficiency strategies.

2.4.4   Storage Devices

Various sudies have investigated the power consumption of Storage devices. In 2008, Hylick et al[52] investigated real-time HDD energy consumption and found significant differences in power consumption between standby, idle and active power states. Cho et al[53] propose various energy estimation models for SSDs after measuring and comparing the energy consumption of different models. The most notabble model-based energy consumption estimation methanisms are presented in section 2.5.4

In constrast to CPU or GPU components, storage devices (HDD, DDS or NVMe-drives) cannot make use of physical power sensors. While a BMC-measurement-based solution would technically be feasable, real-world implementation is impractical: While a BMC might me able to measure the power supply to a storage device, it typically is not exposed through IPMI or redfish. Such measurements would further be compllicated by the use of backplane devices, making measurements for individual devices impossible. For these reasons, storage device energy consumption is typically modelled, not measured (see section 2.5.4).

While storage devices don't expose any energy-consumption specific metrics, many other related metrics are available (and can be used for modeling approaches):

- `NVMe-cli`[54] exposes many metrics of NVMe-drives, including the maximum power draw for each power state (including idle power), the number of power states supported, the current power state and temperature, and others.

- `smartctl`[55] exposes metrics of the *SMART (Self-Monitoring, Analysis and Reporting Technology)*-Functionality implemented in many modern storage drives. While these metrics are vendor-specific, they often include temperature metrics, throughput, and other metrics. Often, HDD speed is exposed. Notably, *SMART* metrics are typically more focussed on lifecycle information such as power-on hours, wear indicators and others.

- Many other performance metrics are exposed by various tools such as `iostat`, `sar`, `/proc/diskstats`, and `blkstat`, such as read/write IOPS, throughput, queue length, latency, utilization, and others. Additional information (such as the interface) is also exposed.

2.4.5 Network devices and other PCIe devices

Peripherals like the Network Interace Card (NIC) are almost always connected via PCIe. As such, many cards support Device power states[56] as specified by the PCIe specifications. Notably, not all NICs support all (or any) power states.These device states allow the the server and device to negotiate a power state for the device, which typically means choosing a trade-off between power consumption and wake-up latency. For devices, PCIe specifies the following device states:

- D0 state (Fully on)

- D1 and D2 states (Intermediate power states)

- D3 State (Off State), with the distinction between D3hot and D3cold

Unfortunately, device power states are not in any way related to physical power specifications: While a specific power state might be useful for simple deductions (e.g. if a device is idling or active), no power figures can be deduced. In the event that a device's idle or maximum power are known, power states might potentially be used for a first estimation (i.e. an idling device is unlikely to consume its specified maximum power, and vice versa), but since a device's power characteristics cannot reliably estimated (especially beyond just NICs), devices power states cannot be used to reliably estimate device power consumption. An attempt at the estimation of NIC power consumption is covered in section 2.5.5.

2.5 Model-based estimation techniques

In the absence of actual power data, power consumption models can be formulated that map variables (such as CPU or 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[57] 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 introducing more complex non-linear models, such as Hsu and Poole[58], who studied the SPECpower_ssj2008-dataset of systems released between December

2007 and August 2010, and suggested the adaptation of two non-linear terms:

$$P_{\text{server}} = \alpha_0 + \alpha_1 u_{\text{cpu}} + \alpha_2 \left(u_{\text{cpu}}\right)^{\gamma_0} + \alpha_3 \left(1 - u_{\text{cpu}}\right)^{\gamma_1} \qquad (2.3)$$

The division of server power consumption into idle (generally static) and dynamic power (modeled with many different methods thoughought related research) has historically been a popular suggestion[59]. Other, broadly similar attempts to model server energy consumption based on only a few variables exist, such as modeling server consumption based on CPU frequency[60].

2.5.1   Component-level Power models

While models like the ones listed above might work well when custom-fitted to specific, multi-purpose 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[61] propose as:

$$P_{\text{server}} = P_{\text{cpu}} + P_{\text{memory}} + P_{\text{disk}} + P_{\text{NIC}} + C \qquad (2.4)$$

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.

2.5.1.1   Advantages and disadvantages of component-level power models

A component-based approach to modeling server power consumption offers increased granularity and adaptability across a diverse range of server architectures. Modern data centers deploy heterogeneous hardware configurations optimized for specific workloads, such as CPU-intensive computing nodes, GPU-accelerated servers for machine learning, or memory-rich systems for in-memory databases. These configurations lead to vastly different power distribution profiles across components[62]. By modeling the energy consumption of individual components, it becomes possible to reflect these structural differences more accurately. Additionally, such models can reveal energy characteristics that would be obscured in aggregate metrics (for instance, a workload that imposes significant stress on storage devices without engaging the CPU may go undetected in simplistic, CPU-centric models). Finally, component-level analysis enables more precise evaluation of energy optimization techniques: the impact of mechanisms like dynamic voltage and frequency scaling (DVFS) or idle power states can be assessed not just in isolation but in terms of their contribution to overall server efficiency.

Despite offering finer granularity, component-based power modeling faces several inherent challenges. While servers are composed of individual components, they function as tightly integrated systems in which no component operates in isolation. The power consumption of one subsystem often depends on the behavior of others (for example, memory access patterns can influence CPU power states, and I/O activity may trigger CPU wake-ups or increased cache usage). These inter-component interactions are difficult to capture accurately and are frequently overlooked in component-level models[13], leading to potentially misleading or incomplete estimations. Furthermore, the development of detailed and accurate models

for each component is significantly more complex than holistic server-level modeling. Such models often require extensive empirical data, sophisticated estimation techniques, and continuous updates to remain valid across hardware generations. This not only increases the research burden but also demands a higher level of expertise for interpretation and practical application, compared to simpler, utilization-based models.

2.5.2 CPU

Existing CPU power models generally model CPU power as a combination of other, existing power figures. Fan et al[57] propose a linear interpolation between idle power and full power based on CPU utilization. Basmadjian et al[63] observe that individual cores in a multi-core CPU can be modeled as individual cores, in addition to an overall CPU idle consumption. Non-linear models are also widely adopted, with Lou et al[64] proposing a polynomial model as a univariate function of CPU utilization. Other models include individual CPU components[65, 66] for more fidelity.

While these models may be helpful to examine the dynamics of CPU power consumption in relation to different inputs, they are not helpful in finding CPU power consumption of an unknown CPU, i.e. without previously known idle or maximum power consumption. The existance of a model capable to accurately estimation CPU power consumption based solely on generalizable input factors (such as utilization or frequency) is questionable due to the great variance in architectures and technologies, as well as technological progress. Unsurprisingly, the author of this thesis was not able to find a model to estimate CPU power consumption.

2.5.3 Memory

Many Memory power models were proposed in literature, many of them with an idle and a dynamic component of memory power consumption. While the idle power consumption is generally assumed to be known, dynamic memory power consumption has been modeled to depend on memory usage[67] or memory accesses[68], the number of cache misses[69], memory state[63], or other factors. Similar to the CPU models presented in the previous section, these models do not propose generalizable models able to predict memory consumption in situations where idle or maximum memory consumption is not previously known, instead focussing on examining power consumption dynamics. The same objections preventing generalizable CPU models apply to memory models as well, namely great variety, specialization and technological progress. As a result, no model-based approaches exist that are capable of accurately estimating memory power consumption based solely on performance metrics.

