Alma Mater Studiorum · Università di Bologna

DOTTORATO DI RICERCA IN

ingegneria elettronica, telecomunicazioni e tecnologie dell'informazione
34 ciclo

Settore Concorsuale: 09/E3 - ELETTRONICA
Settore Scientifico Disciplinare: ING-INF/01 - ELETTRONICA

Monitoring and Prediction of Thermal Emergencies in High Performance Computing Systems

Presentata da: Mohsen Seyedkazemi Ardebili

Supervisore:
Prof. Luca Benini

Coordinatore Dottorato:
Prof. Aldo Romani

Co-supervisore:
Prof. Andrea Bartolini

Esame finale anno 2022

Abstract

Datacenters are at the heart of the AI, industry 4.0, and cloud revolution. Modern scientific discoveries are driven by an unsatisfiable demand for computational resources. High-Performance Computing HPC systems are an aggregation of computing power to deliver considerably higher performance than one typical desktop computer can provide, to solve large problems in science, engineering, or business. An HPC room in the datacenter is a complex controlled environment that hosts thousands of computing nodes that may consume electrical power in the range of megawatts. Due to the increasing total power and power density of computing nodes, the overall datacenter computing capacity is often capped by peak power consumption and temperature dissipation bottlenecks.

In the datacenter, a thermal anomaly is a suspicious/abnormal pattern in the monitoring signals. The severity of the anomaly can be different, and in extreme conditions, it can yield the outage of the datacenter. Although thermal anomalies are very rare events, anomaly detection and prediction in time is vital to avoid IT and facility equipment damage and outage of the datacenter, with severe societal and business losses. For this reason, automated approaches to detect thermal anomalies in datacenters have considerable potential.

This thesis analyzed and characterized the power and thermal properties of a Tier0 datacenter [1] during production and abnormal thermal conditions. Then, a Deep Learning (DL)-powered thermal hazard prediction framework is proposed. Finally, the anomaly detection task in the HPC room is investigated by defining more complex statistical rules-based anomaly detection methods and advanced Deep Learning DL-based thermal anomaly detection methods. The anomaly detection models are validated against real thermal hazard events reported for the studied HPC cluster while in production. To the best of my knowledge, this thesis is the first empirical study of thermal anomaly detection and prediction techniques of a real large-scale HPC system. It is based on real HPC room monitoring data at CINECA. This study is done based on real data of in-production HPC cluster and HPC room facilities and never used any synthetic data or artificial anomalies. [2]

[1] refers to CINECA, the most powerful supercomputing center for scientific research in Italy and one of the most powerful supercomputers in the world, based on the TOP500 list. HPC clusters of CINECA ranked 18th in November 2021, 9th in June 2020, and 21st in June 2019 in the TOP500 list [1–4]

[2] For this thesis, I used a large-scale dataset, monitoring data of tens of thousands of sensors for around 24 months with a data collection rate of around 20 seconds.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my sincere gratitude to my supervisor Prof. Luca Benini for the continuous support of my Ph.D. study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D. study.

My special thanks go to Prof. Andrea Bartolini, who has been my co-supervisor during these years of doctoral study. Andrea has been a mentor and a friend from whom I have learned the vital skills and discipline of academic research. His professional guidance and careful scrutiny of my research work have been invaluable.

Apart from my supervisors, I won't forget to express my gratitude to the rest of the team: Prof. Andrea Acquaviva, Dr. Francesco Beneventi, Prof. Davide Brunelli, and Prof. Davide Rossi for giving the encouragement and sharing insightful suggestions. They all have played a major role in polishing my research writing skills. Their endless guidance is hard to forget throughout my life.

I would like to thank all the CINECA staff, particularly Dr. Carlo Cavazzoni, Dr. Giovanni Bortolotti, and Dr. Daniele Cesarini. Without their precious support and the possibility to use the HPC systems of CINECA, it would not be possible to conduct this research.

I thank my fellow labmates for their technical help and friendship; I have appreciated a lot the fun we have had in these years. In no particular order: Manuele Rusci, Alessandro Capotondi, Daniele Cesarini, Marcello Zanghieri, Alessio Burrello, Tommaso Polonelli, Andrea Borghesi, Davide Rossi, Francesco Beneventi, Francesco Conti, and Giuseppe Tagliavini.

Also, I thank Fabio Cumella, Angela Cavazzini, and Giuseppe Stampati for the invaluable help to fill out the bureaucracy documents during these years.

I would also like to acknowledge the support and encouragement of all my friends (in no particular order): Arash Bozorgchenani, Mohammad Esmalifalak, Moien Bahardoost, Neda Mazaheri, Mohamad Jamali, and Mahsa (Maryam) Deldadeh.

Last but not least, I must express my very profound gratitude to my parents, family, and relatives: My wife Fereshteh Zavvari, My mother Najibeh Ashouraei, my father Mirreza, and my father in law Ahad Zavvari, my mother in law Roghayeh

Zavvari, My brothers Mohamad, Ehsan, My sister in law Shiva, Sara, Zahra, My brother in law Sadra, and Amir for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching. This achievement would not have been possible without you.

The work of this thesis has been conducted in the context of EU H2020-JTI-EuroHPC-2019-1 project REGALE (g.n. 956560) and EU H2020 ICT/2018 project IoTwins (g.n. 857191) and EU H2020 research and innovation programme "European Processor Initiative" (g.n. 826647) and Emilia-Romagna POR-FESR 2014-2020 project "SUPER: SuperComputing Unifier Platform – Emilia-Romagna and CINECA.

Thank you all.
Mohsen

Chapter 1

Introduction

Datacenters are at the heart of the AI, industry 4.0, and cloud revolution. At large, a datacenter is a facility that hosts a network of computing and storage resources, and to ensure continuity of the services, it typically contains redundant components for data communication, power distribution, air conditioning, and fire suppression. One of the services that datacenters provide is High-Performance Computing (HPC). HPC systems are an aggregation of computing power to deliver considerably higher performance than one typical desktop computer can provide, to solve large problems in science, engineering, or business. An HPC room in data centers is a complex controlled environment that hosts several HPC clusters and other required facilities like a high-performance communication network and cooling systems. Each HPC cluster comprises thousands of computing nodes that consume electrical power in the range of megawatts. Due to the increasing total power and power density of computing nodes, the overall datacenter computing capacity is often capped by peak power consumption and temperature dissipation bottlenecks.

1.1 Motivations

Homogeneous Performance To prevent the creation of cold and hot spots in the HPC room, which reduces the cooling system efficiency and the homogeneous performance between all the nodes, requires complex cooling solutions, but they might not be sufficient. Therefore, thermal monitoring HPC room is necessary to capture the effects of the power dissipated by computing nodes to optimize the cooling while precluding thermal hazards. In addition, a complex thermal dissipation system can have severe thermal hazards, which in turn jeopardizes the availability of HPC services.

Thermal Hazard Prediction To dissipate the heat generated by the power consumption of the nodes, forced air/liquid flow is employed, costing millions of Euros per year [6]. Reducing this cost involves using free-cooling (the capability to exploit the outside air, using only the air conditioner blowers to circulate it in the room) and average case design, etc., which can create a cooling shortage and thermal hazards. When a thermal hazard happens, the system administrators and the facility manager must stop the production to avoid IT equipment damage and wear-out. Considering the fact that continuity of some applications or services is vital (mission-critical application) and millions of Euros of capital expenditures (CAPEX) (+ a portion of operating expenses (OPEX)) of a datacenter/HPC cluster, the outage of the datacenter has severe social and financial side effects. Therefore, predicting the failure in advance to prevent the data center's outage is extremely important.

