UNIVERSITEIT VAN AMSTERDAM

Characterization and Modelling of Resource Usage and Energy Consumption in HPC Datacenters by Machine Learning

Master Thesis

Jianyang Gu (2720655)

1st Supervisor: Prof.dr.ir. Alexandru Iosup
Daily Supervisor: Phd Fellow Xiaoyu Chu, M.Sc.
Second Reader: Dr.ir. Animesh Trivedi

August 2023

Abstract

With the ascent of Information and Communication Technology (ICT), the complexity and scale of High-Performance Computing (HPC) and data centers have surged, leading to pressing challenges in energy and resource consumption. These data centers, fundamental to our digital experiences and the backbone of our digital age, require profound understanding and optimization to align with our vision of a sustainable and human-centric technological future. However, existing research reveals gaps in this understanding, often presenting either overly generalized or overly narrow insights. There is an urgent need for state-of-the-art modeling tailored to contemporary, large-scale data center infrastructures.

In this study, we comprehensively characterize and model the resource and energy consumption patterns from the LISA data center dataset. While addressing a noticeable void in existing literature, our detailed characterization spans from a broad overview down to node-specific insights with a focus on CPU utilization and power usage. We give particular emphasis on the often-neglected rack level, differentiating between generic and ML-centric facets. Also, we analyze correlations across matrices and conduct essential pattern and peak analysis, identifying three predominant recurring patterns. Further, by designing state-of-the-art LSTM and Transformer models, we significantly enhance our estimation accuracy, registering a remarkable improvement of over 80% when compared to current statistical methods. When forecasting four steps ahead, both LSTM and Transformer models exhibit good performance, recording approximately 5% relative error for power usage, 2% for temperature, and RMSE values between 3.46 and 3.8 for CPU utilization. We also compare LSTM and the Transformer within the realm of HPC data centers, offering insights into their best-suited applications. Our primary contributions lie in the granularity of our characterization, the application of advanced ML models to data centers, and a comprehensive evaluation of these models for improved sustainability and efficiency in HPC operations.

Acknowledgements

I would like to extend my deepest gratitude to Professor Alexandru Iosup for his invaluable guidance, unwavering support, and insightful feedback throughout this research journey. Your expertise and vision have been instrumental in shaping this work.

A special mention to my daily advisor, Xiaoyu Chu, for her continuous mentorship, patience, and dedication. Your hands-on advice and constructive critiques have been pivotal in ensuring the rigorousness and quality of this study.

Furthermore, I wish to acknowledge the many individuals and resources that have contributed to the successful completion of this project. Their collective wisdom, encouragement, and constructive criticisms have enriched the content and provided the impetus needed to persevere.

To all, I express my heartfelt thanks.

Contents

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In an age of unprecedented technological evolution, our society stands at the crossroads of transformative change. This metamorphosis, which underscores the symbiotic relationship between humans and technology, is anchored in the realm of ICT [33]. The vision of a world where individuals and human-centered organizations are enriched by an automated, sustainable layer of technology is not just a distant dream but a tangible reality we're weaving. ICT has become an indispensable infrastructure in modern society, affecting various aspects of our lives and transforming the way we live, work, and communicate [34]. At the heart of our digital world are data centers [11]. These places, filled with many servers and networks, power our online experiences, from phone apps to major research in various fields [70]. Yet, as indispensable as they are, these data centers present a constellation of challenges, predominantly in terms of resource and energy consumption. HPC and data centers provide the necessary infrastructure for many ICT-based applications and services, and the growth of ICT drives the development and expansion of HPC and data centers. As the demand for ICT-based services continues to grow, it is expected that HPC and data centers will play a crucial role in enabling and shaping the future of ICT [4].

The complexity of data centers has been increasing due to fast-growing data demands and workloads, along with more complex and diverse hardware, software, resources, and services [1]. Furthermore, the scale of the data center, in the US market alone, demand—measured by power consumption to reflect the number of servers a data center can house—is expected to double by 2030, up from 17 TW in 2022 [47]. Consequently, data centers need to expand their storage, architecture, and processing capabilities, leading to a rise in energy consumption and an increase in data center construction [61]. There is a growing focus on sustainability, with data centers adopting more efficient cooling systems, energy-efficient hardware, and renewable energy sources [32], adding to the overall complexity and scale of data center operations.

In this work, we investigate characterization and modeling for resource usage and energy consumption both for estimation and prediction. Resource and energy modeling in HPC data centers facilitates informed decisions regarding resource allocation, reducing capital and operational expenses. Moreover, it enables data center operators to optimize cooling systems, power distribution, and server utilization, thereby improving energy efficiency and reducing carbon emissions [9]. These predictions play a key role in preventing service interruptions [29]. Additionally, they ensure that workloads are distributed optimally for consistent performance [40].

Optimizing the performance and efficiency of data centers is paramount, not just for operational excellence, but also in our pursuit of a sustainable technological future. As we envision a world where technology operates seamlessly as utilities and services under human guidance, the emphasis shifts to ensuring these data centers are not voracious energy consumers but optimized, efficient, and predictable entities. This necessitates a profound understanding of their dynamics, from the nuances of energy and resource characterization to the intricacies of machine learning models that can predict their operational metrics.

It's within this context that this thesis endeavors to contribute. By delving deep into the operational matrices of data centers, identifying patterns of consumption, designing robust machine

learning models for estimation and prediction, and evaluating their prowess, this research hopes to illuminate the path toward data centers that resonate with our grand vision. The journey ahead is both challenging and promising. It is a pursuit not just of technological excellence but also of crafting a world where ICT stands as a beacon of human progress, a fundamental right, and a testament to our collective aspirations.

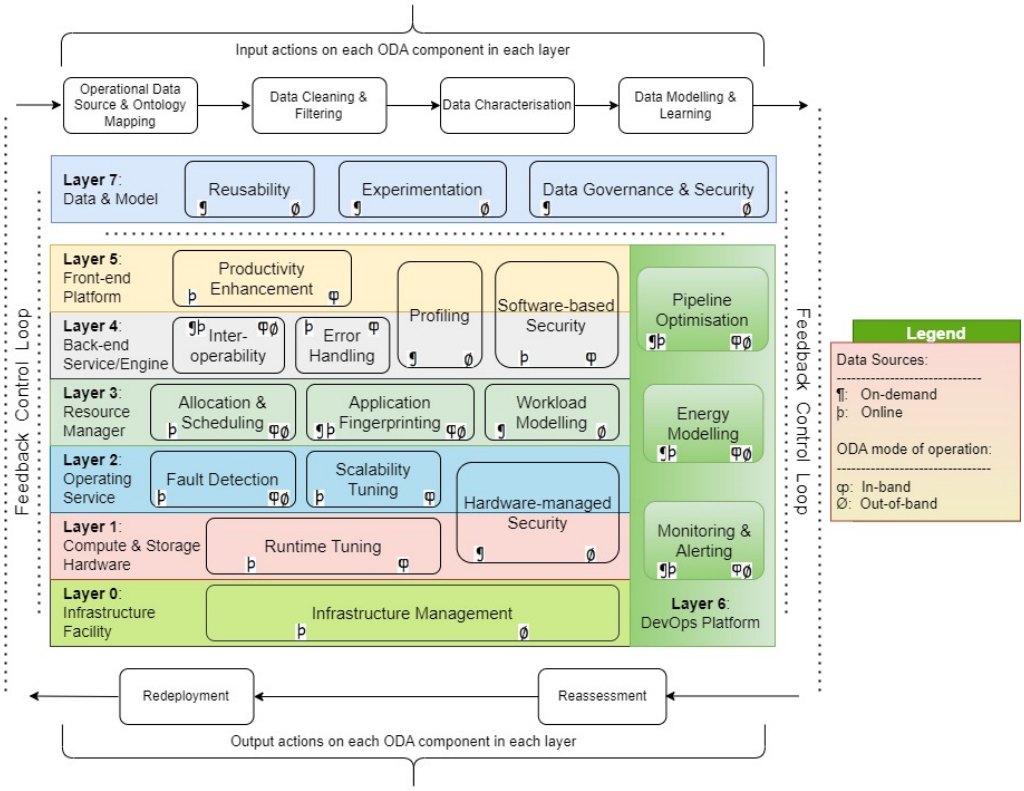


Figure 1.1: Reference architecture for the ODA framework (Source: Suman et al, 2023).

The reference architecture for the ODA framework is illustrated in Figure 1.1. Inputs for individual ODA components across various layers are presented in the figure's upper section. Our research aligns with the segments of Data Characterization and Data Modeling & Learning. Within this framework, the modeling of resources aligns with Layer 3, specifically the Resource Manager, while energy modeling corresponds to Layer 6, dedicated to DevOps Platforms. Characterization serves as the bedrock upon which sustainable and efficient machine-learning models are built for data centers. When we speak of the intricacies and idiosyncrasies of data center operations, it's imperative to recognize that these infrastructures don't operate in isolation; they are an interplay of hardware, software, environmental factors, and human interventions. As such, effective characterization unravels the multifaceted layers of these operations. This granularity offers invaluable foresight, allowing for the anticipation of potential bottlenecks, ensuring resilience against unforeseen challenges, and charting out strategies for seamless scalability. In the era of green computing, where sustainability is paramount, characterization assists in pinpointing the pattern and peaks [39]. Further, machine learning is essential for modeling energy consumption and resource consumption in data centers due to its ability to capture complex patterns and relationships in large-scale, dynamic, and multi-dimensional data [14]. Traditional statistical methods may not be able to accurately model such intricate relationships, while machine learning algorithms can adapt to changes in data center workloads, topology, and other factors. This paves the way for precision-driven predictions, facilitating optimized resource allocation, minimized energy expenditure, and substantial cost benefits for data center operators. Employing advanced analytics, machine learn-

ing, and artificial intelligence techniques, data center operators can accurately predict resource and energy demands, leading to more sustainable and efficient operations. In sum, while machine learning offers predictive prowess, characterization provides clarity and direction, making them indispensable partners in the journey toward next-generation data centers.

1.1 Problem Statement

In an age of unprecedented technological evolution, our society stands at the crossroads of transformative change. This metamorphosis underscores the symbiotic relationship between humans and technology, deeply entrenched in the realm of ICT. The vision of a world where individuals and human-centered organizations are enriched by an automated, sustainable layer of technology is no longer a distant dream but a tangible reality we are weaving.

Central to this vision are data centers, the bedrock of our digital age. These hubs enable the myriad of digital experiences we've come to depend on, from the applications on our phones to computational analyses driving groundbreaking advancements. As essential as they are, these data centers also house a myriad of challenges, notably in resource and energy consumption.

Optimizing the performance and efficiency of data centers is not just about enhancing their operational capacity. It is a commitment to creating a sustainable technological future, aligning with our broader vision of a world where technology serves humanity seamlessly. This implies that data centers must evolve from being mere energy consumers to entities that are optimized, efficient, and inherently predictable. To achieve this, there's an imperative to grasp their dynamics fully, from understanding energy and resource characterization to using machine learning models that can foretell their operational patterns.

However, despite the progress made in understanding data centers, noticeable gaps persist in current research methodologies. Our deep dive into existing literature reveals a pattern of either excessive generalization or extreme specificity. The broader perspective, though essential, often overshadows the equally vital micro level, especially the rack level. This missing link between macro and micro perspectives signifies a pressing need for a holistic research framework that encapsulates the complete data center dynamic.

Moreover, recognizing patterns in energy and resource consumption is important for preempting high-demand scenarios. Temporary surges, though transient, can overburden data center resources and skyrocket energy consumption. Yet, there's a palpable lack of exhaustive research on pattern and peak analysis for these matrices.

Another challenge is the continued reliance on static datasets and traditional models. This approach, anchored in past paradigms, fails to resonate with the dynamic essence of contemporary data center operations. To keep pace with the rapid fluctuations and intricacies characteristic of today's data centers, there's an undeniable need for models that are both responsive and predictive. Machine Learning models, known for their dynamic and adaptive nature, can bridge this gap. They offer a forward-looking approach, enabling proactive decision-making by recognizing emerging patterns and making timely predictions.

In our research, we distinctly highlight the following challenges that pervade the field:

1. Lack of a comprehensive multi-level approach that balances the macro and micro perspectives for energy and resource, especially the absence of focused research on the crucial aspects of pattern and temporary analysis in data center operations, which holds a key to enhanced resource optimization during high-demand situations.

2. Reliance on traditional statistical models that may fail to capture the dynamic nature of current data center operations and the challenges they present. Such an approach, tethered to its past paradigms, is strikingly incongruent with the evolving dynamism intrinsic to contemporary data center operations. The intricacies and rapid fluctuations characteristic of today's data centers necessitate models and data sources that are not only responsive but also predictive. To bridge the gap left by static datasets and traditional models, we advocate the use of Machine Learning models. These models, inherently dynamic and adaptive, are

well-suited to grasp the complex, ever-evolving nature of data center operations. ML models can continuously learn from the data, recognizing emerging patterns, and making predictions that facilitate proactive decision-making. The incorporation of ML models transforms the operational paradigm from a reactive stance to a proactive, forward-looking approach.

3. Prioritize and implement strategies that drive energy efficiency in data centers. By aligning operational optimization with sustainable practices, machine learning could help pave the way for data centers that are both high-performing and environmentally friendly.

Tackling these problems enables us to provide a nuanced understanding of data center dynamics, laying the foundation for data-driven strategies to improve energy efficiency, resource allocation, and overall system resilience. This positions data centers to better navigate the challenges of modern-day demands while prioritizing sustainability and operational excellence.

1.2 Research Questions

This work focuses on characterizing and modeling Resource Usage and Energy consumption using machine learning including estimation and prediction. Our main research question is **How to characterize and model resource use and energy consumption in large-scale data center infrastructures?** To clarify the direction of this study, we elaborate on three research questions(RQs) as follows:

- **RQ1: How to characterize and analyse resource and energy in HPC data center?**
  This research question aims to unravel resource and energy characterization with a focus on CPU utilization and power usage, both crucial for subsequent analytical and predictive endeavors. The inherent challenge here lies in achieving a detailed and precise characterization of resource and energy consumption patterns in the dynamic landscape of data center operations.

  Proper characterization and analysis of resource and energy usage pave the way for efficient data center management. Addressing this question ensures that computational resources are used effectively, resulting in cost savings, reduced energy consumption, and enhanced operational efficiency. This forms the foundation for all subsequent optimizations and predictive strategies.

- **RQ2: How to design an accurate and efficient ML model for data center resource and energy estimation and prediction?**
  Venturing into the architecture and design of ML models tailored to the specific demands of data center operations, this research question confronts the challenge of developing models that are accurate, adaptable, and efficient. The ever-evolving nature of data centers adds complexity, requiring models that address challenges like latency, scalability, and real-time adaptability.

  An optimal ML model tailored for data centers has the potential to forecast computational needs, thereby preempting system failures and inefficiencies. Predictive capabilities can lead to smoother operations, fewer outages, and more streamlined energy consumption. Addressing this question aids in navigating the complexities of modern data centers by offering proactive solutions rather than reactive fixes.

- **RQ3: How to evaluate the ML models to understand resource use and energy consumption?**
  Central to this research question is the formulation of apt evaluation metrics and methodologies tailored for ML models in the context of data center dynamics. The challenge here transcends mere accuracy metrics, diving into model adaptability, and the ability to offer

actionable insights. Crafting a reliable and relevant evaluation framework that can holistically assess a model's performance within the specificities of vast computing infrastructures is of paramount importance.

Evaluating ML models goes beyond mere accuracy; it determines the applicability and relevance of the model in real-world scenarios. Understanding how a model performs under varying conditions, and if it provides actionable insights, is crucial for operational success. Addressing this question ensures that the developed models align with the practicalities and intricacies of data center environments.

1.3 Main Contributions

Our research contributions (Cs) are threefold:

- **C1:** We provide a comprehensive characterization of resource and energy consumption patterns, extending from a holistic overview to node-specific insights. This characterization not only spans broad distributions but also delves deep into the often-neglected rack level, spotlighting peak insights, generic and ML-centric facets, and unveiling three dominant recurring patterns.
- **C2:** We design LSTM and Transformer models for both estimation and prediction modeling of CPU utilization and power consumption in HPC clusters. For estimation, they achieve over 80% enhancement in estimation accuracy compared to current statistical methods. For 4 steps ahead prediction, both models achieve approximately 5% relative error for power usage, 2% for temperature, and an RMSE value range between 3.46 and 3.8 for CPU utilization.
- **C3:** We offer a comprehensive evaluation and comparison between LSTM and Transformer within the context of HPC data centers. This provides insights into the best-suited applications of each model, laying a foundation for improved sustainability and operational efficiency in high-performance computing environments.
- **Original Contribution:**
  1. **Comprehensive characterization on multi-level, fine-grained dataset:** We delve into an in-depth study on the dataset which is unique due to its detailed granularity and multi-layered hierarchy. Such intricate data holds the potential to unlock sophisticated patterns and insights that are typically overlooked in less refined datasets. This depth and breadth lay the foundation for more robust modeling and insightful characterizations.
  2. **Analysis spanning multiple levels (node, rack, overview):** We carry out characterization and analysis across multiple levels, offering insights from each distinct level. At the node level, we investigate individual computational components. At the rack level, we grasp the synergy between nodes. The overview provides a macroscopic perspective, ensuring holistic insights.
  3. **Employment of frequency domain analysis for enhanced pattern detection:** We integrate frequency domain analysis to identify recurring patterns in HPC data centers, detecting underlying periodicities fundamental to understanding HPC behavior.
  4. **ML model comparison and insights into HPC data center:** By leveraging machine learning models and assessing their efficacy against practical metrics in the HPC data center landscape, we illuminate their precise advantages and areas of strength. This analytical process provides pivotal insights into optimizing these models specifically for high-performance computing infrastructures, underscoring their potential to significantly enhance operations and efficiency in the HPC data center domain.

1.4 Societal Relevance

In today's digitized era, data centers have become the heart of our global infrastructure, powering everything from cloud storage solutions to complex simulations that drive scientific advancements. With the increasing demand for computational power, there's a growing strain on our energy and resources, making the efficient and sustainable operation of these data centers paramount. This research not only advances our understanding of the intricacies of power usage and CPU utilization in high-performance computing environments but also introduces machine-learning methodologies that promise optimized operations. As a result, our work has broad societal implications. By enhancing the efficiency of HPC data centers, we not only reduce their environmental footprint but also potentially reduce operational costs, translating to economic benefits.

1.5 Thesis Structure

The structure of this thesis is outlined as follows: Chapter 2 and 3 provide background information and reviews related work. Chapter 4 addresses RQ1 through our characterization studies. For RQ5, our model design is detailed in Chapter 4. Chapter 6 showcases our experimental results and associated discussions. Finally, Chapter 7 offers concluding remarks.

1.6 Statement of Independent Work

I hereby declare that this thesis is entirely my own work. All information derived from other sources has been duly cited and referenced. The work has also not been submitted elsewhere for assessment.

1.7 Open Science

All codes for this work are open and available. Available at https://github.com/MushroomGu/Thesis/tree/main

Chapter 2

Background

In order to characterize and model the data for ODA in HPC, we first elucidate the fundamentals of HPC — encompassing its computing architecture, structural hierarchy, and the range of components and technologies it interfaces with. Subsequently, we delve into the intricacies of understanding HPC behavior and the approaches to the characterization and modeling of ODA.

2.1 What is HPC?

HPC refers to the practice of aggregating computing power in a way that delivers significantly higher performance than one could get from a typical desktop computer or workstation in order to solve large computational problems in science, engineering, or business [27]. Parallel computing in HPC enables HPC clusters to handle sizable workloads, partitioning them into discrete computational tasks that can run concurrently.

HPC systems are evolving. As the computational demands for tasks such as simulations, modeling, and advanced analytics escalate, it's becoming crucial to supplement traditional CPU-based processing with specialized accelerators. These accelerators, adept at performing specific computations more efficiently than general-purpose CPUs, significantly enhance overall system performance. The practice of integrating accelerators, such as Graphics Processing Units (GPUs), Field-Programmable Gate Arrays (FPGAs), or even custom-designed Application-Specific Integrated Circuits (ASICs), with CPUs is gaining traction in HPC, especially in domains like artificial intelligence, data analytics, and scientific computing. While the integration of diverse computational resources introduces complexity in programming and managing HPC systems, it also creates opportunities for substantial improvements in performance, energy efficiency, and cost-effectiveness. Utilizing the right combination of CPUs and accelerators and employing software tools and frameworks that support heterogeneous computing can effectively equip HPC systems to manage the most demanding computational tasks today. The 2000s saw the emergence of commodity clusters – groups of interconnected, commodity computers that work together as a single system – as the dominant HPC architecture, due to their cost-effectiveness and scalability. In the 2010s and beyond, accelerators like GPUs became increasingly important for HPC, driven by the needs of applications like machine learning [48][46]. At the same time, cloud computing emerged as a viable alternative for some HPC workloads, offering flexibility and eliminating the need for upfront capital investments [46].

2.1.1 Computing Architectures in HPC

HPC leverages various computing architectures to perform complex computational tasks more efficiently. These architectures have specific characteristics and use cases. In our case for the data centers, cluster computing, and distributed computing emerge as the predominant architectures

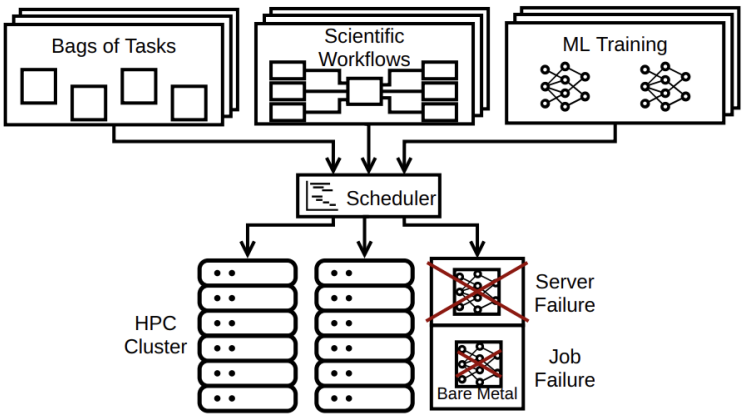


Figure 2.1: Architecture of the HPC [16].

due to their aptness for handling vast amounts of data and computations while ensuring scalability and reliability.

Cluster Computing

In cluster computing, a group of computers, or nodes, work together as if they were a single entity. Typically, these nodes are standard off-the-shelf servers that are interconnected via a high-speed local area network (LAN). In a cluster computing setup, each node typically runs its own operating system and has its own memory and processing power. Each node operates under a similar hardware and software environment, creating a homogeneous network. They are interconnected through a high-speed network that allows them to communicate and share resources effectively. Cluster computing is often used when high computational power is required, and it offers advantages such as high availability, scalability, and cost-effectiveness.

Distributed Computing

Distributed computing is a model where components of a software system are shared among multiple computers to improve efficiency and performance. In distributed computing, the workload is divided into smaller subtasks that are executed concurrently on different machines, and the results are combined to achieve the desired outcome. Each processor has private memory. The processors communicate by sending messages across a network. The primary objective of distributed computing is to harness the combined computing power and resources of multiple machines to tackle complex problems more efficiently than a single machine could. By distributing the workload, tasks can be processed in parallel, leading to faster execution and improved scalability. This architecture is more scalable than shared memory and is often used in large-scale HPC systems. Distributed computing systems can take various forms like Client-Server Model, Peer-to-Peer Model, and Message Passing Model, depending on the architecture and communication model used.

2.1.2 System Model of HPC

In this paper, we are talking about the HPC system model which uses a heterogeneous architecture that includes both traditional CPUs and accelerators. Interconnect technologies continue to evolve, and software tools are constantly improving to help users effectively exploit these complex systems. The hierarchy of HPC in the academic domain can be represented through various layers of abstraction, beginning with computational tasks at the highest level to the actual hardware infrastructure at the lowest level.

Figure 2.1 demonstrates the hierarchy of HPC. In this hierarchy, users submit their jobs to the scheduler, specifying their resource requirements and dependencies. The scheduler then decides when and where (on which cluster) to run the jobs based on these inputs and the current state of the system. We dive deep into each component:

Bags of Tasks

Bags of tasks referred to as parallel tasks, denote a set of tasks that are independent of each other, allowing them to be executed in parallel without any inter-task communication. This property greatly simplifies parallel programming and scheduler design.

Scientific Workflows

Scientific workflows represent complex scientific processes as a sequence of computational tasks, with data dependencies between them. These workflows often include data pre-processing, simulation or modeling, and post-processing or analysis steps. For example, in a climate modeling workflow, initial tasks might preprocess satellite data, followed by tasks that run a climate model, and finally, tasks that analyze and visualize the results. Workflow management systems like Pegasus and Taverna are often used to orchestrate these tasks on HPC resources [49].

ML Training

Machine Learning (ML) training involves learning a model from data. This often involves iterative algorithms, where each iteration requires processing the entire dataset. For large models and datasets, this can be computationally expensive. ML training tasks can be parallelized in various ways, for instance, data parallelism involves dividing the dataset across multiple workers, whereas model parallelism involves dividing the model. Deep learning frameworks like TensorFlow and PyTorch provide support for parallel and distributed training [38].

Scheduler

Schedulers are crucial to managing and utilizing HPC resources efficiently. They accept job submissions from users, decide when and where to run these jobs, and manage their execution.

Schedulers must make these decisions based on a variety of factors, such as job requirements (e.g., number of cores, memory), current system state (e.g., available resources), and scheduling policies (e.g., priority levels, fairness, utilization). For example, a scheduler might delay a large job to avoid fragmenting resources and thus be able to start more small jobs.

Commonly used schedulers in HPC include Slurm, PBS Pro, and HTCondor. Slurm, for example, supports a variety of scheduling policies, advanced reservation of resources, and elasticity (adding/removing resources on the fly).

Clusters

Clusters are the workhorses of HPC. They consist of multiple racks and nodes (servers), each with one or more processors (each with multiple cores), memory, and often local storage. Nodes are interconnected by a network, allowing them to exchange data and work together on parallel tasks.

HPC clusters often include different types of nodes for different workloads. For instance, there might be CPU nodes for general-purpose tasks, large memory nodes for memory-intensive tasks, GPU nodes for tasks that can exploit GPU parallelism (like deep learning), and high-throughput nodes for tasks that require fast I/O.

One of the largest and most powerful HPC clusters is the Fugaku supercomputer in Japan. It has over 150,000 nodes, each with a 48-core Arm-based CPU, and a high-speed, low-latency network [72].

2.1.3 Nodes in HPC

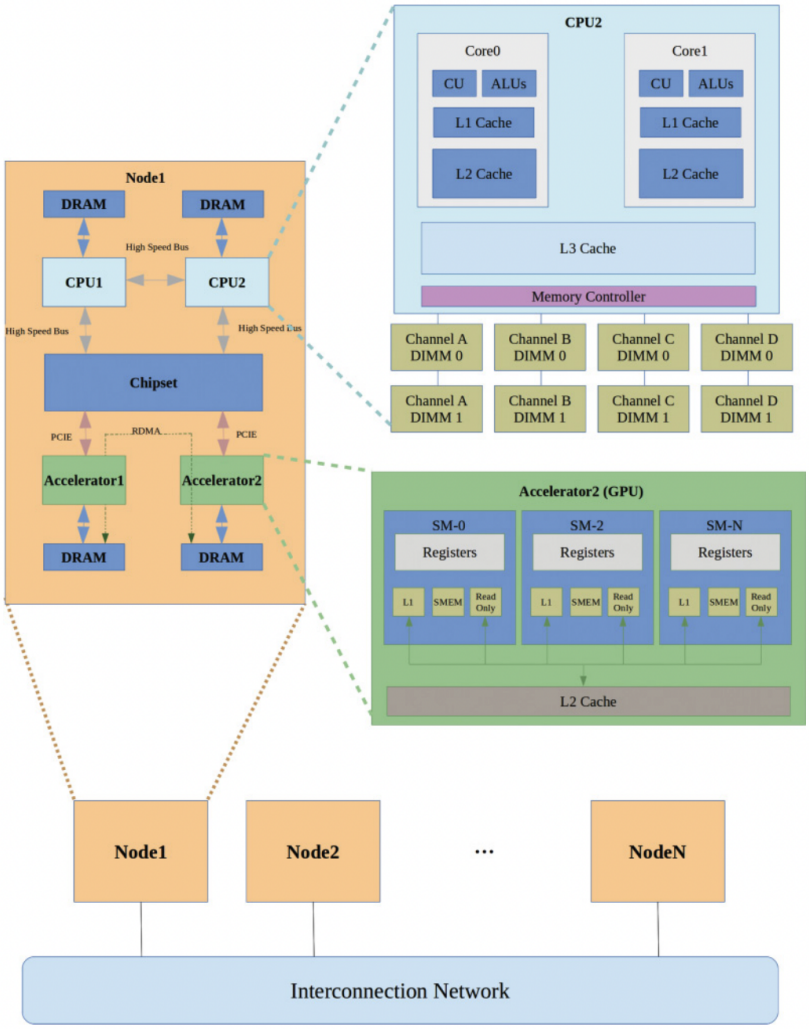


Figure 2.2: Nodes architecture in HPC [16].

