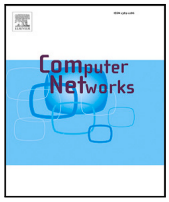
Computer Networks 245 (2024) 110371

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Impact of power consumption in containerized clouds: A comprehensive analysis of open-source power measurement tools

Carlo Centofanti [a,*], José Santos [b], Venkateswarlu Gudepu [c], Koteswararao Kondepu [c]

[a] Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, via Vetoio, L'Aquila, 67100, Italy
[b] Ghent University - imec, IDLab, Department of Information Technology, Technologiepark - Zwijnaarde 126, Gent, 9052, Belgium
[c] Indian Institute of Technology Dharwad, CSE Department, Dharwad, India

ARTICLE INFO

Keywords:
Energy efficiency
Power consumption
Sustainability
Containers
Cloud computing
Service orchestration
Kubernetes

ABSTRACT

Recently, container-based solutions have become *de facto* compute units of modern cloud-native applications. However, the exponential growth in data traffic and the power consumption of these technologies to handle high data traffic alarm the strong need for energy evaluation approaches in containerized clouds. Furthermore, the proliferation of highly distributed edge clouds raises additional concerns regarding the power consumption of future cloud architectures. This article presents a detailed overview of methods and techniques for monitoring power consumption within popular cloud platforms. The study offers an in-depth evaluation of these approaches, demonstrating variations in measured power consumption based on the chosen technique. A well-known container orchestration platform named Kubernetes (K8s) has been applied in our extensive measurements. This work argues that energy-efficient container clouds will play a vital role in building a more sustainable and eco-friendly digital infrastructure by optimizing power consumption and reducing carbon footprint, paving the way for a greener future. The paper also discusses open challenges and future research directions on energy sustainability, leading to the conclusion, offering lessons learned and prospects on potential solutions to foster sustainable practices within the container ecosystem.

1. Introduction

In the recent landscape of cloud computing, the emergence and popularity of containers has significantly revolutionized application development and deployment [1]. Containers provide a flexible and portable manner to package (due to lightweight), distribute, and manage software, enabling seamless integration and scalability. Nonetheless, this evolution meant a significant increase in power consumption within Data Centers (DCs), currently representing 2% of the world's energy consumption, and recent forecasts expect it to grow even further to at least 8% by 2030 [2]. The power consumed in cloud computing is fast outpacing the global energy production due to several factors: explosion in data collection, computational-intensive Artificial Intelligence (AI)-based workloads, and the flattening of Moore's law [3]. The ecological ramifications and long-term sustainability of this heightened energy demand within cloud infrastructures have emerged as pressing concerns, urging a reevaluation of current cloud strategies to power consumption in the digital age.

In addition, with the rapid development of the Internet of Things (IoT) [4], the number of connected smart devices is growing exponentially, resulting in enormous amounts of data, which caused problems regarding bandwidth and latency in traditional cloud deployments. Distributed cloud paradigms such as Fog Computing [5] and Edge Computing [6] emerged as potential solutions, but power consumption related concerns also emerged in these highly distributed cloud environments. These smart devices are significantly dispersed in the network area, demanding substantial power to process and transmit data locally. Also, the need for continuous connectivity and low latency communications further intensifies power usage, posing a hurdle in achieving a sustainable and energy-efficient edge computing ecosystem. Researchers and Industries need to focus on reducing the environmental footprints of increasingly large DCs and distributed edge clouds addressing the complete cloud continuum [7].

The numerous advantages of cloud computing environments, including cost effectiveness, on-demand scalability, and ease of management, encourage service providers to adopt them and offer solutions via cloud models. In turn it facilitates platform providers to increase the underlying capacity of their DCs to accommodate the increasing demand of new customers. In addition to the traditional cloud services such as Infrastructure as a Service (IaaS), Platform as a Service

* Corresponding author.
E-mail address: carlo.centofanti1@univaq.it (C. Centofanti).
[1] Google Cloud: https://cloud.google.com/container-engine/

https://doi.org/10.1016/j.comnet.2024.110371
Received 31 October 2023; Received in revised form 8 March 2024; Accepted 25 March 2024
Available online 27 March 2024

(PaaS), and Software as a Service (SaaS), the leading cloud providers like Google.[1] and Amazon Web Services (AWS) have introduced the novel concept of Containers as a Service (CaaS) This cloud model is reshaping the cloud computing landscape thanks to its lightweight architecture, simplified configuration and management, and notably reduced startup times. CaaS represents a fundamental shift in how applications are designed, built, and deployed. With the containerization of applications and their dependencies, developers now focus on creating modular, self-contained components, commonly known as microservices. These microservices can run consistently across various environments, eliminating the age-old "It works on my machine" problem and simplifying the development process. CaaS platforms provide streamlined workflows for building, testing, and deploying containers, enabling faster development cycles and seamless integration with Continuous Integration/Continuous Deployment (CI/CD) pipelines.

As CaaS continues to gain prominence as a cloud service model, the efficient management of energy resources within DCs becomes increasingly crucial. As a part of power management, monitoring power consumption contributes to cost control and resource optimization and aligns with environmental sustainability goals. It enables organizations to track and reduce carbon emissions, adhere to regulatory requirements, and implement practices that promote energy efficiency. Additionally, power monitoring provides insights into capacity planning, performance optimization, and early detection of issues, facilitating both operational excellence and the achievement of long-term sustainability objectives in the dynamic landscape of cloud-native computing. Several container orchestration platforms have been designed in the last few years, including K8s,[2] Docker swarm,[3] CRI-O,[4] Apache Mesos,[5] Microsoft Azure,[6] Portainer.[7] and Rancher[8] From these, K8s has emerged as the reference platform for the orchestration of containers, allowing automatic management features for containers [8].

Numerous works are focusing on power consumption optimization using a wide range of methodologies, including traditional optimization models and heuristic algorithms [9–16], hardware and software-based solutions [17–19], Genetic Algorithms (GAs) [20,21], and Machine Learning (ML)-based algorithms [22,23]. However, it is worth noting that comprehensive investigations into various open-source tools for power measurements are relatively scarce in the current state-of-the-art. This article presents insights into various open source software tools: *(i) Stress-Terminal UI (s-tui)*; *(ii) Kubernetes-based Efficient Power Level Exporter (Kepler)*; and *(iii) Scaphandre* alongside the Meross MSS310 [24] hardware equipment. By offering precise power consumption measurements for each cluster node and respective containers, these tools empower cloud providers to strategically deploy their applications, ultimately leading to more energy-efficient operations.

This work aims to present valuable insights focused on recent methodologies for monitoring power consumption in container clouds that could lead to further research efforts, seeking to mitigate the environmental impact of cloud computing in a sustainable digital era. The main contributions of the paper are threefold:

- **Towards an energy-efficient cloud infrastructure**: Section 3 details the proposed K8s-based architecture for a more energy-efficient container cloud infrastructure. Several monitoring tools are applied to provide precise power measurements, helping to achieve more energy-efficient management.
- **Comprehensive evaluation of power measurement tools**: The evaluation consisted of various methodologies, employing both hardware and software-based approaches, to determine the differences in monitoring power consumption. Results reveal diverse measurements obtained through these methodologies (Section 5).
- **Open Challenges and promising Research Directions**: This paper details open challenges and proposes future research directions on optimizing power consumption in containerized clouds (Section 6). Also, Section 7.2 presents lessons and future prospects focused on novel trends in the cloud computing field. Carbon-neutral container clouds would contribute significantly to more environmentally conscious and sustainable future.

The remainder of the paper is organized as follows: Section 2 presents the current state-of-the-art focused on energy-efficient strategies for cloud environments. Section 3 details the most promising approaches for accurate power consumption monitoring in the popular K8s platform. Then, Section 4 presents the testbed applied to evaluate the power measurement strategies. Section 6 focuses on open challenges and future directions, while Section 7 presents the lessons learned and discusses the prospects of energy-efficient container clouds focused on emerging trends. Concluding remarks are presented in Section 8.

2. Related work

Optimizing power consumption while ensuring high performance is crucial with the increasing demand for CaaS services. The current literature on power measurement and efficiency in cloud computing involves diverse platforms, methodologies, and tools, each presenting distinct insights and limitations, aiming to improve resource utilization while minimizing power consumption (Table 1).

Theoretical formulations and heuristic methods are mostly used to solve resource allocation focused on power consumption [9–16]. These include linear formulations for workload consolidation and theoretical frameworks using metrics such as Carbon Usage Effectiveness (CUE), and Water Usage Effectiveness (WUE) [10] to assess efficiency. These studies focus mainly on Integer Linear Programming (ILP) and Mixed Integer Linear Programming (MILP) models to find the optimal resource allocation based on a particular goal. However, a significant drawback of these modeling approaches is their inability to attain a feasible solution within a reasonable timeframe. Note that these models can serve as an optimal benchmark for heuristic-based algorithms by offering an optimal reference point for energy-efficient resource allocation strategies.

Hardware and software-based approaches enable precise power measurement and monitoring at different levels of the technology stack [17–19]. At the hardware level, sophisticated power measurement tools and instrumentation provide granular insights into the power usage of individual components, allowing for a detailed analysis of energy patterns and consumption across diverse hardware elements. At the software level, innovative algorithms and tools facilitate power profiling and monitoring, including detailed measurements of the energy footprint of applications, and processes. Moreover, DC-wide monitoring systems such as Dynamo [15] represent a concerted effort to comprehensively monitor and manage power consumption across an entire DC. These systems leverage advanced analytics and real-time data processing to optimize resource allocation and reduce overall carbon emissions, aligning with the global initiative for sustainable computing infrastructures. However, these approaches also exhibit limitations, from potential sub-optimal solutions to focus on specific levels of power measurement or a lack of real-time monitoring capabilities.

Genetic Algorithms (GAs) are optimization techniques inspired by the process of natural selection and evolutionary biology, widely used for solving optimization and search problems, including resource allocation challenges [20,21]. These algorithms improve the solution

[2] https://kubernetes.io/
[3] https://docs.docker.com/engine/swarm/
[4] https://cri-o.io/
[5] https://mesos.apache.org/
[6] https://azure.microsoft.com/en-in
[7] https://www.portainer.io/
[8] https://www.rancher.com/

Table 1
State-of-the-art of the energy efficiency in cloud computing models.