Thermal Anomaly In a datacenter, an anomaly is a suspicious pattern in the monitoring signals of the HPC room. An anomaly can initiate due to; *(i)* an inappropriate working of the cooling system or subsystem, *(ii)* inconsistency between the different cooling systems in the HPC room, *(iii)* abnormal computing demand, *(iv)* an extreme hotspot during the summer can affect the cooling systems' capacity, *(v)* fast variation of some monitoring signals that could not support by other signals, *(vi)* it can be complex temporal and/or spatial relations of the different monitoring signals, which is unclear for human experts but can be identified by machine learning approaches. The severity of the anomaly can be different, and in extreme conditions, it can yield the outage of the datacenter. Although anomalies in HPC systems are very rare events, anomaly detection is vital due to the significant harmful consequence of anomalies. A thermal hazard (which is a kind of anomaly) is a dramatic increase in node temperature, which can lead to the outage of the datacenter. Detecting thermal hazards in time is extremely important to avoid IT and facility equipment damage and outage of the datacenter, with severe societal and business losses. For this reason, automated approaches to detect thermal hazards in datacenters have considerable potential.

1.2 Contributions

It is important to emphasize that to the best of my knowledge; this thesis is the first empirical study of thermal anomaly detection and prediction techniques of a real large-scale HPC system. It is based on real HPC room monitoring data at CINECA, which hosts Marconi HPC clusters (Marconi is the 21st most powerful computing system based on the TOP500 list in June 2019 [2]) and Marconi100 (Marconi100 is the 18th most powerful computing system based on the TOP500

list in November 2021 [1]). This study is done based on real data of in-production HPC cluster and HPC room facilities and never used any synthetic data or artificial anomalies. For this thesis, I used a large-scale dataset, monitoring data of tens of thousands of sensors for around 24 months with a data collection rate of around 20 seconds. The monitoring data is collected employing a holistic monitoring system, namely ExaMon (Chapter 2), one of the state-of-the-art HPC monitoring systems developed by other members of our group at the University of Bologna [7].

The first contribution of the thesis (Chapter 2) is to define the HPC facility, cluster, room, and cooling where results were conducted. This refers to the most powerful supercomputing center for scientific research in Italy, and one of the most powerful supercomputers in the world, based on the TOP500 list (Marconi100 HPC cluster is the 18th most powerful computing system based on the TOP500 list in November 2021 [1], and Marconi HPC cluster is the 21st most powerful computing system based on the TOP500 list in June 2019 [2]).

The second contribution of the thesis (Chapter 3) is, analyzing and characterizing the thermal properties of a Tier0 datacenter deploying advanced cooling technologies. Specifically, the spatial and temporal heterogeneity during production and thermal hazards are studied. Chapter 3 gives quantitative evidence of thermal bottlenecks in real-life production workload, showing the presence of significant spatial thermal heterogeneity, which could be exploited by thermal-aware job scheduling and datacenter-room runtime workload adaptation and distribution. For instance, the inlet temperature of the nodes increases vertically with an average difference of $6.5°C$ from the top and bottom nodes, and measured data confirm that fans of nodes of bottom nodes work with lower speed (RPM) and consume 15.8 Watt less ($\sim 6\%$) than top nodes. Meanwhiles the monthly average inlet temperature for nodes at the same height in the room experienced up to 10.8 $°C$ of difference.

The third contribution of the thesis (Chapter 4) is the study of thermal hazards signatures on a Tier-0 datacenter room's monitoring data during a full year of production. Based on the statistical analysis of true thermal hazard events, a set of statistical rules to identify the thermal hazards on the inlet and outlet temperature measurements of all nodes of a room are defined. This chapter proposed a thermal hazard prediction framework to function on large-scale time-series data, which is composed of three main components: *(i)* data collection and storage part, *(ii)* data extraction, preprocessing (e.g., missed data handling, time alignments), label generator, and data loader part, *(iii)* Deep Learning (DL)-powered thermal hazard prediction system (training and inference). Different classical machine learning and deep learning models in predicting the thermal hazard events are investigated,

which would give ample time for taking proactive countermeasures.

The fourth contribution of the thesis (Chapter 5) is thermal anomaly detection. DL-based thermal hazard prediction and statistical rule-based method for thermal hazard or anomaly definition are possible only when thermal statistics of the HPC room is constant, which is not true due to: the yearly ambient temperature fluctuations, dataset's complexity, monitoring signal's dynamism, manual update of the cooling setpoints, etc. Chapter 5 challenges this task by defining more complex statistical rules-based anomaly detection methods and focusing on advanced DL-based thermal anomaly detection methods. In addition to node-level data, the approaches presented in this chapter utilize cluster/room/facility level metrics (e.g., cooling systems metrics) and create a large dataset (4 months) to train anomaly detection models. Finally, the anomaly detection models are validated against real thermal hazard events reported for the studied HPC cluster while in production.

1.3 Thesis Overview

The following contains a brief overview of the thesis outline. Figure 1.1 outlines chapter dependencies.

Chapter 1 is an introduction that contains a brief description of thesis context and motivations, main contributions of the thesis, and organization of the thesis.

Chapter 2 provides background about HPC facility, cluster, room, and cooling where studies were conducted.

Chapter 3 studies thermal and power characterization of two main computing rooms in the CINECA datacenter, leveraging two completely different approaches.

Chapter 4 proposes and examines a Deep Learning DL-powered thermal hazard prediction framework to function on large-scale time-series data.

Chapter 5, in order to anomaly detection in the monitoring data of computing nodes and HPC facilities, proposes and examines two set tools: 1- Rule-based Statistical Methods and 2- Semi-supervised Machine Learning-based Methods.

Chapter 6 concludes the thesis with a short summary.

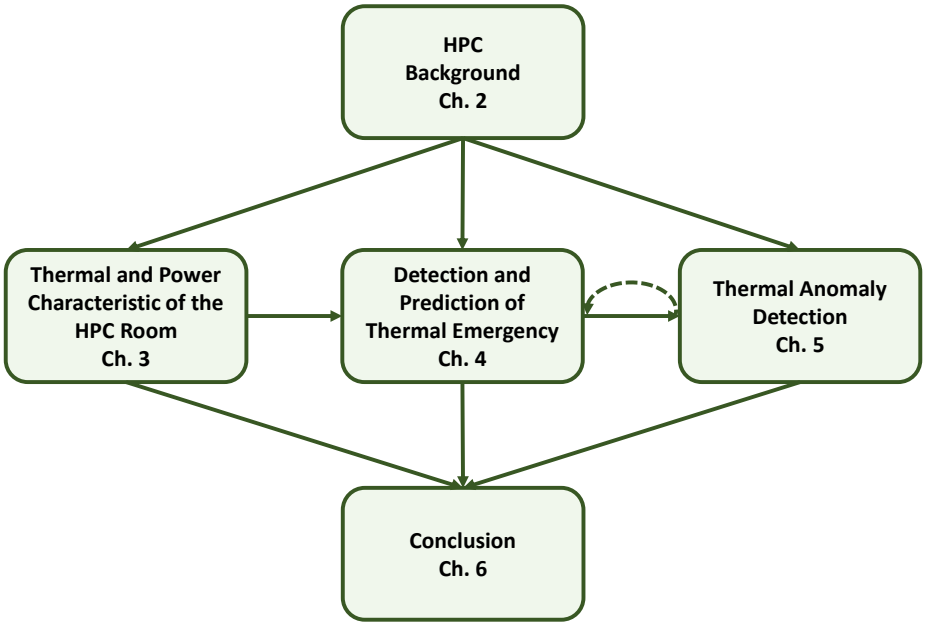


Figure 1.1: Thesis overview and dependencies between chapters.