In the current era, the design and implementation of high-performance computing systems lean heavily toward multicore CPU architectures. These systems are fundamentally composed of nodes wherein each node harnesses the potential of several cores within a CPU. The integration of multicore CPUs with advanced accelerators, such as Graphics Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs), forms the backbone of this high-performance computational setup [50].

The architecture of these nodes is critical for maximizing computational efficiency and throughput. The various cores of a CPU, complemented by the accelerators, work collectively to compute. The accelerators specialize in handling specific types of computations, enabling them to process those much faster than a general-purpose CPU core. For instance, GPUs, with their hundreds of cores, are particularly well-suited for computations involving large arrays of data, as often found in graphics rendering and scientific simulations.

The tight integration between multicore CPUs and these accelerators is vital. Through smart workload distribution - delegating tasks that suit the accelerators while keeping other tasks on the CPU cores - these HPC systems are capable of significantly accelerating a broad spectrum of scientific applications. This synergistic coupling is key to the computational prowess of today's

high-performance scientific computing systems.

As shown in Figure 2.2, the architecture within an individual node in an HPC system is a testament to the complexity and sophistication involved in high-performance computing. The interaction between CPUs and various accelerators reinforces how the architectural design of these nodes contributes to the remarkable computational capabilities of HPC systems.

HPC systems employ nodes that are connected via a fast interconnect, wherein each node houses multiple processors within a shared memory space. A node's components are efficiently managed by a chipset.

Each node comprises CPUs, and each of these CPUs is designed with several cores. These cores are designed to share resources, and each CPU has multiple levels of cache memory. The configuration is such that smaller, faster cache memory units are located closer to the core, while larger, slightly slower cache memory units are shared among the cores. Interactions between the cores of a CPU and the memory managed by another CPU take place via a high-speed on-chip interconnect. Each CPU is designed with an integrated memory controller. This design uses multiple channels to enhance memory bandwidth, and it leverages multiple dual in-line memory modules (DIMMs) to increase the capacity per channel.

Moreover, the CPUs are connected to one or more accelerators - which have limited memory - through a lower-bandwidth communication link. These accelerators have distinct architectures, each with its own hierarchical memory structures. As one can see, the node architecture has become highly heterogeneous and hierarchical [56].

2.2 To Understand HPC Datacenter Behaviors: Operational Data Analysis

As stated above in Section 2.1.2, HPC environments are complex and hierarchical systems that are often characterized by a multitude of interconnected nodes, each with multicore CPUs that are tightly integrated with specialized accelerators. The performance and efficient utilization of these systems is a function of numerous factors, including system architecture, application design, workload characteristics, scheduling policies, and user behavior, among others.

To navigate this complexity and gain insights into the behavior of HPC systems, we need to delve into Operational Data Analysis(ODA). Bourassa et al [5] define ODA as a mechanism to analyze the operation of an HPC system to gain insight into its behavior. This involves the systematic collection and examination of system logs, performance metrics, and other types of operational data generated during the system's operation. This kind of analysis is crucial for both the optimal functioning of HPC systems and the effective execution of computational tasks.

Monitoring frameworks exist that collect data from a multitude of sensors distributed across the infrastructure, extending to the granular level of individual computing nodes. These sensors are embedded within both hardware and software units, capturing detailed operational data. For instance, CPU and memory usage data can reveal hotspots or imbalances in the computational load across different nodes or cores, while network traffic data (such as the count of TCP segments sent or received by a node) can shed light on communication patterns or bottlenecks. Scheduler logs can offer insights into job wait times, turnaround times, or resource allocation efficiency, and application performance data can highlight issues related to scalability, parallel efficiency, or input/output behavior.

By integrating and analyzing these diverse sources of operational data, we can obtain a holistic and detailed view of system behavior, identify performance issues or inefficiencies, and guide system tuning or workload optimization efforts. Furthermore, this data-driven approach can also support predictive modeling or anomaly detection, enabling proactive system management and fault mitigation.

The application of ODA in HPC can be customized to suit various operational levels based on the unique requirements of a particular site. The categorization can be delineated across three tiers: the Data Center Level, the HPC System Level, and the Node Level. Additionally, we

introduce the OpenDC, a simulation platform for data centers, in the subsequent discussion.

2.2.1 Data Center Level

When applied at the data center level, ODA can contribute significantly to optimizing the functionality of infrastructure and systems that span the entire facility, which includes areas such as cooling, communication, and power distribution systems [52]. In addition, ODA can assist in diagnosing and addressing various system issues.

For example, Jiang et al. [31] present a strategy predicated on warm water cooling, proposing an adaptive cooling control framework that responds dynamically to workload fluctuations. They construct this strategy on the basis of an in-depth analysis of real-world cluster traces from industry-leading firms, Google and Alibaba. Their approach seeks to enhance the economic efficiency of water cooling systems within data centers, leveraging ODA at the data center level to achieve this goal. This rigorous methodology substantiates their findings, which have promising implications for contemporary data center operations.

Grant et al. [21] offers an innovative method for examining and quantifying network contention effects stemming from interference, also drawing on ODA data from data centers. Their methodology utilizes a continuously running benchmark application and the deployment of network performance counters, providing a precise, ongoing assessment of network interference impacts. This approach underscores the vital role of data center-level ODA in assessing network performance, thereby contributing to more effective data center management strategies.

Both studies show the importance of data center-level ODA in optimizing operations and improving efficiency, demonstrating the immense potential of optimizing data center management.

2.2.2 HPC System Level

At the level of the HPC system, ODA can serve to enhance resource utilization, energy efficiency, and quality of service. This could involve the deployment of sophisticated scheduling mechanisms for optimal user job placement, often utilizing supplemental system data, such as energy budgets, thermal constraints, or I/O characteristics, to function within defined parameters [52].

For example, Imes et al. [26] presents a methodology for predicting the most advantageous resource configurations for augmenting energy efficiency during application execution. This approach is informed by low-level hardware performance counters, gathered from HPC ODA. The proposed method is capable of dynamically adapting to application behavior changes, providing a highly adaptable and responsive resource management solution, thereby underlining the value of HPC-level ODA.

Verma et al. [68] delve into the application of power management techniques for high-performance computations on modern, energy-efficient servers, equipped with virtualization support. Their focus is specifically on power management strategies involving dynamic consolidation and the employment of dynamic power range. These strategies are enabled by servers' low power states, based on HPC-level ODA for effective power management in high-performance applications.

2.2.3 Node Level

At the compute node level, ODA can substantially enhance energy and resource efficiency and system reliability [52]. This can be facilitated by the implementation of runtime systems, which are designed with the capacity to dynamically adapt system parameters, such as CPU speed and frequency, in response to the behavioral patterns of both hardware and applications from the node-level ODA information. Moreover, it can be used to detect anomaly behaviors and also help with failures happening in the system.

For example, by utilizing node-level resource consolidation, Lin et al. [45] address the challenge of server pool power management by leveraging reinforcement learning techniques.

Guan et al. [23] introduces an adaptive anomaly identification strategy that, by using node-level ODA data, examines the principal components most relevant to various failure types within

cloud computing infrastructures. The strategy integrates the analysis of cloud performance metrics with filtering techniques, facilitating automated, efficient, and precise anomaly detection.

2.2.4 OpenDC

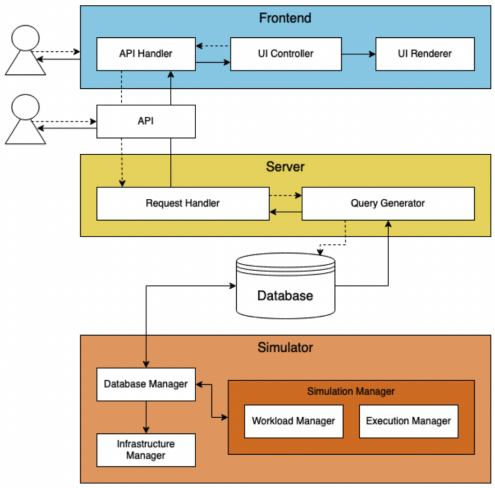


Figure 2.3: OpenDC architecture [37].

OpenDC is a data center simulation platform developed by AtLarge Research. It provides collaborative online data center modeling, diverse and effective DC simulation, and exploratory data center performance feedback. It serves a dual purpose: firstly, it aims to facilitate cloud computing education by providing an open-source platform for learning and experimentation. Secondly, it aims to support research endeavors focused on data centers by offering a comprehensive environment for studying and analyzing various aspects of data center operations.

The architecture of OpenDC consists of four primary components, illustrated in Figure 2.3: a frontend, a web server, a database, and a simulator. To conduct a simulation using OpenDC, two essential elements are required: a topology and a workload trace. The topology specifies the layout of the data center being modeled, including various rooms with racks and machines. Machines are described in terms of RAM and CPU models. The front end interacts with the web server, which processes the received requests, such as experiment specifications, and stores them in the database. The simulator monitors the database, simulates the experiments using the provided topology and workload, and writes the results back to the database. These results can then be accessed and retrieved by the web server.

In summary, OpenDC offers a comprehensive data center simulation platform with a user-friendly front end, a web server for request processing, a database for experiment storage, and a simulator for conducting simulations. Its flexibility and capabilities make it suitable for both cloud computing education and research purposes, empowering users to explore and gain insights into data center operations.

2.3 Data Characterization: Dive Deep Into System Behaviors in HPC

Data characterization, a process that portrays the critical features and attributes of a data set, is a pivotal step in ODA. This in-depth analytical process not only provides a comprehensive overview

of the data but also exposes its strengths, weaknesses, and underlying patterns or relationships that might otherwise go unnoticed. The tasks associated with data characterization are manifold and can involve identifying the types of data present, determining the distribution of this data, and unearthing correlations, changes, and patterns among various data elements.

Data characterization provides a comprehensive understanding of the operational data within the HPC system. It helps researchers gain insights into the nature of the data, its properties, and its behavior, allowing them to make informed decisions during the analysis process. Moreover, By characterizing the operational data, researchers can identify potential bottlenecks, anomalies, or inefficiencies that may affect the performance of the HPC system. This understanding enables them to optimize the system by implementing measures to improve efficiency, resource allocation, and overall performance. It further helps in detecting and mitigating issues that may arise in the HPC system. By analyzing the data, researchers can identify anomalies or abnormal patterns that may indicate problems or potential failures. Early detection allows for timely troubleshooting and mitigation strategies to minimize system downtime and maintain reliable operation, and improve the performance of the HPC system.

In our case within the ODA context, data characterization serves as an essential pillar, aiding in the understanding of the nature of the energy and resource operational data being collected.

2.4 Modeling for ODA: Leveraging Characterized Data

Modeling in HPC systems for ODA involves the utilization of characterized data to develop models capable of interpreting and predicting system behaviors. In our context, modeling in HPC serves several key purposes:

1. Data Representation: it involves determining how operational data will be represented within the computational framework. This includes selecting appropriate data structures, formats, and representations that enable efficient storage, processing, and analysis of the data.

2. Algorithm Design: HPC modeling for ODA entails designing algorithms and computational methods tailored to the specific characteristics of the operational data. This involves developing mathematical models, statistical techniques, or machine-learning approaches for specific purposes.

3. Estimation and Prediction: Modeling in HPC for ODA often includes the estimation and prediction of system behavior based on the available operational data. By building computational models that capture the dynamics and relationships within the data, it becomes possible to estimate and simulate scenarios, predict future outcomes, or evaluate the impact of potential changes or interventions.

In this paper, we focus on the estimation and prediction of resources and energy.

2.4.1 Time Series Estimation

In our context working in HPC for ODA analysis, estimation refers to the process of making educated guesses or approximations about unknown values related to the operational data. Performance Estimation and resource Estimation are all common use cases. For example, Baig et.al [29] proposes an adaptive multi-methods approach that considers different scenarios encountered in a production data center and enables selecting the predictive method that learns best. The approach focuses on training estimation models using different methods and then selecting the one that will yield the best estimation result given the current scenario and the previous batch of collected data. [40] introduces the error preventive score (EPS) in time series forecasting models to improve estimation accuracy. The EPS analyzes the most recent estimations to capture a better result.

2.4.2 Resource and Energy Prediction

Resource prediction in HPC systems is a dynamic field, necessitating an in-depth understanding of computational demands. Accurate prediction facilitates efficient allocation, ensuring that computational resources are neither underutilized nor pushed to their limits. This, in turn, leads to enhanced job scheduling, with minimized wait times and maximized throughput, allowing systems to achieve optimal performance. Furthermore, predicting resource needs contributes to improved system reliability by preempting and avoiding potential resource clashes or overloads. Techniques employed for effective resource prediction span a range of statistical analyses, machine learning models, and simulations. Each offers unique insights, from deciphering patterns and trends to understanding intricate non-linear relationships and anticipating potential challenges through virtualized scenarios.

Parallel to resource prediction is the equally crucial domain of energy prediction. Given the immense computational capabilities and the consequent energy demands of HPC systems, predicting energy consumption isn't merely a cost-saving measure; it's an environmental imperative. Energy prediction seeks to provide precise estimates of power consumption, adapting to different workloads and configurations. It also aims to spotlight the primary energy sinks within the system, such as the power-hungry CPUs, memory units, and network interfaces. As with resource prediction, the methodologies for energy prediction are multifaceted. They encompass statistical methods, which analyze historical energy consumption data to inform future predictions, machine learning approaches tailored for energy use cases, and simulations that mirror real-world energy consumption scenarios, offering insights for system optimization.

Methodologically, both resource and energy prediction in the HPC realm is underpinned by several common techniques. Supervised learning techniques, for instance, harness labeled datasets to train models for future predictions. Reinforcement learning offers a more exploratory approach, optimizing system behaviors through iterative trial and error. Neural networks, with their ability to model intricate relationships, often find application in predicting complex system behaviors. Moreover, statistical models serve as foundational tools, leveraging historical data to extrapolate future patterns. Lastly, Graph-Based Models emerge as valuable tools, especially given the interconnected nature of HPC systems, offering insights into system-wide interactions and dependencies.

In the next chapter, we present related work regarding time series estimation and modeling for energy and resources.

Chapter 3

Survey of Related Work

In this chapter, we present a taxonomy and an analysis of the state-of-the-art in the field of resource and energy characterization, and modeling. We have identified and analyzed scientific articles on the topic of characterization and modeling for ODA in resource, energy. Further, we show the mapping work and comparative table for the modeling techniques. Part of this chapter was produced in collaboration with Wenjun Liang during our joint literature study work.

3.1 Data Characterization in HPC

In the exploration of characterization in HPC systems, several contributions have been made and mostly focus on energy consumption. Ozer et al. [55] initiated this exploration by employing Bayesian Gaussian mixture models, an unsupervised machine learning technique. This approach enabled the characterization of component performance and the detection of anomalies within HPC systems. The proposed model was assessed using real-world data from a production HPC system and demonstrated a capacity to accurately capture performance variations and identify anomalies.

Drawing on the broad theme of performance characterization, Qouneh et al. [57] took a different direction, focusing on the energy and performance profiling of heterogeneous servers with integrated general-purpose graphics processing units (GPGPUs) in virtualized environments. Their work bridged a gap in the research around cloud data centers, taking into account factors influencing CUDA performance, such as virtualization overhead and the size of serial code. Their work mirrors Ozer et al.'s endeavor to enhance system efficiency, albeit from a cloud computing perspective.

Building on these efficiency-oriented efforts, Bugbee et al. [7] introduced a more comprehensive methodology. By analyzing a sample of 10,000 jobs, classifying them, and employing frequency domain analysis and regression modeling, they aimed to predict mean power usage. This work extended the themes of performance characterization and efficiency optimization present in Ozer et al. and Qouneh et al. by introducing power-aware queue simulation for optimal power cap identification.

Following a similar path, Chinnici et al. [15] took a more specific path, examining the correlation between server workload power consumption and the relative number of cores used in Data Centers (DCs). Their analysis of real data from the ENEA-HPC DC facility illuminated the intricate relationship between these variables, pushing the envelope of current productivity metrics. This analysis, although distinctive in its focus on cores, aligns with the overarching themes of performance characterization, efficiency optimization, and power consumption found in the previous works. Collectively, these studies offer complementary insights that significantly contribute to the broader field of energy efficiency in HPC systems.

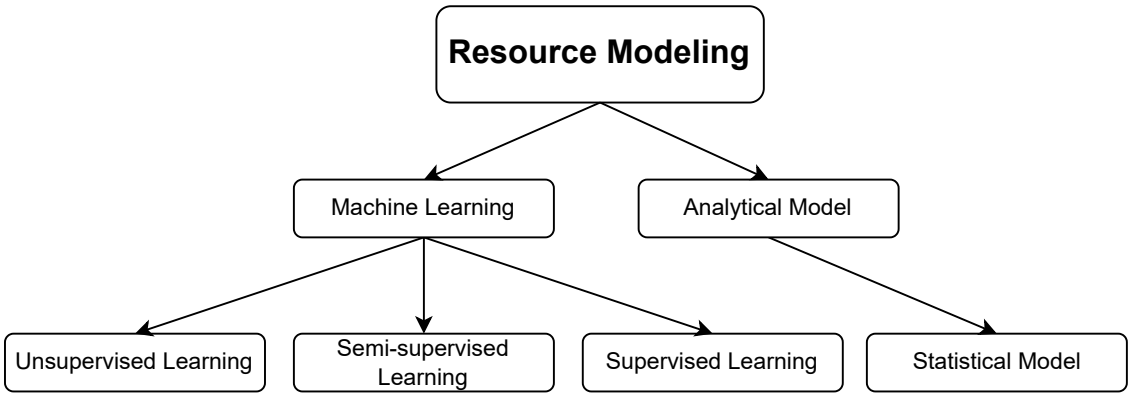


Figure 3.1: Taxonomy of resource modeling.

3.2 Taxonomy of Resource Modeling Techniques

Techniques leveraged in resource modeling encompass machine learning and analytical models, which are utilized to scrutinize and forecast an array of resource metrics such as CPU utilization, memory usage, I/O patterns, and network traffic. These models are pivotal in optimizing the employment of HPC resources and improving the efficiency of applications operating on expansive computing systems.

The taxonomy of resource modeling techniques is illustrated in Figure 3.1, in which we can observe that the Analytical Models category comprises Statistical Models. Meanwhile, the Machine Learning Models category incorporates not only Supervised Learning but also Semi-supervised and Unsupervised Learning as its primary techniques.

Based on our literature study, the acknowledgment of prediction-based resource allocation's importance within the sphere of memory-constrained and heterogeneous computing environments is on the rise. As we observe, this methodology's role in promoting efficient administrative practices in these complex technological ecosystems is becoming increasingly crucial.

For the prediction methods, more methods include the increased application of machine learning techniques including supervised and unsupervised learning for the development of more precise resource models, facilitating better prediction of resource requirements and optimal resource allocation. We additionally observe a growing trend in the application and popularity of tree-based models, such as XGBoost and LightGBM.

Additionally, cloud-based resource modeling has become significant, as cloud computing allows flexible scaling of resources according to demand [51]. Research in this area aims to manage resources in cloud environments efficiently, minimizing costs and enhancing performance. A stronger emphasis on sustainability and energy efficiency in resource modeling is also observable, with the aim to create environmentally friendly and energy-efficient models, particularly in high-energy consumption areas like data centers.

We here dive deep into each method:

3.2.1 Machine Learning

1. **Supervised Learning:** Newaz and Mollah's contribution is pivotal as it presents two regression models trained on large-scale resource utilization data for predicting memory usage categories in HPC systems [53]. The precision of these predictions significantly enhances overall performance and resource utilization, offering a robust foundation for further advancements in HPC resource management.

   Building upon the same theme, Tanash et al. [63] introduce an autonomous, open-source tool that uses the LightGBM tree model to predict memory and time requirements for tasks

submitted to Slurm HPC clusters. The tool alleviates the issues of resource wastage and elongated wait times by accurately predicting application-specific resource needs, offering a breakthrough in HPC resource management.

Li et al. [44] introduce a method to predict resource usage for large memory jobs in HPC clusters, demonstrating a focus on tasks that substantially impact overall memory utilization. By distinguishing large memory jobs from smaller ones, their method improves prediction accuracy, creating a ripple effect that accelerates model training and reduces prediction errors.

On the other hand, in the realm of cloud computing, Ali et al. [58] propose an innovative method combining learning automata-based ensemble prediction models to increase the accuracy of resource usage predictions. This approach's success in enhancing cloud resource management and scheduling proves it as a valuable addition to the toolkit of optimization techniques.

Cao et al. [10] also focus on resource prediction in cloud environments, introducing a dynamic ensemble model to predict CPU load. Their research offers an adaptable and reliable solution for managing CPU resources in cloud platforms.

Similarly, Fang et al. [20] propose an adaptive resource management model, RPPS, for cloud data centers. RPPS uses the ARIMA model for workload prediction and combines coarse and fine-grained resource scaling with a VM-complementary migration strategy. This novel approach improves the efficiency of workload management and VM migration in cloud data centers.

Dong et al. [19] extend the concept of resource prediction by proposing an innovative machine-learning methodology that uses data aggregation and adaptive parameter selection to enhance long-term load prediction accuracy, primarily aimed at Grid environments.

Finally, Iqbal W et al. [29] use machine learning algorithms to predict future resource utilization, offering an adaptive model selection approach. This innovative system uses a classifier and statistical features of historical resource usage to identify the best model adaptively, thus delivering enhanced accuracy, robustness, and speed in resource utilization estimation.

2. **Unsupervised Learning:** Kadda Baghdad Bey et al. [3] present an unsupervised learning, sophisticated soft computing approach for predicting CPU load that leverages both fuzzy clustering methodologies and a naive Bayesian network. This model comprises two critical phases: the first stage involves breaking down the CPU load time series into distinct clusters; the second stage employs the Adaptive Neuro-Fuzzy Inference System (ANFIS) on each isolated cluster for precise prediction. The hybrid algorithm deployed during the training phase is responsible for the conditioning of Linear Adaptive Predictors (LAPs) pertinent to each data cluster. Alongside this, a naive Bayesian network is concurrently trained using the Expectation-Maximization (EM) algorithm. When subjected to testing, CPU load time series are systematically categorized into pre-defined clusters, and the related LAP is used to predict the CPU load for each specific time instance. A performance evaluation, conducted utilizing time series traces collated by Yang and Dinda, demonstrated the superior efficiency of this model over a single ANFIS model applied uniformly to all CPU time series without preliminary clustering, thus validating the cluster-based machine learning approach implemented for CPU load prediction.

   Chen et al. [12] presents a novel, self-adjusting method named ESFCFNN designed for precise resource demand prediction in Infrastructure as a Service (IaaS) cloud environments. It adopts an ensemble model combined with a subtractive-fuzzy clustering-based fuzzy neural network. The selected features are then clustered via a subtractive-fuzzy clustering algorithm to provide input for the fuzzy neural network. Subsequently, the fuzzy neural network is trained with the clustered input data, and multiple such networks are trained with different input subsets in the ensemble model. The outputs of these networks are then amalgamated

for the final prediction. Remarkably, the ESFCFNN method is self-adjusting, capable of modifying the number of fuzzy neural networks in the ensemble model in accordance with the prediction accuracy. The real-world effectiveness of this method was verified using actual IaaS cloud environment datasets, where it outperformed other existing methods in prediction accuracy and stability. Hence, ESFCFNN serves as a robust and accurate approach for resource demand prediction in IaaS environments, promising enhanced efficiency and cost-effectiveness in cloud computing. The method's workload is manageable and its capability of self-adjusting the number of neural networks based on prediction accuracy underscores its innovation and superiority over existing methods.

3. **Semi-supervised Learning**:

   Khosla et al. [37] proposed various prediction models based on semi-supervised learning to predict CPU utilization during peak load conditions. This approach mitigates the risk of system failure in large enterprise applications. One of the models employs expectation maximization (EM) and envelope methods for feature extraction from CPU load patterns and virtual user simulation. Experimental results highlight the model's accuracy and its potential integration into monitoring and resource management procedures, presenting a promising solution for CPU load prediction in enterprise environments.

   Guan et al. [24] discuss the importance of proactive failure management in cloud computing systems and the challenges in predicting system failures. They employed both unsupervised and semi-supervised learning approaches to create reliable cloud systems, integrating Bayesian models for unsupervised fault detection and decision tree classifiers for fault prediction. This approach emphasizes the application of machine learning for proactive failure management, enhancing the reliability of cloud systems.

3.2.2 Analytical Model

- **Statistical Model:** Brandt et al. [6] present a real-time statistical analysis approach that utilizes advanced resource monitoring, analysis, and configuration tools to manage a large-scale heterogeneous environment. These tools allow for dynamic access to and interpretation of platform and application state information, enabling more effective and adaptable use of resources. The dynamic nature of these tools also allows for resource utilization, optimizing performance, and cost-effectiveness.

3.3 Taxonomy of Energy Modeling Techniques

As depicted in Figure 3.2, the taxonomy of energy modeling techniques is segmented into four key categories:

1. *Machine Learning*: This includes both Supervised Learning and Reinforcement Learning techniques. Supervised Learning employs labeled datasets to create predictive models, while Reinforcement Learning utilizes a system of rewards and punishments to guide decision-making processes.

2. *Neural Networks*: This category refers to a subset of Deep Learning methods designed to mimic the way a human brain functions, aiding in complex decision-making tasks.

3. *Analytical Models*: Comprising of both Statistical Models and PUE (Power Usage Effectiveness) Models. The former involves the use of mathematical formulations that capture the intricate relationships between data variables, while the latter specifically focuses on modeling and optimizing power usage efficiency in data centers.

4. *Graph-Based Models*: These models use graph structures to depict the interrelationships between entities, enabling a more comprehensive understanding of complex systems.

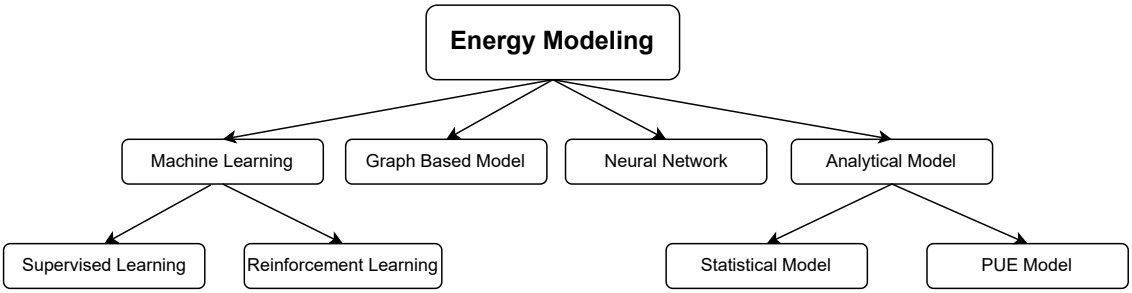


Figure 3.2: Taxonomy of energy modeling.

Based on our literature study, machine learning techniques are being more and more used to develop energy-efficient scheduling in large-scale computing systems. Machine learning algorithms are being extensively applied to forecast energy consumption patterns with higher accuracy.

Moreover, the recent surge in IoT and big data technologies has boosted the development of innovative frameworks for real-time energy modeling. These frameworks analyze real-time energy data to instantaneously predict potential energy fluctuations, facilitating immediate adjustments and thereby enhancing energy efficiency in ODA. Such predictive models contribute significantly to the effective allocation and utilization of resources within large-scale computing ecosystems, mitigating energy wastage. Consequently, reinforcement learning and neural network-based methodologies have gained traction as emerging trends within these real-time energy modeling frameworks.