Work	Plat.	Meth.	Eval.	Methodology	Limitations
Piraghaj et al. [9]	C	T	S	Optimizes container placement for minimal server usage.	Dynamic workloads in DCs may lead to sub-optimal solutions.
Van De Voort et al. [10]	DC	T	S	Introduces theoretical calculations for CUE and WUE evaluation in DC power consumption, assessing carbon emissions and water usage.	Concentrates on specific component-level power measurements.
Vijouyeh et al. [11]	C	T & H	S	Introduces a theoretical formulation for application deployment in Fog computing, considering user delay requirements and traffic processing constraints.	Evaluation consisted of generic application scenarios.
Santos et al. [12]	C	T	S	Proposes a MILP formulation addressing the service allocation problem in Fog Computing that considers multiple objectives.	Focus mainly on reducing deployment costs as energy savings. The MILP model scales badly.
Zorello et al. [13]	VMs & DC	T & H	S	MILP formulation that minimizes the DU/CU placement power consumption.	The power consumption assumes only theoretical models.
Piontek et al. [14]	C	H	K8s	CO_2-aware workload scheduling algorithm implemented in K8s to shift non-critical jobs in time and reduce carbon emissions.	Lacks real-time energy measurements and monitoring capabilities.
Wu et al. [15]	DC	H	D	Monitors the complete power hierarchy in a DC and makes coordinated control decisions to ensure safe and efficient utilization of provisioned power.	Provides power consumption data only at the infrastructure level.
Okhovvat et al. [16]	WSAN	H	S	Assign tasks to WSAN nodes following a joint optimization considering energy and execution time.	The emphasis is on optimizing process execution time in WSAN nodes while not prioritizing battery saving.
Bellin et al. [17]	VMs & C & BM	HW	M	Measuring 5G Core Network (5G CN) power consumption using hardware equipment to scale computational resources for different 5G CN Virtual Network Functions (VNFs).	Concerns include connectivity, privacy, security, and compatibility related to the use of Wi-Fi-based Meross MSS3105
Lando et al. [18]	C	SW	5G RAN	Evaluated various open-source 5G CNs such as free5GC, open5GS, and OpenAirInteraface (OAI) for their power consumption during user registration and authentication.	Does not account for power consumption by the 5G CN during the user service period.
Schmitt et al. [19]	C	SW	K	Introduces a monitoring and modeling approach to estimate power consumption for cloud applications.	Focuses exclusively on application-layer power measurement, potentially overlooking lower-level power consumption.
Saxena and Singh [20]	VMs	GA	S	Introduces a GA for server consolidation, optimizing resource usage and minimizing power consumption within a DC.	GAs can yield sub-optimal solutions within the search space. Power measurement is limited to the infrastructure.
Ilbeigi et al. [21]	OF	GA & ML	S	Proposes a GA method to optimize the power consumption of buildings. Results show that the number of occupants has the greatest influence on power consumption.	The main validation of the approach was based on simulation results.
Ma and Ding [22]	VMs & DC	ML	S	Introduces an Reinforcement Learning (RL)-based approach for power consumption evaluation in cloud computing.	It relies entirely on simulation results.
Shapi et al. [23]	DC	ML	C	The authors propose a predictive ML-based model for energy consumption in cloud.	The proposed algorithm is quite time-consuming.

Platform (Plat.): VMs = Virtual Machines, C = Containers, BM = Bare Metal, DC = Data Center infrastructure, OF = Office Building, WSAN = Wireless Sensor and Actuator Network.
Method (Meth.): T = Theoretical assumptions, H = Heuristic algorithm, GA = Genetic Algorithm, HW = Hardware-based equipment, SW = Software-based tool, ML = Machine Learning.
Evaluation (Eval.): C = Cloud-based testbed, S = Simulations, M = Meross MSS3105, 5GRAN = my5GRAN Tester, D = Dynamo, K8s = Kubernetes, K = Kieker[9].

in each generation iteratively, converging towards an optimal or near-optimal solution for the problem at hand. These can efficiently explore large search spaces while accommodating multiple objectives. However, this comes at a high computational cost since evaluating each individual in a population can be time-consuming, making GAs impractical for real-time applications or situations requiring fast responses.

ML-based solutions [22,23] have gained considerable traction, representing a pivotal approach to optimize energy efficiency. These methods aim to build a model for resource estimation under a specific workload. One of the standout features of these algorithms is their inherent robustness in handling fluctuating and unpredictable demands, often encountered in modern computing environments. These

[9] http://kieker-monitoring.net/

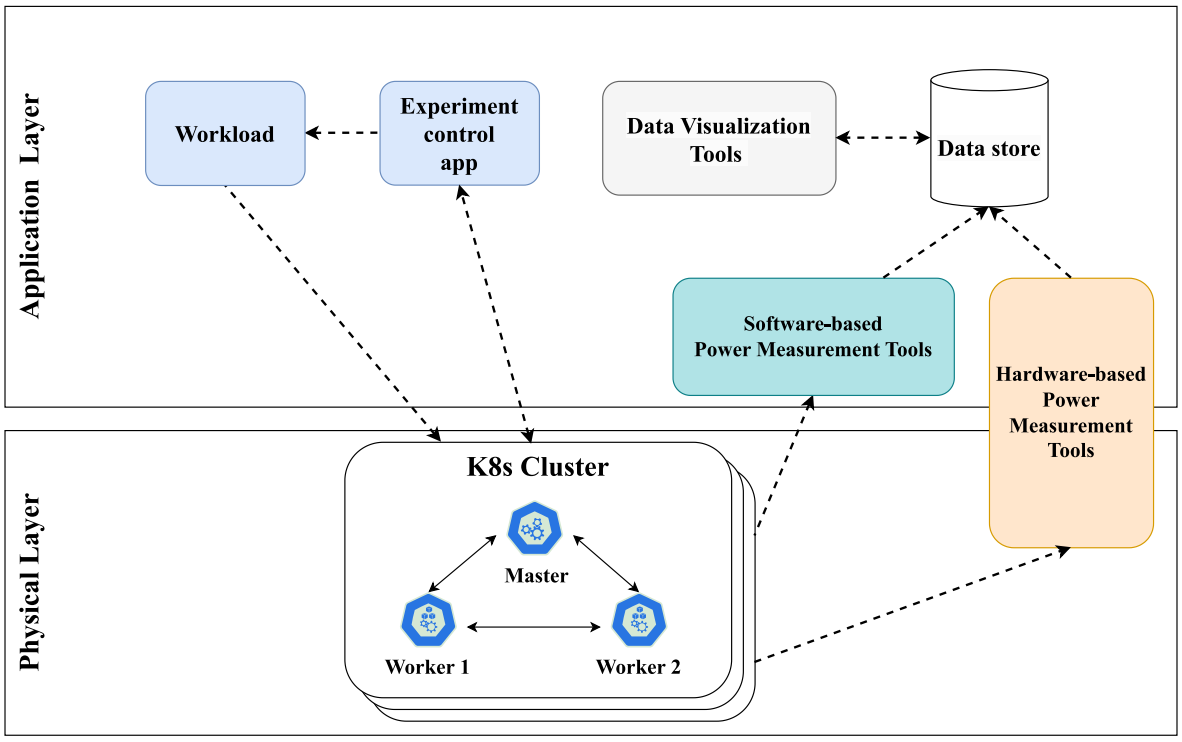


Fig. 1. High-level view of the Energy-efficient K8s-based infrastructure.

algorithms can adjust their model parameters in real-time whenever a notable event occurs, ensuring high degrees of adaptability (*i.e.*, online learning). In contrast, offline training may demand substantial manual intervention to fine-tune the model and achieve optimal accuracy. The main drawback of ML-based approaches is the high execution time to converge to a stable model and thus causes allocation and management assignments to perform poorly during the learning period.

In summary, this work aims to evaluate various power measurement tools and hardware-based equipments, which are imperative to advance research in energy-efficient cloud computing. This work provides valuable insights to researchers and practitioners aiming to provide detailed analysis of power measurement tools and enable more effective strategies for improving energy efficiency in DCs and cloud infrastructures. The comprehensive understanding of available tools facilitates resource allocation, benchmarking efforts, and the development of advanced energy-efficient computing solutions. As cloud computing evolves, the need for sustainable practices becomes increasingly prominent. Optimizing power consumption aligns with current environmental goals aiming to reduce energy costs and enhance scalability of cloud services. Addressing the challenges and limitations in existing energy-efficient models and tools is pivotal for realizing a future where cloud services are both high-performing and eco-friendly. Our study provides a crucial foundation for enhancing the sustainability and performance of modern CaaS based services.

3. Towards energy sustainability in container clouds

This section presents the proposed architecture towards energy sustainability in cloud computing focused on containerization. The K8s-based infrastructure and the importance of monitoring are discussed, alongside the detailed description of several monitoring tools available as open-source.

3.1. Kubernetes (K8s)-based infrastructure - architecture overview

The advent of novel architectural paradigms enables the deployment of different applications on computational resources from the cloud up to the edge, providing several benefits such as low latency and mobility support. An application has been decomposed into sets of loosely coupled services that can be developed, deployed, and maintained independently. Each service is typically responsible for a single task and communicates with the other services through lightweight protocols. Containers are currently the most promising alternative to the traditional monolithic application paradigm, where almost everything is centralized and code-heavy. Due to their low resource usage and high portability, containers are also the main alternative to conventional Virtual Machines (VMs). With their massive adoption, several orchestration solutions for containerized applications have been designed in recent years by IT companies and researchers, K8s being the most widely used today. K8s is highly customizable and provides numerous hooks for extending its behavior by implementing new functionalities, as demonstrated in recent research [25]. Fig. 1 presents a high-level view of the proposed energy-efficient cloud computing infrastructure, showing its main architectural concepts based on the K8s platform. K8s follows the master-worker model, where at least one master node manages containers across multiple worker nodes. Kubernetes already provides several components (*e.g.*, API server, Kubelet, Controller Manager) to handle the complete life cycle workflow of containerized applications. Containers in K8s are deployed via a *pod*, which is the smallest deployable workload unit in K8s. Each *pod* is capable of hosting one or more containers running within the same execution environment [8].

Fig. 1 also shows our monitoring architecture depicting the *physical layer*, the *monitoring layer*, and the *application layer*. The proposed architectural solution employs different monitoring tools designed specifically for power consumption measurement in response to the pressing issues of high power consumption in cloud platforms. The aim is to analyze the current state of the art for precise and real-time measurements from the K8s-based infrastructure, thereby providing cloud administrators with the necessary insights to implement sustainable practices in the orchestration and management of energy sustainable clusters. Here, power measurement tools are required in order to measure energy consumption. These measurement tools will play a major

role in enabling observability and optimization of power consumption across numerous cloud environments. These tools will allow measuring the power consumption of different cluster nodes as well as distinct applications deployed through *pod* instances. This granular level of information is instrumental in enabling making informed decisions about service placement, and scaling operations within the cloud infrastructure based on the current power consumption. By providing comprehensive power consumption data, these tools can empower cloud administrators to explore new research directions for energy efficiency. This valuable information will be key for implementing novel energy-efficient scheduling and auto-scaling practices. These practices will not only enhance the overall performance and resource allocation within the cloud but also significantly can help saving operational costs. By leveraging these insights, cloud administrators can proactively steer their cloud deployments towards more energy-efficient and eco-friendly management. The problem we address in this work is an analysis of the current open-source power measurement tools to determine their maturity level and the reliability of these tools in effectively measuring the energy consumption of cloud systems.