Chapter 2

Background

2.1 Overview

High-performance computing (HPC) most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business [8]. A rack is a container designed to house servers, networking devices, cables, and other data center computing equipment; in the context of this thesis, a rack is a container that contains multiple chassis, and each chassis can host one or multiple computing nodes. An HPC room is a room in a datacenter that hosts one or multiple HPC clusters. Thousands of computing nodes in the HPC room may consume megawatts of electrical power, which is entirely converted into heat; efficiently dissipating heat requires a sophisticated cooling system [9].

2.2 CINECA

CINECA is a non-profit consortium of 69 Italian universities, 27 national public research centers, the Italian Ministry of Universities and Research (MUR), and the Italian Ministry of Education (MI), and was established in 1969 Casalecchio di Reno, Bologna [10]. It is the most powerful supercomputing center for scientific research in Italy, and one of the most powerful supercomputers in the world, based on the TOP500 list of the: Marconi100 HPC cluster, with about 32 PFlop/s, is ranked 18th (list of November 2021) most powerful computing system in the world [1] and Marconi HPC clusters (A2, A3), with about 20 PFlop/s, is ranked 21st (list of June 2019) most powerful computing system in the world [2]. CINECA has three HPC rooms F, M, and N, hosting HPC clusters and other facilities. This thesis mostly focused on rooms F and N, which host powerful HPC clusters of CINECA. The

CINECA datacenter features a holistic monitoring framework, namely ExaMon (more detail in section 2.3), which aggregates a wide set of telemetry data [7]. ExaMon is one of the state-of-the-art datacenter monitoring systems [11]

2.2.1 CINECA Cooling Technologies

CINECA HPC Rooms are cooled with Computer Room Air Conditioning (CRAC) units by the Direct Expansion (DX) Airconditioning system. In DX Air-conditioning, the air used for cooling the room is directly passed over the cooling coil. Some of these CRAC units support the Direct Free Cooling (DFC) system, which is referred to by the CRAC+DFC in this thesis. The DFC system is designed to reduce energy dissipation and improve the carbon footprint by utilizing the external cold air for cooling the room. In this case, the DFC system starts to work when the outdoor temperature is lower than 18^oC. Without the DFC system, the CRAC units work in standard air recirculation mode with refrigeration-based cooling. Empowering the CRAC units with a DFC system can reduce the compressor's operation. The hot/cold aisle approach is employed to cool the room (figure 2.1). The cold airflow moves under the raised floor and gets to the loaded areas; then, the hot air returns to the CRAC units above the raised floor.

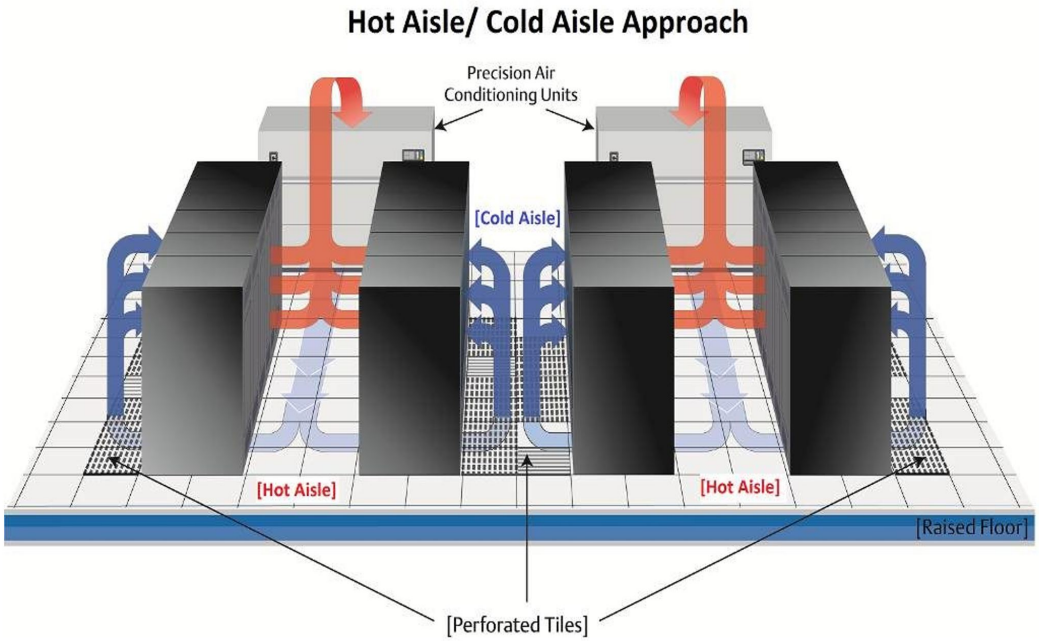


Figure 2.1: HPC Room Air Cooling System [5].

Also, there is a water cooling system for Rear Door Heat Exchangers (RDHX), with the chiller loop (cold loop) temperature around 12^oC to 17^oC, and RDHX

loop (hot loop) temperature around 23^oC to 30^oC (figure 2.2). The RDHX device is placed in front of the hot outlet airflow of the compute node. During operation, the compute node's hot airflow is forced through the RDHX device by the compute node fans, and exchanges heat from the hot air to circulating water from a chiller. Thus, the compute node outlet air temperature reduces before its discharge into the datacenter. RDHX is used to augment the computing density in air-cooled computing rooms.

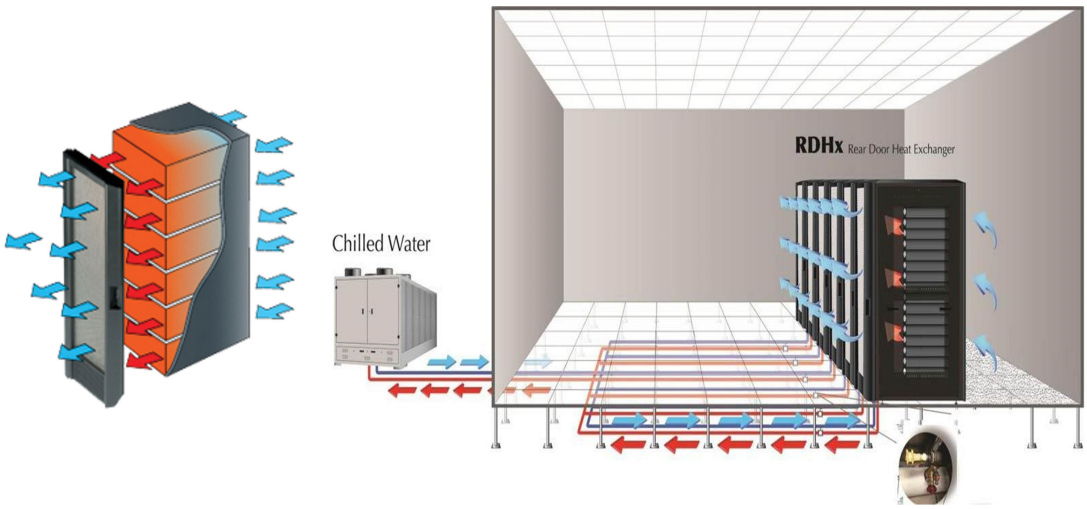


Figure 2.2: Rear Door Heat Exchanger (RDHX) [5].

2.2.2 Room F - Marconi A2 (KNL)

The Marconi is Tier-0 cluster in the CINECA datacenter, which is based on the Lenovo NeXtScale platform. Room F before Marconi100 hosted the largest partition of Marconi A2. The Marconi A2 cluster is based on the 68-cores Intel Xeon Phi7250 (KnightLandings) at 1.4 GHz, with many-core architecture (Intel OmniPath Cluster), provided about 250 thousand cores(68 cores/node, 244.800 cores in total) with the computational power of around 11Pflop/s. Each node has 16 GB/node MCDRAM + 96 GB/node DDR4 [12].

