# Minimizing Energy Consumption in Data Centers Using Embedded Sensors and Machine Learning

Nalveer Moocheet

A Thesis
in
Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Master's of Applied Software Engineering at
Concordia University
Montréal, Québec, Canada

September 2023

© Nalveer Moocheet, 2023

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Nalveer Moocheet**

Entitled: **Minimizing Energy Consumption in Data Centers Using Embedded Sensors and Machine Learning**

and submitted in partial fulfillment of the requirements for the degree of

**Master's of Applied Software Engineering**

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________ Chair
*Dr. Mirco Ravanelli*

______________________ Examiner
*Dr. Mirco Ravanelli*

______________________ Examiner
*Dr. Yann Gaël Guéheneuc*

______________________ Thesis Supervisor
*Dr. Brigitte Jaumard*

______________________ Thesis Supervisor
*Dr. Tristan Glatard*

Approved by ______________________
Dr. Leila Kosseim, Graduate Program Director

September 29, 2023 ______________________
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

# Abstract

Minimizing Energy Consumption in Data Centers Using Embedded Sensors and Machine Learning

Nalveer Moocheet

Cloud Data Centers (DCs) consume extensive amounts of energy, making a significant contribution to environmental concerns. Moreover, with the emergence of 5G and future B5G networks, which are increasingly inclined towards software orientation and reliant on cloud computing, there is an urgent requirement for optimizing the energy consumption of DCs. We address this issue by proposing an energy-aware Virtual Machine (VM) placement solution for energy minimization.

In the first part of this study, we propose a highly accurate model for predicting the dynamic power consumption of cloud computing devices. Our proposal takes advantage of the various sensors that are now embedded in physical machines, or more generally in cloud server machines, as well as Performance Monitoring Counters (PMCs) to implement a highly accurate Machine Learning (ML) power prediction model. The core part of this study then integrates the novel feature space of real-time sensors' measurements and the predictive power model to propose a scalable placement algorithm, enabling proactive and energy-aware Virtual Machine placements. In addition, it utilizes a new set of temperature-related features that enables proactive hotspot avoidance.

Our ML predictive models, as well as our proposed placement algorithm, were extensively evaluated on a cluster of real physical machines and demonstrated a significantly higher performance as compared to the implemented reference models and algorithms, reducing energy consumption by up to 7%, CPU temperature by 2%, and overloading by

28%.

# Acknowledgments

I would like to express my deepest gratitude to my supervisor Dr. Brigitte Jaumard for her dedicated support and guidance. I am also grateful for the support of my co-supervisor Dr. Tristan Glatard and the team at GAIA Ericsson. I would like to acknowledge Pierre Thibault for his valuable technical support. Lastly, I sincerely appreciate the support of my family during my Graduate Studies.

This research project was supported by a MITACS - Ericsson Internship and Concordia University.

# Contents

**1 Introduction** **1**

1.1 Motivation . . . . . 1

1.2 Background . . . . . 2

1.2.1 Virtual Machine . . . . . 3

1.2.2 Virtual Machine Management & Operation . . . . . 3

1.3 Project Definition: Virtual Machine Placement for energy minimization . . 5

1.4 Key References . . . . . 5

1.5 Our Contributions . . . . . 7

1.6 Plan of the Thesis . . . . . 8

**2 Experiment Environment** **9**

2.1 Cluster Architecture . . . . . 9

2.2 Experiments . . . . . 10

2.2.1 Load Emulation and Data Collection . . . . . 10

2.2.2 Static VM Deployment . . . . . 12

**3 A Sensor Predictive Model for Power Consumption using Machine Learning** **13**

3.1 Abstract . . . . . 13

3.2 Introduction . . . . . 14

3.3 Literature Review . . . . . 17

3.3.1 Power Consumption Models with CPU only . . . . . . . . . . . . . 17
3.3.2 More General Power Consumption Models . . . . . . . . . . . . . 18
3.3.3 Temperature: A Key Parameter for Power Prediction . . . . . . . . 19
3.3.4 Concluding Remarks . . . . . . . . . . . . . . . . . . . . . . . . 19
3.4 Real-time Sensor Measurements vs. Analytical Formulas . . . . . . . . . . 20
3.4.1 Workload Characterization . . . . . . . . . . . . . . . . . . . . . 20
3.4.2 Heterogeneous PMs & Temperature . . . . . . . . . . . . . . . . . 22
3.5 Machine Learning Model . . . . . . . . . . . . . . . . . . . . . . . . . 23
3.5.1 Performance Monitoring Counters . . . . . . . . . . . . . . . . . 24
3.5.2 Embedded Sensors . . . . . . . . . . . . . . . . . . . . . . . . . 25
3.5.3 Data pre-processing . . . . . . . . . . . . . . . . . . . . . . . . 26
3.6 Machine Learning Algorithms . . . . . . . . . . . . . . . . . . . . . . . 29
3.6.1 Multiple Linear Regression Model (MLR) . . . . . . . . . . . . . 29
3.6.2 Extreme Gradient Boosting Algorithm (XGBoost) . . . . . . . . . 29
3.6.3 Multi-layer Perceptron (MLP) Regressor Model . . . . . . . . . . . 29
3.6.4 Long Short-Term Memory (LSTM) Neural Network . . . . . . . . 30
3.7 Numerical Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
3.7.1 Experimental Setup, Load Emulation & Data Collection . . . . . . 31
3.7.2 Performance Evaluation . . . . . . . . . . . . . . . . . . . . . . 33
3.7.3 Observations . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
3.8 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38

**4 Virtual Machine Placement using Embedded Sensors and Machine Learning for Energy Minimization 39**

4.1 Abstract . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
4.2 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
4.3 Literature Review . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44

4.3.1 Consolidation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
4.3.2 Thermal-Aware Approaches . . . . . . . . . . . . . . . . . . . . . 45
4.3.3 Reinforcement Learning . . . . . . . . . . . . . . . . . . . . . . 46
4.3.4 Concluding Remarks . . . . . . . . . . . . . . . . . . . . . . . . 48
4.4 Need For Thermal-aware Techniques . . . . . . . . . . . . . . . . . . . 48
4.4.1 Influence of Temperature & Need for Enhanced Feature Space . . . 49
4.4.2 Critical Temperature . . . . . . . . . . . . . . . . . . . . . . . . 51
4.5 A Proactive Energy & Thermal-aware Approach . . . . . . . . . . . . . . 52
4.5.1 Power Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52
4.5.2 Criticality of a Physical Machine . . . . . . . . . . . . . . . . . . 54
4.6 Virtual Machine Placement . . . . . . . . . . . . . . . . . . . . . . . . 55
4.6.1 Problem formulation . . . . . . . . . . . . . . . . . . . . . . . . 56
4.6.2 Criticality-based Random Sampling . . . . . . . . . . . . . . . . 58
4.6.3 Greedy Heuristic Algorithm . . . . . . . . . . . . . . . . . . . . 59
4.7 Numerical Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
4.7.1 Environment . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
4.7.2 Load Emulation & Data Collection . . . . . . . . . . . . . . . . 60
4.7.3 Evaluation: Predictive ML Models . . . . . . . . . . . . . . . . . 61
4.7.4 Evaluation: VM Placement . . . . . . . . . . . . . . . . . . . . . 63
4.8 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65

**5 Conclusion & Future Works** **68**
5.1 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68
5.2 Future Works . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69

**Bibliography** **71**

**A Additional Experiment Results** **76**

A.1 Experiment 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76

A.2 Experiment 2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78

# List of Figures

1 Expected Increase in global DC Energy Consumption [23]. . . . . . . . . . 2

2 Virtualization of a physical machine [46]. . . . . . . . . . . . . . . . . . . . 4

3 Illustration of Virtual Machines deployment over cloud Data Centers. . . . 4

4 Three-Node Architecture of OpenStack [39] . . . . . . . . . . . . . . . . . . 10

5 VM Load Emulation. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

6 Emulation v/s Trace CPU Utilization pattern . . . . . . . . . . . . . . . . . . 11

7 Heat Map of a HPE ProLiant BL460C Gen8 PM . . . . . . . . . . . . . . . 16

8 CPU utilization to Power relation for different workloads. . . . . . . . . . 21

9 Inlet temperature under similar load . . . . . . . . . . . . . . . . . . . . . . 23

10 CPU temperature under similar load. . . . . . . . . . . . . . . . . . . . . . 23

11 2D Heat Map of a PM (HP ProLiant BL460c G8) with Sensors location . . 26

12 Snapshot of a real-time sensor reading & description . . . . . . . . . . . . 27

13 MLP Neural Network Architecture . . . . . . . . . . . . . . . . . . . . . . . 30

14 Performance of ML models for each Workload . . . . . . . . . . . . . . . . 35

15 Feature weights of best performing model . . . . . . . . . . . . . . . . . . 37

16 Heat Map of a HPE ProLiant BL460C Gen8 PM . . . . . . . . . . . . . . . 43

17 Resource Utilization v/s Measured Power and impact of Temperature . . . . 50

18 Snapshot of a real-time sensor reading & description . . . . . . . . . . . . 51

19 Increase in average power prediction error at high temperature . . . . . . . 54
20 Measured Power Distribution in Datasets . . . . . . . . . . . . . . . . . . . 62
21 Measured Criticality Distribution in Datasets . . . . . . . . . . . . . . . . . 62
22 Power Consumption Comparison of Algorithms. . . . . . . . . . . . . . . . . 66
23 Percentage of time each PM spends at an overloaded CPU state. ($t_{\text{PM}}^{CPU}$) . . 66
24 Percentage of time each PM spends to the peak power state. ($t_{\text{PM}}^{POWER}$) . . . 66
25 Experiment 1 - Power Consumption Comparison of Algorithms. . . . . . . 76
26 Experiment 1 - Percentage of time each PM spends at an overloaded CPU state. ($t_{\text{PM}}^{CPU}$) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
27 Experiment 1 - Percentage of time each PM spends to the peak power state. ($t_{\text{PM}}^{POWER}$) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
28 Experiment 1 - Average Temperature at Processor 1 ($t_{\text{PM}}^{CPU}$) . . . . . . . . . 77
29 Experiment 1 - Average Temperature at Processor 2 ($t_{\text{PM}}^{POWER}$) . . . . . . . 78
30 Experiment 2 - Power Consumption Comparison of Algorithms. . . . . . . 78
31 Experiment 2 - Percentage of time each PM spends at an overloaded CPU state. ($t_{\text{PM}}^{CPU}$) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
32 Experiment 2 - Percentage of time each PM spends to the peak power state. ($t_{\text{PM}}^{POWER}$) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
33 Experiment 2 - Average Temperature at Processor 1 ($t_{\text{PM}}^{CPU}$) . . . . . . . . . 79
34 Experiment 2 - Average Temperature at Processor 2 ($t_{\text{PM}}^{POWER}$) . . . . . . . 80

# List of Tables

1 Cluster & Physical Machine's Description. . . . . . . . . . . . . . . . . . . . 10

2 Performance Monitoring Counters (PMCs) . . . . . . . . . . . . . . . . . . . 25

3 Average Resource utilization by Workloads . . . . . . . . . . . . . . . . . . 32

4 Model Performance ($R^2$ score) on each workload . . . . . . . . . . . . . . . 36

5 Average Model Performance (All Workloads) . . . . . . . . . . . . . . . . . 37

6 Models' Performance. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63

7 Performance Comparison . . . . . . . . . . . . . . . . . . . . . . . . . . . 65

8 Experiment 1 Results. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76

9 Experiment 2 Results. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78

# Chapter 1

# Introduction

## 1.1 Motivation

Cloud computing is experiencing an unprecedented surge, driven by the continuous evolution of technology, particularly with the emergence of 5G/B5G which is shifting to be more software-oriented. This growth is further propelled by the rapid expansion of low-latency technologies like the Internet of Things (IoT), smart cities, and self-driving cars. However, this increased adoption of cloud services also leads to a rise in energy consumption by Cloud Data Centers (DCs). According to the International Energy Agency (IEA) Data Centers account for around 1.5% of global electricity demand and are responsible for 1% of all energy-related greenhouse gas emissions. Furthermore, as illustrated in Figure 1, these numbers are expected to increase exponentially in the coming years [23]. Thus, with this ever-increasing demand, coupled with the rising costs of energy and the growing focus on environmental sustainability, the Data Center industry has heightened the urgency to improve their operational efficiency, more specifically the optimization of the computing sub-system's resource management(e.g., Virtual Machine Scheduling, Placement, and Migration), aiming to minimize their overall energy consumption and carbon footprints.

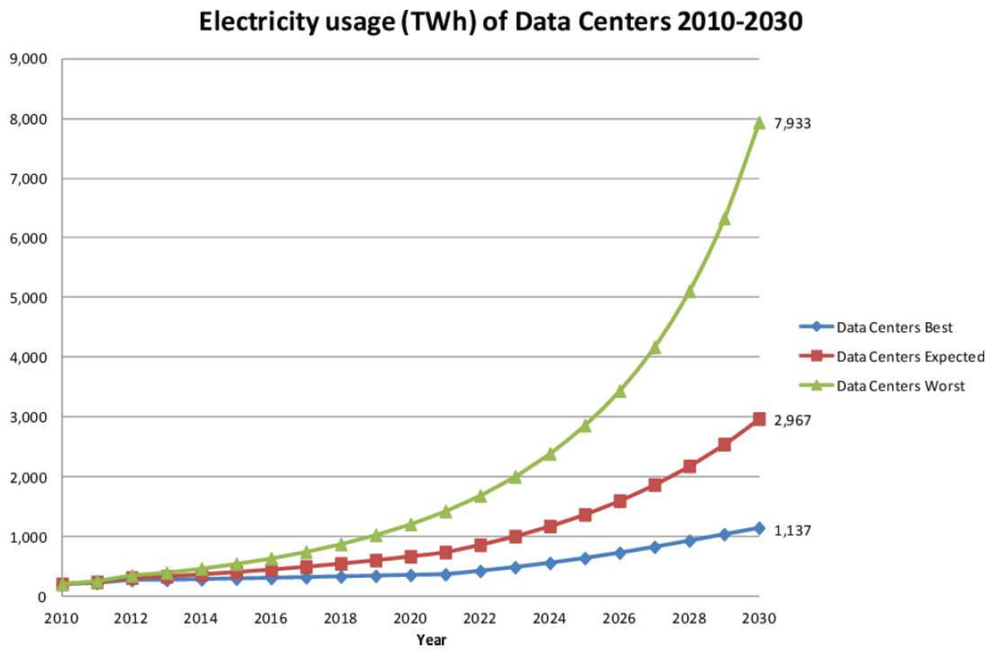


Figure 1: Expected Increase in global DC Energy Consumption [23].

## 1.2 Background

Compute servers and cooling systems are the two primary components to consider in relation to energy consumption. A typical DC houses thousands of physical machines (PMs) mounted on multiple racks or chassis, along with other essential equipment such as network switches and cables that are vital for the compute system's operation. In fact, these computing components (e.g., Server, Networking, and Storage) can account for 50% of the total data center's energy consumption [17]. Similarly, an almost equal amount of energy is consumed by the cooling system for handling the resulting massive heat generation of those hardware components. In addition, excess heat may lead to the formation of hotspots that greatly impact the system's reliability. These hotspots can trigger hardware failures, resulting from damage to silicon components [24].

Therefore, optimizing the operation and management of those PMs, remains a key area of focus for energy minimization in Cloud Data Centers. In Infrastructure as a Service

(IaaS), PMs are responsible for hosting and executing computational tasks or workloads mostly in the form of Virtual Machines (VMs) or Containers. Efficiently orchestrating the VMs among the PMs in the Data Center holds significant importance for both effective energy management and hotspot avoidance. As such, it forms the central focus of this thesis. In the following sub-sections, we offer a comprehensive description of Virtual Machines and their management, as they serve as the fundamental components of this study.

### 1.2.1 Virtual Machine

Virtual Machines (VMs) are an essential aspect of virtualization technology, which enables the partition of computing resources into separate environments. As illustrated in Fig. 2, a hypervisor that runs on a PM, allows each VM to provide an independent operating system (OS) for its applications, essentially creating multiple isolated virtual components within the PM [46]. VMs play a vital role as a core component of cloud computing and have experienced a remarkable surge in usage due to the emergence of distributed software-oriented network architectures. This evolution is driven by the rapid adoption of technologies such as Network Function Virtualization (NFV) and Virtual Network Functions (VNFs). Unlike traditional non-virtualized networks, which rely on vendor-specific software and hardware to implement network functions, VMs enable the deployment of VNFs that encompass a wide array of network functionalities, such as virtual firewalls, routers, and load balancers, promoting greater flexibility and efficiency in the cloud environment.

### 1.2.2 Virtual Machine Management & Operation

The operation and management of VMs consist of 3 main components: Scheduling, Placement, and Migration. Scheduling first accepts and decides the VMs' deployment time or order based on their precedence graph, hierarchy, or priority. After one or a set of VMs are ready for deployment, Placement determines the allocation of the VMs to specific PMs

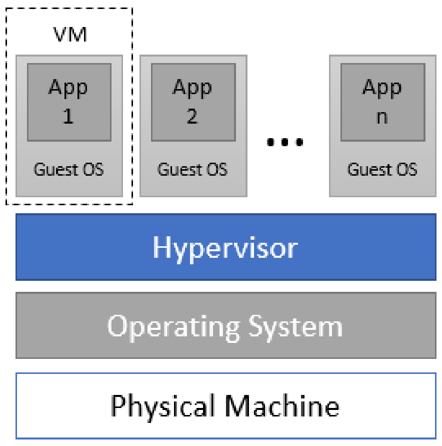


Figure 2: Virtualization of a physical machine [46].

(Can be across multiple DCs, see Fig. 3) based on certain policies or strategies for efficient resource allocation. Finally, Dynamic Migration handles the execution of certain required migrations, that is the transfer of a VM from one PM to another, either while active (hot migration) or inactive (cold migration). Dynamic migration has been a popular area of

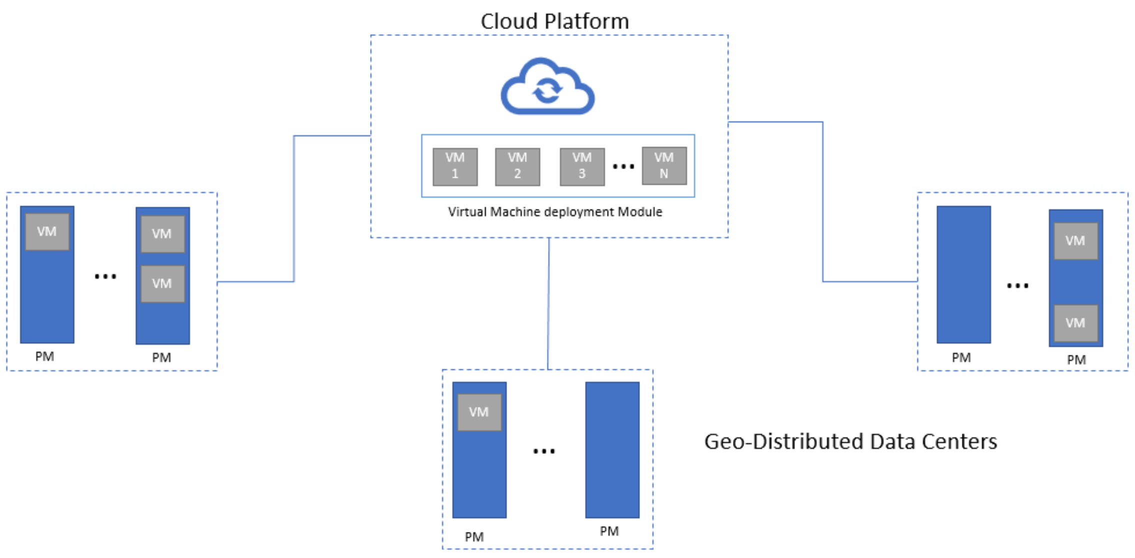

Cloud Platform
VM 1
VM 2
VM 3
...
VM N
Virtual Machine deployment Module
VM
VM
VM
PM
PM
VM
PM
PM
VM
VM
PM
PM
Geo-Distributed Data Centers

Figure 3: Illustration of Virtual Machines deployment over cloud Data Centers.

focus in relation to optimizing the operations of VMs. Through specific strategies such as consolidation, VMs undergo periodic or reactive rearrangements to handle fluctuations in workload to optimize energy and handle over-loaded or over-heated machines. However, there are various challenges associated with migration, especially the live transfer of VMs since it requires significant network bandwidth and computing resources [9]. In fact, a high number of inefficient migrations poses a serious risk of Quality of Service (QoS) and energy-efficiency degradation. Thus, several studies have instead shifted towards optimizing VM Placement.

## 1.3 Project Definition: Virtual Machine Placement for energy minimization

The goal of a VM placement algorithm is to select the most suitable PMs for VMs allocation in order to optimize multiple, very often conflicting objectives such as resource allocation, performance, energy efficiency, hotspot avoidance, and Quality of Service. Furthermore, a placement algorithm can be utilized for both, initial placement which involves the first selection of PMs for VMs' deployment, as well as, in the context of migration, for which the algorithm helps determine the new destination for a VM's transfer [54]. Hence, the goal of this research project is to develop an efficient and scalable VM placement algorithm. The primary objectives are to minimize energy consumption, ensure proper resource allocation, maintain high performance, and avoid hotspots within the cluster of server machines.