2.5.4 Storage devices

2.5.4.1 Generalization-based estimation

There is an urgent need in the storage industry for research into the area of workload-dependant power estimation[70]. Estimating the energy consumption of a storage device is challenging especially due to the great variation between different devices. Some of these model variables can be determined on a running server system (e.g. device type, I/O operation type, access pattern, workload intensity, state and more),

while other variables are unknown to the server (e.g. Flash Transition Layer and flash factor, NAND organization, garbage collection, and more). A large storage device market has led to a high variation in devices with sometimes drastically different target uses (e.g. low-latency storage devices, high-concurrency storage devices, low-power storage devices).

Storage controllers further complicate the energy consumption estimation of storage devices by introducing an additional layer of abstraction between the operating system and the physical storage hardware. Their internal operations consume energy independently of the actual read/write workload observed by the host system. This makes it difficult to directly correlate application-level I/O activity with actual device-level power usage. Moreover, in many server configurations, multiple drives are managed behind a single controller, obscuring per-device energy attribution and introducing variability that model-based estimations often cannot accurately capture.

Scope clarification   In this thesis, only storage devices physically installed in the server are considered for power estimation. This includes devices such as HDDs, SSDs, and NVMe drives directly attached to the server. Dedicated external storage systems such as Storage Area Networks (SAN) or Network-Attached Storage (NAS) are not within the scope of this analysis. While such systems are important in data center environments, their energy consumption is not attributable at the granularity required for the workload-level estimation pursued in this thesis.

As a result, research into storage device energy consumption measurement that is generally applicable to all devices has been limited. For practical applications, generalizations are often used, such as the following tables 2.3a to 2.3d. While these approximations cannot be used in the context of this thesis, they may serve as an initial guideline.

HDD Type	Read/write power (W)	Idle Power (W)	Standby Power (W)
HDD (2.5" SATA)	1.5 – 3.0	0.5 – 1.2	0.1 – 0.3
HDD (3.5" SATA)	6 – 12	4 – 8	0.5 – 2.0
HDD (Enterprise)	7 – 15	5 – 10	0.5 – 2.5

(A) Typical HDD power consumption[71]

HDD Type	Read/write power (W)	Idle Power (W)	Standby Power (W)
5400 RPM HDD	6 – 9	4 – 6	0.5 – 1.5
7200 RPM HDD	8 – 12	6 – 8	0.6 – 1.8
10,000+ RPM HDD	10 – 16	8 – 12	1.0 – 2.5

(B) Common HDD RPM power consumption[71]

SSD Type	Read Power (W)	Write Power (W)	Idle Power (W)
2.5" SATA	4.5 – 8	4.5 – 8	0.30 – 2
mSATA	1 – 5	4 – 8	0.20 – 2
M.2 SATA	2.5 – 6	4 – 9	0.40 – 2

(C) Typical SATA SSD power consumption[72]

NVMe Type	Read/write power (W)	Peak Power (W)	Standby Power (W)
M.2 NVMe PCIe 3.0	3 – 5	6 – 9	0.4 – 1.5
M.2 NVMe PCIe 4.0	5 – 7	8 – 12	0.5 – 2
M.2 NVMe PCIe 5.0	8 – 12	12 – 18	0.8 – 3

(D) Typical NVMe SSD power consumption[72]

TABLE 2.3: Power consumption for various storage device types.

Apart from simple estimations like shown in tables 2.3a to 2.3d, a few works have concentrated on the energy consumption of individual storage devices. In 2015, Cho et al developed Energysim[53], an SSD energy modeling framework avancing the understanding of component-level (i.e. the subcomponents of a storage device) energy consumption in storage devices. Its validation against real-world SSD measurements against an Intel X25-M yielded a less than 8% error. The work underscores the difficulty of modeling storage energy accurately due to high variability across architectures and workloads. Unfortunately, Energysim uses many model parameters such as NAND organization, idle and active current consumption, and as a result cannot reneralized to other storage devices where these are unknown.

In 2014, Li and Long[73] present a workload-aware modeling framework to estimate the energy consumption of storage systems, challenging the assumption that SSDs are inherently more energy-efficient than HDDs. By classifying I/O workloads into capability workloads (performance-driven) and capacity workloads (storage size-driven), they develop mathematical models that account for the number of devices needed, workload execution time, and device power states (active, idle, standby). Their validation, based on empirical measurements using Seagate HDDs and a Samsung SSD, shows that SSDs are generally more efficient for high-performance workloads, while HDDs can outperform SSDs in archival or low-access scenarios,particularly when effective power management (e.g., spin-down) is employed. Unfortunately,

similar to the research by Cho et al, the presented models make use of various non-generalizable variables, most notably a devices idle, standby and busy power consumption. In the context of these thesis, these are unknown and the presented model consequently cannot be applied.

2.5.4.2 GSPN Modeling for hybrid storage systems (active power states)

In 2022, Borba et al[74] proposed a number of models based on generalized stochastic Petri nets (GSPN) for performance and energy consumption evaluation for individual and Hybrid (HDD + SSD) storage systems. GSPN is a suitable formalism for storage system design, as, differently from queueing network models (for instance), synchronization, resource sharing, and conflicts are naturally represented. Also, phase approximation technique may be applied for modeling non-exponential activities, and events with zero delays (e.g., workload selection) may adopt immediate transitions.

The authors propose a single-storage model (either for a single storage device or a hybrid system as a blackbox) and a multiple storage model.

The Hybrid storage power consumption model proposed by Borba is parameterized by I/O type (read/write), access pattern (sequential/random), object size (4KB, 1MB), and thread concurrency. The model explicitly imcorporates power consumption per operation (e.g. random-read-4KB on SSD).

The following notation is adoped:

- $E\{\#p\}$ represents the mean value of the inner expression, in which $\#p$ denotes the number of tokens in place.

- $W(T)$ represents the firing rate associated with transition T.

- $\eta : T_{\text{imm}} \rightarrow [0, 1]$ maps each immediate transition ($t \in T_{\text{imm}}$) to a normalized weight. Weights represent the transition firing probablilty in a conflict set.

- $pRequests(N)$ denotes the amount of concurrent requests from simultaneous clients (workers)

Single-device storage energy consumption is estimated as follows:

$$EP_w = \kappa \cdot (EP_{w1} \cdot \alpha \cdot \beta + EP_{w2} \cdot (1 - \alpha) \cdot \beta$$
$$+ EP_{w3} \cdot \alpha \cdot (1 - \beta) + EP_{w4} \cdot (1 - \alpha) \cdot (1 - \beta)) \qquad (2.5)$$

$$EP_r = (1 - \kappa) \cdot (EP_{r5} \cdot \alpha \cdot \beta + EP_{r6} \cdot (1 - \alpha) \cdot \beta$$
$$+ EP_{r7} \cdot \alpha \cdot (1 - \beta) + EP_{r8} \cdot (1 - \alpha) \cdot (1 - \beta)) \qquad (2.6)$$

$$EC = (EP_w + EP_r) \cdot TH \cdot \text{time} \qquad (2.7)$$

where EP_w and EP_r are the mean power consumption for a read (r) or write (w) operation, which is estimated using the mean power of each workload feature. For instance, EP_{w1} denotes the power of a write operation (w) using random access (α) and a small object (β). System throughput (i.e., IOPS) is estimated as $TH = E\{\#p_{\text{Ack}}\} \times W(t_{\text{Communicating}})$.For the single device model, the following weights are taken into

account: $\eta(t_{\text{Write}}) = \kappa$; $\eta(t_{\text{Read}}) = 1 - \kappa$; $\eta(t_{\text{Random}}) = \alpha$; $\eta(t_{\text{Sequential}}) = 1 - \alpha$; $\eta(t_{\text{Small}}) = \beta$; and $\eta(t_{\text{Large}}) = 1 - \beta$.