Figure 2.3 depicts the layout of the HPC Marconi A2 (KNL) room F in CINECA. In the Marconi A2 room F, 46+1 racks (one of them is a rack of switches) are located in three rows. Each rack is composed of 18 chassis in different height, and each chassis has four computing nodes. Chassis one (C1) is in the bottom, and chassis 18 (C18) is the highest one. There are six computer room air conditioning (CRAC) units that support the two cold aisles. Four of these CRAC units have the Direct Free Cooling system (DFC). All racks are equipped with RDHX, and RDHX of racks are in the hot aisle. The CINECA datacenter features a holistic monitoring

framework, namely ExaMon (more detail in section 2.3), which aggregates a wide set of telemetry data [7]. For each node and its associated components, such as voltage regulators and fans, the Intelligent Platform Management Interface (IPMI) provides remote telemetry access to the built-in sensors [13]. The ExaMon monitoring system collects sensor data with the IPMI interface with 20 seconds sampling rate [7]. ExaMon monitored data is stored in its internal KairosDB database as time traces and remotely accessible through RESTfull APIs [7].

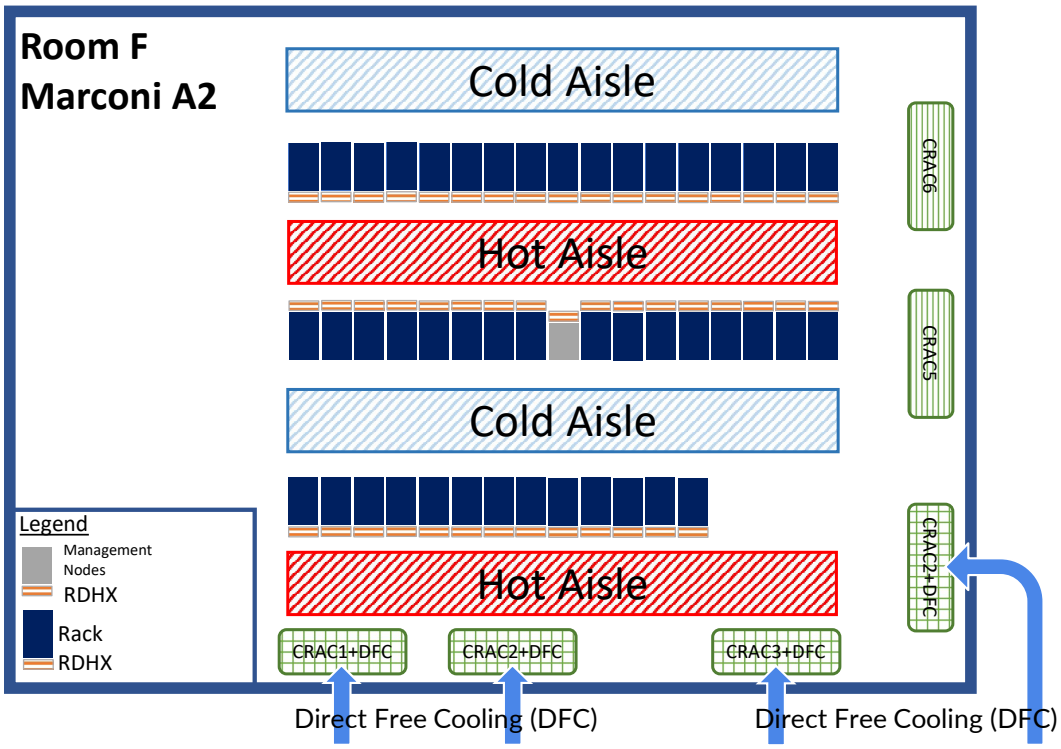
This thesis studies the Marconi A2 cluster for the entire 2019, and the study completely employed the in-production Marconi A2 cluster's real monitoring dataset, and even anomalies (which are rare events) that used in this study are real physical failures during cluster's production, and synthetic data did not employ for this study.

Marconi A2 stopped production in January 2020, and the same room F with the same cooling facilities hosted a new Marconi 100 cluster from April 2020.

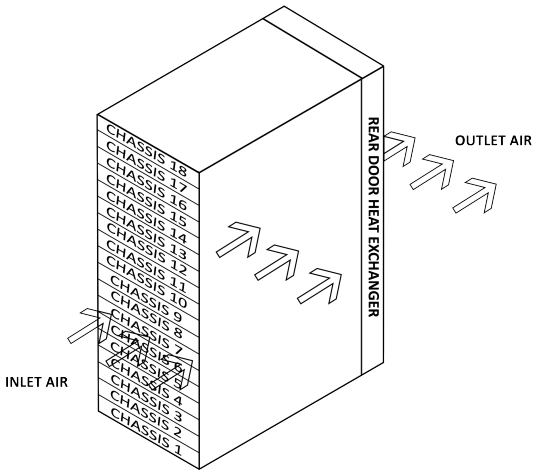
2.2.3 Room F - Marconi 100

After stopping Marconi A2 in January 2020, Room F hosts the Marconi 100 cluster, one of the world's most potent computing systems (18th most powerful computing system based on TOP500 list November of 2021 [1]). Marconi 100 cluster start production from April 2020. Figure 2.4 depicts the rack arrangement of Marconi 100 in Room F and cooling facilities. Marconi 100 is the accelerated cluster based on IBM Power9 architecture and Volta NVIDIA GPUs. Its computing capacity is about 32 PFlops. Room F (figure 2.4) contains 55 racks (49 computing), and each rack has 20 chassis, and each chassis host one computing node. Marconi 100 is composed of 980 nodes; each node has 2x16 cores IBM POWER9 (@3.1 GHz) processors and is empowered with 4 x NVIDIA Volta V100 GPU accelerators (16GB), RAM: 256 GB/node.

Room F contains six computer room air conditioning (CRAC) units that support the two cold aisles. Four of these CRAC units have the Direct Free Cooling system (DFC). All racks are equipped with RDHX, and RDHX of racks are in the hot aisle.



(a) Marconi A2 (KNL) Room F in CINECA Datacenter.



(b) Schematic of One Rack of Marconi A2.

Figure 2.3: Racks Arrangements of Marconi A2 (KNL) Room F in CINECA Datacenter.

