Zurich University of Applied Sciences

Department School of Engineering

Institute of Computer Science

MASTER THESIS

Tycho:

An Accuracy-First Architecture for Server-Wide Energy Measurement and Process-Level Attribution in Kubernetes

---

Author:
Caspar Wackerle

Supervisors:
Prof. Dr. Thomas Bohnert
Christof Marti

Submitted on
January 31, 2026

Study program:
Computer Science, M.Sc.

Imprint

Project: Master Thesis
Title: Tycho: An Accuracy-First Architecture for Server-Wide Energy Measurement and Process-Level Attribution in Kubernetes
Author: Caspar Wackerle
Date: January 31, 2026
Keywords: process-level energy consumption, cloud, kubernetes, kepler

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

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

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

Abstract

Abstract

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

Contents

List of Figures

List of Tables

The global climate crisis is one of humanity's greatest challenges in this century.
With this work, I hope to contribute a small part in the direction we urgently need to go.

Chapter 1

Introduction

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
REVISE ENTIRE CHAPTER LATER XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

1.1 Motivation

Energy consumption in data centers continues to rise as demand for compute-intensive and latency-sensitive services increases. Modern cloud platforms host diverse workloads such as machine learning inference, analytics pipelines, and high-density microservices, all of which collectively contribute to a growing global electricity footprint. Container orchestration frameworks amplify these trends by enabling dense consolidation of workloads across shared servers. While this improves resource efficiency, it also introduces abstraction layers that obscure the relationship between workload behaviour and physical energy use.

As interest in sustainable cloud operations intensifies, there is increasing demand for precise, workload-level energy visibility. Fine-grained and reproducible energy measurements are essential for research domains such as performance engineering, scheduling, autoscaling, and the design of energy-aware systems. Existing tools provide valuable approximations but prioritise portability and low operational overhead, and therefore do not target the upper bounds of measurement fidelity. Research environments, by contrast, require methodologies that prioritise accuracy, control, and verifiability over deployability.

This thesis is motivated by the need for an accuracy-focused measurement approach that supports rigorous experimental work on containerised systems. Rather than proposing new optimisation mechanisms, this work concentrates on establishing a reliable methodological foundation for observing and analysing workload-induced energy consumption in controlled settings.

1.2 Problem Context

Modern multi-tenant servers host many short-lived and highly dynamic workloads that execute concurrently and compete for shared hardware resources. On such systems, the aggregate power draw represents the combined activity of numerous interacting subsystems, while the contributions of individual workloads remain deeply entangled. Containerisation further complicates this picture: processes belong to containers, containers belong to pods, and pods may change state rapidly under

orchestration. These abstractions improve system management but obscure how computational activity translates into power consumption.

At the same time, servers expose a heterogeneous collection of telemetry sources. Each source reflects different aspects of hardware behaviour, updates at its own cadence, and provides only a partial view of system activity. Because workload state changes and telemetry updates occur independently, they do not naturally align in time. The resulting temporal misalignment limits the reliability of workload-level energy attribution and leads to uncertainty in short-duration or phase-sensitive analyses.

Kubernetes introduces additional challenges. Workloads may start and terminate within milliseconds, metadata may appear with delays, and lifecycle events may interleave in complex ways. Existing tools often rely on coarse sampling windows or heuristic models that mask these inconsistencies. While sufficient for operational monitoring, such abstractions constrain the achievable accuracy in research settings. An accuracy-oriented approach requires explicit treatment of timing, metadata consistency, and correlation across heterogeneous measurement sources.

1.3 Position Within Previous Research

This thesis builds upon two earlier stages of work. The implementation-focused VT1 project developed an initial measurement pipeline and explored practical aspects of collecting hardware and system-level metrics in a Kubernetes environment. The subsequent VT2 project examined the state of the art in server-level energy measurement, validated the behaviour of commonly used telemetry sources, and identified methodological and technical limitations in existing tools such as Kepler. Both works are included in the appendix as supporting material.

The present thesis integrates these earlier insights but does not repeat them. Instead, it synthesises the essential findings from VT2 in a condensed form (Chapter 2), and introduces the conceptual foundations required to reason about accurate energy attribution (Chapter **??**). These chapters provide the background necessary to understand the accuracy-focused architecture developed later in this thesis.

1.4 Problem Statement

Accurately determining how much energy individual workloads consume in a Kubernetes cluster remains a challenging open problem. Clusters host many short-lived and overlapping workloads whose behaviour evolves rapidly, while server-level power telemetry is exposed through heterogeneous interfaces that update asynchronously and lack consistent timestamps. These timing mismatches, combined with the abstraction layers introduced by container orchestration, obscure the relationship between workload activity and physical energy use. Existing approaches provide high-level estimates but cannot deliver the temporal alignment, attribution fidelity, or reproducibility required for rigorous experimental analysis. This thesis therefore addresses the problem of designing a measurement methodology and prototype system capable of producing time-aligned, workload-level energy attribution with sufficient accuracy for research environments.

1.5 Goals of This Thesis

The overarching goal of this thesis is to develop an accuracy-focused approach for measuring energy consumption in Kubernetes-based environments. To achieve this, the work pursues four concrete objectives:

- **Methodological objective:** Define a measurement methodology that aligns heterogeneous telemetry sources with dynamic workload behaviour under a unified temporal model suitable for controlled research settings.
- **Architectural objective:** Design an accuracy-first system architecture that explicitly handles timing, metadata consistency, and correlation across diverse metrics without relying on heuristic abstractions.
- **Prototype objective:** Implement a research prototype that realises this architecture on commodity server hardware and integrates workload metadata, timing information, and server-wide telemetry into a coherent measurement pipeline.
- **Foundational objective for future work:** Establish the methodological and architectural basis for subsequent validation studies that will evaluate measurement fidelity and explore trade-offs between accuracy, overhead, and operational constraints.

1.6 Research Questions

1. What properties can an accuracy-first energy measurement system realistically guarantee when attributing energy consumption to workloads in Kubernetes environments with heterogeneous and delayed telemetry sources?
2. How does an explicit, semantics-driven attribution methodology influence the interpretability and explanatory power of workload-level energy measurements compared to implicit or estimation-oriented approaches?
3. In which experimental and research contexts does high-fidelity, accuracy-first energy measurement justify its complexity and overhead, and where do simpler approaches remain preferable?

1.7 Contributions

This thesis makes several conceptual and methodological contributions to the study of energy measurement in container-orchestrated environments. First, it introduces an accuracy-focused measurement approach that prioritizes temporal consistency, reproducibility, and the faithful representation of workload behaviour. The work defines a methodology for unifying heterogeneous sources of server telemetry under a shared timing model, enabling coherent interpretation of workload activity and system-level energy use.

A second contribution is the development of a prototype system that operationalizes this methodology and provides a concrete platform for exploring the limits of

high-fidelity energy measurement in Kubernetes-based environments. The prototype integrates workload metadata, timing information, and server-wide telemetry into a coherent measurement pipeline designed for research and controlled experimentation.

Third, the thesis establishes a foundation for reliable workload-level attribution by describing a structured process for correlating dynamic workload behaviour with system energy consumption. This provides a basis for analysing short-lived workload phases, transient resource usage patterns, and other phenomena that require fine-grained temporal alignment.

Finally, the work prepares the methodological groundwork for subsequent validation studies by outlining experimental procedures, calibration strategies, and evaluation principles suited to accuracy-oriented measurement. Together, these contributions advance the methodological state of the art and offer a practical reference point for future research on energy transparency in modern cloud infrastructures.

1.8 Scope and Boundaries

This thesis focuses on high-level principles and methods for energy measurement in multi-tenant server environments. The primary scope includes conceptual design, prototype development, and preparation of the methodological foundation for subsequent evaluation work. The emphasis is on accuracy, reproducibility, and consistency rather than operational deployability or production-grade integration.

Several areas remain outside the scope of this work. The thesis does not propose scheduling policies, predictive models, or system-level optimisation mechanisms. It does not modify Kubernetes or introduce changes to cloud operators' workflows. The prototype developed in this thesis is intended for controlled research environments and does not aim to provide a turnkey solution for general-purpose use. The work assumes access to a server environment where low-level telemetry and measurement interfaces are accessible under suitable conditions.

1.9 Origin of the Name "Tycho"

The prototype developed in this thesis is named *Tycho*, a reference to the astronomer Tycho Brahe. Brahe is known for producing exceptionally precise astronomical measurements, which later enabled Johannes Kepler to formulate the laws of planetary motion. The naming reflects a similar relationship: while the upstream *Kepler* project focuses on modelling and estimation, this thesis explores the upper bounds of measurement accuracy. Tycho thus signals both continuity with prior work and a shift toward an accuracy-first design philosophy.

1.10 Methodological Approach

1.11 Thesis Structure

Chapter 2

Background and Related Research

This chapter summarises the current state of research and industrial knowledge on server-level energy measurement. Its focus is limited to what the literature reports about available telemetry sources, measurement techniques, and existing attribution tools. The discussion is descriptive rather than conceptual: it does not introduce attribution principles, methodological reasoning, or design considerations, which are addressed in Chapter **??**. Extended background material is available in Appendix A and the present chapter integrates only those findings that are directly relevant for understanding the research landscape.

2.1 Energy Measurement in Modern Server Systems

The energy consumption of modern servers arises from a heterogeneous set of subsystems, including CPUs, GPUs, memory, storage devices, network interfaces, and platform management components. Prior research highlights that these subsystems expose highly unequal visibility into their power behaviour, since measurement capabilities, granularity, and accuracy differ significantly across hardware generations and vendors [2, 3]. Some domains provide direct telemetry, while others can only be approximated through software-derived activity metrics. As a result, no single interface offers complete or temporally consistent power information, and most studies rely on a single source or combine multiple sources to approximate system-level consumption. This fragmented measurement landscape forms the basis for much of the existing work on power modelling, validation, and multi-source energy estimation in server environments.

2.1.1 Energy Attribution in Multi-Tenant Environments

Several studies identify containerised and multi-tenant systems as challenging environments for energy attribution. Containers share the host kernel and rely on common processor, memory, storage, and network subsystems, which removes the isolation boundaries present in virtual machines and prevents direct measurement of per-container power. Research reports that workloads running concurrently on the same node create interference effects across hardware domains, leading to utilisation patterns that correlate only loosely with actual energy consumption [2]. Modern orchestration platforms further increase attribution difficulty through highly dynamic execution behaviour: containers are created, destroyed, and rescheduled at high frequency, often numbering in the thousands on large clusters. These rapid lifecycle changes produce volatile metadata and short-lived resource traces that are difficult

to align with node-level telemetry. Collectively, the literature treats container-level energy attribution as an estimation problem constrained by incomplete observability, heterogeneous measurement quality, and continuous runtime churn.

2.1.2 Telemetry Layers in Contemporary Architectures

Modern servers expose power and activity information through two largely independent telemetry layers. The first consists of in-band mechanisms that are visible to the operating system, including on-die energy counters, GPU management interfaces, and kernel-level resource statistics. These interfaces typically offer higher sampling rates and finer granularity, but their accuracy and coverage vary across hardware generations and vendors. Prior work notes that in-band telemetry often represents estimated rather than directly measured power and that several domains, such as network and storage devices, expose only partial or indirect information.

The second layer is out-of-band telemetry provided by baseboard management controllers through interfaces such as IPMI or Redfish. These systems aggregate sensor readings independently of the host and report stable, whole-system power values at coarse temporal resolution. Empirical studies show that out-of-band telemetry provides useful system-level accuracy, although update intervals and measurement precision differ substantially between vendors [4]. Compared with instrument-based measurements, which remain the benchmark for high-fidelity evaluation but are impractical at scale, both in-band and out-of-band methods represent trade-offs between granularity, availability, and measurement reliability.

Combined, these layers form a heterogeneous telemetry landscape in which sampling rates, accuracy, and domain coverage differ significantly, motivating the use of multi-source measurement approaches in research.

2.1.3 Challenges for Container-Level Measurement

Existing research identifies several factors that complicate accurate energy measurement for containerised workloads. Large-scale trace analyses show that cloud environments exhibit substantial churn, with many tasks being short-lived and resource demands changing rapidly over time [5]. Such dynamism limits the observability of fine-grained resource usage and makes it difficult to capture short execution intervals with sufficient temporal resolution.

Monitoring studies further report inconsistencies across the different layers that expose resource information for containers. In multi-cloud settings, observability often depends on heterogeneous monitoring stacks, leading to fragmented visibility and non-uniform coverage of system activity [6]. Even within a single host, performance counters obtained from container-level interfaces may diverge from system-level measurements. Empirical evaluations demonstrate that container-level CPU and I/O counters can underestimate actual activity by a non-negligible margin, and that co-located workloads introduce contention effects that distort these metrics [7].

These findings indicate that container-level measurement operates under conditions of rapid workload turnover, heterogeneous monitoring behaviour, and imperfect resource visibility. As a consequence, the literature treats container energy attribution as a problem constrained by incomplete and potentially biased measurement signals rather than as a directly measurable quantity.

2.2 Hardware and Software Telemetry Sources

This section outlines the primary telemetry sources used to observe power and resource behaviour in modern server systems. It summarises established research on external measurement devices, firmware-level interfaces, on-die energy counters, accelerator telemetry, and kernel-exposed resource metrics. The emphasis is on reporting the properties and empirical characteristics documented in prior work, without interpreting these signals conceptually or analysing their temporal behaviour, which are addressed in later sections. A comprehensive technical discussion is provided in Appendix A, Chapter **??**; the present section extracts only the findings relevant for understanding the measurement landscape.

2.2.1 Direct Hardware Measurement

Direct physical instrumentation remains the most accurate method for measuring server power consumption. External power meters or inline shunt-based devices can capture node-level energy usage with high fidelity, and research frequently uses such instrumentation as a ground truth for validating software-reported power values. Studies employing dedicated measurement setups, such as custom DIMM-level sensing boards, demonstrate that high-frequency sampling and component-level granularity are technically feasible but require bespoke hardware and non-trivial integration effort [8]. Lin et al. classify these approaches as offering very high data credibility but only coarse spatial granularity and limited scalability in operational environments [2].

Recent work on specialised sensors, such as the PowerSensor3 platform[9] for high-rate voltage and current monitoring of GPUs and other accelerators, illustrates ongoing interest in hardware-centric power measurement. However, these systems share the same fundamental drawback: deployment across production servers is complex, costly, and incompatible with large-scale or multi-tenant settings. As a consequence, direct instrumentation is predominantly used in controlled experiments or for validation of other telemetry sources, rather than as a primary measurement mechanism in real-world server infrastructures.

2.2.2 Legacy Telemetry Interfaces (ACPI, IPMI)

Early power-related telemetry on server platforms was primarily exposed through ACPI and IPMI. ACPI provides a standardised interface for configuring and controlling hardware power states, but it does not offer real-time energy or power readings. The interface exposes only abstract performance and idle states defined by the firmware [10], and these states do not include the instantaneous power information required for empirical energy measurement. Consequently, ACPI has seen little use in modern power estimation research.