The marking of place $pResource(R)$ (for both read or write activity) may denote the adopted technology. For instance, for traditional SSDs (SATA interface), the marking place $pResource$ is 1, as only one operation it the time is carried out. Concerning SSDs-NVMe, $pResource$ assumes the number of threads of concurrently processing I/O requests (generally 8).

The proposed multi-storage model expands the model for multiple devices:

$$EC_h = \left(\sum_{d=0}^{n} \eta(tForward_d) \cdot EP_d \right) \cdot TH_h \cdot \text{time} \tag{2.8}$$

where the immediate transitions $tForward_d$ denote a request redirection to storage d.

Validation   The model proposed by Borba et al. was validated using controlled experiments with the Fio benchmarking tool, which generated synthetic I/O workloads to measure and correlate storage system performance and energy consumption across varying request sizes, access patterns, and read/write ratios. Model estimates consistently falling within the 95% confidence intervals of observed system metrics. This statistical consistency indicates that the model's predictions are not significantly different from real-world values, supporting its applicability for performance and energy analysis in large-scale storage systems.

Limitations   The authors acknowledge that a large number of devices significantly increases modeling complexity due to state space size explosion and recommend simulation as a viable workaround. Additionally, the authors acknowledge their focus on active energy states (not idle, standby states or state transitions), treating them as delays between requests.

In a running server system, this approach could be adapted to create an accurate and fine-grained energy consumption estimation of a read/write workload on specific storage devices, albeit with limitations:

- Instead of needing to be estimated, (device-specific) throughput (TH) can be measured.

- An intial calibration run is necessary to experimentally determine the respective device-specific variables.

- In a multi-storage device server, the resulting state explosion may lead to significan calculation overhead, resulting also in higher energy consumption of the measurement itself.

- Instead of modelling transitions to a storage device as a function (as done in $\eta(tForward_d)$), device usage would actively need to be measured, which would essentially transform the multi-storage model into a simple addition of single-storage models. This would drastically reduce the number of total states, making calculations less demanding.

- Due to the authors not considering idle and standby-states, a small, constant idle power consumption would need to be added to the model. This is especially important for accurate storage device power consumption modeling on idling or overprovisioned servers.

2.5.5   Network devices

Estimating the total power consumption of a network infrastructure requires a clear definition of system boundaries. Since most server clusters operate within larger, interconnected systems, a full assessment of network energy consumption (such as for CO_2 footprint calculations) is generally infeasible. This thesis limits the system boundary to the server itself, considering only internal network components, primarily the Network Interface Card (NIC). While this allows detailed modeling of NIC power usage, it excludes broader network activity, such as inter-node communication in multi-node clusters.

Although the overall energy consumption of a data center network could be estimated by including access, aggregation, and core switches, attributing this consumption to specific workloads remains highly challenging. This chapter therefore focuses on model-based methods for estimating NIC-level power as a proxy for server-side network energy usage.

2.5.5.1   NIC power consumption characteristics

While a lot of research was done to analyze the power consumption of network equipment like switches, routers or gateways, NICs (especially non-wireless NICs) have not received as much attention. While there are several methods that modern NICs employ to save power (e.g. PCIe Link power states and D-states, *Active State Power Management (ASPM)*) or *Energy Efficient Ethernet (EEE)*, there are not widely available mechanisms for fine-grained NIC power consumption estimation. As a consequence, NIC power can only be approximated based on the few available metrics,

Sohan et al[75] measured and compared the power consumption of six 10 Gbps and four multiport 1 Gbps NICs at a fine-grained level. While he does not provide a method to estimate NIC energy consumption, and notices great variation in power consumption between different NICs. Unfortunately, it cannot be ruled out that some of the results are cherry-picked: Solarflare-NICs tend to dominate the introduced metics, and a Communication spokesperson is prominently credited with contact information. Regardless, some findings are found irrespective of the NIC manufacturer, and remain consistent with other literature sources[76]. While these findings cannot directly contribute to a potential NIC power consumption estimation approach, they are relevant to understand underlying mechanisms and to assess the relative importance of the NIC compared to other server components.

- Idle Power

    - The measured NICs show a power consumption of between 5–20W

    - Link connection status has little effect on idle energy consumption

- – Physical media influences power consumption: CX4 models have the lowest power consumpttion due to the simple design of the CX4 interconnect. This is followed by Fober models. Finally the Base-T models consume significantly more power due to the signal processing component int he card.

- Active Power

  - – There is very little difference in the power usage of an active NIC compared to an idle one. For all measured NICS, the differenc in power usage was less than 1W.

  - – Throughput performance varied widely, and no correlation between power usage and performance was observed.

  - – Power consumption increases in correlation to the number of ports.

In 2012, Basmadjian et al[77] modelled a NIC by separating NIC power consumption into idle mode and dynamic mode (same as they did for their CPU and RAM models). If $P_{NIC_{idle}}$ is the power of the idle interface and $P_{NIC_{dynamic}}$ is the power when active, the total NIC energy consumption would be given by

$$E_{NIC} = P_{NIC_{idle}} T_{idle} + P_{NIC_{dynamic}} T_{dynamic} \tag{2.9}$$

where T_{idle} and $T_{dynamic}$ are the total idle and dynamic times, respectively. Consequently, the average power during period T would be given by

$$P_{\text{NIC}} = \frac{(T - T_{\text{dynamic}}) P_{NIC_{idle}} + P_{NIC_{dynamic}} T_{\text{dynamic}}}{T} \tag{2.10}$$

$$= P_{NIC_{idle}} + (P_{NIC_{dynamic}} - P_{NIC_{idle}}) \rho \tag{2.11}$$

where $\rho = \frac{T_{\text{dynamic}}}{T}$ is the channel utilization. While this formula is only helpful when NIC idle and max power consumption are already known, we can additionally see that the NIC power consumption is assumed to rise linearly with channel utilization.

Arjona Aroca et al[78] modeled NIC efficiency based on their previous measurements. They find that NIC efficiencies for both sending and receiving are almost linear with the transfer rate and deduct a linear dependency to the network throughput.

In 2016, De Maio et al[79] proposed a network energy consumption model for node-to-node transfers in order to estimate the entire energy consumption required for virtual machine migration. Unfortnuately their model does not specifically handle NIC energy consumption, opting for modelling the entire node energy consumption of the node.

Another approach is presented by Dargie and Wen[80] in 2013, who use stochastic modelling to examine the relationship between the utilization of a NIC and its power consumption, expressing these quantities as random variables or processes. They use curve fitting to determine the relationship between utilization and measured energy consumption of their specific NIC, after creating a data set using a SPECpower benchmark. They assume a uniformly distributed bandwidth utilization in the interval [0,125] MBps. Interestingly, their model shows only a slight effect

of utilization on the predicted power consumption, mirroring the findings of Sohan et al.

The most recent NIC power model was proposed by Baneshi et al[81] in 2024, analyze per-application energy consumption. The authors cite that NIC idle power consumption may contribute up to 90% of the total NIC energy consumption. They propose the following model for per-application NIC power consumption:

$$E_{\text{active}} = \sum_i \left(BW_i \cdot T_{\text{interval}} \cdot \frac{P_{\text{max}} - P_{\text{idle}}}{BW_{\text{aggregated}}} \right) \tag{2.12}$$

$$E_{\text{idle}} = \sum_i \left(BW_i \cdot T_{\text{interval}} \cdot \frac{P_{\text{idle}}}{BW_{\text{used}}} \right) \tag{2.13}$$

where BW_i is the bandwidth of application i, $BW_{\text{aggregated}}$ is the aggregate bandwith of both the uplink and downlink of the NIC, and BW_{used} is the used bandwidth of links (uplinks, downlinks or both). The authors combine these formulae with power figures of their specific use case (Total network power consumption in a fog computing scenario), which unfortunately are not applicable in the context of this thesis. Regardless, while these formulae cannot be used to estimate the maximum and idle NIC power, they can be applied irrespective of server specifications in the event that idle and maximum power NIC power consumption are known.