## 1.4 Key References

Data centers are typically designed to handle peak traffic to avoid SLA violations, overload, or hotspot conditions. Consequently, resources are frequently over-provisioned, leading to energy inefficiency, as idle servers may consume up to 70% of their maximum energy [6]. Hence, VM consolidation is a widely used technique, aiming to minimize active PMs and power off underutilized/idle ones. Several authors, e.g., [4], [5] and [36] consider VM placement as an energy-aware bin packing problem, employing best-fit heuristics to deploy VMs on a minimum number of physical machines (PMs). However, these approaches may cause numerous hotspots due to sudden traffic spikes, leading to frequent virtual machine migrations and significant energy and resource costs. Additionally, delays in reactivating inactive machines may lead to Service Level Agreement (SLA) violations.

Some studies, e.g., [21], [14], and [43] tackle these issues by presenting proactive approaches for VM consolidation. These methods consider both current and future resource utilization by employing prediction models to approximate the future resource needs of VMs and PMs based on their historical data, enabling better placement or migration decisions. [9] raises the need for different placement strategies based on respective traffic patterns and workload categories, or the DC operator's goal. The proposed approach introduces a dynamic VM placement method based on DQN (Deep Q-Network) which utilizes six placement heuristics in its action space. The objective is to enable the agent to select the most suitable placement heuristic for specific situations and goals, ultimately improving the efficiency and effectiveness of VM placement.

Consolidating VMs onto a smaller number of active physical machines (PMs) can lead to higher peak temperatures, requiring a colder air supply at the inlets to maintain standard operational temperatures. This trade-off between reduced compute energy consumption and increased cooling energy usage may often have negative outcomes on the overall energy consumption, Thus, a few studies have explored thermal-aware or holistic solutions. For instance, [31] proposes GRANITE, a holistic approach based on using overall total power consumption including the cooling system's. [20] propose a mixed integer linear programming (MILP) and a greedy heuristic for a temperature-aware virtual data center embedding scheme. Similarly, [24] proposes a greedy algorithm that allocates VMs on PMs with the lowest predicted CPU and inlet temperature.

## 1.5 Our Contributions

Our major contributions are summarized as follows;

1. We implement predictive Machine Learning (ML) models to enhance the accuracy of power consumption prediction for physical server machines (PMs). This is achieved through the introduction of a novel feature space that incorporates real-time data from the PMs' embedded heat sensors, internal fan speed, and Performance Monitoring Counters (PMCs).

2. We introduce the concept of "Criticality" and the use of all internal components' critical temperature thresholds for precise hotspot detection and avoidance.

3. We develop a scalable, proactive, energy-aware, and thermal-aware VM Placement algorithm that effectively reduces energy consumption while ensuring appropriate resource allocation and hotspot avoidance.

4. We perform a series of experiments on an actual cluster of Physical Machines (PMs) within a private data center. These experiments provide a comprehensive evaluation of all Machine Learning (ML) models, as well as our implemented VM placement algorithm.

To the best of our knowledge today, there is no existing work that considers the temperature and critical temperature thresholds of all PM's internal components in the context of power prediction, placement, or hotspot avoidance. Moreover, there is very limited research works that implement and test their solutions in a real data center.

## 1.6 Plan of the Thesis

This thesis is organized as follows: In Chapter 1, we presented the motivation and background of our study. Chapter 2 describes in further detail the environment and load emulation methodology for all the experiments conducted in this study. Chapter 3 provides the implementation of power models that will play a critical role in our proposed Virtual Machine placement algorithm that is presented in Chapter 4. Chapter 3 and Chapter 4 are organized as separate Conference and Journal papers, respectively. Finally, we conclude this thesis and discuss future works in Chapter 5.

# Chapter 2

# Experiment Environment

This chapter outlines our experimental setup that is utilized for conducting all the experiments of the research papers presented in Chapter 3 and Chapter 4. Additionally, we detail our methodology for generating Virtual Machine (VM) loads and the collection of both software and hardware-related data.

## 2.1 Cluster Architecture

All experiments have been conducted in a private data center on a cluster of 8 HP ProLiant BL460c G8 physical machines (PMs) mounted on an HP C7000 chassis. As described in Table 1, each of those PMs consists of 2 processors of 8 cores, 16 threads, and 128 GB RAM. Our test bed runs with OpenStack as the cloud platform for the management of the Virtual Machines (VMs) and follows a Three-Node architecture that consists of 3 systems: Controller Node, Compute Node, and Storage Node [39] (See Figure 4). One PM is used as the controller node where most of the OpenStack services run and supply API, scheduling, and other shared services for the cluster. 6 PMs run as compute nodes where VM instances, also known as Nova compute instances are deployed, and 1 PM is used as the storage node to host data. In addition, a custom VM management module is

| | |
|---|---|
| Chassis | HP C7000 $\times$ 1 |
| Blade/PM | ProLiant BL460c Gen8. $\times$ 8 |
| Processor | Intel(R) Xeon(R), 8 core, 16 threads, 2.70GHz. $\times$ 2 |
| Memory | 128GB RAM (DIMM DDR3). $\times$ 2 |
| Disk | HDD 900GB. $\times$ 2 |

Table 1: Cluster & Physical Machine's Description.

implemented on the controller node for conducting data collection, executing experiments, and integrating new VM placement algorithms.

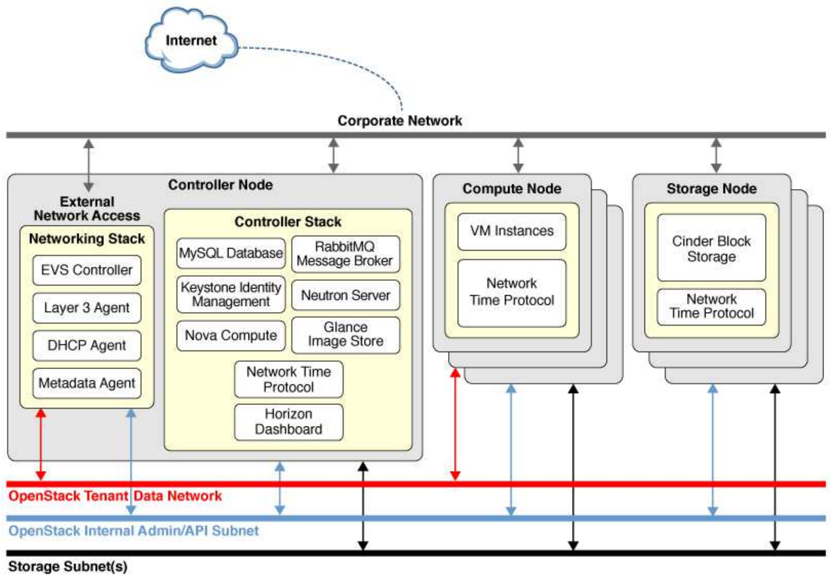


Figure 4: Three-Node Architecture of OpenStack [39]

## 2.2 Experiments

### 2.2.1 Load Emulation and Data Collection

Our environment can handle a maximum of 180 VMs with each compute node/PM having a VM limit set to 30. All VMs use an Ubuntu 20.04 LTS image [53] with a flavor of 2 VCPU, 4 GB Memory, and 20GB of disk. We illustrate in Figure 5 our process for generating real-like loads on VMs. All VMs are created with an image consisting of a load emulation Python script and stress software (stress-ng). Stress-ng [51] is a tool for workload generation that can subject a system to a configurable measure of resource utilization such as CPU, memory, and disk stress. First, a unique VM trace file from a dataset of Microsoft Azure's cloud VM traces [11] is transferred through SSH and SCP to the VM. After a trigger signal is sent, the load emulator uses stress-ng to emulate the load by replicating the load levels as recorded in the Azure trace file as a time series. This enables the VMs to have real-like resource usage levels and patterns on the PMs. Indeed, Figure 6 illustrate the results of an experiment demonstrating the similarity in usage pattern between the measured CPU utilization of a VM and the respective trace during load emulation.

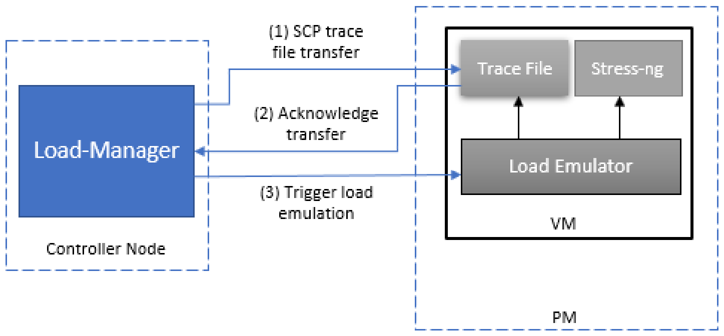


Figure 5: VM Load Emulation.

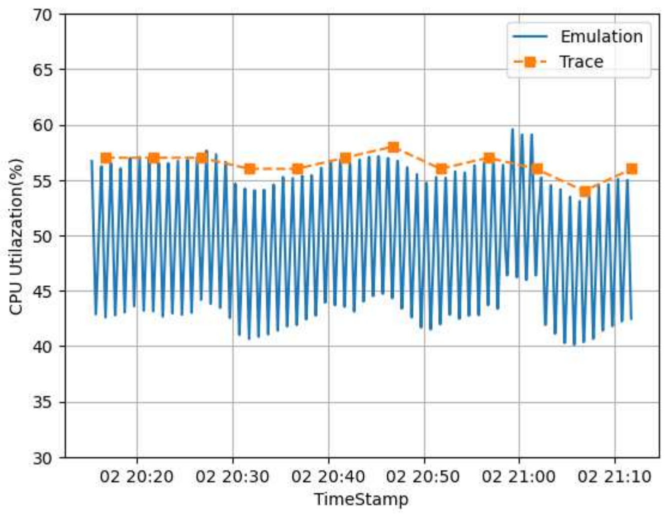


Figure 6: Emulation v/s Trace CPU Utilization pattern

The controller node is also used for collecting and storing the monitoring data of each PM. OpenStack and Linux commands (e.g., vmstat, lsCPU) are used for recording all software-related data such as the Performance Monitoring Counters (PMCs). In terms of hardware data, the real-time heat sensor data, internal fan speed, and power measurements are collected using IPMItool [40] through the Intelligent Platform Management Interface (IPMI) which is a standardized message-based hardware management interface.

### 2.2.2 Static VM Deployment

During our experiments, we encountered several technical issues. These included VM deployments failing or experiencing large delays, as well as connection failures during load emulations. These challenges mainly arose from technical problems related to the network interfaces and the presence of faulty cables. Thus, to guarantee failure-free and consistent experiments, we employ static instead of dynamic VM deployment. VMs are first predeployed, checked for their operational status and network connection, and kept idle. After the state of the machines has stabilized, dynamic VM deployments are emulated by selecting an idle VM from the VM pool, establishing a connection, and emulating workloads based on specific trace files. Since all VMs are homogeneous with similar configurations and capacities, idle VMs can be picked randomly and characterized using only their assigned trace file. Additionally, any variation caused by static deployments only impacts the overall static or idle power usage, which has a minimal influence on our study since our primary focus is dynamic power consumption.

# Chapter 3

# A Sensor Predictive Model for Power Consumption using Machine Learning

This chapter was accepted for publication in the IEEE CloudNet 2023 Conference, titled "A Sensor Predictive Model for Power Consumption using Machine Learning", written by N. Moocheet, B. Jaumard, P. Thibault, and L. Eleftheriadis.

## 3.1 Abstract

Reducing the power consumption of computing devices remains a challenge for the data center industry. In 2022, it represents approximately 2% of global electricity consumption and 1% of global greenhouse gas emissions. In addition, data centers must integrate the 5G and B5G challenges into their strategies, by increasing the computing resources available to face higher-quality service constraints. Indeed, 5G and B5G future networks are increasingly software-oriented and therefore, rely heavily on cloud computing to process large amounts of data from multiple sources in real-time.

Several research works on energy management have been proposed to ensure a reduction of the energy consumed by the various components of a data center (e.g., software,

computing devices, or cooling systems). However, to optimize the energy consumption of computing devices (e.g., virtual machines/container operations), it is essential to have an accurate model for predicting power consumption. Thus, we propose in this study a new sensor predictive model to predict the dynamic power consumption of cloud computing devices with high accuracy.

Our proposal takes advantage of the various sensors that are now embedded in physical machines, or more generally in cloud server machines, as well as Performance Monitoring Counters to implement a Machine Learning power prediction model.

The performance evaluation results confirm that our power consumption prediction models outperform previous literature models in terms of accuracy. Indeed, our best model achieves a $R^2$ score of 93.6% which is higher than the compared baseline model by 21.1%.

## 3.2 Introduction

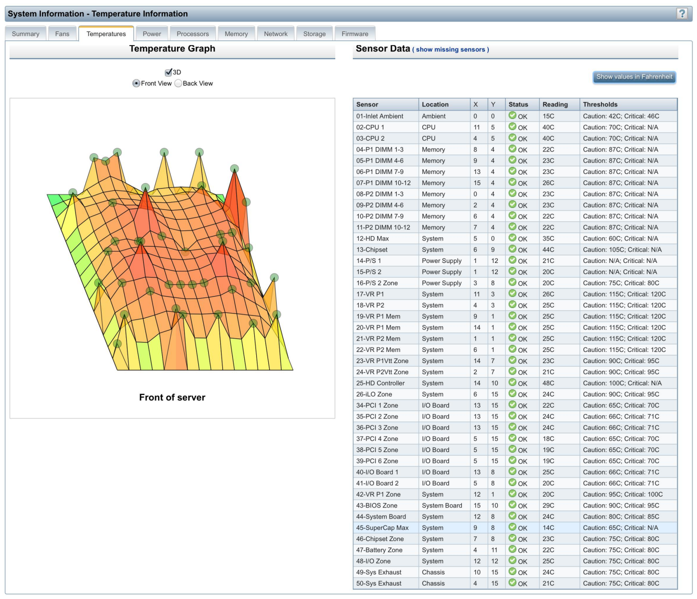
With the emergence of 5G and B5G future networks, there is an ever-increasing demand for improved cloud service capabilities and energy efficiency. Future networking technologies are evolving to be more software-oriented (as virtual processes replace physical processes) and thus, rely significantly on cloud resources and the associated infrastructures. Furthermore, the rapid growth of highly network-dependent and low-latency technologies such as Internet of Things (IoT), Smart Cities, and Self Driving Cars, increases, even more, the strain on Data Centers (DC). As a result, there is a global rise in energy consumption by Data Centers (DC). The International Energy Agency estimates that 1-1.5% of all global electricity is used by data centers and that by 2025, they will consume 1/5 of the world's power supply. Data Center energy minimization has therefore become a huge challenge internationally. Communication Service Providers (CSP) continue to invest heavily to address the energy issue. Even a 1% improvement of energy efficiency in a Data Center can lead to significant savings in operational costs and in reducing carbon footprint [23].

The key concepts that are being applied to achieve DC energy efficiency are energy-aware virtual machine (VM) or container operations: scheduling, placement, and migration [36]. The main idea consists of managing VMs among the physical machines (PM) of a Data Center in such a way that the overall energy consumption is reduced. To implement energy-aware DC VM management strategies, it is essential to have an accurate model for power consumption prediction. However, existing solutions, mostly analytical power models, fail to capture multiple non-linear inter-dependencies that affect the power consumption of PMs in a data center. In addition, most of these models use resource utilization, more specifically, Central Processing Unit (CPU) utilization as the only feature for power prediction. These models fail to capture the other essential dependencies such as the heat generation inside the PMs or possible differences in workload characteristics such as being I/O intensive, thus, leading to inaccurate power models.

In this paper, we propose a novel sensor predictive model for accurate power consumption prediction for server physical machines (PMs) in the context of dynamic traffic. Server PMs are today equipped with multiple sensors: heat sensors, fan speed meters, and power meters. Sensors help with their real-time measurements, to detect hotspots, i.e., high temperatures that can lead to unnecessary downtime [28]. For instance, power meters are very useful for detecting if the current power consumption reaches or exceeds the configured power budget. Yet, there is a need for predicting the required power (the objective of the present study) in order to proactively manage virtual machines in order to minimize the DC energy consumption, and facilitate the DC management [35].

Figure 7 illustrates a typical heatmap that can be generated, using various temperature sensor measurements, for the physical machine we used in our experiments. Therein, each dot represents the temperature of one particular sensor. A detailed description of the sensors is provided in Section 3.5. In addition, we propose the use of Performance Monitoring Counters (PMCs) for improved characterization of the dynamic workload and resource utilization.

We conducted various experiments on a private DC with a 6 PM cluster, with each PM equipped with multiple onboard heat sensors, fan speed counters, and power meters to investigate traffic load factors that impact energy consumption. We then implemented four machine learning models: Multiple Linear Regression (MLR), Multi-Layer Perceptron (MLPRegressor), Long Short-Term Memory (LSTM), and XGBoost. All four models used real-time sensor measurements and PMCs to provide accurate power predictions. These are compared to the power prediction derived from the most commonly used analytical formulas, see, e.g., Fan *et al.* [12]. The reported results show that our proposed machine learning models can accurately predict dynamic power values, much more than the classical analytical power formula, under different traffic load characteristics. The paper is



Sensor Data ( show missing sensors )

| Sensor | Location | X | Y | Status | Reading | Thresholds |
|---|---|---|---|---|---|---|
| 01-Inlet Ambient | Ambient | 0 | 0 | ✓ OK | 15C | Caution: 42C; Critical: 46C |
| 02-CPU 1 | CPU | 11 | 5 | ✓ OK | 40C | Caution: 70C; Critical: N/A |
| 03-CPU 2 | CPU | 4 | 5 | ✓ OK | 40C | Caution: 70C; Critical: N/A |
| 04-P1 DIMM 1-3 | Memory | 8 | 4 | ✓ OK | 22C | Caution: 87C; Critical: N/A |
| 05-P1 DIMM 4-6 | Memory | 9 | 4 | ✓ OK | 23C | Caution: 87C; Critical: N/A |
| 06-P1 DIMM 7-9 | Memory | 13 | 4 | ✓ OK | 23C | Caution: 87C; Critical: N/A |
| 07-P1 DIMM 10-12 | Memory | 15 | 4 | ✓ OK | 26C | Caution: 87C; Critical: N/A |
| 08-P2 DIMM 1-3 | Memory | 0 | 4 | ✓ OK | 23C | Caution: 87C; Critical: N/A |
| 09-P2 DIMM 4-6 | Memory | 2 | 4 | ✓ OK | 23C | Caution: 87C; Critical: N/A |
| 10-P2 DIMM 7-9 | Memory | 6 | 4 | ✓ OK | 22C | Caution: 87C; Critical: N/A |
| 11-P2 DIMM 10-12 | Memory | 7 | 4 | ✓ OK | 22C | Caution: 87C; Critical: N/A |
| 12-HD Max | System | 5 | 0 | ✓ OK | 35C | Caution: 60C; Critical: N/A |
| 13-Chipset | System | 6 | 9 | ✓ OK | 44C | Caution: 105C; Critical: N/A |
| 14-P/S 1 | Power Supply | 1 | 12 | ✓ OK | 21C | Caution: N/A; Critical: N/A |
| 15-P/S 2 | Power Supply | 1 | 12 | ✓ OK | 20C | Caution: N/A; Critical: N/A |
| 16-P/S 2 Zone | Power Supply | 3 | 8 | ✓ OK | 20C | Caution: 75C; Critical: 80C |
| 17-VR P1 | System | 11 | 3 | ✓ OK | 26C | Caution: 115C; Critical: 120C |
| 18-VR P2 | System | 4 | 3 | ✓ OK | 25C | Caution: 115C; Critical: 120C |
| 19-VR P1 Mem | System | 9 | 1 | ✓ OK | 25C | Caution: 115C; Critical: 120C |
| 20-VR P1 Mem | System | 14 | 1 | ✓ OK | 25C | Caution: 115C; Critical: 120C |
| 21-VR P2 Mem | System | 1 | 1 | ✓ OK | 25C | Caution: 115C; Critical: 120C |
| 22-VR P2 Mem | System | 6 | 1 | ✓ OK | 25C | Caution: 115C; Critical: 120C |
| 23-VR P1Vtt Zone | System | 14 | 7 | ✓ OK | 23C | Caution: 90C; Critical: 95C |
| 24-VR P2Vtt Zone | System | 2 | 7 | ✓ OK | 21C | Caution: 90C; Critical: 95C |
| 25-HD Controller | System | 14 | 10 | ✓ OK | 48C | Caution: 100C; Critical: N/A |
| 26-iLO Zone | System | 6 | 15 | ✓ OK | 24C | Caution: 90C; Critical: 95C |
| 34-PCI 1 Zone | I/O Board | 13 | 15 | ✓ OK | 22C | Caution: 65C; Critical: 70C |
| 35-PCI 2 Zone | I/O Board | 13 | 15 | ✓ OK | 24C | Caution: 66C; Critical: 71C |
| 36-PCI 3 Zone | I/O Board | 13 | 15 | ✓ OK | 24C | Caution: 66C; Critical: 71C |
| 37-PCI 4 Zone | I/O Board | 5 | 15 | ✓ OK | 18C | Caution: 65C; Critical: 70C |
| 38-PCI 5 Zone | I/O Board | 5 | 15 | ✓ OK | 19C | Caution: 65C; Critical: 70C |
| 39-PCI 6 Zone | I/O Board | 5 | 15 | ✓ OK | 19C | Caution: 65C; Critical: 70C |
| 40-I/O Board 1 | I/O Board | 13 | 8 | ✓ OK | 25C | Caution: 66C; Critical: 71C |
| 41-I/O Board 2 | I/O Board | 5 | 8 | ✓ OK | 20C | Caution: 66C; Critical: 71C |
| 42-VR P1 Zone | System | 12 | 1 | ✓ OK | 20C | Caution: 95C; Critical: 100C |
| 43-BIOS Zone | System Board | 15 | 10 | ✓ OK | 29C | Caution: 90C; Critical: 95C |
| 44-System Board | System | 12 | 8 | ✓ OK | 24C | Caution: 80C; Critical: 85C |
| 45-SuperCap Max | System | 9 | 8 | ✓ OK | 14C | Caution: 65C; Critical: N/A |
| 46-Chipset Zone | System | 7 | 8 | ✓ OK | 23C | Caution: 75C; Critical: 80C |
| 47-Battery Zone | System | 4 | 11 | ✓ OK | 22C | Caution: 75C; Critical: 80C |
| 48-I/O Zone | System | 12 | 12 | ✓ OK | 25C | Caution: 75C; Critical: 80C |
| 49-Sys Exhaust | Chassis | 10 | 15 | ✓ OK | 24C | Caution: 75C; Critical: 80C |
| 50-Sys Exhaust | Chassis | 4 | 15 | ✓ OK | 21C | Caution: 75C; Critical: 80C |

Figure 7: Heat Map of a HPE ProLiant BL460C Gen8 PM

organized as follows. Section 3.3 contains an overview of the literature review on the prediction of server power consumption. A first study in Section 3.4 exhibits the requirements and challenges for designing accurate power prediction models considering the signature characteristics of the PMs using PMs and sensor measurements vs. analytical formulas.