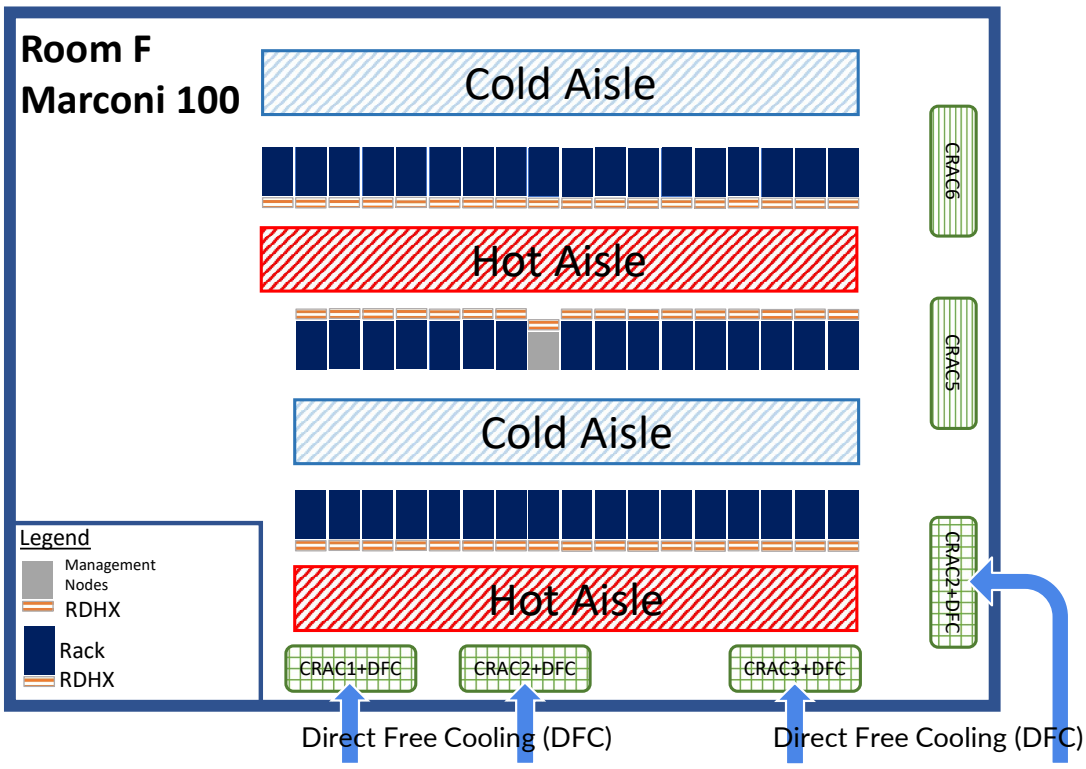


Figure 2.4: CINECA HPC Room F Marconi 100 Cluster.

The CINECA datacenter features a holistic monitoring framework, namely ExaMon (more detail in section 2.3), which aggregates a wide set of telemetry data [7]. For each node and its associated components, such as voltage regulators and fans, the Intelligent Platform Management Interface (IPMI) provides remote telemetry access to the built-in sensors [13]. The ExaMon monitoring system collects sensor data with the IPMI interface with 20 seconds sampling rate [7]. From April 2021, ExaMon, in addition to nodes metrics, starts to collect important metrics of HPC room facilities (CRAC units, RDHX, and Modbus). ExaMon monitored data is stored in its internal KairosDB database as time traces and remotely accessible through RESTfull APIs [7].

Although Marconi 100 has been available from April 2020, the monitoring system implementation for computing systems and HPC room facilities was completed in April 2021. This thesis studies the Marconi 100 cluster for 4 months of the year 2021 (2021-04-08 to 2021-08-18), and the study entirely employed in-production Marconi 100 cluster's real monitoring dataset, and even anomalies (which are rare events) that used in this study are real physical failures during cluster's production, and synthetic data did not employ for this study.

2.2.4 Room N

Figure 2.5 depicts the rack arrangement and cooling facilities of CINECA HPC room N. Room N comprises three main clusters: the Marconi A1 partition, the Marconi A3 partition, and the Galileo cluster. Marconi A1, a preliminary system, was in production from June 2016, based on Intel® Xeon® processor E5-2600 v4 product family (Broadwell) with the computational power of 1Pflop/s. Marconi A1 was closed in September 2018. Marconi is the Tier-0 system is based on the LENOVO NeXtScale platform. In August 2017, a Marconi A3 partition was added, based on Intel Xeon 8160 (SkyLake). 1.512 nodes at first, followed by about 800 more a few months later(Peak Performance: 8 PFlop/s) [14]. Galileo: Starting from January 2018, Galileo has been reconfigured with Intel Xeon E5-2697 v4 (Broadwell) nodes, inherited from the Marconi system. Starting from March 2021 was gradually turned off to give space to more performant Infrastructure Galileo100 [15].

Room N is cooled by the Direct Expansion (DX) Air-conditioning system with 14 CRAC units. Five of these CRAC units support the Direct Free Cooling (DFC) system, which is referred to by the CRAC+DFC in this thesis. The DFC system is designed to reduce energy dissipation and improve the carbon footprint by utilizing the external cold air for cooling the room. In this case, the DFC system starts to work when the outdoor temperature is lower than 18^oC. Without the DFC system, the CRAC units work in standard air recirculation mode with refrigeration based cooling. By combining compressors with the DFC system, we can reduce the compressor's operation.

There are two cages on the clusters of Galileo and Marconi A1 to shield the cold part of these two clusters. Also, there is a water cooling system for RDHX on cluster Marconi A3.

2.3 The ExaMon Framework

To enable datacenter automation, it is essential to collect and analyze data from different sets of sensors. This section gives a high-level description of the monitoring infrastructure and its main components, which integrates all the different heterogeneous sensors sources.

2.3.1 System Overview

The monitoring framework is composed of several components. From Figure 2.6, we can distinguish four main layers. Starting from the bottom:

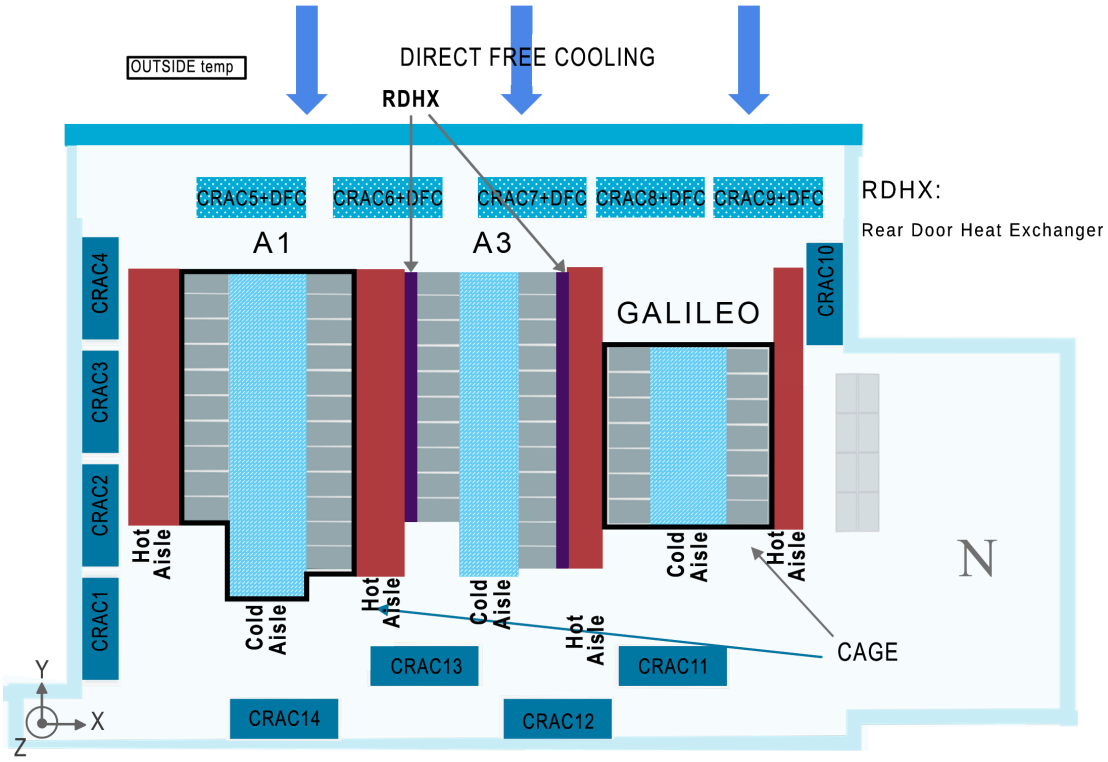


Figure 2.5: CINECA Room N

Data Collection

ExaMon collects different kinds of data: physical data measured with sensors, workload information obtained from the job dispatcher, and software information collected from software probes. These are the low-level components having the task of reading the data from several sensors scattered across the system and deliver them, in a standardized format, to the upper layer of the stack. These software components are composed of two main objects, the MQTT API and the Sensor API object. The former implements the MQTT protocol functions, and it is the same among all the collectors, while the latter implements the custom sensor functions related to the data sampling and is unique for each kind of collector. Considering the specific sensor API object, we can distinguish collectors that have direct access to hardware resources like PMU, IPMI, accelerators, sensor nodes, and collectors that sample data from other applications as batch schedulers (PBS and Slurm) and switchboards Modbus collectors.