In contrast to the formulae presented by Basmadjian and Arjona Aroca, the formulae not only account for time intervals, but also bandwidth used. As a result, the formulae presented by Baneshi et al represent the current best approach to estimate NIC power consumption, even though this estimation still requires an initial guess of the idle and maximum power consumption. The author of this thesis is not aware of a more detailed formuala currently available. A generalizable formula for estimating overall NIC power consumption is unlikely to exist due to the vast variety of NICs, and the great differences between manufacturers (as found by Sohan et al).

2.5.6   Other devices

While much of the research on server energy consumption focuses on primary components such as the CPU, memory, storage, and network interfaces, a complete energy model must also account for additional hardware subsystems. These include the motherboard, power supply unit (PSU), system fans, and potentially other auxiliary devices. Though their individual energy consumption may appear minor compared to high-performance components, they collectively contribute a non-negligible share to the overall server power draw.

Despite their importance, these secondary components have received limited attention in energy modeling literature. In most cases, they are either omitted or treated as part of the residual power not attributable to the main computational subsystems.

The motherboard, for instance, includes voltage regulators, chipset logic, and peripheral interfaces. While these components may not be individually monitored, the Baseboard Management Controller (BMC) (see section 2.3.2.1)may expose aggregated power telemetry via vendor-specific sensors or interfaces like IPMI or Redfish. However, this level of detail varies greatly across hardware platforms and is seldom fine-grained enough for component-level attribution.

2.5.6.1 PSU

The power supply unit (PSU) is another often-overlooked consumer. When modeling the power usage of a PSU itself (distinct from the power it delivers to other components), the key factor is its conversion efficiency. PSUs consume more power than they deliver due to losses during AC–DC transformation and voltage regulation. According to Basmadjian et al. [77], the power consumed by the PSU can be approximated for various scenarios: If the monitoring system provides information at the PSU level, its power consumption is given by

$$P_{PSU} = \frac{measuredPower \cdot (100 - e)}{100} \tag{2.14}$$

where e is the efficiency of the PSU.

If the monitoring system proves information at the server level, the power consumption of any of the n PSU is given by the following formuala, assuming that measured power is evenly distributed among PSUs:

$$P_{PSU} = \frac{\frac{measuredPower}{n} \cdot (100 - e)}{100} \tag{2.15}$$

If the monitoring system does not provide PSU power consumption, it can be deduced by

$$P_{PSU} = \frac{P_{Mainboard} + P_{Fans}}{n \cdot e} \cdot 100 - \frac{P_{Mainboard} + P_{Fans}}{n} \tag{2.16}$$

2.5.6.2 Fans

Cooling systems, particularly fans, also represent a meaningful share of the total energy budget. Most servers employ multiple fans controlled via Pulse-Width Modulation (PWM). While the BMC or operating system tools (e.g. `lm-sensors`) often report fan RPM or PWM duty cycle, actual fan power consumption is rarely exposed directly. Furthermore, RPM alone is insufficient to estimate power accurately, as fan power depends on physical factors such as the fan diameter, pressure increase or air flow delived[77]:

$$P_{Fan} = d_p \cdot q = \frac{F}{A} \cdot \frac{V}{t} = \frac{F \cdot d}{t} \tag{2.17}$$

where p_d denotes total pressure increase of the fan (Pa or N/m^2), q denotes the air volume flow (m^3/s), F denotes force (N), A denotes fan area (m^2), V denotes volume (m^3), and t denotes time (seconds).

Based on observations, F is proportional with to the sqare of RPM. This can be combined with formula 2.17:

$$P_{Fan} = \frac{c \cdot RPM^2 \cdot d}{3600} \tag{2.18}$$

where for each individual fan, $c = \frac{3600 \cdot P_{Max}}{RPM_{Max}^2 \cdot d}$ remains constant.

Unfortunately, with the wide variety of fans in servers (especially with fan size restrictions due to server heights), these formulae are only helpful when paired with more detailed information on fan characteristics like Maximum RPM and Power.

While these can more reasonably be assumed, this remains a rough estimation at best.

2.5.6.3   Attribution of secondary component power consumption to individual workloads

Despite these measurement limitations, estimating the energy consumption of secondary components is often less critical for attributing energy to workloads. This is because components like fans, mainboards, and PSUs primarily support the operation of primary subsystems. Their power consumption scales with the activity level of CPU, memory, disk, and networking devices: more computation leads to higher heat dissipation, increased power delivery, and thus greater fan and PSU activity.

Consequently, if the total server power consumption is known—for example, via wall power monitoring or BMC/Redfish readings—the residual power (i.e., total power minus the sum of measured CPU, RAM, disk, and network power) can be reasonably attributed to power delivery and thermal management subsystems. This residual can then be proportionally distributed among active workloads based on the power consumption of the primary components they utilize. In this context, fine-grained modeling of secondary components becomes unnecessary for workload attribution, as their energy use correlates closely with that of the primary subsystems they support.

2.5.7   Issues with model-based power estimation techniques

This thesis pursues two inherently conflicting objectives: achieving high-resolution, accurate energy consumption measurements while simultaneously developing a solution that remains broadly applicable across heterogeneous server environments without requiring extensive manual calibration, or the manual input of complex device-specific information. Striking a balance between these goals is particularly challenging in the context of model-based energy estimation for devices that do not expose power telemtry data: Due to the wide variability among devices for a variety of factors, energy consumption models must necessarily abstract away much of the underlying complexity. Developing a model that is simultaneously fine-grained, highly accurate, and universally applicable across different technologies is, in practice, an unattainable goal.

As a result, any model integrated into a general-purpose energy estimation framework must err on the side of relative simplicity to preserve generality. While this approach diminishes the precision of device-specific energy attribution, it remains valuable for broader optimization tasks. For instance, autoscaling mechanisms, load balancers, and schedulers can still benefit significantly from approximate energy profiles. Likewise, cluster administrators aiming to improve energy efficiency holistically, or developers seeking to optimize their workloads, can gain useful directional insights even from coarse-grained models.

However, this simplicity imposes significant limitations for use cases that require device-specific energy optimization. General-purpose models are ill-suited for evaluating the energy efficiency of different device types, testing firmware-level adjustments, or validating the impact of power-saving features such as low-power states. In such scenarios, the model's abstraction may not just be insufficient, but actively misleading.

In the context of this thesis, this limitation is considered acceptable. The overarching objective is to facilitate scalable and portable energy estimation mechanisms for containerized environments, not to provide a diagnostic tool for hardware-level energy analysis. Nonetheless, this constraint should be kept in mind when interpreting the results and assessing their suitability for device-centric evaluation tasks.

2.6 Power Modeling based on Machine Learning Algorithms

In a taxonomy of power consumption modeling approaches, Lin et al[13] analyze various machine learning-based power models in current literature, categorizing them into supervised, unsupervised, and reinforcement learning. A detailed reiteration of this taxonomy (as well as a methodological overview of machine learning and neural networks) is omitted here.

In the context of this thesis, machine learning-based approaches are not considered for the following reasons:

- The author was unable to identify a promising and reliable approach that is sufficiently generalizable to function across a wide range of server configurations. Likewise, no component-level models were found that met these criteria. While it is certainly possible to train machine learning models to estimate energy consumption for specific server setups with high accuracy, the aim of this thesis is to provide generalizable estimation methods applicable to varied systems.