IPMI, accessed through the baseboard management controller, represents an older class of out-of-band telemetry that predates Redfish. Although widely supported across server hardware, IPMI power values are known to be coarse, slowly refreshed, and often inaccurate when compared with external instrumentation. Empirical studies report multi-second averaging windows, substantial quantisation effects, and unreliable idle power readings [11, 12]. These limitations, together with the availability of more precise alternatives, have led IPMI to be largely superseded by Redfish on contemporary server platforms.

2.2.3 Redfish Power Telemetry

Redfish is the modern out-of-band management interface available on contemporary server platforms and is designed as the successor to IPMI. It exposes system-level telemetry through a RESTful API implemented on the baseboard management controller (BMC), providing access to whole-node power readings derived from on-board sensors. Prior work consistently shows that Redfish delivers higher precision than IPMI, with lower quantisation artefacts and more stable readings across power ranges [4]. In controlled experiments, Redfish achieved a mean absolute percentage error of roughly three percent when compared to a high-accuracy power analyser, outperforming IPMI in all evaluated power intervals.

A key limitation of Redfish is its temporal granularity. Empirical studies report that power values exhibit non-negligible staleness, with refresh delays of approximately 200 ms [4]. This latency restricts the ability of Redfish to capture short bursts of activity or rapid fluctuations in dynamic workloads. Accuracy and responsiveness also vary across vendors, reflecting differences in embedded sensors, BMC firmware, and management controller architectures.

The interface is widely deployed in real-world infrastructure. Modern enterprise servers from Dell, HPE, Lenovo, Cisco, and Supermicro routinely expose power telemetry via Redfish as part of their standard BMC firmware [13]. Out-of-band monitoring studies further highlight that Redfish avoids the overheads and failure modes associated with in-band agents [14]. In practice, Redfish implementations tend to provide stable low-frequency updates suitable for coarse-grained power reporting.

Preliminary measurements conducted for this thesis also observed irregular update intervals on the evaluated hardware, occasionally extending into the multi-second range. While this behaviour is specific to a single system and not generalisable, it reinforces the literature's position that Redfish telemetry exhibits meaningful vendor-dependent variability and remains unsuitable for fine-grained temporal correlation.

Overall, Redfish provides accessible, reliable whole-node power telemetry at coarse temporal resolutions, making it valuable for long-interval monitoring and for validating other measurement sources, but inappropriate for attributing energy consumption to short-lived or rapidly fluctuating containerised workloads.

2.2.4 RAPL Power Domains

Running Average Power Limit (RAPL) provides hardware-backed energy counters for several internal power domains of a processor package. Originally introduced by Intel and later adopted in a compatible form by AMD, RAPL exposes energy measurements via model-specific registers that can be accessed directly or through higher-level interfaces such as the Linux `powercap` framework or the `perf-events` subsystem [15, 16]. Raffin et al. provide a detailed comparison of these access mechanisms, noting that MSR, powercap, perf-events, and eBPF differ mainly in convenience, required privileges, and robustness; all can retrieve equivalent RAPL readings when implemented correctly [16]. They recommend accessing RAPL via the powercap interface, which is easiest to implement reliably and suffers from no overhead penalties when compared with more low-level methods.

Intel platforms typically expose several well-established RAPL domains, including the processor package, the core subsystem, and (on many server architectures) a DRAM domain [17]. These domains have been validated extensively against external measurement equipment. Studies report that the combination of package and DRAM energy tracks CPU-and-memory power with good accuracy from Haswell onwards, which has led to RAPL becoming the primary fine-grained energy source in server-oriented research [8, 18–20]. More recent work on hybrid architectures such as Alder Lake confirms that RAPL continues to correlate well with external measurements under load, while precision decreases somewhat in low-power regimes [21]. Across these studies, RAPL is generally regarded as sufficiently accurate for scientific analysis when its domain boundaries and update characteristics are considered [16].

AMD implements a RAPL-compatible interface with a similar programming model but a reduced set of domains. Zen 1 through Zen 4 processors expose package and core domains only, without a dedicated DRAM domain [16, 22]. Schöne et al. show that, as a consequence, memory-related energy may not be represented explicitly in AMD's RAPL output, leading to a smaller portion of total system energy being observable through the package domain alone [22]. This limitation primarily concerns domain completeness rather than measurement correctness: for compute-intensive workloads, package-domain values behave consistently, but workloads with significant memory activity exhibit a larger gap relative to whole-system measurements because DRAM energy is not separately reported. Raffin et al. further note that, on the evaluated Zen-based server, different kernel interfaces initially exposed inconsistent domain sets; this was later corrected upstream, illustrating that AMD support is evolving and still maturing within the Linux ecosystem [16].

Technical considerations also apply to both Intel and AMD platforms. RAPL counters have finite width and wrap after sufficiently large energy accumulation, requiring consumers to implement overflow correction [16, 23]. The counters do not include timestamps, and empirical work shows that actual update intervals may deviate from nominal values, complicating precise temporal correlation with other telemetry [18, 24]. On some Intel platforms, security hardening measures such as energy filtering reduce temporal granularity for certain domains to mitigate side-channel risks [21, 25, 26]. In virtualised environments, RAPL access may be trapped by the hypervisor, increasing latency and introducing small deviations from bare-metal behaviour [24].

In summary, RAPL provides a widely used and comparatively fine-grained source of processor-side energy telemetry. Intel platforms typically offer multiple validated domains, including DRAM, enabling a broader view of CPU-and-memory energy. AMD platforms expose fewer domains and therefore provide a more limited perspective on total system power, particularly for memory-intensive workloads. These differences in domain coverage, measurement scope, and software integration need to be taken into account when using RAPL as a basis for energy analysis.

2.2.5 GPU Telemetry

Unlike CPUs, where power and utilization telemetry is supported through standardised interfaces, GPU energy visibility relies primarily on vendor-specific mechanisms. For NVIDIA devices, two interfaces dominate this landscape: the *NVIDIA Management Library* (NVML), which has become the industry standard, and the *Data*

Center GPU Manager (DCGM), a less widely used management layer that also exposes telemetry.

2.2.5.1 NVML

NVML is NVIDIA's primary interface for device-level monitoring and underpins tools such as `nvidia-smi`. It provides access to power, energy (on selected data-center GPUs), GPU utilization, memory usage, clock frequencies, thermal state, and various health and throttle indicators. Among these, power and utilization are most relevant for energy analysis.

NVML power values represent board-level estimates derived from on-device sensing circuits and are shaped by internal averaging and architecture-dependent update behaviour. Recent empirical studies across modern devices show that NVML produces fresh samples only intermittently and applies smoothing that reduces the visibility of short-lived power changes, while steady-state power levels remain comparatively accurate [27]. On the Grace–Hopper GH200, these effects are pronounced: NVML reflects a coarse internal averaging interval and therefore underrepresents short kernels and transient peaks relative to higher-frequency system interfaces [28]. These findings indicate that NVML captures long-term power behaviour reliably but inherently limits fine-grained visibility. Despite these constraints, existing studies consistently find that NVML provides reasonably accurate steady-state power estimates on modern data-center GPUs and currently represents the most reliable and widely supported mechanism for obtaining GPU power telemetry in practical systems [28].

GPU utilization provides contextual information about device activity. It reports the proportion of time during which the GPU is executing any workload rather than the fraction of computational capacity in use, making it a coarse activity indicator rather than a detailed performance metric [29].

2.2.5.2 DCGM

DCGM is NVIDIA's management and observability framework designed for data-center deployments. It aggregates telemetry, performs health monitoring, exposes thermal and throttle state, and provides detailed visibility in environments that employ Multi-Instance GPU (MIG) partitioning. However, DCGM's power and utilization metrics are derived from the same underlying measurement sources as NVML. In practice, DCGM is far less commonly used for energy analysis because it does not provide higher-fidelity power telemetry; instead, it applies additional aggregation and is typically deployed with coarse sampling intervals, especially when used through exporters in cluster monitoring systems. DCGM therefore represents an alternative access path to the same measurements rather than a distinct source of energy-related information.

DCGM is considerably less common in both research and operational practice, with most GPU monitoring systems relying primarily on NVML while DCGM appears only occasionally in cluster-level deployments [29].

2.2.5.3 Summary

NVML and DCGM jointly define the available mechanisms for GPU telemetry in cloud environments. NVML is the dominant and broadly supported interface for power and utilization measurement, while DCGM extends it with operational metadata and management integration. Current studies consistently show that both interfaces expose averaged, device-level power estimates that capture long-term behaviour but are inherently limited in their ability to represent short-duration activity or fine-grained workload structure. These characteristics form the scientific foundation for later discussions of temporal behaviour and measurement methodology.

2.2.6 Software-Exposed Resource Metrics

In addition to hardware telemetry, Linux and Kubernetes expose a wide range of software-level resource metrics that describe system and workload activity. These metrics do not measure power directly but provide essential behavioural context that complements RAPL, Redfish, and GPU telemetry.

2.2.6.1 CPU and Memory Activity Metrics

Linux provides several complementary mechanisms for tracking CPU and memory usage. Global counters such as `/proc/stat` record cumulative CPU time since boot, while per-task statistics in `/proc/<pid>` expose user-mode and kernel-mode execution time with high granularity [30]. Control groups (cgroups) provide container-level CPU and memory accounting and form the primary basis for utilisation metrics inside Kubernetes [31, 32]. Higher-level tools such as cAdvisor and metrics-server aggregate this information via Kubelet, but at significantly lower update rates.

Event-driven approaches provide substantially finer resolution. eBPF allows dynamic attachment to kernel events such as context switches, scheduling decisions, and I/O operations, enabling near-real-time capture of per-task CPU activity with low overhead [33, 34]. Hardware performance counters accessed through `perf` offer insight into instruction counts, cycles, cache behaviour, and stalls [35]. These sources provide detailed behavioural information but still represent utilisation rather than energy.

2.2.6.2 Storage Activity Metrics

Storage subsystems do not expose real-time power telemetry, yet Linux provides a rich set of activity indicators. Per-process statistics in `/proc/<pid>/io` track bytes read and written, while cgroup I/O controllers report aggregated container-level metrics. Subsystem-specific tools such as `smartctl` and `nvme-cli` reveal additional device characteristics, queue behaviour, and state transitions [36, 37].

In the absence of hardware power sensors, multiple works propose workload-dependent energy models for storage devices [38–40]. These models can yield accurate estimates when calibrated for a specific device but do not generalise across heterogeneous hardware due to differences in flash controllers, firmware, and internal data paths.

2.2.6.3 Network and PCIe Device Metrics

Network interfaces provide byte and packet counters via `/proc/net/dev`, but expose no dedicated power telemetry. Research models for NIC energy consumption exist [41–43], yet all rely on device-specific idle and active power characteristics that are not available at runtime. Similarly, PCIe devices support abstract power states as defined by the PCIe specification [44], but these states do not reflect instantaneous power usage and thus offer only coarse activity signals.

2.2.6.4 Secondary System Components

Components such as fans, motherboard logic, and power delivery subsystems rarely expose fine-grained telemetry. Although some BMC implementations report coarse sensor values, these readings are inconsistent across platforms and generally unsuitable for high-resolution analysis. Consequently, research commonly treats these subsystems as part of the residual power that scales with the activity of primary components [42].

2.2.6.5 Model-Based Estimation Approaches

Because software-visible metrics capture detailed workload behaviour, many works propose inferring energy consumption from utilisation using regression or stochastic models [45–48]. While these models can be effective when fitted to a specific hardware platform, their accuracy depends heavily on device-specific parameters, making them unsuitable as a general mechanism for heterogeneous server environments. Machine-learning-based estimators share the same limitation: high accuracy when trained for a fixed configuration, poor portability without extensive retraining.

2.2.6.6 Summary

Software-exposed metrics provide high-resolution visibility into CPU, memory, I/O, and network activity. They are indispensable for correlating workload behaviour with hardware power signals, especially for components that lack native telemetry. Model-based estimation remains possible but inherently platform-specific, and therefore unsuitable as a universal foundation for fine-grained attribution in heterogeneous environments.

2.3 Temporal Behaviour of Telemetry Sources

A comprehensive treatment of temporal characteristics can be found in Appendix A, Chapter **??**, but the present section focuses on the empirical, source-specific behaviours that constrain fine-grained power and energy estimation on real systems. Modern server platforms expose a heterogeneous set of telemetry interfaces, and their timing properties vary substantially: some update at fixed intervals, others employ internal averaging or smoothing, several expose counters without timestamps, and many lack guarantees on refresh regularity. These behaviours shape the effective temporal resolution with which workload-induced power changes can be observed.

The purpose of this section is not to develop a conceptual theory of sampling or to explain why timing matters for attribution (both are deferred to Chapter **??**), nor to introduce Tycho's timing engine (Chapter **??**). Rather, it establishes the empirical

constraints imposed by the telemetry sources themselves. These include sensor refresh intervals, stability of consecutive updates, delays between physical behaviour and reported values, the presence or absence of timestamps, and the distinction between instantaneous versus internally averaged measurements.

The subsections that follow describe these temporal properties for each telemetry source individually and summarise the practical limits they impose on high-resolution energy analysis.

2.3.1 RAPL Update Intervals and Sampling Stability

RAPL exposes energy *counters* rather than instantaneous power values. These counters accumulate energy since boot and can be read at arbitrarily high frequency, but their usefulness is determined entirely by how often the internal measurement logic refreshes them, a timing behaviour that is undocumented and domain-dependent.

Domain-specific internal update rates. Intel specifies the RAPL time unit as 0.976 ms for the slowest-updating domains, while others, notably the PP0 (core) domain, may refresh significantly faster [21]. In practice, however, these theoretical limits do not translate into usable temporal resolution because RAPL provides no timestamps: the moment of counter refresh is unknown to the reader. At sub-millisecond sampling rates, the lack of timestamps combined with irregular refresh behaviour introduces substantial relative error, since differences between consecutive reads may reflect counter staleness rather than actual power dynamics [23].

Noise introduced by security-driven filtering. To mitigate power-side channels such as Platypus, Intel optionally introduces randomised noise through the `ENERGY_FILTERING_ENABLE` mechanism [26]. This filtering increases the effective minimum granularity from roughly 1 ms to approximately 8 ms for the PP0 domain [21]. While average energy over longer intervals remains accurate, instantaneous increments become less reliable at very short timescales.

Practical sampling limits. Despite the nominal sub-millisecond timing, empirical work consistently shows that high-frequency polling offers no practical benefit. Multiple studies report that sampling faster than the internal update period only produces repeated counter values and amplifies read noise [23]. Jay et al. demonstrate that at polling rates slower than 50 Hz, the relative error falls below 0.5 % [24]. Consequently, typical measurement practice (and the limits adopted in this thesis) treats RAPL as reliable only at tens-of-milliseconds resolution, not at the theoretical millisecond scale suggested by its nominal time unit.

Summary. Although RAPL counters can be read extremely quickly, the effective temporal resolution is constrained by undocumented refresh intervals, absence of timestamps, optional security filtering, and substantial measurement noise at high polling rates. For practical purposes, sampling at approximately 20–50 ms intervals yields the most stable and accurate results, while sub-millisecond polling is inadvisable due to high relative error and counter staleness.