Here we dive deep into each method:

3.3.1 Machine Learning

1. **Supervised Learning:** Van Bui et al. [8] employs an infrastructure that utilizes a linear model informed by on-chip performance hardware counters, a novel approach to modeling power consumption for modern multiprocessor and multicore systems. The structure incorporates application elements along with performance and power measurement and analysis components, collecting performance data using the TAU performance component. The power model is employed in the analysis of a PETSc-based parallel fluid dynamics application using the PerfExplorer component, and the Common Component Architecture (CCA) offers the component standard for scientific computation. This infrastructure is part of a larger initiative focused on computational quality of service (CQoS). This work demonstrates the potential of component-based software engineering to greatly influence the efficiency and effectiveness of performance analysis and tuning of scientific applications. Notably, the optimal linear method has been found to offer the best power efficiency for a large number of processors. While power dissipation generally increases with higher optimization levels, energy consumption tends to decrease as more aggressive compiler optimizations are applied.

   Feng et al. [62] utilize multi-variable regression modeling to analyze the statistical significance of HPL parameters and employ the energy and execution time modeling to calculate power and performance indirectly. The technical route of the article involves the use of ANOVA tests and F-ratios to test the significance of predictor variables, while p-values are used to determine statistical significance. The performance-to-power ratio metric is utilized to evaluate energy efficiency and transformation techniques are employed to test the statistical significance of predictor variables. The research conclusions indicate that multi-variable regression models can accurately predict the HPL configuration for achieving maximum energy efficiency. The best energy efficiency is attained by executing the benchmark at Reff, which represents the performance achieved while executing the HPL benchmark at the maximum energy efficiency possible based on the energy efficiency metric. Energy-efficient tuning can

aid in optimizing the energy efficiency of scientific computing, considering power and energy consumption as significant challenges for HPC in the upcoming decade.

Kestor G et al. [35] advocates for the use of a System Monitor Interface (SMI) to provide detailed per-core power data based on regression analysis, thereby facilitating the development of power-aware software algorithms. A proxy power sensor model, which gauges the active power of each core by examining the cores' activities via performance counters, is employed to monitor the system's power consumption accurately. Observing the system power consumption and per-core performance counters through micro-benchmarks, the authors amass data to create a statistical regression model that accurately predicts the power consumption of unobserved configurations. This entails running training benchmarks on a selected system configuration to monitor the system's power consumption and per-core performance counters. The significance of the work lies in its illumination of the necessity for power-aware software algorithms to enhance efficiency in exascale computing systems and regulate power consumption as a resource, with SMI providing the requisite detailed per-core power information. SMI, as a platform-independent system employing a proxy power sensor model, accurately estimates each core's active power consumption, which lends it portability. Coupled with the statistical regression model, this system accurately predicts the consumption of unobserved configurations, improving the sampling frequency and precision of power profiling.

Gschwandtner P et al. [22] introduce linear regression as a suitable method for modeling energy consumption in HPC systems. The authors present in-band energy consumption models based on hardware counters for the IBM POWER7 processor. They employ linear regression with various benchmarks and applications, exploring parallelism and compiler setups. The models demonstrate high accuracy, with a maximum error of 5.3% and an average error of approximately 1% when using GCC. Furthermore, the authors identify variations in energy consumption resulting from different compiler effects and parallelism, which serves as a unique contribution not explored in related work. The significance of this work lies in its proposal of accurate models for energy consumption in parallel programs, taking into account constraints on computational cost and complexity. The innovation of using linear regression for energy consumption modeling, the utilization of high-level benchmarks for training, and the identification of differences in energy consumption caused by compiler effects and parallelism are critical contributions of this research.

Khan W et al. [36] utilized Linear Regression Modelling to probe the nexus between thermal and IT aspects of workload, aiming to enhance the center's thermal efficiency. Utilizing the SEMMA process for exploratory analysis, we scrutinized data from four different areas, namely workload, cooling, environment, and compute nodes sensor data, recorded from the HPC cluster - CRESCO6. Through data cleansing, transformation, analysis, and a blend of supervised learning and linear regression modeling, they examined diverse data attributes. The methodology empowered energy consumption forecasting and optimized thermal management. Furthermore, feature extraction and selection, based on a positive linear correlation, yielded pertinent data attributes for effective energy management. These findings underline the significance of energy efficiency in data centers for sustainable development and provide a foundation for future data analytics modeling of energy consumption and resource utilization.

Similarly, Ozer G et al. [54] focus on using supervised learning techniques to predict CPU power and instructions retired for HPC systems. The model utilizes statistical features derived from time series data of hardware metrics and employs a multi-output random forest regressor for learning. The technical approach of the article involves collecting sensor data and performance metrics using GEOPM and DCDB frameworks, applying regression for prediction, and evaluating the model using five benchmarks from the Coral-2 suite. Two approaches were tested: GEOPM data only and GEOPM combined with DCDB data. The results showed that incorporating DCDB data improved the model's performance, which was generic enough to handle the diversity of HPC workloads. The significance of this work

lies in enabling energy-efficient CPU frequency selection during runtime, leading to reduced HPC energy consumption. The model demonstrates good performance by leveraging fine-granularity data and the ability to be retrained with recent data to adapt to system behavior changes over time. Future research directions include evaluating the model with additional data and implementing an online frequency-tuning agent.

[74] presents a dynamic capacity provisioning (DCP) system designed to optimize energy use in cloud data centers while still meeting service level agreements (SLAs). The DCP framework is made up of four components: a predictive module utilizing the ARIMA model, an optimization module, a capacity provisioning module, and a Model Predictive Control (MPC) controller that regulates the active servers based on demand, energy prices, and reconfiguration costs. When tested on a Google compute cluster of about 12,000 machines using trace-driven simulations, the DCP demonstrated a significant reduction in energy costs, potentially lowering operational costs by 18.5-50% depending on the scheduling delay. This flexible and lightweight architecture shows potential for practical implementation, balancing energy savings and capacity reconfiguration costs.

2. **Reinforcement Learning:** Wang Z et al. [71] utilize the Modular Reinforcement Learning (MRL) approach to enhance energy consumption. The theoretical basis revolves around applying the MRL approach to improve the transferability of knowledge in RL agents. The technical route of the article involves selecting and combining modules such as value functions, policy representations, and state representations, followed by implementing gating mechanisms to weigh these modules. The performance of the approach is evaluated through experiments conducted in both simple and complex environments, and the results are compared with conventional methods. The significance of this work lies in the fact that the MRL approach enhances the generalization of RL agents, enabling the transfer of learned knowledge. The MRL approach demonstrates superior performance in both simple and complex environments compared to conventional methods. However, it faces limitations such as longer computation time for gating mechanisms and the scalability issue of modules. The research concludes that the MRL approach leads to improved generalization, faster convergence, and better generalization ability in RL agents, while also acknowledging the identified limitations and proposing areas for future research.

   In the study of Lin X et al., [45], a reinforcement learning-based approach is utilized to address the energy-efficient management of data centers in dynamic environments. The theoretical basis involves defining the state space using a fuzzy state representation and applying the TD-learning algorithm to derive the optimal power management policy. The technical route of the article comprises designing the state space, action space, and reward within the reinforcement learning-based framework. The effectiveness of the proposed framework is verified using real Google Cluster Data Traces. The proposed framework achieves significant energy savings of up to 24.5% while ensuring a reasonable job response time. It addresses the need for energy-efficient data center management and operates effectively in non-Markovian environments without specific assumptions about job arrival and processing. The utilization of fuzzy state representation reduces the time complexity and accelerates the convergence rate of the reinforcement learning algorithm.

3. **Neural Network:** Tiwari A et al. [64] leveraged artificial neural networks (ANNs) to formulate power and energy models for three significant kernels, using kernel-specific compiler-based optimization parameters and hardware tables as inputs. The experimental setup included an Intel Xeon workstation equipped with a power measurement harness, with a source-to-source code transformer used to generate code variants. The findings established that ANNs can effectively predict the component-level power draw and energy usage of certain high-performance computing (HPC) computational kernels. The derived models have various applications including minimizing the number of actual benchmark runs, guiding dynamic clock frequency selections, and making better decisions in search-based auto-tuners. These models are rooted solely in compiler-level optimization parameters and demonstrate

high accuracy when subjected to well-studied compiler optimization strategies. The research's significance lies in providing insights into compiler optimizations' impact on energy usage, applicable to other application/kernel domains.

3.3.2 Analytical Model

1. **Statistical Model:** Kunkel J et al. [41] utilized advanced statistical methods, including correlation analysis, the Kolmogorov-Smirnov test, and principal component analysis (PCA), to examine power consumption in relation to hardware and software characteristics. Employing sophisticated statistical techniques such as correlation analysis, the Kolmogorov-Smirnov test, and clustering, the authors identify outliers and patterns in the data, enhancing models and facilitating the development of benchmarks for future architectures. In terms of significance, the research highlights the importance of devising new strategies to optimize and conserve energy, showcasing how statistical techniques aid in identifying meaningful behavior and reducing the number of features requiring examination. Furthermore, the study presents a methodology that employs advanced statistical methods to analyze measured data from HPC platforms, resulting in more precise power models and benchmarks. It demonstrates that a combination of select hardware counters and resource utilization metrics can provide reasonably accurate power consumption estimations, eliminating the need for complex and costly wattmeters on large-scale platforms.

2. **Power Usage Effectiveness(PUE) Model:** Jarus M et al. [30] revolves around the creation of specialized power usage effectiveness models for diverse classes of real-life applications, which are clustered based on their distinctive attributes. Decision trees are leveraged to select apt models for the current system load, based on an analysis of performance counters. A dedicated power measuring device was used to collect power usage data from the server, enriching this data with additional information regarding power usage and transforming it into formats suitable for further analysis. Clustering was used to group similar programs, and decision tree induction was employed to automatically select a model fitting the current system load. The introduced methodology achieved a mean percentage error of less than 5% in all experiments, establishing itself as a more practical and cost-effective solution compared to hardware devices for power monitoring. The proposed approach holds significance as an efficient, practical means of monitoring and estimating the power usage of HPC servers in real time. The study's novelty is primarily grounded in the use of performance counters, specialized power usage models for different classes of real-life applications, clustering, and decision trees. The authors conclude by highlighting the effectiveness of the proposed methodology, noting its mean percentage error of less than 5% in all experiments, and its practicality and cost-effectiveness compared to hardware-based power monitoring. Clustering and decision tree induction are underscored as effective methods for enhancing power usage estimation accuracy.

   Lei N et al. [43] proposed framework takes a thermodynamics-based PUE models approach and consists of four parts: energy analysis of IT equipment, energy analysis of data center infrastructure, bottom-up energy analysis with temporal and spatial resolution, and data center energy index decomposition analysis. The energy analysis of IT equipment focuses on analyzing the energy usage of IT equipment within a data center, while the energy analysis of data center infrastructure examines the energy consumption of cooling and power provisioning infrastructure. The bottom-up energy analysis provides insights into the energy usage of a data center at different time intervals and locations. Lastly, the data center energy index decomposition analysis identifies energy-saving potentials within data centers. The significance of this work lies in providing a flexible and robust methodology for quantifying data center energy with temporal and spatial resolution, addressing knowledge barriers, and uncovering technology-related obstacles that hinder energy-saving potential in data centers. The proposed modeling framework is expected to be a valuable tool for policymakers, data center operators, and researchers, offering a better understanding of data center en-

ergy demand and energy-saving opportunities, and enabling informed decision-making. The research highlights that data center energy management practices in North America, Asia Pacific, and Western Europe will have a significant impact on global data center energy consumption in the near term.

3.3.3 Graph Based Model

Lastovetsky A et al. [42] introduces a graph-based model optimization approach, which considers the complex and non-linear relationship between energy consumption and problem size aimed at minimizing execution time and energy consumption in data-parallel applications on contemporary homogeneous multicore clusters. The proposed algorithm involves several steps: representing the cluster as an undirected graph and the application as a directed graph, decomposing the directed graph into strongly connected components using the Kosaraju algorithm, determining the optimal execution order of the components by traversing them in topological order and optimizing the execution time and energy consumption of each strongly connected component. The proposed approach, which incorporates efficient algorithms and considers resource contention and NUMA, demonstrates good performance in minimizing execution time and energy consumption in data-parallel applications on modern multicore CPUs and clusters.

3.4 Mapping for Modeling Techiniques

3.4.1 Resource Modeling

As illustrated in Table 3.1, they provide structured insights into a diverse range of resource modeling methods including statistical models, as well as supervised, semi-supervised, and unsupervised learning. These varied techniques significantly contribute to predicting a broad array of resource metrics, encompassing aspects like CPU utilization, memory usage, I/O patterns, and network traffic. These predictions, in turn, facilitate optimal resource allocation, enhancing the performance of applications on expansive computing infrastructure.

At the core of these methodologies, Brandt's work introduces a statistical model distinguished by its dynamic nature, effectively optimizing resource utilization. Adding to this, Newaz and Mollah offer supervised learning techniques, demonstrating an outstanding predictive accuracy of 90.6% for HPC memory usage within the test data. Further enriching the methods, Chen et al. propose an unsupervised learning approach, characterized by its adaptability in resource demand prediction. Remarkably, this method markedly lowers error rates compared to unclustered baseline models.

In parallel, Khosla et al.'s semi-supervised learning approach stands out in predicting CPU utilization during peak load conditions. Remarkably, it exhibits an 8.05% enhancement in accuracy when juxtaposed with tree-based models, underlining the efficacy of their method.

In conclusion, each of these modeling techniques brings forth the capacity to understand and predict complex resource metrics. They showcase the power of machine learning in managing resources and offer compelling evidence of its potential to bolster application performance on large-scale computing platforms.

Table 3.1: Comparative table for resource modeling

Study	**Title**	**Year**	**Methodology**	**Improvement**	**Simulation**

Table 3.1 – continued from previous page

Study	Title	Year	Methodology	Improvement	Simulation
Newaz et al. [53]	Memory Usage Prediction of HPC Workloads Using Feature Engineering and Machine Learning	2023	Supervised Learning	The random forest model predicted memory usage in 90.6% of the jobs in the testing dataset, and the memory usage class matched the ground truth for 89.6% of the testing jobs	No
Tanash et al. [63]	AMPRO-HPCC: A Machine-Learning Tool for Predicting Resources on Slurm HPC Clusters	2021	Supervised Learning	Reduce the average waiting time for submitted jobs from 680 hours to 8.0 hours and the average turnaround time from 692 hours to 16.4 hours	No
Li et al. [44]	Practical resource usage prediction method for large memory jobs in HPC clusters	2019	Supervised Learning	The prediction on resource usage error numbers are less than 10%	No
Iqbal et al. [29]	Adaptive prediction models for data center resources utilization estimation	2019	Supervised Learning	CPU utilization prediction improves around 5% comparing with nonadaptive models	No
Rahmanian et al. [58]	A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment	2018	Supervised Learning	The performance on resource prediction improves 3.3% comparing with unassembled models	Yes
Chen et al. [12]	Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network	2015	Unsupervised Learning	Comparing with the unclustered method, the prediction performance is improved with less error	No
Cao et al. [10]	CPU load prediction for cloud environment based on a dynamic ensemble model	2014	Supervised Learning	Comparing with similar pattern models, the prediction performance improves 4.4%	No

Table 3.1 – continued from previous page

Study	Title	Year	Methodology	Improvement	Simulation
fang et al. [20]	Rpps: A novel resource prediction and provisioning scheme in cloud data center	2012	Supervised Learning	The resource prediction primarily maintains an error margin of less than 10% in terms of underestimation or overestimation for most of the duration	No
Dong et al. [19]	An effective data aggregation based adaptive long term CPU load prediction mechanism on the computational grid	2012	Supervised Learning	The experimental findings demonstrate that the algorithm DALP point surpasses preceding prediction methods on CPU load in terms of reducing the mean square error. Furthermore, the algorithm demonstrates enhanced representational capability, resulting in lower prediction errors	No
Benhammadi et al. [3]	CPU load prediction using neuro-fuzzy and Bayesian inferences	2011	Unsupervised Learning	The prediction model is suited for online use due to its lower computational demand compared to the DENFIS model. Furthermore, the refined prediction models need approximately 100 us	No

Table 3.1 – continued from previous page

Study	Title	Year	Methodology	Improvement	Simulation
Khosla et al. [37]	Forecast Extreme CPU Usages Under Peak Load Using Envelop EM Semi-supervised Learning	2022	Semi-supervised Learning	Their new practical approach expedited mitigation strategy implementation from a week to 3-4 hours. Validated in an integrated test environment, their model alerted when CPU utilization of combined servers crossed the 75% critical limit. This strategy has proven advantageous in managing, planning, and optimizing IT resources in complex enterprise IT environments.	Yes
Qiang et al. [24]	Proactive Failure Management by Integrated Unsupervised and Semi-Supervised Learning for Dependable Cloud Systems	2011	Semi-supervised Learning	Both the ensemble of Bayesian submodels and decision tree classifiers can accurately predict failures in the health-related dataset without mislabeling many normal instances. Utilizing all eight significant features selected through relevance deduction improves prediction accuracy compared to using just two features selected from relevance and redundancy reduction procedures, without significantly increasing computational overhead.	No

3.4.2 Energy Modeling

Table 3.2 shows how energy modeling contributes to facilitating informed decision-making and effective allocation for optimized energy use. Energy modeling techniques such as supervised learning, reinforcement learning, neural networks, statistical models, PUE models, and graph-based models contribute significantly by enhancing energy efficiency in large-scale computing systems. Machine learning algorithms, as evidenced by Van Bui et al.[8], Feng et al.[62], Khan W et al.[36], Ozer G et al.[54], and [74], are extensively used to forecast energy consumption patterns with higher accuracy and are instrumental in optimizing power consumption. The integration of IoT

and big data technologies has enabled the development of real-time energy modeling frameworks, which leverage reinforcement learning and neural networks, as highlighted in studies by Lin X et al.[45], and Tiwari A et al.[64], respectively, to predict potential energy fluctuations instantaneously. These predictive models enhance the allocation and utilization of resources, thereby reducing energy wastage. Other approaches such as the PUE model [30, 43], statistical models [41], and graph-based models [42] further aid in optimizing energy efficiency by creating power usage models and analyzing power consumption patterns. Therefore, energy modeling plays a crucial role in the DevOps platform by enabling energy-efficient operations, optimizing resource utilization, and reducing operational costs.

Table 3.2: Comparative table for energy modeling

Study	**Title**	**Year**	**Methodology**	**Improvement**	**Simulation**
Khan et al. [36]	Exploratory data analysis for data center energy management	2022	Supervised Learning	The exploratory data analysis and time series decomposition, which empowers the future prediction of energy usage and the enhancement of thermal management efficiency in data centers	No
Ozer et al. [54]	Towards a predictive energy model for HPC runtime systems using supervised learning	2020	Supervised Learning	The model can predict metric values accurately with an average relative error less than 15.3%, and it is especially precise for the power package metric comparing with naive approach	No
Lei et al. [43]	A robust modeling framework for energy analysis of data centers	2020	PUE Model	The model, even amidst potential uncertainty in yearly simulations, could produce satisfactory PUE value estimates with a maximum relative error of ±4%	Yes

Table 3.2: Comparative table for energy modeling

Study	**Title**	**Year**	**Methodology**	**Improvement**	**Simulation**
Kunkel et al. [41]	Understanding hardware and software metrics with respect to power consumption	2018	Statistical Model	This found that a small combination of hardware counters and resource utilization metrics can accurately estimate power consumption, potentially negating the need for complex wattmeters and aiding in the creation of more efficient, energy-saving software such as green schedulers	No
Wang et al. [71]	Modular reinforcement learning for self-adaptive energy efficiency optimization in multicore system	2017	Reinforcement Learning	method can enhance energy efficiency by 20% when compared to the individual learning method	No
Lin et al. [45]	A reinforcement learning-based power management framework for green computing data centers	2016	Reinforcement Learning	The proposed power management framework successfully aligns with the goal of reducing server pool energy consumption while maintaining an acceptable average job response time. It achieves energy savings of 5.4%, 17.8%, and 24.5% with a job response time penalty of 0%, 29%, and 73% respectively	Yes
Lastovetsky et al. [42]	New model-based methods and algorithms for performance and energy optimization of data parallel applications on homogeneous multicore clusters	2016	Graph Based Model	The minimum, average, and maximum percentage reductions in energy for the first dataset were 1%, 24%, and 68% respectively, which optimizing for performance alone can lead to a significant reduction in energy	Yes

Table 3.2: Comparative table for energy modeling

Study	**Title**	**Year**	**Methodology**	**Improvement**	**Simulation**
Gschwandtner et al. [22]	Modeling CPU energy consumption of HPC applications on the IBM Power7	2014	Supervised Learning	Despite the complex nature of the POWER7 processor, the study concluded that linear regression could achieve high accuracy with a limited set of input variables, provided the compiler and multi-threading settings are appropriately managed	No
Jarus et al. [30]	Runtime power usage estimation of HPC servers for various classes of real-life applications	2014	PUE Model	Improvement in the estimates on all other applications reach even almost 20% compared with the one presented by the tree	No
Kestor et al. [35]	Enabling accurate power profiling of HPC applications on exascale systems	2013	Supervised Learning	Coarse-grained power measurements might obscure crucial details, like power spikes or the fact that different threads within an application can have varying power profiles	No
Zhang et al. [74]	Dynamic energy-aware capacity provisioning for cloud computing environments	2012	Supervised Learning	This method decreases energy costs by $7 per hour, equating to a 20% reduction, and can lower the total operational cost by about 18.5 - 50% while still maintaining the target scheduling delay	Yes
Tiwari et al. [65]	Modeling power and energy usage of HPC kernels	2012	Neural Network	The method significantly lower the number of necessary benchmark runs by filling most of the benchmark results space with modeled outcomes, thereby reducing the associated overhead	No

Table 3.2: Comparative table for energy modeling

Study	**Title**	**Year**	**Methodology**	**Improvement**	**Simulation**
Subramaniam et al. [62]	Statistical power and performance modeling for optimizing the energy efficiency of scientific computing	2010	Supervised Learning	Improved the energy efficiency of the HPL benchmark by automating the process of identifying optimal parameters through our models	No
Bui et al. [8]	A component infrastructure for performance and power modeling of parallel scientific applications	2008	Supervised Learning	The effectiveness of this approach largely hinges on the quality of the interfaces for database access and performance analysis	Yes

3.5 Summary

In this chapter, we delved into diverse related works. We began by exploring data characterization in HPC systems, emphasizing energy efficiency and performance optimization. Subsequently, we turned our attention to various modeling techniques for resource and energy, crafting a taxonomy for each domain. Furthermore, we mapped these techniques, presenting a comparative table that outlines their methodologies, enhancements, and whether they were evaluated in a simulation environment.

Chapter 4

Resource and Energy Data Characterization

This section addresses Research Question 1 (RQ1): "How can we Characterize and Analyze Resource and Energy Usage?" We dissect this question and provide comprehensive answers through several stages. Firstly, Section 4.1 presents a requirements analysis specific to the characterization of resource and energy usage as per the context of this thesis. Secondly, in Section 4.2, we describe our chosen dataset and elaborate on the data processing techniques applied. In the rest of Section 4.3 4.4 4.5, we showcase our in-depth analysis and characterization.

4.1 Requirements Analysis

In this section, we extend the knowledge obtained in Chapter 2 to structure the stakeholders, their associated concerns, and the essential requirements for the resource and energy characterization that will adequately address the research question. The comprehensive requirement analysis undertaken here serves as a directive for the subsequent analytical processes.

We first describe stakeholders in this section. Subsequently, we illustrate the anticipated use cases wherein these stakeholders; this discussion is encapsulated in a subsequent section. Drawing on these projected use cases, we proceed to categorize and explicate both the functional and non-functional requirements, each in their dedicated sections. This structured approach allows us to ensure a thorough understanding from various perspectives.

4.1.1 Stakeholders

The main stakeholders we identified are *Data Center Operators, Scientific Researchers, IT Managers, and Administrators.* These stakeholders are envisioned to be the primary users.

- **S01. Data center operators:** These are the individuals directly involved in the day-to-day operations and maintenance of the data center. Their work significantly influences the data center's resource usage and energy efficiency. Improving these aspects is crucial for them as it directly impacts the data center's performance and reliability.
- **S02. Scientific researchers:** They approach data centers from a research perspective. They analyze the data generated from resource and energy characterization to identify patterns, understand efficiency metrics, and devise new strategies to optimize data center performance and minimize environmental impact. Their fields of study can range from computer science and engineering to environmental science. Their research paves the way for advancements in data center technology and broadens the scientific and technological understanding of energy-efficient computing.

- **S03. IT managers and administrators:** Tasked with overseeing the entire IT infrastructure, which includes the data center, these professionals handle capacity planning, cost control, and performance optimization. Hence, they require precise and comprehensive details about resource and energy consumption.

4.1.2 Use Cases

We define general use cases for the stakeholders, which help us in the requirements specification phase. The use cases are drawn from stakeholder concerns.

- **UC1. Efficient resource and energy allocation:** Data center operators aim to ensure optimal performance by monitoring resource and energy utilization, particularly power usage and CPU utilization, across the infrastructure. By identifying and rectifying issues related to underutilized or overutilized resources, they can rebalance or reassign resources more effectively. The Resource and Energy Characterization offers crucial data regarding current utilization and consumption levels, thus aiding in decision-making for optimization. This addresses the concerns of stakeholder S01.
- **UC2. Analyzing energy and CPU utilization trends and patterns:** Stakeholders such as scientific researchers, with interests in data center efficiency or sustainability, might focus on the examination of CPU usage and energy consumption trends within data centers. Resource and Energy Characterization data allows them to observe pattern evolutions over time. They can evaluate the effects of introducing energy-efficient hardware or modifying workload schedules on energy consumption. Additionally, this data aids in the study of resource utilization trends. Such insights can lead to informed recommendations for resource allocation strategies or for drafting efficient data center management policies. This in-depth analysis not only provides practical strategies for enhancing data center operations but also contributes to a broader scientific comprehension of fields like green computing and sustainable IT infrastructure management. This addresses the concerns of stakeholders S01, S02, and S03.
- **UC3. Capacity planning:** IT Managers aim to predict and plan for future capacity requirements based on current resource usage trends and anticipated growth. With Resource and Energy Characterization, they can comprehend current usage patterns and make future demand forecasts. This intelligence can shape decisions regarding the procurement of additional resources, infrastructure upgrades, or strategizing to maximize existing capacity utilization. This addresses the concerns of stakeholder S03.

4.1.3 Requirements

For each stakeholder, the requirements are derived from their respective use cases. Among them, recognizing patterns(REQ2) in data is crucial for proactive decision-making, driving operational efficiency, and gaining deeper insights into system behaviors. In data centers, such analysis plays a pivotal role by enabling accurate forecasting, resource optimization, and revealing interactions over time.

- **REQ1. Provide generic statistics and distribution of resource usage and power consumption:** Catering to UC1, UC2, and UC3, this requirement emphasizes the necessity for basic yet comprehensive statistical analysis, particularly of CPU resource usage and power consumption. Presenting distributions could entail employing graphical representations to visualize the distribution and frequency of various resource usage and power consumption levels.
- **REQ2. Find repeating patterns in historical data:** Addressing UC1, UC2, and UC3, this requirement underscores the system's capability to evaluate historical data to discern recurrent patterns. These patterns might manifest as periodic spikes in resource

usage, correlations between specific workloads and escalated consumption, or other trends that recur over time. Recognizing these patterns is pivotal for predicting future trends and refining planning.

- **REQ3. Identify changes over time:** Serving the needs of UC1, UC2, and UC3, this requirement signifies the importance of monitoring alterations in resource usage and power consumption over extended periods rather than merely offering a current snapshot. This might involve visualizing consumption trends over time, computing the change rate, or pinpointing moments when substantial changes transpired. Recognizing the evolution in usage and consumption grants insights into the data center's dynamic needs and operational efficiency.
- **REQ4. Compare different nodes and racks:** Relevant for UC1, UC2, and UC3, this requirement spotlights the necessity to juxtapose different nodes and racks within the system. This could shed light on efficiency variances across the data center, flagging particularly effective or suboptimal nodes or racks. Additionally, it could aid in evenly distributing workloads throughout the data center.

4.2 Dataset Introduction

In this section, we delve into the specifics of the dataset under examination. This particular dataset stands out due to its multi-level categorization and fine-grained granularity. Such intricate detailing enables a thorough and in-depth analysis, facilitating more nuanced insights and accurate predictions. The diversity in data levels provides a comprehensive overview, ensuring that both macro and micro perspectives are captured. The dataset's fine granularity ensures that even subtle nuances are recorded, allowing for a detailed exploration of trends, patterns, and anomalies. By leveraging the richness of this dataset, we can derive actionable insights that can significantly impact decision-making and strategy formulation.