- Machine learning fundamentally relies on large datasets that are both highly accurate and granular, ideally matching the quality expectations of the resulting model. As discussed in previous sections, such high-quality training data is rarely available in the domain of fine-grained power measurement. When such datasets do exist, they typically reflect highly specific hardware and workload configurations, making them unsuitable for generalization. Although it would be theoretically possible to generate a large dataset by systematically benchmarking thousands of CPUs, memory modules, GPUs, storage, and network devices across millions of configurations, such an undertaking is not practically feasible. Furthermore, any such dataset would require ongoing expansion to remain representative of new hardware generations.

- Finally, many of the technical implementation details underlying key telemetry features remain proprietary or undocumented. A notable example is the RAPL interface, whose internal workings are not publicly disclosed. At the same time, existing RAPL metrics already offer abstracted energy readings suitable for direct integration into power estimation tools, eliminating the need for an intermediate machine learning-based step.

In theory, machine learning-based power estimation models hold significant promise and may one day be realized. Such models could leverage complex, nonlinear relationships between hardware components and workloads—relationships that are inherently difficult or even impossible to capture through traditional analytical models. The predictive and adaptive capabilities of machine learning offer the potential for highly accurate, fine-grained estimations across a broad range of configurations. However, as of today, the development of such a comprehensive and generalizable

model remains out of reach. Realizing this vision would require extensive collaboration between original equipment manufacturers, cloud providers, data center operators, and research institutions to generate, standardize, and share high-quality telemetry data across diverse hardware and workload scenarios. Given the role of cloud computing in the current economic landscape, where proprietary knowledge and performance optimization constitute a competitive advantage at the corporate, national, and geopolitical levels, such broad cooperation appears unlikely. Consequently, machine learning remains a promising but currently impractical direction for universal server power modeling.

2.7 Container energy estimation based on hardware power estimation

xxxxxxxxxxxxxxxxxxxx

2.8 Component-specific summaries

This section offers practical guidelines for measuring or estimating the energy consumption of individual hardware components. While earlier chapters focused on theory and research, the focus here is on implementation: what works best, what challenges to expect, and how to improve accuracy with minimal effort. For each component, the most accurate method is highlighted, along with alternatives and fallback options. Simple improvements like entering datasheet values or running calibration workloads are discussed where relevant.

2.8.1 CPU

The most accurate and widely adopted method for measuring CPU energy consumption is Intel's RAPL interface. It offers high temporal resolution, low overhead, and requires no external hardware. Among the available interfaces, `perf-events` is generally recommended due to its balance between usability, performance, and access control. The `powercap` interface offers simpler integration via `sysfs`, though with some limitations in domain structure and overflow handling. MSR access is discouraged due to complexity and privilege requirements. eBPF-based methods are powerful and used in advanced tools, but introduce high development complexity, kernel dependency, and lower portability.

On AMD systems, the `amd_energy` driver offers a RAPL-compatible interface, but it exposes fewer domains (e.g., no DRAM) and is generally less feature-rich than Intel's implementation.

Despite some limitations (such as non-atomic register updates, idle power inaccuracies, and the absence of timestamps), RAPL is considered sufficiently accurate for both research and production use. Its drawbacks can often be mitigated through careful sampling strategies (e.g., overflow-safe polling intervals, timestamp alignment), and correction techniques for overflow and measurement jitter.

ACPI, while historically relevant, does not expose real-time power data and is unsuitable for precise energy measurement. Although some theoretical estimation

based on P-states is possible, it is coarse-grained and impractical for modern CPUs with dynamic frequency scaling.

If RAPL is unavailable, statistical models based on utilization, frequency, and other metrics may be used. However, these require prior calibration or hardware-specific profiling. Without access to idle or peak power values, such models become highly unreliable due to architectural variability. In such cases, estimation accuracy can be modestly improved by inserting static power values from processor datasheets or using fixed coefficients for known CPU families.

2.8.1.1 Container-level implications

RAPL's granularity and domain separation make it suitable for correlating CPU energy usage with container activity, allowing for reasonably accurate attribution when combined with CPU usage metrics (e.g., cgroup CPU accounting or eBPF). In contrast, model-based estimations or ACPI-derived values are too coarse and lack temporal resolution, limiting their use to static or linear power distribution based on workload share, insufficient for fine-grained or bursty container workloads.

2.8.2 Memory

Memory power consumption can be measured using the DRAM domain exposed by Intel RAPL on supported server-grade processors. When available, this provides low-overhead, fine-grained energy telemetry integrated with other CPU domains. However, DRAM measurement accuracy depends heavily on processor architecture. It is generally reliable for Haswell-generation CPUs, but later architectures may exhibit a constant power offset or measurement inaccuracies due to off-DIMM voltage regulators and evolving memory subsystems.

If RAPL DRAM telemetry is unavailable or deemed unreliable, no equivalent in-band method exists. In such cases, estimation must rely on model-based approaches. Many models in literature attempt to correlate memory power with usage, memory access frequency, or cache behavior, but they are not generalizable across systems. Most require prior calibration using known idle and peak memory power figures, which are rarely available in practice. Without these, estimation accuracy remains low. Manual insertion of idle and active power values from vendor datasheets can slightly improve results, but still yields only coarse-grained estimates.

2.8.2.1 Container-level implications

The RAPL DRAM domain, when accurate, allows correlation between energy consumption and workload-level memory metrics such as usage or memory bandwidth. This enables container-level attribution if per-container memory activity is available. Without RAPL, model-based estimates only support static or proportional energy attribution based on usage share, which is insufficient for capturing the energy impact of memory-intensive or bursty workloads.

2.8.3 GPU

Accurate GPU power measurement remains a challenge in containerized environments. The most accessible solution is NVIDIA's NVML interface (e.g., via nvidia-smi), which exposes power metrics through on-board sensors. While widely used,

NVML suffers from sampling delays, averaging artifacts, and limited temporal resolution—especially during transient workloads. Nevertheless, it offers acceptable accuracy for steady-state measurements and is supported across many data center deployments.

Alternative tools, such as AccelWattch and FinGraV, provide finer temporal granularity and more precise modeling, but are either architecture-specific or tightly coupled to particular hardware (e.g., AMD MI300X). Hardware-based solutions like PowerSensor3 achieve excellent accuracy at high sampling rates but are cost-prohibitive and impractical for large-scale deployment. No general-purpose, software-only solution currently matches the accuracy and portability of CPU-side tools like RAPL.

2.8.3.1   Container-level implications

GPU power attribution in Kubernetes is limited by the granularity and accuracy of current tools. While NVML can be queried from within containers or sidecars, it does not natively support multi-tenant attribution, and virtualization layers (e.g., vGPU, MIG) complicate per-container visibility. Accurate, container-level GPU energy tracking remains an open problem, requiring either architectural integration (e.g., with MIG-aware scheduling) or improved temporal sampling. As such, GPU measurements are currently only viable for coarse-grained, workload-level profiling—not fine-grained container energy attribution.

2.8.4   Storage devices

Storage device energy consumption is typically estimated rather than measured. Unlike CPUs or GPUs, storage devices lack onboard power sensors, and BMC-based per-device readings are generally unavailable, especially when using backplanes, RAID controllers, or SATA interfaces. Consequently, power usage is inferred from device metrics and modeled behavior.