We then propose in Section 3.5, four different machine learning models, which rely on PMCs and real-time sensor measurements. The prediction ML algorithms are discussed in Section 3.6. Finally, we present our performance results in Section 3.7. Conclusions are drawn in the last section.

## 3.3 Literature Review

Numerous studies have been proposed in the area of power prediction in the context of energy management in data centers (DCs). Predictive energy or power consumption models are crucial for energy management in DCs, as most energy-efficient virtual machine management solutions require their use. For example, Beloglazov *et al.* [4] proposed energy-aware scheduling based on allocating each VM on a PM that has the lowest predicted increase in power consumption, using a linear to CPU utilization model inspired by Fan *et al.* [12].

### 3.3.1 Power Consumption Models with CPU only

Simple regression model is the most popular of all models for power consumption prediction [25]. Fan *et al.* [12] used the relationship between the CPU utilization and total power consumption of a server, to propose a simple linear model. They assume that the power increases linearly to the CPU utilization starting from the idle power to the power at maximum utilization. In the same work, they also proposed a non-linear model that has the same dependencies with the addition of a calibration variable that aims to minimize the square error of their model. Beloglazov *et al.* [4], Kavanagh *et al.* [27] and Lien *et al.* [33] are other examples of research works that propose power models that are mainly dependent on CPU utilization, idle state power consumption, and peak utilization state power consumption. However, while these models are simple and practical to use, they are only suitable for power modeling in CPU-intensive servers. They are likely to produce large prediction errors when the servers are running workloads that are for example, more I/O or memory intensive.

### 3.3.2 More General Power Consumption Models

To obtain better accuracy, several works propose power models that take into account several server components in addition to the CPU. For example, Basmadjian *et al.* [3] proposed an additive model that considers the memory, disk, fan speed, and CPU frequency to model the energy consumption. Similarly, Perumal *et al.* [41] and Song *et al.* [50] model the server energy consumption as the sum of the energy consumed by the CPU, memory, disk, network interface card (NIC), and mainboard. Tudor et al. [52] uses the clock frequency and service time of a memory and I/O request to model the energy consumption of advanced RISC machine (ARM) multicore servers. While more accurate than simple regression models, the analytical formulas of these additive models are often valid for only a few sets of servers since the power consumption of each component depends on its configurations and hardware. Although more accurate than simple regression models, the analytical formulas of these additive models lack generality: they are often only valid for a few sets of servers since the power consumption of each component depends on its configurations and hardware. Alan *et al.* [1] and Li *et al.* [32] propose Multiple Regression Models that use factors such as memory access rate, hard disk I/O, and network I/O, in addition to CPU utilization. Furthermore, Kansal *et al.* [26] propose Joulemeter, a solution to provide virtual machine (VM) power metering functionality by considering the number of last level cache (LLC) misses, in addition to the generic components such as I/O, memory, and CPU. Xiao *et al.* [59] and Bircher *et al.* [7] propose a different category of power modeling technique that uses performance monitoring counters (PMCs). This implies the use of the system or application-level performance-related statistics for modeling power consumption at the sub-system's or VM's level. For example, Bircher *et al.* consider PMCs such as LLC, translation lookaside buffer (TLB), Direct memory access (DMA), and Interrupt to account for the power consumption of components such as the processor, memory, disk, and I/O controller.

### 3.3.3 Temperature: A Key Parameter for Power Prediction

Temperature is another essential feature that very few power and energy prediction-related works considered so far.

Rezaei *et al.* [44] and Wang *et al.* [56] show the direct effect of temperature on the power consumption of cloud data centers. They both propose an improved version of the very popular linear analytical formulas from Fan *et al.* [12] that uses the inlet and outlet temperature of a server/physical machine to improve the power consumption prediction accuracy. Similarly, Witkowski *et al.* [58] make use of clock frequency and CPU cores' temperature in their proposed technique for power consumption estimation of physical machines or systems in High-Performance Computing (HPC) clusters.

### 3.3.4 Concluding Remarks

While the first generation of studies for server power modeling relied mostly on analytical formulas, the second generation try to take advantage of temperature-aware solutions. However, most of these last studies only consider the CPU, ambient, or inlets' temperature while the temperature of the other system-related components is also crucial for an accurate power prediction model. The goal of our study is indeed to investigate these other critical parameters for accurate power prediction.

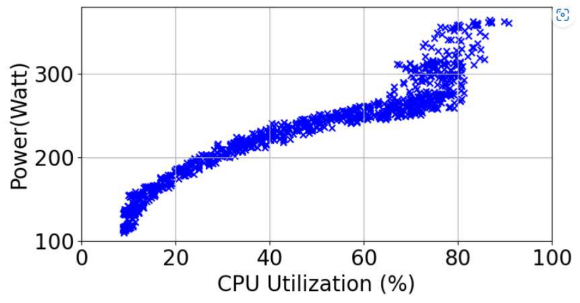
## 3.4 Real-time Sensor Measurements vs. Analytical Formulas

We discuss here the benefits of using a data-driven approach for an accurate prediction of power consumption while identifying the right features. We first investigate the key features to be considered for the traffic workload and then those for the computing resources in order to accurately predict the power consumption. We then compare the results of a real-time data-driven approach against the use of analytical formulas which are not only for static traffic but too generic to meet the need for accurate power prediction.

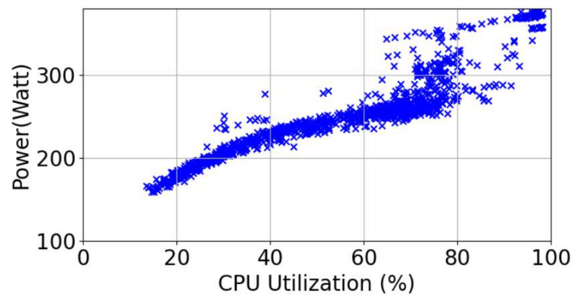
### 3.4.1 Workload Characterization

A data center (DC) is a facility composed of networked computers, storage systems, and computing infrastructure. Each server also called a host or physical machine (PM), typically manages virtual machines (VMs) or containers of multiple categories. PMs can be used as, e.g., database servers, file servers, web servers, game servers, and application servers [25]. Today, 5G networks are foreseen to rely on the cloud which will be physically distributed among multiple DCs in different locations to dynamically deploy Virtualized Network Functions (VNFs) [47]. VNFs are software implementations of network equipment of several categories, such as routers, firewalls, load balancers, or mobile core network components that can be deployed on Virtual Machines (VMs).

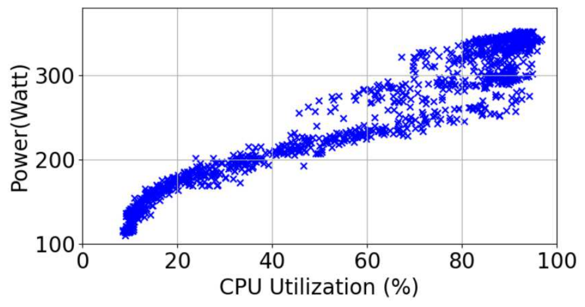
Therefore, due to these multiple categories of VMs/containers, the workload of PMs is highly heterogeneous. PMs may be operated with different software/hardware components according to their VMs' workload characteristics. For example, a PM used mostly as a game server will have considerably more CPU usage than those used as database servers. Power models that are linear to CPU utilization assume that the impact of those different



(a) Exp. 1: CPU Only



(b) Exp. 2: CPU & Memory



(c) Exp. 3: CPU & Disk

Figure 8: CPU utilization to Power relation for different workloads.

load characteristics is negligible, however, we demonstrate otherwise in Fig. 8. We conducted several experiments, with emulated loads of varying characteristics. On a set of 120 VMs, we first run a CPU-usage-based load emulation, followed by other experiments with similar CPU usage levels but with additional simulated memory and disk operations. Indeed, Exp. 2 and Exp. 3 in Fig. 8, shows that the relation between the CPU utilization and power consumption varies based on the load characteristic. Details of the experimental setup and load simulation is later provided in Section 3.7.

A first conclusion is that it is essential to have a real-time accurate characterization of the VM loads and tasks running on the different PMs to accurately model their power consumption. While resource utilization such as CPU utilization is the most used metric to characterize the workload of a PM, other parameters should be collected in the workload traces, while preserving security or privacy agreements. It is acknowledged that the CPU is the heaviest power consumer [6] and is also relatively easy to measure, thus, the majority of DC energy optimization-related works use models that are solely dependent on CPU utilization and ignore the power consumption of the other load or task components. However, as discussed in [2], the total estimated power consumption by these components is not insignificant. In addition, as shown in Fig. 8 and previously explained, for workloads of different characteristics, there will be large variations in power consumption even at similar CPU utilization levels, which will result in large prediction errors for those CPU Utilization dependent predictive models.

### 3.4.2 Heterogeneous PMs & Temperature

Data Centers, especially large-scale ones, are highly heterogeneous. They contain a huge number of Physical Machines (PMs) which are very likely to be from different manufacturers. The physical architecture, configuration, and performance of those PMs will have large variations based on their manufacturer and hardware type. Thus, using standard analytical models is highly impractical for such data centers, since they require to be calibrated through various experiments for each type of hardware. Furthermore, in a dynamic data center, each PM is very likely to have different inlet and internal temperatures based on their location inside the DC, in relation to the cooling units and room air circulation. Indeed, Fig. 9 and Fig. 10 displays the difference in the inlet and CPU temperature between 3 PMs, all with similar hardware (HP ProLiant BL460c G8), with similar configurations including energy-based configurations, and running similar loads. Thus, temperature is a key factor that must be considered in relation to power consumption prediction, since in general, much of the electrical energy that goes into those server PMs gets turned into heat. However, most existing thermal-aware power or energy models, only consider the CPU or inlet temperature and ignore the other internal components of the PM. Getting a PM's inlet temperature without sensors often consists of modeling DC room heat circulation through Computational Fluid Dynamics (CFD) which is highly computational and thus, not appropriate for dynamic large-scale DC.

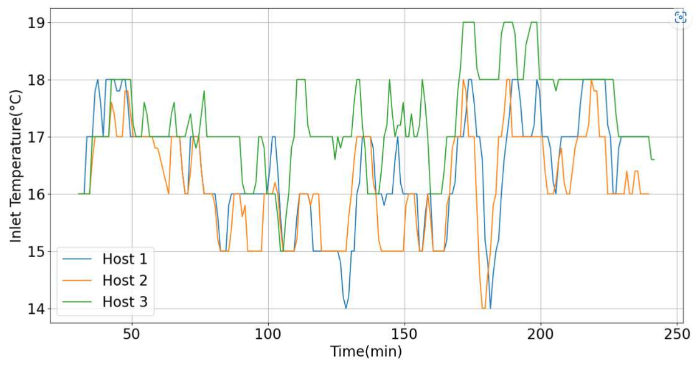


Figure 9: Inlet temperature under similar load

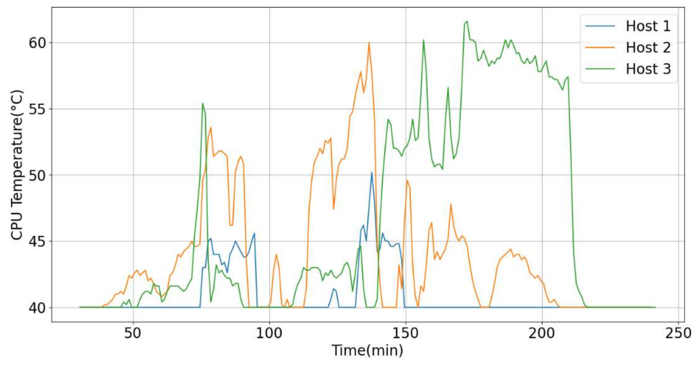


Figure 10: CPU temperature under similar load.

## 3.5 Machine Learning Model

To develop an effective model for accurately predicting the power consumption of Physical Machines (PMs) within the context of Virtual Machine (VM) placement for heterogeneous dynamic data centers that handle multiple categories of traffic load, it is imperative to consider comprehensive factors such as detailed resource utilization and temperature across all PM components. Thus, in this section, we propose our methodology which consists of the following;

- To use Performance Monitoring Counters (PMCs) for improved workload characterization and a detailed measure of resource utilization.
- To consider the temperature of various internal components of physical machines, as well as their internal fan speeds.
- Implementing a Multiple Linear Regression, Multi-layer Perceptron Regressor (MLPRegressor), Long Short-Term Memory (LSTM), and XGBoost models that take advantage of the enhanced feature space to provide highly accurate power predictions.
- Predict the power consumption of a PM at a time t using the CPU utilization of the PM at t, PMCs at t-1, and all sensors' data at time t-1, as input features for our ML models, since PMCs & temperature cannot be extrapolated as easily as the CPU utilization.

### 3.5.1 Performance Monitoring Counters

Performance Monitoring Counters (PMCs) consist of various kinds of system data such as CPU, memory, and disk usage that are often used by system administrators to monitor systems for performance or behavior problems, as well as, to examine the resource usage of systems. We take advantage of those PMCs, especially the system data related to resource usage to improve the characterization of the workload and resource utilization on the physical machines (PMs). PMCs can be used to model the power consumption of the memory, disk, and other subsystems outside of the processor.[59][7]. For example, the amount of Interrupts, context switching, and time spent by the CPU waiting for I/O captures the I/O

Table 2: Performance Monitoring Counters (PMCs)

| **PMCs** | **Description** |
|---|---|
| *r* | Number of processes in a running state |
| *b* | Number of processes in uninterruptible sleep state |
| *swpd* | Amount of virtual memory used |
| *free* | Amount of idle memory |
| *buff* | Amount of memory used as buffers |
| *cache* | Amount of memory used as cache |
| *si* | Amount of memory swapped in from disk (/s) |
| *so* | Amount of memory swapped to disk (/s) |
| *bi* | Blocks received from a block device (blocks/s) |
| *bo* | Blocks sent to a block device (blocks/s) |
| *in* | Number of interrupts per second |
| *cs* | Number of context switches per second |
| *us* | CPU time spent running non-kernel code |
| *sy* | CPU time spent running kernel code |
| *us_t* | Overall CPU utilization |
| *id* | CPU time spent idle |
| *wa* | CPU time spent waiting for I/O |
| *st* | CPU time stolen from a virtual machine |
| *cpu_freq* | CPU frequency |

usage and thus, the estimated power consumed by these I/O operations. While there exist several PMCs that can be collected through various tools, in this work we consider those performance statistics that can be easily collected in real-time through built-in monitoring utilities such as ’vmstat’ [34]. Table 2 lists and describes the set of PMCs that we collect and consider in our experiments and model implementation.

### 3.5.2 Embedded Sensors

Most physical machines (PMs) in data centers now contain multiple temperature sensors embedded in their internal components to monitor their temperature and used to mitigate hotspots. Each of these sensors has a critical temperature threshold which, when reached, can force the PM to shut down. Moreover, reaching high temperatures also influence power consumption due to the increase in silicon leakage current. Thus, we consider the temperature of those components as key feature for an accurate power model. We make use of an open-source tool called 'ipmitool' [40], to collect those sensor data in real time to derive a complete temperature profile for each PM. As a result, we are able to take into account the temperature of several components in addition to the CPUs, such as the Memory, I/O, and other system-related components, including the heat generated at the outlets (exhausts). Additionally, we also collect the speed of internal fans as they also influence temperature and power consumption. Adding component temperature and fan speed along with performance monitoring counters (PMCs) to our feature space allows our models to derive and accurately account for the power consumption of all the sub-components inside the physical machine, thus greatly improving the performance of the power model. Figure 11 shows a 2-dimensional version of Fig. 7, which is a heat map generated for one of our PMs, as well as the location of the 28 internally embedded temperature sensors that we consider in this study. Fig. 12 then provides the corresponding sensors' description, coordinates, critical thresholds, and an example of real-time temperature readings.

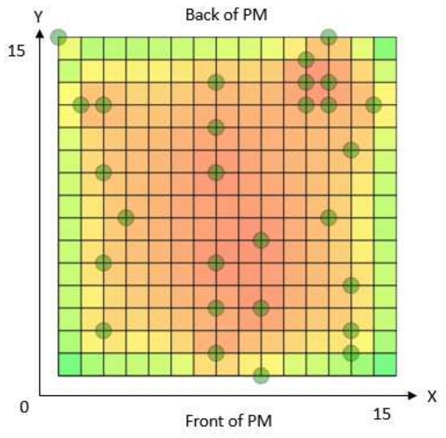
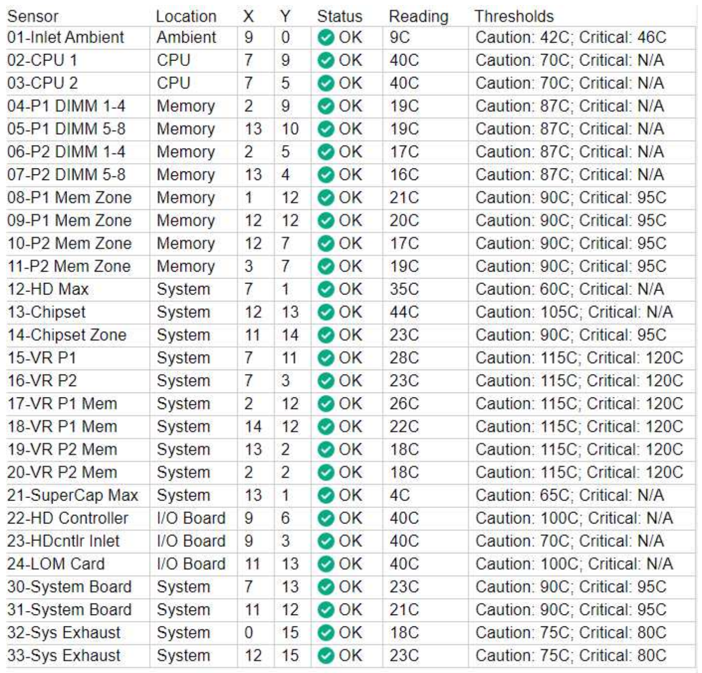


Figure 11: 2D Heat Map of a PM (HP ProLiant BL460c G8) with Sensors location

### 3.5.3 Data pre-processing

Data pre-processing is a crucial step that can affect the performance of machine learning (ML) models, especially for data from a dynamic environment with high fluctuations and



| Sensor | Location | X | Y | Status | Reading | Thresholds |
|---|---|---|---|---|---|---|
| 01-Inlet Ambient | Ambient | 9 | 0 | ✓ OK | 9C | Caution: 42C; Critical: 46C |
| 02-CPU 1 | CPU | 7 | 9 | ✓ OK | 40C | Caution: 70C; Critical: N/A |
| 03-CPU 2 | CPU | 7 | 5 | ✓ OK | 40C | Caution: 70C; Critical: N/A |
| 04-P1 DIMM 1-4 | Memory | 2 | 9 | ✓ OK | 19C | Caution: 87C; Critical: N/A |
| 05-P1 DIMM 5-8 | Memory | 13 | 10 | ✓ OK | 19C | Caution: 87C; Critical: N/A |
| 06-P2 DIMM 1-4 | Memory | 2 | 5 | ✓ OK | 17C | Caution: 87C; Critical: N/A |
| 07-P2 DIMM 5-8 | Memory | 13 | 4 | ✓ OK | 16C | Caution: 87C; Critical: N/A |
| 08-P1 Mem Zone | Memory | 1 | 12 | ✓ OK | 21C | Caution: 90C; Critical: 95C |
| 09-P1 Mem Zone | Memory | 12 | 12 | ✓ OK | 20C | Caution: 90C; Critical: 95C |
| 10-P2 Mem Zone | Memory | 12 | 7 | ✓ OK | 17C | Caution: 90C; Critical: 95C |
| 11-P2 Mem Zone | Memory | 3 | 7 | ✓ OK | 19C | Caution: 90C; Critical: 95C |
| 12-HD Max | System | 7 | 1 | ✓ OK | 35C | Caution: 60C; Critical: N/A |
| 13-Chipset | System | 12 | 13 | ✓ OK | 44C | Caution: 105C; Critical: N/A |
| 14-Chipset Zone | System | 11 | 14 | ✓ OK | 23C | Caution: 90C; Critical: 95C |
| 15-VR P1 | System | 7 | 11 | ✓ OK | 28C | Caution: 115C; Critical: 120C |
| 16-VR P2 | System | 7 | 3 | ✓ OK | 23C | Caution: 115C; Critical: 120C |
| 17-VR P1 Mem | System | 2 | 12 | ✓ OK | 26C | Caution: 115C; Critical: 120C |
| 18-VR P1 Mem | System | 14 | 12 | ✓ OK | 22C | Caution: 115C; Critical: 120C |
| 19-VR P2 Mem | System | 13 | 2 | ✓ OK | 18C | Caution: 115C; Critical: 120C |
| 20-VR P2 Mem | System | 2 | 2 | ✓ OK | 18C | Caution: 115C; Critical: 120C |
| 21-SuperCap Max | System | 13 | 1 | ✓ OK | 4C | Caution: 65C; Critical: N/A |
| 22-HD Controller | I/O Board | 9 | 6 | ✓ OK | 40C | Caution: 100C; Critical: N/A |
| 23-HDcntlr Inlet | I/O Board | 9 | 3 | ✓ OK | 40C | Caution: 70C; Critical: N/A |
| 24-LOM Card | I/O Board | 11 | 13 | ✓ OK | 40C | Caution: 100C; Critical: N/A |
| 30-System Board | System | 7 | 13 | ✓ OK | 23C | Caution: 90C; Critical: 95C |
| 31-System Board | System | 11 | 12 | ✓ OK | 21C | Caution: 90C; Critical: 95C |
| 32-Sys Exhaust | System | 0 | 15 | ✓ OK | 18C | Caution: 75C; Critical: 80C |
| 33-Sys Exhaust | System | 12 | 15 | ✓ OK | 23C | Caution: 75C; Critical: 80C |

Figure 12: Snapshot of a real-time sensor reading & description

features of various measurement units and scales. The dataset was first cleaned by removing samples with any missing values. Samples for each 1-minute interval were then grouped and averaged for data reduction and more importantly to account for high fluctuations. Spearman rank-order correlation coefficient was then used for feature selection. Spearman's rank coefficient of correlation is a nonparametric measure of the statistical dependence of ranking between two variables or features. More specifically, it measures the strength and direction of the association between two ranked variables, especially for non-linear correlations. The Spearman coefficient varies between -1 and +1, with 0 implying no correlation and -1 or +1 implying an exact monotonic relationship. Thus, after calculating the correlation between each independent feature and the measured power consumption, we discarded all features that have a Spearman coefficient of 0. Spearman rank-order correlation coefficient is written:

$$\rho = 1 - (6 \sum_i d_i^2)/n(n^2 - 1), \tag{1}$$

where $n$ is the number of data points of the two features and $d_i$ the difference in ranks of the $i$th element. The features that were found to have 0 correlation with the power, thereby, removed from our feature set were the 'HD MAX', 'HDcntlr Inlet', and 'LOM Card' component's temperature and the 'swpd', 'si', 'so', and 'st' from the PMCs.