2.3.2 GPU Update Intervals and Sampling Freshness

GPU power telemetry is exposed primarily through NVML, with DCGM providing an alternative access path that builds on the same underlying measurement source. Unlike CPU-side interfaces integrated into the processor package, GPU power monitoring is performed entirely by the device itself: internal sensing circuits and firmware determine how often new values are produced, how they are averaged, and when they are published to software. As a result, refresh behaviour varies substantially across architectures, and the temporal properties of the reported values depend on device-internal update cycles rather than the rate at which the host system issues queries, which limits the achievable resolution of any external sampling strategy.

Internal update cycles and sampling freshness. Empirical studies consistently show that NVML publishes new power values only intermittently, even when queried at high frequency. Yang et al. report sampling availability as low as roughly twenty–twenty-five percent across more than seventy modern data-center GPUs, meaning that the majority of polls return previously published values rather than fresh measurements [27].

Typical internal update periods fall on the order of tens to several hundreds of milliseconds, with architectural variation between GPU generations. Hernandez et al. report that newer architectures apply more aggressive smoothing and exhibit longer gaps between updates, reflecting slower publication cadence at the firmware level [28]. Overall, empirical evaluations show that NVML's internal update interval may lie on the order of hundreds of milliseconds and that repeated queries do not guarantee the retrieval of a new sample at every call [27]. NVML power readings do not represent instantaneous electrical measurements; they reflect firmware-level integration and smoothing over a device-internal averaging window, the duration of which varies by GPU generation and is not publicly documented..

Reaction delay to workload-induced power changes. A related characteristic is NVML's reaction delay: when GPU power changes due to workload activity, the corresponding update becomes visible only after a lag. Multiple studies document delays in the range of approximately one to three hundred milliseconds before a new NVML value reflects the underlying power transition [27]. This delay is distinct from averaging effects and arises from deferred publication of internally accumulated measurements. On some recent architectures, the delay can be longer due to device-level smoothing layers that defer updates until sufficient internal samples have been collected [28].

Update regularity and jitter. NVML update cycles are not perfectly periodic. Even when a nominal internal cadence is observable, individual publish times exhibit modest jitter, and occasional missed or skipped updates can result in sequences of identical values. These effects are pronounced on certain consumer-class devices and in configurations that partition the GPU, such as MIG, although they are also present to a lesser degree on data-center accelerators [27]. Such irregularity introduces uncertainty regarding the true measurement time of any retrieved value, especially in the sub-second range.

DCGM sampling behaviour. DCGM relies on the same underlying measurement path as NVML and therefore inherits NVML's internal update characteristics. In practice, DCGM is commonly accessed through its exporter, which introduces an additional periodic sampling stage (typically around one second) resulting in markedly coarser temporal behaviour than NVML's native cadence. As a result, DCGM-based power telemetry rarely offers sub-second resolution in operational environments [29].

GPU utilization update cycles. NVML's GPU utilization metric follows its own internal update cadence, separate from power. It is typically refreshed more frequently (on the order of tens of milliseconds) although the exact timing remains undocumented. While this metric does not track computational efficiency, its shorter update interval provides a comparatively more responsive indicator of device activity [29].

2.3.3 Redfish Sensor Refresh Intervals and Irregularity

Redfish exposes power telemetry through the baseboard management controller (BMC) and therefore inherits the temporal behaviour of its embedded sensing hardware and firmware. In contrast to on-chip interfaces such as RAPL or NVML, Redfish is designed for management-plane observability rather than high-frequency monitoring. Prior studies consistently report that Redfish refreshes whole-node power values at coarse intervals, typically ranging from several hundred milliseconds to multiple seconds, with the exact cadence depending on vendor, BMC firmware, and underlying sensor design [4, 14]. The Redfish standard does not define a minimum update frequency, and available documentation provides little insight into internal sampling or averaging strategies.

Measurement semantics of Redfish power values. Redfish does not expose instantaneous electrical measurements. Instead, the reported values originate from on-board monitoring chips connected to shunt-based sensors and are subsequently processed inside the BMC. Vendor documentation indicates that these sensors inherently integrate power over tens to hundreds of milliseconds, and that additional firmware-level smoothing may be applied before values are published through the Redfish API [14]. Empirical evaluations support this interpretation: Wang et al. show that Redfish exhibits reaction delays of roughly two hundred milliseconds and displays particularly stable behaviour under steady loads, consistent with block-averaged rather than instantaneous sampling [4]. Because neither the sensor integration window nor any BMC filtering policies are defined in the standard, the temporal semantics of published values remain implementation-dependent.

Redfish power readings include a timestamp field, but this value reflects the BMC's observation time rather than the sampling instant of the physical power sensor. In many implementations, timestamps are rounded to seconds, which limits their utility for reconstructing sub-second dynamics and prevents reliable inference of the underlying sampling moment.

Beyond published work, empirical observations from the system used in this thesis reveal that Redfish update intervals may exhibit substantial variability. While nominal refresh periods appear regular over longer windows, individual samples occasionally show multi-second gaps, repeated values, or irregular spacing. Such

behaviour is consistent with a telemetry source operating on management-plane scheduling and BMC workload constraints rather than real-time guarantees. These observations do not generalise across vendors but illustrate the degree of temporal uncertainty that can occur in practice.

Overall, Redfish provides a widely supported mechanism for obtaining whole-system power readings and is well suited for coarse-grained monitoring or validation of other telemetry sources. Its coarse refresh intervals, lack of sensor-level timestamps, and implementation-dependent irregularities, however, make it unsuitable for analysing short-duration phenomena or for use as a primary source in high-resolution energy attribution.

2.3.4 Timing of Software-Exposed Metrics

Software-exposed resource metrics differ fundamentally from hardware-integrated telemetry sources: rather than publishing sampled power or energy values at device-defined intervals, the Linux kernel exposes cumulative counters whose temporal behaviour is almost entirely determined by when they are read. These interfaces therefore provide quasi-continuous visibility into system activity, but without intrinsic update cycles or timestamps that would define the sampling moment of the underlying measurement.

Cumulative counters in `/proc` and cgroups. Kernel interfaces such as `/proc/stat`, per-task entries under `/proc/<pid>`, and the CPU accounting files in cgroups expose resource usage as monotonically increasing counters. These values are updated by the kernel during scheduler events, timer interrupts, and context-switch accounting, rather than at fixed intervals. As a consequence, their effective temporal resolution is determined entirely by the user's polling cadence: reading them more frequently produces more detailed deltas, but the kernel does not provide any guarantee about when a counter was last updated. None of these counters include timestamps, and their update timing may vary across systems due to tickless operation, kernel configuration, and workload characteristics.

Disk and network I/O statistics. I/O-related counters follow the same principle. Entries such as `/proc/<pid>/io`, cgroup I/O files, and interface statistics in `/proc/net/dev` are incremented as part of the corresponding driver paths when I/O operations occur. They do not refresh periodically and therefore exhibit update patterns that mirror workload activity rather than a regular cadence. Temporal interpretation again depends entirely on the polling rate of the monitoring system.

eBPF-based event timing. In contrast to cumulative counters, eBPF enables event-driven monitoring with explicit timestamps. Kernel probes attached to scheduler events, I/O paths, or tracepoints can record event times with high precision using the kernel's monotonic clock. As a result, eBPF metrics provide effectively instantaneous temporal resolution and are limited only by the overhead of probe execution and user-space consumption of BPF maps. No internal refresh cycle exists; events are timestamped at the moment they occur.

Performance counters and `perf`-based monitoring. Hardware performance monitoring counters (PMCs), accessed via `perf_event_open`, advance continuously

within the processor. They do not follow a publish interval, and their timing semantics are defined solely by the instant at which user space reads the counter. This provides fine-grained and low-latency access to execution metrics such as cycles and retired instructions, with overhead rising only when polling is performed at very high frequencies.

Overall, software-exposed metrics behave as cumulative or event-driven signals rather than sampled telemetry sources. Their temporal characteristics are dominated by polling strategy and kernel-level event timing, with eBPF representing the only interface that attaches precise timestamps directly to system events.

2.4 Existing Tools and Related Work

Energy observability in containerized environments has attracted increasing attention in recent years, leading to the development of several tools that combine hardware, software, and statistical telemetry to estimate per-workload energy consumption. Despite this diversity, only a small number of tools attempt to attribute energy at container or pod granularity with sufficient detail to inform system-level research. Among these, *Kepler* has emerged as the most widely adopted open-source solution within the cloud-native ecosystem, while *Kubewatt* represents the first focused research effort to critically evaluate and refine Kepler's attribution methodology. Other frameworks, such as Scaphandre, SmartWatts, or PowerAPI, offer relevant ideas but differ in scope, telemetry assumptions, or operational goals. For this reason, the remainder of this section concentrates primarily on Kepler and Kubewatt, using these two tools to illustrate the architectural and methodological challenges that motivate the research gaps identified at the end of this chapter.

2.4.1 Kepler

2.4.1.1 Architecture and Metric Sources

Kepler[49] is a node-local energy observability agent designed for Kubernetes environments. Its architecture follows a modular dataflow pattern: a set of collectors periodically ingests telemetry from hardware and kernel interfaces, an internal aggregator aligns and normalizes these inputs, and a Prometheus exporter exposes the resulting metrics at container, pod, and node granularity. This structure allows Kepler to integrate heterogeneous telemetry sources while presenting a unified metric interface to external monitoring systems.

Kepler's collectors obtain process, container, and node telemetry from standard Linux and Kubernetes subsystems. Resource usage statistics are taken from `/proc`, cgroup hierarchies, and Kubernetes metadata, while hardware-level energy data is read from RAPL domains via the `powercap` interface. Optional collectors provide GPU metrics through NVIDIA's NVML library and platform-level power measurements via Redfish or other BMC interfaces. All inputs are treated as cumulative counters or periodically refreshed state, and their effective resolution is therefore determined by Kepler's sampling configuration. All metrics are updated at same interval and at the same time (default: 60 seconds for redfish, 3 seconds for all other sources). A central responsibility of the aggregator is to map raw per-process telemetry to containers and pods, using cgroup paths and Kubernetes API metadata. The derived metrics

are finally exposed via a Prometheus endpoint, enabling integration into common cloud-native observability stacks.

In contrast to generic system monitoring agents, Kepler's architecture is tailored specifically to Kubernetes. Its emphasis on container metadata, cgroup-based accounting, and workload-oriented metric aggregation distinguishes it from tools that operate primarily at the host or VM level. At the same time, its reliance on standard Linux interfaces keeps deployment overhead low, requiring only node-local access to `/proc`, cgroups, and the `powercap` subsystem.

Overall, Kepler's architectural design reflects a trade-off between flexibility and granularity: while it can ingest diverse telemetry sources and attribute energy at container level, its accuracy is constrained by the timing and resolution of the underlying metrics, as well as the unified sampling cadence chosen for the collectors.

Kepler updates all metrics within a single synchronous loop that triggers every sampling interval. This design simplifies integration but enforces a uniform cadence across heterogeneous telemetry sources, which contributes to the timing and alignment issues discussed in § 2.4.1.3. The structure is shown in Figures 2.1.

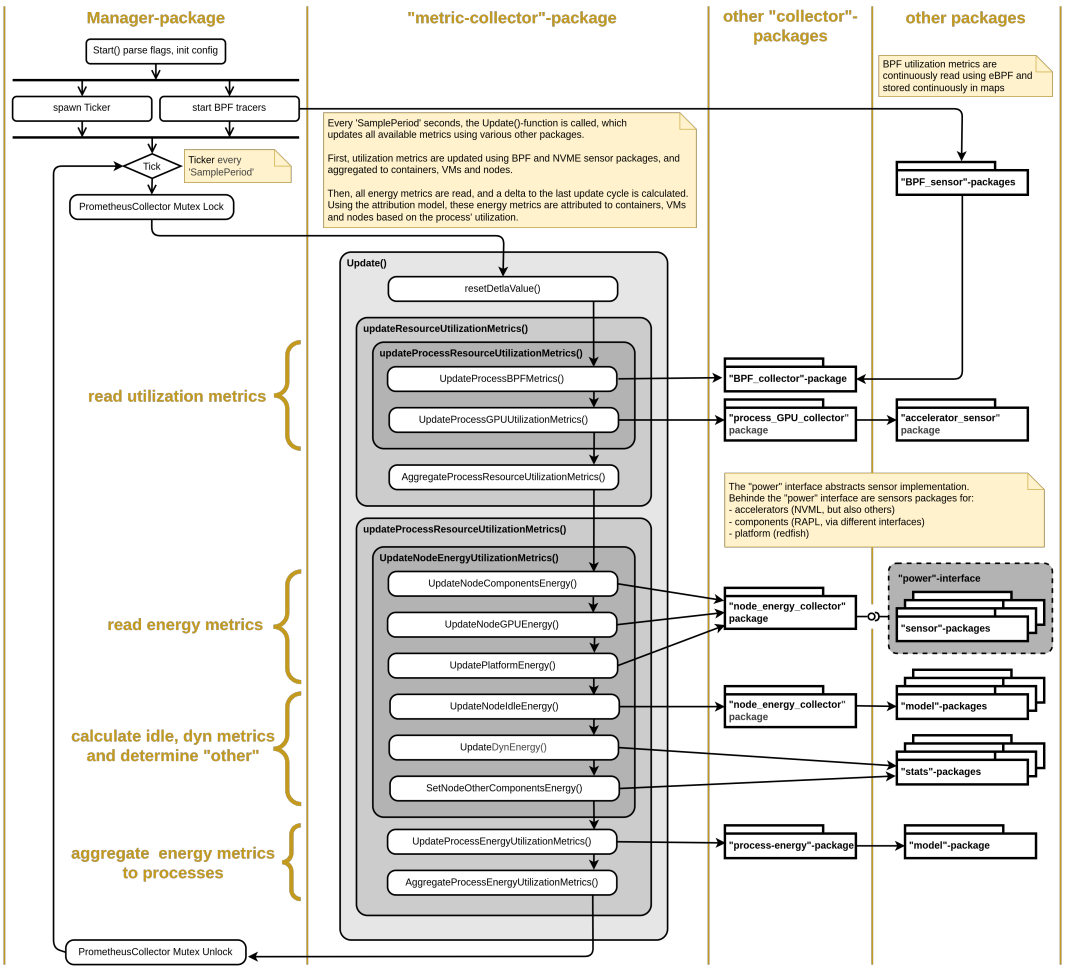

FIGURE 2.1: Kepler's synchronous update loop, where all collectors run at a unified sampling interval.

2.4.1.2 Attribution Model

Kepler's attribution logic follows a two-stage structure. First, node-level energy is decomposed into *idle* and *dynamic* components for each available power domain (package, core, uncore, DRAM, and optionally GPU or platform-level readings). Second, the dynamic portion is distributed across processes and containers according to their observed resource usage, while idle power is assigned using a domain-specific default policy.

Dynamic power is attributed proportionally using ratio-based models. For each domain, Kepler computes the energy delta over the sampling interval and distributes it according to a usage metric selected for that domain. Instructions, cache misses, and CPU time are used as primary signals, with fallbacks when a metric is unavailable. GPU dynamic power is attributed based on GPU compute utilization. Platform-level power, when available, is treated as a residual domain: after subtracting CPU and DRAM power, the remaining portion is shared across workloads using the designated default metric or, if none is configured (which is the case), an equal split.