Telemetry is only available for specific device types. For NVMe drives, `nvme-cli` exposes detailed metrics such as supported power states, current power state, idle/active power ratings, and temperature. However, these are not available for SATA SSDs or HDDs. `smartctl` provides vendor-specific SMART data (e.g., temperature, power-on hours, wear) if available, but energy-related insights are limited. Standard Linux tools (`iostat`, `sar`, `/proc/diskstats`, etc.) expose generic performance counters such as IOPS, throughput, queue length, and utilization, which can support rough estimation.

Various model-based approaches estimate power using activity-based metrics (e.g., read/write rates or interface speed), often requiring idle and active power values from datasheets. These models are only accurate when tailored to specific hardware. No general-purpose estimator exists for unknown or heterogeneous storage types without prior calibration.

2.8.4.1   Container-level implications

Because disks are shared resources, and per-container telemetry is unavailable, energy attribution must rely on proportional estimation using observable metrics like I/O volume or latency. This approach works for long-lived workloads but lacks the granularity to capture energy dynamics of bursty or short-lived container activity.

Accurate container-level attribution remains infeasible for SATA SSDs and HDDs and is only marginally better for NVMe devices, assuming access to detailed device metrics.

2.8.5 Network devices

NIC power consumption cannot be measured directly via software. Although many cards support PCIe power states (e.g. D0–D3), these states only approximately correlate with actual power draw and are not sufficient for energy estimation. Furthermore, NICs lack onboard power sensors, and BMC-based per-device readings are generally unavailable. As such, NIC energy consumption must be estimated using model-based approaches.

Various research models estimate NIC power using idle and dynamic components. The most promising approach, proposed by Baneshi et al., linearly scales NIC power with bandwidth utilization, assuming idle and maximum power values are known. While earlier models correlate energy use with channel utilization or throughput, they often oversimplify or lack generalizability. Real-world measurements show minimal power variation between idle and active states (often <1W difference), with idle power dominating overall NIC energy use. Estimates can be improved slightly by incorporating known idle and peak wattage from datasheets, but generalization across different NICs remains unreliable due to architectural and vendor variability. In the absence of these values, the only remaining option is to guess these values based on NIC PHY medium.

Telemetry support is limited: tools like `ethtool` expose link speed and status, but do not report power. No standard Linux tool provides direct NIC energy metrics, and throughput-based estimators must rely on indirect metrics like bytes transmitted per interval.

2.8.5.1 Container-level implications

Because NICs are shared across containers and lack per-container telemetry, only indirect attribution is possible. Energy consumption can be distributed proportionally based on container-level bandwidth usage (e.g., via cgroup network statistics), assuming idle and peak NIC power are known. However, the minimal dynamic variation in NIC power limits the usefulness of fine-grained attribution. In practice, NIC power is best modeled as a mostly static overhead, with marginal gains from utilization-based scaling.

2.8.6 Other Devices

Secondary components such as the motherboard, PSU, and fans contribute a non-trivial share to total server power consumption but are rarely modeled with precision. These devices typically lack direct power telemetry, and their energy use is either approximated or inferred indirectly.

PSU losses can be estimated from efficiency ratings if total input or output power is known. Fan power is difficult to measure and depends on physical factors like airflow and pressure; at best, it can be roughly estimated using RPM and vendor data. The motherboard and onboard controllers (e.g., voltage regulators, chipset) are usually modeled as part of residual power.

2.8.6.1 Best-practice approach

If system-level power data is available (e.g. via IPMI or Redfish), the difference between total server power and known component estimates can be treated as residual power. This residual can be linearly distributed across containers based on the finer-grained power estimation of the CPU, assuming secondary device power scales with primary component power consumption.

2.8.6.2 Container-level implications

Because these components do not map directly to container usage, their energy must be attributed indirectly. Linear distribution based on known, container-attributed metrics (e.g., CPU time or workload duration) is a practical, though imprecise, fall-back for ensuring full power accounting in containerized environments.

Chapter 3

Correlating Power Consumption to Containers

Chapter 3: Correlating Power Consumption to Containers 3.1 The Challenge of Power Attribution in Shared Systems 3.2 What Makes a Good Correlation? Accuracy Temporal granularity Causality vs. correlation Real-time vs. post-processing trade-offs 3.3 Power Attribution Philosophies Usage-based, proportional, fixed-share, hybrid Dealing with idle power Dealing with system containers / background processes 3.4 Contextual Use Cases Cost/billing Sustainability / carbon reporting Resource optimization SLA verification / fairness in multi-tenant environments 3.5 Limitations and Practical Considerations Metric availability (AMD/ARM, vendor lock-in) Interference and contention Metric resol ution and sync Scheduler behavior Measurement overhead

Chapter 4

Existing Tools and Approaches

4.1 Overview of Tool Landscape KEPLER, Scaphandre, CodeCarbon, PowerAPI, Cloud Carbon Footprint, etc. 4.2 Tool Analysis Framework Accuracy, data sources, correlation method, platform support, etc. 4.3 Detailed Evaluation of Selected Tools One subchapter per tool: 4.X KEPLER 4.X Scaphandr ... 4.4 Comparison Summary Table of tradeoffs Strengths and weaknesses Missing features / open gaps

Chapter 5

Designing a Container Power Attribution Architecture

5.1 Design Goals Generalizability, minimal overhead, real-time capability, etc. 5.2 Architecture Components Metric sources, data aggregation, correlation layer, exporter 5.3 Attribution Logic CPU (RAPL + cgroups), RAM, NET, DISK, idle, system 5.4 Proposed Correlation Model Hybrid models (e.g. direct for CPU, proportional for network) 5.5 Open Questions and Future Improvements

Chapter 6

Methodology for Evaluation and Benchmarking

6.1 Requirements for Evaluation Ground-truth measurement, reproducibility, load generation 6.2 Testbed Design Kubernetes cluster layout, benchmarking tools (e.g., stress-ng, fio, iperf3) 6.3 Benchmarking Scenarios Synthetic benchmarks (CPU/memory-heavy, IO-heavy, mixed) Real-world workloads (web apps, ML inference, etc.) 6.4 Evaluation Metrics Attribution accuracy, overhead, scalability, stability

Chapter 7

Conclusion and Outlook

7.1 Summary of Findings 7.2 Implications for Practice 7.3 Limitations of the Study 7.4 Outlook on Future Work

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". In: *Procedia Computer Science* 171 (2020), pp. 1419–1428. ISSN: 1877-0509. DOI: 10.1016/j.procs.2020.04.152.

[8] Roberto Morabito. "Power Consumption of Virtualization Technologies: An Empirical Investigation". In: *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*. Dec. 2015, pp. 522–527. DOI: 10.1109/UCC.2015.93. (Visited on 05/21/2025).

[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 (Apr. 2022), p. 111848. ISSN: 0378-7788. DOI: 10.1016/j.enbuild.2022.111848. (Visited on 04/20/2025).

[11] Chaoqiang Jin et al. "A Review of Power Consumption Models of Servers in Data Centers". In: *Applied Energy* 265 (May 2020), p. 114806. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2020.114806. (Visited on 03/16/2025).

[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)*. Sept. 2024, pp. 76–80. DOI: 10.1109/CLUSTERWorkshops61563.2024.00020. (Visited on 04/20/2025).

[13] Weiwei Lin et al. "A Taxonomy and Survey of Power Models and Power Modeling for Cloud Servers". In: *ACM Comput. Surv.* 53.5 (Sept. 2020), 100:1–100:41. ISSN: 0360-0300. DOI: 10.1145/3406208. (Visited on 04/20/2025).

[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*. MEMSYS '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pp. 455–470. DOI: 10.1145/2989081.2989088. (Visited on 05/21/2025).