The data-set is standardized for the Machine Learning (ML) models to converge faster and have better performance. Indeed, features are measured at different scales and therefore, do not contribute equally to the model fitting which might create biases. Thus, we transform the features to a relatively similar scale or close to a normal distribution. A feature is standardized using equation (2) which results in a distribution with a standard deviation equal to 1 and an approximate mean of 0. The mean $\mu$ and standard deviation $\sigma$ of the training set are used to standardize the testing set to prevent the leakage of its distribution information. The standardization is written:

$$z_i = (x_i - \mu)/\sigma, \tag{2}$$

where $z_i$ is the $i$th value of the standardized feature, $x_i$ is the $i$th value of feature $x$, $\mu$ is the mean and $\sigma$ the standard deviation for feature $x$ .

Three datasets are derived: first dataset only contains PMC features; second dataset contains the fan speed and temperature-related features; third dataset contains all the features from datasets 1 and 2. For each ML model implementation, we derive 3 types of models using these 3 datasets.

All datasets are time-series data of 1 minute intervals with new shifted columns so that each row contains CPU utilization (labeled as us_$t$_$N$) and Power values at time $t$ and the required feature values at time $t-1$.

## 3.6 Machine Learning Algorithms

We now discuss the various machine learning algorithms.

### 3.6.1 Multiple Linear Regression Model (MLR)

Multiple linear regression (MLR) is an extension of simple linear regression that makes it possible to evaluate the linear relationships between a dependent variable and several independent or explanatory variables. We use MLR technique to model the relationship between our selected set of 45 independent features and our prediction target variable which is the power consumption of a physical machine. The advantage of this approach is that it leads to a more accurate and precise understanding of the association of each feature with the prediction output. A MLR model is defined as follows:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_i X_i + \varepsilon \tag{3}$$

where $Y$ is the dependent feature, $\beta_0$ is a constant term for the $y$-intercept, $\beta_i$ is the slope coefficient for $X_i$, which is the $i$th independent variable, and $\varepsilon$ the residual of the model.

### 3.6.2 Extreme Gradient Boosting Algorithm (XGBoost)

Extreme Gradient Boosting is an efficient, scalable, and open-source implementation of a tree-boosting system proposed by Chen *et al.* [10].

### 3.6.3 Multi-layer Perceptron (MLP) Regressor Model

A Multi-layer Perceptron Regressor (MLPRegressor) is a class of feed-forward artificial neural networks that performs regression. In a feed-forward neural network, the units are arranged into a graph without any cycle for sequential computations. Our model (Fig. 13

consists of a fully connected network with a first layer being the input-layer $(n = 45)$ that takes the values from the input features, 3 hidden layers, and the output-layer that outputs the power consumption prediction. The first hidden layer has 3 perceptrons or units, the

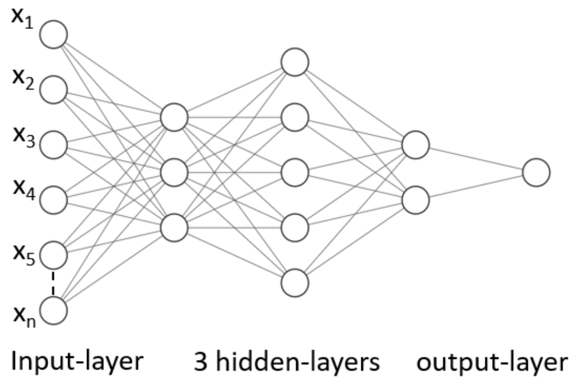


Figure 13: MLP Neural Network Architecture

second hidden layer has 5 units and the last hidden layer has 2 units. We use 'lbfgs' which is an optimizer in the family of quasi-Newton methods for weight optimization, as a solver in our MLP model training. As for the activation function, we use the 'identity' or Linear Activation Function which returns $f(x) = x$, where $x$ is the input value.

### 3.6.4 Long Short-Term Memory (LSTM) Neural Network

Long Short-Term Memory Network (LSTM) which was introduced by Hochreiter & Schmidhuber[22], is a specialized type of Recurrent Neural Network (RNN) designed for learning long-term dependencies. It is a specific recurrent neural network (RNN) architecture that was designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs, and therefore particularly effective on time-series data. Our Neural Network consists of an LSTM layer of 32 units, 2 dense hidden layers with 16 and 8 perceptrons or units, and finally an output layer of 1 unit. In addition, we add a dropout layer with a dropout rate of 0.1 after the first hidden layer to prevent overfitting. As for the activation function of the hidden layers, we use a simple identity function $f(x) = x$

which is commonly used for regression tasks. The optimization algorithm used to update the neural network during training is the Adaptive Moment Estimation (Adam) which is known to achieve faster convergence and better generalization.

## 3.7 Numerical Results

We implemented all proposed machine learning algorithms described in Section 3.6 with the models presented in Section 3.5. The models are then evaluated on a cluster of real VMs in a private data center. Results are then compared with the work of Fan *et al.* [12] and Wang *et al.* [56].

### 3.7.1 Experimental Setup, Load Emulation & Data Collection

All experiments have been conducted in a private data center on a cluster of 6 HP ProLiant BL460c G8 physical machines(PMs). OpenStack [57], an open-source cloud computing infrastructure software, is used for the deployment and management of Virtual Machines (VMs) on the cluster of server PMs. The workload consists of a set of 120 VMs of an Ubuntu 20.04 LTS image [53] and a flavor of 2 VCPU, 4 GB Memory, and 20GB of disk. Four of the PMs are used as compute nodes, one as a controller node to manage data collection and VM deployment, and lastly one PM as local block storage.

Load emulations on the VMs are done through stress software with Microsoft Azure's cloud VM traces [11]. Stress [51] is a tool for workload generation that can subject a system to a configurable measure of CPU, memory, and disk stress. The traces contain the measurements of the average CPU usage at 5-minute intervals for real VMs running on Azure servers. Thus, using these traces, we are able to emulate real-like levels of PM's CPU utilization. Memory and disk operations are added through the stress tool for generating traffic of different characteristics. Indeed, as seen previously in Fig. 8, multiple workloads

| **Workload** | **CPU(%)** | **Memory(%)** | **Disk (%)** | **PMs** |
|---|---|---|---|---|
| Training Sets | | | | |
| TR-CPU | 42 | 34 | 28 | PM1,PM2,PM3 |
| TR-MEM | 55 | 44 | 29 | PM1,PM2,PM3 |
| TR-DISK | 55 | 42 | 31 | PM1,PM2,PM3 |
| Testing Sets | | | | |
| TS-CPU | 47 | 35 | 29 | PM4 |
| TS-MEM | 54 | 46 | 29 | PM4 |
| TS-DISK | 57 | 43 | 31 | PM4 |
| TS-ALL | 48 | 48 | 31 | PM4 |

Table 3: Average Resource utilization by Workloads

with different characteristics are generated by varying load configurations of the stress tool. Details of these workloads, such as the average percentage resource utilization are provided in table 3. Workloads with the 'CPU' labels imply that they solely consisted of default CPU-based emulation from the Azure VM traces while the '-MEM' and '-DIsk' consisted of added memory and disk operations along with the VM traces, respectively. Workload 'TS-ALL' consists of combined CPU, memory, and disk emulations, while also having several dips and peaks in utilization level to investigate the impact of a high rate of change in temperature and utilization levels on the prediction models.

At the start of the experiments, the controller node transfers the load emulation python scripts and a unique VM trace file to each VM deployed on PMs. After the load emulation is triggered, OpenStack and Linux commands (e.g., vmstat, lsCPU) are used to start recording the Performance Monitoring Counters (PMCs). In addition, the controller node runs scripts that use IPMItool [40] for collecting and storing sensors' data including the fan speed and power consumption for all compute nodes. Each PM is then subject to load emulations of different categories with a 16 hour experiment duration.

For non-biased results, the evaluation is done using test sets derived from a server machine labeled 'PM4', whose data was never used during model training. Each VM on this PM has a unique VM trace file, thus having distinct utilization levels. Moreover, the test set also consists of workloads of different characteristics as previously described and shown in Table 3.

### 3.7.2 Performance Evaluation

We provide here a detailed performance evaluation and analysis of the proposed Multiple Linear Regression (MLR), Multi-Layer Perceptron Regressor (MLP), Long Short-Term Memory (LSTM), and XGBoost models on our different types of VM traffic loads. For performance evaluation, we use the mean absolute percentage error (MAPE), Root Mean Square Error (RMSE), and $R^2$ score.

MAPE is the most common measure (percentage) of error for forecast systems [29]. Similarly, the RMSE is also a measure of error and is defined by the square root of the mean of the square of all the model's prediction errors. R-squared ($R^2$) is known as the coefficient of determination and is a statistical measure representing the proportion of the variance for a dependent variable that's explained by the independent variables. As compared to MAPE, RMSE, and other KPIs that have an arbitrary range, $R^2$ can be expressed as a percentage with a fixed scale and thus, is more informative. The best-performing model will be the one that has the highest $R^2$ score and the lowest MAPE and RMSE.

$$\text{MAPE} = \left(\frac{1}{N}\sum_{i=1}^{N} |(A_i - P_i)|/A_i\right) \times 100 \tag{4}$$

$$\text{RMSE} = \sqrt{\left(\sum_{i=1}^{N}(P_i - A_i)^2\right)/N} \tag{5}$$

$$R^2 = \left(1 - \left(\sum_{i=1}^{N}(A_i - P_i)^2 / \sum_{i=1}^{N}(A_i - \bar{A})^2\right)\right) \times 100, \tag{6}$$

where $N$ is the number of data points, $A_i$ denotes the actual, and $P_i$ the predicted values at data point $i$, and $\bar{A}$ is the average of the actual values. To better interpret the performance of our proposed models, we compare them with the nonlinear analytical formula of Fan *et al.* [12] and the model of Wang *et al.* [56], i.e., an improved version of the linear formula of Fan *et al.* [12] with the addition of the inlet temperature:

$$
\begin{aligned}
P^{\text{Fan } et\ al.}(u) =&\ P_{\text{IDLE}} + (2u - u^r)(P_{\text{BUSY}} - P_{\text{IDLE}}) \\
P^{\text{Wang } et\ al.}(u) =& P_{\text{IDLE}} + u(P_{\text{BUSY}} - P_{\text{IDLE}}) + a_0 + a_1 T_{\text{INLET}} + a_2 T_{\text{INLET}}^2
\end{aligned}
$$

where *u* is the CPU utilization scaled to range (0, 1), $P(u)$ is the PM predicted power consumption at overall CPU utilization *u*, $P_{\text{IDLE}}$ and $P_{\text{BUSY}}$ are the power consumption of the PM at idle state and peak usage state, respectively, and $T_{\text{INLET}}$ is the inlet temperature. $r, a_0, a_1$, and $a_2$ are calibration parameters to fit the model and minimize the forecast error. To achieve the best fit for our PMs using the training set, we set the values of *r*, $a_0$, $a_1$, and $a_2$ as 1.5, -20, 0.005, and 0.1, respectively.

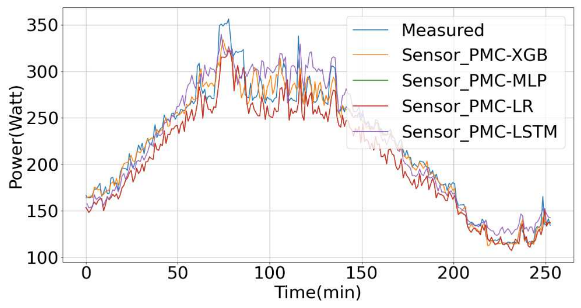
As described in our model description in Section 3.5, we have 3 categories of data-sets to derive 3 different power prediction models for each of the ML algorithms we considered:

*(i)* PMC-LSTM, PMC-LR, PMC-MLP, and PMC-XGB are PMC-based LSTM, MLR, MLP, and XGBoost models, respectively.

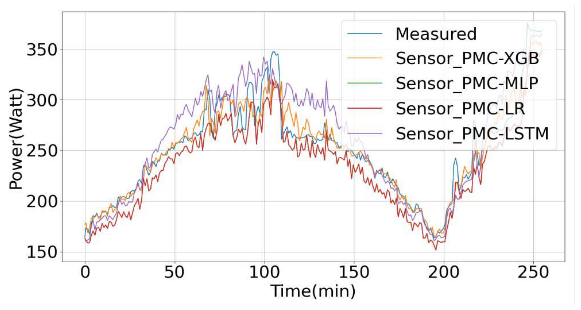
*(ii)* Sensor-LSTM, Sensor-LR, Sensor-MLP, and Sensor-XGB are the sensor-based ML models that make use of only temperature sensors and fan speed features.

*(iii)* PMC_Sensor-LSTM, PMC_Sensor-LR, PMC_Sensor-MLP and PMC_Sensor-XGB that combine the PMC and sensor features.

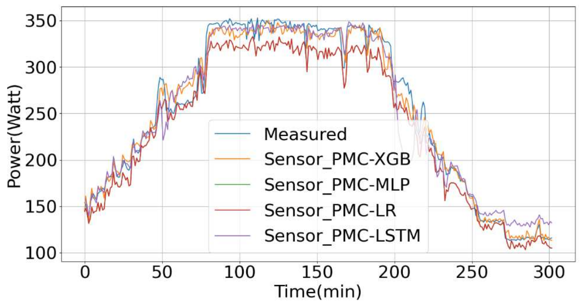
The best model performance was that of the Sensor_PMC-XGB with a RMSE of 11.9 and a $R^2$ score of 93.6%: It outperforms Wang *et al*'s best reference model's $R^2$ score by 21.1% and improves the RMSE by 48.0%. A key performance factor of our study is the ability to provide accurate results for multiple load categories. A detailed assessment for each of the workloads described in Table 3 is therefore provided in 4. The average performance over all test datasets is then presented in Table 5. Additionally, for each workload



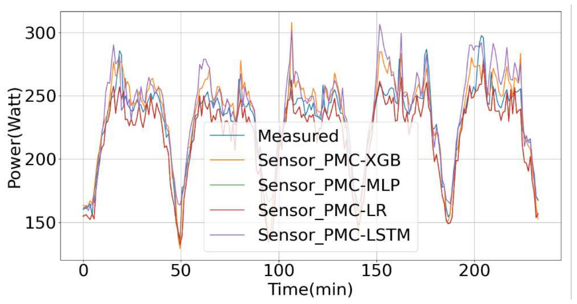
(a) TS-CPU



(b) TS-MEM



(c) TS-DISK



(d) TS-ALL

Figure 14: Performance of ML models for each Workload

| **Models** | **TS-CPU** | **TS-MEM** | **TS-DISK** | **TS-ALL** |
|---|---|---|---|---|
| Analytical Formulas-$R^2$ score | | | | |
| Fan *et al.* | 81.8 | 58.9 | 89.2 | -28.8 |
| Wang *et al.* | 87.8 | 78.7 | 94.3 | 29.4 |
| Machine Learning Models-$R^2$ score | | | | |
| PMC-LSTM | 95.1 | 86.9 | 97.5 | 63.6 |
| PMC-LR | 95.1 | 90.9 | 95.7 | 83.8 |
| PMC-MLP | 95.1 | 90.9 | 95.7 | 83.9 |
| PMC-XGB | 96.0 | 90.8 | 94.5 | 86.1 |
| Sensor-LSTM | 85.5 | 68.9 | 95.6 | 15.3 |
| Sensor-LR | 95.0 | 90.7 | 97.6 | 83.8 |
| Sensor-MLP | 95.0 | 90.7 | 97.0 | 83.9 |
| Sensor-XGB | 96.6 | 91.9 | 97.0 | 86.1 |
| Sensor_PMC-LSTM | 93.2 | 75.5 | 95.3 | 76.2 |
| Sensor_PMC-LR | 90.9 | 84.8 | 94.5 | 83.1 |
| Sensor_PMC-MLP | 90.9 | 84.8 | 94.5 | 82.9 |
| Sensor_PMC-XGB | 96.5 | 92.1 | 98.4 | 87.2 |

Table 4: Model Performance ($R^2$ score) on each workload

(TS-CPU, TS-MEM, TS, DISK, and TS-ALL), we illustrate in Fig. 14, the performance of the Sensor_PMC based models.

### 3.7.3 Observations

As previously discussed in Section 3.4, analytical and linear models fail to capture the multiple dependencies that affect power consumption. As a result, for all considered workloads, the XGBoost ML model with a feature space consisting of multiple PMCs and Sensors' data outperforms considerably the other baseline models. All evaluated models are seen to have the lowest performance on Workload 'TS-ALL'. This may be due to the several sudden peaks and dips in the utilization levels that cause a high rate of change in the temperature of internal components. Moreover, Since the evaluation is performed on a different server machine, analytical models such as Fan *et al.* and Wang *et al.* that were calibrated based on the training set, have extremely poor performance due to the difference in the behavior of the PMs. Fig. 15 illustrates the weights of the best prediction model's features (refer to Table 2 and Fig. 12 for feature description). The weights of all sensor components and the significantly high weight of the internal Fan's speed prove our arguments presented in this paper. The sensor-related features were in fact more important than

| **Models** | **MAPE (%)** | **RMSE** | $R^2$ (%) |
|---|---|---|---|
| Analytical Formulas | | | |
| Fan *et al.* | 10.5 | 30.5 | 50.3 |
| Wang *et al.* | 7.7 | 22.9 | 72.5 |
| Machine Learning Models | | | |
| PMC-LSTM | 4.95 | 16.0 | 85.7 |
| PMC-LR | 5.03 | 14.7 | 92.0 |
| PMC-MLP | 5.02 | 14.7 | 91.4 |
| PMC-XGB | 3.86 | 14.7 | 91.9 |
| Sensor-LSTM | 8.48 | 24.5 | 66.3 |
| Sensor-LR | 4.76 | 14.1 | 91.6 |
| Sensor-MLP | 4.76 | 14.1 | 91.7 |
| Sensor-XGB | 3.51 | 12.6 | 92.9 |
| Sensor_PMC-LSTM | 5.85 | 18.5 | 85.1 |
| Sensor_PMC-LR | 6.21 | 17.7 | 88.3 |
| Sensor_PMC-MLP | 6.23 | 17.7 | 88.2 |
| Sensor_PMC-XGB | 3.34 | 11.9 | 93.6 |

Table 5: Average Model Performance (All Workloads)

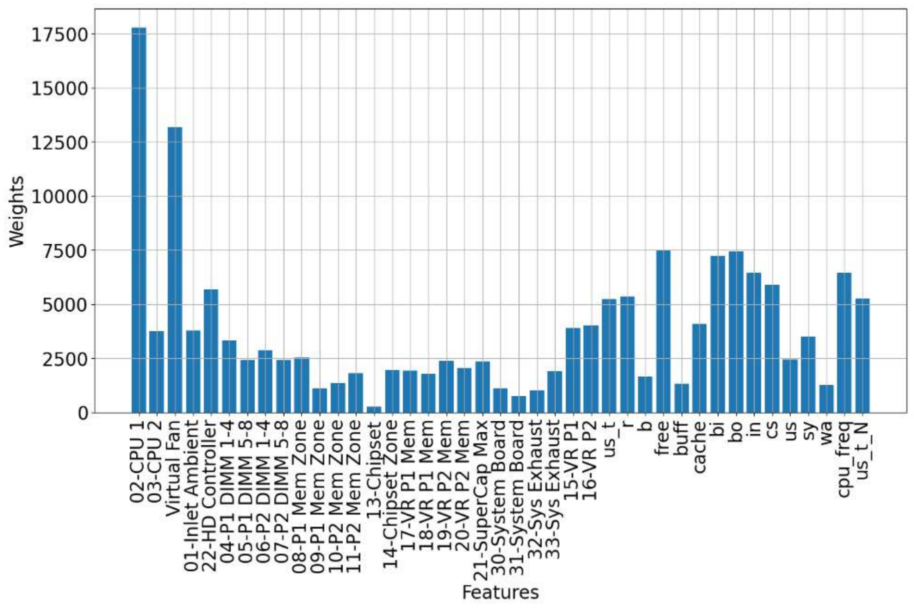


Figure 15: Feature weights of best performing model

the CPU utilization, which is considered by most existing works as the main dependency of power models. Additionally, as discussed in Section 3.4, even if the CPU is the largest power-consuming component, other parts such as memory or disk-related components do have a significant contribution to the power consumption. Indeed, this is proven by the considerably high feature importance of components such as Buffer, Cache, Interrupts, and Context Switching.