3.2. Server power consumption modeling

Power consumption modeling involves creating mathematical or computational models to estimate and analyze the power consumption of servers in different scenarios. These models are essential to optimize power consumption in DCs and ensuring sustainable computing practices. Literature often considers multiple factors to develop accurate models, such as hardware components, cooling systems, and workload patterns [26,27]. By simulating different conditions, experts can identify energy-efficient configurations and strategies, leading to reduced environmental impact and cost savings. The main components affecting power consumption of a server are the following:

- *CPU* significantly contributes to power consumption. More powerful CPUs typically consume additional power, especially when operating at higher clock speeds.
- *Memory (RAM)* modules consume power, especially during read/write operations. Larger memory configurations or high-performance RAM modules can increase power usage.
- *Storage* (*e.g.*, hard drives) consumes varying amounts of power. Solid State Drives (SSDs) are generally more energy-efficient than traditional hard drives.
- *Other Components* such as motherboard, fans, network interface cards, and other peripherals contribute to the overall power consumption of the server.

The models can cover physical machines and VMs and be classified according to the applied calculation formula. CPU typically is the largest contributor to power consumption, followed by peripheral slots [28]. The most used formula is to add the power contributions of all components Eq. (1) and each component has its own power consumption model. For example, the power consumption of the CPU is a function of *voltage*, *frequency*, and *utilization*, while the power consumption of the memory is a function of its *size*, *frequency*, and *usage*. In addition, in DCs, servers can be idle since these are not always active. Then, the power consumption formula can be divided into two parts: baseline and active power. Baseline power corresponds to the power consumption when the machine is idle, including the power consumption of the fans, CPU, memory, I/O and other motherboard components in their idle state, often considered as a fixed value. Active power corresponds to the power consumption under certain workloads, depending on how the machine utilizes CPU, memory and I/O components.

$$P_{total} = P_{CPU} + P_{RAM} + P_{Storage} + \cdots + P_{Fans} + P_{Motherboard} \quad (1)$$

3.3. CPU power consumption metrics

Accurately estimating and managing CPU power consumption stands as a significant challenge, primarily due to the ambiguity surrounding the power usage data provided by most CPU manufacturers. Despite obstacles, an important metric named Thermal Design Power (TDP) offers insights into the power consumption behaviors of CPUs under specific conditions. TDP is traditionally comprehended as the maximum amount of heat generated by a CPU that the cooling system in a computer is required to dissipate under any workload. While not a direct measurement of power consumption, TDP approximates the average of the maximum power a CPU is expected to consume when operating at base frequency under high computational loads for a reasonable amount of time [25]. TDP, indicative of the worst-case scenario for a specific CPU workload capacity, serves as a pivotal reference point for understanding the potential energy footprint of DC and cloud environments. Measuring the average power consumption of a CPU over an extended period is typically approximated to the TDP value. However, it is crucial to note that TDP alone cannot provide a comprehensive view of a CPUs power consumption dynamics. The instantaneous CPU power usage can significantly vary depending on the workload, operating frequency, and its specific architecture. As such, relying solely on TDP values may lead to oversimplified assumptions about the energy efficiency of cloud infrastructures, given that it is uncommon for all CPUs to operate at 100%.

The Running Average Power Limit (RAPL) interface emerges as a critical tool in the estimation and management of CPU power consumption. Introduced with Intel's Sandy Bridge architecture, RAPL provides a sophisticated means for software platforms to monitor, control, and receive notifications regarding the power consumption of the System on a Chip (SoC). RAPL facilitates this by categorizing the platform into distinct domains, including the entire CPU package, the Dynamic Random Access Memory (DRAM) controller, individual CPU cores (Power Plane 0), and the graphics uncore (Power Plane 1), among others. This fine-grained control and monitoring capability allows for a more detailed understanding of energy consumption across different components of the CPU and attached peripherals such as DRAM and on-chip Graphics Processing Unit (GPU). Over the years, Intel has augmented RAPL with additional features to broaden its applicability across various computing environments, enhancing its utility in power management. By leveraging RAPL, developers and system administrators can gain unprecedented insights into the power dynamics of CPUs, enabling more accurate predictions of energy consumption and efficiency. Unlike TDP, which offers a generalized view of power consumption under specific conditions, RAPL provides the means to track and manage power usage in real-time across multiple dimensions, presenting a comprehensive framework for understanding and mitigating the energy footprint of modern computing systems.

3.4. The importance of monitoring

Monitoring in cloud computing is vital for optimizing performance, ensuring security and compliance, managing deployment costs, and maintaining a seamless user experience. It helps cloud providers fulfill service level agreements while enhancing overall system performance and reliability. Prometheus is an advanced open-source monitoring system designed for collecting, storing, and querying time-series data and metrics widely used in industrial settings [29]. It employs a pull-based approach, periodically scraping metrics from endpoints exposed by applications, services, or infrastructure components. These metrics are then stored in Prometheus's time-series database. Also, it provides a powerful query language, *PromQL*, for real-time data analysis and the creation of custom alerts. It is highly adaptable and commonly used for monitoring cloud-native environments, enabling organizations to gain insights into the health and performance of their infrastructure.

Grafana [30], on the other hand, complements Prometheus by serving as a versatile visualization and dashboarding platform. It is designed to create interactive and customizable dashboards that display monitoring data in a user-friendly manner. It supports various data sources, including Prometheus, InfluxDB, Elasticsearch, and more, making it agnostic to the underlying monitoring system. With Grafana, users can build dynamic dashboards, set up alerting rules, and share insights across the organization. This tool improves the observability of complex systems and aids in identifying trends, anomalies, and issues through rich visualizations.

In addition, *Service Monitors* simplify the integration of Prometheus with K8s-native applications by defining how Prometheus should discover and scrape metrics from Pods, Services, and other resources in a K8s cluster. *Service Monitors* streamline the configuration process by specifying which targets to monitor, the frequency of scraping, and label mappings for efficient data querying. These resources are especially valuable in dynamic environments where pods are frequently created and terminated, as these automate the discovery and monitoring of these transient components, ensuring comprehensive observability within K8s clusters.

3.5. Power measurement tools

This section presents an overview of methods to measure energy consumption while detailing existing software and hardware-based power measurement tools such as Scaphandre[10] and Kubernetes Efficient Power Level Exporter (Kepler) [31].

3.5.1. Overview

Power monitor [32] is one of the most accurate strategy to measure energy consumption. Typically, these hardware tools connect to the power source of the device and measure the actual power leveraged at any instant of time. Despite being extremely precise, power monitors are also extremely difficult to set up. Often, these require custom changes to the devices being monitored. Energy profilers [33] are another method that typically does not require special hardware or power sensors. These have an estimation model of the power cost of the different hardware components. Based on which components are active during execution, the profiler estimates a particular energy cost. Scaphandre and Kepler use Intel RAPL [34,35], which is mainly used by almost all energy profilers for Intel devices. Intel RAPL is a hardware feature that allows to monitor energy consumption across different domains of the CPU chip, attached DRAM and on-chip GPU. This feature has been introduced in Intel Sandy Bridge architecture and has evolved in later versions of Intel processing architecture. With RAPL, it is possible to programmatically obtain real-time data on the power consumption of the CPU package and its components, as well as the DRAM memory that the CPU is managing. Intel RAPL integrated within modern CPUs, employs energy counters to measure the joules consumed by the processor and its associated memory subsystems. The granularity of these measurements, represented in energy units, allows for high-resolution monitoring over a broad range of values. Importantly, the conversion rate of these energy units to microjoules can vary between different generations of CPUs, highlighting the need to take it into account when comparing power consumption across various hardware platforms. Intel RAPL provides two main functionalities: (i) measure energy consumption at a fine granularity and a high sampling rate, and (ii) limiting (or capping) the average power consumption of different components inside the processor, limiting also the processors thermal output. Kepler uses the energy consumption measurement capability. Intel RAPL technology provides a more nuanced approach to monitor energy consumption than relying solely on TDP values.

3.5.2. Stress terminal UI (s-tui)

s-tui[11] is an open-source software utility designed for stress-testing and monitoring system performance in a terminal or command-line interface (CLI) environment. This tool allows users to simulate and assess the system's behavior under heavy loads and stress conditions, making it valuable for testing and optimizing hardware and software configurations. *S-tui* typically provides a user-friendly interface that displays real-time information about system resource usage, including CPU utilization, temperature, and other relevant metrics. Users can configure stress tests to evaluate how their systems respond to high computational workloads, helping to identify potential bottlenecks, overheating issues, or system instability.

S-tui extracts power consumption along with some other parameters as follows: (i) *Power:* monitors the power consumption of the host system (in Watts); (ii) *Util:* tracks the utilization of each core and also provides an average of all the cores; (iii) *Frequency*: reports the CPU frequency (*i.e.*, the number of cycles a CPU performs per second.) of each core; and (iv) *Temp*: monitors the CPU temperature specific to different aspects.

3.5.3. Kubernetes efficient power level exporter (Kepler)

Kepler [31] is an open-source project started by Red Hat with early contributions from IBM Research and Intel. The aim is to collect power consumption metrics across multiple platforms to better understand power consumption in container cloud infrastructures. Kepler utilizes software counters and power sources such as RAPL, Advanced Configuration and Power Interface (ACPI) and NVIDIA GPU to measure power consumption based on hardware resources within a K8s cluster. Kepler is an extended Berkeley Packet Filter (eBPF)-based approach, which allows to attribute power consumption to specific processes, containers and even K8s pods. This granular visibility enables efficient power management and optimization. Kepler also converts power usage into energy estimates based on custom ML trained models that help in resource planning and scheduling tasks. By accessing Kepler metrics via Prometheus, cloud administrators gain valuable insights into the power consumption of individual pods and the carbon footprint of their cloud infrastructure. This visibility empowers them to identify energy-intensive workloads, optimize resource allocation and make informed decisions regarding workload placement. Kepler can retrieve meaningful power consumption metrics from K8s clusters as follows:

- *kepler_container_core_joules_total*: the total power consumption on CPU cores of the container. Typically, if the system has access to RAPL metrics, this metric corresponds to the power consumed by all CPU cores in the socket.
- *kepler_container_cpu_cycles_total*: the total CPU cycles used by the container using hardware counters, which are particularly desirable due to its granularity and precision.
- *kepler_container_cpu_instructions_total*: the total CPU instructions used by the container using hardware counters.
- *kepler_node_core_joules_total*: the total power consumption of a node by aggregating all containers running on the node and operating system.

3.5.4. Scaphandre

Scaphandre, a power monitoring tool, employs a combination of hardware-based power sensors and system-level data to measure power consumption on computing systems. These power sensors, such as Intel's RAPL technology, track energy counters associated with key hardware components like CPUs and GPUs to measure the power consumption. Simultaneously, Scaphandre monitors resource utilization metrics such as CPU and memory usage to correlate resource activity with power consumption. At the process level, Scaphandre

[10] https://hubblo-org.github.io/scaphandre-documentation/index.html

[11] https://github.com/amanusk/s-tui

tracks individual process activity, analyzing small time intervals known as *jiffies*. It calculates the power consumption of the entire system by combining data from power sensors, resource utilization, process-level activity, and jiffies, providing information on the power consumption of individual processes.