[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)*. June 2016, pp. 648–655. DOI: 10.1109/CLOUD.2016.0091. (Visited on 04/23/2025).

[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. ISSN: 1532-0634. DOI: 10.1002/cpe.5124. (Visited on 04/23/2025).

[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*. Portland OR USA: ACM, July 2020, pp. 112–119. ISBN: 978-1-4503-6689-2. DOI: 10.1145/3311790.3396624. (Visited on 04/23/2025).

[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*. June 2019, p. 1. (Visited on 04/23/2025).

[20] UEFI Forum. *Advanced Configuration and Power Interface Specification Version 6.6*. Accessed April 2025. Sept. 2021. URL: https://uefi.org/sites/default/files/resources/ACPI_Spec_6.6.pdf.

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

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

[23] Robert Schöne et al. "Energy Efficiency Aspects of the AMD Zen 2 Architecture". In: *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. Sept. 2021, pp. 562–571. DOI: 10.1109/Cluster48925.2021.00087. (Visited on 04/28/2025).

[24] 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)*. May 2023, pp. 106–118. DOI: 10.1109/CCGrid57682.2023.00020. (Visited on 04/21/2025).

[25] Tom Kennes. *Measuring IT Carbon Footprint: What Is the Current Status Actually?* June 2023. DOI: 10.48550/arXiv.2306.10049. arXiv: 2306.10049 [cs]. (Visited on 04/23/2025).

[26] 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* 36.1 (Jan. 2025), pp. 96–107. ISSN: 1558-2183. DOI: 10.1109/TPDS.2024.3492336. (Visited on 04/02/2025).

[27] 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.

[28] 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*. London United Kingdom: ACM, May 2024, pp. 95–106. ISBN: 979-8-4007-0444-4. DOI: 10.1145/3629526.3645040. (Visited on 04/07/2025).

[29] Daniel Hackenberg et al. "An Energy Efficiency Feature Survey of the Intel Haswell Processor". In: *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. May 2015, pp. 896–904. DOI: 10.1109/IPDPSW.2015.70. (Visited on 04/28/2025).

[30] 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)*. Apr. 2013, pp. 194–204. DOI: 10.1109/ISPASS.2013.6557170. (Visited on 04/28/2025).

[31] Lukas Alt et al. "An Experimental Setup to Evaluate RAPL Energy Counters for Heterogeneous Memory". In: *Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering*. ICPE '24. New York, NY, USA: Association for Computing Machinery, May 2024, pp. 71–82. ISBN: 979-8-4007-0444-4. DOI: 10.1145/3629526.3645052. (Visited on 04/02/2025).

[32] Kashif Nizam Khan et al. "RAPL in Action: Experiences in Using RAPL for Power Measurements". In: *ACM Trans. Model. Perform. Eval. Comput. Syst.* 3.2 (Mar. 2018), 9:1–9:26. ISSN: 2376-3639. DOI: 10.1145/3177754. (Visited on 04/07/2025).

[33] Marcus Hähnel et al. "Measuring Energy Consumption for Short Code Paths Using RAPL". In: *SIGMETRICS Perform. Eval. Rev.* 40.3 (Jan. 2012), pp. 13–17. ISSN: 0163-5999. DOI: 10.1145/2425248.2425252. (Visited on 05/21/2025).

[34] "Detailed and Simultaneous Power and Performance Analysis - Servat - 2016 - Concurrency and Computation: Practice and Experience - Wiley Online Library". In: (). (Visited on 05/21/2025).

[35] Moritz Lipp et al. "PLATYPUS: Software-based Power Side-Channel Attacks on X86". In: *2021 IEEE Symposium on Security and Privacy (SP)*. May 2021, pp. 355–371. DOI: 10.1109/SP40001.2021.00063. (Visited on 05/21/2025).

[36] 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.

[37] Green Coding Berlin. *RAPL, SGX and energy filtering - Influences on power consumption*. Accessed May 2025. 2022. URL: https://www.green-coding.io/case-studies/rapl-and-sgx/.

[38] Intel Corporation. *Running Average Power Limit (RAPL) Energy Reporting*. Accessed May 2025. 2022. URL: https://www.intel.cn/content/www/cn/zh/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.html.

[39] Norman P. Jouppi et al. "In-Datacenter Performance Analysis of a Tensor Processing Unit". In: *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ISCA '17. New York, NY, USA: Association for Computing Machinery, June 2017, pp. 1–12. ISBN: 978-1-4503-4892-8. DOI: 10.1145/3079856.3080246. (Visited on 05/21/2025).

[40] Kubernetes Documentation. *GPUs in Kubernetes*. Accessed: 2025-05-09. 2025. URL: https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/.

[41] Inc. Datadog. *2024 Container Report*. Accessed: 2025-05-09. 2024. URL: https://www.datadoghq.com/container-report/.

[42] TensorFlow Documentation. *Running TensorFlow on Kubernetes*. Accessed: 2025-05-09. 2025. URL: https://www.tensorflow.org/tfx/serving/serving_kubernetes.

[43] NVIDIA Corporation. *NVIDIA Virtualization Resources*. Accessed: 2025-05-09. 2025. URL: https://www.nvidia.com/de-de/data-center/virtualization/resources/.

[44] AMD Corporation. *AMD Instinct Virtualization Documentation*. Accessed: 2025-05-09. 2025. URL: https://instinct.docs.amd.com/projects/virt-drv/en/latest/index.html.

[45] NVIDIA Corporation. *NVIDIA Multi-Instance GPU (MIG) User Guide*. Accessed: 2025-05-09. 2025. URL: https://docs.nvidia.com/datacenter/tesla/mig-user-guide/index.html.

[46] NVIDIA Corporation. *NVIDIA GPU Passthrough Documentation*. Accessed: 2025-05-09. 2025. URL: https://docs.nvidia.com/datacenter/tesla/gpu-passthrough/index.html.

[47] NVIDIA Corporation. *NVIDIA System Management Interface (nvidia-smi)*. Accessed: 2025-05-09. 2025. URL: https://developer.nvidia.com/system-management-interface.

[48] Zeyu Yang, Karel Adamek, and Wesley Armour. "Accurate and Convenient Energy Measurements for GPUs: A Detailed Study of NVIDIA GPU's Built-In Power Sensor". In: *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. Nov. 2024, pp. 1–17. DOI: 10.1109/SC41406.2024.00028. (Visited on 05/09/2025).

[49] Vijay Kandiah et al. "AccelWattch: A Power Modeling Framework for Modern GPUs". In: *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO '21. New York, NY, USA: Association for Computing Machinery, Oct. 2021, pp. 738–753. ISBN: 978-1-4503-8557-2. DOI: 10.1145/3466752.3480063. (Visited on 05/09/2025).

[50] Varsha Singhania, Shaizeen Aga, and Mohamed Assem Ibrahim. *FinGraV: Methodology for Fine-Grain GPU Power Visibility and Insights*. Mar. 2025. DOI: 10.48550/arXiv.2412.12426. arXiv: 2412.12426 [cs]. (Visited on 05/09/2025).

[51] Steven van der Vlugt et al. *PowerSensor3: A Fast and Accurate Open Source Power Measurement Tool*. Apr. 2025. DOI: 10.48550/arXiv.2504.17883. arXiv: 2504.17883 [cs]. (Visited on 05/09/2025).

[52] Anthony Hylick et al. "An Analysis of Hard Drive Energy Consumption". In: *2008 IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems*. Sept. 2008, pp. 1–10. DOI: 10.1109/MASCOT.2008.4770567. (Visited on 05/21/2025).