## 3.8 Conclusion

We proposed a highly accurate prediction model for data center power consumption based on machine learning models with a performance monitoring counter (PMC) feature space and detailed temperature values. The best XGBoost model obtains a $R^2$ score of 93.6% on multiple datasets with different workload characteristics. Thus, our proposed Machine Learning models can be used for virtual machine scheduling or placement algorithms aimed at minimizing energy (or maximizing energy efficiency) by making energy-aware decisions based on the predicted power consumption of a data center's Physical Machines.

## Acknowledgment

Work of the first author was supported by a MITACS - Ericsson internship.

# Chapter 4

# Virtual Machine Placement using Embedded Sensors and Machine Learning for Energy Minimization

This chapter has been submitted as a Journal paper titled "Virtual Machine Placement for Energy Minimization using Embedded Sensors and Machine Learning" written by N. Moocheet, B. Jaumard, and P. Thibault.

## 4.1 Abstract

Cloud Data Centers (DCs) consume vast amounts of energy and contribute significantly to environmental concerns. Moreover, with the advent of 5G and B5G future networks which are increasingly software-oriented and becoming heavily dependent on cloud computing, there is an urgent need for optimizing their energy usage. Thus, in this study, we present a Virtual Machine placement algorithm that minimizes the energy consumption of a cluster of server machines. Our solution is embodied through the utilization of embedded sensors inside server physical machines, which enables the introduction of novel features for

sensitive thermal awareness and proactive hotspot avoidance. Leveraging this significantly enhanced feature space, we implement data-driven predictive Machine Learning models together with a heuristic placement algorithm (CPP), enabling proactive VM placements that are both energy-aware and thermal-aware. Indeed, experiments conducted on a cluster of real server physical machines demonstrate high performance by both the ML models and placement algorithm (CPP). Relative to the best baseline algorithm, our solution reduced energy consumption and temperature by 7% and 2%, respectively, while avoiding hotspots and maintaining effective load distribution reducing PMs over-loading by 28%.

## 4.2 Introduction

The emergence of cloud computing has brought a multitude of benefits, including increased flexibility, enhanced security, and the opportunity for consumers to reduce their infrastructure investment. Reliance on cloud computing continues to grow at an unprecedented rate as the technology continues to evolve, especially with the advent of 5G/B5G becoming more software-oriented, coupled with the rapid proliferation of low-latency technologies such as the Internet of Things (IoT), smart cities and self-driving cars. As a result, there is an overall increase in energy consumed by Cloud Data Centers (DCs).

According to the International Energy Agency (IEA), Data centers consume 1-1.5% of global electricity and are responsible for 1% of all energy-related greenhouse gas emissions. The rising costs of energy and the growing focus on environmental sustainability have heightened the urgency for the DC industry to improve its operational efficiency. However, this remains a challenge due to the complex nature of modern large-scale DCs consisting of multiple mechanical and electrical equipment including cooling, computing, and control systems. Indeed, A data center consists of several rack-mounted physical machines (PMs), networking equipment, monitoring devices, a cooling system to maintain adequate equipment operational temperature, and many other facility-related subsystems.

Some of those rack-mounted machines have the potential to consume up to 1,000 watts of power each, reaching peak temperatures as high as 100 degrees Celsius [24]. Therefore, computing and cooling subsystems are the two primary sources of a DC's power consumption and the key areas of focus for research on energy efficiency or minimization. Improving the cooling subsystem's energy efficiency involves the consideration of the intricate architecture and components of the DC, such as coolers, pumps, and heat sensors to optimize the cold air circulation and equipment configuration control. A notable example is Jim & Guo [15], a Machine Learning solution by Google Cloud Service which aims to increase their Power Usage Efficiency (PUE) by considering indoor cooling system components, as well as, outdoor factors such as wind speed, direction, and air humidity. Similarly, optimizing energy consumption in the computing subsystem holds significant importance. It not only accounts for the highest energy consumption but is also a large contributor to the energy and thermal inefficiency of a DC. The computing subsystem consists of core components and infrastructure, most specifically, Physical Machines (PMs) or Blades that are responsible for hosting, processing, or executing computational tasks and workloads mostly in the form of Virtual Machines (VMs) or Containers. The orchestration of these VMs among the PMs of a Data Center is crucial for energy management. Indeed, the key concepts that are being applied to achieve DC energy management are energy-aware virtual machine (VM) or Container management: scheduling, placement, and migration. The main idea consists of strategically orchestrating VMs among the physical machines (PM) of a Data Center to achieve optimal energy consumption.

The objective of this work is to develop and implement a Virtual Machine (VM) placement algorithm that achieves energy minimization and thermal-aware orchestration of VMs on a cluster of Physical Machines/Blades. More specifically, we aim to determine the best PM to deploy each VM so as to minimize power consumption and peak temperatures. Existing research works have extensively explored dynamic consolidation strategies as a popular solution. These strategies aim to minimize the number of active Physical Machines by consolidating multiple VMs into fewer PMs. By effectively managing resources, this approach significantly reduces power consumption in data centers by shutting off idle machines. However, consolidation relies heavily on accurate traffic prediction for adequate functioning. In the event of sudden increases in traffic volumes, the under-provisioning of resources will result in the over-utilization of PMs, leading to high response times, delays, and overheating of physical components. Consequently, implementing most consolidation techniques in real-world data centers, which often handle large volumes of heterogeneous workloads and unpredictable traffic patterns, may not be feasible or practical. In fact, most Data Centers today mostly employ over-provisioning of resources to guarantee Quality of Service (QoS). In general, the DC operator selects the VM orchestration strategy of a Data Center based on their traffic profile, volume, and the contractual agreement or commitment of service level with their clients. To ensure no violation of Service Level Agreements (SLA) occurs, Load Balancing (LB) strategies specific to the traffic profile are preferred to maintain appropriate levels of PM utilization, handle unpredicted traffic peaks, and reduce overheating of machines. In fact, the default placement policies employed by main Cloud Computing Virtualization platforms such as Kubernetes, vSphere, and OpenStack are all based on efficient resource allocation through filtering and scoring based on affinity specifications, data locality, and Load Balancing [30], [16], [38]. Thus, the challenge is to design a VM placement algorithm that maintains the key aspects of LB while also reducing energy consumption. Our approach is based on the fact that Physical Machines including those of similar hardware, may have different power consumption even at similar load and resource utilization due to differences in their heat profile. Unlike most thermal-aware works that focus on reducing peak rack heat generation to lower the DC's cooling energy cost, our solution takes a different approach aiming to instead, minimize the overall energy consumption of the compute/server subsystem itself without the need for consolidation and PM power offs. We achieve this by considering the high-accuracy heat profile of each individual PM through their multiple embedded heat sensors. Figure 16 illustrates a typical heatmap that can be generated using various temperature sensor measurements for our physical machines. Therein, each dot represents the temperature of one particular sensor.

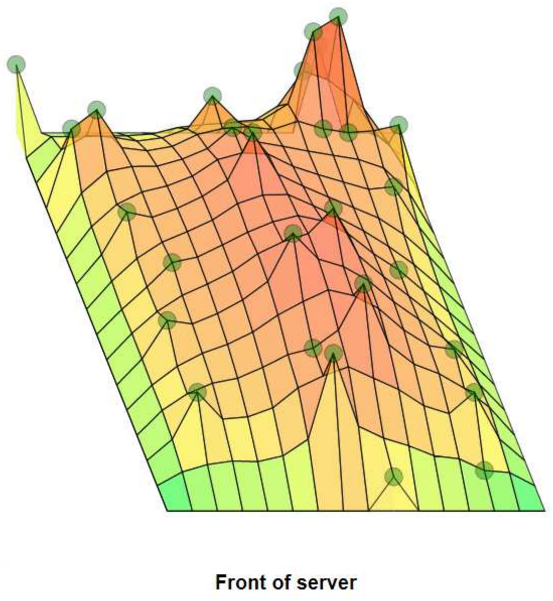


Figure 16: Heat Map of a HPE ProLiant BL460C Gen8 PM

Taking advantage of the enhanced feature space, we first introduce the concept of PMs' criticality by taking into account the critical temperature thresholds of each internal component. We then implement predictive Machine Learning (ML) models (XGBoost) for power and criticality predictions. Those models are then used by a greedy heuristic algorithm for proactive energy-aware and thermal-aware VM placement. Our contributions in this paper can be summarized as follows:

1. Take advantage of the embedded heat sensors and Internal Fan Speed Meters to get detailed heat profiles of Physical Machines.

2. Consider the critical temperature threshold of the internal sensors to introduce the concept of a PM's criticality.
3. Implement highly accurate Power and Criticality prediction Machine Learning models.
4. Provide a weighted random sampling algorithm that improves scalability.
5. Implement a Greedy Heuristic Placement algorithm for a proactive energy-efficient and thermal-efficient VM placement.

Our proposed solution is evaluated on a cluster of real Physical Machines (HP ProLiant BL460c G8) in a private Data Center and is shown to have a significant reduction in overall energy consumption and temperature. Our paper is structured as follows: Section 4.3 provides an overview of existing literature related to Virtual Machine orchestration. Section 4.4 exhibits the need for thermal-aware solutions and introduces a new feature set of real-time sensor data. The implementation of Machine Learning predictive models for power and criticality prediction is discussed in Section 4.5. Section 4.6 then provides our proposed Virtual machine placement algorithm. Finally, all models and placement algorithm evaluation results are presented in Section 4.7, and Conclusions are drawn in Section 4.8.

## 4.3 Literature Review

### 4.3.1 Consolidation

Data centers are generally designed to handle peak traffic for preventing SLA violations, overload, or hotspot conditions. As a result, resources are very often over-provisioned, thus leading to energy inefficiency since idle servers can consume up to 70% of their maximum energy [6]. Therefore, VM consolidation which consists of minimizing the number of active PMs and switching off those that are underutilized/idle is a widely used technique for increasing energy efficiency. Most authors, e.g., [4], [5] and [36], consider VM placement as an energy-aware bin packing problem and implement best-fit heuristics to deploy VMs on a minimum number of physical machines (PMs). Beloglazov *et al.* [5] also proposed threshold-based overload and underload detection heuristics, as well as a VM selection algorithm for a time-driven dynamic consolidation. Similarly, [18] proposes a bin-packing VM placement and an Integer Linear Programming (ILP) algorithm to maximize the number of idle servers while satisfying power limits and resource availability constraints. Using these types of heuristics and reactive consolidation policies often results in a high number of hotspots due to sudden peaks in the DC's dynamic traffic. The need for immediate hotspot management drives a large number of virtual machine migrations that come at a considerable cost in energy and resources. Additionally, delays in reactivating inactive machines also lead to potential SLA violations.

Thus, Haghshenas and Mohammadi [21] and Farahnakian *et al.* [14] propose proactive approaches to VM consolidation that take into account the use of both current and future resources. They use a regression-based model to approximate the future resource utilization of VMs and PMs based on their historical data. Ranjbari and Torkestani [43] explore another proactive solution that uses a learning automata-based overload detection that enhances the VM consolidation by predicting the CPU usage of PMs. However, these solutions are only guaranteed to work on the specific traffic pattern that they were trained with, thus, not appropriate for real data centers with large variations in traffic patterns and characteristics. Moreover, although these solutions can be effective in certain scenarios, they overlook the impact of heat generation.

### 4.3.2 Thermal-Aware Approaches

Consolidating virtual machines (VMs) onto a smaller number of active physical machines (PMs) often increase peak temperatures, which in turn requires a colder air supply at the inlets to maintain standard operational temperatures. This trade-off between reduced compute energy consumption and increased cooling energy usage can lead to negative outcomes, particularly depending on the type of cooling system utilized in the data center. In order to minimize overall energy consumption in data centers, some researchers have explored thermal-aware or holistic solutions. For instance, Guo *et al.* [20] propose a mixed integer linear programming (MILP) and a greedy heuristic for a temperature-aware virtual data center embedding scheme. Their solutions focus on mapping VMs with high CPU demand to colder racks in order to reduce the required inlet temperature. Similarly, Ilager *et al.* [24] propose a greedy algorithm that allocates VMs on PMs with the lowest predicted CPU and inlet temperature. In another approach, Gill *et al.* [19] implement a deep learning-based framework to predict the thermal impact of VMs and allocate them to PMs based on their temperature profiles. Li *et al.* [31] proposes a holistic approach called GRANITE that places VMs based on their predicted total power consumption and performs threshold-based migrations at regular intervals to respect SLA and utilization level constraints. However, these temperature-aware solutions typically only consider the temperature at the rack level or only a few components such as the CPU and Inlet temperature.

### 4.3.3 Reinforcement Learning

Reinforcement learning (RL) has emerged as a promising approach for addressing the challenges of VM placement and consolidation in cloud environments. Unlike traditional heuristic or optimization-based solutions, RL-based solutions are autonomous and adaptive to changing conditions in real-time. RL enables an agent to learn an efficient policy for allocating VMs through repeated interactions with its environment and feedback in the form of rewards or penalties for its actions. Farahnakian *et al.* [13] employed Q-learning, a popular RL technique, to implement dynamic VM consolidation with an agent which made informed decisions on the power mode of physical machines (PMs) based on resource utilization information. Similarly, Qin *et al.* [42] uses Q-learning in the context of VM placement to optimize two objectives, that are the minimization of resource wastage and energy consumption. While simplistic state spaces consisting only of resource information like CPU utilization may be sufficient for simpler environments, they are inadequate for complex ones like data centers where there exist several factors influencing the behavior of its components.

Thus, Qin *et al.* [55] propose a VM scheduling framework that uses a Deep Neural Network (DNN) Quality of Service (QoS) feature extraction model based on a denoising autoencoder which enables more robust feature information to be used as state information for their Q-Learning RL agent. However, a more detailed state space also implies a higher number of state-action pairs, which considerably enlarges the search space and convergence time, thus, limiting its scalability in complex environments. A common solution for this issue is Deep Q-Network (DQN), a variant of Q-learning that uses a deep neural network to approximate the Q-values of state-action pairs instead of storing them in a Q-table. For instance, Caviglione *et al.* [9] proposes a DQN-based approach for dynamic VM placement. In their approach, the action space consisted of six placement heuristics including three novel ones, the aim being for the agent to select the best placement heuristic appropriate for specific situations and goals. Another possible way of addressing the issue of slow convergence time is the use of domain knowledge or expert advice. For example, Shaw *et al.* [48] proposes the use of an advanced reward shaping technique called Potential Based Reward Shaping (PBRS) to encourage more optimal decision-making in the earlier stages of the Q-learning exploration phase. All these RL-based solutions similar to most other existing works were implemented and evaluated on Cloud Simulators. Indeed, it is quite challenging to train RL models on a real data center since it requires performing actual Virtual Machines deployments and load emulations for large time periods, thus very costly in terms of resources and energy. In addition, there is also a lack of flexibility for scale-up in cases where any addition of new PMs would require the model to be re-trained due to the change in the model's state/observation space.

### 4.3.4 Concluding Remarks

Energy-efficient resource management is a well-known necessity, and several studies have explored VM orchestration solutions with energy awareness. However, these solutions have predominantly relied on consolidation-based approaches, which are often reactive, and employ analytical formulas and threshold-based methods. Consequently, they are prone to failure in complex and dynamic environments. Additionally, very few studies consider thermal-aware solutions and they focus solely on the temperature at the rack level or specific components such as CPUs or inlets, disregarding the impact of heat generation at other internal components of the PMs. Indeed, this work addresses these shortcomings by considering all internal components through embedded sensors and investigating proactive, data-driven models for energy-aware and thermal-aware VM placement.

## 4.4 Need For Thermal-aware Techniques

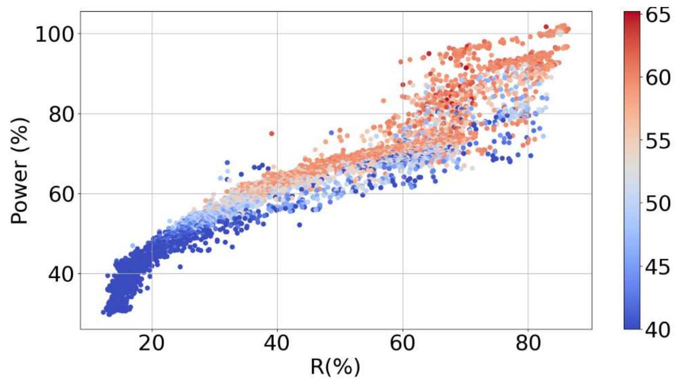
Physical machines (PMs) in a large-scale data center (DC) are often densely arranged in rack, cabinet, or enclosure structures with an optimized modular design to minimize the use of physical space. These architectures are likely to cause server machines to overheat due to the dense arrangement of electronic devices, consequently increasing the energy cost of the cooling subsystem and further decreasing system reliability due to failures under high thermal conditions. Moreover, it also directly influences the energy consumption of physical machines due to the increased internal fan speeds for dissipating internal heat. High temperatures may also cause increased leakage current of electronic devices' silicon which calls for new semiconductor technologies that use much less switching energy than current CMOS transistors [8]. Thus, thermal management is essential for the overall energy-efficient operation of a data center. Indeed, every degree increase in the peak temperature of a data center incurs significant operational costs, amounting to millions of dollars [49].

### 4.4.1 Influence of Temperature & Need for Enhanced Feature Space

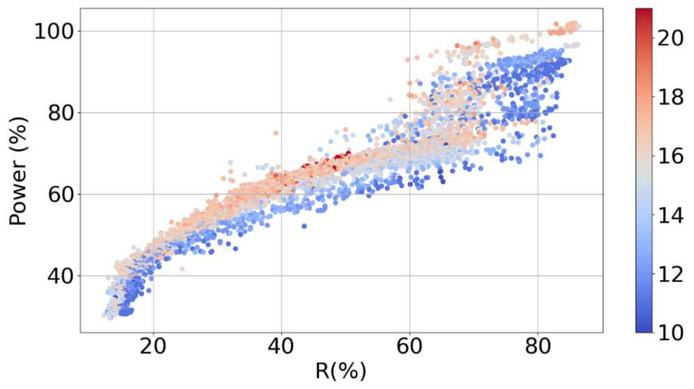
The importance and complexity of thermal awareness are often underestimated in studies, which consider only a very limited number of features as discussed in Section 4.3.2. In the sequel, we demonstrate the need for an increased number of features, having in mind, e.g., the characteristics of the different classes of 5G traffic (CPU greedy vs. RAM intensive traffic). Stress testing experiments were conducted on a set of PMs in a private data center. Results are summarized in Figures 17a and 17b, which illustrate the relation between the resource utilization level which is a weighted sum of the CPU, memory, and disk utilization, and the measured power consumption for a set of PMs. The resource utilization of a PM is given as;

$$R(\%) = w^{\text{CPU}} R_{CPU} + w^{\text{MEM}} R_{MEM} + w^{\text{DISK}} R_{DISK} \tag{7}$$

where $R_{CPU}$, $R_{MEM}$,and $R_{DISK}$ are the percentage CPU, Memory, and Disk Utilization levels, respectively. The weight variables are set as $w^{cpu} = 0.8$ and $w^{mem} = w^{disk} = 0.1$ since the CPU is the highest contributor to a PM's power consumption. Several instances reveal large differences in power consumption despite similar levels of resource utilization. For example, at $R = 80\%$, there is a difference of up to 30% between the lowest power-consuming sample and the highest. While most of those inefficiencies can be attributed to the high temperature of the CPUs or Inlets (as indicated in the color maps in Fig.17a and Fig. 17b), several samples, particularly at $R > 60\%$ exhibit relatively higher power consumption, even at low CPU and Inlet temperature. Therefore, this validates the need for



(a) CPU Temperature ($^oC$)

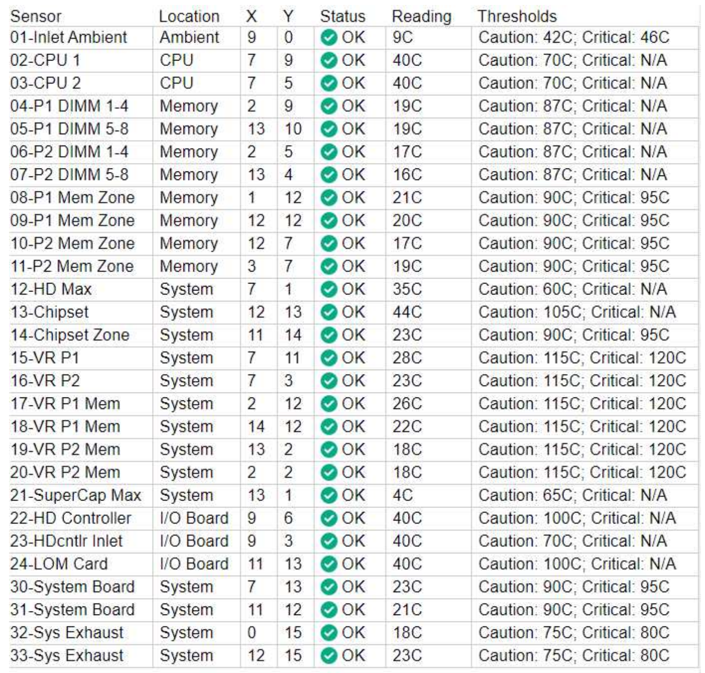


(b) Inlet Ambient Temperature ($^oC$)

Figure 17: Resource Utilization v/s Measured Power and impact of Temperature

an enhanced feature space with more features such as the temperature of other components (e.g. memory, power supply units, etc..) and additional resource utilization data.