Idle energy is handled separately. Kepler maintains a rolling estimate of minimum node-level power for each domain and treats these values as idle baselines. In the default configuration, idle energy is divided evenly across all active workloads during the attribution interval. While this behaviour differs from protocol recommendations that scale idle power by container size, Kepler applies a uniform policy across domains to ensure attribution completeness.

Attribution operates at process granularity, with container and pod values obtained by aggregating the processes mapped to each cgroup. This approach allows Kepler to attribute energy to short-lived or multi-process containers while retaining compatibility with Kubernetes metadata.

Kepler performs attribution at a fixed internal update interval (default: 3 s). All usage metrics and energy deltas within that interval are aggregated before attribution is computed. Because Prometheus scrapes occur independently of Kepler's internal loop, the exported time series may reflect misalignment when the scrape period is not a multiple of the update interval. This can lead to visible step patterns or oscillations, particularly for dynamic workloads. Despite these limitations, the model provides a coherent and workload-oriented view of node-level energy consumption suitable for cloud-native observability scenarios.

2.4.1.3 Observed Behavior and Limitations

Several studies and code-level inspections reveal that Kepler's attribution behaviour exhibits systematic limitations that affect accuracy and interpretability. The most comprehensive empirical evaluation to date, conducted by Pijnacker et al., demonstrates that attribution inaccuracies arise even when node-level power estimation is reliable[50]. Their experiments highlight that idle power is often distributed to containers that are no longer active, including Completed pods, and that dynamic power can be reassigned inconsistently when containers are added or removed. These effects stem from the coupling of process-level accounting with container lifecycle events, which may lag behind cgroup or Kubernetes metadata updates.

Timing mismatches further contribute to attribution artifacts. High-frequency CPU

and cgroup statistics are combined with slower telemetry sources such as Redfish, whose update intervals may span tens of seconds. When workload intensity changes during such periods, Kepler may assign disproportionately large or small dynamic energy shares to individual containers. Similar behavior occurs at the Prometheus interface when scrape intervals do not align with Kepler's internal update loop, producing visible oscillations in the exported time series.

Source-code inspection reinforces these observations. Numerous unimplemented or placeholder sections (e.g. TODO markers) affect key components of the ratio-based attribution model and default configuration paths. In particular, some domains lack explicit usage metrics, leading to fallback behaviour and equal-cost splitting regardless of container activity. GPU attribution relies on a single utilization metric and is therefore sensitive to the temporal behaviour of NVML's sampling. Together, these issues introduce variability across domains and reduce the transparency of the resulting per-container energy values.

Lifecycle handling also presents challenges. Because container metadata is aggregated from process-level information, short-lived or Completed pods may retain residual energy assignments. Conversely, system processes that cannot be mapped cleanly to Kubernetes abstractions may absorb unassigned power, obscuring the relationship between application behaviour and observed consumption.

Overall, these limitations underscore that Kepler provides a practical but imperfect approximation of container-level energy consumption. The observed behaviour motivates the research gaps identified at the end of this section, particularly the need for finer temporal resolution, explicit handling of idle and residual power, configurable attribution models, and more robust reconciliation across heterogeneous telemetry sources.

2.4.1.4 Kepler v0.10.0

In July 2025, Kepler underwent a substantial architectural redesign with the release of version 0.10.0[51]. The new implementation replaces many of the privileged operations used in earlier versions, removing the need for `CAP_BPF` or `CAP_SYSADMIN` and reducing reliance on kernel instrumentation. Instead, Kepler now obtains all workload statistics from read-only `/proc` and cgroup interfaces. This reduction in privilege requirements significantly improves deployability and security, particularly for managed Kubernetes environments where eBPF- or perf-based approaches are infeasible.

The redesign also introduces a markedly simplified attribution model. Whereas earlier versions combined multiple hardware and software counters (e.g. instructions, cache misses, GPU utilization) to estimate dynamic energy, Kepler v0.10.0 relies exclusively on CPU time as the usage metric. Node-level dynamic energy is computed by correlating RAPL deltas with aggregate CPU activity, and each workload receives a proportional share of this value based solely on its CPU time fraction. Idle energy is not distributed to containers and instead remains part of a node-level baseline. Containers, processes, and pods are treated as independent consumers drawing from the same active-energy pool, with no dependence on process-derived aggregation.

These changes increase robustness and predictability: the simplified model is easier to reason about, less sensitive to heterogeneous workloads or timing mismatches,

and compatible with environments where kernel-level measurement facilities are unavailable. However, the loss of metric flexibility substantially reduces modeling fidelity. Fine-grained distinctions between compute-bound and memory-bound tasks are no longer observable, and the attribution model presumes a strictly linear relationship between CPU time and power consumption. As a result, Kepler v0.10.0 no longer targets high-accuracy energy attribution but instead emphasises operational stability and minimal overhead.

For the purposes of this thesis, Kepler v0.10.0 is relevant primarily as an indication of the project's strategic shift toward simplicity and broad deployability. Its CPU-time-only model is not suitable as a basis for Tycho, whose objectives require higher temporal resolution, more diverse metric inputs, and explicit handling of domain-level energy contributions. Accordingly, the remainder of this chapter focuses on the behaviour of Kepler v0.9.x, which remains the most representative version for research-oriented attribution discussions.

2.4.2 KubeWatt

KubeWatt is a proof-of-concept exporter developed by Pijnacker as a direct response to the attribution issues uncovered in Kepler.[50, 52] Rather than extending Kepler's complex pipeline, KubeWatt implements a deliberately narrow but transparent model that focuses on correcting three specific problems: misattribution of idle power, leakage of energy into generic "system processes", and unstable behaviour under pod churn. It targets Kubernetes clusters running on dedicated servers and assumes that a single external power source per node is available (in the prototype, Redfish/iDRAC).

A central design decision is the strict separation between *static* and *dynamic* power. Static power is defined as the baseline cost of running the node and its control plane in an otherwise idle state. KubeWatt measures or estimates this baseline once and treats it as a constant; it is *not* attributed to containers. Dynamic power is then computed as the difference between total node power and this static baseline and is the only quantity distributed across workloads. Control-plane pods are explicitly excluded from the dynamic attribution set, so their idle consumption remains part of the static term and does not pollute application-level metrics.

To obtain the static baseline, KubeWatt provides two initialization modes. In *base initialization*, the cluster is reduced to an almost idle state (only control-plane components), and node power is sampled for a few minutes. The static power value is computed as a simple average, yielding a highly stable estimate under the test conditions. When workloads cannot be stopped, *bootstrap initialization* fits a regression model to time series of node power and CPU utilization collected during normal operation. The regression is evaluated at the average control-plane CPU usage to infer the static baseline. This mode is more sensitive to workload characteristics and SMT effects but provides a practical fallback when base initialization is not feasible.

During normal operation, KubeWatt runs in an *estimation mode* that attributes dynamic node power to containers proportionally to their CPU usage. CPU usage is obtained from the Kubernetes metrics API (`metrics.k8s.io`) at node and container level. The denominator explicitly sums only container CPU usage; system

processes, cgroup slices, and other non-container activity are excluded by construction. This corrects a key source of error in Kepler, where slice-level metrics and kernel processes could receive non-trivial fractions of node power. Under stable workloads and at the relatively coarse sampling interval used in the prototype, KubeWatt achieves container-level power curves that align well with both iDRAC readings and observed CPU utilisation, and it behaves robustly when large numbers of idle pods are created and deleted.

The scope of KubeWatt is intentionally narrow. It is CPU-only, uses a single external power source per node, assumes that Kubernetes is the only significant workload on the machine, and does not attempt to model GPU, memory, storage, or network energy. It also inherits the temporal limitations of the Kubernetes metrics pipeline and treats Redfish power readings as instantaneous, without explicit latency compensation. Nevertheless, KubeWatt demonstrates that a simple, well-documented ratio model with explicit static–dynamic separation and strict cgroup filtering can eliminate several of Kepler's most problematic attribution artefacts. These design principles are directly relevant for the attribution redesign pursued in this thesis and inform the requirements placed on Tycho's more general, multi-source architecture.

2.4.3 Other Tools (Brief Overview)

Beyond Kepler, several tools illustrate the methodological diversity in container- and process-level energy attribution, although they are not central to the Kubernetes-specific challenges addressed in this thesis. Scaphandre[53] provides a lightweight proportional attribution model based exclusively on CPU time and RAPL deltas. Its design emphasises simplicity and portability, offering basic container mapping through cgroups but limited control over sampling behaviour or attribution semantics. SmartWatts[54], by contrast, represents a more sophisticated approach: it builds performance-counter-based models that self-calibrate against RAPL measurements and adapt dynamically to the host system. While effective in controlled environments, SmartWatts requires access to perf events, provides only CPU and DRAM models, and is not deeply integrated with Kubernetes abstractions.

A broader ecosystem of lightweight tools (e.g. CodeCarbon[55] and related library-level estimators) demonstrates further variation in scope and assumptions, but these generally target high-level application profiling rather than system-wide workload attribution. Collectively, these tools highlight a spectrum of design choices (from simplicity and portability to model-driven estimation) but none address the combination of high-resolution telemetry, multi-tenant attribution, and Kubernetes metadata integration that motivates the development of Tycho.

2.4.4 Cross-Tool Limitations Informing Research Gaps

Across the surveyed tools, several structural limitations recur despite substantial differences in design philosophy and implementation. First, temporal granularity remains insufficient: although hardware interfaces such as RAPL support millisecond-level updates, most tools aggregate measurements over multi-second intervals. This obscures short-lived workload behaviour and reduces attribution fidelity, particularly in heterogeneous or bursty environments. Second, all tools depend on telemetry sources whose internal semantics are only partially documented. Ambiguities regarding RAPL domain coverage, NVML power reporting, or BMC-derived node

power constrain both the interpretability and the auditability of reported metrics, reinforcing the black-box character of current measurement pipelines.

Idle-power handling presents a further source of inconsistency. Tools differ widely in how idle power is defined, whether it is attributed, and to whom. These choices are often implicit, undocumented, or constrained by implementation artefacts, leading to attribution patterns that are difficult to interpret or reproduce. Multi-domain coverage is similarly limited: existing tools focus primarily on CPU and, to a lesser extent, DRAM or GPU consumption, leaving storage, networking, and other subsystems unmodelled despite their relevance to node-level energy use.

Metadata lifecycle management also emerges as a common limitation. Rapid container churn, transient pods, and the interaction between Kubernetes and cgroup identifiers can produce incomplete or stale workload associations, affecting attribution stability. Finally, attribution models themselves are typically rigid. Most tools hard-code a specific proportionality assumption (commonly CPU time or a single hardware counter) and provide limited support for calibration, uncertainty quantification, or alternative modelling philosophies.

Taken together, these limitations reveal structural gaps in current approaches to container-level energy attribution. They motivate the need for tools that combine high-resolution telemetry handling, transparent and configurable attribution logic, robust metadata management, and principled treatment of uncertainty. The next section distills these observations into concrete research gaps that inform the design objectives of Tycho.

2.5 Research Gaps

This section synthesises the findings from the preceding analyses of telemetry sources, temporal behaviour, and existing tools. Across these perspectives, a set of structural limitations emerges that fundamentally constrains accurate and explainable energy attribution in Kubernetes environments. These limitations arise at three intertwined layers: the measurement interfaces exposed by hardware and kernel subsystems, the attribution models built on top of these measurements, and the operational context in which Kubernetes workloads execute. Taken together, they demonstrate the absence of a framework that provides high temporal precision, transparent modelling assumptions, and robustness to container lifecycle dynamics. The gaps identified below define the technical requirements that motivate the design of Tycho.

(1) Measurement Gaps: Temporal Resolution and Telemetry Semantics

Existing tools do not exploit the full temporal capabilities of modern hardware telemetry. Interfaces such as RAPL offer fast, reliable update frequencies, yet tools operate on fixed multi-second loops, causing short-lived or bursty activity to be temporally averaged away. Moreover, latency mismatches between high-frequency utilization signals (e.g. cgroups, perf counters) and low-frequency power interfaces (e.g. Redfish/BMC) introduce structural attribution errors that are not explicitly modelled or corrected.

A related issue is the opacity of hardware telemetry. RAPL, NVML, and BMC power sensors provide indispensable data, but their domain boundaries, averaging windows, and internal update behaviour are insufficiently documented. This prevents rigorous interpretation of reported values and inhibits the development of calibration or uncertainty models. Finally, measurement coverage remains incomplete: while CPU and DRAM domains are widely supported, no standardised telemetry exists for storage, networking, or other subsystems. Current tools treat these components either implicitly (as part of "platform" power) or not at all.

(2) Attribution Model Gaps: Rigidity, Idle Power, and Domain Consistency

Current attribution models rely on rigid proportionality assumptions (typically CPU time, instructions, or a single hardware counter) without considering alternative modelling philosophies. Idle power remains a persistent source of inconsistency: tools variously divide it evenly, proportionally, or not at all, often without documenting the rationale. These choices have substantial effects on per-container energy values, particularly in lightly loaded or heterogeneous systems.

At the domain level, attribution methods are not unified. CPU, DRAM, uncore, GPU, and platform energy are treated through incompatible heuristics, and many domains fall back to equal distribution when no clear usage signal is defined. None of the surveyed tools quantify uncertainty, despite relying on noisy, coarse, or undocumented telemetry sources. As a result, attribution outputs appear deterministic even when they rest on incomplete or ambiguous measurement assumptions.

(3) Metadata and Lifecycle Gaps: Churn, Timing, and Virtualization

Container-level attribution requires consistent mapping between processes, cgroups, and Kubernetes metadata. Existing tools struggle in scenarios with rapid container churn, ephemeral or Completed pods, and multi-process containers. Mismatches between metadata refresh cycles and metric sampling lead to stale or missing associations, which propagate into attribution artefacts.

Energy attribution inside virtual machines remains essentially unsolved. No standard mechanism exists for exposing host-side telemetry to guest systems in a way that preserves temporal alignment and attribution consistency. The limited QEMU-based passthrough available in Scaphandre is not generalisable, and conceptual proposals (e.g. Kepler's hypercall mechanism) remain unimplemented. Given the prevalence of cloud-hosted Kubernetes clusters, this constitutes a major practical limitation.

(4) Usability, Transparency, and Operational Gaps

For most tools, implementation assumptions, fallback paths, and attribution decisions are implicit. Users cannot easily distinguish measured values from estimated ones, nor identify the assumptions underlying attribution outputs. This lack of transparency reduces trust and complicates debugging.

Operational constraints further restrict applicability. Tools that require privileged kernel instrumentation (eBPF, `perf_event_open`) are unsuitable for many production clusters, while tools designed around unprivileged access often sacrifice

modelling fidelity. At the same time, developers, operators, and researchers have fundamentally different observability needs, yet existing tools optimise for only one audience at a time. None provide configurable attribution modes or role-specific abstractions.

(5) Missing Support for Calibration and Validation

Beyond isolated exceptions, existing tools provide limited mechanisms for systematic calibration or validation of their attribution models. KubeWatt is one of the few systems that performs explicit baseline calibration, offering both an idle-power measurement mode and a statistical fallback for environments without idle windows. Kepler offers no structured calibration workflow, and its estimator models lack reproducible training procedures. SmartWatts introduces online model recalibration but focuses narrowly on performance-counter regression, leaving node-level baselines, multi-domain alignment, and external ground-truth integration unaddressed.