[53] Seokhei Cho et al. "Design Tradeoffs of SSDs: From Energy Consumption's Perspective". In: *ACM Trans. Storage* 11.2 (Mar. 2015), 8:1–8:24. ISSN: 1553-3077. DOI: 10.1145/2644818. (Visited on 05/18/2025).

[54] Linux NVMe Maintainers. *nvme-cli: NVMe management command line interface*. https://github.com/linux-nvme/nvme-cli. Accessed May 2025. 2025.

[55] smartmontools developers. *smartmontools: Control and monitor storage systems using S.M.A.R.T.* https://github.com/smartmontools/smartmontools/. Accessed May 2025. 2025.

[56] TechNotes. *Deciphering the PCI Power States*. Accessed June 2025. Feb. 2024. URL: https://technotes.blog/2024/02/04/deciphering-the-pci-power-states/.

[57] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. "Power Provisioning for a Warehouse-Sized Computer". In: *SIGARCH Comput. Archit. News* 35.2 (June 2007), pp. 13–23. ISSN: 0163-5964. DOI: 10.1145/1273440.1250665. (Visited on 05/21/2025).

[58] 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*. Apr. 2011, pp. 227–236. DOI: 10.1109/ISPASS.2011.5762739. (Visited on 05/21/2025).

[59] Anton Beloglazov et al. "Chapter 3 - A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems". In: *Advances in Computers*. Ed. by Marvin V. Zelkowitz. Vol. 82. Elsevier, Jan. 2011, pp. 47–111. DOI: 10.1016/B978-0-12-385512-1.00003-7. (Visited on 06/09/2025).

[60] E. N. (Mootaz) Elnozahy, Michael Kistler, and Ramakrishnan Rajamony. "Energy-Efficient Server Clusters". In: *Power-Aware Computer Systems*. Ed. by Babak Falsafi and T. N. Vijaykumar. Berlin, Heidelberg: Springer, 2003, pp. 179–197. ISBN: 978-3-540-36612-6. DOI: 10.1007/3-540-36612-1_12.

[61] 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*. E2SC '13. New York, NY, USA: Association for Computing Machinery, Nov. 2013, pp. 1–8. ISBN: 978-1-4503-2504-2. DOI: 10.1145/2536430.2536435. (Visited on 05/21/2025).

[62] Avita Katal, Susheela Dahiya, and Tanupriya Choudhury. "Energy Efficiency in Cloud Computing Data Center: A Survey on Hardware Technologies". In: *Cluster Computing* 25.1 (Feb. 2022), pp. 675–705. ISSN: 1573-7543. DOI: 10.1007/s10586-021-03431-z. (Visited on 06/04/2025).

[63] Robert Basmadjian et al. "A Methodology to Predict the Power Consumption of Servers in Data Centres". In: *Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking*. E-Energy '11. New York, NY, USA: Association for Computing Machinery, May 2011, pp. 1–10. ISBN: 978-1-4503-1313-1. DOI: 10.1145/2318716.2318718. (Visited on 06/09/2025).

[64] L. Luo, W.-J Wu, and F. Zhang. "Energy modeling based on cloud data center". In: *Ruan Jian Xue Bao/Journal of Software* 25 (July 2014), pp. 1371–1387. DOI: 10.13328/j.cnki.jos.004604.

[65] Robert Basmadjian and Hermann De Meer. "Evaluating and modeling power consumption of multi-core processors". In: *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*. 2012, pp. 1–10.

[66] Osman Sarood et al. "Maximizing throughput of overprovisioned hpc data centers under a strict power budget". In: *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE. 2014, pp. 807–818.

[67] Weiwei Lin et al. "A cloud server energy consumption measurement system for heterogeneous cloud environments". In: *Information Sciences* 468 (2018), pp. 47–62.

[68] Patricia Arroba et al. "Server power modeling for run-time energy optimization of cloud computing facilities". In: *Energy Procedia* 62 (2014), pp. 401–410.

[69] Aman Kansal et al. "Virtual machine power metering and provisioning". In: *Proceedings of the 1st ACM symposium on Cloud computing*. 2010, pp. 39–50.

[70] Miriam Allalouf et al. "Storage Modeling for Power Estimation". In: *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*. SYSTOR '09. New York, NY, USA: Association for Computing Machinery, May 2009, pp. 1–10. ISBN: 978-1-60558-623-6. DOI: 10.1145/1534530.1534535. (Visited on 05/18/2025).

[71] StoreDbits. *Hard Drive Power Consumption (HDD)*. Accessed May 2025. 2023. URL: https://storedbits.com/hard-drive-power-consumption/.

[72] StoreDbits. *SSD Power Consumption*. Accessed May 2025. 2023. URL: https://storedbits.com/ssd-power-consumption/.

[73] Yan Li and Darrell D.E. Long. "Which Storage Device Is the Greenest? Modeling the Energy Cost of I/O Workloads". In: *2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*. Sept. 2014, pp. 100–105. DOI: 10.1109/MASCOTS.2014.20. (Visited on 05/19/2025).

[74] Eric Borba, Eduardo Tavares, and Paulo Maciel. "A Modeling Approach for Estimating Performance and Energy Consumption of Storage Systems". In: *Journal of Computer and System Sciences* 128 (Sept. 2022), pp. 86–106. ISSN: 0022-0000. DOI: 10.1016/j.jcss.2022.04.001. (Visited on 05/18/2025).

[75] Ripduman Sohan et al. "Characterizing 10 Gbps Network Interface Energy Consumption". In: *IEEE Local Computer Network Conference*. Oct. 2010, pp. 268–271. DOI: 10.1109/LCN.2010.5735719. (Visited on 05/30/2025).

[76] Corey Gough, Ian Steiner, and Winston A. Sanders. *Energy Efficient Servers: Blueprints for Data Center Optimization*. Apress, 2015. ISBN: 9781430266372. DOI: 10.1007/978-1-4302-6638-9.

[77] Robert Basmadjian et al. "Cloud Computing and Its Interest in Saving Energy: The Use Case of a Private Cloud". In: *Journal of Cloud Computing: Advances, Systems and Applications* 1.1 (June 2012), p. 5. ISSN: 2192-113X. DOI: 10.1186/2192-113X-1-5. (Visited on 06/01/2025).

[78] Jordi Arjona Aroca et al. "A Measurement-Based Analysis of the Energy Consumption of Data Center Servers". In: *Proceedings of the 5th International Conference on Future Energy Systems*. E-Energy '14. New York, NY, USA: Association for Computing Machinery, June 2014, pp. 63–74. ISBN: 978-1-4503-2819-7. DOI: 10.1145/2602044.2602061. (Visited on 06/01/2025).

[79] Vincenzo De Maio et al. "Modelling Energy Consumption of Network Transfers and Virtual Machine Migration". In: *Future Generation Computer Systems* 56 (Mar. 2016), pp. 388–406. ISSN: 0167-739X. DOI: 10.1016/j.future.2015.07.007. (Visited on 06/04/2025).

[80] Waltenegus Dargie and Jianjun Wen. "A Probabilistic Model for Estimating the Power Consumption of Processors and Network Interface Cards". In: *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. July 2013, pp. 845–852. DOI: 10.1109/TrustCom.2013.103. (Visited on 06/04/2025).

[81] Saeedeh Baneshi et al. "Analyzing Per-Application Energy Consumption in a Multi-Application Computing Continuum". In: *2024 9th International Conference on Fog and Mobile Edge Computing (FMEC)*. Sept. 2024, pp. 30–37. DOI: 10.1109/FMEC62297.2024.10710253. (Visited on 05/30/2025).