While existing techniques consider the temperature at the PM or Server level (e.g., [24], [19], [31]), rack level [20], and DC's room level [20], in this study, we consider the micro-level consisting of the internal components of each individual Physical Machine (PM). Indeed, in modern Server Physical Machines, various sensors are installed, including heat sensors, fan speed meters, and power meters. These heat sensors offer real-time measurements of multiple internal components, enabling us to gain valuable insights into the system's thermal behavior and accurately detect hotspots. Fig. 18 displays a snapshot of the list of sensors along with their location, related components, and critical thresholds.



| Sensor | Location | X | Y | Status | Reading | Thresholds |
|---|---|---|---|---|---|---|
| 01-Inlet Ambient | Ambient | 9 | 0 | ✓ OK | 9C | Caution: 42C; Critical: 46C |
| 02-CPU 1 | CPU | 7 | 9 | ✓ OK | 40C | Caution: 70C; Critical: N/A |
| 03-CPU 2 | CPU | 7 | 5 | ✓ OK | 40C | Caution: 70C; Critical: N/A |
| 04-P1 DIMM 1-4 | Memory | 2 | 9 | ✓ OK | 19C | Caution: 87C; Critical: N/A |
| 05-P1 DIMM 5-8 | Memory | 13 | 10 | ✓ OK | 19C | Caution: 87C; Critical: N/A |
| 06-P2 DIMM 1-4 | Memory | 2 | 5 | ✓ OK | 17C | Caution: 87C; Critical: N/A |
| 07-P2 DIMM 5-8 | Memory | 13 | 4 | ✓ OK | 16C | Caution: 87C; Critical: N/A |
| 08-P1 Mem Zone | Memory | 1 | 12 | ✓ OK | 21C | Caution: 90C; Critical: 95C |
| 09-P1 Mem Zone | Memory | 12 | 12 | ✓ OK | 20C | Caution: 90C; Critical: 95C |
| 10-P2 Mem Zone | Memory | 12 | 7 | ✓ OK | 17C | Caution: 90C; Critical: 95C |
| 11-P2 Mem Zone | Memory | 3 | 7 | ✓ OK | 19C | Caution: 90C; Critical: 95C |
| 12-HD Max | System | 7 | 1 | ✓ OK | 35C | Caution: 60C; Critical: N/A |
| 13-Chipset | System | 12 | 13 | ✓ OK | 44C | Caution: 105C; Critical: N/A |
| 14-Chipset Zone | System | 11 | 14 | ✓ OK | 23C | Caution: 90C; Critical: 95C |
| 15-VR P1 | System | 7 | 11 | ✓ OK | 28C | Caution: 115C; Critical: 120C |
| 16-VR P2 | System | 7 | 3 | ✓ OK | 23C | Caution: 115C; Critical: 120C |
| 17-VR P1 Mem | System | 2 | 12 | ✓ OK | 26C | Caution: 115C; Critical: 120C |
| 18-VR P1 Mem | System | 14 | 12 | ✓ OK | 22C | Caution: 115C; Critical: 120C |
| 19-VR P2 Mem | System | 13 | 2 | ✓ OK | 18C | Caution: 115C; Critical: 120C |
| 20-VR P2 Mem | System | 2 | 2 | ✓ OK | 18C | Caution: 115C; Critical: 120C |
| 21-SuperCap Max | System | 13 | 1 | ✓ OK | 4C | Caution: 65C; Critical: N/A |
| 22-HD Controller | I/O Board | 9 | 6 | ✓ OK | 40C | Caution: 100C; Critical: N/A |
| 23-HDcntlr Inlet | I/O Board | 9 | 3 | ✓ OK | 40C | Caution: 70C; Critical: N/A |
| 24-LOM Card | I/O Board | 11 | 13 | ✓ OK | 40C | Caution: 100C; Critical: N/A |
| 30-System Board | System | 7 | 13 | ✓ OK | 23C | Caution: 90C; Critical: 95C |
| 31-System Board | System | 11 | 12 | ✓ OK | 21C | Caution: 90C; Critical: 95C |
| 32-Sys Exhaust | System | 0 | 15 | ✓ OK | 18C | Caution: 75C; Critical: 80C |
| 33-Sys Exhaust | System | 12 | 15 | ✓ OK | 23C | Caution: 75C; Critical: 80C |

Figure 18: Snapshot of a real-time sensor reading & description

### 4.4.2 Critical Temperature

The critical threshold of each component is another key factor that has been ignored by existing thermal-aware solutions. As shown in Fig. 18, each component has a caution or critical temperature threshold that may differ significantly from each other. For instance, the critical temperature of the considered PM's (HPE ProLiant BL460C Gen8 PM ) CPUs is 70 °C as compared to the Inlet Ambient's which is 40 °C. Hence, relying solely on a single temperature threshold to identify overheating or hotspot conditions, while considering temperature values alone, fails to accurately assess the true criticality of the PMs. In our study, we address this limitation by incorporating critical thresholds into our feature space. We achieve this by scaling the temperature of each component based on its respective critical threshold. This scaling process yields a metric that represents the component's temperature as a percentage of its critical temperature, as described in (8).

$$T_c = \frac{T_{measured}}{T_{critical}} \times 100 \tag{8}$$

Additionally, similar to the temperature we also scale the power consumption of each PM as a percentage of its maximum power consumption (power cap) since in a typical data center, PMs have different power caps due to differences in hardware, capacity, and energy configurations.

## 4.5 A Proactive Energy & Thermal-aware Approach

We now present the two key components to achieve proactive energy-aware and thermal-aware placement of virtual machines. First, we propose, in Section 4.5.1, a machine learning power model that enables informed decisions based on energy/power for virtual machine placement. In view of its limitations and a need for thermal awareness, we next introduce the concept of PM's criticality in Section 4.5.2, The latter plays an important role in proactive hotspot avoidance and facilitates energy-efficient placement.

### 4.5.1 Power Model

To implement proactive energy-aware VM placement strategies, it is essential to have an accurate model for power consumption prediction. However, existing studies (e.g., [19], [20], [31]), mostly employ analytical power models which fail to capture multiple non-linear inter-dependencies that affect the power consumption of DC's PMs. Indeed, as discussed in Section 4.4, it is imperative to consider temperature for explaining or modeling the power consumption behavior of physical machines (PMs). The heterogeneous workload of large data centers necessitates having a more detailed characterization of the workload in terms of the utilization of computational resources (e.g., i/o intensive video streaming), rather than using models depending solely on CPU utilization.

Therefore, we make use of our previous work [37] which tackles this issue by introducing an enhanced feature space of the real-time sensor measurements (Fig.18) and Performance Monitoring Counters(PMCs) for highly accurate power predictions. In the latter, the ML models predict the power consumption of a PM for time $t$ using the CPU utilization of the PM at $t$, PMCs at $t-1$, and all components' temperatures measured at time $t-1$ since PMCs & temperature cannot be extrapolated or estimated as easily as the CPU utilization. Extreme Gradient Boosting (XGBoost) was found to be the best model after evaluation and comparison to other models such as Long Short-Term Memory (LSTM), Multi-Layer Perceptron Network (MLP)), and analytical formulas of Fan *et al.* [12] and Wang *et al.* [56].

In this study, we improve the XGBoost model implemented in [37] by re-training using a significantly enlarged dataset (10,990 data points) derived from deploying up to 180 VMs on 6 PMs. The detailed experiment setup is later described in Section 4.7.1 and results are provided in Section 4.7.3.

**Observation**

The XGBoost model had an average $R^2$ score of 96.5$\%$, RMSE of 11.7 W, and MAPE of 2.8$\%$, outperforming the best baseline model's $R^2$ score by 6.1$\%$ and improves the RMSE by 40.0$\%$. See Section 4.7 for details on the models' performance evaluation and results. While on average the XGBoost power model provides accurate predictions, the high fluctuations in power at high temperature and resource utilization levels as shown in Fig. 17a and Fig. 17b, cause large prediction errors. Indeed, Fig. 19 shows the increase in average prediction error of the XGBoost power model when reaching high-temperature levels (e.g. at CPU and Power-Supply). These PM states of high power, temperature, and resource utilization are the main contributors to energy inefficiency and hence, where proactive thermal and energy-aware VM placement plays a key role. Relying solely on power predictions that are susceptible to errors at such states may have a negative impact in relation to hotspot avoidance. Therefore, this issue is addressed in the next section by introducing a second feature to be considered during placement in addition to power.

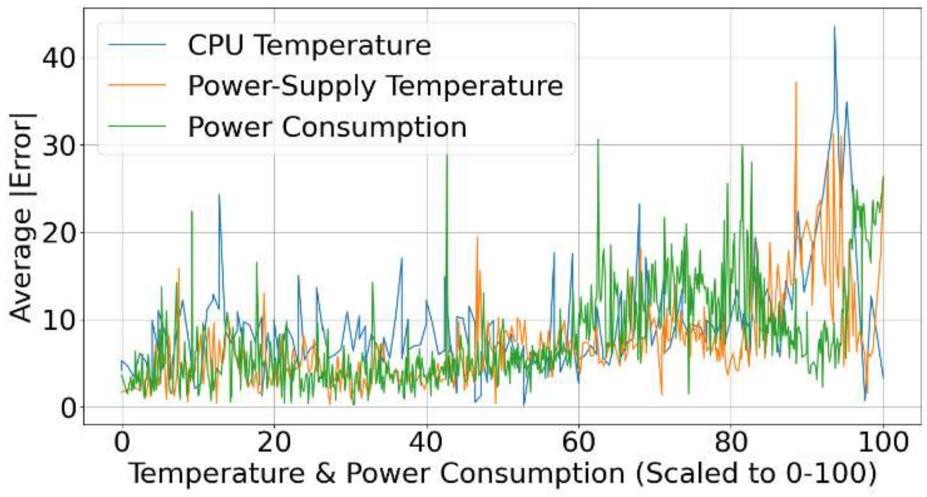


Figure 19: Increase in average power prediction error at high temperature

### 4.5.2 Criticality of a Physical Machine

Following the findings of the previous section, a PM is significantly influenced by its heat profile and highly susceptible to high power fluctuations when reaching certain high-temperature states. Moreover, after reaching certain peak power states or power caps, power predictions become obsolete since any added load would result in a nearly zero power increase. Indeed, the aim of an efficient VM placement would be to prevent PMs from such inefficient states. Hence, we present in this section a new feature that is the criticality of a PM.

The common method of identifying those states often labeled as hotspots is the use of pre-defined thresholds, for instance, resource utilization level thresholds, maximum power, and CPU temperature thresholds. Our approach consists of taking advantage of the enhanced feature space ($T_c$ of all PM's components) to accurately identify such inefficient/hotspot states. We introduce the concept of a PM's criticality which is a measure of how close the PM is to reaching that hotspot status. After stressing a PM, several samples with relatively high power and temperature can be identified (e.g., in Fig. 17) and labeled as their worst reachable states. By collecting and averaging the state vectors of these collected samples, a state vector $\vec{S}_{\text{HOTSPOT}}$ is generated that represents a hotspot status for that specific PM hardware model. The state vector of a PM consists of all its components' temperature scaled over their critical temperature ($T_c$). The criticality of a PM is then calculated by employing a similarity measure such as the Cosine Similarity (similarity between two vectors of an inner product space) between the PM's state vector and the hardware type's respective pre-calculated $\vec{S}_{hotspot}$. The Criticality of a PM with measured state $\vec{S}_{\text{PM}}$ is defined as:

$$\text{C}_{\text{PM}} = \frac{\vec{S}_{\text{PM}} \cdot \vec{S}_{\text{HOTSPOT}}}{||\vec{S}_{\text{PM}}|| \times ||\vec{S}_{\text{HOTSPOT}}||}. \tag{9}$$

**Criticality Prediction**

For a proactive approach, similar to the power consumption prediction, a Machine Learning model is implemented for predicting the Criticality of PMs. More specifically, a model that predicts how close a PM will be to the hotspot status after deployment of a particular VM. The criticality of a PM at a time/interval $t$ is predicted using the estimated CPU utilization of the PM at $t$ and measured heat profile (temperature ($T_c$) of all components), Memory, and Disk Utilization at time $t-1$. The data collection, pre-processing, and the investigated ML models are similar as in Section 4.5.1. XGBoost was once again the selected model with an $R^2$ score of 90.9%, RMSE of 0.05, and MAPE of 7.3%. Model evaluation and results are presented in Section 4.7.

## 4.6 Virtual Machine Placement

We now propose a dynamic VM placement algorithm that proactively minimizes the energy consumption and temperature of PMs in a data center using the Machine Learning models presented in Section 4.5.1 and 4.5.2.

### 4.6.1 Problem formulation

For a timestep $t$, where $n$ VMs are in the scheduling queue ready for deployment in a DC with $m$ active PMs, the objective is to place the $n$ VMs among the $m$ PMs such as the overall increase in power consumption and criticality is minimized at $t+1$. Using the ML models to predict the power and criticality of each PM for time-step $t+1$, an energy-aware and thermal-aware decision can be made based on the expected power increase and criticality. However, the importance of these 2 factors may vary based on workload levels or the DC operator's priorities. For example, when a PM reaches a certain level of resource utilization (close to 100%), its power consumption reaches the power cap causing a very small or no increase in power for additional load. In such states, power-based decisions have a very low impact or a negative outcome. In comparison, a temperature-based measure such as criticality has a higher significance since these states have higher heat generation. Therefore, we define $PC_{t+1}$, a weighted sum of the predicted increase in power consumption and criticality, with weights $w^{\text{C}}$ and $w^{\text{P}}$. It is defined as follows:

$$PC_{\text{PM}}^{t+1} = w^{\text{P}} \left( \frac{P_{\text{PM}}^{t+1}(u_{t+1}) - P_{\text{PM}}^{t}}{P_{\text{PM}}^{\max} - P_{\text{PM}}^{\text{idle}}} \right) + w^{\text{C}} C_{\text{PM}}^{t+1}(u_{t+1}) \tag{10}$$

where $P_{\text{PM}}^{t}$ is the measured power at $t$, $P_{\text{PM}}^{\max}$ and $P_{\text{PM}}^{\text{idle}}$ are the maximum and idle PM power, used for scaling the increase in dynamic power on a scale of 0 to 1, $P_{\text{PM}}^{t+1}$ and $C_{\text{PM}}^{t+1}$ are the predicted power consumption and criticality, respectively, for an estimated resource utilization $u_{t+1}$ at $t+1$. As previously explained, $w^{\text{C}}$ and $w^{\text{P}}$ are weight parameters for balancing the importance of the power and criticality. In this study, we use weights $w^{\text{C}} = w^{\text{P}} = 0.5$ at Criticality ($C_{\text{PM}}^{t}$) $< 0.99$, otherwise $w^{\text{P}} = 0$ and $w^{\text{C}} = 1$ to maximize the importance of the criticality upon reaching peak power.

We next develop a mathematical model for optimal VM placement aiming to minimize energy consumption and criticality. It requires a single set of variables: $x_{\text{PMVM}}$ indicates the assignment of a particular VM to a specific PM, at a given time-step. Below $t$ indices are omitted in order to alleviate the notations.

$$\min \qquad \sum_{\text{PM} \in V^{\text{PM}}} \sum_{\text{VM} \in V^{\text{VM}}} x_{\text{PM,VM}} PC_{\text{PM}} \tag{11}$$

subject to:

$$\sum_{\text{PM} \in V^{\text{PM}}} x_{\text{PM,VM}} = 1 \qquad \text{VM} \in V^{\text{VM}} \tag{12}$$

$$\sum_{\text{VM} \in V^{\text{VM}}} x_{\text{PM,VM}} \leq \theta_{\text{VM}}^{max} \quad \text{PM} \in V^{\text{PM}} \tag{13}$$

$$R_{\text{PM}}^{\text{CPU}} \leq \theta_{\text{CPU}} \qquad \text{PM} \in V^{\text{PM}} \tag{14}$$

$$R_{\text{PM}}^{\text{RAM}} \leq \theta_{\text{RAM}} \qquad \text{PM} \in V^{\text{PM}} \tag{15}$$

$$R_{\text{PM}}^{\text{DISK}} \leq \theta_{\text{DISK}} \qquad \text{PM} \in V^{\text{PM}} \tag{16}$$

$$T_{\text{PM},s} \leq T_s^{MAX} \qquad s \in S^{\text{TEMP}} \tag{17}$$

$$x_{\text{PM,VM}} \in \{0,1\} \qquad \text{VM} \in V^{\text{VM}}, \text{PM} \in V^{\text{PM}}. \tag{18}$$

Constraints (12) enforce the condition that each VM can be assigned to only one PM. Constraints (13) impose a VM threshold $\theta_{\text{VM}}^{\max}$ to prevent overloading of PMs (if PMs have different characteristics, the threshold may need to depend on the PM). Similarly, Constraints (14)-(16) sets PM thresholds for the utilization level of the compute resources, here, CPU, Memory, and Disk, denoted by $R_{\text{PM}}^{\text{CPU}}$, $R_{\text{PM}}^{\text{RAM}}$, and $R_{\text{PM}}^{\text{DISK}}$, respectively. In addition, Constraints (17) make sure that the temperature ($T_c$) at all components $s$ of a PM must be maintained below a threshold $T_s^{MAX}$ to avoid hotspots.

### 4.6.2 Criticality-based Random Sampling

Large-scale data centers operated by major cloud providers can have thousands or even millions of PMs that handle vast numbers of VMs. An optimal solution where every single PM is considered for each VM placement is highly impractical. Indeed, the deployment execution time is a Key Performance Indicator (KPI) for an effective VM placement. Moreover, our solution consists of using predictive ML models that require a considerable amount of real-time data collection(e.g., Sensors) which can have significant execution time based on the network's bandwidth availability.

Therefore, a sampling strategy is employed to reduce the size of the PM candidate set for each VM's allocation, more specifically a weighted random sampling based on the classical Accept-Reject method [45]. A random sample $V^{\text{RAND}}$ of size $k$ is derived from a set $V^{\text{PM}}$ of $m$ PMs where $k < m$. The inclusion probability $P_{\text{PM}}$ of a PM is proportional to its weight which is equal to its criticality $C_{\text{PM}}$. The criticality-based weight adds a thermal-aware aspect to the randomness of the sampling, reducing the probability of high criticality PMs being selected, hence, also improving hotspot avoidance. The sampling method is described in Algorithm (1).

**Algorithm 1** : Criticality-based Accept-Reject Sampling

```
$V^{\text{ALL}} \leftarrow V^{\text{PM}}, V^{\text{RAND}} \leftarrow \emptyset$
$k \leftarrow$ Sub-sample size, $\quad k < m$
while $|V^{\text{RAND}}| < k$ do
    $j \leftarrow$ Generate uniform random integer, $1 \leq j \leq m$
    $P_{\text{PM}} \leftarrow$ Criticality of $PM_j, PM_j \in V^{\text{PM}}, 0 \leq P_{\text{PM}} \leq 1$
    $u \leftarrow$ Generate uniform random integer, $0 \leq u \leq 1$
    if $u > P_{\text{PM}}$ then
        $V^{\text{RAND}} \leftarrow V^{\text{RAND}} \cup PM_j$
    $V^{\text{ALL}} \leftarrow V^{\text{ALL}} \backslash \{PM_j\}$
return $V^{\text{RAND}}$
```

## 4.6.3 Greedy Heuristic Algorithm

Finding the optimal solution for this problem (Equation 11) is an NP-hard problem, still significantly complex and unpractical for large-scale data centers despite the random sampling process, due to the high computational time. Therefore, for a practical solution that can give a near-optimal solution, we provide a simple greedy heuristic VM placement labeled (CPPS) described in Algorithm (2).