Across all tools, there is no standardized path to incorporate external measurements (for example from wall-power sensors or BMC-level telemetry) to validate or refine model behaviour. Idle power is seldom isolated as a first-class parameter, attribution error is rarely quantified, and no system provides uncertainty estimates that reflect measurement or modelling limitations. Without such calibration and validation capabilities, attribution accuracy cannot be assessed, corrected, or improved over time—an essential requirement for any system intended to provide trustworthy, high-resolution energy insights in Kubernetes environments.

2.6 Summary

The analyses in this chapter reveal that modern server platforms provide a heterogeneous and only partially documented set of telemetry interfaces whose temporal and semantic properties fundamentally constrain container-level energy attribution. Hardware-integrated sources such as RAPL and NVML expose valuable domain-level energy information but differ substantially in update behaviour, averaging semantics, and domain completeness. Out-of-band telemetry via Redfish provides stable whole-system measurements but at coarse and irregular temporal granularity. Software-exposed metrics offer fine-grained visibility into workload behaviour, yet they measure utilisation rather than power and depend entirely on polling strategies for temporal interpretation.

Temporal irregularities, internal averaging, and undocumented sensor behaviour reduce the effective precision of all telemetry sources, especially when attempting to capture short-lived workload dynamics. Existing tools aggregate these heterogeneous signals using fixed multi-second sampling loops and rigid proportionality assumptions, which leads to systematic attribution artefacts. Idle power is treated inconsistently across systems, residual power is frequently conflated with workload activity, and multi-domain attribution remains fragmented in both semantics and implementation. Metadata churn and asynchronous refresh cycles further complicate the mapping between processes, containers, and Kubernetes abstractions, reducing the stability and interpretability of attribution outputs.

The cross-tool evaluation confirms these structural limitations. Kepler provides broad telemetry integration but struggles with timing mismatches, incomplete domain semantics, and opaque idle handling. Kubewatt demonstrates the importance of explicit baseline separation and cgroup filtering, yet remains limited to single-domain CPU-based estimation. Other tools illustrate a spectrum of design choices but do not address the combined challenges of high-frequency telemetry, multi-domain attribution, container lifecycle dynamics, and Kubernetes integration.

Together, these findings motivate the need for a framework that provides high temporal fidelity, transparent modelling assumptions, unified domain treatment, explicit handling of idle and residual energy, and robust metadata reconciliation. These requirements form the conceptual and architectural foundations developed in Chapter **??**, which introduces the methodological principles guiding the design of Tycho.

Chapter 3

Experimental Evaluation of Tycho

3.1 Evaluation Scope and Evidence Types

This chapter evaluates Tycho as an accuracy-focused energy measurement and attribution system operating under realistic observability constraints. The objective is not to establish ground-truth energy values, but to assess whether Tycho's outputs are internally consistent, temporally coherent, and physically interpretable given the characteristics of its input signals.

Tycho integrates heterogeneous measurement sources spanning hardware, firmware, kernel, and container layers, each with distinct temporal resolution, latency, and semantic scope. Accordingly, many relevant correctness properties are structural rather than scalar, concerning conservation, attribution completeness, stability under repeated execution, and coherent cross-layer behavior rather than numerical agreement with an external reference.

Two complementary evidence types are used, ordered by dependency.

First, Tycho underwent continuous qualitative validation throughout development. New mechanisms were exercised until their behavior aligned with design intent across relevant operating regimes before further features were added. Validation relied on direct inspection of intermediate system state, including logs and analysis traces, live dashboards, targeted host- and Kubernetes-level stress scenarios, and systematic cross-checking of raw input signals against derived attribution outputs. This tier establishes internal correctness, semantic coherence, and robustness against structural failure modes such as energy leakage, temporal misalignment, or attribution instability.

Second, Tycho was exercised through a structured repeated execution plan comprising 30 repetitions of a multi-phase workload suite on both an idle and a busy server. These runs provide broad scenario coverage for additional consistency checks and yield a controlled dataset from which a small number of representative workload scenarios are analyzed in depth. The resulting analyses are illustrative rather than verificatory, exposing Tycho's behavior, capabilities, and boundary conditions under realistic execution. Where repetitions are evaluated quantitatively, they serve only to illustrate stability and variability; no claims of statistical significance are made.

Together, these two evidence tiers support an evaluation focused on interpretability, consistency, and robustness within the limits imposed by the available measurement

signals.

3.2 Qualitative Validation and Consistency Assessment

This section establishes internal correctness and semantic coherence of Tycho through invariant-driven qualitative validation, providing the foundation on which the targeted experimental analyses in the following section rely.

3.2.1 Invariant-Driven Qualitative Validation Methodology

Qualitative validation in this work is defined as the systematic verification of structural correctness properties that must hold independently of workload choice, operating regime, or numerical calibration. Rather than evaluating individual output values against an external reference, validation focuses on invariants whose violation would render attribution results uninterpretable, irrespective of apparent numerical plausibility.

Validation was performed continuously and at feature granularity throughout development. Each newly introduced mechanism was exercised until its behavior aligned with design intent across the relevant operating regimes before subsequent features were added. This process relied on direct inspection of intermediate system state, including structured logs and analysis traces, live dashboards, and selectively enabled debug metrics exposing internal buffers, windows, and attribution state. Targeted host- and Kubernetes-level stress scenarios were used to provoke boundary conditions under which structural failures would be expected to surface.

The validation strategy was explicitly invariant-driven. Core properties monitored across all executions included conservation of energy across attribution boundaries, monotonicity of counters, completeness of attribution, stable separation of idle, dynamic, and residual components, and coherent behavior under repeated, steady, bursty, and concurrent workloads. These invariants impose global constraints on system behavior and provide strong failure detectability: violations propagate across layers and time windows and manifest as observable inconsistencies between raw inputs, intermediate representations, and exported metrics.

No artifact-oriented quantitative validation dataset was produced for this tier. This is a consequence of the validation objective rather than a limitation of testing effort. Correctness was established through sustained, adversarial inspection under diverse execution conditions, where invariant violations would be immediately visible. Accordingly, this validation tier establishes internal correctness and semantic coherence of Tycho's attribution pipeline, but does not claim numerical accuracy with respect to an external ground truth.

3.2.2 Layer-Specific Validation Focus

Qualitative validation was applied across all major layers of the Tycho pipeline, with layer-specific checks designed to surface structural failures at their point of origin while preserving end-to-end interpretability.

3.2.2.1 Signal Ingestion and Metric Integrity

Validation of signal ingestion focused on excluding classes of errors that would compromise the semantic integrity of downstream attribution. Given the heterogeneity of Tycho's input signals, correctness at this layer is primarily concerned with continuity, plausibility, and stable interpretation over time rather than with numerical agreement across sources.

The principal ingestion risks considered were missing or irregular samples, counter discontinuities, inconsistent units or scaling, unstable identity mapping across devices or workloads, and silent gaps that could propagate undetected into higher-level metrics. Validation therefore emphasized monotonicity of cumulative counters, absence of unexplained discontinuities, stable labeling and device identity resolution across successive observations, and predictable behavior under steady, bursty, and concurrent activity.

Independent signal sources were routinely cross-checked for qualitative plausibility. CPU, GPU, and system-level metrics were inspected jointly to ensure that observed activity patterns were mutually consistent and that energy attribution in one domain did not exhibit unexplained coupling to unrelated activity in another. Sustained agreement in qualitative behavior across independent sources provided confidence that ingestion pipelines preserved intended semantics.

Known limitations of individual sources were explicitly accounted for during validation. In particular, system-level power metrics obtained via Redfish were observed to exhibit variable reporting latency relative to CPU and GPU counters. This behavior was treated as an expected characteristic of the source rather than as an ingestion defect. Validation confirmed that such latency manifested as delayed but coherent updates, without introducing discontinuities, counter violations, or semantic contradictions that would undermine subsequent analysis.

3.2.2.2 Temporal Coherence and Delay Handling

Temporal validation focused on ensuring coherent analysis behavior in the presence of heterogeneous timing characteristics, asynchronous updates, and source-specific latency. Given that Tycho does not enforce strict timestamp alignment at ingestion, correctness at this layer depends on stable window materialization, preserved causal ordering, and predictable interpretation of delayed signals during analysis.

Validation targeted failure modes such as temporal misalignment between sources, unstable or inconsistent window boundaries, sensitivity to short-term jitter, and systematic bias introduced by delayed updates. Intermediate timing state, analysis windows, and derived metrics were inspected under controlled transitions between idle, steady, bursty, and concurrent workloads, as well as during long-running executions where accumulated drift or buffer exhaustion would become observable.

Variable latency of system-level power metrics was a central consideration. Rather than attempting to minimize or mask such delay, validation emphasized robustness of temporal fusion, confirming that late-arriving samples were incorporated consistently without violating conservation, monotonicity, or attribution completeness. Pathological temporal behavior would manifest as negative attribution, unstable

residuals, or inconsistent phase transitions; no such effects were observed outside explicitly characterized boundary conditions.

Observed lag and smoothing effects in derived metrics were therefore interpreted as expected consequences of source characteristics and aggregation granularity rather than as temporal failures. Under all tested execution regimes, temporal handling preserved stable, interpretable behavior across interacting metric streams.

3.2.2.3 Attribution Semantics and Conservation

Validation of attribution semantics centered on enforcing conservation of energy as a non-negotiable invariant. For every analysis window, all observed energy must be accounted for exactly once, either through explicit workload attribution or as transparently exposed residual or system-level energy, up to numerical tolerance. Any silent loss, duplication, or implicit discarding of energy would immediately undermine interpretability and was therefore treated as a structural failure.

Validation focused on confirming stable separation and interaction of idle, dynamic, and residual components across execution regimes. Attribution behavior was inspected under steady, bursty, and concurrent workloads to ensure that conservation and attribution completeness held independently of scheduling dynamics, workload overlap, or resource contention. Violations would manifest as unexplained discrepancies between aggregate and attributed energy or as unstable residual behavior and were actively monitored throughout development and testing.

Boundary cases were explicitly identified and characterized. In particular, extreme latency of certain system-level power signals can induce transient distortions in residual attribution. These effects are attributable to input signal characteristics rather than to attribution semantics and remain bounded and detectable. Their presence does not violate conservation and does not compromise interpretability when properly contextualized.

3.2.2.4 End-to-End Observability and Sanity Checks

End-to-end observability served as a primary validation instrument throughout development. Tycho was continuously inspected as a live system across multiple abstraction levels, including raw input metrics, intermediate analysis state, and exported outputs, enabling direct cross-checks between independently derived views of system behavior.

Validation emphasized a small set of canonical sanity patterns that must hold for any interpretable attribution system. These included predictable responses to workload start and termination, stability under steady-state execution, symmetry under equivalent workload configurations, and absence of unexplained oscillations or phase-dependent artifacts. Departures from these patterns would indicate semantic or temporal inconsistencies and were actively sought during exploratory and adversarial testing.

Tycho's debug mode, which selectively exposes internal buffers, windows, and attribution state, enabled targeted inspection of intermediate behavior during runtime. This capability allowed rapid localization of potential issues and ensured that

observed high-level behavior was grounded in coherent internal state rather than emergent artifacts.

Across sustained observation under diverse workload regimes, raw metrics, intermediate analysis state, and exported outputs converged toward consistent qualitative behavior. No unexplained or contradictory patterns were observed, providing strong assurance of end-to-end semantic coherence.

3.2.3 Summary of Qualitative Assurance

The qualitative validation described above establishes high confidence in Tycho's internal correctness and semantic coherence. Across sustained and adversarial inspection of all major subsystems, no violations of fundamental invariants, unexplained inconsistencies, or contradictory behaviors were observed. This assurance provides the foundation for the targeted experimental analyses that follow, which illustrate Tycho's behavior and limitations under concrete workload scenarios rather than reestablishing correctness.

3.3 Targeted Experimental Evaluation

This section presents a set of targeted, end-to-end experiments designed to illustrate Tycho's behavior under structured and repeatable workload conditions. The experiments do not serve to verify correctness, which is established through qualitative validation, but to expose observable behavior, strengths, and boundary conditions in concrete scenarios. Accordingly, the analyses are descriptive and interpretive, and no claims of formal correctness or statistical significance are made.

3.3.1 Overview of Executed Test Scenarios

All targeted experiments were derived from a single declarative test plan and executed end-to-end under controlled and repeatable conditions. This subsection describes the execution model, test environment, and scenario structure, and serves as a reference point for the analyses presented in the following sections.

3.3.1.1 Test Setup and Execution Environment

All experiments were performed on a dedicated compute node acting as the system under test (SUT). The node was a DALCO-integrated server (product `G494-ZU0-AAP1-000`, MegaRAC SP-X BMC) equipped with an AMD EPYC 9554 (64 cores), 192 GB DDR5 memory, dual NVMe storage devices, and two GPUs (NVIDIA RTX 4000 Ada and NVIDIA T4). The SUT executed only the monitoring stack, workload pods, and the Tycho exporter, ensuring that observed signals reflect workload and system behavior rather than Kubernetes control-plane activity.

The operating system was Ubuntu 22.04 LTS with a Linux 6.8 kernel (`6.8.0-90-generic`, `PREEMPT_DYNAMIC`) on `x86_64`. GPU support was provided by the NVIDIA proprietary driver (`580.95.05`) with CUDA 13.0. Kubernetes was deployed directly on the servers using PowerStack [1]. To minimize measurement interference, the Kubernetes control plane ran on a separate node, and the SUT operated exclusively as a worker. Full root access was available and required for eBPF-based instrumentation.

A known hardware limitation of the AMD platform is the absence of a RAPL DRAM energy domain. All Tycho functionality related to DRAM attribution was therefore validated independently on an Intel-based control node and is excluded from the AMD-based experimental scenarios presented here.

Experimental analyses focus on energy domains and metrics that are fully supported and observable on the SUT, in particular CPU package and core energy via RAPL and GPU energy via vendor telemetry. These domains provide stable counters and sufficient dynamic range for illustrating attribution behavior under the tested workload scenarios.

3.3.1.2 Test Plan Structure and Scenario Catalog

All experiments were executed from a single declarative test plan that defines workload structure, execution order, repetition count, and timing parameters. An Ansible-based automation workflow materializes the plan, provisions an isolated Kubernetes testing namespace, and orchestrates execution end-to-end without manual intervention. This approach ensures reproducibility across runs while allowing workload structure to be modified independently of deployment mechanics.

Unless stated otherwise, all scenarios were executed with 30 repetitions and analyzed at a fixed aggregation granularity of 3 s. Each repetition executes the complete scenario sequence in a deterministic order, yielding a consistent temporal structure across runs.

The test plan consists of an ordered sequence of workload scenarios, each designed to exercise a specific attribution-relevant aspect of Tycho's behavior. Explicit sleep phases are inserted between all workload phases to allow the system to return to an idle state and to reduce thermal, scheduling, and buffer carry-over effects. For analysis, only steady-state execution intervals are considered; warm-up phases and transient effects at workload boundaries are excluded.