Furthermore, Scaphandre offers exporters to facilitate the export of power-related data, making it compatible with monitoring systems like Prometheus for comprehensive power analysis and environmental impact assessment. Scaphandre can retrieve meaningful power consumption metrics from VMs/containers as follows:

- *scaph_host_power_microwatts*: Aggregates measurements from various sources, including RAPL domains, to provide an estimate of host power consumption.
- *scaph_process_power_consumption_microwatts*: It collects the power consumption of a specific process on the system, indicated by labels such as *exe*, *cmdline*, *instance*, and *pid*.
- *scaph_host_energy_microjoulesl*: Provides power consumption data for the entire host in microjoules.
- *scaph_host_cpu_frequency*: Provides the global frequency of all the CPUs in MegaHertz (MHz).

Scaphandre stands out as a pioneering solution capable of discerning the power usage attributable to individual processes within a computing environment. Its operational model involves navigating a complex array of factors for precise measurements, with a foundational understanding essential for optimal utilization, especially in scenarios with multiple processors or VMs. At the core of its methodology is the principle of timesharing, where processes are allocated CPU time in brief intervals known as jiffies. This nuanced understanding of resource utilization across processes is crucial for measuring power consumption on a per-process basis. Power sensors, such as Intel's RAPL technology, monitor energy consumption in real-time, correlating jiffies with overall power usage to calculate the power consumption attributable to individual processes. The PowercapRAPL sensor operates synergistically with the powercap Linux kernel module, recording energy consumption metrics for real-time monitoring and extensive data processing through its exporters. Acknowledging the evolving landscape of computing workloads, especially those with heavy GPU utilization, Scaphandre is poised for further enhancements. Future iterations aim to integrate more granular tracking mechanisms for GPU and hard drive energy consumption, bridging monitoring gaps and providing a holistic view of a host's power profile. Understanding its operational model is crucial for leveraging its full potential, ensuring accurate and insightful power analysis in diverse computing environments.

3.5.5. Meross MSS310

One of the options available for measuring power consumption is to use the Meross MSS310 smart plug.[12] These devices from Meross are smart plugs that enable users to obtain metrics on the current power consumption via the internet. The device itself is straightforward, consisting of a male-to-female power socket, a button, and a microcontroller equipped with a wireless antenna that can connect to a basic Wireless Local Area Network (WLAN) with internet access. When the device is properly configured and set up, it transmits data to the Meross Message Queuing Telemetry Transport (MQTT) server. The MQTT topic is accessible through the *meross_iot*[13] Python library and requires an email and password for authentication. A straightforward Python script enables us to gather data from the devices and automate the process.

Meross MSS310 device can measure: (i) *Power*: the current active power; (ii) *Voltage*: the current voltage; and (iii) *Current*: the current intensity of the electrical current. While the Meross MSS310 smart plug provides an accessible means for monitoring power consumption metrics, it is pertinent to note the inherent constraints associated with its commercial nature. Specifically, the technical specifications and the underlying methodologies employed by the device for power measurement are not publicly disclosed by the manufacturer. This lack of detailed technical information may pose challenges for researchers and practitioners seeking a deep understanding of the measurement precision and the algorithmic processes involved. However, it is essential to recognize that the Meross MSS310 is designed as a measurement tool, not merely an estimator. This distinction emphasizes its capability to provide actual measurements of power, voltage, and current intensity, rather than inferred estimates. The device achieves this through its integrated sensors and electronic components, which directly quantify the electrical parameters flowing through it. Consequently, despite the absence of explicit technical documentation, the Meross MSS310's utility as a reliable tool for real-time power consumption monitoring remains indisputable.

3.6. Performance analysis with stress-ng

Stress-ng is an important tool for generating controlled loads across various components of the system. Its wide range of stressors makes it an ideal candidate for mimicking real-world workloads. However, to enhance the flexibility, portability, and ease of deployment across various test environments, we aim to containerize the stress-ng tool. Thus, we can ensure a uniform testing environment across different platforms, mitigating the variability introduced by external dependencies and environment-specific configurations.

The containerized *stress-ng*[14] is designed to retain the same level of configurability as its bare-metal counterpart. This enables researchers and developers to replicate actual workload scenarios with precision, but also provides a modular framework where stress parameters can be adjusted dynamically. By deploying stress-ng in a container and allocating all available CPU resources within a K8s cluster, the tool can precisely mirror the entire spectrum of workload intensities. This method allows for the simulation of CPU usage from 0% (idle) to 100% (peak load) with a singular container instance. In practical deployments, while it may be unusual for a single container to monopolize such a vast expanse of resources. However, the scenario where multiple containers cumulatively consume a significant amount of CPU resources is a usual thing. Hence, the simulation conducted with a single stress-ng container, consuming an equivalent amount of CPU resources, provides a valid approximation of a real-time scenario where multiple containers are operating under varying loads. Furthermore, the efficacy of stress-ng in simulating CPU workloads extends seamlessly to multi-core systems, thanks to K8s flexible resource allocation policies. Within a K8s environment, a container, including those running stress-ng, is not inherently restricted in its CPU resource consumption. This means that, unless explicitly limited through K8s resource management features, the container has the potential to utilize all CPU resources available on the node where it is deployed. This capability is particularly significant in the context of multi-core systems, where stress-ng can be leveraged to generate stress loads that span across all cores, effectively mimicking scenarios where the system is subjected to varied degrees of utilization from idle to peak load.

4. Experimental evaluation setup

Fig. 2 (based on Fig. 1) shows our designed experimental evaluation reference architecture setup. Here, an *application layer* is responsible for (i) automating the experiment procedure; (ii) collecting, aggregating, and storing data; and (iii) visualizing the power consumption. Whereas, a physical layer includes the physical facilities required to run the software. This section describes the deployed evaluation setup.

[12] https://www.meross.com/en-gc/smart-plug/smart-plug-google-home/6
[13] https://pypi.org/project/meross-iot/
[14] https://anonymous.4open.science/r/stress-ng-ubuntu-4E06/README.MD

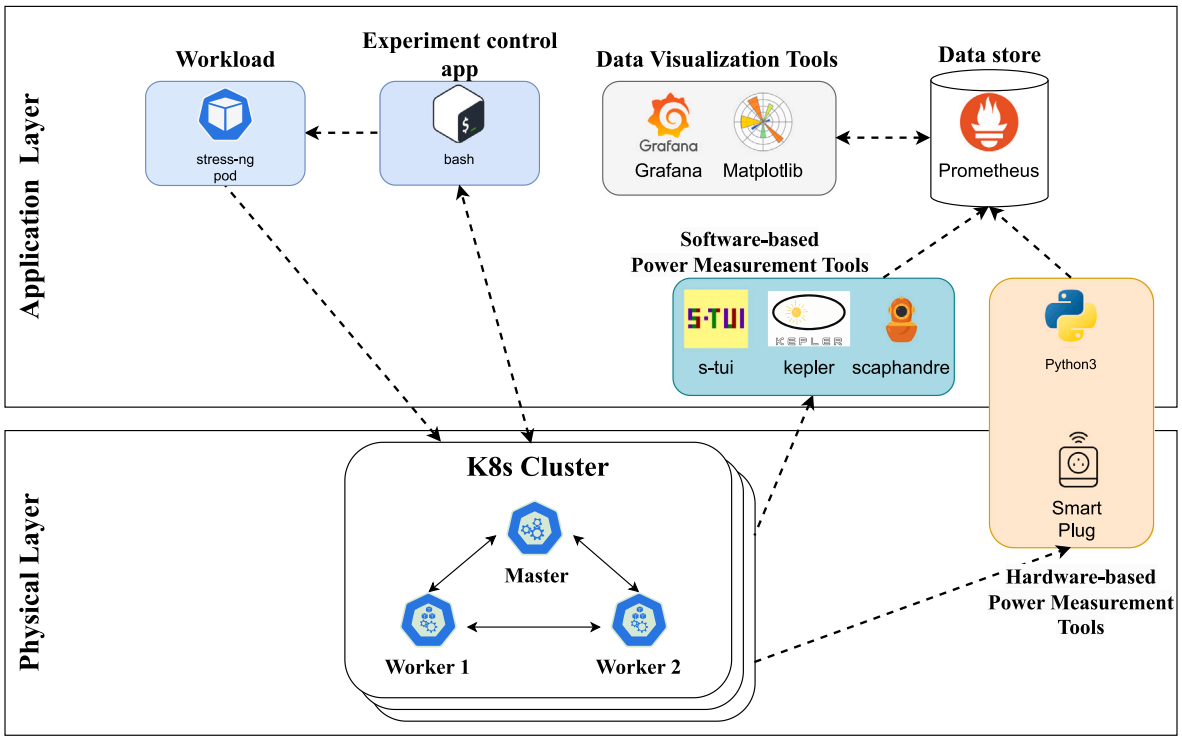


Fig. 2. Experimental evaluation reference architecture setup.

(a) *Virtual Wall (vWall)* testbed [36]. (b) *eCluster* testbed. (c) *piCluster* testbed.

Fig. 3. Overview of each K8s-based infrastructures used in the evaluation.

4.1. Physical layer

The physical layer represents where the computation and storage workloads takes place. Fig. 3 illustrates the considered K8s-based infrastructures. A set of K8s clusters compose the infrastructure to which we run the experimentation. Each K8s cluster is composed of a master node and two worker nodes. A three-node deployment is selected for all clusters to ensure reliable and highly available computing resources, since three represents the minimum node count required for a high-reliability cluster deployment. This choice does not affect our measurements, and a cluster with a greater number of nodes can be easily selected without invalidating the experimental process.

We deployed three distinct clusters across various architectures to validate the results through different scenarios. The first cluster (Fig. 3(a)) is deployed in the imec Virtual Wall (vWall) infrastructure [36] at IDLab, Belgium. The second and third clusters, namely the *eCluster* and the *piCluster*, are implemented within the *MEC Lab* infrastructure at the University of L'Aquila, Italy, as depicted in Fig. 3(b) and Fig. 3(c), respectively.

Table 2 details the hardware configurations of each cluster and Table 3 lists the applied software versions, including monitoring and performance assessment tools.

Table 2
The Hardware Configuration of each cluster.

Virtual Wall (vWall) Cluster		
Node	CPU	Memory
Master Worker 1 Worker 2	Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60 GHz	48 GB
eCluster		
Node	CPU	Memory
e1 e2 e3	Intel(R) Core(TM) i9-9900K CPU @ 3.60 GHz	64 GB
piCluster		
Node	CPU	Memory
pione pitwo pithree	Broadcom BCM2711 ARM v8 @ 1.8 GHz	4 GB

Table 3
Software used in the different K8s clusters.