**Algorithm 2** : VM Placement - CPPS

```
Input: VM_list, PM_list
Output: VMs assignment to PMs
  $map \leftarrow \emptyset$
  for each $\text{VM} \in V^{\text{VM}}$ do
      $V^{\text{RAND}} \leftarrow$ Random Sampling: Algorithm $(1)$
      $destination \leftarrow Null$
      $PC^{min} \leftarrow \infty$
      for each $\text{PM} \in V^{\text{RAND}}$ do
          $PC_{\text{PM}} \leftarrow PC_{\text{PM}}^{t+1}$ : Equation (10)
          if $(PC_{\text{PM}} < Pc^{min})$ and Constraints (12) $-$ (18) Satisfied then
              $destination \leftarrow \text{PM}$
              $PC^{min} \leftarrow PC_{\text{PM}}$
      if $destination = Null$ then
          reject VM
      else
          $map \leftarrow map \cup (\text{VM}, destination)$
  return $map$
```

For each VM, a sub-sample of PM candidates is generated based on the accept-reject sampling described in Algorithm (1). The VM is then placed on the PM with the lowest $PC_{\text{PM}}$: predicted increase in power consumption and criticality for the next interval $t + 1$, while respecting the constraints of resource utilization and temperature thresholds described in Section 4.6.1. The worst-case and best-case complexity of CPPS is therefore, $O(NM)$ and $O(Nk)$, respectively, where N is the number of VMs, M is the number of PMs, and $k$ is the pre-defined sub-sample size of candidate PMs for placement.

## 4.7 Numerical Results

### 4.7.1 Environment

All experiments have been conducted in a private data center on a cluster of 8 HP ProLiant BL460c G8 physical machines(PMs) mounted on an HP C7000 chassis. Each of those PMs consists of 2 processors of 8 cores, 16 threads, and 128 GB RAM. Our test bed runs with OpenStack as the cloud management platform for Virtual Machines (VMs) management and follows a Three-Node architecture that consists of 3 systems: Controller Node, Compute Node, and Storage Node [39]. One PM is used as the controller node where most of the shared OpenStack services run and supply API, scheduling, and other shared services for the cluster. 6 PMs run as compute nodes where VM instances, also known as Nova compute instances are deployed, and 1 PM is used as the storage node to host data. Our placement algorithm is part of a custom VM management module implemented on the controller node for monitoring and managing the VMs on those 6 compute PMs.

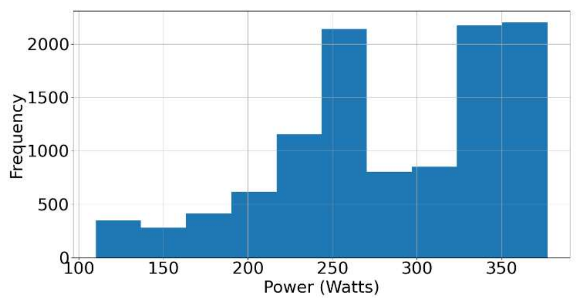
### 4.7.2 Load Emulation & Data Collection

Our environment can handle a maximum of 180 VMs with each compute node/PM having a limit of 30 VMs ($\theta_{vm}^{max} = 30$ Equation 13). All VMs are of an Ubuntu 20.04 LTS image [53] with a flavor of 2 VCPU, 4 GB Memory, and 20GB of disk. Load emulations on the VMs are done through stress software with Microsoft Azure's cloud VM traces [11] for real-like resource utilization patterns of data centers' VMs. Stress [51] is a tool for workload generation that can subject a system to a configurable measure of CPU, memory, and disk stress. The traces contain the measurements of the average CPU usage at 5 minutes intervals for real VMs running on Azure servers. Thus, using these traces, we are able to emulate real-like levels of PM's CPU utilization. Additionally, memory and disk operations are also added through the stress tool for generating traffic of different characteristics. At the start of the experiments, the placement module from the controller transfers the load emulation Python scripts with a unique VM trace file to the deployed VMs. OpenStack and Linux commands (e.g., vmstat, lsCPU) are then used for recording the Performance Monitoring Counters (PMCs). In addition, the placement module runs scripts that use IPMItool [40] for collecting and storing sensors' data including the fan speed and power consumption for all compute PMs.

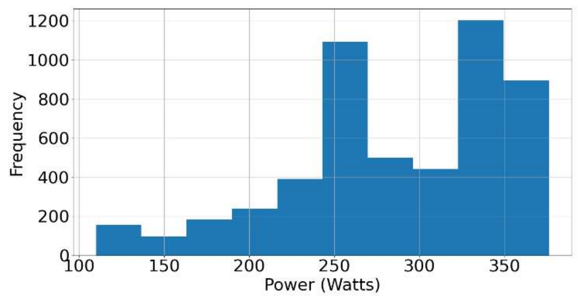
### 4.7.3 Evaluation: Predictive ML Models

This section presents a detailed performance evaluation and analysis of our proposed power and criticality prediction ML models labeled as 'XGB-Power' and 'XGB-Criticality', respectively. We conduct this evaluation using a test dataset (5,194 data points) derived by running multiple experiments on the setup presented in Section 4.7.1. The distribution of measured power and criticality in both the testing & training sets are illustrated in Fig. 20 and Fig. 21.

To gauge the power model's performance, we implement two baseline models for comparison. Firstly, an analytical formula from Wang *et al.*'s [56] that is dependent on a PM's idle & busy power, CPU utilization, and inlet temperature. In addition, we train an XGBoost model labeled 'XGB_R-power with a feature space limited to resource utilization features only (CPU, Memory, and Disk). All models are then evaluated using three metrics: mean absolute percentage error (MAPE), Root Mean Square Error (RMSE), and $R^2$ score expressed in Equations (19), (20), and (21), where $N$ is the number of data points, $A_i$ denotes the actual, and $P_i$ the predicted values at data point $i$, and $\bar{A}$ is the average of the actual values. Table 6 provides the evaluation results of the criticality and power models including the baselines.

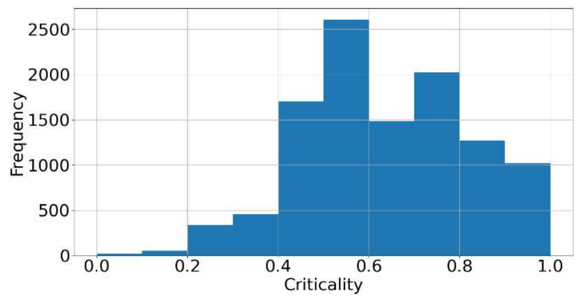


(a) Training Set

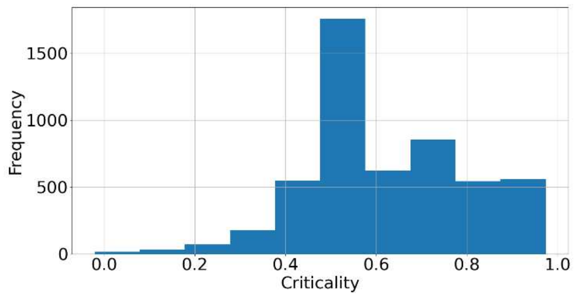


(b) Testing Set

Figure 20: Measured Power Distribution in Datasets



(a) Training Set



(b) Testing Set

Figure 21: Measured Criticality Distribution in Datasets

| **Models** | $R^2$ **(%)** | **RMSE** | **MAPE (%)** |
|---|---|---|---|
| XGB-Criticality | 90.9 | 0.05 | 7.3 |
| XGB-Power | 96.5 | 11.7 | 2.8 |
| XGB_R-Power | 90.4 | 19.5 | 4.9 |
| Wang *et al.* | 77.7 | 29.7 | 8.3 |

Table 6: Models' Performance.

$$\text{MAPE} = \left(\frac{1}{N}\sum_{i=1}^{N}\frac{|(A_i - P_i)|}{A_i}\right) \times 100 \tag{19}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(P_i - A_i)^2}{N}} \tag{20}$$

$$R^2 = \left(1 - \frac{\sum_{i=1}^{N}(A_i - P_i)^2}{\sum_{i=1}^{N}(A_i - \bar{A})^2}\right) \times 100, \tag{21}$$

### 4.7.4 Evaluation: VM Placement

All placement algorithms are implemented as part of the placement module on the controller node in the setup described in Section 4.7.1. Since the experiments run on a real environment limited to 6 compute nodes/physical machines, the random sampling procedure was expected to decrease the algorithm's performance. Hence, in addition to CPPS (Algorithm 2) with $k$ set to 3, we consider an algorithm labeled as CPP which omits the random sampling phase. To access our proposed solutions, we implement and evaluate three placement algorithms as baselines for comparisons: a resource-based load balancing (LB-R) which considers the PM with the lowest measured resource utilization as the best destination (Given in Algorithm 3), VM-based load balancing (LB-VM) which maintains an equal distribution of VMs among the PMs, and a greedy thermal-aware placement algorithm (TAS) from a VM scheduling framework proposed by Ilager *et al.* [24]. The evaluation metrics employed are the overall dynamic energy consumption and the average time fraction during which PMs operate near overload states, including instances of CPU bottlenecks and peak power usage. In addition, we also consider the CPU's and PMs' average temperature ($T_{avg}^{CPU}$& $T_{avg}^{PM}$) to evaluate heat generation. The metrics are described as follows;

**Algorithm 3** : VM Placement - (LB-R)

```
$map \leftarrow \emptyset$
for each $\text{VM} \in V^{\text{VM}}$ do
    $destination \leftarrow Null$
    $R^{min} \leftarrow \infty$
    for each $\text{PM} \in V^{\text{PM}}$ do
        $R_{\text{PM}} \leftarrow$ Equation (7)
        if $(R_{\text{PM}} < R^{min})$ and Constraints (12) $-$ (18) Satisfied then
            $destination \leftarrow \text{PM}$
            $R^{min} \leftarrow R_{\text{PM}}$
    if $destination = Null$ then
        reject VM
    else
        $map \leftarrow map \cup (\text{VM}, destination)$
return $map$
```

$$E_{Dynamic} = \sum_{\text{PM} \in V^{\text{PM}}} (P_{\text{PM}}^{avg} - P_{\text{PM}}^{idle}) \times t \tag{22}$$

$$Overload_{CPU} = \frac{1}{|V^{\text{PM}}|} \sum_{\text{PM} \in V^{\text{PM}}} \frac{t_{\text{PM}}^{CPU}}{t} \tag{23}$$

$$Overload_{Power} = \frac{1}{|V^{\text{PM}}|} \sum_{\text{PM} \in V^{\text{PM}}} \frac{t_{\text{PM}}^{POWER}}{t} \tag{24}$$

where $P_{\text{PM}}^{avg}$ is the average measured power, $P_{\text{PM}}^{idle}$ is the power consumption of the PM at idle state, t is the total experimentation time, $t_{\text{PM}}^{CPU}$ is the time spent by a PM with its run-queue size (number of active processes) larger than the number of logical CPUs, and $t_{\text{PM}}^{POWER}$ the time spent close to peak power consumption ($> 360W$).

| **Algorithm** | $E_{dynamic}(kWh)$ | $Overload_{CPU}$ | $Overload_{Power}$ | $T_{avg}^{CPU}(^{\circ}C)$ | $T_{avg}^{PM}(^{\circ}C)$ |
|---|---|---|---|---|---|
| CPP | 9.86 | 0.23 | 0.06 | 47.4 | 32.7 |
| CPPS | 10.24 | 0.26 | 0.06 | 47.8 | 32.8 |
| TAS | 10.59 | 0.32 | 0.13 | 48.5 | 32.8 |
| LB-R | 10.88 | 0.32 | 0.14 | 48.8 | 33.4 |
| LB-VM | 11.12 | 0.41 | 0.15 | 48.4 | 33.0 |

Table 7: Performance Comparison

We conducted several experiments, each ranging from 160 to 180 VMs deployed over a period of up to 10 hours. Those experiments sustained high utilization levels on all machines for an extended time period thus, ensuring a robust evaluation that enabled us to thoroughly assess the placement algorithms. Our proposed solution demonstrated its efficiency in handling high workloads while reducing the power consumption relative to the baseline algorithms. Indeed, CCP and CCPS reduced the total dynamic energy consumption $E_{Dynamic}$ by up to 6.9% and 3.4%, respectively, when compared to the best reference algorithm TAS. Our proposed algorithms also demonstrated efficient load allocation with CPP reducing $Overload_{CPU}$ by 28% and $Overload_{Power}$ by 57%. Since the PMs benefit from appropriate cooling with the temperature at inlets mostly maintained around 15 $^{\circ}C$, no hotspots were detected, that is, none of the components reached their critical thresholds during the experiments. However, CPP still reduces the CPUs' average temperature and the PMs' average temperature by 2.3% and 0.3%, respectively. Finally, Table 7 provides the detailed results of the experiment illustrated in Figure 22. Figure 23 and Figure 24 displays the $t_{\text{PM}}^{CPU}$ and $t_{\text{PM}}^{POWER}$ of each PM to demonstrate the proper load distribution of our algorithms.

## 4.8 Conclusion

The key motivation of this study was to implement a Virtual Machine placement algorithm that minimizes energy consumption while effectively managing workload distribution and

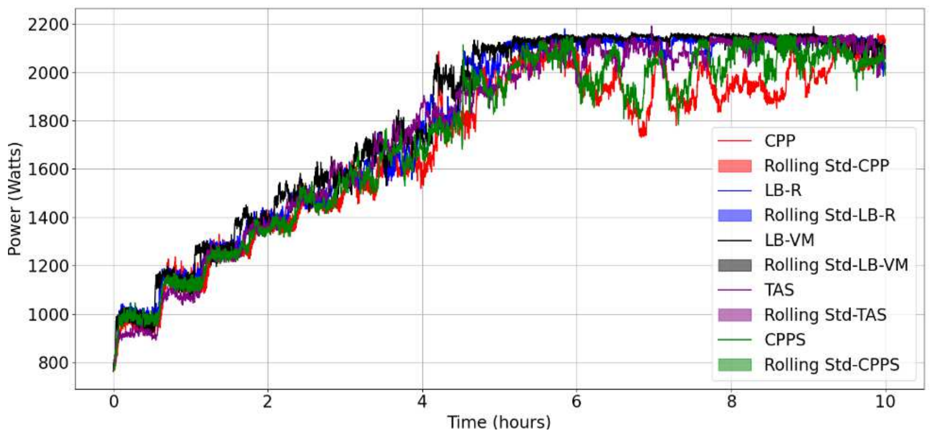


Figure 22: Power Consumption Comparison of Algorithms.

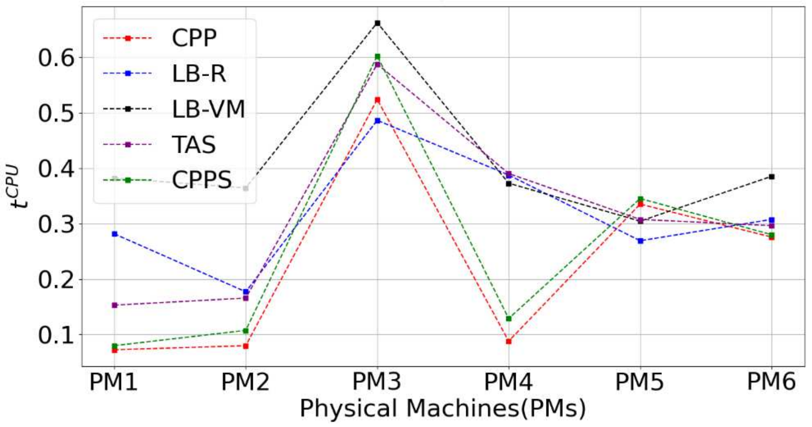


Figure 23: Percentage of time each PM spends at an overloaded CPU state. ($t_{\text{PM}}^{CPU}$)

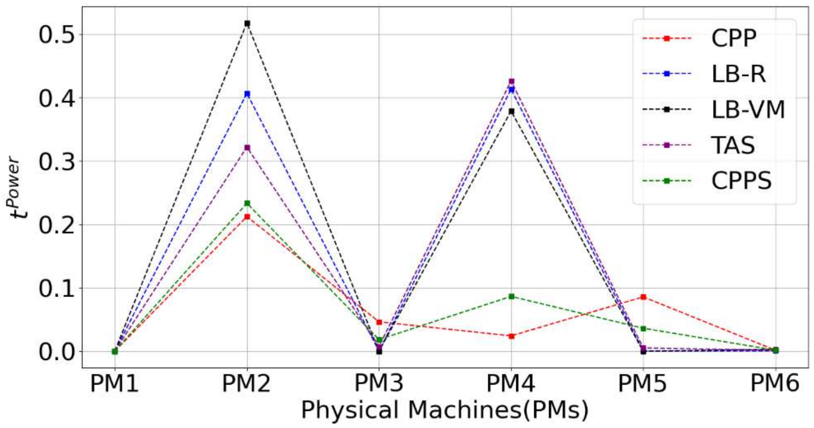


Figure 24: Percentage of time each PM spends to the peak power state. ($t_{\text{PM}}^{POWER}$)

avoiding hotspots. By making use of embedded heat sensors that provide real-time temperature data of physical machines, a new set of features was introduced for implementing highly accurate learning-based power and criticality prediction models. We then implemented a simple and highly scalable greedy algorithm that leverages the predictive model's output for proactive energy-aware and thermal-aware placements. Indeed, after a thorough performance evaluation of our proposed solution in an actual data center, it was found to reduce energy consumption by 7% while also maintaining an efficient load distribution and reducing the average CPU's temperature by 2%. Furthermore, this also implies a 7% reduction in the total cooling energy since the cooling power required to dissipate the heat produced by the physical machines is proportional to their consumed power. In the context of large-scale data centers, this holds immense potential, as even a 1% improvement can result in substantial cost savings, amounting to millions of dollars, and a significant reduction in carbon footprints. For instance, extrapolating our experiments' results for 1000 PMs and assuming a ratio of 1:0.8 for the cooling system's consumption, we can estimate a reduction of 220 kWh in the overall energy consumption. Moreover, the application of the placement algorithm can also be extended to VM destination selection during dynamic migrations. Indeed, our future work consists of improving this study by considering a complete VM's lifecycle: scheduling, placement, and dynamic migration for more specific DC traffic such as 5G Network's Virtual Machines or Containers.

# Chapter 5

# Conclusion & Future Works

## 5.1 Conclusions

The motivation for this study was the minimization of energy consumption in cloud data centers through efficient Virtual Machine placements. In the initial phase, we implemented a highly accurate prediction model for DC servers' power consumption. First, through several experiments, we investigated the challenges involved and demonstrated the ineffectiveness of analytical formulas in the context of data centers with heterogeneous traffic (e.g., 5G traffic with various applications) and physical machines. We addressed these issues by proposing the use of machine learning models with an enhanced feature space of performance monitoring counters (PMCs) and real-time sensor measurements. Indeed, our models showed a significant improvement in prediction accuracy. In the next phase, the power model was integrated into the core part of the study for a proactive energy-efficient Virtual Machine placement. Using the embedded heat sensors and the critical temperature thresholds of all PMs' internal components, we presented "Criticality" as a new feature that was employed for high-precision hotspot detection and energy-efficient VM placements. Moreover, we showed how the "Criticality" can be used for promoting a scalable placement based on Accept-Reject random sampling while maintaining thermal awareness. Our placement algorithm achieved up to 7% reduction in energy consumption, all the while maintaining an efficient load distribution reducing PM overload by up to 28%. In conclusion, our solution is able to produce significant improvement in multiple objectives with a reduction in both the compute and cooling subsystems' energy consumption. In the context of large-scale data centers, this result holds significant importance since even a 1% improvement in a DC can result in substantial cost savings, amounting to millions of dollars, and a reduction in carbon footprints.

## 5.2 Future Works

This study can propel future works for further improvements in multiple directions;

- It can be extended to apply in a complete framework of Virtual Machine management for further improved energy and temperature minimization. Indeed, the placement explored in this thesis is only concerned with the initial placement, which may lose its efficiency with time as the workload and states of the machines change dynamically. Thus, the next phase would be to apply this study to dynamic migration algorithms that handle the re-optimization of VMs' placement.

- Container-based services have gained substantial popularity and similar to VMs, urgently demand energy-efficient approaches. An interesting direction would be the adaptation of this study to different scenarios and architectures such as Container placement on server physical machines (bare-metal) or on Virtual Machines.

- One of the key arguments discussed in this thesis is the fact that different load profiles or characteristics heavily influence the behavior of the servers in relation to energy consumption. While this work can be applied generally to all Data Centers, another prospective study would be investigating its application to specific Data Centers or workload types such as 5G traffic, video streaming servers, or gaming servers.

# Bibliography

[1] I. Alan, E. Arslan, and T. Kosar. Energy-aware data transfer tuning. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 626–634, 2014.

[2] L. A. Barroso, U. Hölzle, and P. Ranganathan. The datacenter as a computer: Designing warehouse-scale machines. *Synthesis Lectures on Computer Architecture*, 13(3):i–189, 2018.

[3] R. Basmadjian, N. Ali, F. Niedermeier, H. de Meer, and G. Giuliani. A methodology to predict the power consumption of servers in data centres. pages 1 – 10, May 2011.

[4] A. Beloglazov, J. Abawajy, and R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing centers. *Future Generation Computer Systems*, 28(5):755–768, 2012.

[5] A. Beloglazov and R. Buyya. OpenStack Neat: a framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds. *Concurrency and Computation: Practice and Experience*, 27(5):1310–1333, 2015.

[6] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya. Chapter 3 - a taxonomy and survey of energy-efficient data centers and cloud computing systems. In M. Zelkowitz, editor, *Advances in Computers*, volume 82, pages 47–111. Elsevier, 2011.

[7] W. L. Bircher and L. K. John. Complete system power estimation using processor performance events. *IEEE Transactions on Computers*, 61(4):563–577, 2012.

[8] D. Bizo. Silicon heatwave: the looming change in data center climates. Technical Report 74, Uptime Institute, 2022.

[9] L. Caviglione, M. Gaggero, M. Paolucci, and R. Ronco. Deep reinforcement learning for multi-objective placement of virtual machines in cloud datacenters. *Soft Computing*, 25(19):12569–12588, 2021.

[10] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 785–794, 2016.