The following catalog enumerates the executed scenarios and serves as the reference structure for the targeted analyses presented in the subsequent subsections.

1. **Idle baseline (start).** Establishes the initial noise floor and short-term stability of idle attribution.
    - Duration: 180 s
    - Workloads: none (sleep only)
    - Purpose: idle variance estimation, initial drift detection
2. **CPU warm-up ramp.** Transitional phase used exclusively for preconditioning.
    - Workload: `stress-ng` CPU ramp (`matrixprod`)
    - Duration: 60 s
    - Structure: stepped load increase
    - Purpose: stabilization of subsequent CPU measurements

- Note: excluded from quantitative analysis

3. **CPU steady baseline (matrix multiplication).** Reference case for steady-state CPU dynamic energy attribution.
   - Workload: `stress-ng` CPU (`matrixprod`)
   - Duration: 120 s
   - CPU request: 16 000 mCPU
   - Concurrency: single pod
   - Purpose: steady-state stability and repeatability
4. **CPU burst train.** Periodic on/off CPU activity to probe temporal aggregation and duty-cycle behavior.
   - Workload: `stress-ng` CPU burst (`matrixprod`)
   - Duration: 180 s
   - Structure: 9 s on / 9 s off
   - CPU request: 16 000 mCPU
   - Purpose: burst attribution fidelity and temporal smearing
5. **CPU jitter train.** Irregular CPU burst timing to assess robustness under non-uniform activity.
   - Workload: `stress-ng` CPU jitter (`matrixprod`)
   - Duration: 180 s
   - Structure: random gaps between 3 s and 15 s
   - CPU request: 16 000 mCPU
   - Purpose: attribution stability under irregular timing
6. **CPU discrimination (concurrent, heterogeneous pair I).** Concurrent execution of computationally distinct CPU workloads.
   - Workloads: `stress-ng` `int128` and `fft`
   - Duration: 180 s
   - CPU request: 8 000 mCPU per workload
   - Concurrency: two pods
   - Purpose: workload-level energy differentiation
7. **CPU discrimination (concurrent, heterogeneous pair II).** Alternative discrimination scenario with different computational characteristics.

- Workloads: `stress-ng` `bitops` and `matrixprod`
- Duration: 180 s
- CPU request: 8 000 mCPU per workload
- Concurrency: two pods
- Purpose: method-dependent energy signatures

8. **CPU idle-allocation fairness (busy vs. noop).** Co-scheduled active and inactive workloads to probe idle energy assignment semantics.
   - Workloads: active `stress-ng` CPU (`bitops`) and idle sleep pod
   - Duration: 180 s
   - CPU request: 8 000 mCPU per workload
   - Purpose: idle energy sharing behavior
9. **GPU idle baseline.** Baseline measurement of GPU idle energy in the absence of GPU workloads.
   - Duration: 120 s
   - Workloads: none (GPU idle)
   - Purpose: GPU idle stability and cross-domain isolation
10. **GPU steady baseline.** Reference case for steady-state GPU dynamic energy attribution.
    - Workload: steady GPU burn
    - Duration: 180 s
    - GPU request: 1 device
    - Purpose: stability and repeatability of GPU attribution
11. **GPU burst train.** Periodic GPU activity to probe temporal aggregation limits.
    - Workload: GPU burst burn
    - Duration: 240 s
    - Structure: 3 s on / 3 s off
    - GPU request: 1 device
    - Purpose: burst visibility under coarse aggregation
12. **GPU concurrency (two workloads).** Concurrent GPU workloads to assess proportional energy splitting.

- Workloads: two identical steady GPU burns
- Duration: 180 s
- GPU request: 1 device per workload
- Purpose: two-way GPU energy attribution

13. **GPU concurrency (three workloads).** Higher-concurrency GPU scenario to probe scaling and slicing robustness.
    - Workloads: three identical steady GPU burns
    - Duration: 180 s
    - GPU request: 1 device per workload
    - Purpose: multi-workload GPU attribution consistency
14. **Idle baseline (end).** Re-establishes idle conditions after all workload phases.
    - Duration: 180 s
    - Workloads: none (sleep only)
    - Purpose: end-of-run drift and stability check

3.3.2 CPU Idle-Allocation Fairness (Busy vs. Noop)

Objective This scenario probes Tycho's idle CPU energy attribution semantics under concurrent workloads with identical resource declarations, contrasting an active workload with an idle one.

Observed Behavior Figures 3.1 and 3.2 show the workload-level CPU package power attributed to an active and a noop pod under idle and busy system conditions, respectively.

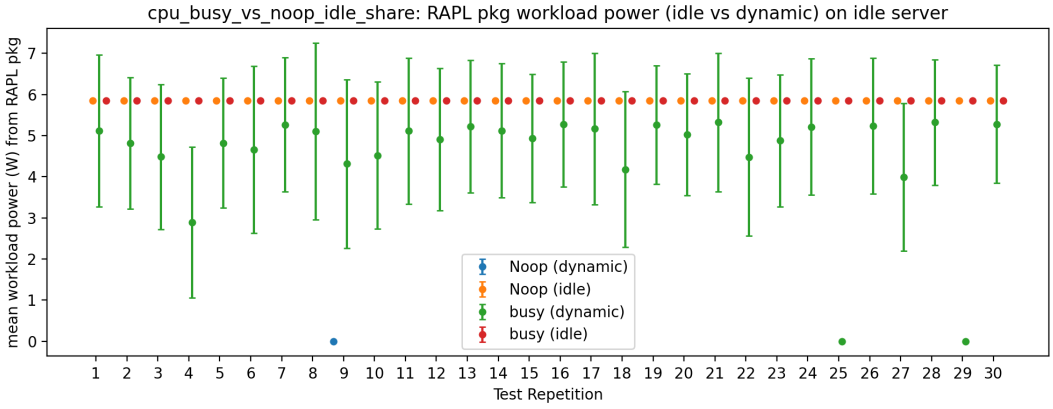

FIGURE 3.1: Idle-node execution: RAPL PKG workload power attributed to a busy and a noop pod. Dynamic power is attributed exclusively to the active workload, while idle power is evenly shared between both pods across repetitions.

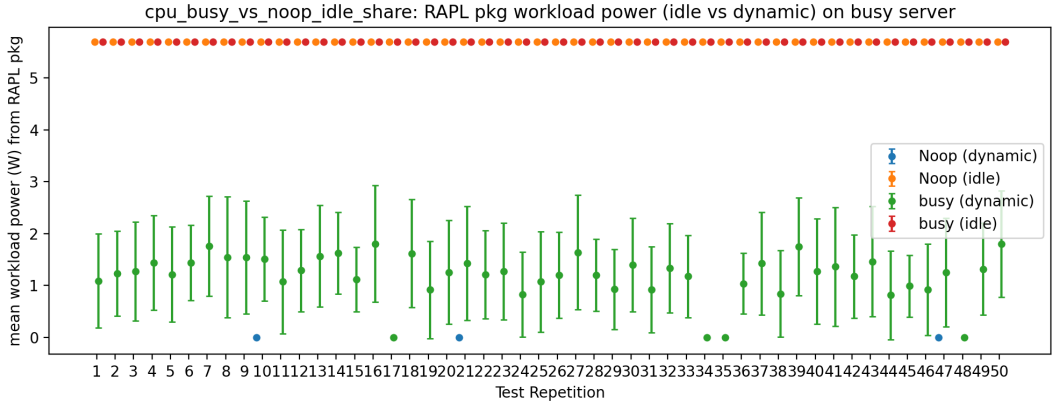

FIGURE 3.2: Busy-node execution: RAPL PKG workload power under background load. Idle power attribution remains unchanged, while dynamic power attributed to the active workload is reduced due to contention.

In both executions, the active workload is consistently attributed substantial dynamic CPU energy, while the noop workload exhibits no measurable dynamic component. In contrast, idle CPU package energy is attributed nearly equally to both workloads despite their markedly different execution behavior. Under background load, only the dynamic component attributed to the active workload is reduced.

Interpretation The observed behavior indicates that idle CPU energy is attributed based on declared resources rather than instantaneous execution intensity. Dynamic attribution remains sensitive to effective CPU availability, while idle allocation remains invariant under differing runtime behavior and background system load.

Limitations This scenario does not assess absolute attribution accuracy, asymmetric resource requests, or fine-grained temporal effects outside steady-state execution.

3.3.3 CPU Discrimination under Heterogeneous Concurrent Workloads

Objective This scenario examines whether Tycho can consistently distinguish energy consumption between heterogeneous CPU-bound workloads executing concurrently under identical resource declarations.

Observed Behavior Figures 3.3 and 3.4 show the workload-level CPU core-domain power attributed to four heterogeneous workloads under idle and busy system conditions, respectively.

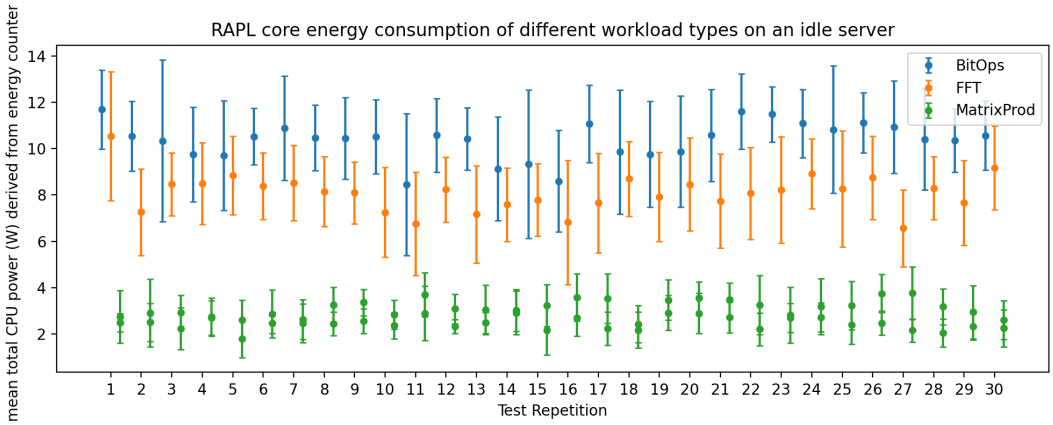

FIGURE 3.3: Idle-node execution: RAPL core-domain workload power attributed to four heterogeneous CPU workloads with identical resource requests. Distinct workload-dependent energy consumption levels are consistently observable across repetitions.

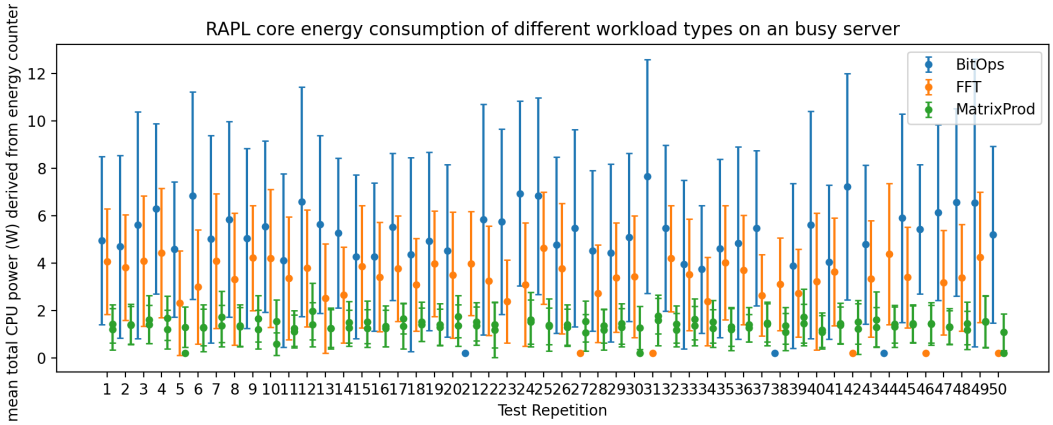

FIGURE 3.4: Busy-node execution: RAPL core-domain workload power under background load. Variance increases due to contention, while the relative separation between workload types remains clearly visible.

Across repetitions on the idle system, each workload exhibits a clearly distinguishable energy consumption profile. The relative ordering between workload types remains stable, with substantial separation between their attributed power levels. Under background system load, the overall variance of the attributed energy increases, but the qualitative separation between workloads persists.

Interpretation The observed behavior indicates that Tycho preserves workload-specific energy attribution characteristics under concurrent execution. While contention affects absolute magnitude and variability, it does not eliminate relative differentiation between computational patterns with distinct execution characteristics.

Limitations This scenario does not assess absolute attribution accuracy, short-lived execution phases, or sensitivity to differing CPU request sizes, nor does it explore interactions with non-CPU resources.

3.3.4 GPU Workload Separation under Concurrent Execution

Objective This scenario evaluates whether Tycho can attribute GPU energy independently to concurrent workloads executing on different physical devices.

Observed Behavior Figure 3.5 shows the workload-level GPU power attributed to two concurrent workloads, each bound to a distinct physical GPU.

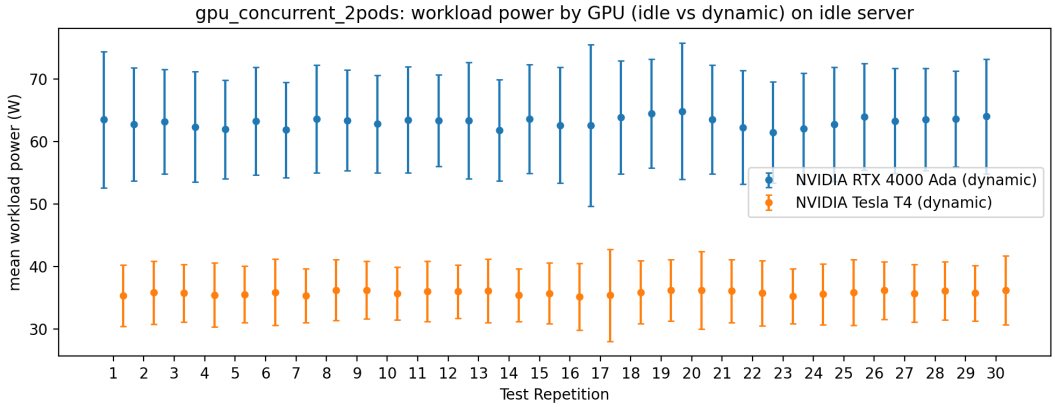

FIGURE 3.5: Concurrent GPU workloads executing on two physical GPUs. The higher-performance device consistently exhibits higher power consumption across repetitions. Observed attribution remains stable under background CPU load.

Across all repetitions, the two GPUs exhibit stable and clearly separated energy consumption levels. The higher-performance device consistently consumes more power than the lower-tier GPU, with limited variability between runs. Background CPU load does not introduce any systematic change in the attributed GPU power for either workload.

Interpretation The observed behavior indicates that GPU energy attribution in Tycho is device-specific and isolated from host CPU activity. Concurrent execution on multiple GPUs does not introduce cross-device interference in the attributed energy signals.

Limitations This scenario does not examine contention on shared GPU resources, short-lived kernels, or mixed GPU workloads on a single device.