4.2.1 Overview

LISA System

We utilize the dataset sourced from the LISA data center [18]. The temporal granularity is achieved through a sampling rate of 30 seconds, spanning around 5 months of data collection from June 2022 to November 2022. The spatial granularity is enriched by the extensive collection of high- and low-level server and rack metrics by Surf's data center operators. The fine temporal and spatial granularities present in this dataset offer the potential for more in-depth and innovative insights into the operations of data centers, which we explore in this study.

The LISA data center consists of a total of 349 nodes distributed across 20 racks, displaying heterogeneity. The racks in the data center are categorized as either generic, containing nodes with CPUs only, or designated for ML, featuring nodes equipped with both CPUs and GPUs. To identify ML nodes, the data center operators analyzed the workloads executed on them. These ML nodes are reserved and accessible only with special privileges assigned by the data center administrators. Each rack can accommodate up to 32 generic nodes or up to 7 ML nodes, with the specific distribution depending on the CPU and GPU models used and the power-consumption limitations imposed by the cooling system [69].

SLURM and Prometheus: Job and Node Data

For the workload submitted in LISA, each job is exclusive to a user, with no multi-user jobs or workflows implemented at present. The cluster manager SLURM [60] is utilized to enable users to queue jobs for different node types. Machines can be either reserved for a specific duration or jobs can be submitted to a job queue, where they are executed on a resource that meets the specified requirements. Job scheduling follows a first-in, first-out (FIFO) approach per stakeholder,

ensuring fair sharing across stakeholders. The data center offers nodes with co-allocation of jobs or exclusive use through the utilization of distinct queues.

For the nodes in the data center, certain nodes serve as entry, administrator, and compilation nodes, allowing users to compile libraries or programs, process data, generate outputs, etc., without interfering with the jobs running on other nodes. For the purpose of our analyses, these nodes were excluded.

SLURM is responsible for monitoring the usage of nodes and the status of machine and job queues, and it logs this information, enabling queries. At the same time in LISA, Prometheus is adopted to capture detailed temporal and spatial data of each node.

4.2.2 Data Preprocessing

Data cleaning is a crucial step in the further process, aimed at improving the quality and reliability of the dataset. In this work, due to focusing on data characterization on resource usage and energy consumption, we finally selected 8 related features in the node dataset and 12 features in the job dataset as shown in Paragraph Feature Selection 4.2.2. For resource usage, we focus on the system CPU load metric using the feature *'load1'*, which captures the average number of threads that were in the running or waiting states over the last minute. Since this is an average taken over a time period, it can be a fractional number. For energy consumption, we focus on the Node power consumption information using the feature *'surfsara power usage'* and *'nvidia gpu power usage milliwatts'*. Further, we did the time-series cleaning and missing value checking for further data characterization and analysis. Based on correlation analysis, some highly related features are also included in characterization and analysis like temperature-related features.

Feature Selection

We focus on energy and resource data characterization, so we first select related features as shown in Table 4.1 and 4.2.

We focus on CPU utilization, power usage, and temperature. CPU utilization acts as a barometer for system workloads, revealing how intensively the computational heart of the system is operating. Power usage, on the other hand, directly translates to operational costs and environmental impact, making it essential for gauging energy efficiency. Meanwhile, temperature, a byproduct of computational activities, offers insights into system reliability and the effectiveness of cooling solutions. Collectively, these metrics complement and reinforce the foundations of our modeling endeavors.

Table 4.1: Feature descriptions Prometheus dataset.

Feature Name	Description
id	Job ID
timestamp	Data timestamp
node	Node name
node_load1	Node load average over the last 1 minute
surfsara_power_usage	Node power consumption information
node_hwmon_temp_celsius	Node temperature in degree Celsius
nvidia_gpu_power_usage_milliwatts	Node Nvidia GPU power usage in milliwatts
nvidia_gpu_temperature_celsius	Node Nvidia GPU temperature information in degree Celsius

Time-series Cleaning

We first perform cleaning on the job dataset. Firstly, we filter out jobs whose start time falls outside the specified start and end time range of the dataset. This step helps us focus on the relevant jobs within the desired timeframe.

Table 4.2: Feature descriptions SLURM dataset.

Feature Name	Description
id	Unique identifier for a job
start_date	Start time of the job
end_date	End time of the job
node	Node where the job ran
nodetypes	Node type information
numnodes	Number of nodes used by the job
numcores	Total number of cores used by the job
sharednode	Whether the job ran on a shared node
submit	Submission time of the job
start	Actual start time of the job
end	Actual end time of the job
state	Status of the job

Missing Value Checking

We look into the missing value percentage in each feature as shown in Table4.3. For some features related to GPU, there are missing values due to only specific nodes having GPUs, so we keep the missing values. For other features missing values, we address missing values by removing them considering only a very little percentage of missing values.

Table 4.3: Percentage of missing values across the dataset.

Feature Name	Missing Percentage
id	0.00%
timestamp	0.00%
node	0.00%
node_load1	0.40%
surfsara_power_usage	1.30%
node_hwmon_temp_celsius	0.40%
nvidia_gpu_power_usage_milliwatts	90.11%
nvidia_gpu_temperature_celsius	90.11%
id	0.00%
start_date	0.00%
end_date	0.00%
node	0.00%
nodetypes	0.00%
numnodes	0.00%
numcores	0.00%
sharednode	0.00%
submit	0.00%
start	0.00%
end	0.00%
state	0.00%

4.3 Generic Statistic and Distribution of Resource Usage and Power Consumption

In this section, we begin our analysis with a general distribution overview, progress to rack-level specifics, and then delve into node-level details.

4.3.1 General Distribution

Table 4.4: Statical description of CPU load and power usage in the dataset.

Matrix	Mean	Median	Std	Min	Max
CPU Load	10.8	4.0	49.9	0.0	4910.3
Node Power Usage(W)	171.5	132.0	211.2	32.0	1680.0

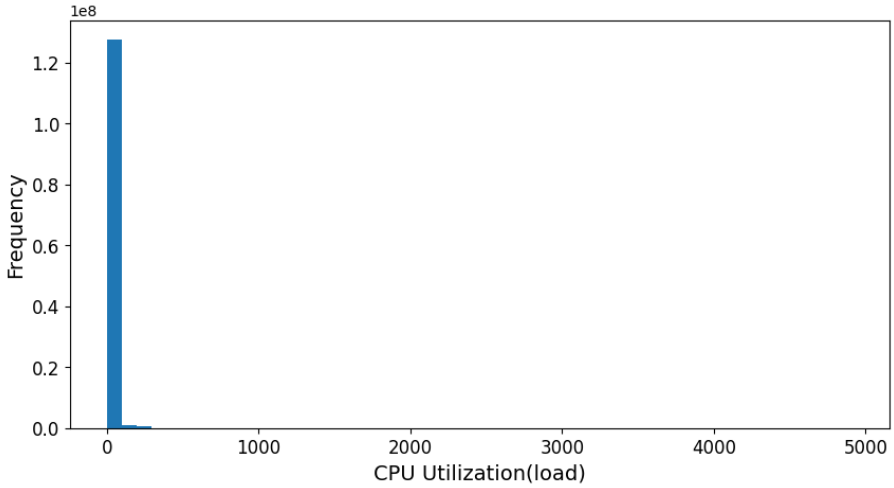


Figure 4.1: CPU load distribution in the dataset.

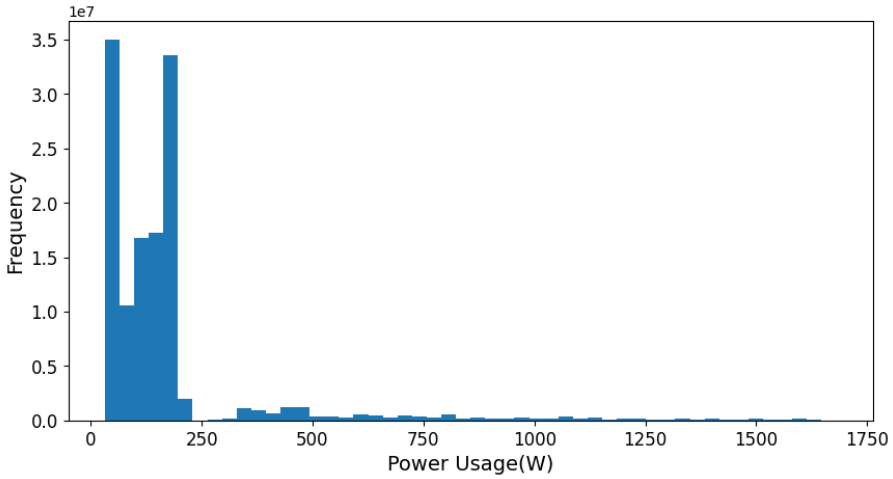


Figure 4.2: Power usage distribution in the dataset.

O1: The mean CPU Load is 10.8, suggesting an average CPU utilization, while a median of 4.0 indicates the CPU Load distribution skews toward lower values.

O2: The CPU Load's standard deviation is 49.9, indicating a significant variability.

O3: The mean and median values for Node Power Usage are 171.5 and 132.0, respectively, emphasizing power consumption's concentration at the lower end.

O4: The standard deviation for Node Power Usage is 211.2, pointing to substantial variability in power consumption among nodes.

In this part, we first examine the generic statistics and distribution of resource usage and power consumption, with a specific emphasis on CPU load and power usage as shown in Table 4.4 and Figure 4.1,4.2. Understanding the statistical properties and distributions of CPU load and power usage is essential for optimizing resource allocation, improving energy efficiency, and designing effective power management strategies.

For CPU Load as shown in Figure 4.1, the mean value of 10.8 indicates the average level of CPU utilization across the observed data. The median value of 4.0 represents the middle point, dividing the data into two equal halves. This suggests that the distribution of CPU Load is skewed toward lower values, as evidenced by the significant difference between the median and mean. This observation is further supported by Figure 1, which depicts the distribution of CPU Load. It is evident from the figure that a substantial number of values are zero, indicating instances of low CPU utilization. Additionally, the distribution is skewed toward lower values, with a higher concentration of data points in the lower range. The standard deviation of 49.9 reflects the degree of variability or dispersion in CPU Load values around the mean. The minimum and maximum values of 0.0 and 4910.3, respectively, reveal the range of CPU Load observed in the dataset.

Regarding Node Power Usage, from Figure 4.2, it is evident that the distribution of Node Power Usage is skewed towards lower values, particularly in the range of 32 to 250. This means that a significant number of nodes exhibit relatively lower power consumption. The mean value of 171.5 represents the average power consumption across all nodes, indicating the overall power usage level. The median value of 132.0, being the middle point of the distribution, further emphasizes the concentration of power usage towards the lower end. The standard deviation of 211.2 highlights the variability in power consumption among the nodes. This suggests that there is considerable diversity in power usage patterns, with some nodes consuming significantly more power than others. The minimum value of 32.0 indicates the lowest observed power usage, while the maximum value of 1680.0 represents the highest power consumption recorded in the dataset. These values provide valuable insights into the range of power usage, indicating the potential for both low and high-power-consuming nodes within the system.

Overall, this analysis reveals that the Node Power Usage distribution is skewed towards lower values, especially in the range of 32 to 250. This suggests a prevalence of nodes with relatively lower power consumption, potentially indicating efficient power utilization or the presence of energy-saving mechanisms in the system. However, the considerable standard deviation implies variations in power consumption levels, highlighting the need for further investigation into the factors influencing power usage variations among the nodes.

4.3.2 Rack Level

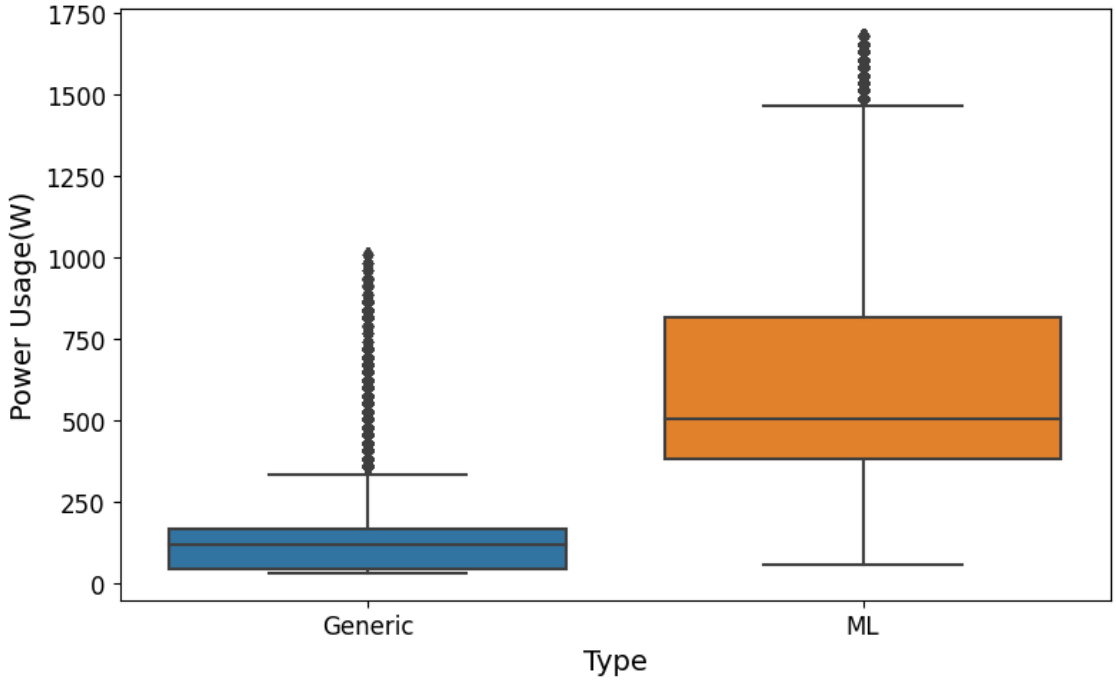


Figure 4.3: Rack level power consumption comparison grouped by generic nodes and ML nodes.

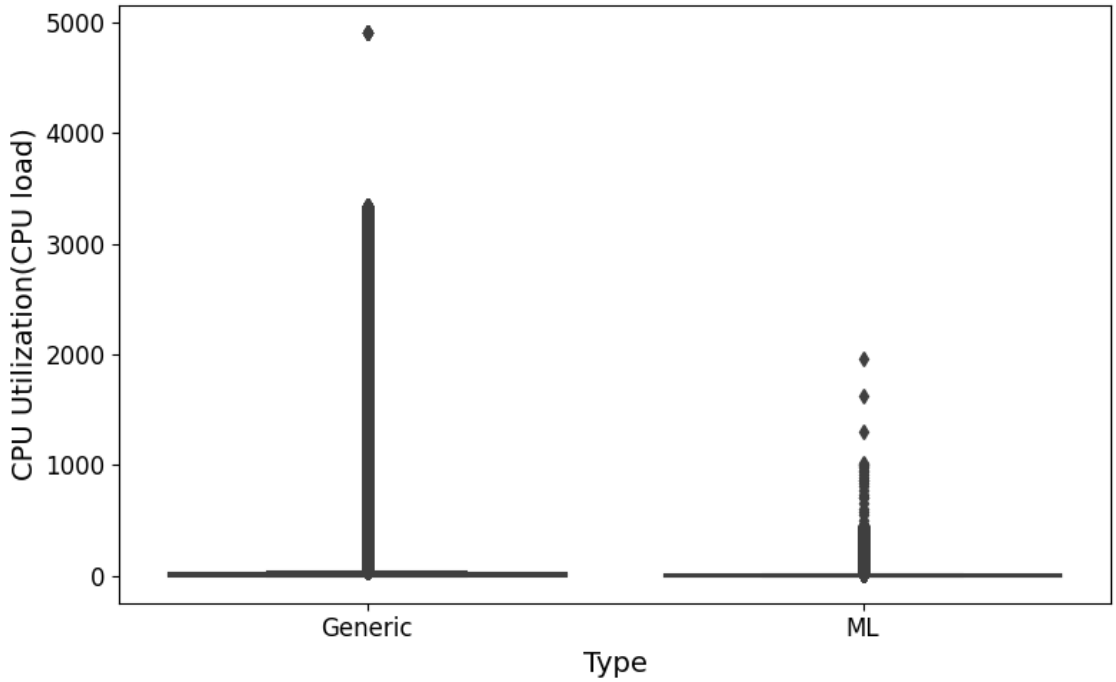


Figure 4.4: Rack level CPU utilization comparison grouped by generic nodes and ML nodes.

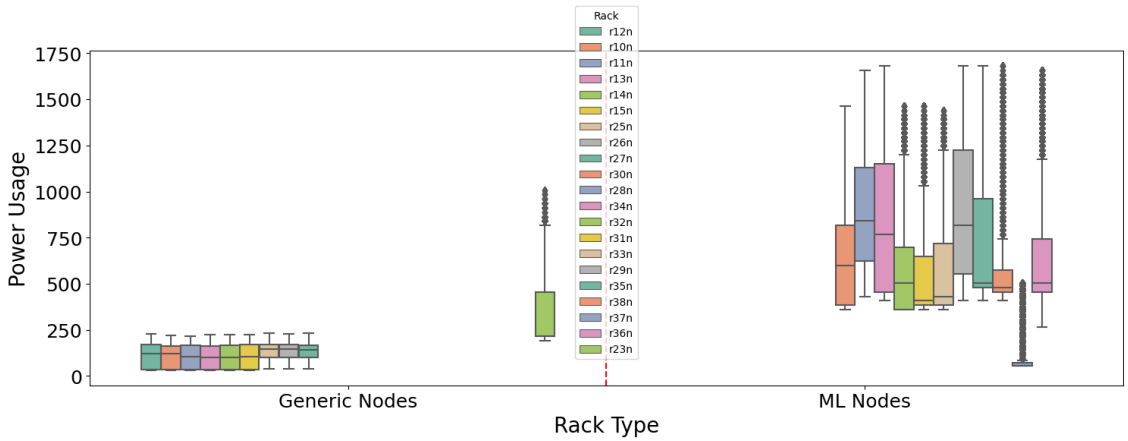


Figure 4.5: Distributions of rack power consumption grouped by generic nodes and ML nodes.

O5: Nodes in Generic racks generally consume less power than those in ML racks.

O6: Power usage within Generic racks is stable, indicating predictable energy performance, with rack 23 as an outlier due to its 48 CPU cores—quadruple the average.

O7: CPU utilization patterns between generic and ML racks are largely similar, indicating comparable workload demands across both types.

O8: An exception is rack 12, which has nodes with exceptionally high CPU utilization, suggesting it handles more computationally demanding tasks than other racks.

O9: ML racks have a higher temperature than Generic racks by around 3.5 °C.

Figure 4.3, 4.4, and 4.5 provides a comparative analysis of power usage and CPU load at the rack level, specifically between generic racks and ML racks. We investigate the power consumption of individual nodes within each rack. The box plot captures the middle 50% of the data, with the line inside denoting the median, while the 'whiskers' show variability outside the upper and lower quartiles, and points beyond them represent outliers.

The majority of the nodes in generic racks exhibit significantly lower power consumption compared to their ML counterparts, demonstrating their energy efficiency. Furthermore, the power usage within the generic racks is relatively stable, pointing to their predictable and consistent energy performance. An outlier within the generic rack category is rack 23, which consumes notably more power than the rest of the generic racks. After further investigation, this node is equipped with 48 CPU cores, which is four times more than the typical count found in generic nodes.

We also checked the temperature difference between the Generic rack and the ML rack. In terms of thermal performance, it is worth noting that ML racks register a higher temperature than generic racks by approximately 3.5 °C. This observation underscores the thermal implications of handling machine learning workloads and calls for specialized cooling solutions tailored for such environments.

In contrast, the differences in CPU utilization between the generic and ML racks are less discernible. Both types of racks show similar patterns and distributions in CPU utilization, suggesting that workload demand does not significantly differ between them. However, an exception is rack 12, which houses certain nodes experiencing extraordinarily high CPU utilization. This implies that rack 12, which deals with normal requests, is under heavier computational demand.

4.3.3 Node Level

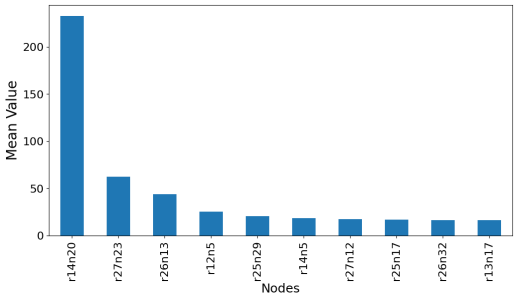


Figure 4.6: CPU load consumption Top10 nodes

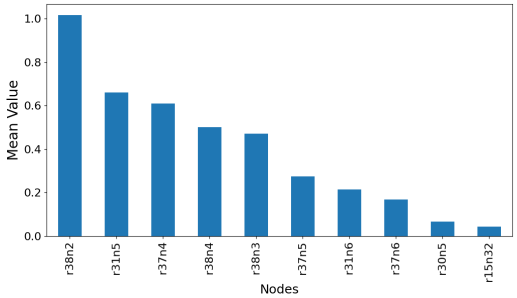


Figure 4.7: CPU load consumption Last10 nodes

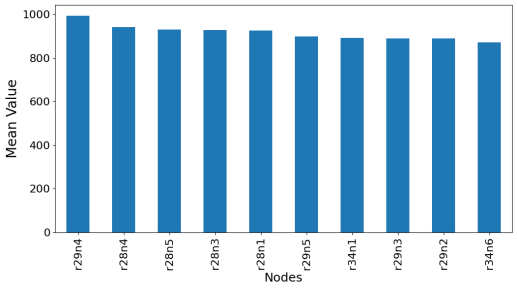


Figure 4.8: Power usage Top10 nodes

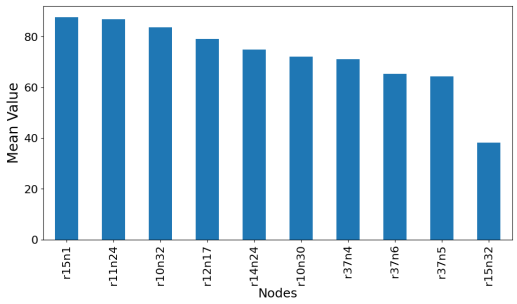


Figure 4.9: Power usage Last10 nodes

O10: A marked discrepancy exists in CPU load among nodes, with node *'r14n20'* having a significantly higher mean CPU load, exceeding 200.

O11: Nodes with the least CPU load primarily originate from racks *'r37', 'r38'*, which consist mainly of GPU nodes, indicating less intensive CPU utilization in GPU-focused systems.

O12: For power usage, there aren't any prominent outliers, but a significant portion of nodes with high power consumption belongs to the *'r29'* rack, which predominantly houses GPU nodes.

O13: Nodes in the 'Normal' rack, such as 'r15n30', display reduced power consumption. Node *'r10n32'*, a generic node in the *Normal* rack, exhibits both the lowest mean power usage and the smallest CPU load.

O14: Some GPU nodes, particularly *'r37n4', 'r37n6', and 'r37n5'*, also show low average power consumption.

Further, we look into CPU load and power consumption at the node level – the performance of each node.

In terms of CPU load, there is a notable discrepancy observed among the nodes, with node *'r14n20'* registering a significantly higher mean value as shown in Figure4.6. Specifically, the mean CPU load for this node exceeds 200, a magnitude that is far greater when compared to the node with the second-highest mean CPU load, which is approximately 60. It is pertinent to highlight that node *'r14n20'* is part of the *'r14'* rack, categorized under the umbrella of generic nodes. Furthermore, the partition associated with this specific rack is designated as 'shared'.

Furthermore, upon examination of the nodes with the least CPU load, as shown in Figure4.7, a majority are observed to originate from the racks *'r37', 'r38'*. Notably, these racks are primarily

composed of GPU nodes. This could suggest that systems primarily dependent on GPU computing might not utilize their CPU resources as intensively, leading to a lower CPU load.

In contrast to CPU load, there are no distinct outliers with regards to power usage as shown in Figure4.8; no individual node exhibits substantially higher average power consumption. Interestingly though, the majority of the nodes with the highest power usage are found in the *'r29'* rack. It is noteworthy that these nodes are predominantly GPU nodes, suggesting a correlation between GPU usage and power consumption.

It is noteworthy that node *'r10n32'*, identified as a generic node assigned to the *Normal* rack, displays the lowest mean power usage, as shown in Figure 4.9. This node also has the smallest CPU load, which supports the expected correlation between power usage and CPU load. Other nodes within this rack, such as 'r15n30', also show lower power consumption. This could suggest that certain configurations or usage patterns within this rack are contributing to improved energy efficiency. Interestingly, some GPU nodes, specifically *'r37n4', 'r37n6', and 'r37n5'*, also report low average power usage. This observation aligns with the CPU load analysis of the last ten nodes, suggesting that specific workloads or operational conditions could lead to reduced power usage. This insight emphasizes the relationship between workload, operational circumstances, and resource utilization.

4.4 Feature Correlation Regarding Resource and Energy

The increasingly complex and dynamic nature of data center operations necessitates a comprehensive understanding of various metrics associated with their functioning. This insight is essential in system monitoring, predictive analysis, and enhancing the overall efficiency of data centers. In our study, we focus on the correlation of resource and energy-related features with other features in the dataset. The insights deep dive into low-level metrics correlation and the implication for data collection and analysis. We aim to investigate the correlation between diverse metrics in understanding data center behavior. Specifically, based on the insights, we can determine whether all resource and energy-related metrics in our dataset are necessary, or if some can be inferred from others through correlation.

Feature correlation analysis in time series data is a pivotal step toward successful modeling and prediction, which will also be included in this paper. It aids in discerning the most relevant features for a predictive model, especially those that have a strong correlation with the target variable. At the same time, it helps identify and handle multicollinearity, a scenario where features are highly correlated with each other, which can lead to unstable models and obscure interpretations of predictor effects. Consequently, redundant features can be eliminated, improving the computational efficiency of the model – a critical factor when working with extensive datasets or intricate models. Moreover, the analysis offers insights into the temporal dynamics among variables, including lagged relationships, which are essential for constructing accurate predictive models. It also enhances model performance by including features that have strong direct or lagged correlations with the target variable.

4.4.1 Correlation Calculation Methods

We use both Pearson and Spearman correlation coefficients to provide a comprehensive view of relationships in data. While Pearson detects linear relationships and requires data to be normally distributed, Spearman identifies monotonic trends without needing data normality. The P-value, meanwhile, is vital to validate the statistical significance of any observed correlation. A low P-value suggests a statistically significant correlation, ensuring the relationship isn't just due to random chance. Always combining visualization, such as scatter plots, with these statistics gives a clear picture of data associations.

1. **Pearson** The Pearson correlation coefficient, denoted by r, is a widely used statistical measure to understand the strength and direction of the linear relationship between two variables [59]. Mathematically, it is computed as follows:

First, calculate the mean of each variable, where the mean (μ_x for x and μ_y for y) is computed as the sum (Σ) of all values (x_i for x and y_i for y) divided by the number of values (n). Second, compute the deviation from the mean for each value ($x_i - \mu_x$ and $y_i - \mu_y$). Next, calculate the product of these deviations for each pair of values. The Pearson correlation coefficient (r) is then calculated as the sum of these products ($\Sigma(x_i - \mu_x)(y_i - \mu_y)$) divided by the square root of the product of the sums of squared deviations ($\sqrt{\Sigma(x_i - \mu_x)^2 \Sigma(y_i - \mu_y)^2}$) [17].

$$r = \frac{\Sigma(x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\Sigma(x_i - \mu_x)^2 \Sigma(y_i - \mu_y)^2}} \tag{4.1}$$

As per Equation 4.1, the Pearson correlation coefficient r ranges between -1 and +1, providing an estimate of the strength and direction of the linear relationship between two variables. It ranges between -1 and +1, where -1 signifies a perfect negative linear relationship, +1 indicates a perfect positive linear relationship, and 0 suggests no linear relationship.

2. **Spearman** The Spearman's rank correlation coefficient often represented as ρ or r_s, is a non-parametric statistical measure that assesses the strength and direction of the monotonic relationship between two ranked variables. Unlike the Pearson correlation coefficient which assumes a linear relationship and normally distributed variables, the Spearman correlation does not make these assumptions, making it more versatile in its applications. It is also more robust to outliers and can be used for ordinal, interval, and ratio data.