The second typology of data regards the jobs running in the system and its workload. In order to gather this data, we need to extend the job scheduler by adding a software component that collects the information and sends it as an

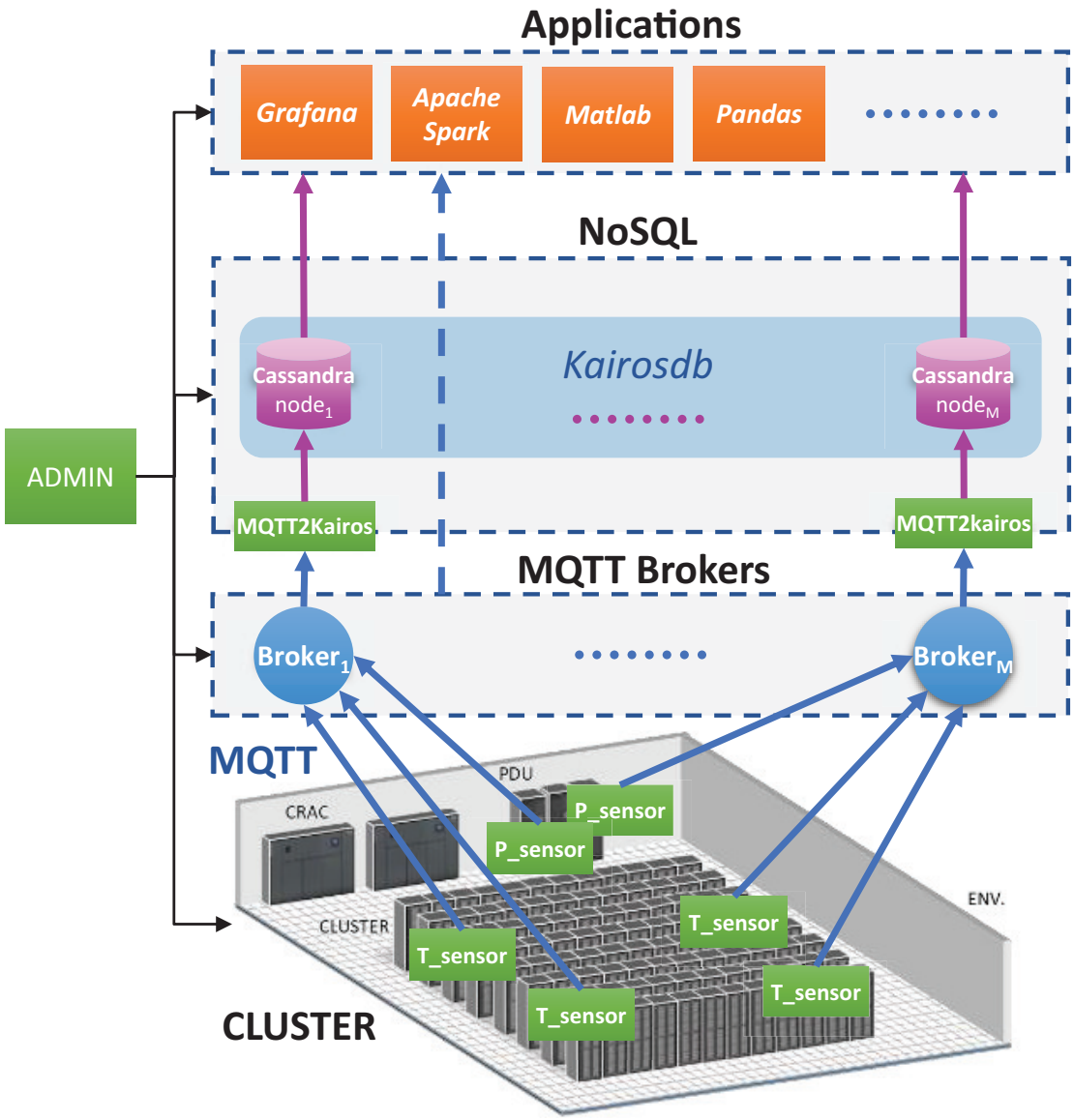


Figure 2.6: Monitoring Framework.

MQTT message to the upper layers of the framework. Current state-of-the-art schedulers usually expose a set of APIs that developers use to add custom functions and behaviors [16, 17].

Communication Layer

The framework is built around the MQTT protocol. It implements the "publish-subscribe" messaging pattern and requires three different agents to work: (i) The "publisher", having the role of sending data on a specific "topic". (ii) The "subscriber", that needs certain data, so it subscribes to the appropriate topic. (iii) The "broker", that has the functions of (a) receiving data from publishers, (b) making topics available to subscribers, (c) delivering data to subscribers. The basic MQTT communication mechanism is as follows. When a publisher agent sends some data having a certain topic as a protocol parameter, the topic is created and available at the broker. Any subscriber to that topic will receive the associated data as soon as it is available to the broker. In this scenario, collector agents have the role of "publishers".

Storage Layer

The monitoring framework provides a mechanism to store metrics mainly for visualization and analysis of historical data. We use a distributed and scalable time-series database (KairosDB) that is built on top of a NoSQL database (Apache Cassandra) as a back-end. A specific MQTT subscriber (MQTT2Kairos) is implemented to provide a bridge between the MQTT protocol and the KairosDB data insertion mechanism. The bridge leverages the particular MQTT topics structure of the monitoring framework to automatically form the KairosDB insertion statement. This gives a twofold advantage: first, it lowers the computational overhead of the bridge since it is reduced to a string parsing operation per message; and secondly, it makes it easy to form the database query starting only from the knowledge of the matching MQTT topic.

Applications Layer

The data gathered by the monitoring framework can serve multiple purposes, as presented in the application layer. For example, machine learning techniques can be applied to extract predictive models or devise online fault detection mechanisms. Another important application is real-time visualization using web-based tools - a powerful instrument for both facility administrators and system users.

2.3.2 Collector Measurements Format

At the end of the sampling stage, each collector delivers each metric to the MQTT broker under a hierarchical topic structure:

- for the per-node metrics it is:
`org/<organization name>/cluster/<cluster name>/node/<node name>/plugin/<plugin name>/chnl/data/<metric name>`
- for the case of per-cluster metrics, it is:
`org/<organization name>/cluster/<cluster name>/plugin/<plugin name>/chnl/data/switchboard/<switchboard ID>/<metric name>`
- for the case of per-room metrics, it is:
`org/<organization name>/cluster/<cluster name>/room/<room name>/plugin/<plugin name>/chnl/data/sensorID/<sensor node ID>/<metric name>`.

The payload of the MQTT message for all the cases is: `<value>;<timestamp>`. The MQTT broker is a daemon process that runs on non-computing nodes as login nodes to minimize infrastructure intrusiveness. Now that the data is available at the broker, as a temporal sequence of samples, it can be ingested and processed by the computing engine.

Switchboards

The *switchboard_pub* collector is devoted to the sampling of switchboards meters data normally available over the Modbus interface. It executes on the management nodes of the cluster where the site-level data collection software is running[1]. A software daemon periodically (every hour) reads the average of the Modbus meters collected by SiteScan and publishes it to the MQTT broker through MQTT messages.