3.3.5 GPU Workload Behavior under Oversubscription

Objective This scenario examines GPU energy attribution under oversubscription, where more concurrent workloads are present than available physical GPUs, requiring multiple workloads to share a single device via time-multiplexed execution.

Observed Behavior Figure 3.6 shows the workload-level GPU power attributed to three concurrent workloads executing on two physical GPUs, resulting in one single-workload and one two-workload device configuration.

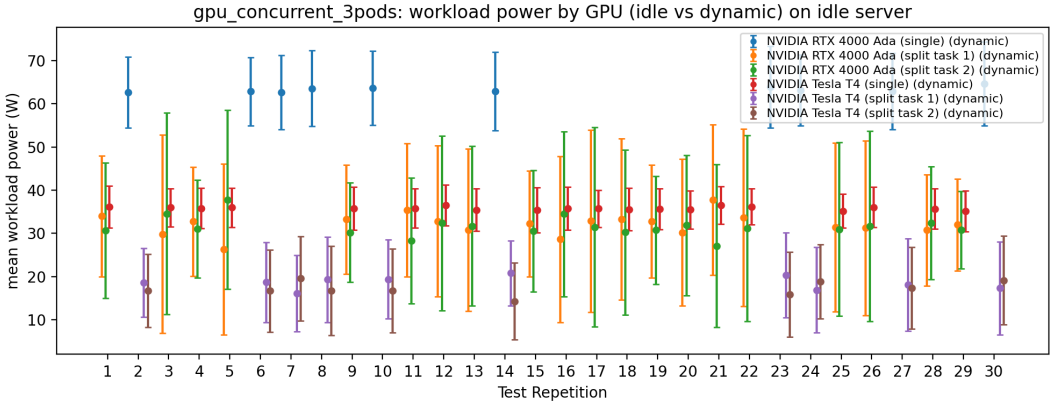

FIGURE 3.6: Three concurrent GPU workloads executing on two physical GPUs. Single-workload execution exhibits stable power with low variance, while co-located workloads share GPU power and show increased variability due to time-multiplexed execution. Total device power remains approximately constant across configurations.

When a GPU executes a single workload, the attributed power is high and consistent, with comparatively low variance across repetitions. When two workloads share a GPU, both exhibit similar average power consumption, while the total power attributed to the device remains approximately constant. Workloads co-located on the same GPU show a noticeably higher variance than single-workload execution, consistent with time-multiplexed scheduling. These patterns remain stable across repetitions and are unaffected by background CPU load.

Interpretation The observed behavior indicates that Tycho conserves GPU energy under oversubscription and partitions device power across concurrent workloads without duplication or inflation. Increased variability reflects time-multiplexed execution rather than attribution instability.

Limitations This scenario does not examine finer-grained GPU scheduling behavior, short-lived kernels, memory-bound workloads, or oversubscription levels beyond two concurrent workloads per device.

3.4 Summary of Evaluation Findings

The evaluation presented in this chapter demonstrates that Tycho exhibits stable, interpretable, and internally consistent behavior across a diverse range of workload patterns and system conditions. Qualitative validation establishes confidence in the correctness and semantic coherence of the attribution pipeline, while targeted experimental scenarios illustrate how these properties manifest under realistic execution.

Taken together, the results show that Tycho preserves key structural invariants, responds predictably to changes in workload structure and system load, and exposes meaningful attribution behavior within the limits imposed by available measurement signals. The evaluation does not aim to establish numerical ground truth accuracy, but provides a sound empirical basis for reasoning about Tycho's capabilities, limitations, and intended use as an analysis and research instrument.

A synthesized interpretation of these findings and their implications is presented in the following chapter.

Chapter 4

Evaluation and Synthesis

4.1 Purpose and Structure of the Evaluation

This chapter provides an interpretive synthesis of the results established in the preceding experimental evaluation. It does not introduce new experiments, data, metrics, or figures. Instead, it evaluates the observed behavior of the system at a global level, with the aim of assessing what the empirical findings imply about Tycho as a measurement approach and as a research instrument.

The focus of this chapter is therefore not on individual workloads or test cases, but on recurring patterns, system-level properties, and the consequences of the design decisions explored throughout the thesis. Evaluation is conducted at the level of measurement semantics, attribution behavior, and interpretability under realistic constraints imposed by external telemetry and system heterogeneity.

To provide structure to this assessment, the chapter is organized around the research questions introduced earlier in the thesis. These questions are used as a guiding framework for synthesizing the empirical and qualitative evidence, rather than as independent hypotheses or isolated evaluation targets. Each question is discussed analytically and answered explicitly, with attention to the assumptions and boundary conditions under which the conclusions hold.

This chapter thus serves as an intermediate layer between the presentation of empirical evidence and the concluding discussion. While the experimental evaluation establishes what was observed, the following sections articulate how these observations should be interpreted, what limitations they reveal, and what can be concluded about the scope and validity of the proposed measurement approach.

4.2 Global Summary of Empirical Findings

Across the evaluated scenarios, several consistent patterns emerged that characterize Tycho's behavior as a measurement system.

These observations form the empirical basis for the following evaluation and are not tied to individual experiments, but to recurring system-level behavior across the experimental space explored in this thesis.

4.3 Research Question 1: Guarantees and Feasibility of Accuracy-First Measurement

> What properties can an accuracy-first energy measurement system realistically guarantee when attributing energy consumption to workloads in Kubernetes environments with heterogeneous and delayed telemetry sources?

Interpretation

In the context of workload-level energy attribution, the term *guarantee* does not refer to numerical accuracy bounds or ground-truth recovery. Instead, it denotes semantic and structural properties that are enforced by system design and that hold independently of workload behavior, as long as stated assumptions about input telemetry are respected. An accuracy-first measurement system is therefore evaluated not by how closely its outputs approximate an unknown true value, but by which invariants it preserves, which failure modes it exposes explicitly, and how its outputs relate to its inputs under imperfect observability.

Within this interpretation, Tycho's guarantees arise from construction rather than inference. The system enforces closed energy accounting within its modeled scope, ensuring that all observed system energy is either attributed to explicit workloads or represented as residual energy. Attribution decisions are derived from measured quantities wherever possible and from explicitly modeled relations otherwise, avoiding silent redistribution or implicit smoothing. Temporal coherence is established by accounting for source-specific observation delays during analysis, ensuring that correlations across heterogeneous metrics reflect real system behavior rather than artifacts of misaligned timelines.

Accuracy-first design in this sense does not imply correctness under all conditions. Instead, it prioritizes explicitness about what can and cannot be guaranteed. When telemetry is coarse, delayed, or internally inconsistent, these limitations are surfaced in the attribution outputs rather than masked by heuristic correction. As a result, Tycho's guarantees are best understood as properties of interpretability, consistency, and bounded meaning, rather than as claims about absolute precision.

Boundary Conditions

The guarantees described above are conditional on the quality and granularity of the available telemetry. In particular, system-level energy sources with low temporal resolution and variable delay constrain the strength of attribution at fine time scales. While Tycho explicitly models such behavior and mitigates its effects through temporal alignment and multi-source fusion, it cannot fully compensate for missing, delayed, or internally unstable measurements. In these cases, attribution results may exhibit increased residual energy or reduced temporal specificity, reflecting limitations of the upstream data rather than failures of the attribution logic.

Similarly, Tycho's guarantees apply within a Kubernetes-aware attribution scope and do not extend to cross-node effects, scheduler behavior, or cluster-wide fairness properties. Non-negativity and plausibility of derived metrics are not enforced unconditionally, but violations are explicitly exposed when they arise, allowing implausible values to be interpreted as indicators of telemetry or modeling limits. This

design choice favors transparency over cosmetic validity and ensures that weakening assumptions result in degraded interpretability rather than misleading confidence.

Answer to Research Question 1

> An accuracy-first energy measurement system can realistically guarantee structural and semantic properties of attribution rather than numerical accuracy in the absolute sense. Under stated assumptions about telemetry quality and timing, Tycho guarantees closed energy accounting within its modeled scope, explicit representation of unattributed energy, and temporally coherent attribution based on modeled source delays. These guarantees are enforced by construction and degrade explicitly when upstream measurements become coarse, delayed, or inconsistent. Accuracy-first measurement therefore does not eliminate estimation, but ensures that its limits are visible, bounded, and interpretable rather than implicit.

4.4 Research Question 2: Interpretability and Explanatory Power of Semantics-Driven Attribution

> How does an explicit, semantics-driven attribution methodology influence the interpretability and explanatory power of workload-level energy measurements compared to implicit or estimation-oriented approaches?

Interpretation

In this research question, interpretability refers to the scientific interpretability of workload-level energy measurements, that is, the extent to which reported values have well-defined semantics, obey declared invariants, and can be meaningfully reasoned about in relation to system behavior. Explanatory power denotes the ability of a measurement system to support causal reasoning, such as explaining differences between workload behaviors or relating observed energy consumption patterns to changes in system activity. Both notions depend less on numerical precision than on semantic clarity, internal consistency, and the explicit treatment of uncertainty.

A semantics-driven attribution methodology emphasizes explicit meaning over presentation-oriented continuity. In Tycho, this is realized through explicit idle and dynamic energy decomposition, closed accounting with residual energy, delay-aware temporal alignment, and the absence of silent smoothing or implicit gap filling. Each reported metric corresponds to a declared conceptual quantity, and deviations from expected behavior are preserved rather than heuristically suppressed. As a result, attribution outputs can be interpreted as constrained measurement statements rather than as interpolated estimates optimized for visual continuity or completeness.

Compared to implicit or estimation-oriented approaches, which often rely on hidden redistribution, smoothing, or model-driven gap filling, explicit semantics shift the role of attribution from producing a single best-looking signal to exposing the structure and limits of what can be inferred from the available data. This shift enhances explanatory power by allowing observed patterns to be analyzed in terms

of conservation, temporal coherence, and cross-metric consistency. Differences between workloads or system states can be examined through stable sanity patterns, such as consistent decomposition behavior or agreement between component-level and system-level observations, rather than inferred indirectly from smoothed aggregates.

Boundary Conditions

The interpretability benefits of explicit, semantics-driven attribution are bounded by the quality of the underlying telemetry and by the scope of the modeled system. When measurements are coarse, delayed, or incomplete, explicit semantics do not eliminate uncertainty but instead make it visible through residuals, reduced temporal specificity, or weakened cross-metric agreement. While this behavior supports scientific reasoning, it may be less suitable in contexts where uninterrupted or visually smooth signals are prioritized over semantic transparency.

Furthermore, semantics-driven attribution entails higher system complexity and overhead, as well as stronger requirements on the execution environment. Access to low-level telemetry, elevated privileges, and bare-metal deployment are often necessary to preserve semantic fidelity. These requirements limit applicability in virtualized or tightly constrained production environments, but are typically acceptable in experimental and research settings where interpretability and explanatory power are primary objectives.

Answer to Research Question 2

> An explicit, semantics-driven attribution methodology increases the interpretability and explanatory power of workload-level energy measurements by ensuring that reported values correspond to well-defined conceptual quantities and obey explicit invariants. By avoiding silent smoothing and implicit gap filling, such an approach enables causal reasoning about workload and system behavior, supports consistency checks across metrics, and makes uncertainty and measurement limits visible rather than implicit. These benefits come at the cost of increased complexity and stricter deployment requirements, but provide substantially stronger semantic foundations for scientific analysis than implicit or estimation-oriented attribution approaches.

4.5 Research Question 3: Contexts and Trade-offs of Accuracy-First Energy Measurement

> In which experimental and research contexts does high-fidelity, accuracy-first energy measurement justify its complexity and overhead, and where do simpler approaches remain preferable?

Interpretation

This research question addresses the appropriateness of accuracy-first energy measurement rather than its correctness or interpretability. The core issue is not whether such a system can produce meaningful results, but under which conditions the additional complexity, deployment requirements, and analytical overhead are justified

by the insights gained. This distinction is particularly important for systems such as Tycho, whose design explicitly prioritizes measurement fidelity and semantic clarity over ease of use or minimal system intrusion.

A useful separation can be drawn between the justification of developing an accuracy-first measurement system and the justification of running it in a given environment. From a development perspective, the added complexity of Tycho's architecture enables substantially finer-grained and more informative energy analysis than approaches that rely on coarse aggregation or estimation-heavy models. For research questions that require detailed understanding of workload behavior, causal relationships, or the interaction between system components, this complexity is justified by the quality and explanatory depth of the resulting measurements.

From an operational perspective, the justification depends on measurement intent. Tycho is not designed as a general-purpose observability or monitoring tool. Instead, it is optimized for question-driven analysis, where measurement is performed to investigate specific hypotheses or optimization problems rather than to provide continuous, presentation-oriented system metrics. In such contexts, the additional granularity and explicit semantics provided by accuracy-first measurement enable insights that simpler approaches cannot support.

Boundary Conditions

The applicability of accuracy-first measurement is constrained by practical and environmental factors. Tycho relies on access to low-level telemetry sources, including kernel instrumentation and hardware interfaces, which typically require elevated privileges and bare-metal deployment. These requirements limit its suitability in virtualized or tightly controlled production environments, where such access is often unavailable or undesirable.

While Tycho's runtime overhead remains within the range of existing observability tooling, its analytical complexity and deployment assumptions make it less attractive for scenarios where coarse trends, aggregate estimates, or billing- oriented measurements are sufficient. In such cases, simpler attribution approaches may provide adequate information with lower operational burden and fewer infrastructure constraints.

Deployment complexity has been intentionally addressed through automation, and Tycho can be deployed reproducibly using a single, fully automated Ansible-based workflow provided by the PowerStack framework [1]. This streamlining reduces operational friction but does not alter the fundamental scope and requirements of accuracy-first measurement.

Answer to Research Question 3

> High-fidelity, accuracy-first energy measurement is primarily justified in experimental and research contexts where detailed, semantically grounded insight into workload and system behavior is required. In such settings, the additional complexity and deployment requirements enable forms of analysis and explanation that simpler approaches cannot support. For continuous observability or operational monitoring tasks that prioritize

low overhead and broad applicability over interpretability, simpler attribution methods often remain preferable. Accuracy-first measurement therefore represents a targeted instrument for question-driven analysis rather than a universal solution for energy monitoring.

4.6 Positioning of Tycho Within the Measurement Landscape

This section positions Tycho conceptually within the broader landscape of workload-level energy analysis systems. The goal is not to compare individual tools or approaches, but to clarify along which dimensions Tycho should be evaluated and which design priorities shape its behavior and outputs.

4.6.1 Measurement and Estimation Boundaries

Workload-level energy attribution unavoidably involves estimation. While total system energy consumption can be measured directly, attributing portions of that energy to individual workloads, containers, or execution states requires inference based on partial observability, delayed telemetry, and modeling assumptions. Tycho does not attempt to eliminate this estimative component. Instead, it seeks to constrain it as tightly as possible and to make it explicit where and why estimation is required.

Two guiding principles inform this stance. First, Tycho prioritizes the collection of high-quality, directly observable input data wherever feasible, favoring measured quantities over derived signals. Second, given the resulting data, Tycho aims to extract the maximum amount of interpretable information without introducing unexamined assumptions or implicit smoothing. Throughout the system, design choices reflect a preference for accounting-based reasoning over predictive reconstruction.