Software in common across all clusters	
Software	Version
Container Runtime	containerd://1.6.12
Kube-prometheus	v0.68.0
s-tui	1.0.0-beta3
Kepler	0.5.5
Scaphandre	0.1.1
Stress-ng	0.13.12
Python3	3.8.20
Virtual Wall (vWall) Cluster	
Software	Version
Kubeadm & Kubectl	v1.26.1
Operating System	Ubuntu 20.04.2 LTS
Linux Kernel	5.4.0-33-generic
eCluster	
Software	Version
Microk8s	v1.27.5
Operating System	Ubuntu 20.04.6 LTS
Linux Kernel	5.15.0-86-generic
piCluster	
Software	Version
Microk8s	v1.27.5
Operating System	Debian 11
Linux Kernel	6.1.21-v8+

4.2. Application layer

The application layer is responsible for running and controlling the experiment, collecting all the data into a centralized data store, and visualizing the collected data. The *workload* is based on our stress-ng container implementation presented in Section 3.6. The *Experiment control app* is responsible to run the experiment. It manages the current *workload*, generating the required pods and deploying them to the appropriate node within theCluster. The *Software-based* and *Hardware-based* power measurements tools report to the *Prometheus* datastore all collected metrics during the experiment. It is worth noting that the *Hardware-based Power Measurement Tools* are composed by an hardware and an application part. The hardware part is the bare metal socket plugged to the energy socket while the *Python* application is the agent able to collect and forward metrics to the *Data store*. *Software-based Power Measurement Tools* only require to run their agents that forwards data into the *Data store*, as they estimate power measurement by software, as discussed in Section 3. This setup enable real time data visualization through the Grafana web-based Graphical User interface.

4.3. Mapping energy monitoring tools to hardware platforms

As illustrated in Table 4, the selection of an appropriate energy monitoring tool is contingent upon the underlying hardware platforms. Our comprehensive analysis presents the compatibility of various tools (*i.e.*, Scaphandre, Kepler, S-tui, and Meross) with prominent hardware platforms such as Intel, AMD, and ARM. Notably, Scaphandre, Kepler, and S-tui exhibit universal applicability across recent Intel and AMD platforms (support enabled from 2012 by the RAPL technology). A green checkmark (✓) means no restrictions on tool usage. Conversely, these tools are not suited for ARM-based systems, as indicated by a red cross (✕), suggesting their inability to provide accurate measurements or function effectively within such hardware platforms.

The Meross tool, however, is a hardware-based tool and presents a unique case; its application is feasible across all examined platforms, albeit with a conditional requirement for physical access to the system, as highlighted by a yellow checkmark (✓). This concern is generally more relevant in the context of cloud service providers and may not be as significant in on-demand DC operations. Given these insights, this study proposes a deliberate approach to choosing adequate energy monitoring components that focuses on matching the requirements of these tools with the particular characteristics of the hardware platform being monitored. This alignment is crucial for achieving accurate energy consumption measurements and facilitating effective energy-aware strategies within containerized cloud environments.

4.4. Experimental run

To evaluate the power measurement tools, it is essential to execute a workload on the cluster that emulates real-world behavior. We manage the workload deploying our *stress-ng* pod for different CPU loads, as illustrated in Fig. 4 as a job in kubernetes. The first thing is to run the *s-tui* software. Next, if the *Meross* smart plug is available in the current cluster, the Python agent (which runs on an external hardware to avoid adding computation to the system under-test) to forward data from meross to the data center needs to be run. At this point, only monitoring agents are running on the system and the power consumption of them is negligible compared to our available computational resources. Then, the experiment automation begins by deploying the first job in the cluster representing the first step of the experiments, accordingly to Fig. 4. After a job completes, there is a deliberate waiting interval before deploying the next job to begin the subsequent step. This interval is strategically implemented to allow the system to cool down and to ensure a clear separation between experiments, thereby minimizing thermal interference and potential carry-over effects from previous jobs. This cooling-off period is crucial for maintaining the integrity of each experiment by ensuring that the starting conditions are consistent and not influenced by the residual heat or system state from preceding tasks Once all jobs are completed, the results can be consolidated into a CSV file, marking the end of the experiment. In each step, the *stress-ng* pod generates a given CPU load over a period of 300 s (*i.e.*, 5 min). The load percentage is determined based on the total available CPU in the node. Thus, 50% of 4 CPUs means loading 4 CPUs to half their maximum capacity for the duration of the job and *stress-ng* ensures this behavior. For instance, if a workload is configured to utilize 30% of the CPU resources over a given time, *stress-ng* uses exactly 30% of the CPU's capabilities. Should other concurrent processes or pods be operating on the system during the same time, the CPU usage recorded by the system reflects the cumulative resource utilization by all active entities. As future work, it is important to highlight that when evaluating the power consumption of containerized applications, *stress-ng* would not be needed since measurements are collected based on these running pods on the K8s cluster. Though, to quantify the power consumption of the different evaluated clusters, we opted for developing a *stress-ng* pod to understand the power consumption behavior in a K8s cluster when different CPU resources are attributed. This ensures that our experimental setup accurately simulates real-time conditions, allowing us to measure the power consumption and CPU utilization with high fidelity. Each step is followed by a waiting interval of 30 s to cool down the system and let us better separate each job from the previous one in the collected values.

During our experiments, the collection interval of Prometheus is increased (*i.e.*, every 1 s) since the sampling rate may affect the accuracy of the reported values. As default, Prometheus collects at a lower sampling rate (*i.e.*, typically every 10–15 s), which may miss short spikes or fluctuations in power consumption, leading to underestimation or overestimation of overall power usage. Note that a higher sampling rate is considered to reduce the impact of these fluctuations in our measurements.

The experiment starts with a 5% CPU load, with the count growing linearly with each step, reaching 100% in the final step. During this process, the power consumption is measured by using different tools at every step for subsequent analysis (*i.e.*, *s-tui*, *Kepler*, *scaphandre*, and *Meross* smart plug). To enhance data visualization, we extracted

Table 4
Mapping of energy monitoring tools to hardware platforms.

	Energy Monitoring Tool			
Hardware Platform	**Scaphandre**	**Kepler**	**S-tui**	**Meross**
Intel	✓	✓	✓	✓(if physical access is possible)
AMD	✓	✓	✓	✓(if physical access is possible)
ARM	×	×	×	✓(if physical access is possible)

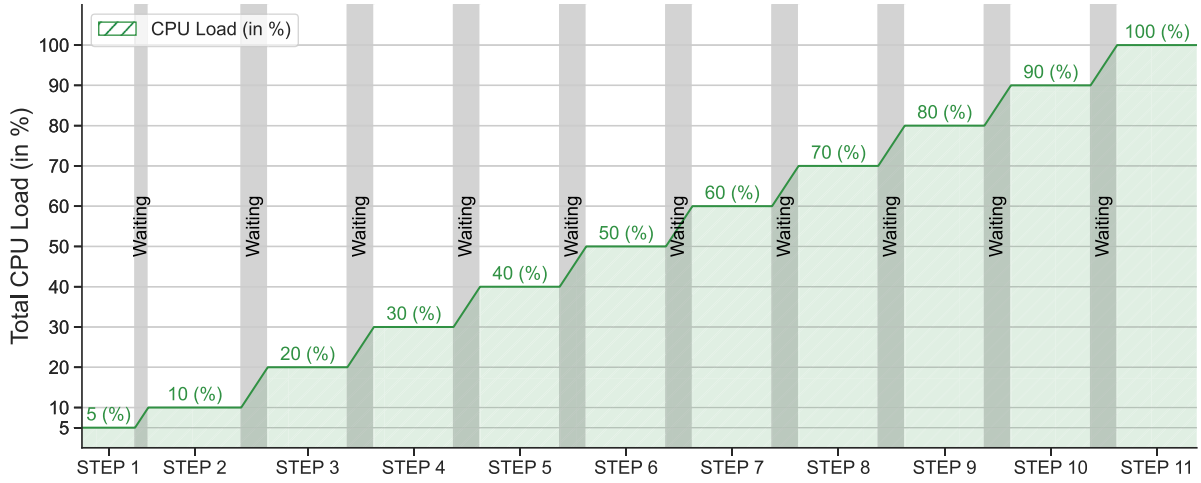


Fig. 4. Diagram showing a single experimental run: CPU load incrementally increases from 5% to a maximum of 100%. Each step lasts 300 s (5 min), followed by a 30-second waiting interval.

Table 5
The Resource consumption of all methods in the vWall testbed.

Tool	CPU (in m)	Memory (in Mi)
Kepler	125.49 ± 3.26	4.35 ± 0.50
Scaphandre	66.96 ± 12.88	1.21 ± 0.45
s-tui	51.30 ± 23.60	22.41 ± 3.51

CSV files from Prometheus, and subsequently plotted the values offline using *Matplotlib*. *ServiceMonitors* for Scaphandre and Kepler agents and the Python agent for *Meross* need to be running within the K8s-based clusters during the whole experiment to collect data (Fig. 5). It should be noted that due to limited server accessibility, the *Meross* smart plug is not installed on the *vWall* cluster. Furthermore, both *Kepler* and *Scaphandre* are incompatible with the ARM architecture, preventing us from collecting data on the *pi cluster*. The *s-tui* runs on ARM but is not able to estimate power consumption.

5. Results

Resource Consumption is measured using different power measurement tools at the vWall infrastructure.

Table 5 shows the measured CPU and memory while performing the experiment. Note that the impact of the considered tools on resource consumption is also observed and Kepler uses more CPU than both Scaphandre and s-tui, whereas s-tui consumes significantly more memory than the other two tools.

vWall: Fig. 6(a) compares the obtained power consumption measurements at the *vWall* testbed with three distinct power measuring tools: *Scaphandre*, *Kepler*, and *S-tui*. vWall testbed is accessed via the *stress-ng* pod developed for our experiments. The Meross device is not installed due to limited physical access to the machines. There are notable differences in the power consumption values reported by the three tools at identical CPU loads. The considered tools exhibit an increase in power consumption with the increase of CPU load as expected. However, *Kepler* provides low power consumption values compared to both *Scaphandre* and *S-tui* due to the inherent behavior of their implementation. Kepler's measurements show a near-linear correlation with CPU load, starting from approximately 13 watts at 0%, and concluding around 150 watts at 100%.

eCluster: Fig. 6(b) depicts the relationship between both the power consumption and the total CPU load for four different power measurement tools: *Scaphandre*, *Kepler*, *S-tui*, and *Meross*. As stated, Kepler consistently underestimates the power consumption across all the CPU loads when compared to the other tools. The discrepancy widens as the CPU load increases, suggesting that its approximation method becomes less accurate under heavier loads. Results collected from *Scaphandre* are very close to the one given by Kepler and the gap between the two tools remains relatively stable at low and high CPU loads. *S-tui* starts by a higher point on the graph at lower CPU loads but tends to increase as the load increases. Meross data are very close to the *S-tui* ones from 5% to 40% CPU load. After 40% of CPU load, the Meross data diverges from *S-tui* and tends to a constant maximum value of 150 W.