   Spearman's correlation coefficient is computed through the following steps:

   1. Each value in the dataset is ranked, with the smallest value assigned to rank 1. For values that share the same data point, they are assigned the average of the ranks they would have received if they were slightly different.

   2. The difference in ranks, denoted as d, is calculated for each data pair.

   3. The differences are squared, resulting in d^2.

   4. The squared differences are summed to produce Σd^2.

   The correlation coefficient (ρ) is then computed using the following formula:

$$\rho = 1 - \frac{6\Sigma d^2}{n(n^2 - 1)} \tag{4.2}$$

   where n is the number of data pairs. This coefficient can range from -1 to 1, with -1 indicating a perfect decreasing relationship, 1 indicating a perfect increasing relationship, and 0 indicating no relationship.

3. **P-value** The p-value is a critical concept in statistical hypothesis testing, including when assessing the correlation between variables. When measuring correlation, the p-value reflects the probability of observing the obtained data (or data that's more extreme) assuming that the null hypothesis, typically that there's no correlation, is true.

   In our work, the correlation coefficient can tell the strength and direction of the relationship between two variables. However, it does not tell whether that relationship is statistically significant, which indicates that it occurred by chance or not. A hypothesis test determines whether the correlation could be due to chance, yielding a p-value. If the p-value is below a set significance level (usually 0.05), it is statistically significant, leading to the rejection of the null hypothesis and suggesting a non-zero correlation in the population [28]. Conversely, if the p-value exceeds the threshold, we fail to reject the null hypothesis, indicating no strong evidence for a population correlation.

4.4.2 Results and Insights

We have analyzed the correlation for CPU utilization and power usage to better understand their interrelationship. Here are the insights and results from our examination.

4.4.2.1 CPU Utilization Correlation

Figure 4.10 and 4.11 showcase the interrelation between metric pairs and CPU utilization in the node dataset. Just as we examined the correlation in terms of power usage, a similar exploration was carried out for metrics in relation to CPU utilization.

From the 68 metrics scrutinized, a group of 6 displayed a pronounced linear interdependence with CPU utilization as per the Pearson correlation, registering a coefficient of 0.8 or higher. Switching our attention to the Spearman correlation, a distinct set of 9 metrics was identified that bore a strong monotonic association with CPU utilization.

Diving deeper, an intriguing observation is the nexus of these significantly correlated features to node temperature and power consumption. This could imply that as CPU utilization surges, there is a corresponding uptick in node temperature and power usage. Such interdependencies highlight the importance of efficient thermal and power management in systems with fluctuating CPU demands. Recognizing these relationships can assist system administrators in preemptively managing potential stress on nodes, ensuring smoother and more sustainable operations.

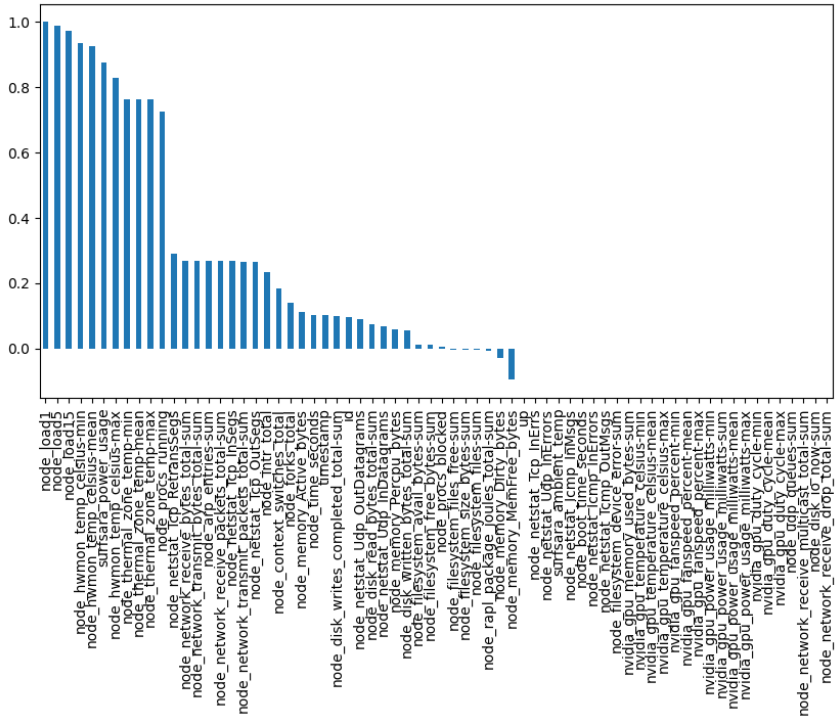


Figure 4.10: Pearson correlation for CPU utilization.

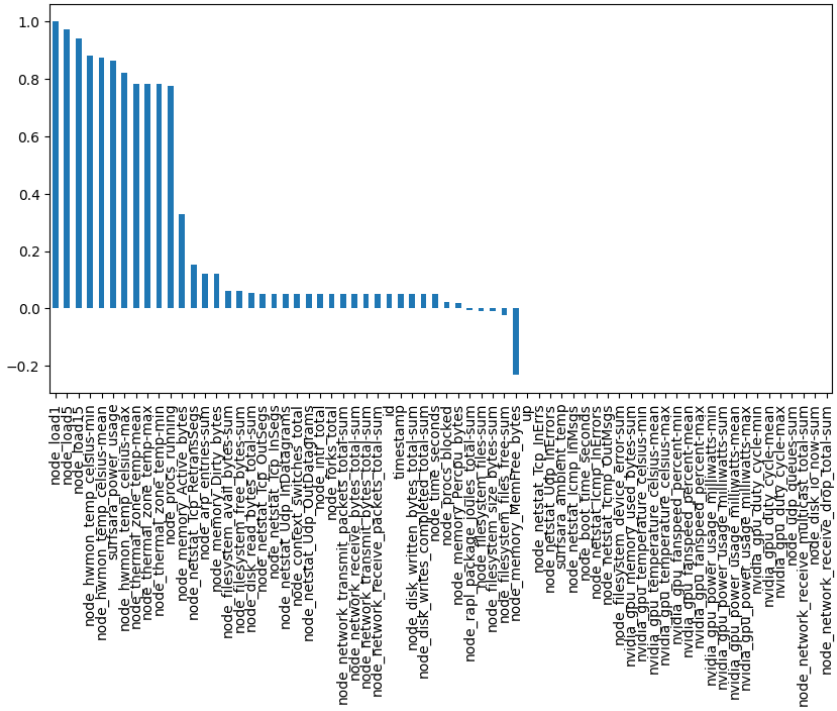


Figure 4.11: Spearman correlation for CPU utilization.

4.4.2.2 Power Usage Correlation

Figure 4.12 and 4.13 present the correlation matrices for metric pairs in the node dataset in relation to power usage. This analytical lens into the dataset brings to light several nuanced relationships and patterns.

From the entire set of 68 metrics, it's noteworthy that only a fraction, specifically 6, manifested a substantial linear dependency with power usage as highlighted by a Pearson correlation coefficient exceeding 0.8. Transitioning to the Spearman correlation method, a slightly broader set of 9 metrics emerged, exhibiting a pronounced monotonic bond with power consumption.

What captures attention is the common thread tying these metrics: they all pivot around node temperature and CPU utilization. This suggests a potential cascade effect. As CPU utilization intensifies or the node temperature escalates, power consumption experiences a concurrent surge. This interrelation underscores the delicate balance in computational systems. If CPU activity or temperature rises uncontrollably, it might trigger a disproportionate increase in power consumption, challenging the system's efficiency and sustainability.

In light of these findings, one could speculate that by implementing adaptive thermal controls or dynamic CPU task scheduling, it might be possible to modulate power consumption more efficiently. Recognizing these intricate relationships not only deepens our understanding of system behavior but also catalyzes the development of innovative strategies to refine power efficiency in data centers.

4.4.2.3 Correlated Pairs

To verify that the matrix pairs that show strong correlations are significant, we first verify all p-values of the pairs stated above. All p-values are less than 5×10^{-8}, which indicates that the results hold substantial statistical significance.

Based on the results of the correlation, These correlations suggest a reciprocal influence between CPU utilization and both power usage and node temperature. To be more specific, it shows the mutual impact that CPU utilization, power usage, and node temperature exert on each other.

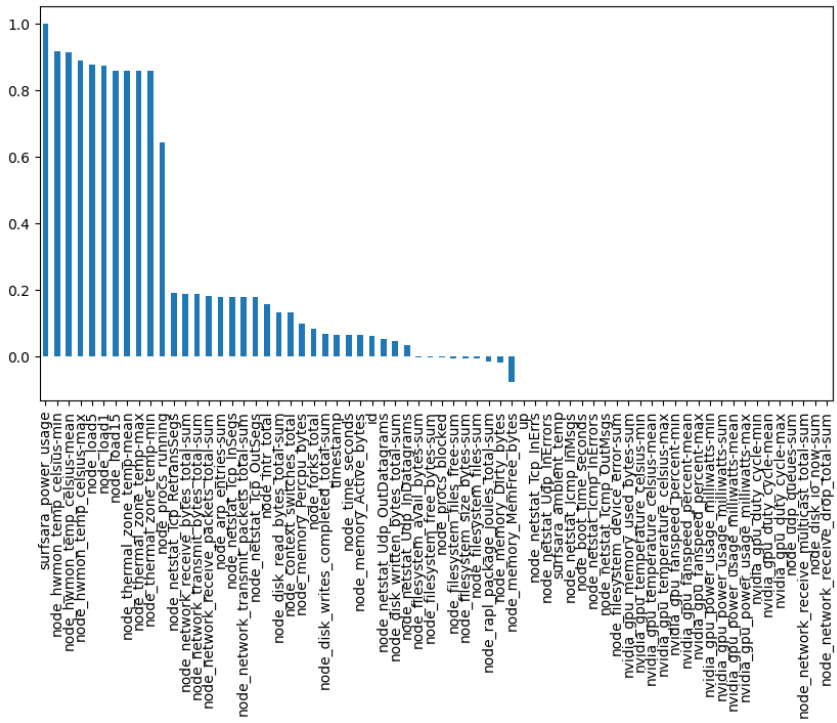


Figure 4.12: Pearson correlation for power usage.

As CPU utilization increases, it leads to a rise in power usage and node temperature due to higher energy consumption and heat generation. Conversely, alterations in power usage and node temperature can influence CPU utilization. For instance, in situations where node temperature escalates excessively, the cooling system would work and can decrease CPU utilization to avoid overheating, a process known as thermal throttling. Similarly, if power usage approaches its maximum capacity, the system might reduce CPU utilization to prevent overload. Thus, the relationship between these factors is characterized by reciprocal influence, where changes in one can instigate changes in others. It opens avenues for further examination into ways to investigate how to manage CPU utilization efficiently to control power usage and maintain optimal node temperature.

Moreover, the feature *'node_procs_running'*, which is a count of the currently running processes on the node, shows a strong correlation with both CPU utilization and power consumption as well (correlation values all above 0.6). A threshold of 0.6 was selected as a benchmark to denote strong correlation, based on domain-specific standards and the relevance of the relationship in the context of our study. From a CPU utilization perspective, a higher count of running processes would typically require more CPU resources, leading to higher CPU utilization. As processes demand computational power, the CPU responds by processing more tasks, which increases its utilization percentage. Thus, a strong positive correlation would be expected between the number of running processes and CPU utilization. From a power consumption standpoint, as the number of running processes increases, the demand for computational power increases, requiring more energy usage. More power is needed to support the CPU and other system components in processing and managing these tasks. Hence, there's also a strong positive correlation between the number of running processes and power consumption. This information can be useful in capacity planning and energy efficiency measures. For instance, if a node continually runs near its maximum process limit and experiences high CPU utilization and power consumption, it may be nearing its capacity. Such a situation could warrant an upgrade or redistribution of processes to other nodes. Similarly, identifying processes that cause significant increases in CPU utilization and power consumption

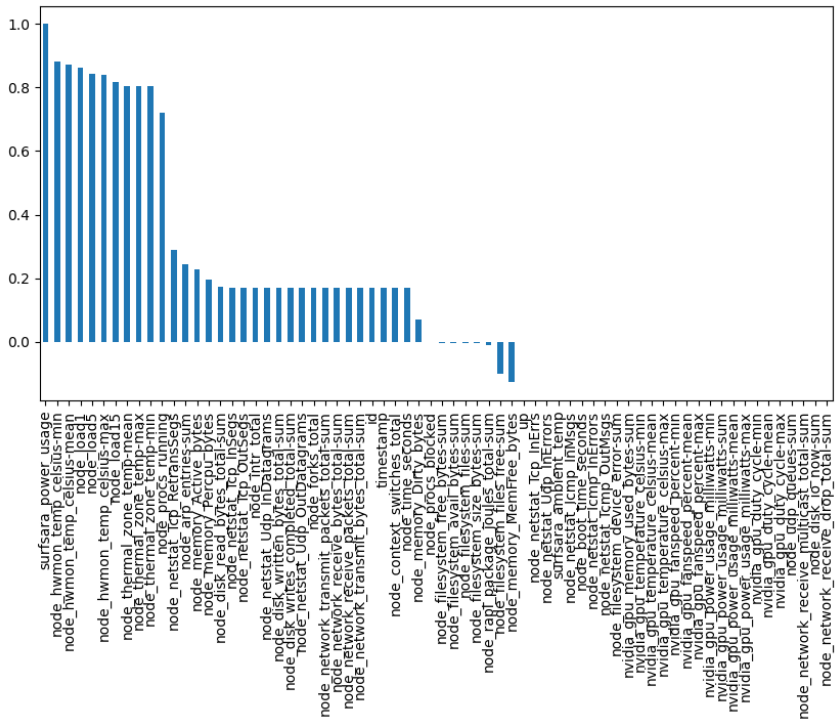


Figure 4.13: Spearman correlation for power usage.

can highlight opportunities for optimization, potentially reducing energy use and extending the node's effective capacity. Additionally, this correlation could inform predictive models or anomaly detection systems, helping identify potential issues like malfunctioning processes or hardware problems if CPU utilization or power consumption deviates significantly from expected levels given the number of running processes.

4.5 Temporal Dependence, Pattern, and Peak Analysis

In this section, we introduce the methods we use in section 4.5.1, and we show the results and insights in section 4.5.1. As the result in section4.4.2 indicates the high correlation between power usage and CPU utilization, we only focus on the power usage for the pattern and peak analysis.

4.5.1 Methods

In this study, we employ a combination of analytical methods—namely Auto-correlation [2], Discrete Fast Fourier Transform (DFFT) [66], and Continuous Wavelet Tree [13] —to carry out the comprehensive pattern and peak analysis. We use auto-correlation, to identify any recurring patterns within the data and to ascertain if past data values influence future ones. Then, we use the Discrete Fast Fourier Transform (DFFT) to discern the overarching frequency components, providing valuable insights into the overall behavior of the data. Lastly, the Continuous Wavelet Tree will provide a time-frequency representation of the signal and show the peak and pattern. This methodology is particularly advantageous for analyzing non-stationary signals, as it will offer crucial information about not only the frequency content of the signal but also the exact timing of these frequencies. By leveraging these powerful techniques in tandem, we aim to provide a robust and comprehensive analysis of the patterns and peaks present in the data, which will ultimately lead to more nuanced and insightful conclusions. Here we give a more detailed introduction to each method:

1. **Auto-correlation** [2] Auto-correlation is a statistical tool that measures the degree of similarity between a given time series and a lagged version of itself over successive time intervals. Instead of quantifying the relationship between two separate random variables, as in standard correlation, autocorrelation evaluates the relationship of a single random variable with its own past and future values.

   In time series analysis, autocorrelation can identify seasonality or periodic trends. It quantifies the strength and type of relationship between a variable and its historical values. Autocorrelation can expose the relationships between past and future values, aiding in trend identification and forecasting.

2. **Discrete Fast Fourier Transform(DFFT)** [66] The Discrete Fast Fourier Transform is an algorithm used for the computation of the discrete Fourier Transform (DFT) and its inverse. The DFT converts a sequence of values (typically a time series of measurements) from the time domain into the frequency domain. In simpler terms, it's a tool to extract frequency information from a time-based signal. The Fast Fourier Transform (FFT) is a version of the DFT that is computationally efficient.

   Understanding the frequency components of a signal offers critical insights into the nature and properties of the system or process that generated the signal. It can identify periodic components in signals, allowing for the recognition of recurring patterns and their respective frequencies, such as the fundamental frequency and harmonics in a sound signal.

3. **Continuous Wavelet Tree** [13] The Continuous Wavelet Transform (CWT) Peak Detection algorithm is a powerful tool used to detect and characterize transient, or short-lived, events in time-series data.

   In the context of peak detection, the CWT works by convolving the signal with wavelets of various frequencies and scales. A wavelet is a waveform of effectively limited duration that has an average value of zero. The key feature of wavelets is their ability to simultaneously localize a signal in both the time and frequency domain - the so-called uncertainty principle - which makes them ideal for peak detection in signals.

   In the CWT Peak Detection algorithm, wavelets are used to examine the original signal at different levels of resolution. At each scale, the wavelet transform identifies regions where the signal significantly matches the wavelet (peaks in the transformed modulus). Peaks in the wavelet-transformed signal correspond to distinctive features, or events, in the original signal.

4.5.2 Results and Insights

In this section, we begin by presenting our findings and insights from a holistic standpoint, considering the entire system as a unified whole.

4.5.2.1 Temporal Dependence and Predictability of Power Usage

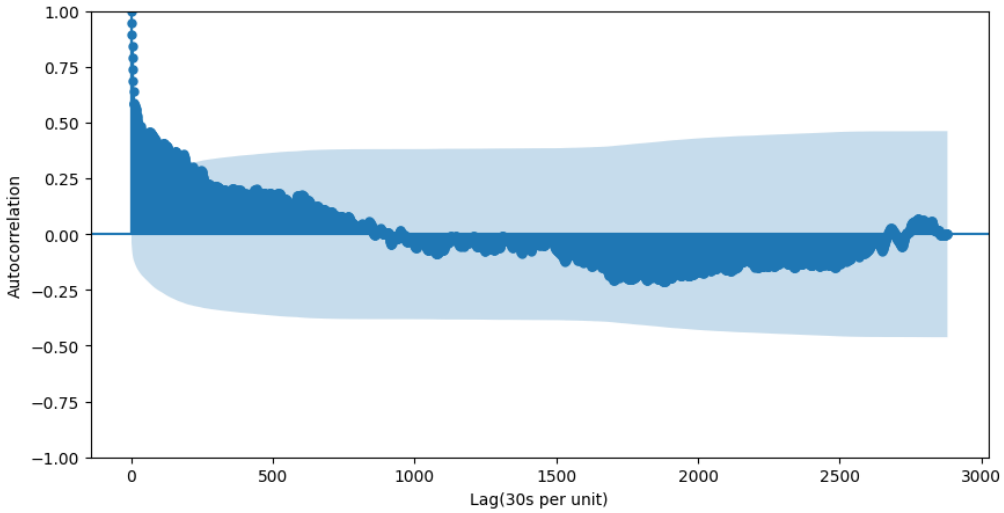


Figure 4.14: Autocorrelation for power usage.

In the auto-correlation figure as shown in Figure 4.14, the dark blue line represents the autocorrelation coefficient for each lag in the data. The autocorrelation coefficient measures the degree of similarity between a given time series and a lagged version of itself over successive time intervals. It ranges from -1 to +1, with +1 indicating perfect positive correlation, -1 indicating perfect negative correlation, and 0 indicating no correlation. The light blue shadow generally depicts the confidence intervals around zero, signifying the range where the autocorrelation coefficient is not statistically different from zero. If the dark blue line (the autocorrelation coefficient) falls within this area, it suggests that the autocorrelation at that particular lag may be due to chance, and is not statistically significant.

Each lag means 30 seconds, and tracking power usage at half-minute intervals over 24 hours would result in 2880 data points.

Here are some key observations from the autocorrelation figure:

O15: The sharp decrease observed in the dark blue region suggests that as lag time increases, there is a rapid decrease in its correlation with the original series. This highlights that the time series values are predominantly influenced by their immediate preceding values and have a diminishing relation with values from the more distant past.

O16: The trajectory of the line transitioning into the light blue-shaded region around 250 lags(around 1.5 hours) indicates a point where the autocorrelation ceases to be statistically significant. This suggests that any perceived correlation beyond this 1.5-hour point is potentially due to random variations, taking into account the defined confidence intervals.

To sum up, for the power usage in the system, the rapid decrease of the autocorrelation observed in the dark blue line as the lag time increases suggests that the given time series is primarily influenced by recent events, demonstrating a strong time dependency in the short term. However, the relationships with more distantly lagged observations become less significant, implying that historical data has less predictive power for the future values in this series. Furthermore, once the lag reaches around the 1.5-hour mark, the autocorrelation drops into the blue shadow, indicating that correlations beyond this point may not be statistically significant. This finding implies a potential boundary for effective predictive modeling based on autocorrelation, beyond which the level of certainty is insufficient to confidently predict future behavior. Hence, for this specific time series, any reliable prediction or pattern analysis should consider a time lag of up to approximately

1.5 hours. Beyond this lag, the relationships may not be reliable or could simply be random coincidences.

This result aligns with our understanding of HPC environments where power usage is largely event-driven, primarily triggered by incoming job requests. Consequently, the diminishing impact of historical data on the current state, as reflected in our autocorrelation analysis, is consistent with this operational nature. The history of power usage does not directly influence current power usage as it doesn't dictate the timing or the computational requirements of incoming jobs. Thus, the short-term dependencies observed up to approximately 1.5 hours could be related to the typical job duration or queueing behavior in the system, beyond which the power usage appears to be less predictable based on past behavior. This underlines the highly dynamic and responsive nature of power consumption patterns in HPC environments, reinforcing the need for real-time or near real-time monitoring and control strategies rather than relying on historical trends.

4.5.2.2 Patterns and Peak

1. **Power Usage Hour of the Day**

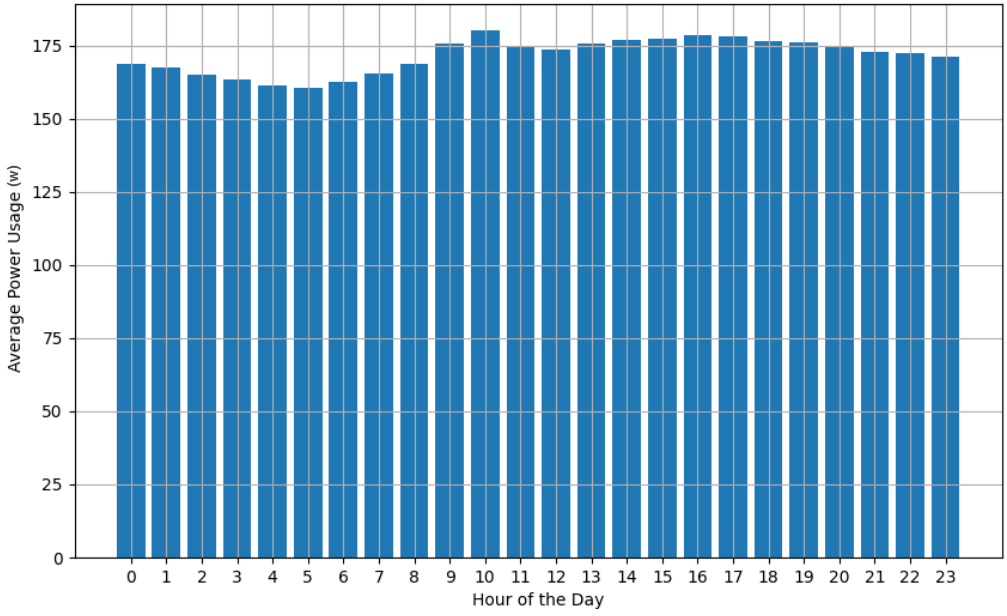


Figure 4.15: Power usage per hour of the day.

O17: An evident diurnal pattern characterizes the power consumption across all nodes, with discernible peaks and troughs.

O18: The power usage culminates at 10 a.m., and a consistent heightened consumption interval is observed from 9 a.m. to 7 p.m., potentially aligned with regular business operations or intensified computational activity.

O19: After 7 p.m., there is a consistent decline in power usage, culminating in the lowest consumption at 5 a.m.

In our endeavor to understand the dynamics of power usage, we first discern the power consumption patterns with respect to different hours of the day across all nodes, thereby pinpointing the periods of heightened power usage. The hourly power usage, illustrated in Figure 4.15, offers insightful revelations to this end.

We observe a clear diurnal pattern in power usage across all nodes, with the peak power consumption occurring at 10 a.m., and the lowest power consumption at 5 a.m., as shown in Figure 4.15. The period from 9 a.m. to 7 p.m. represents a window of high power usage, demonstrating a possible correlation with regular business hours or a time of heightened computational activity. Post 7 p.m., the power usage consistently declines, reaching its nadir at 5 a.m. These insights could be instrumental in strategizing for load balancing or implementing energy-saving measures during periods of lower power consumption. For example, following the elevated consumption phase, there's a systematic decline starting post 7 p.m., reaching its lowest point at 5 a.m., which suggests potential opportunities for strategic load management and energy conservation during these off-peak hours.

2. **Frequency Domain Analysis**

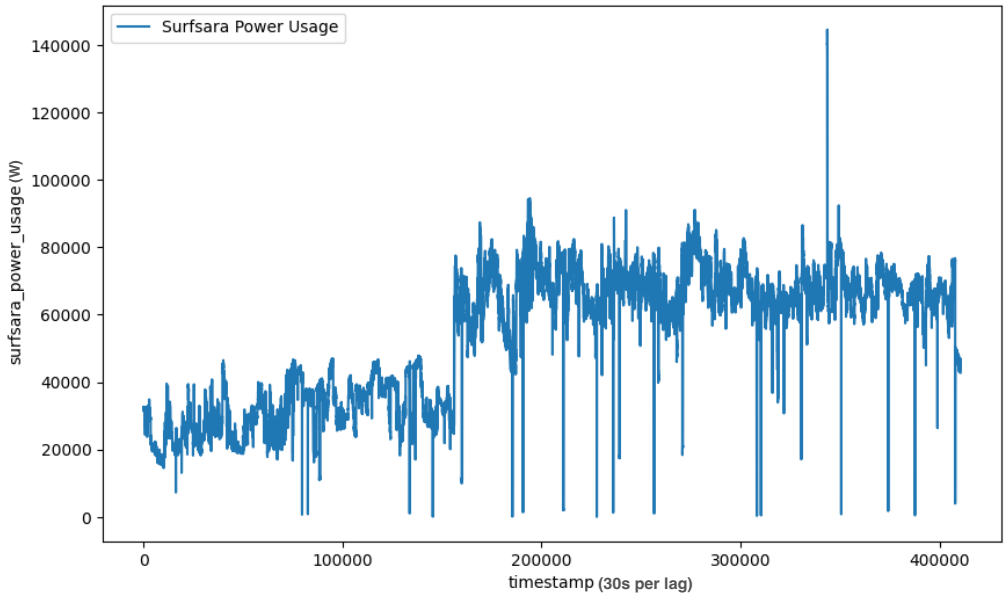


Figure 4.16: Power usage over time by 24 hours

Our study uncovers a rhythmic pattern in power consumption, an indicator of underlying computational and I/O operations. This discovery hints at the possibility of recognizing, and ultimately forecasting, these periodic patterns to aid in detailed scheduling decisions. To methodically discern the primary harmonics within this cyclic behavior, we employ a discrete fast Fourier transform (DFFT), translating the series into the frequency domain. We then identify prominent peaks with the assistance of a continuous wavelet tree (CWT) algorithm. Utilizing this methodology, we manage to isolate the first three peak periods and their respective amplitudes within the power consumption data.

Figure 4.16 discerns cyclical patterns can be observed. These patterns seem to manifest themselves as recurring 'peaks', which appear at regular intervals throughout the dataset. Such periodic fluctuations might suggest that there are underlying operational routines within the data center that repeat over time.

Moreover, the figure also reveals certain instances of 'shaking' - rapid, small-scale fluctuations in power usage. These could be the result of transient processes or instantaneous variations in the workload of the system, which cause short-term changes in power consumption.

The presence of these periodic peaks and small-scale fluctuations points to a complex interplay of processes within the data center, all of which contribute to the overall power usage.

By identifying and understanding these patterns, we hope to gain insights that could help optimize energy consumption within the infrastructure.