As a consequence, Tycho cannot compensate for missing or incorrect telemetry. Rather than masking data limitations through interpolation or heuristic completion, the system exposes their effects directly in the attribution results. This positioning emphasizes transparency and bounded interpretation over completeness.

4.6.2 Interpretability Versus Smoothing and Convenience

Tycho explicitly favors interpretability over convenience-oriented signal processing. Attribution outputs are constructed to preserve semantic meaning, even when this results in residual categories, conservative bounds, or non-smooth time series. Such artifacts are treated as informative rather than undesirable, as they reflect the structure and limitations of the underlying data.

This design choice stands in contrast to approaches that prioritize visually continuous or intuitively smooth outputs. In Tycho, smoothing and aggregation are applied only where they can be justified in terms of attribution semantics, and never as a default mechanism for improving presentation quality. The intent is to ensure that observed structure in the output corresponds directly to observable structure in the system.

4.6.3 Research Instrumentation Versus Operational Monitoring

Tycho is designed primarily as a research instrument. Its architecture prioritizes traceability, explicit semantics, and controllable assumptions over ease of deployment or minimal operational overhead. This positioning reflects the system's intended use in experimental analysis, validation of measurement models, and investigation of workload energy behavior.

As a result, Tycho makes design choices that would be atypical for purely operational monitoring systems, including explicit handling of residual energy and conservative treatment of uncertainty. These choices are evaluated in this chapter in terms of their analytical consequences rather than their suitability for production environments.

4.6.4 Explicit Semantics and Assumption Visibility

Across its components, Tycho emphasizes explicit semantic definitions over implicit assumptions. Attribution rules, timing relationships, and aggregation decisions are encoded directly in the analysis logic rather than embedded implicitly in processing pipelines. This approach allows assumptions to be examined, challenged, and revised as part of the evaluation process.

By making these semantics explicit, Tycho supports critical interpretation of its outputs and avoids conflating measurement uncertainty with modeling convenience. This property is central to its role as an evaluative and explanatory system rather than a black-box estimator.

4.7 Scientific and Technical Contributions

This thesis contributes Tycho, an accuracy-first system for container-level energy measurement and attribution in Kubernetes environments. Tycho is designed to prioritize measurement fidelity, semantic transparency, and conservative accounting over simplicity and convenience, explicitly accepting the complexity required to extract higher-quality information from heterogeneous and imperfect telemetry. The system ingests data from multiple hardware and software sources, including eBPF-based utilization metrics, CPU energy counters, GPU power and energy telemetry, and system-level energy interfaces, as well as operating system and Kubernetes metadata.

These inputs are transformed into a coherent set of energy and utilization metrics that describe system-level and container-level behavior across major server components. Tycho produces component-resolved energy metrics for CPU, memory, GPU, and system domains, with explicit separation into idle and dynamic contributions, and exposes these measurements through a Prometheus-compatible interface. Energy not attributable to explicitly modeled components is retained as a residual term, enabling closed accounting and transparent interpretation under incomplete observability.

To support meaningful attribution in multitasking environments, Tycho emphasizes temporal fidelity and adaptive interpretation of telemetry. The system operates on fine-grained measurement data, accommodates heterogeneous timing and latency

characteristics, and adjusts internal modeling parameters based on observed behavior rather than fixed assumptions. As a result, Tycho enables component- level energy attribution for Kubernetes workloads with explicit confidence boundaries that reflect the quality and limitations of the underlying data.

The remainder of this section enumerates the specific scientific and technical contributions that jointly realize this system.

1. **Historically buffered, high-frequency metric retention for workload-level energy attribution.** Tycho introduces the retention of fine-grained, high-frequency raw measurement data in bounded historical buffers rather than relying solely on start–end deltas over fixed analysis windows. Metrics are collected at source-appropriate intervals (tens to hundreds of milliseconds) and retained beyond the duration of a single analysis window, yielding a temporally rich measurement history. This design enables post hoc alignment, correlation, and re-aggregation of heterogeneous energy and utilization signals, and allows short-lived or sequential workloads within the same window to be distinguished based on their energy characteristics. By providing substantially more informational context to attribution and fusion models, historical buffering forms the basis for finer-grained and more discriminative workload energy analysis than window-delta-based approaches.

2. **Independent, asynchronously timed metric collectors with source-specific polling frequencies.** Tycho introduces a collection architecture in which each telemetry source is polled independently at a frequency suited to its measurement characteristics, rather than enforcing a global sampling clock. Quasi-instantaneous metrics such as RAPL and eBPF are sampled at high frequency, while sources with internal update mechanisms, such as NVML and Redfish, are polled in a manner that maximizes information content while minimizing redundant queries. This design yields independent discrete time series per metric source, preserving their native temporal structure and avoiding artificial synchronization. By extracting the highest feasible granularity from each source and deferring temporal reconciliation to the analysis stage, this approach enables richer cross-metric correlation and places analytical interpretation, rather than collection-time alignment, at the center of the attribution process.

3. **Explicit modeling of metric delay in temporal alignment and attribution.** Tycho introduces explicit consideration of metric-specific observation delay into the temporal alignment and analysis model. When combining telemetry sources that react to workload changes at different speeds, ignoring such delays leads to misalignment and inconsistent interpretation. Tycho addresses this by incorporating delay directly into timeline comparison and attribution logic, aligning metrics based on inferred real-world behavior rather than raw observation time. The system further supports automatic delay calibration for selected sources and accounts for both stable and variable delay characteristics, enabling robust alignment even when telemetry latency is non-constant.

4. **Phase-aligned GPU metric polling for improved temporal accuracy at low overhead.** Tycho introduces a phase-aligned polling strategy for GPU telemetry that accounts for the internal publish cycles of GPU metrics. Because GPU power and energy metrics are generated according to device-internal timing,

naïve polling can retrieve values that are already stale, while hyperpolling incurs significant overhead. Tycho estimates the publication phase of GPU metrics and aligns polling operations accordingly, ensuring that samples are collected close to their actual generation time. This self-healing alignment strategy improves temporal accuracy of GPU energy observations without incurring the cost and energy overhead associated with aggressive polling.

5. **Composite GPU energy modeling from heterogeneous NVML power and energy metrics.** Tycho introduces a composite GPU energy modeling approach that combines multiple NVML telemetry signals describing power and energy consumption, including instant power samples, averaged power, and cumulative energy counters. Rather than relying on a single coarse-grained metric, each signal is interpreted according to its native semantics and integrated into a unified energy model. By fusing these complementary measurements, Tycho constructs a temporally finer-grained GPU energy representation than any individual raw metric provides. This higher-resolution model aligns naturally with other high-frequency system metrics and enables more accurate workload-level GPU energy attribution.

6. **System-level energy refinement through fine-grained multi-source fusion of Redfish telemetry.** Tycho introduces a modeling approach that refines coarse system-level Redfish energy telemetry using higher-frequency subsystem measurements, including eBPF utilization, RAPL energy, and GPU power data. While Redfish metrics provide a ground-truth aggregate at low temporal resolution and with variable delay, Tycho treats fine-grained subsystem signals as proxies that capture how system energy responds to workload activity. By leveraging retained historical measurement data, the model fills temporal gaps between Redfish samples and continuously adjusts its parameters based on observed behavior, allowing it to adapt to changing system conditions and configurations. The resulting model accommodates variable telemetry delay and produces a temporally finer system energy representation suitable for workload-level attribution.

4.8 Chapter Summary

This chapter evaluated Tycho as an accuracy-first energy measurement system by synthesizing experimental observations and qualitative validation. The research questions were answered explicitly, clarifying the guarantees, interpretability, and applicability of Tycho's approach. The following chapter concludes the thesis and outlines directions for future work.

Chapter 5

Conclusion and Perspectives

5.1 Conclusion

This thesis set out to investigate whether workload-level energy consumption can be measured and attributed accurately in modern Kubernetes environments, despite heterogeneous hardware, delayed telemetry, and limited observability. Rather than pursuing estimation or prediction, the work adopted an accuracy-first perspective, prioritizing conservative accounting, interpretability, and explicit treatment of uncertainty.

Through the design and implementation of Tycho, this thesis demonstrated that such an approach is not only conceptually viable, but practically realizable. While accurate energy measurement in complex systems remains fundamentally constrained by external data sources, the results show that principled system design can significantly improve the coherence and explanatory power of workload-level energy observations.

5.2 Summary of Contributions

The primary contribution of this thesis is the development and evaluation of Tycho, an accuracy-first energy measurement system designed explicitly as a research instrument for Kubernetes environments. Beyond the concrete system itself, the work advances a methodological stance on energy measurement that emphasizes explicit temporal modeling, conservative attribution, and transparency of residual uncertainty.

By unifying heterogeneous telemetry sources under a coherent timing model and treating attribution as a constrained accounting problem rather than an estimation task, this thesis demonstrates that higher-fidelity and more interpretable workload-level energy measurements are achievable in practice. The implemented system shows that these ideas are not merely theoretical, but can be realized in a functioning measurement pipeline and subjected to systematic experimental evaluation.

5.3 Perspectives and Future Work

Several directions for future work emerge naturally from this thesis.

Improved and Alternative Telemetry Sources

Automated Calibration and Adaptation

Scaling and Broader Experimental Studies

Integration into Energy-Aware Research Workflows

5.4 Final Remarks

Tycho is not intended as a universal solution for production energy monitoring. Its complexity and reliance on detailed telemetry make it most suitable as a research instrument rather than an operational tool. However, within this intended scope, the work presented in this thesis argues that accuracy, interpretability, and explicit treatment of uncertainty are not optional luxuries, but essential properties for meaningful energy analysis.

By demonstrating that accuracy-first measurement is both achievable and informative, this thesis contributes to a more principled understanding of energy behavior in complex systems and provides a foundation for future research that treats energy as a first-class, measurable system property rather than a derived estimate.

Bibliography

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

[2] 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).

[3] 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).

[4] 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).

[5] Charles Reiss et al. "Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis". In: *Proceedings of the Third ACM Symposium on Cloud Computing*. SoCC '12. New York, NY, USA: Association for Computing Machinery, Oct. 2012, pp. 1–13. ISBN: 978-1-4503-1761-0. DOI: 10.1145/2391229.2391236. (Visited on 11/26/2025).

[6] Muhammad Waseem et al. *Containerization in Multi-Cloud Environment: Roles, Strategies, Challenges, and Solutions for Effective Implementation*. July 2025. DOI: 10.48550/arXiv.2403.12980. arXiv: 2403.12980 [cs]. (Visited on 11/26/2025).

[7] Emiliano Casalicchio and Stefano Iannucci. "The State-of-the-Art in Container Technologies: Application, Orchestration and Security". In: *Concurrency and Computation: Practice and Experience* 32.17 (2020), e5668. ISSN: 1532-0634. DOI: 10.1002/cpe.5668. (Visited on 11/26/2025).

[8] 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).

[9] 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).

[10] 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.

[11] 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).

[12] 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).

[13] Magnus Herrlin. "Accessing Onboard Server Sensors for Energy Efficiency in Data Centers". In: (Sept. 2021). (Visited on 11/27/2025).

[14] Ghazanfar Ali et al. "Redfish-Nagios: A Scalable Out-of-Band Data Center Monitoring Framework Based on Redfish Telemetry Model". In: *Fifth International Workshop on Systems and Network Telemetry and Analytics*. Minneapolis MN USA: ACM, June 2022, pp. 3–11. ISBN: 978-1-4503-9315-7. DOI: 10.1145/3526064.3534108. (Visited on 11/27/2025).

[15] 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.

[16] 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).

[17] 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).

[18] 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).

[19] 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).

[20] 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).

[21] 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).

[22] 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).

[23] 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).

[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] 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).

[26] 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.

[27] 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).

[28] Oscar Hernandez et al. "Preliminary Study on Fine-Grained Power and Energy Measurements on Grace Hopper GH200 with Open-Source Performance Tools". In: *Proceedings of the 2025 International Conference on High Performance Computing in Asia-Pacific Region Workshops*. Hsinchu Taiwan: ACM, Feb. 2025, pp. 11–22. ISBN: 979-8-4007-1342-2. DOI: 10.1145/3703001.3724383. (Visited on 11/27/2025).

[29] Le Mai Weakley et al. "Monitoring and Characterizing GPU Usage". In: *Concurrency and Computation: Practice and Experience* 37.3 (2025), e8341. ISSN: 1532-0634. DOI: 10.1002/cpe.8341. (Visited on 11/27/2025).

[30] The Linux Kernel Community. *The proc Filesystem*. https://www.kernel.org/doc/html/latest/filesystems/proc.html. Accessed: 2025-06-17. 2025.

[31] The Linux Kernel Community. *Control Group v1 — Linux Kernel Documentation*. https://www.kernel.org/doc/html/latest/admin-guide/cgroup-v1/index.html. Accessed: 2025-06-17. 2025.

[32] The Linux Kernel Community. *Control Group v2 — Linux Kernel Documentation*. https://www.kernel.org/doc/html/latest/admin-guide/cgroup-v2.html. Accessed: 2025-06-17. 2025.

[33] Cilium Authors. *eBPF and XDP Reference Guide*. https://docs.cilium.io/en/latest/reference-guides/bpf/index.html. Accessed: 2025-06-17. 2025.

[34] Cyril Cassagnes et al. "The Rise of eBPF for Non-Intrusive Performance Monitoring". In: *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. Apr. 2020, pp. 1–7. DOI: 10.1109/NOMS47738.2020.9110434. (Visited on 06/14/2025).

[35] Brendan Gregg. *CPU Utilization is Wrong*. Blog post. Accessed 29 June 2025. May 2017. URL: https://www.brendangregg.com/blog/2017-05-09/cpu-utilization-is-wrong.html.

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

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

[38] 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).

[39] 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).

[40] 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).

[41] 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).

[42] 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).

[43] 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).

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

[45] 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).

[46] 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).

[47] 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).

[48] 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).

[49] Inc. Meta Platforms. *Kepler v0.9.0 (pre-rewrite): Kubernetes-based power and energy estimation framework*. Accessed: 2025-04-28. 2023. URL: https://https://github.com/sustainable-computing-io/kepler/releases/tag/v0.9.0.

[50] Bjorn Pijnacker. "Estimating Container-level Power Usage in Kubernetes". MA thesis. University of Groningen, Nov. 2024. (Visited on 03/17/2025).

[51] Linux Foundation Energy and Performance Working Group. *Kepler: Kubernetes-based Power and Energy Estimation Framework*. Accessed: 2025-11-14. 2025. URL: https://github.com/sustainable-computing-io/kepler.

[52] Bjorn Pijnacker, Brian Setz, and Vasilios Andrikopoulos. *Container-Level Energy Observability in Kubernetes Clusters*. Apr. 2025. DOI: 10.48550/arXiv.2504.10702. arXiv: 2504.10702 [cs]. (Visited on 07/02/2025).

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

[54] Guillaume Fieni, Romain Rouvoy, and Lionel Seinturier. "SmartWatts: Self-Calibrating Software-Defined Power Meter for Containers". In: *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. May 2020, pp. 479–488. DOI: 10.1109/CCGrid49817.2020.00-45. (Visited on 05/21/2025).

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