Fig. 7(a) compares the power consumption over time between a single node labeled as *e2* — from the eCluster as shown in Fig. 3(b) — and the entire eCluster, measured from Meross devices. Both the single node *e2* and the whole eCluster exhibits fluctuating power consumption over the given period, suggesting multiple periods of high and low activity or usage. These values are directly tied to the actual workload being executed on the *e2*node, according to Section 4. The power consumption of the cluster consistently remains higher than *e2* as expected. The peaks and lows in the consumption curves align at several points, meaning the two other nodes of the cluster (*i.e.*, e1, and e2) are only running core K8s components, so resulting in constant power consumption. The *stress-ng* workload consumption is slightly higher than the noise coming from the normal oscillation of the baseline power consumption of the other two nodes. The baseline power consumption values for both *e2* and *eCluster* can be observed at the end of the experiment upon the termination of the *stress-ng* pod.

piCluster: Fig. 7(b) compares the power consumption over time between a single node, labeled as *pitwo*, and the entire PiCluster. In the *piCluster* we are not able to install and collect data from Kepler and Scaphandre due to hardware limitations and software compatibility issues. Both the *pitwo* node and the PiCluster exhibits fluctuating power consumption over time. It is worth highlighting that in the *piCluster*, the absolute power consumption values are an order of magnitude

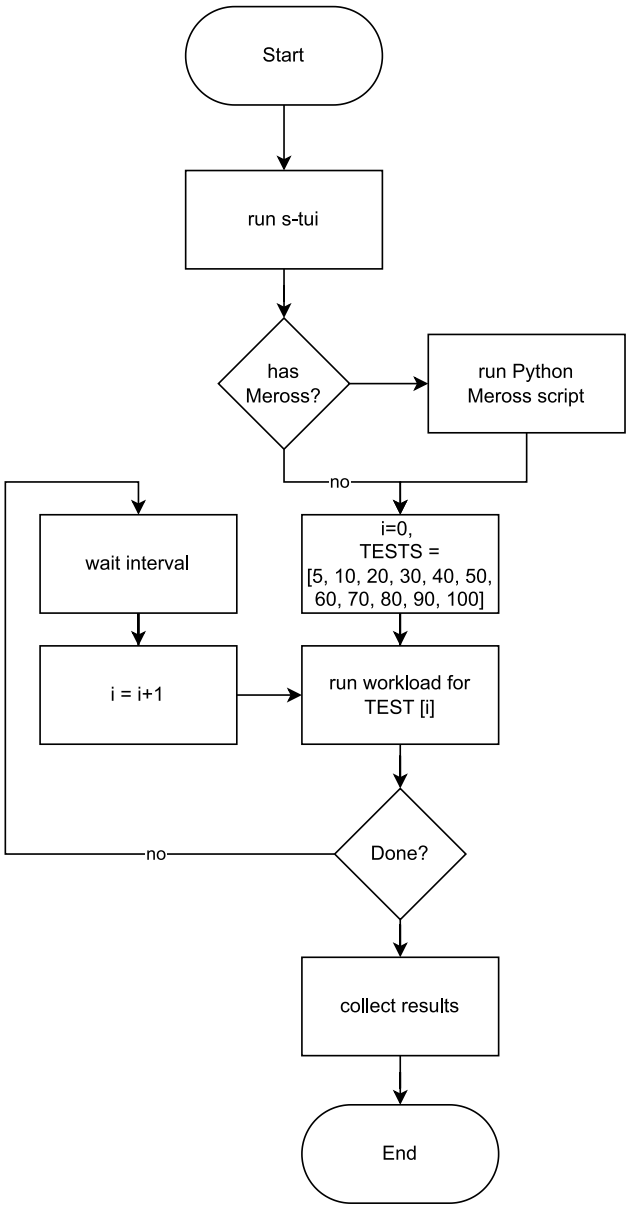
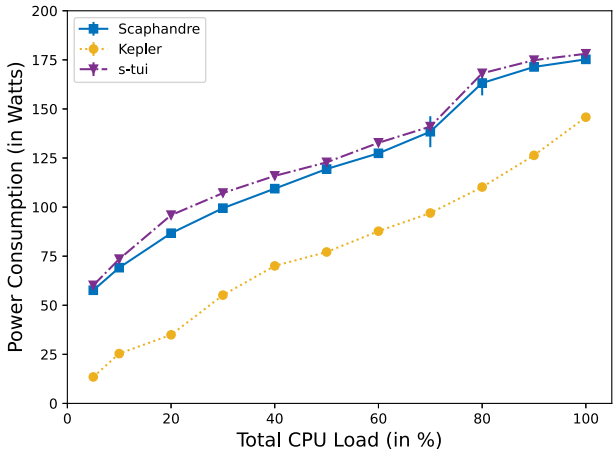
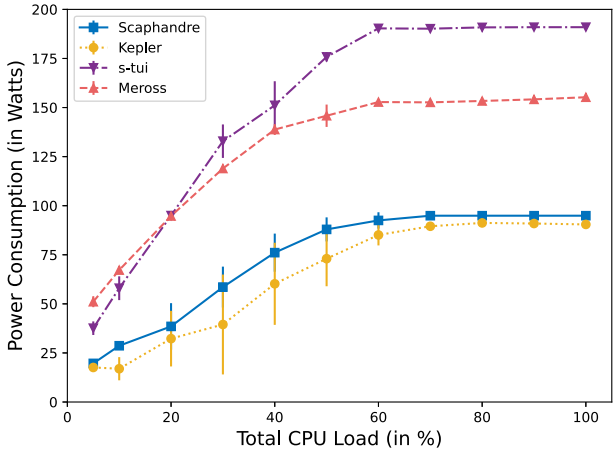


Fig. 5. The block diagram of our experimental run.



(a) *Virtual Wall (vWall)* testbed.



(b) *eCluster* testbed.

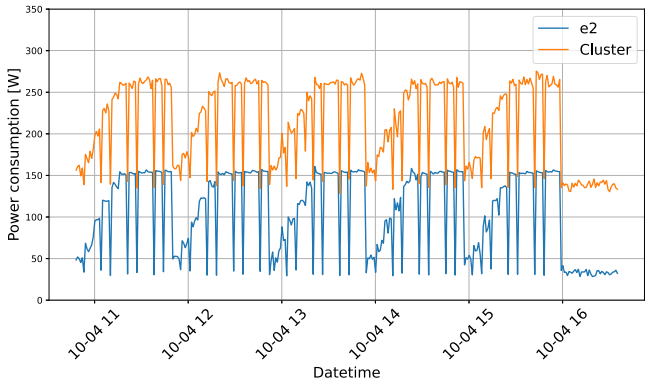
Fig. 6. Overview of the energy consumption for both K8s-based infrastructures.

lower than those in the *eCluster*. This disparity can be attributed to the superior energy efficiency of the Raspberry Pi compared to workstation-grade computers involved in *eCluster* and *vWall Cluster*. The general trend in the data aligns with that observed for the *eCluster*, where the cluster's power consumption generally mirrors the power increase in *pitwo*. A key distinction, however, is that in this context, the power fluctuations originating from the *pione* and *pithree* nodes significantly impact the measurement and cannot be overlooked. This noise is evident at the end of the graph, where no workload is pushed into the cluster, and we observe noise that is slightly higher in percentage compared to that of the *eCluster*.

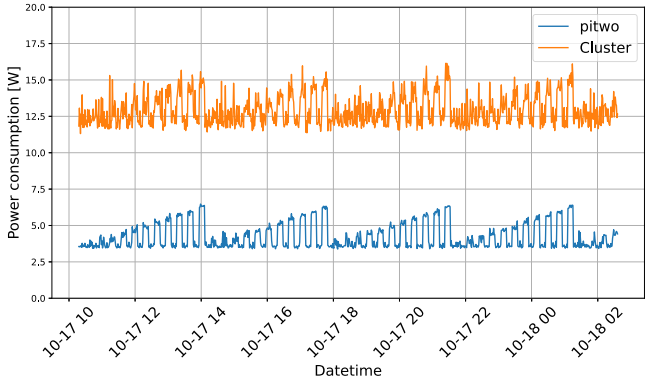
In summary, all the tools applied in the evaluation showed power consumption measurements. All these tools show significant discrepancies for the three K8s-based clusters due to the nature of the tools developed. Scaphandre and Kepler use relatively complex algorithms to estimate the power consumption based on different factors such as the system activity, and CPU utilization. Kepler includes a ML-based model to estimate the power consumption of different workloads. The discrepancies obtained in our experiments can be related to the different algorithms employed by these tools to measure power consumption. Discrepancies among the use of different methods for power consumption estimation reported in previous works [37], in which the impact of using container orchestration platforms such as K8s has not been evaluated. TDP can serve as a metric for estimating power consumption in electronic devices, including processors. However, its reliability can be compromised due to various factors, leading to inaccuracies in power consumption estimations. Factors such as variations in workload intensity, ambient temperature, and manufacturing discrepancies can all contribute to deviations between estimated TDP and actual power consumption. Thus, there is a pressing need for further investigation to evaluate the robustness of these monitoring tools in diverse environmental conditions. Our experiments indicate that Scaphandre and Kepler exhibit promising performance characteristics and may serve as suitable candidates for accurate power consumption monitoring in various settings, especially in Bare metal servers. Though, these tools are not yet optimized to operate on the ARM architecture. However, continued research is essential to validate and refine these findings, ensuring reliable power consumption estimation across different environments and use cases.

6. Open challenges & future directions

This section discusses the challenges of cloud sustainability is not yet fully addressed by literature and highlights future directions.



(a) *eCluster* testbed.



(b) *piCluster* testbed.

Fig. 7. Comparison of power consumption over time between a single node (e2) and the entire Cluster (eCluster).

Containers have revolutionized application development and deployment, offering high flexibility and enhancements in scheduling and scaling features. However, this transition is accompanied by multiple challenges related to power consumption and sustainability, making it essential to address these issues to ensure the sustainability and cost-effectiveness of container-based cloud infrastructures.

Power-aware resource allocation for emerging use cases based on application stringent requirements such as extremely low latency, and high bandwidth presents a substantial avenue for reducing power consumption. Existing works [38–40] propose theoretical formulations and heuristics to allocate VNF resources based on power consumption. However, future studies should address intelligent algorithms for dynamic resource allocation to minimize idle resources and enhance overall energy efficiency. In addition, optimizing service placement and traffic routing in cloud infrastructures will help cloud providers reduce their deployment costs while providing sufficient levels of Quality of Service (QoS) to their users.

Integration of Machine Learning (ML) features in container management features can lead to autonomous functionalities as a potential solution for inappropriate human interventions. Leveraging ML and predictive analytics holds promise in anticipating workload patterns and modifying the number of deployed containers according to the current service demand. Future research efforts should concentrate on developing accurate predictive scaling algorithms capable of adjusting to dynamic workload patterns. Recently, RL has shown tremendous potential in resource allocation and auto-scaling tasks [41,42] based on its performance and scalability. Combining ML trends can lead to energy-efficient automated networks with minimum human intervention, providing self-configuration and self-repairing features focused on energy sustainability.