Wireless Sensor Network Nodes

The LoRaWAN network structure relies on a commercial product, the MultiConnect® ConduitTM [18], that manages the gateway and the server layer. It is a highly configurable, manageable, and scalable gateway for IoT applications, supporting a wide-span of programming tools, such as Node-Red and nodejs. Thanks to a browser-based editor and smart wiring tools, Node-Red is a perfect program for our needs, forwarding the collected data from LoRaWAN end-devices to the MQTT broker through MQTT messages.

[1] Liebert SiteScan

2.4 Summary

This chapter provided an introduction and preliminary definitions related to the thesis, a brief description of the HPC system and HPC room facilities. Some technical characteristics of Marconi A1, Marconi A2, Marconi A3, Galileo, and Marconi 100, which are located in CINECA HPC rooms N and F, were illustrated. These HPC clusters may consume megawatts of electrical power, which is entirely converted into heat; therefore, rooms are equipped with sophisticated cooling systems. It explained CRAC units (+DFC) and water cooling systems (RDHX). CINECA is equipped with ExaMon (Exascale Monitoring), one of the state-of-the-art datacenter monitoring systems. This chapter had a brief overview of ExaMon, which is composed of Data Collection, Communication Layer, Storage Layer, and Application Layer [7].

Chapter 3

Thermal and Power Characteristic of the HPC Room

3.1 Overview

High-performance computing (HPC) systems are large and complex industrial plants [19] which are gaining importance in today's society and industry [20]. Recent reports quantify the Return on Investment (ROI) produced by applying HPC in an industrial environment: in Europe, each Euro invested in HPC generates, on average, 867€ of increased revenues and 69€ in profit, while in the US, a single dollar spent in HPC generates, on average, 43$ of profit [20].

HPC systems are hosted in computing rooms, each containing multiple racks that pack tens/hundreds of computing nodes. Each computing node is composed of multiple computing elements (CPUs/GPUs) based on multi/many-core processors. The power consumption of these installations ranges from few to tens of MWatts. Additional power is required to remove the heat generated by the active electronics. Performance evolution of computing systems is faced by the end of Dennard's scaling and the so-called "Thermal and Power Wall" [21]. Indeed, the power density of computing devices has increased across generations: higher power density increases silicon temperature, which in turn increases cooling costs, complexity, and/or jeopardizes performance.

Summit [4], which is today one of the most powerful supercomputers worldwide, consumes 11 MWatts for the computation and an additional 1.32 MWatts for cooling. To achieving this cooling efficiency, Summit adopts a sophisticated computing node design and hot water cooling solution [22]. A study shows that even highly tuned Google datacenters pay, on average, an additional 12% of power consumption for power delivery and cooling dissipation [23].

Traditional cooling methods, based on Computer Room Air Conditioners

(CRAC) or Computer Room Air Handlers (CRAH), have been enhanced with free-cooling mode, i.e., the capability to exploit the outside air, using only the Air Conditioner (AC) blowers to circulate it in the room [24–27]. Moreover, cooling energy can be significantly reduced if hot water cooling is used to remove heat [22, 28, 29]. In both these cases, a coolant hotter than the traditional chilled coolant is used to remove the heat, often leading to a higher silicon temperature in the computing units [19, 30]. Rear Door Heat Exchangers (RDHX) are used to augment the computing density in air-cooled computing rooms.

Monitoring the temperature of the coolants (Air and Liquid) in the computing room is necessary to capture the effects of the power dissipated by the computing machines, optimize the cooling while preventing thermal hazard [31, 32]. Temperature monitoring devices, in traditional commercial solutions, use wired sensors with few measurement points, due to the high installation cost [33]. In this field, Wireless Sensor Networks (WSNs) are optimal for scattered and ubiquitous deployments; devices can be placed freely in mobile objects, and also in critical or hard-to-reach areas, to measure different parameters, such as temperature, power consumption, and humidity [34–36].

This chapter characterizes the two main computing rooms in the CINECA datacentre leveraging two completely different approaches. For the CINECA HPC Room N, which hosts Marconi A1, Marconi A3, and Galileo, we deployed a WSN together switchboards power consumption. For the CINECA HPC Room F, which hosts Marconi A2 (Marconi A2 closed in January 2020 and was replaced with Marconi 100) and Marconi 100, we leverage fine-grain metrics collected directly from the nodes. The characterization analysis and methodology have been adapted to the different nature of measurements used (time and spatial granularity). We refer later to Wireless sensor network-based characterization (applied to Room N) and Big data-based characterization.

3.1.1 Wireless Sensor Network-based Characterization - CINECA HPC Room N

In section 3.3, we propose a distributed system designed to measure the temperature evolution in a datacenter, aiming to improve cooling efficiency. Each sensor embeds a LoRa (Long Range) transceiver [37]. LoRa is a wide-area IoT communication technology, developed by Semtech, with unique spread spectrum modulation. We described a novel deployment of a distributed temperature monitoring system in a Tier0 datacenter, hosting three HPC clusters, using the LoRa technology. The detail of the WSN system is out of the scope of this thesis and just the data collected from this monitoring system has been analyzed in this chapter.

In this study, the relation between the power dissipated by the computing

clusters during production conditions, their spatial position, and the monitored temperature in a real Tier0 HPC room have been studied. The impact of the sensor's location, pre-processing strategy, and data collection rate has been analyzed. Experiments were run in CINECA room N, the Tier0 supercomputing center for scientific research in Italy, which is ranked 18th (@2018) in the list of the most powerful supercomputer worldwide and features hybrid and free-cooling technologies. [2]. Based on the analysis of the more than 7 million data samples of the room's temperature and the power consumption which are collected in ExaMon; has been shown that the different cooling technologies used in the datacenter room create heterogeneous thermal zones, and horizontal spatial proximity does not imply thermal coupling; also, the bottom and top of the racks are thermally decoupled. Although the temperature measured with the sensors is expected to be directly related to power consumption, some sensors have an inverse correlation with power consumption, which means that these parts are more affected by CRAC units than the direct effects of power consumption. We demonstrate that the data collection rate and transmission rates can be reduced by two orders of magnitude w.r.t. an initial rate of 120 samples per hour.

3.1.2 Big Data-based Characterization - CINECA HPC Room F

In section 3.4, we characterized the temperature distribution of a Tier0 datacenter hosting the Marconi supercomputer [2, 5], which is ranked 21st (JUNE 2019) in the list of the most powerful supercomputer worldwide and features hybrid and free-cooling technologies. To carry out the analysis, we have collected the entire Marconi node's telemetry data for a month of activity. During the selected period (01.06.2019 - 01.07.2019), the ambient temperature has ranged from 12^oC to 38^oC. Our analysis shows that:

- The inlet temperature of the nodes increases vertically. With an average difference of 6.5^oC from the top and bottom nodes. Moreover, the bottom nodes face a higher variability of the inlet temperature than the top nodes in the rack as an effect of a stronger dependency of their inlet air with the CRAC outlet temperature. This is less strong with top nodes in the rack due to a stronger dependency of their inlet air from heat re-circulation.
- The inlet temperature significantly changes in the floorplan. We measured up to 10.8^oC difference for the monthly average chassis temperature for chassis at the same height in the racks. Interestingly the monthly average hotspot position in the floorplan is correlated with the chassis height.
- In the observed period, the datacenter faced a thermal hazard which has compromised the liquid cooling capacity of the room (used by the RDHX). We carefully

analyze room temperature during this rare but extremely critical event. Our measurement shows that the effect of the thermal emergency caused an increase in the average temperature of the computing nodes with a modification of the hotspot location.

Results of the study show that the inlet temperature in a datacenter is heterogeneous and significant patterns are stable for long periods and visible on the monthly average. If accurately modeled, this information can be used to improve job scheduling and improve the datacenter's cooling efficiency.