[11] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *26th Symposium on Operating Systems Principles*, pages 153–167, 2017.

[12] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. *ACM SIGARCH computer architecture news*, 35(2):13–23, 2007.

[13] F. Farahnakian, P. Liljeberg, and J. Plosila. Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning. In *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 500–507. IEEE, 2014.

[14] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, N. T. Hieu, and H. Tenhunen. Energy-aware VM consolidation in cloud data centers using utilization prediction model. *IEEE Transactions on Cloud Computing*, 7(2):524–536, 2019.

[15] J. Gao. Machine learning applications for data center optimization. *Google Research*, 2014.

[16] P. Gayam and A. Jagadeeshwara. Load balancing performance of DRS in vSphere 7.0, performance study. `https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/performance/drs-vsphere7-perf.pdf`, 2020.

[17] G. Ghatikar, M. A. Piette, S. Fujita, A. McKane, J. H. Dudley, A. Radspieler, K. Mares, and D. Shroyer. Demand response and open automated demand response opportunities for data centers. Technical report, Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2009.

[18] C. Ghribi, M. Hadji, and D. Zeghlache. Energy efficient VM scheduling for cloud data centers: Exact allocation and migration algorithms. In *IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid)*, pages 671–678, 2013.

[19] S. Gill, S. Tuli, A. Toosi, F. Cuadrado, P. Garraghan, R. Bahsoon, H. Lutfiyya, R. Sakellariou, O. Rana, S. Dustdar, and R. Buyya. Thermosim: Deep learning based framework for modeling and simulation of thermal-aware resource management for cloud computing environments. *Journal of Systems and Software*, 166(110596):234 – 248, April-June 2020.

[20] C. Guo, K. Xu, G. Shen, and M. Zukerman. Temperature-aware virtual data center embedding to avoid hot spots in data centers. *IEEE Transactions on Green Communications and Networking*, 5(1):497–511, 2021.

[21] K. Haghshenas and S. Mohammadi. Prediction-based underutilized and destination host selection approaches for energy-efficient dynamic vm consolidation in data centers. *The Journal of Supercomputing*, pages 1–18, 2020.

[22] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[23] S. Ilager and R. Buyya. Energy and thermal-aware resource management of cloud data centres: A taxonomy and future directions. *ArXiv*, abs/2107.02342, 2021.

[24] S. Ilager, K. Ramamohanarao, and R. Buyya. Thermal prediction for efficient energy management of clouds using machine learning. *IEEE Transactions on Parallel and Distributed Systems*, 32(5):1044 – 1056, 2021.

[25] C. Jin, X. Bai, C. Yang, W. Mao, and X. Xu. A review of power consumption models of servers in data centers. *Applied Energy*, 265:114806, 2020.

[26] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya. Virtual machine power metering and provisioning. In *1st ACM symposium on Cloud computing*, pages 39–50, 2010.

[27] R. Kavanagh and K. Djemame. Rapid and accurate energy models through calibration with IPMI and RAPL. *Concurrency and Computation: Practice and Experience*, 31(13):e5124, 2019.

[28] C. Kelley, H. Singh, and V. Smith. Data center efficiency and it equipment reliability at wider operating temperature and humidity ranges. *The Green Grid, White Paper*, 50, 2012.

[29] S. Kim and H. Kim. A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, 32(3):669–679, 2016.

[30] Kubernetes. Kubernetes scheduler. `https://kubernetes.io/docs/concepts/scheduling-eviction/kube-scheduler/`.

[31] X. Li, P. Garraghan, X. Jiang, Z. Wu, and J. Xu. Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy. *IEEE Transactions on Parallel and Distributed Systems*, 29(6):1317 – 1331, 2018.

[32] Y. Li, Y. Wang, B. Yin, and L. Guan. An online power metering model for cloud environment. In *IEEE 11th International Symposium on Network Computing and Applications*, pages 175–180, 2012.

[33] C.-H. Lien, Y.-W. Bai, and M.-B. Lin. Estimation by software for the power consumption of streaming-media servers. *IEEE Transactions on Instrumentation and Measurement*, 56(5):1859–1870, 2007.

[34] Linux vmstat. vmstat(8) - Linux man page. `https://linux.die.net/man/8/vmstat`.

[35] R. Milocco, P. Minet, E. Renault, and S. Boumerdassi. Proactive data center management using predictive approaches. *IEEE Access*, 8:161776–161786, 2020.

[36] F. Moges and S. Abebe. Energy-aware VM placement algorithms for the OpenStack neat consolidation framework. *Journal of Cloud Computing*, 8(1):1–14, 2019.

[37] P. T. Nalveer Moocheet, Brigitte Jaumard and L. Eleftheriadis. A sensor predictive model for power consumption using machine learning. In *12th International Conference on Cloud Networking*. IEEE, 2023.

[38] OpenStack. Scheduling. `https://docs.openstack.org/mitaka/config-reference/compute/scheduler.html`.

[39] Oracle. Installing and configuring openstack (kilo) in oracle® solaris, three-node architecture overview. `https://docs.oracle.com/cd/E65465_01/html/E61044/archover.html`.

[40] Oracle. Using ipmitool to view system information.

[41] V. Perumal and S. Subbiah. Power-conservative server consolidation based resource management in cloud. *International Journal of Network Management*, 24:415–432, Nov./Dec. 2014.

[42] Y. Qin, H. Wang, S. Yi, X. Li, and L. Zhai. Virtual machine placement based on multi-objective reinforcement learning. *Applied Intelligence*, 50:2370–2383, 2020.

[43] M. Ranjbari and J. Torkestani. A learning automata-based algorithm for energy and SLA efficient consolidation of virtual machines in cloud data centers. *Journal of Parallel and Distributed Computing*, 113:55–62, 2018.

[44] M. Rezaei-Mayahi, M. Rezazad, and H. Sarbazi-Azad. Temperature-aware power consumption modeling in hyperscale cloud data centers. *Future Generation Computer Systems*, 94:130–139, 2019.

[45] R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo method.* John Wiley & Sons, 2016.

[46] T. Salah, M. J. Zemerly, C. Y. Yeun, M. Al-Qutayri, and Y. Al-Hammadi. Performance comparison between container-based and vm-based services. In *2017 20th Conference on Innovations in Clouds, Internet, and Networks (ICIN)*, pages 185–190. IEEE, 2017.

[47] S.D.A.Shah, M. Gregory, and S. Li. Cloud-native network slicing using software defined networking based multi-access edge computing: A survey. *IEEE Access*, 9:10903–10924, 2021.

[48] R. Shaw, E. Howley, and E. Barrett. An advanced reinforcement learning approach for energy-aware virtual machine consolidation in cloud data centers. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 61–66. IEEE, 2017.

[49] M. Song and K. Chen. Numerical study on the optimal power distribution of server racks in a data center. *Building Simulation*, 16:983 — 995, 2023.

[50] S. Song, K. J. Barker, and D. J. Kerbyson. Unified performance and power modeling of scientific workloads. In *1st International Workshop on Energy Efficient Supercomputing (E2SC)*, 2013.

[51] L. Stress. stress(1) - linux man page. `https://linux.die.net/man/1/stress`.

[52] B. Tudor and Y. Teo. On understanding the energy consumption of arm-based multicore servers. In *ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*, pages 267–278, 2013.

[53] Ubuntu. Ubuntu 20.04.5 lts (focal fossa), 2023.

[54] Z. Usmani and S. Singh. A survey of virtual machine placement techniques in a cloud data center. *Procedia Computer Science*, 78:491–498, 2016. 1st International Conference on Information Security and Privacy 2015.

[55] B. Wang, F. Liu, and W. Lin. Energy-efficient VM scheduling based on deep reinforcement learning. *Future Generation Computer Systems*, 125:616–628, 2021.

[56] Y. Wang, D. Nörtershäuser, S. Le Masson, J.-M. Menaud, et al. An empirical study of power characterization approaches for servers. In *International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, pages 1–6, 2019.

[57] X. Wen, G. Gu, Q. Li, Y. Gao, and X. Zhang. Comparison of open-source cloud management platforms: OpenStack and OpenNebula. In *9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 2457 – 2461, 2012.

[58] M. Witkowski, A. Oleksiak, T. Piontek, and J. Węglarz. Practical power consumption estimation for real life HPC applications. *Future Generation Computer Systems*, 29(1):208–217, 2013.

[59] P. Xiao, Z. Hu, D. Liu, G. Yan, and X. Qu. Virtual machine power measuring technique with bounded error in cloud environments. *Journal of Network and Computer Applications*, 36(2):818–828, 2013.

# Appendix A

# Additional Experiment Results

## A.1 Experiment 1

180 Virtual Machine was deployed over a period of close to 7 hours. CPP was recorded to have 5.1% less energy consumption than the best reference algorithm TAS. Average PM's overload time and CPU temperature were reduced by 22.3% and 0.8%, respectively. Figure 25 to Figure 34 and Table 8 provide the detailed results of Experiment 1.

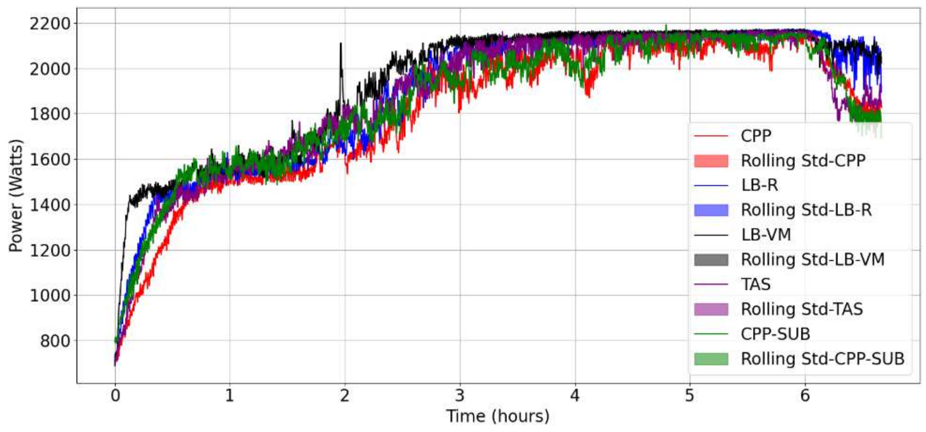


Figure 25: Experiment 1 - Power Consumption Comparison of Algorithms.

| Algorithm | $E_{dynamic}(kWh)$ | $Overload_{CPU}$ | $Overload_{Power}$ | $T_{avg}^{CPU}(°C)$ | $T_{avg}^{PM}(°C)$ |
|---|---|---|---|---|---|
| CPP | 7.333 | 0.271 | 0.085 | 49.35 | 33.71 |
| CPPS | 7.607 | 0.312 | 0.091 | 49.60 | 33.59 |
| TAS | 7.731 | 0.349 | 0.098 | 49.76 | 33.59 |
| LB-R | 7.892 | 0.389 | 0.164 | 49.88 | 34.05 |
| LB-VM | 8.188 | 0.464 | 0.215 | 50.16 | 34.00 |

Table 8: Experiment 1 Results.

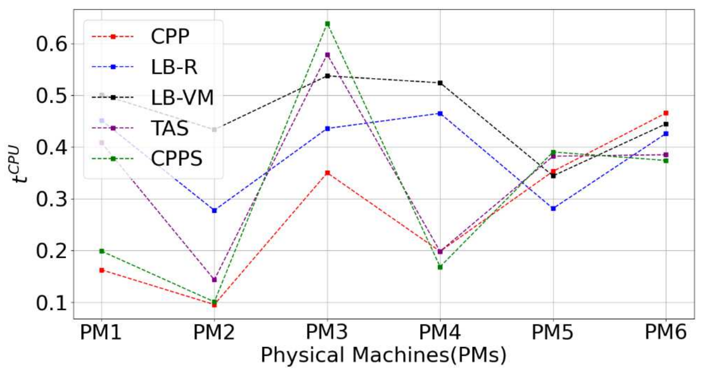


Figure 26: Experiment 1 - Percentage of time each PM spends at an overloaded CPU state. ($t_{\mathrm{PM}}^{CPU}$)

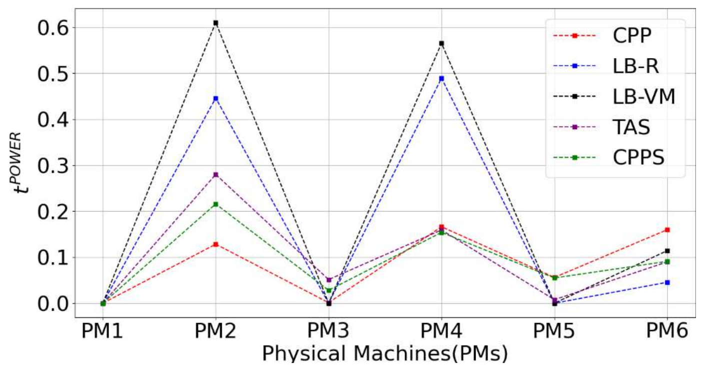


Figure 27: Experiment 1 - Percentage of time each PM spends to the peak power state. ($t_{\mathrm{PM}}^{POWER}$)

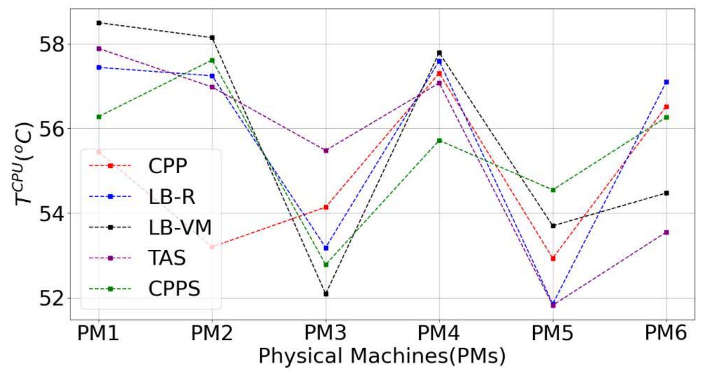


Figure 28: Experiment 1 - Average Temperature at Processor 1 ($t_{\mathrm{PM}}^{CPU}$)

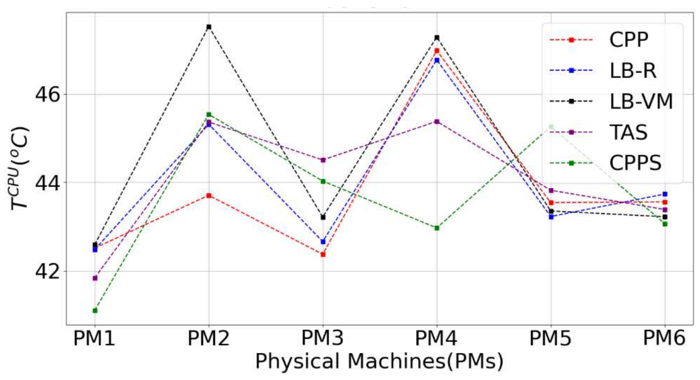


Figure 29: Experiment 1 - Average Temperature at Processor 2 ($t_{\text{PM}}^{POWER}$)

## A.2 Experiment 2

160 Virtual Machine was deployed over a period of close to 8.5 hours with a batch of 10 VMS deployed at every 20-minute interval. CPP was recorded to have 4.4% less energy consumption than the best reference algorithm TAS. Average PM's overload time and CPU temperature were reduced by 26.0% and 0.2%, respectively. Figure 30 to Figure 34 and Table 9 provide the detailed results of Experiment 2.

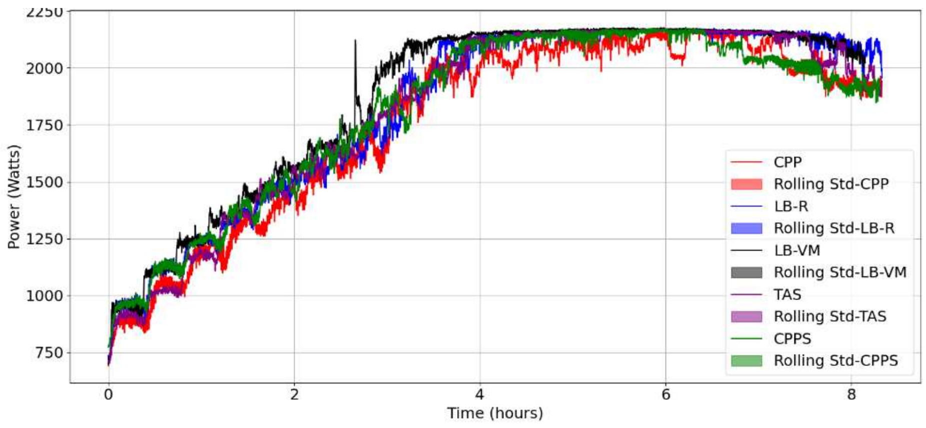


Figure 30: Experiment 2 - Power Consumption Comparison of Algorithms.

| **Algorithm** | $E_{dynamic}(kWh)$ | $Overload_{CPU}$ | $Overload_{Power}$ | $T_{avg}^{CPU}(^{\circ}C)$ | $T_{avg}^{PM}(^{\circ}C)$ |
|---|---|---|---|---|---|
| CPP | 8.804 | 0.368 | 0.131 | 48.72 | 32.80 |
| CPPS | 9.134 | 0.440 | 0.172 | 48.99 | 33.07 |
| TAS | 9.209 | 0.497 | 0.192 | 48.83 | 32.84 |
| LB-R | 9.383 | 0.474 | 0.173 | 48.71 | 32.81 |
| LB-VM | 9.673 | 0.531 | 0.218 | 49.27 | 33.22 |

Table 9: Experiment 2 Results.

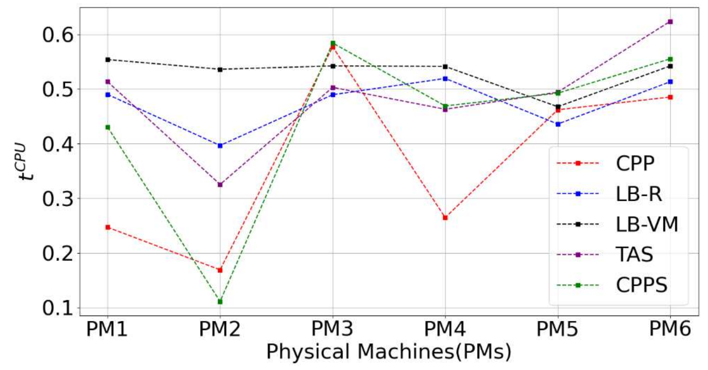


Figure 31: Experiment 2 - Percentage of time each PM spends at an overloaded CPU state. ($t_{\text{PM}}^{CPU}$)

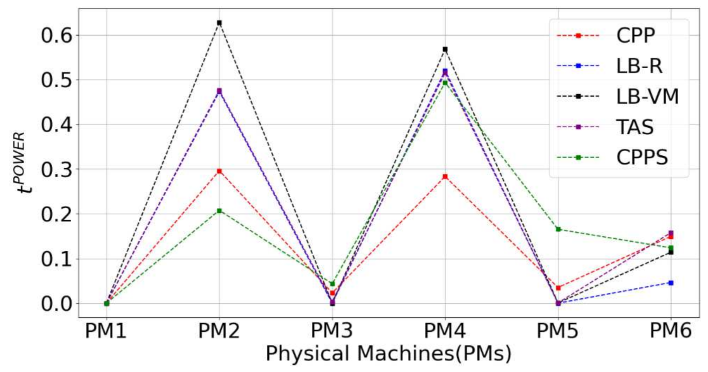


Figure 32: Experiment 2 - Percentage of time each PM spends to the peak power state. ($t_{\text{PM}}^{POWER}$)

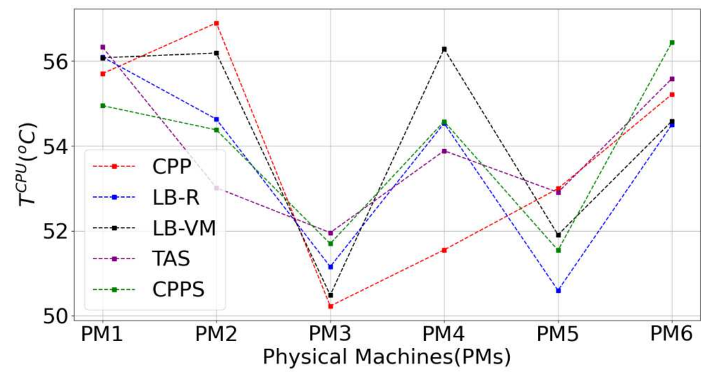


Figure 33: Experiment 2 - Average Temperature at Processor 1 ($t_{\text{PM}}^{CPU}$)

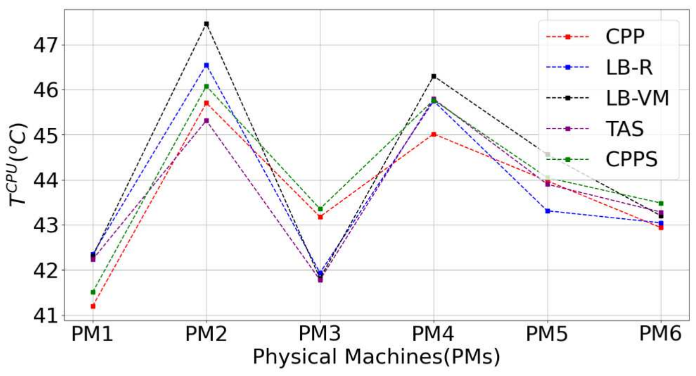


Figure 34: Experiment 2 - Average Temperature at Processor 2 ($t_{\text{PM}}^{POWER}$)