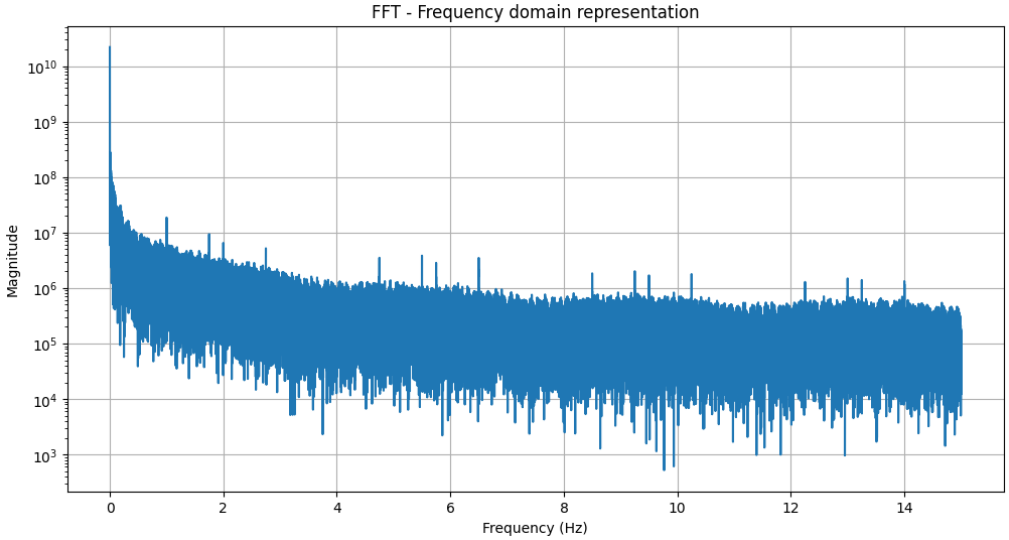


Figure 4.17: Frequency Domain(DFFT) of power usage.

The Figure 4.17 provided illustrates the frequency domain representation of power usage across our dataset. By transforming the original time-series data into the frequency domain, we are able to identify patterns and cycles that are not immediately apparent in the time domain.

On the x-axis, we have frequency, which denotes how often a certain pattern repeats within a given time frame. Lower frequencies represent patterns that repeat over longer intervals, such as daily cycles, while higher frequencies correspond to shorter intervals, such as hourly patterns.

The y-axis represents the amplitude or strength of each frequency in the dataset. A higher amplitude indicates that the corresponding frequency pattern is more prevalent or significant within the data. Peaks, or significant protrusions in the plot, represent the most dominant frequencies, implying the presence of strong, repeating patterns at these intervals.

By interpreting this figure, we can uncover key periodic behaviors in our power usage data, which can be instrumental in predictive modeling and identifying anomalies.

In our analysis, we focused on the top three patterns as these represent the most dominant cycles in the data. By concentrating on these primary frequencies, we aim to capture the most significant, recurring patterns in power usage. We notice the amplitudes drop fast after the top three frequencies. This selection helps to simplify the complexity of the time series while still preserving key periodic behaviors.

We first analyze the entire time series, rather than a subset or shorter duration, which allows us to account for all possible patterns and cycles that may occur. This holistic approach maximizes our understanding of the system's behavior over time, ensuring that any insights or patterns we identify are representative of the system as a whole, rather than being specific to a particular period.

Our Fourier analysis yields three dominant frequencies in the power usage data. These frequencies correspond to cyclic patterns that recur over different time intervals. The specific periods of these cycles were calculated by taking the reciprocal of each frequency and multiplying by 30 seconds, the time unit used in our dataset.

- The first frequency, 0.00029223886630802467 Hz, corresponds to a cycle that repeats approximately every 1.32 days. The amplitude of this frequency, 1327846457.924892, indicates the strength of this cycle in the data. This suggests a diurnal pattern that is slightly offset from the typical 24-hour day-night cycle.
- The second frequency, 0.0007305971657700617 Hz, corresponds to a cycle that repeats approximately every 0.5 day or 12 hours. Its amplitude is 627682824.5249662. This could represent a twice-daily pattern in the data.
- The third frequency, 0.0015342540481171295 Hz, corresponds to a cycle that repeats approximately every 5.42 hours. Its amplitude is 626185142.6696367. This could represent a more frequent pattern in the data.

These findings provide a foundation for a deeper understanding of power usage in our HPC environment and could inform strategies for more efficient energy management.

4.6 Summary

In this chapter, we first did the requirement analysis and then we delved deeply into the nuances of resource and energy consumption, transitioning seamlessly from a broad perspective to intricate node-specific insights revealing vital insights, and emphasizing both conventional and ML-centric facets of data center functioning. Further, we explored the matrix's correlations. Our subsequent temporal and pattern analyses illuminated patterns in 24 hours and three significant recurring consumption trends using auto-correlation, DFTT, and continuous wavelet tree techniques.

Chapter 5

Modeling Power Usage and CPU Utilization

The characterization work provides an understanding of Power Usage and CPU Utilization. This chapter, we dedicate to addressing Research Question 2 (RQ2): How to Design an Accurate and Efficient ML Model for Data Center Resource and Energy Estimation and Prediction?

Initially, Section 5.1 will provide an in-depth analysis of the requirements necessary for the modeling of power usage and CPU utilization. Subsequently, the intricacies of the model design will be described in Section 5.2.

5.1 Requirement Analysis

The modeling process is split into two essential components. The initial aspect revolves around the real-time estimation of power usage, providing an immediate snapshot of the data center's energy consumption. Previously, the estimation of power usage primarily relied on basic statistical models, which employed simple calculations. These models, though straightforward and easy to understand, often struggled to provide accurate estimates due to their inability to effectively incorporate the complex interdependencies among various data center parameters. The application of machine learning methods has emerged as a promising approach to address these challenges.

The subsequent facet is centered on the prediction of CPU utilization and power usage, leveraging past and present data to forecast future CPU utilization and energy consumption. With the rapid expansion and complexity of data centers, relying solely on current utilization statistics is not sufficient. Real-time data, while vital, provides only a snapshot of the current system state. To ensure smooth operations and prevent system overloads or under-utilization, it's essential to anticipate future demands. In this context, predicting future CPU utilization and power usage becomes critical.

5.1.1 Functional Requirements

- **FR1. Real-Time Estimation:** The ML model should be able to estimate real-time power usage.
- **FR2. CPU Utilization Prediction:** The ML model should be capable of predicting the CPU utilization on the node level. The model should factor in the past and present CPU utilization trends, workload patterns, and system configurations to generate accurate forecasts. This can aid in effective resource allocation, load balancing, and preventing system overloads.
- **FR3. Power Usage Prediction:** The model should predict future power usage patterns based on the current and historical usage data.

5.1.2 Non-Functional Requirements

- **NFR1. Accuracy:** The ML model should have a high degree of accuracy in both estimation and prediction tasks. The model should be trained and validated on representative datasets to ensure its reliability.
- **NFR2. Efficiency:** The model should provide results quickly, ideally in real-time or near real-time. The algorithms should be optimized for efficiency to ensure timely response and decision-making. The system should be capable of handling large volumes of data without any significant delays or computational bottlenecks.
- **NFR3. Scalability:** The system should be scalable to manage and analyze data from large data centers. It should be capable of processing and analyzing large volumes of data and should be designed to scale up or down based on the load and the size of the data center.
- **NFR4. Reliability:** The system must be highly reliable, given that data center management often deals with mission-critical operations. It should provide robust error handling and fault tolerance capabilities to prevent disruptions in its operations.
- **NFR5. Interoperability:** The system should be able to interface effectively with existing data center management systems. It should adhere to standard data formats and communication protocols to ensure seamless integration with the existing IT infrastructure.
- **NFR6. Adaptability:** The system should be adaptable to changing needs and conditions. It should be designed to accommodate changes in workload patterns, technology advancements, or shifts in organizational priorities and business objectives.

5.2 Methodology and Modeling

In the following, we propose our estimation modeling for power usage and prediction modeling for both power usage and CPU utilization. Our study involves two distinct models for output generation.

The first one, referred to as the Estimation Model, is tasked with estimating power usage based on the current time CPU utilization. This model serves as a direct translator, transforming the present state of CPU utilization into a corresponding power usage value.

Firstly, the power consumption estimation model is a critical component in data center resource management. The existing approach for CPU utilization estimation relies on the formula of CPU speed and frequency and has proven to be reliable. However, the existing power usage model based solely on CPU utilization lacks the necessary accuracy required for efficient power management. Currently, a simple statistical model is used for power usage estimation, which can not capture the complex and dynamic relationships between CPU utilization and power consumption accurately.

The problem at hand is to design and develop an advanced power consumption estimation model that surpasses the limitations of the existing simple statistical model. The objective is to create a more sophisticated and robust model capable of providing highly accurate power consumption predictions, thereby facilitating proactive resource allocation, energy optimization, and overall data center performance enhancement.

The second prediction modeling part forecasts the next several lags in the sequence. We use multi-index forecasting for energy prediction, targeting both power consumption and temperature. For resource allocation, our focus is on predicting CPU utilization. We start from four lags for forecasting. The choice of predicting the next four lags is motivated by our characterization study, which revealed that these future intervals are highly relevant and can contribute significant information to the prediction process. By providing insight into the near future states, this model aids in efficient resource allocation and effective system management.

The prediction modeling entails leveraging historical and real-time data to forecast future power usage and CPU utilization patterns. This approach allows us to capture the dynamic and evolving

nature of resource demands within the data center, providing administrators with valuable insights into potential capacity issues and energy usage trends. By incorporating sophisticated machine learning techniques, we aim to build predictive models that can effectively handle the complexities inherent in time series data, thus enabling accurate and reliable predictions.

In the subsequent sections, we detail the design of the modeling.

5.2.1 Model Selection

In the pursuit of designing an accurate and efficient model for CPU utilization and power consumption, we carefully consider various machine learning algorithms. Two models stand out as promising candidates for achieving the desired forecasting capabilities: Long Short-Term Memory (LSTM) networks and Transform models.

When estimating and predicting intricate dynamics like CPU utilization and power consumption, we require models that are both versatile and powerful. Our choice of LSTM networks and Transform models is rooted in several key considerations:

1. **Temporal Dependencies:** Both LSTM and Transform models excel at modeling sequences, crucial for time series data. LSTMs, with their unique memory cell structures, can remember and retrieve information over long intervals, making them adept at capturing long-term dependencies in time series data. Transform models, on the other hand, use attention mechanisms that weigh the importance of different time steps, allowing them to discern and prioritize crucial patterns over others.

2. **Complexity of Relationships:** The nonlinear nature of CPU utilization and power consumption requires models that can capture intricate relationships. LSTM and Transform models are inherently nonlinear, offering an edge in modeling the complex interplay of variables in our data.

3. **Scalability:** As data centers grow and the volume of data surges, scalability becomes paramount. Transform models, particularly, have showcased remarkable scalability, handling large datasets with ease, while LSTMs can be efficiently trained with the right optimizations.

4. **Proven Track Record:** Both models come with a storied legacy of successes. LSTMs have been the go-to for numerous sequence prediction tasks, from speech recognition to financial forecasting. Transform models, since their introduction, have revolutionized several domains, especially natural language processing, and their prowess is now being recognized in time series forecasting as well.

5. **Complementary Strengths:** While both models are strong contenders individually, they possess complementary strengths. LSTMs thrive in scenarios where sequential memory is vital, while Transform models leverage their attention mechanisms to give prominence to specific impactful events. This makes the comparison between their performances even more riveting, as it can shed light on which attributes of the data are more predictive.

In essence, while there are myriad models available, our choice of LSTM and Transform models is a confluence of their proven capabilities, their alignment with the attributes of our data, and the potential insights that arise from juxtaposing their performances. In our quest for efficient forecasting in data center resource management, these models not only offer promising results but also illuminate the underlying dynamics of CPU utilization and power consumption.

5.2.1.1 Long Short-Term Memory(LSTM) Model

Long Short-Term Memory, or LSTM, is a type of Recurrent Neural Network (RNN) specifically constructed to process sequential data with intricate long-term dependencies [73]. Because of

their capability to capture temporal dynamics and hold memory over elongated periods, LSTM networks have found substantial use in time-series forecasting tasks.

A traditional RNN, while powerful for sequential data, suffers from issues known as vanishing and exploding gradient problems. The vanishing gradient problem occurs when the gradients, which play a pivotal role in updating weights during backpropagation, drastically reduce in magnitude as they move back through the network. This scenario often culminates in subpar weight updates, leading to the model's inability to learn long-range dependencies. Conversely, the exploding gradient problem is an event where these gradients dramatically increase as they traverse through the network, potentially causing unstable learning and exaggerated weight updates.

LSTM networks alleviate these issues by incorporating a memory cell along with a trio of gates: the input gate, output gate, and forget gate. These components collectively facilitate the preservation and manipulation of information over extensive periods without losing significant context. They essentially control the flow of information in and out of the memory cell, which allows the network to maintain relevant information while discarding the less significant details.

The memory cell sustains its state over time, allowing it to carry forward the context, and its value is updated selectively. The input gate decides the extent to which incoming information should update the memory cell, the forget gate determines what proportion of the memory should be retained, and the output gate decides what the next hidden state should be. As a result, LSTM networks can learn to bridge minimal or substantial time lags, adaptively ignoring or remembering information in the input sequences.

LSTM networks have shown impressive performance in a wide array of sequential data prediction tasks, providing particularly significant value when dealing with time-series data characterized by intricate and nonlinear patterns. These traits make them a strong choice for complex predictive modeling tasks that require understanding long-term dependencies in data.

5.2.1.2 Transform Model

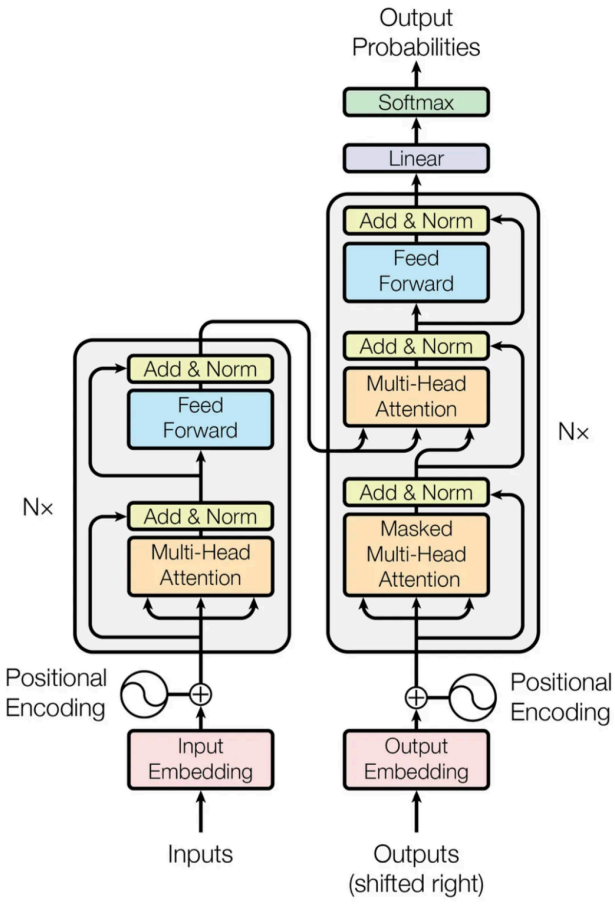


Figure 5.1: Transformer architecture [67]

As shown in Figure 5.1, it is a typical transformer model founded on an encoder-decoder structure, which utilizes attention mechanisms extensively [25]. Transform models have recently emerged as powerful alternatives for time-series forecasting tasks. The Transformer model was devised to surpass the constraints inherent in LSTM and RNN, such as struggling with very long sequences or when the important information is located far from where it is needed in the sequence. The Transformer consists of two primary elements: the encoder block, which scrutinizes the input data to construct an exhaustive context, and the decoder block, which leverages this context to formulate the output sequence. These models are based on the idea of self-attention mechanisms, enabling them to focus on relevant time steps and identify important patterns in the data. The attention mechanism allows the model to weigh the significance of different temporal points, resulting in more effective feature extraction and representation. Transform models have gained popularity for their ability to handle long-range dependencies in time-series data efficiently, making them a compelling choice for accurate power consumption prediction.

5.2.2 Model Design Overview

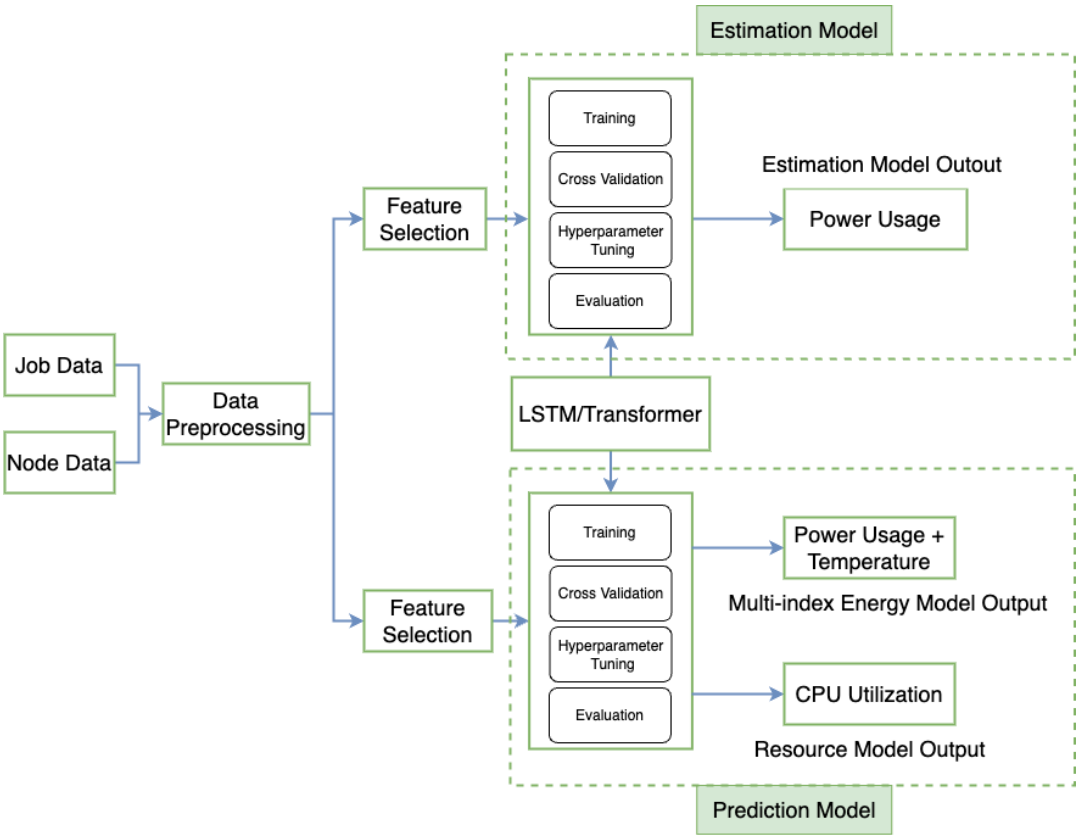


Figure 5.2: Modeling design overview

The design overview is visually presented in Figure 5.2, which depicts the distinct yet interconnected frameworks of the Estimation Model and the Prediction Model. These models, although different in their final objectives, follow a shared path in their initial stages of data utilization and preprocessing.

The procedure starts with the collection of historical data, encompassing both job-specific and node-related information. This data is subjected to a meticulous preprocessing phase, which includes a systematic approach to data cleaning, transformation, and integration. These steps are elaborated upon in Chapter 3 of the thesis.

Both the Estimation and Prediction Models engage in a process of feature selection and pick different features that serve different purposes. This process is crucial for optimizing the performance of the models and effectively managing dimensionality. Feature selection not only reduces the computational complexity but also enhances the predictive power of the models by eliminating redundant or irrelevant features.

After feature selection, the models are trained using the prepared datasets. Cross-validation is employed at this stage to assess the models' ability to generalize on unseen data and to prevent overfitting. Additionally, the models' hyperparameters are meticulously tuned to optimize their performance, striking a balance between bias and variance.

As for the output, both models function at the node level. The Estimation Model is designed to generate an output encompassing Power Usage. The Prediction Model, which utilizes a different modeling framework, produces energy multi-index predictions for Power Usage along with Temperature; For resource prediction modeling, the focus is on predicting CPU Utilization. This distinction in outputs stems from the unique objectives of the two models, catering to different

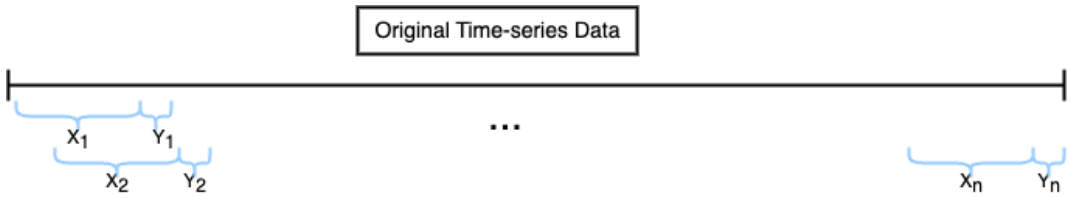


Figure 5.3: Sampling from the original time-series using moving window.

aspects of the system's operational metrics.

In the following sections, we delve into a more detailed exploration of the modeling process.

5.2.2.1 Job Feature Integration

For a comprehensive understanding of the system's behavior, we have integrated both job and node data. The job data contributes unique insights into the system's workload dynamics and can augment the prediction and estimation capabilities of our model.

The integration proceeds as follows:

- **Number of jobs submitted/started:** This metric indicates how many jobs have been initiated within the last 30 seconds. It provides insight into the inflow of tasks and can suggest periods of high activity or load on the system.
- **Number of jobs running:** This quantifies the number of jobs actively executing during the last 30 seconds. It illustrates the ongoing processing load and can potentially highlight bottlenecks in the system.
- **Number of jobs completed/failed/canceled/timeout/out of memory/node fails:** These parameters report the number of jobs in the last 30 seconds that have either completed successfully, failed, got canceled, timed out, run out of memory, or led to node failures. These metrics are critical to understanding the system's robustness, efficiency, and potential issues.

By utilizing these job-based features in conjunction with node data, we aim to create a holistic modeling approach that accounts for both the node level and the operational characteristics (job data) of the system. This integration is expected to lead to a more accurate and reliable estimation and prediction model.

5.2.2.2 LSTM Model for Estimation and Prediction

In an endeavor to enhance the learning process and elevate model performance, we segment the original time-series data into smaller sub-samples using a sliding window approach. As illustrated in Fig. 5.3, the sliding window moves systematically through the primary time-series data, generating a series of overlapping sub-samples. These sub-samples not only allow the model to exploit localized temporal information more effectively but also increase the quantity of training data, thereby enhancing the model's ability to generalize and capture dynamic temporal dependencies. The size of the sliding window is crucial in this process, as a wider window may introduce noise or irrelevant information, while a narrower window could potentially overlook longer-term dependencies. Therefore, optimizing the window size is a critical aspect of our approach.

Let L represent the sequence length. Then, the t^{th} input and output of the LSTM network are given by the sequence $(x_t, x_{t+1}, ..., x_{t+L-1})$ and $y_{t+L} = x_{t+L}$, respectively. Here, L is a positive integer that signifies the number of prediction steps ahead. Additionally, T is the total count of subsamples, which is determined by the length of the original time series and the sequence length L.

For the estimation and prediction, the LSTM network graph displays the sequential nature of the model, with multiple LSTM layers capturing the temporal patterns in the input data. The

model receives historical power consumption data as input, and through the recurrent connections of the LSTM cells, it learns to extract essential temporal features and make accurate predictions for future power usage.

During each timestep, the LSTM cell ingests two key pieces of information: the corresponding input for that particular timestep, and the hidden state values from the preceding timestep. Additionally, the final hidden values from the last LSTM cell are utilized as input for the ensuing dense layer. To extrapolate the system's future behavior, we can employ a dense layer formulated as follows:

$$y_t = W_d h_t + b_d \tag{5.1}$$

Here, W_d and b_d denote the weights and bias term within the dense layer, respectively, and h_t represents the hidden state from the final LSTM cell. It's worth noting that the dimensionality of the hidden state corresponds to the predefined number of neurons. Importantly, the final hidden state values incorporate information from all previous inputs as they are a function of both the cell memory and the output gate.

An LSTM layer consists of a set of recurrently connected blocks, known as memory blocks. These blocks can be thought of as a differentiable version of the memory chips in a digital computer. Each one contains one or more recurrently connected memory cells and three multiplicative units - the input, output, and forget gates - that provide continuous analogs of write, read and reset operations for the cells. More precisely, the input to the cells is multiplied by the activation of the input gate, the output to the net is multiplied by that of the output gate, and the previous cell values are multiplied by the forget gate. The net can only interact with the cells via the gates.

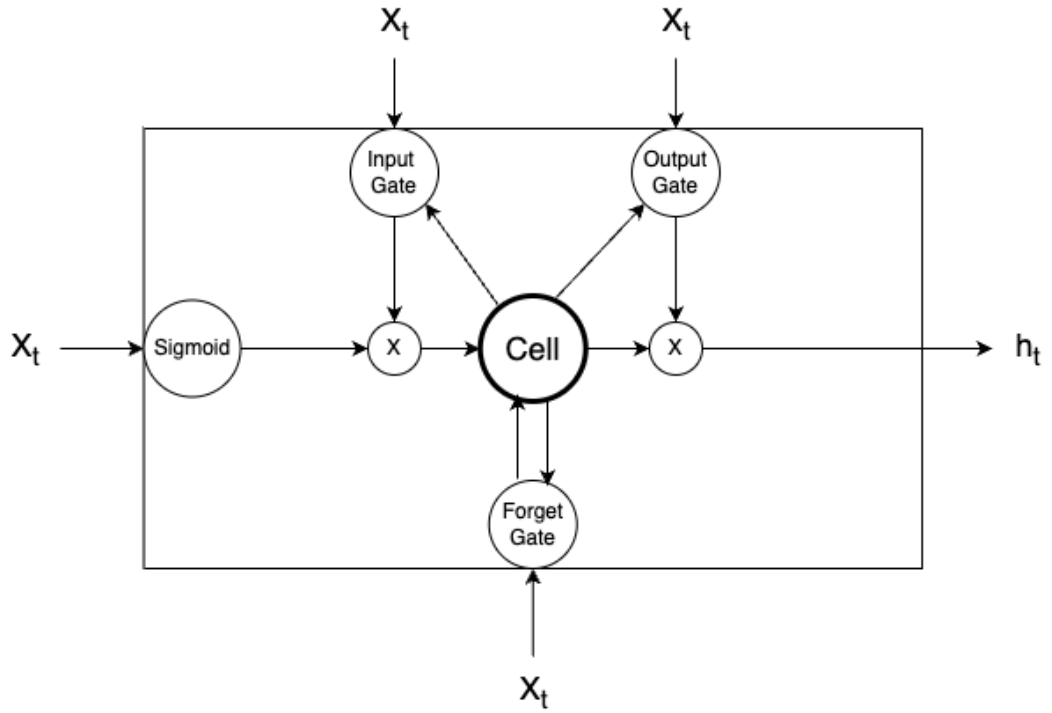


Figure 5.4: LSTM unit structure [25]

For our model for estimation and prediction, as the LSTM unit structure shown in Figure 5.4, the regulation of the gates is achieved by the Sigmoid neural network layers, which are tasked with deciding the information that is stored, discarded, and output from the memory cell. The input gate manages the admission of new information into the memory cell. The forget gate determines which information is no longer needed and thus is removed from the memory cell.

Finally, the output gate oversees the delivery of information from the memory cell to the network output. This configuration effectively manages the information entering and exiting the memory cell. These gates operate under the control of Sigmoid neural network layers, which determine what information is stored in the memory cell, what is discarded, and what is to be output. They consist of memory cells with gates that regulate the flow of information, allowing them to learn from historical data and make predictions for the future.