Energy-efficient hardware and carbon-neutral processing will be a major research topic in the next few years [43]. Existing works mainly address power consumption reduction in the cloud infrastructure and its network links [44]. The integration of renewable energy (*e.g.*, solar, wind) sources into data centers could significantly reduce power consumption [45]. Future research should focus on developing intelligent algorithms that maximize the utilization of renewable energy sources when available and seamlessly transition to conventional power sources as needed. In addition, hardware manufacturers should focus on developing specialized processors and hardware components (*e.g.*, efficient memory systems and storage solutions) optimized for containerized environments. Efficient cooling mechanisms that maximize energy savings is also a potential research direction that could lead to overall reduction of power consumption in DCs [46].

Efficient Carbon Footprint Measurement of containerized applications is crucial [46–50]. Research collaborations between industry, academia, and non-profit stakeholders are essential to drive knowledge sharing and collective efforts to address sustainability challenges in container clouds and lead to standardized methodologies and industry-wide best practices. An example, is the recent Cloud Native Computing Foundation (CNCF) Technical Advisory Group (TAG) Environmental Sustainability initiative[15] advocating for sustainable practices in cloud native technologies. Sharing data and research project outcomes can accelerate the progress in this field by developing and disseminating guidelines for optimizing power consumption and sustainability, promoting consistent implementation across diverse organizations. Open-source communities could play a major role in coming years to achieve a standardized way of measuring power consumption in future cloud infrastructures.

In summary, this section emphasizes the critical need for extensive research in the cloud domain related to energy sustainability to push for innovations that address power consumption and sustainability challenges in container clouds. Ongoing research studies, interdisciplinary collaborations, and the pursuit of standardized best practices are essential elements in the journey towards more sustainable and energy-efficient container-based cloud infrastructures. All these research areas will contribute for a greener cloud future.

7. Lessons learned and prospects

This section reflects on several lessons derived from our literature review and findings. In addition, future prospects are detailed, including the need for open collaboration between academia and Industry and substantial efforts in standardization.

7.1. Lessons learned

Optimizing Power Consumption in cloud-based environments is vital for cloud computing sustainability. Efficient power management can directly affect operational costs, overall performance of applications and the reliability of cloud services. Energy costs represent a significant portion of the operational expenses of DCs. Energy-efficient practices such as dynamic resource allocation and advanced cooling systems will directly translate to cost savings. Currently, DCs are notable contributors to carbon emissions and environmental pollution due to their massive power demands, often relying on non-renewable energy sources. By adopting renewable sources, cloud providers could significantly decrease their carbon footprint, aligning with global sustainability goals. Finding a balance between performance and energy efficiency is a delicate yet critical task, since this trade-off can significantly impact the environmental footprint and scalability of DCs. Power-aware management strategies will enable cloud providers to

[15] https://tag-env-sustainability.cncf.io/

adapt quickly to sudden demands, ensuring seamless and responsive user experiences.

Leveraging Precise Monitoring is crucial for understanding, analyzing, and optimizing power usage in the dynamic and rapidly evolving cloud environment. While various measurement tools are available, this article highlights that the measured power consumption can vary depending on the chosen technique. Thus, accurate monitoring becomes essential to provide real-time, granular data on power usage across various components and processes within the cloud infrastructure. This enhanced data is invaluable for identifying inefficiencies and anomalies at different levels (*e.g.*, server and rack). Sudden spikes or unusual power consumption can lead to performance issues, so detecting these issues is vital to maintaining efficient power consumption. In addition, accurate monitoring will be crucial for optimizing resource allocation and load balancing, minimizing energy wastage. Proactive strategies can be applied to ensure the cloud infrastructure can effectively meet future demands while reducing unnecessary resource consumption. In conclusion, advanced monitoring tools will allow cloud providers to make informed decisions to increase energy efficiency, reduce operational costs, and minimize environmental impact.

Collaboration and Standardization are key to really achieve energy efficiency goals in the coming years. Researchers, industry professionals, and policymakers can share knowledge, best practices, and insights, leading to comprehensive strategies for sustainable cloud computing. Establishing a standardized manner for measuring and benchmarking power consumption in cloud environments will lead to consistent assessment and continuous improvement across the cloud computing area. Engaging collaborations between academia and industry will push research and innovation forward by facilitating the development of practical solutions that are both technologically advanced and economically viable. By leveraging collective expertise and advocating for standardized practices, the cloud computing community can accelerate progress towards a greener and more sustainable future.

7.2. Future prospects

The development of sophisticated Power-Aware Algorithms that can dynamically adapt resource allocation in near real-time based on power availability and workload demands will be extremely important towards a more sustainable cloud infrastructure. These novel algorithms should address multiple factors (*e.g.*, diverse traffic patterns, energy cost of different sources) to find a balance between optimizing power consumption while maintaining high performance and user experience. This balance will lead to significant energy savings and improved resource utilization while avoiding power wastage during idle or low-demand periods. Recently, ML-based algorithms have shown their benefits compared to more traditional approaches, being able to optimize energy consumption dynamically. Future research on power-aware algorithms is fundamental to a more sustainable and energy-efficient cloud computing landscape.

Integration of Renewable Energy Sources holds immense promise for transforming the landscape of cloud computing. The integration of renewable energy into DCs and cloud infrastructures is a crucial step towards reducing the carbon footprint. For example, solar energy is one of the most accessible renewable sources. Cloud providers could significantly reduce their reliance on fossil fuels by harnessing solar power through photovoltaic cells installed on DC rooftops. Wind turbines could also generate substantial amounts of electricity if placed in regions with consistent wind patterns. Their integration into DCs could provide a reliable and clean power source, contributing to a greener operational profile. Hydropower Solutions could also offer a consistent energy source if DCs are in regions with access to rivers or streams. Nonetheless, robust energy storage solutions are needed to accomplish an efficient integration of renewable energy sources. Novel energy storage solutions (*e.g.*, improved large-scale batteries, new hydrogen storage options) are needed to ensure a consistent and reliable power supply, improving the resilience of cloud computing. By leveraging renewable energy sources, cloud providers can transition towards a more sustainable and environmentally responsible operational model, fostering a greener digital future.

Optimized Containerization offers a promising avenue for improving energy efficiency in cloud deployments. Containers revolutionized application deployment in modern cloud platforms, but further enhancements focused on energy efficiency are needed. Studying lightweight, energy-efficient container packaging and runtimes will be crucial in achieving sustainable cloud computing. These runtimes could prioritize energy conservation while maintaining application performance, contributing to sustainable container operations. The efficient utilization of containers reduces resource wastage and contributes significantly to energy savings. In addition, container workloads to match the power profile of the underlying hardware are essential. Power-aware operations can be achieved by investigating hardware-specific power usage patterns and adapting container configurations accordingly. Further research on container recycling strategies could also help reduce the overhead of creating and destroying containers, minimize the associated power consumption, and contribute to a more sustainable container ecosystem.

8. Conclusions

Containers have revolutionized how applications are deployed and managed today, but challenges in power consumption and sustainability in cloud computing persist. This paper outlines key challenges and proposes future directions, stressing the importance of academic and industry collaboration to measure power consumption efficiently. Accurate monitoring and estimation of power consumption are vital for efficient energy-aware management in future cloud infrastructures. Energy-aware algorithms and the addition of renewable energy sources can significantly contribute to reducing carbon footprints of current DCs. We emphasize that standardizing best practices to measuring the power is essential for fostering a sustainable container ecosystem. Our results show that current power measurement technologies provide different results. Modifications to the underlying hardware can influence the behavior of power measurement curves. However, we understood that many existing tools are not yet optimized to operate on the ARM architecture. Moreover, we observed that the different methodologies and algorithm estimations of the considered tools — software-based estimation, sensor readings in power socket — may impact the results, as demonstrated in power consumption for the different K8s clusters. The obtained results demonstrate that further research and gaps in energy measurement tools implementation to be addressed for precise measurement of power consumption in future cloud and edge infrastructures. As future work, energy-aware scheduling methods will be designed and developed for the K8s platform based on the collected metrics with these monitoring tools. The aim is to improve scheduling decisions in K8s by ensuring pods are not placed in overloaded nodes, currently consuming an excessive amount of power. Also, efficient (de)scheduling policies will be studied to terminate pods in overloaded nodes, ensuring a more sustainable and energy efficient placement in container-based computing environments.

CRediT authorship contribution statement

Carlo Centofanti: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Resources, Software, Validation, Writing – original draft, Writing – review & editing. **José Santos:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Venkateswarlu Gudepu:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Koteswararao Kondepu:** Methodology, Project administration, Supervision, Validation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Results added to the repository we shared in our work.

Acknowledgments

José Santos is funded by the Research Foundation Flanders (FWO), grant number 1299323N. This work has been partially funded by the European Union - NextGenerationEU under the Italian Ministry of University and Research (MUR) National Innovation Ecosystem grant ECS00000041 - VITALITY - CUP E13C22001060006. This work has been partially supported by the European Commission through SNS Joint Undertaken under grant agreement No. 101096120(SEASON) and H2020-MSCA-RISE OPTIMIST project (GA: 872866). This work partly received funding from the DST SERB Startup Research Grant (SRG-2021-001522), the SGNF project (“Reliability Evaluation of Virtualised 5G”).