5.2.2.3 Transformer Model for Estimation and Prediction

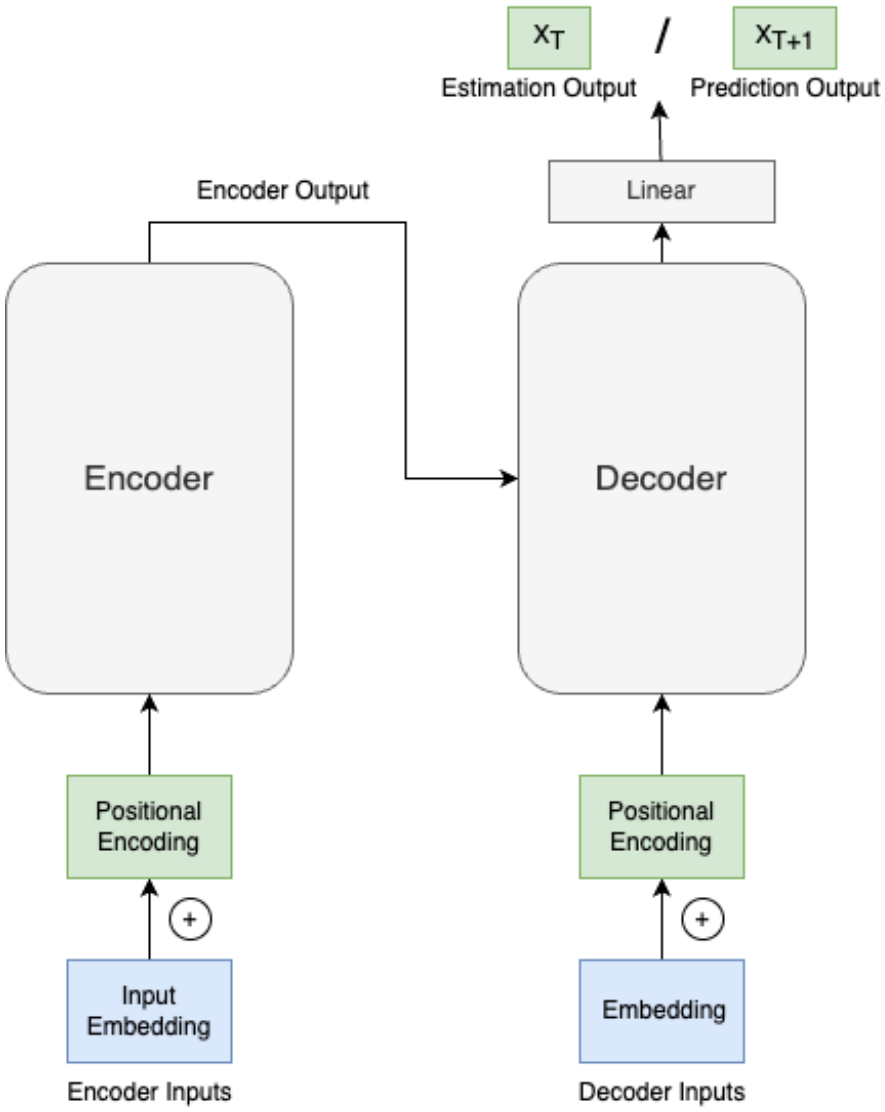


Figure 5.5: Transformer model design in this work

The design of our Transformer model, tailored for our time-series forecasting, is illustrated in Figure 5.5. The architecture seamlessly integrates the principles of attention mechanisms with traditional feed-forward networks, culminating in a powerful predictive model.

Input Embedding and Positional Encoding The raw input data, a sequence of values, first undergoes a transformation into a dense vector representation using the `nn.Linear()` layer.

This transformation serves as an embedding mechanism, converting rudimentary data points into meaningful vectors.

To ensure the model retains an understanding of the inherent sequence order—vital for time-series data—each embedded vector is enriched with a positional encoding. This encoding is essentially a unique signature assigned to each sequence position, enabling the model to discern temporal patterns.

Encoder: Context Representation The heart of the Transformer model, the encoder, receives the positionally encoded embeddings. It comprises a series of layers that apply self-attention mechanisms and feed-forward networks. The encoder's primary objective is to discern intricate patterns within the sequence and encapsulate this information into a compact context representation. This representation carries the essence of the input data, abstracting key patterns and relationships for subsequent decoding. It's comprised of multiple identical layers, with each layer consisting of a self-attention mechanism and a fully connected feed-forward network. Input embeddings are used to transform the input into dense, low-dimensional vectors. To accommodate both the content and its position in the sequence, positional embeddings are integrated with the input embeddings.

Our implementation leverages the `torch.nn.TransformerEncoderLayer`, which inherently encapsulates both the self-attention and feed-forward components.

Decoder: Context Interpretation While the encoder focuses on pattern abstraction, the decoder's role pivots towards leveraging this abstracted context for precise predictions. It possesses its set of embedded inputs, which, similar to the encoder, are enriched with positional encodings.

Mirroring the encoder's layers, the decoder employs a combination of self-attention, feed-forward networks, and an additional attention layer tailored toward the encoder's output. This specialized attention layer allows the decoder to make informed predictions by emphasizing relevant parts of the encoded context.

Upon interpreting the context, the decoder's output is channeled through a final `nn.Linear()` layer. This fully connected layer serves as a mapping mechanism, converting the decoder's high-dimensional output into our desired forecast size, thus producing the final prediction.

The *Self-Attention Mechanism* within the encoder allows each element in the input sequence to consider all other elements, applying relevance-based weights. This mechanism is operationalized as a dot-product operation involving a query vector, a key vector, and a value vector for each sequence element. The outputs of the attention and feed-forward layers are then amalgamated and layer normalization is applied in the "Add and norm" operation.

The *Feedforward Network* within the encoder is a straightforward neural network with fully connected layers, which is utilized to learn non-linear transformations.

Decoder Block: Structurally similar to the encoder but equipped with an additional attention mechanism, the decoder transforms the encoded sequence into the output sequence. It employs attention over both the encoder outputs and the decoder inputs. The decoder is constructed from several identical layers, each layer featuring a self-attention mechanism, an attention mechanism over the encoded sequence, and a fully connected feed-forward network.

5.3 Summary

In this chapter, we initiated our discussion with a requirement analysis for energy and resource modeling in HPC data centers. Subsequently, the LSTM and Transformer models are our chosen ML frameworks and meticulously crafted the modeling architecture for our experiments, encompassing both estimation and prediction. An in-depth exploration of these two models is also presented within the chapter.

Chapter 6

Experiments and Evaluation for Power Usage and CPU Utilization Modeling

In this chapter, we tackle our third research question: "How can we evaluate the ML Model to comprehend resource utilization and energy consumption in large-scale data infrastructures?" We outline the experimental setup and procedure in Sections 6.1 and 6.2, respectively. Following that, we introduce the evaluation metrics in Section 6.3. We then present our experimental findings and delve deeper into model comparisons and discussions in Sections 6.4 and 6.5.

6.1 Experiment Setup

Our experimental setup utilizes the dataset drawn from the LISA System. The specifics of this dataset, including its description and the preprocessing steps undertaken, are delineated in the work of characterization. We conducted our experiment on Node r11n. To ensure the robustness of our modeling, we test our model in different time intervals. The first dataset is from June 2022 to November 2022. The second dataset is from January 2020 to August 2020. This aimed to evaluate the performance consistency of our models across various temporal phases. By employing this methodology, we were better positioned to assess if the performance metrics were inherently linked to specific timeframes or if they remained consistent irrespective of the time period under consideration. Furthermore, the systematic partitioning of the dataset into different time windows aided in elucidating any potential periodic trends or seasonality effects that could be present. This is paramount in understanding whether our models can adapt and predict effectively across varied time frames or if they display an affinity towards specific periods. The preprocessing steps, which were imperative for refining the dataset and ensuring its readiness for model ingestion, included normalization, outlier detection, and handling of any missing values. These crucial steps enhanced the quality of the data and ensured a seamless transition into the modeling phase, thus maximizing the chances of obtaining reliable and generalizable results. By testing the models on different time periods from the same data source, we aimed to confirm the stability and reliability of our chosen algorithms, thereby enhancing the scientific rigor and validity of our experimental findings.

We begin by predicting four steps ahead. To ensure a robust evaluation of our models, we implemented cross-validation on our dataset. Time series cross-validation is distinct due to the sequential nature of the data. Unlike standard cross-validation techniques where data is randomly sampled, time series cross-validation maintains the temporal order. Initially, the model is trained on an early segment of the data and tested on a subsequent segment. In the next step, the test segment from the previous step is added to the training set, and the model is tested on the next segment. This "rolling" process continues, ensuring that the model always predicts future points

Table 6.1: Grid search for the Transformer. The underlined values represent the hyperparameters used in this research.

Hyperparameters	1	2	3
d model	512	256	128
Attention heads	8	4	2
Hidden units	2048	1024	512
Batch size	64	48	32
Dropout rate	0.05	0.25	0.5

Table 6.2: Grid search for the LSTM. The underlined values represent the hyperparameters used in this research.

Hyperparameters	1	2	3	4
Number of layers	6	4	2	1
Hidden units	2048	1024	512	256
Batch size	64	48	32	26

based on past data.

For our initial experiments, both the LSTM and Transformer models were set up with a learning rate of 0.1, a batch size of 32, and the Adam optimizer. A dropout rate of 0.05 was used to mitigate underfitting, based on empirical findings from preliminary tests. Given the complex nature of our data and the models, hyperparameter tuning was essential. These experiments were evaluated using a validation set, with the goal of minimizing validation loss. In the context of the Transformer architecture, the d model refers to the dimensionality of the embeddings. This is a core hyperparameter in the Transformer design, influencing both the size and the capacity of the model. The LSTM model was further configured with a sequence length of 4 which means the model was trained to use the past 4 time steps of data to estimate or predict the power usage.

The hyperparameters for both the LSTM and the Transformer were optimized through experimental trials and grid search methods. The configuration grid for the LSTM and the Transformer can be viewed as below 6.1 6.2.

The subsequent section is dedicated to detailing the specific procedures implemented throughout our experiments.

6.2 Experiment Procedure

The sequence of steps adhered to during our experiment is outlined as follows:

1. **Data Preprocessing:** As explained in Chapter 4, the necessary preprocessing steps were undertaken on our dataset. This included the treatment of missing values, outlier detection, and feature scaling. Additionally, categorical variables underwent one-hot encoding, thereby transforming them into a machine-readable format.

2. **Model Building:** We leveraged the PyTorch library, a popular open-source machine learning framework, to construct our models. For the Transformer model, we built upon PyTorch's existing implementation, adjusting the number of encoder and decoder layers, the size of input embeddings, and the number of heads for multi-head attention as described in the "Model Design" section. For the LSTM model, we utilized PyTorch's LSTM module and customized the number of LSTM layers and the dimension of the hidden state according to our design.

For both models, the Adam optimizer was selected for its adaptive learning rate properties, which can help expedite convergence during training. As our task is prediction-based, we employed the Mean Squared Error (MSE) loss function, a common choice for regression problems. The MSE loss calculates the average squared difference between the actual and predicted values, providing a measure of prediction error. The models were compiled with these configurations, setting the stage for the training phase.

3. **Model Training:** Both models were trained using the training dataset. During this phase, the models' weights were adjusted based on feedback from the backpropagation process and the Adam optimization algorithm. We monitored the models' performance on the validation set at the end of each epoch, tuning hyperparameters and guarding against overfitting. The training was halted when the validation set performance started to degrade, signaling the start of overfitting.

4. **Model Evaluation:** Upon completion of training, the model's performance was evaluated on the unseen test set, which provided an unbiased estimation of the model's ability to generalize to new data. Additionally, we compare our results with the baseline statistical model in OpenDC employing a linear method, thereby gauging the extent of improvement achieved by our models.

5. **Hyperparameter Tuning:** We employed a validation set for hyperparameter tuning. Parameters such as the learning rate, dropout rate, batch size, and number of layers in the Transformer and LSTM models were optimized to enhance model performance.

6. **Result Analysis:** The results were analyzed based on the chosen evaluation metrics. This analysis allowed us to gain insights into how well our models were performing and identify areas for improvement.

6.3 Evaluation Metrics

To assess the performance of our models, we adopted two primary metrics: Root Mean Squared Error (RMSE) and Relative Error (RE). Both of these metrics are standard for regression tasks, providing insight into the model's prediction accuracy.

1. **Root Mean Squared Error (RMSE)**

   The RMSE is the square root of the MSE. The MSE measures the average of the squares of the errors, that is, the average squared difference between the estimated values and the actual value. A lower MSE indicates that our model's predictions are closer to the observed data, which is the desired outcome. The formula for calculating MSE is given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{6.1}$$

   where Y_i represents the actual value, $\hat{Y}_i$ represents the predicted value, and n is the total number of data points.

   For the RMSE, it has the advantage of using the same units as the quantity being estimated, making it easier to interpret in terms of the magnitude of the error. Similar to MSE, a lower value of RMSE is better as it indicates a closer fit to the data. The RMSE is calculated as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \tag{6.2}$$

2. **Relative Error (RE)**

   Relative Error is a metric that provides insight into the relative accuracy of a predictive model, expressed as a percentage of the actual values. It quantifies the difference between the predicted and actual values relative to the actual value. The formula for calculating the relative error for a single data point is:

$$RE = \frac{|\hat{y} - y|}{|y|} \tag{6.3}$$

   Where:

   - $\hat{y}$ represents the predicted value.
   - y represents the actual value.

6.4 Experiment Results

In this section, we detail our estimation and prediction outcomes in Section 6.4.1 6.4.2. Subsequent insights and discussions can be found in Section **??**.

6.4.1 Estimation Results

Model	**RMSE**	**RE**
LSTM	0.923	0.39%
Transformer Model	0.954	0.4%
Statistical Model	38.02	21.1%

Table 6.3: Power Usage estimation performance comparison between Transformer, LSTM models, and baseline statistical model

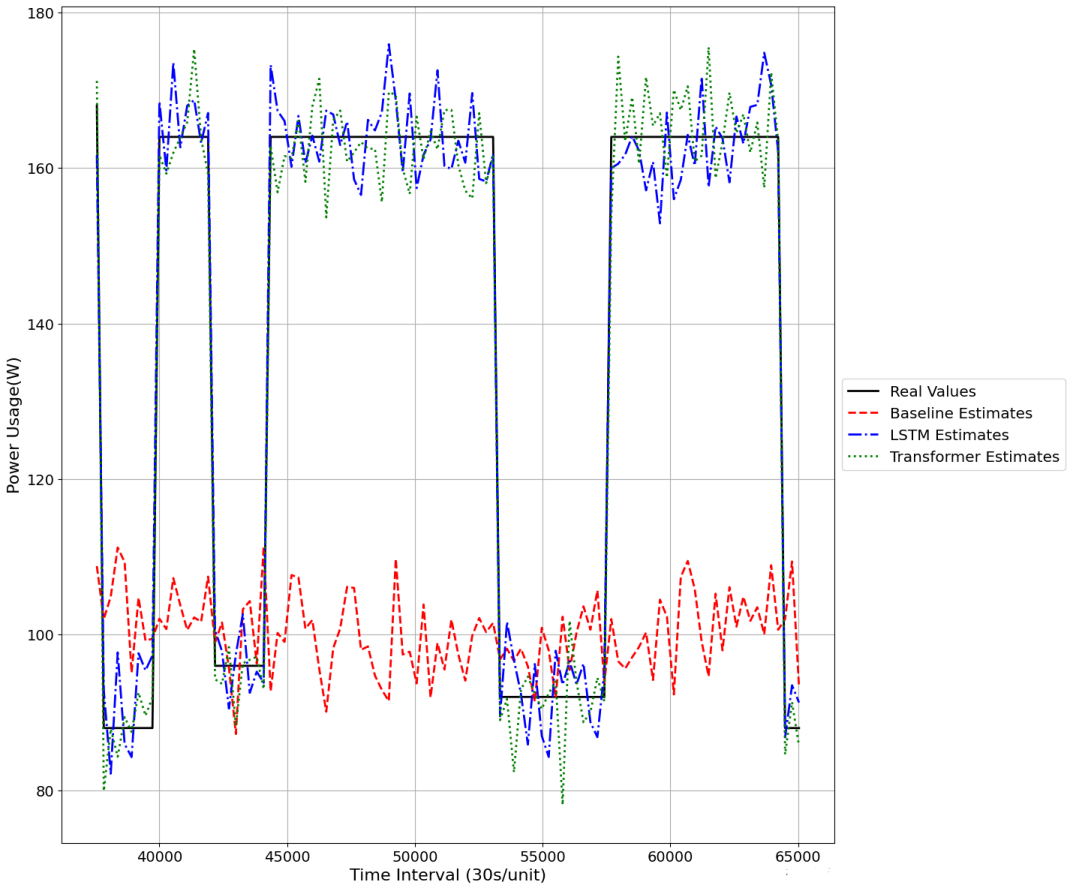


Figure 6.1: Snippet comparison of power usage estimations: Baseline, LSTM, and Transformer models.

Referring to Table 6.3 and Figure 6.1, we observe that both the LSTM and Transformer models significantly outperform the baseline statistical model. The latter demonstrates a substantial deficiency in adapting to the nonlinear fluctuations in power usage, reflecting a drastic performance deficit of 82% and 85% as compared to LSTM and Transformer models, respectively.

The LSTM and Transformer models, on contrast, effectively encapsulate the nonlinear information, exhibiting adeptness in responding to the intricacies of power usage variations. Particularly, the Transformer model, with its architectural advantage of attending to different portions of the input sequence differently, lends it proficiency in managing complex dependencies, thus slightly edging out even the LSTM model.

6.4.2 Prediction Results

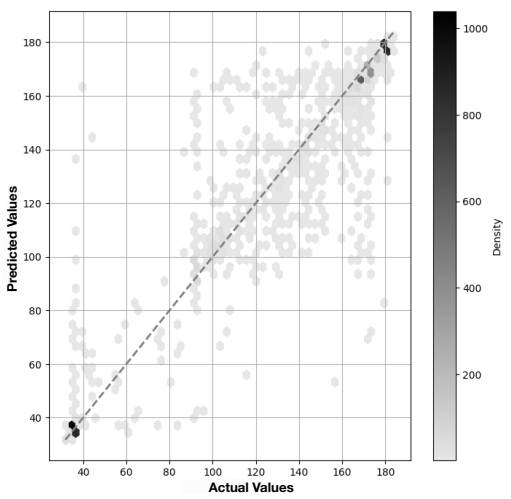
Model	Power Usage		Temperature		CPU Utilization	
	RMSE	RE	RMSE	RE	RMSE	RE
LSTM	9.55	5.3%	1.77	2.4%	3.80	-
Transformer Model	8.92	5.1%	1.54	2.1%	3.46	-

Table 6.4: Prediction performance comparison between Transformer and LSTM models across different targets. RE values are not provided for CPU utilization due to the presence of numerous zero values, which can lead to ambiguities in relative error calculations.

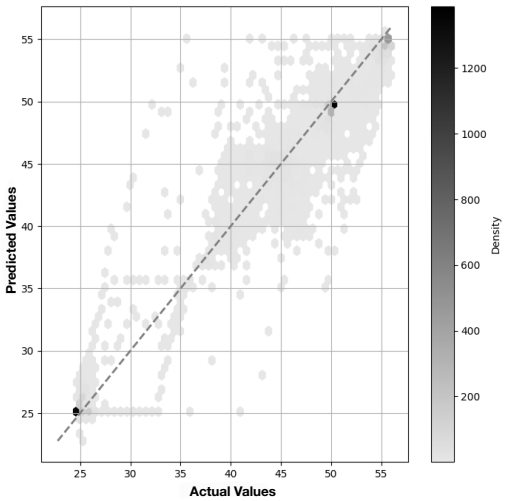
As shown above 6.4, the Transformer Model consistently slightly outperforms the LSTM in all metrics (Power Usage, Temperature, and CPU Utilization) in terms of both RMSE and RE.

Besides, the difference in performance is more significant in the "Power Usage" and "CPU Utilization" metrics, where the Transformer Model demonstrates a lower RMSE by 0.63 and 0.338 respectively compared to LSTM.

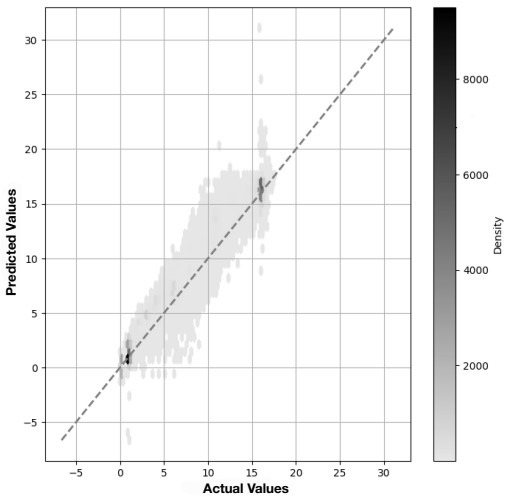
Specifically, for Power Usage, the Transformer surpasses the LSTM with an RMSE improvement of 0.63 units and a reduced RE by 0.5%. This trend of enhanced performance is also evident in Temperature predictions, where the Transformer leads with an RMSE advantage of 0.23 and an RE improvement of 0.3%. In the context of CPU Utilization, the Transformer's predictions are closer to the observed values, evident from its RMSE advantage of 0.338. A key insight derived from these results is the potential adaptability of the Transformer's architecture for time-series prediction tasks. Its inherent ability to process various parts of the input data in parallel, capturing both immediate and long-range dependencies, enables the Transformer to identify intricate temporal relationships more effectively than the sequentially processing LSTM.



(a) Power usage

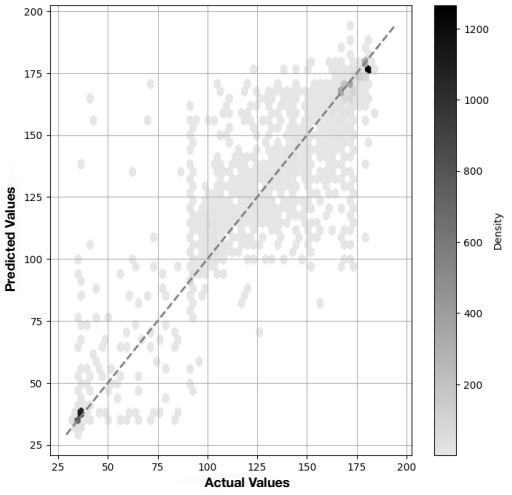


(b) Temperature

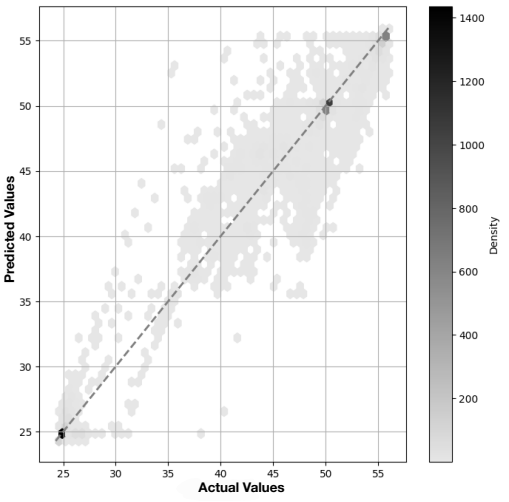


(c) CPU utilization

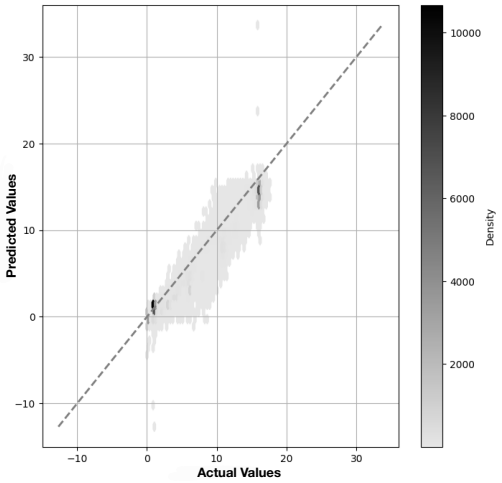
Figure 6.2: LSTM Model performance.



(a) Power usage



(b) Temperature



(c) CPU utilization

Figure 6.3: Transformer Model performance.

The figures above function visually compare actual and predicted values using hexagonal bins. The darkness of each hexagon indicates the density of data points, with darker hexagons signifying a higher concentration of points. A reference diagonal line represents where predictions match actual values perfectly. Thus, hexagons close to this line indicate accurate predictions. When viewing the plot, assess the color concentration and proximity to the diagonal: a concentration of dark hexagons near this line suggests precise predictions, while those distant may highlight prediction discrepancies or outliers.

From the results visualized in the figure, several key points can be distilled, as enumerated below:

- For both models, a significant concentration of data points lies along the diagonal, indicating that the predicted values are frequently aligned with the actual values.
- While both models generally align with the diagonal, noticeable deviations highlight their challenges in accurately predicting sudden changes in energy consumption and resource al-

location, potentially driven by complex data patterns or unpredictable external factors.

- The Transformer model exhibits a slightly higher density of points clustered closely around the diagonal compared to the other model, suggesting that its predictions can be more accurate or consistent.

The observation that a majority of points for both models align with the diagonal is promising. It signifies that both models generally provide accurate predictions for the 4-step prediction. However, the presence of points that deviate from the diagonal suggests that there are certain instances where the models' predictions diverge from the actual values. These deviations might be attributed to sudden changes in energy consumption and resource allocation, which remain challenging for the models to capture accurately. Such fluctuations can be driven by unpredictable external factors or inherent complexities in the data patterns. While the models demonstrate commendable predictive capabilities overall, further refinements might be needed to better handle these unexpected shifts and improve the forecasting robustness.

The slight difference in the concentration of points around the diagonal for the Transformer model suggests its superior capability in making predictions that are very close to the actual values. The denser clustering for the Transformer model around the diagonal could indicate that it is better equipped to handle the intricacies and patterns in the dataset, especially CPU utilization, potentially due to the self-attention mechanisms inherent in Transformer architectures.

In general, the results presented in these figures underscore the utility and effectiveness of both models, with a slight edge in favor of the Transformer model.

6.4.3 Prediction Performance for Different Time Steps

The following visual 6.7 delves into the model's performance across varying time lags for three different metrics: Power Usage, Temperature, and CPU Utilization. A noticeable trend across these figures is the pronounced decline in model accuracy when forecasting around 200 lags ahead for both models. This drop in performance aligns with our previous observations from the autocorrelation analysis, underscoring the challenges of making long-term predictions based on past data. The autocorrelation results, presented earlier indicated that the strength of the relationship between observations diminishes significantly after approximately 200 lags(1.5h - 2h). This present analysis brings to light the practical implications of that finding. As the models beyond the 200-lag mark, both the LSTM and Transformer models grapple with increased uncertainty, culminating in pronounced errors.

Also, In our analysis of performance across varying time steps, both models exhibit commendable accuracy for short-term predictions within approximately 100 lags, equating to a timeframe of about 50 to 60 minutes. When venturing into mid-term predictions, spanning between 100 to 200 lags or roughly 1 to 1.5 hours, the Transformer maintains its stability and robustness, whereas the LSTM shows a discernible decline in performance. However, for forecasting beyond the 200-lag mark, which translates to predictions exceeding 1.5 hours, there is a noticeable degradation in the accuracy of both models. This discrepancy indicates that while the LSTM can capture short-term dependencies effectively, the Transformer's self-attention mechanism possibly enables it to harness longer dependencies more robustly.

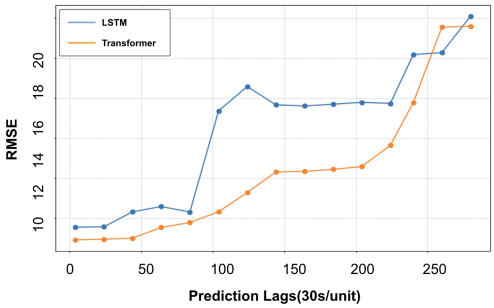


Figure 6.4: Power Usage

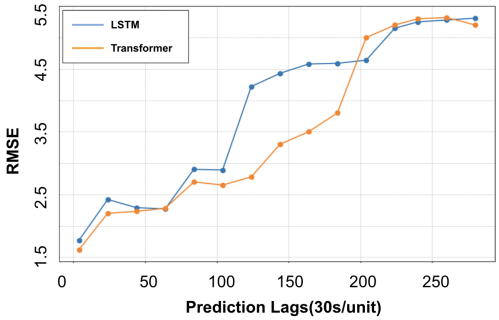


Figure 6.5: Temperature

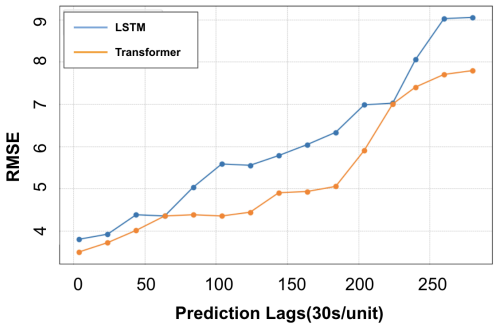


Figure 6.6: CPU Utilization

Figure 6.7: Model performance in predicting different time lags

6.5 Discussion and Model Comparison

Table 6.5: Model comparison between LSTM and Transformer

Aspect	**LSTM**	**Transformer**
Performance	Acceptable	Slight Better
Consistency	High	Moderate
Train Time (Epoch)	around 50s	around 30s
Inference Time	around 5.6ms/pred	around 13ms/pred
Model Size	Large(171k+)	Small(17k+)
Param Efficiency	Lower	Higher
Retrain Stability	Stable	Varies

Table 6.5 elucidates the strengths and weaknesses of both the LSTM and Transformer models in the context of our application. Several insights emerge from this direct comparison.

The Transformer, while exhibiting slightly superior predictive performance, is plagued by inconsistencies during retraining sessions. Beyond mere performance metrics, the Transformer exhibits inconsistency during retraining. This unpredictability, especially when retaining the same hyperparameters, positions it as a less dependable choice in settings that necessitate regular model recalibrations. The LSTM, on the other hand, stands out for its remarkable stability and predictability across training sessions. In such environments, the LSTM's consistency becomes invaluable. It remains steadfast and predictable across training sessions, highlighting its robustness and reli-

ability for iterative training with evolving data.

When it comes to training speed, the Transformer is faster than the LSTM. However, this advantage gets reduced during inference, where the LSTM works faster. This difference highlights the need to know where and how we plan to use these models, especially when deciding which part (training or inference) should be more efficient.

There's a clear difference between the two models when we look at their design and the number of parameters they use. The LSTM has many more parameters, which might make it seem more complicated compared to the simpler design of the Transformer. What's surprising is that the Transformer, even with fewer parameters, performs really well. This shows that the Transformer's design is efficient and can do a lot with less.

Based on another dataset spanning a different time period, both models exhibit comparable performance. When comparing outcomes to the current dataset, the LSTM model shows variances of approximately 5%, 3%, and 7% for CPU utilization, Temperature, and Power Usage, respectively. Meanwhile, the Transformer model exhibits differences of 6%, 2%, and 5% for the same metrics.

Our examination of performance across various time steps suggests that the Transformer's inherent self-attention mechanism might be adept at capturing and leveraging longer dependencies within the data. This capacity makes the Transformer particularly suited for scenarios requiring predictions across extended steps. Consequently, when the task necessitates longer-term forecasting or when the data's dependencies span across extended intervals, the Transformer emerges as a more favorable choice compared to models without such attention-based capabilities.

At the same time, in our context of forecasting energy and resource consumption, the capacity to continually retrain with fresh data — without compromising the robustness of a previously high-performing model — is paramount. Given that the reductions in parameters didn't significantly enhance the training speed, the inclination would still be toward the reliability of the LSTM. Consequently, given its consistent performance and efficiency, the LSTM's reliability stands out as particularly suited to our specific requirements.

6.6 Summary

In this chapter, we meticulously crafted experiments to assess the efficacy of modeling techniques, particularly focusing on LSTM and Transformer models. When applied to estimation and prediction tasks, both models demonstrated significant prowess. For estimation, there was an impressive over 80% uptick in accuracy relative to contemporary statistical methods. In a 4-step ahead prediction, the models consistently produced results with around 5% relative error for power usage, 2% for temperature, and an RMSE ranging from 3.46 to 3.8 for CPU utilization.

Delving deeper, we undertook a thorough assessment, pitting the LSTM against the Transformer, specifically in the domain of HPC data centers. This comparative analysis yielded valuable insights, spotlighting the aptness of each model for distinct applications, and consequently, charting the path toward enhanced sustainability and operational proficiency in high-performance computing scenarios. Notably, the Transformer, fortified by its self-attention mechanism, outperforms in tasks demanding the discernment of long-term patterns, making it especially suitable for extended forecasting endeavors. On the other hand, for applications such as energy and resource consumption forecasting, where continual model updates are paramount, the LSTM stands out for its unwavering stability and reliability. While trimming parameters doesn't necessarily accelerate the Transformer's training, the LSTM's track record of consistency in dynamic datasets often earmarks it as the preferred pick for such nuanced tasks.

Chapter 7

Conclusion and Future Work

In this chapter, we conclude the main contributions of this thesis, propose several potential directions for future research, and clarify the threats to validity.

7.1 Conclusion

In this paper, we address the main problem of characterizing and modeling resource use and energy consumption in large-scale data center infrastructures, aiming to enhance societal sustainability. Our methodology hinges on three research questions, focusing on the characterization and modeling of resource and energy consumption, leveraging machine learning for accurate estimation and prediction. Addressing these concerns has multifaceted societal implications: it aligns with fostering operational savings that could benefit end consumers and empowers organizations to anticipate resource needs, ensuring uninterrupted and efficient digital services. Through our findings, we seek to contribute to a sustainable, economical, and dependable digital future. To be more specific, we address three main research questions on characterizing and modeling Resource Usage and Energy consumption using machine learning including estimation and prediction as shown below. Original contributions are also indicated.

- **RQ1: How to Characterize and Analyse Resource and Energy?**
  In Chapter 4, we present an exhaustive characterization of resource and energy consumption trends, transitioning from a macroscopic view right down to granular, node-specific observations. Our investigation covers a vast spectrum, yet simultaneously catches the frequently overlooked details at the rack level. We highlight peak insights, examine matrix correlations, analyze daily trends, and ultimately uncover three dominant recurring patterns.

- **RQ2: How to Design An Accurate and Efficient ML Model for Data Center Resource and Energy Estimation and Prediction?**

  In Chapter 5, we design two contemporary machine learning architectures: LSTM and Transformer. These models are specifically tailored for the estimation and prediction of CPU utilization and power consumption in our dataset. Our approach involves a comprehensive process of data preprocessing, model tuning, and validation to ensure that the ML models are not only accurate but also efficient for real-world applications in data centers.

- **RQ3: How to Evaluate the ML Models to Understand Resource Use and Energy Consumption?**

  In Chapter 6, we design experiments to evaluate the performance of the modeling. For estimation and prediction modeling, both LSTM and Transformer models show sufficient

predictable power. For estimation, they achieve over 80% enhancement in estimation accuracy compared to current statistical methods. For 4 steps ahead prediction, both models achieve approximately 5% relative error for power usage, 2% for temperature, and an RMSE value range between 3.46 and 3.8 for CPU utilization.

We further offer a comprehensive evaluation and comparison between LSTM and Transformer within the context of HPC data centers. This provides insights into the best-suited applications of each model, laying a foundation for improved sustainability and operational efficiency in high-performance computing environments. The Transformer, with its self-attention mechanism, excels in capturing long-term dependencies, making it ideal for tasks demanding extended forecasting. Conversely, for applications like energy and resource consumption forecasting where constant model updates are vital, the LSTM's stability and consistency shine through. While parameter reductions don't significantly boost the Transformer's training speed, the LSTM's proven reliability in evolving datasets often positions it as the go-to choice for such specific requirements.

- **Original Contribution:**

  1. **Comprehensive characterization on multi-level, fine-grained dataset:**

     We delve into an in-depth study on the dataset which is unique due to its detailed granularity and multi-layered hierarchy. Such intricate data holds the potential to unlock sophisticated patterns and insights that are typically overlooked in less refined datasets. This depth and breadth lay the foundation for more robust modeling and insightful characterizations.

  2. **Analysis spanning multiple levels (node, rack, overview):** We carry out characterization and analysis across multiple levels, offering insights from each distinct level. At the node level, we investigate individual computational components. At the rack level, we grasp the synergy between nodes. The overview provides a macroscopic perspective, ensuring holistic insights.

  3. **Employment of frequency domain analysis for enhanced pattern detection:**

     We integrate frequency domain analysis to identify recurring patterns in HPC data centers, detecting underlying periodicities fundamental to understanding HPC behavior.

  4. **ML model comparison and insights into HPC data center:** By leveraging machine learning models and assessing their efficacy against practical metrics in the HPC data center landscape, we illuminate their precise advantages and areas of strength. This analytical process provides pivotal insights into optimizing these models specifically for high-performance computing infrastructures, underscoring their potential to significantly enhance operations and efficiency in the HPC data center domain.

7.2 Directions for Future Research

In this section, we propose several potential directions for future research.

Conduct simulation experiments to validate the model One promising avenue for future research lies in conducting simulation experiments to validate the model. Simulated environments allow for the controlled testing of various parameters and scenarios, ensuring a comprehensive understanding of the model's strengths and limitations. By employing simulations, we can generate synthetic data under controlled conditions, offering a more rigorous testing ground for our model's robustness. This approach not only supplements real-world validation but also highlights potential areas of improvement that might not be evident under typical operating conditions. It would be valuable to determine how the model performs under varying conditions, which can help refine it further for optimal real-world performance.

Proactive resource and energy scheduling: harnessing estimation and prediction algorithms for optimized HPC operations Building upon our current research, a significant direction for future exploration would be the development of a systematic energy and resource scheduling mechanism, grounded in the estimation and prediction algorithms we've examined. Such a scheduling system would aim to optimize the allocation and use of resources in real time, harnessing the predictive power of our algorithms to anticipate future demands. This proactive approach could lead to substantial energy savings, minimize resource wastage, and ensure the highest possible operational efficiency. Further, integrating feedback loops into the system could refine prediction accuracy over time, making the scheduler increasingly adaptive to the evolving demands and patterns of the HPC clusters.

7.3 Threats to Validity

In our related work part, we recognize several validity threats. Firstly, the selection process might unintentionally favor specific viewpoints due to our set criteria, such as limiting our scope to scientific research and English language publications. This approach overlooks valuable insights from commercial sectors and non-English sources. Moreover, publication bias, favoring positive outcomes, could lead to a skewed representation, potentially exaggerating effect sizes. Lastly, given the ever-evolving nature of research, our review, though current, may soon miss the latest studies, affecting its long-term relevance.

In the characterization, while our work is already detailed and fine-grained, it's essential to recognize its inherent limitations. The scope and matrixes of data can still be expanded upon. As we dive into our findings, it's pivotal to remember that there's always room for broadening the data spectrum and refining our interpretations based on emerging datasets.

In the modeling part, different architectural variations or custom modifications may be more suitable for specific tasks than others. For example, the number of layers, the size of the attention heads, and the overall dimensionality in the Transformer model can influence the model's performance. The wide array of possible design choices, each potentially affecting the outcome, means that ensuring the most appropriate design for a given task becomes a complex and critical endeavor. The flexibility in model design, while a strength, can lead to uncertainties in the choice of configurations and parameters, which can be a threat to validity.

In addition to the architectural considerations, the scope of our experiments was constrained to specific nodes and a singular resource within the dataset. This focused approach, though streamlined, might introduce limitations. Testing only on designated nodes could reduce the generality of our results, potentially making them less applicable across different nodes or datasets. Furthermore, by concentrating solely on one resource of the dataset, we might miss out on nuances or patterns present in other resources, which could offer valuable insights or even challenge our current findings. For a more comprehensive understanding, future studies should contemplate expanding the experimentation landscape.

Bibliography

[1] Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, and Albert Zomaya. A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in computers*, 82:47–111, 2011. 1

[2] James R Bence. Analysis of short time series: correcting for autocorrelation. *Ecology*, 76(2):628–639, 1995. 48, 49

[3] Farid Benhammadi, Zahia Gessoum, Aicha Mokhtari, et al. Cpu load prediction using neuro-fuzzy and bayesian inferences. *Neurocomputing*, 74(10):1606–1616, 2011. 19, 27

[4] Simon Elias Bibri. The iot for smart sustainable cities of the future: An analytical framework for sensor-based big data applications for environmental sustainability. *Sustainable cities and society*, 38:230–253, 2018. 1

[5] Norman Bourassa, Walker Johnson, Jeff Broughton, Deirdre McShane Carter, Sadie Joy, Raphael Vitti, and Peter Seto. Operational data analytics: Optimizing the national energy research scientific computing center cooling systems. In *Proceedings of the 48th International Conference on Parallel Processing: Workshops*, pages 1–7, 2019. 11

[6] Jim Brandt, Ann Gentile, Jackson Mayo, Philippe Pebay, Diana Roe, David Thompson, and Matthew Wong. Resource monitoring and management with ovis to enable hpc in cloud computing environments. In *2009 IEEE International Symposium on Parallel & Distributed Processing*, pages 1–8. IEEE, 2009. 20

[7] Bruce Bugbee, Caleb Phillips, Hilary Egan, Ryan Elmore, Kenny Gruchalla, and Avi Purkayastha. Prediction and characterization of application power use in a high-performance computing environment. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 10(3):155–165, 2017. 17

[8] Van Bui, Boyana Norris, Kevin Huck, Lois Curfman McInnes, Li Li, Oscar Hernandez, and Barbara Chapman. A component infrastructure for performance and power modeling of parallel scientific applications. In *Proceedings of the 2008 compFrame/HPC-GECO workshop on Component based high performance*, pages 1–11, 2008. 21, 28, 32

[9] Rajkumar Buyya, Anton Beloglazov, and Jemal Abawajy. Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. *arXiv preprint arXiv:1006.0308*, 2010. 1

[10] Jian Cao, Jiwen Fu, Minglu Li, and Jinjun Chen. Cpu load prediction for cloud environment based on a dynamic ensemble model. *Software: Practice and Experience*, 44(7):793–804, 2014. 19, 26

[11] Zheyi Chen, Jia Hu, Geyong Min, Chunbo Luo, and Tarek El-Ghazawi. Adaptive and efficient resource allocation in cloud datacenters using actor-critic deep reinforcement learning. *IEEE Transactions on Parallel and Distributed Systems*, 33(8):1911–1923, 2021. 1

[12] Zhijia Chen, Yuanchang Zhu, Yanqiang Di, and Shaochong Feng. Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network. *Computational intelligence and neuroscience*, 2015:17–17, 2015. 19, 26

[13] T Cheng, B Rivard, GA Sánchez-Azofeifa, J Feng, and M Calvo-Polanco. Continuous wavelet analysis for the detection of green attack damage due to mountain pine beetle infestation. *Remote sensing of Environment*, 114(4):899–910, 2010. 48, 49

[14] Ce Chi, Kaixuan Ji, Penglei Song, Avinab Marahatta, Shikui Zhang, Fa Zhang, Dehui Qiu, and Zhiyong Liu. Cooperatively improving data center energy efficiency based on multi-agent deep reinforcement learning. *Energies*, 14(8):2071, 2021. 2

[15] Marta Chinnici, Davide De Chiara, and Andrea Quintiliani. An hpc-data center case study on the power consumption of workload. In *Applied Physics, System Science and Computers II: Proceedings of the 2nd International Conference on Applied Physics, System Science and Computers (APSAC2017), September 27-29, 2017, Dubrovnik, Croatia*, pages 183–192. Springer, 2019. 17

[16] Xiaoyu Chu, Sacheendra Talluri, Laurens Versluis, and Alexandru Iosup. How do ml jobs fail in datacenters? analysis of a long-term dataset from an hpc cluster. In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering*, pages 263–268, 2023. 8, 10

[17] Jacob Cohen, Patricia Cohen, Stephen G West, and Leona S Aiken. *Applied multiple regression/correlation analysis for the behavioral sciences*. Routledge, 2013. 44

[18] Neil J Cornish and Shane L Larson. Lisa data analysis: Source identification and subtraction. *Physical Review D*, 67(10):103001, 2003. 35

[19] Fang Dong, Junzhou Luo, Aibo Song, Jiuxin Cao, and Jun Shen. An effective data aggregation based adaptive long term cpu load prediction mechanism on computational grid. *Future Generation Computer Systems*, 28(7):1030–1044, 2012. 19, 27

[20] Wei Fang, ZhiHui Lu, Jie Wu, and ZhenYin Cao. Rpps: A novel resource prediction and provisioning scheme in cloud data center. In *2012 IEEE Ninth International Conference on Services Computing*, pages 609–616. IEEE, 2012. 19, 27

[21] Ryan E Grant, Kevin T Pedretti, and Ann Gentile. Overtime: A tool for analyzing performance variation due to network interference. In *Proceedings of the 3rd Workshop on Exascale MPI*, pages 1–10, 2015. 12

[22] Philipp Gschwandtner, Michael Knobloch, Bernd Mohr, Dirk Pleiter, and Thomas Fahringer. Modeling cpu energy consumption of hpc applications on the ibm power7. In *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 536–543. IEEE, 2014. 22, 31

[23] Qiang Guan and Song Fu. Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures. In *2013 IEEE 32nd International Symposium on Reliable Distributed Systems*, pages 205–214. IEEE, 2013. 12

[24] Qiang Guan, Ziming Zhang, and Song Fu. Proactive failure management by integrated unsupervised and semi-supervised learning for dependable cloud systems. In *2011 Sixth International Conference on Availability, Reliability and Security*, pages 83–90, 2011. 20, 28

[25] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34:15908–15919, 2021. 59, 62

[26] Connor Imes, Steven Hofmeyr, and Henry Hoffmann. Energy-efficient application resource scheduling using machine learning classifiers. In *Proceedings of the 47th International Conference on Parallel Processing*, pages 1–11, 2018. 12

[27] InsideHPC. What is HPC? - insideHPC. `https://insidehpc.com/hpc-basic-training/what-is-hpc/`, 2023. Accessed: 2023-06-17. 7

[28] John PA Ioannidis. The proposal to lower p value thresholds to. 005. *Jama*, 319(14):1429–1430, 2018. 44

[29] Waheed Iqbal, Josep Lluis Berral, Abdelkarim Erradi, David Carrera, et al. Adaptive prediction models for data center resources utilization estimation. *IEEE Transactions on Network and Service Management*, 16(4):1681–1693, 2019. 1, 14, 19, 26

[30] Mateusz Jarus, Ariel Oleksiak, Tomasz Piontek, and J Weglarz. Runtime power usage estimation of hpc servers for various classes of real-life applications. *Future Generation Computer Systems*, 36:299–310, 2014. 24, 29, 31

[31] Weixiang Jiang, Ziyang Jia, Sirui Feng, Fangming Liu, and Hai Jin. Fine-grained warm water cooling for improving datacenter economy. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 474–486, 2019. 12

[32] Nicola Jones et al. How to stop data centres from gobbling up the world's electricity. *Nature*, 561(7722):163–166, 2018. 1

[33] Love Obiani Arugu JP and Chilaka Francis Chigozie. Information and communication technology (ict) application in social and political system. *European Journal of Research in Social Sciences Vol*, 4(1):51–63, 2016. 1

[34] Henning Kagermann. Change through digitization—value creation in the age of industry 4.0. In *Management of permanent change*, pages 23–45. Springer, 2014. 1

[35] Gokcen Kestor, Roberto Gioiosa, Darren J Kerbyson, and Adolfy Hoisie. Enabling accurate power profiling of hpc applications on exascale systems. In *Proceedings of the 3rd International Workshop on Runtime and Operating Systems for Supercomputers*, pages 1–8, 2013. 22, 31

[36] Wania Khan, Davide De Chiara, Ah-Lian Kor, and Marta Chinnici. Exploratory data analysis for data center energy management. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, pages 571–580, 2022. 22, 28, 29

[37] Nitin Khosla and Dharmendra Sharma. Forecast extreme cpu usages under peak load using envelop em semi-supervised learning. In Luigi Troiano, Alfredo Vaccaro, Roberto Tagliaferri, Nishtha Kesswani, Irene Díaz Rodriguez, Imene Brigui, and Domenico Parente, editors, *Advances in Deep Learning, Artificial Intelligence and Robotics*, pages 25–34, Cham, 2022. Springer International Publishing. 13, 20, 28

[38] Volodymyr Kindratenko, Dawei Mu, Yan Zhan, John Maloney, Sayed Hadi Hashemi, Benjamin Rabe, Ke Xu, Roy Campbell, Jian Peng, and William Gropp. Hal: Computer system for scalable deep learning. In *Practice and experience in advanced research computing*, pages 41–48. 2020. 9

[39] Alagumalai Krishnapandi, Balamurugan Muthukutty, Shen-Ming Chen, Kumaravelu Thanigai Arul, Huang Ji Shiuan, and Muthusamy Selvaganapathy. Bismuth molybdate incorporated functionalized carbon nanofiber as an electrocatalytic tool for the pinpoint detection of organic pollutant in life samples. *Ecotoxicology and Environmental Safety*, 209:111828, 2021. 2

[40] Jitendra Kumar and Ashutosh Kumar Singh. Cloud datacenter workload estimation using error preventive time series forecasting models. *Cluster Computing*, 23(2):1363–1379, 2020. 1, 14

[41] Julian Kunkel and Manuel F Dolz. Understanding hardware and software metrics with respect to power consumption. *Sustainable Computing: Informatics and Systems*, 17:43–54, 2018. 24, 29, 30

[42] Alexey Lastovetsky and Ravi Reddy Manumachu. New model-based methods and algorithms for performance and energy optimization of data parallel applications on homogeneous multicore clusters. *IEEE Transactions on Parallel and Distributed Systems*, 28(4):1119–1133, 2016. 25, 29, 30

[43] Nuoa Lei. A robust modeling framework for energy analysis of data centers. In *Proceedings of the 7th International Conference on ICT for Sustainability*, pages 177–180, 2020. 24, 29

[44] Xiuqiao Li, Nan Qi, Yuanyuan He, and Bill McMillan. Practical resource usage prediction method for large memory jobs in hpc clusters. In *Supercomputing Frontiers: 5th Asian Conference, SCFA 2019, Singapore, March 11–14, 2019, Proceedings 5*, pages 1–18. Springer, 2019. 19, 26

[45] Xue Lin, Yanzhi Wang, and Massoud Pedram. A reinforcement learning-based power management framework for green computing data centers. In *2016 IEEE International Conference on Cloud Engineering (IC2E)*, pages 135–138. IEEE, 2016. 12, 23, 29, 30

[46] Satoshi Matsuoka, Jens Domke, Mohamed Wahib, Aleksandr Drozd, and Torsten Hoefler. Myths and legends in high-performance computing. *arXiv preprint arXiv:2301.02432*, 2023. 7

[47] mckinsey. Investing in the rising data center economy, 2023. 1

[48] Dejan Milojicic. Accelerators for artificial intelligence and high-performance computing. *Computer*, 53(02):14–22, 2020. 7

[49] Ryan Mitchell, Loc Pottier, Steve Jacobs, Rafael Ferreira da Silva, Mats Rynge, Karan Vahi, and Ewa Deelman. Exploration of workflow management systems emerging features from users perspectives. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4537–4544. IEEE, 2019. 9

[50] Matthias Möller and Cornelis Vuik. On the impact of quantum computing technology on future developments in high-performance scientific computing. *Ethics and information technology*, 19:253–269, 2017. 10

[51] Henry M Monti, Ali R Butt, and Sudharshan S Vazhkudai. Catch: A cloud-based adaptive data transfer service for hpc. In *2011 IEEE International Parallel & Distributed Processing Symposium*, pages 1242–1253. IEEE, 2011. 18

[52] Alessio Netti, Woong Shin, Michael Ott, Torsten Wilde, and Natalie Bates. A conceptual framework for hpc operational data analytics. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 596–603. IEEE, 2021. 12

[53] Md Nahid Newaz and Md Atiqul Mollah. Memory usage prediction of hpc workloads using feature engineering and machine learning. In *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, HPC Asia '23, page 64–74, New York, NY, USA, 2023. Association for Computing Machinery. 18, 26

[54] Gence Ozer, Sarthak Garg, Neda Davoudi, Gabrielle Poerwawinata, Matthias Maiterth, Alessio Netti, and Daniele Tafani. Towards a predictive energy model for hpc runtime systems using supervised learning. In *Euro-Par 2019: Parallel Processing Workshops: Euro-Par 2019 International Workshops, Göttingen, Germany, August 26–30, 2019, Revised Selected Papers 25*, pages 626–638. Springer, 2020. 22, 28, 29

[55] Gence Ozer, Alessio Netti, Daniele Tafani, and Martin Schulz. Characterizing hpc performance variation with monitoring and unsupervised learning. In *High Performance Computing: ISC High Performance 2020 International Workshops, Frankfurt, Germany, June 21–25, 2020, Revised Selected Papers 35*, pages 280–292. Springer, 2020. 17

[56] Kenneth O'brien, Ilia Pietri, Ravi Reddy, Alexey Lastovetsky, and Rizos Sakellariou. A survey of power and energy predictive models in hpc systems and applications. *ACM Computing Surveys (CSUR)*, 50(3):1–38, 2017. 11

[57] Amer Qouneh, Nilanjan Goswami, Ruijin Zhou, and Tao Li. On characterization of performance and energy efficiency in heterogeneous hpc cloud data centers. In *2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, pages 315–320. IEEE, 2014. 17

[58] Ali Asghar Rahmanian, Mostafa Ghobaei-Arani, and Sajjad Tofighy. A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment. *Future Generation Computer Systems*, 79:54–71, 2018. 19, 26

[59] Joseph Lee Rodgers and W Alan Nicewander. Thirteen ways to look at the correlation coefficient. *American statistician*, pages 59–66, 1988. 43

[60] Osman Sarood, Akhil Langer, Abhishek Gupta, and Laxmikant Kale. Maximizing throughput of overprovisioned hpc data centers under a strict power budget. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 807–818. IEEE, 2014. 35

[61] Arman Shehabi, Sarah J Smith, Eric Masanet, and Jonathan Koomey. Data center growth in the united states: decoupling the demand for services from electricity use. *Environmental Research Letters*, 13(12):124030, 2018. 1

[62] Balaji Subramaniam and Wu-chun Feng. Statistical power and performance modeling for optimizing the energy efficiency of scientific computing. In *2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, pages 139–146. IEEE, 2010. 21, 28, 32

[63] Mohammed Tanash, Daniel Andresen, and William Hsu. Ampro-hpcc: A machine-learning tool for predicting resources on slurm hpc clusters. In *ADVCOMP... the... International Conference on Advanced Engineering Computing and Applications in Sciences*, volume 2021, page 20. NIH Public Access, 2021. 18, 26

[64] Ananta Tiwari, Michael A Laurenzano, Laura Carrington, and Allan Snavely. Modeling power and energy usage of hpc kernels. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, pages 990–998. IEEE, 2012. 23, 29

[65] Ananta Tiwari, Michael A. Laurenzano, Laura Carrington, and Allan Snavely. Modeling power and energy usage of hpc kernels. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum*, pages 990–998, 2012. 31

[66] Charles Van Loan. *Computational frameworks for the fast Fourier transform*. SIAM, 1992. 48, 49

[67] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 59

[68] Akshat Verma, Puneet Ahuja, and Anindya Neogi. Power-aware dynamic placement of hpc applications. In *Proceedings of the 22nd annual international conference on Supercomputing*, pages 175–184, 2008. 12

[69] Laurens Versluis, Mehmet Cetin, Caspar Greeven, Kristian Laursen, Damian Podareanu, Valeriu Codreanu, Alexandru Uta, and Alexandru Iosup. Less is not more: We need rich datasets to explore. *Future Generation Computer Systems*, 142:117–130, 2023. 35

[70] Bin Wang, Zhengwei Qi, Ruhui Ma, Haibing Guan, and Athanasios V Vasilakos. A survey on data center networking for cloud computing. *Computer Networks*, 91:528–547, 2015. 1

[71] Zhe Wang, Zhongyuan Tian, Jiang Xu, Rafael KV Maeda, Haoran Li, Peng Yang, Zhehui Wang, Luan HK Duong, Zhifei Wang, and Xuanqi Chen. Modular reinforcement learning for self-adaptive energy efficiency optimization in multicore system. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 684–689. IEEE, 2017. 23, 30

[72] Wikipedia. Fugaku (supercomputer) - wikipedia. `https://en.wikipedia.org/wiki/Fugaku_(supercomputer)`, 2023. Accessed: 2023-06-18. 9

[73] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019. 57

[74] Qi Zhang, Mohamed Faten Zhani, Shuo Zhang, Quanyan Zhu, Raouf Boutaba, and Joseph L Hellerstein. Dynamic energy-aware capacity provisioning for cloud computing environments. In *Proceedings of the 9th international conference on Autonomic computing*, pages 145–154, 2012. 23, 28, 31