References

[1] F. Douglis, J. Nieh, Microservices and containers, IEEE Internet Comput. 23 (6) (2019) 5–6.
[2] N. Jones, et al., How to stop data centres from gobbling up the world’s electricity, Nature 561 (7722) (2018) 163–166.
[3] J. Shalf, The future of computing beyond Moore’s Law, Phil. Trans. R. Soc. A 378 (2166) (2020) 20190061.
[4] F. Dahlqvist, M. Patel, A. Rajko, J. Shulman, Growing Opportunities in the Internet of Things, McKinsey & Company, 2019, pp. 1–6.
[5] J. Santos, T. Wauters, B. Volckaert, F. De Turck, Fog computing: Enabling the management and orchestration of smart city applications in 5G networks, Entropy 20 (1) (2017) 4.
[6] K. Cao, Y. Liu, G. Meng, Q. Sun, An overview on edge computing research, IEEE Access 8 (2020) 85714–85728.
[7] J. Santos, T. Wauters, B. Volckaert, F. De Turck, Towards low-latency service delivery in a continuum of virtual resources: State-of-the-art and research directions, IEEE Commun. Surv. Tutor. 23 (4) (2021) 2557–2589.
[8] B. Burns, J. Beda, K. Hightower, L. Evenson, Kubernetes: Up and Running, “O’Reilly Media, Inc.”, 2022.
[9] S.F. Piraghaj, A.V. Dastjerdi, R.N. Calheiros, R. Buyya, A framework and algorithm for energy efficient container consolidation in cloud data centers, in: IEEE International Conference on Data Science and Data Intensive Systems, IEEE, 2015, pp. 368–375.
[10] T. Van De Voort, V. Zavrel, I.T. GALDIZ, J. Hensen, Analysis of performance metrics for data center efficiency, REHVA J. (February) (2017).
[11] L.N. Vijouyeh, M. Sabaei, J. Santos, T. Wauters, B. Volckaert, F. De Turck, Efficient application deployment in fog-enabled infrastructures, in: 16th International Conference on Network and Service Management, CNSM, IEEE, 2020, pp. 1–9.
[12] J. Santos, T. Wauters, B. Volckaert, F. De Turck, Towards end-to-end resource provisioning in fog computing over low power wide area networks, J. Netw. Comput. Appl. 175 (2021) 102915.
[13] L.M.M. Zorello, M. Sodano, S. Troia, G. Maier, Power-efficient baseband-function placement in latency-constrained 5G metro access, IEEE Trans. Green Commun. Netw. 6 (3) (2022) 1683–1696.
[14] T. Piontek, K. Haghshenas, M. Aiello, Carbon emission-aware job scheduling for Kubernetes deployments, J. Supercomput. (2023) 1–21.
[15] Q. Wu, Q. Deng, L. Ganesh, C.-H. Hsu, Y. Jin, S. Kumar, B. Li, J. Meza, Y.J. Song, Dynamo: Facebook’s data center-wide power management system, ACM SIGARCH Comput. Archit. News 44 (3) (2016) 469–480.
[16] M. Okhovvat, M.T. Kheirabadi, M.R. Okhovvat, A. Nodehi, Joint time and energy-optimal approach to allocate task to actors in wireless sensor actor networks, Comput. Netw. (2023) 110018.
[17] A. Bellin, M. Centenaro, F. Granelli, A preliminary study on the power consumption of virtualized edge 5G core networks, in: IEEE 9th International Conference on Network Softwarization, NetSoft, IEEE, 2023, pp. 420–425.
[18] G. Lando, L.A.F. Schierholt, M.P. Milesi, J.A. Wickboldt, Evaluating the performance of open source software implementations of the 5G network core, in: IEEE/IFIP Network Operations and Management Symposium, IEEE, 2023, pp. 1–7.
[19] N. Schmitt, L. Iffländer, A. Bauer, S. Kounev, Online power consumption estimation for functions in cloud applications, in: IEEE International Conference on Autonomic Computing, ICAC, IEEE, 2019, pp. 63–72.
[20] D. Saxena, A.K. Singh, Energy Aware Resource Efficient-(EARE) server consolidation framework for cloud datacenter, in: G.S. Hura, A.K. Singh, L. Siong Hoe (Eds.), Advances in Communication and Computational Technology, Springer Nature Singapore, Singapore, 2021, pp. 1455–1464.
[21] M. Ilbeigi, M. Ghomeishi, A. Dehghanbanadaki, Prediction and optimization of energy consumption in an office building using artificial neural network and a genetic algorithm, Sustainable Cities Soc. 61 (2020) 102325.
[22] H. Ma, A. Ding, Method for evaluation on energy consumption of cloud computing data center based on deep reinforcement learning, Electr. Power Syst. Res. 208 (2022) 107899.
[23] M.K.M. Shapi, N.A. Ramli, L.J. Awalin, Energy consumption prediction by using machine learning for smart building: Case study in Malaysia, Develop. Built Environ. 5 (2021) 100037.
[24] R. Fedrizzi, A. Bellin, C.E. Costa, F. Granelli, Building the digital twin of a MEC node: a data driven approach, in: IEEE 9th International Conference on Network Softwarization, NetSoft, 2023, pp. 444–449.
[25] C. Centofanti, W. Tiberti, A. Marotta, F. Graziosi, D. Cassioli, Latency-aware Kubernetes scheduling for microservices orchestration at the edge, in: IEEE 9th International Conference on Network Softwarization, NetSoft, IEEE, 2023, pp. 426–431.
[26] B. Martinez, M. Monton, I. Vilajosana, J.D. Prades, The power of models: Modeling power consumption for IoT devices, IEEE Sens. J. 15 (10) (2015) 5777–5789.
[27] C. Jin, X. Bai, C. Yang, W. Mao, X. Xu, A review of power consumption models of servers in data centers, Appl. Energy 265 (2020) 114806.
[28] T.L. Vasques, P. Moura, A. de Almeida, A review on energy efficiency and demand response with focus on small and medium data centers, Energy Efficiency 12 (2019) 1399–1428.
[29] J. Dogani, R. Namvar, F. Khunjush, Auto-scaling techniques in container-based cloud and edge/fog computing: Taxonomy and survey, Comput. Commun. (2023).
[30] M. Chakraborty, A.P. Kundan, Grafana, in: Monitoring Cloud-Native Applications: Lead Agile Operations Confidently using Open Source Software, Springer, 2021, pp. 187–240.
[31] M. Amaral, H. Chen, T. Chiba, R. Nakazawa, S. Choochotkaew, E.K. Lee, T. Eilam, Kepler: A framework to calculate the energy consumption of containerized applications, in: IEEE 16th International Conference on Cloud Computing, CLOUD, IEEE, 2023, pp. 69–71.
[32] D. Zoni, A. Galimberti, W. Fornaciari, A survey on run-time power monitors at the edge, ACM Comput. Surv. (2023).
[33] E. Jagroep, J.M.E. van der Werf, S. Jansen, M. Ferreira, J. Visser, Profiling energy profilers, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing, 2015, pp. 2198–2203.
[34] S. Benedict, Energy-aware performance analysis methodologies for HPC architectures—An exploratory study, J. Netw. Comput. Appl. 35 (6) (2012) 1709–1719.
[35] M. Hähnel, B. Döbel, M. Völp, H. Härtig, Measuring energy consumption for short code paths using RAPL, ACM SIGMETRICS Perform. Eval. Rev. 40 (3) (2012) 13–17.
[36] Virtual Wall, The Virtual Wall emulation environment, 2023, [Online]. Available: https://doc.ilabt.imec.be/ilabt/virtualwall/index.html. (Accessed on 28 May 2023).
[37] M. Jay, V. Ostapenco, L. Lefèvre, D. Trystram, A.-C. Orgerie, B. Fichel, An experimental comparison of software-based power meters: Focus on CPU and GPU, in: CCGrid 2023-23rd IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, IEEE, 2023, pp. 1–13.
[38] K. Hejja, X. Hesselbach, Offline and online power aware resource allocation algorithms with migration and delay constraints, Comput. Netw. 158 (2019) 17–34.
[39] A. Kaur, S. Kumar, D. Gupta, Y. Hamid, M. Hamdi, A. Ksibi, H. Elmannai, S. Saini, Algorithmic approach to virtual machine migration in cloud computing with updated SESA algorithm, Sensors 23 (13) (2023) 6117.
[40] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Gener. Comput. Syst. 28 (5) (2012) 755–768.
[41] G. Tesauro, N.K. Jong, R. Das, M.N. Bennani, A hybrid reinforcement learning approach to autonomic resource allocation, in: IEEE International Conference on Autonomic Computing, IEEE, 2006, pp. 65–73.
[42] J. Santos, T. Wauters, B. Volckaert, F. De Turck, gym-hpa: Efficient auto-scaling via reinforcement learning for complex microservice-based applications in Kubernetes, in: IEEE/IFIP Network Operations and Management Symposium, IEEE, 2023, pp. 1–9.
[43] S.S. Gill, H. Wu, P. Patros, C. Ottaviani, P. Arora, V.C. Pujol, D. Haunschild, A.K. Parlikad, O. Cetinkaya, H. Lutfiyya, et al., Modern computing: Vision and challenges, Telematics Inform. Rep. (2024) 100116.

[44] V. Gudepu, B. Chirumamilla, R.R. Tella, A. Bhattacharyya, S. Agarwal, L. Malakalapalli, C. Centofanti, J. Santos, K. Kondepu, EARNEST: Experimental analysis of RAN energy with open-source software tools, in: 16th International Conference on COMmunication Systems & NETworkS, COMSNETS, 2024, pp. 1148–1153.
[45] C. Flanagan, How Data Centers Are Driving The Renewable Energy Transition, Forbes, 2023.
[46] A.-M. Jorge, B.C. Luis del, S. Manuel, B. Juan Carlos, H. Matthew, S. Cristina Olivera, Green Data Centers: Opportunities for Decarbonization - Exploiting Sustainable Power-Supply Opportunities, ARTHUR D. LITTLE, 2023.
[47] T. Anderson, A. Belay, M. Chowdhury, A. Cidon, I. Zhang, Treehouse: A case for carbon-aware datacenter software, ACM SIGENERGY Energy Inform. Rev. 3 (3) (2023) 64–70.
[48] B. Acun, B. Lee, F. Kazhamiaka, K. Maeng, U. Gupta, M. Chakkaravarthy, D. Brooks, C.-J. Wu, Carbon explorer: A holistic framework for designing carbon aware datacenters, in: Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, vol. 2, 2023, pp. 118–132.
[49] K. McMillan, Carbon Accounting for Sustainable Computing in Cloud Provisioned Data Centers (Ph.D. thesis), Massachusetts Institute of Technology, 2023.
[50] H. Zhu, D. Zhang, H.H. Goh, S. Wang, T. Ahmad, D. Mao, T. Liu, H. Zhao, T. Wu, Future data center energy-conservation and emission-reduction technologies in the context of smart and low-carbon city construction, Sustainable Cities Soc. 89 (2023) 104322.

Dr. Carlo Centofanti received his Ph.D. in Information and Communication Technology from the Department of Information Engineering, Computer Science, and Mathematics at the University of L'Aquila, Italy. He earned his M.Sc. degree in Computer Science Engineering and his Bachelor's degree in ICT Engineering from the same university. His research areas include Edge Computing, Software Defined Networking, and Network Slicing. He has actively contributed to ECSEL-RIA AfarCloud Project (GA: 783221), and is actively contributing to MSCA-RISE OPTIMIST Project (GA: 872866), and SNS-SEASON Project (GA: 101096120).

José Santos obtained his M.Sc. degree in Electrical and Computers Engineering in July 2015 from the University of Porto, Portugal. Recently, he completed his doctoral studies at Ghent University in April 2022. He is currently a Postdoctoral Researcher in the Internet Technology and Data Science Lab (IDLab) Research Group at Ghent University - imec, Belgium. His research interests include Cloud Computing, the Internet of Things (IoT), Container Scheduling and Auto-scaling, Service Function Chaining, and Reinforcement Learning. His work has been published in more than 20 scientific publications. He received the Ph.D. Excellence award 2022 from imec (Belgium) and the Best Dissertation Award at NOMS 2023 (Miami, USA).

Venkateswarlu Gudepu (Graduate Student Member, IEEE) received his B.Tech degree in computer science from Rajiv Gandhi University of Knowledge Technologies - Nuzvid (RGUKT-Nuzvid) , India in 2017, and the M.Tech in Artificial Intelligence (AI) from National Institute of Technology (NIT - Uttarakhand), India. He is currently pursuing the Ph.D. in computer science and engineering with the Indian Institute of Technology - Dharwad (IIT Dharwad), India. His area of research interest includes 5G and Beyond technology, AI/ML for Networks, O-RAN standards implementation, and Energy sustainability for B5G Networks.

Koteswararao Kondepu (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from the Institute for Advanced Studies Lucca (IMT), Italy, in July 2012. He is currently an Assistant Professor with the Indian Institute of Technology Dharwad, Dharwad, India. His research interests include 5G, optical networks design, energy-efficient schemes in communication networks, and sparse sensor networks.