The following of this chapter is organized as follow: after the state-of-the-art, the thermal and power characteristic of the CINECA HPC room N, which hosts the clusters A1, A3, Gailelo, is described in section 3.3 , then with the almost similar study but with different methods, the thermal and power characteristic of the CINECA HPC room F, which hosts the cluster Marconi A2 in normal and thermal emergency conditions is investigated in section 3.4 and finally a summary of the chapter in section 3.5.

3.2 State of the Art

Several works in literature have analyzed the impact of heat dissipation in datacenter components. The first set of works focus on the chip-level thermal effects and show that at chip-level exists hotspots and significant thermal gradients which can be exploited for improving core's performance and energy-efficiency [19, 38–41]. Druzhinin et al. have studied the impact of the coolant temperature increase in a datacenter blade with hot water cooling. The authors show that an increment of 40^oC in the coolant causes 20% of additional leakage power and a consequent decrease in the performance of 0.5% [30]. The second set of works focus on the machine level [29, 42, 43]. These works characterize the effect of performance variability between nominally equal computing nodes. Marathe et al. [43] show that in power-constrained computing nodes, the hardware control logic turns the process variation effects into a performance and core's frequency variation. This can lead to significant application time-to-solution overheads in parallel applications.

Thermal management of datacenters has been studied in depth in the last few years [44], exploring diverse strategies and approaches to reduce the cooling infrastructure power consumption by improving facility efficiency. In this context, many solutions have been presented such as: automatic cooling mode selection which optimizes overall power, under defined quality of service and thermal requirements [33]. The works proposed in [45] and [46] aim to fine-tune the speed of the rack fans, whereas in [47], the CRACs internal temperatures are used as the reference to minimize the resources used to dissipate the heat. In [48], the effort is put into selecting the number of active subfloor tiles and blowers speed to optimize

the Computer Room Air Handlers (CRAHs) cooling system.

The authors of paper [49] introduced two virtual sensors (volumetric airflow sensor and outlet temperature sensor) to control the volume of cold airflow generated by the CRAC units and needed to cool the compute nodes. They use the aggregated volumetric airflow of compute nodes to control the Air Conditioning Unit (ACU). In their simulations and they estimated an annual PUE reduction (Power Usage Effectiveness) from 1.92 to 1.6. Authors of [50] focus on Dynamic Thermal Management (DTM) for placing workloads in the datacenter to reach an uniform temperature distribution and achieve more efficient cooling.

Authors of [26] study the intelligent placement of liquid-cooled servers and intelligent coordination of different cooling techniques (air cooling, liquid cooling, and free air cooling), considering the dynamic workload allocation to minimize the cooling and server power of a datacenter. In [51], authors focus on air management to improving cooling energy efficiency in the datacenter.

Authors of [52] investigate the temperature management in datacenters by considering the reliability of the storage subsystem, the memory subsystem, and compute node reliability as a whole. They propose temperature management strategies for saving energy while limiting adverse effects on system reliability and performance.

Authors in [53] proposed a dynamic thermal model that can be used as a basis for model predictive control algorithms. In [54], the authors study thermal-benchmarking techniques to extract the servers' thermal profile and thermal statistics that can be used in thermal efficient datacenter management. In [55], the authors compare steady and dynamic models to illustrate the computational interactions and thermal relationships among the datacenter components. Moreover, the authors define an energy minimization problem, which is solved by a two-time-scale control method.

In [56], the authors propose a real-time monitoring system to optimize the cooling system's energy consumption by minimizing the number of active CRACs. They suggest a four-layer technology for monitoring, which is a cloud-based application for data collection, analysis, visualization, and reporting. With the proposed approach, authors are capable of reducing the number of underutilized CRACs and increases the number of turned off CRACs (from 2 to 7 on a total of 15 CRACs), keeping the same average return temperature. In [57], the authors present an optimal control policy for hybrid systems featuring free, liquid, and air cooling. The policy features a predictive model of the cooling system based on environmental, room, and IT temperature measurements. It is important to note that all models based on the predictive approach implicitly assume the availability of high-quality temperature data, both off-line for training and online for driving control decisions.

Focusing on the room temperature monitoring infrastructure, in [58], the authors

proposes a green cooling system; the management model collects information from a WSN that utilizes temperature sensors to control the ventilation system and the air conditioning. The WSN is based on the ZigBee protocol and includes the actuators. This approach generates a highly sophisticated network and a complex deployment, with 10 boards scattered in only 20 m^2. A novel WSN designed for datacenter facilities monitoring is presented in [59]. It describes a system based on Zigbee sensor nodes supplied by 2000 mAh batteries. The deployment consists of 10 devices in a 30 m^2 area. Since the network is configured as a very dense mesh, several boards operate as a router, and the communication from the farthest node needs up to 4 hops. While [58, 59] demonstrate the key functionality of temperature monitoring via a WSN, the power consumption of the sensor nodes is hardly compatible with battery operation. As a reference, a single sensor in [59] drew from the battery 73 mA on average, which is two orders of magnitude higher than our LoRa solution. Furthermore, the range of connectivity of the nodes is limited, requiring multihop networks with several continuously powered routers, which greatly complicate the deployment.

While all the previously mentioned studies highlight and characterize the side-effects of inlet temperature, performance variation, energy efficiency and parallel job performance in a datacenter, none of them has characterized the temperature variation in a datacenter's room. Therefore, we characterize the temperature distribution of a Tier0 datacenter hosting the Marconi supercomputer [5, 60], which is ranked 18th (@2018) in the list of the most powerful supercomputer worldwide and features hybrid and free-cooling technologies.

3.3 Thermal and Power Characteristic of the CINECA HPC Room N

3.3.1 Methodology

Sensor Node: The sensor node (Figure 3.1) is a low power and versatile circuit, which includes an internal temperature and humidity sensor in parallel with an efficient (up to 90%) DC converter. Moreover, multiples types of transducers, both analog and digital, can be connected to its expansion port.

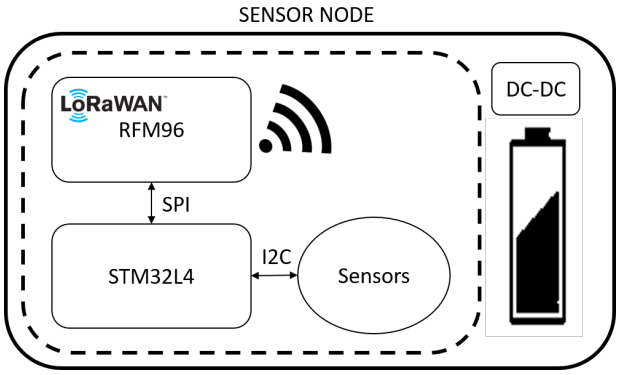


Figure 3.1: Block Diagram of the Sensor Node.

WSN Setup and Description

A LoRaWAN WSN test with 14 sensor nodes was carried out to model the room temperature. The goal was to verify in a realistic operating condition the improvements using LoRaWAN instead of multi-hop protocols in terms of energy consumption and transmission reliability. The radios packet consists of 29 bytes of payload and 6 symbols of the preamble. Each end-node was configured to send a sensor data packet every 30 seconds. The final network experimental deployment was carried out with 14 sensor nodes, arranged in critical points of the datacenter's high power density room (Fig. 3.7 - Room N). The room is composed of three main clusters, the Marconi A1 partition, the Marconi A3 partition, and the GALILEO cluster. The Marconi supercomputer (A1 and A3) is composed of 3216 computing nodes, while the Galileo supercomputer is composed of 400 nodes. We positioned all the devices in hallways between racks (Fig. 3.2 (a)), close to the CRAC output and under the floor (Fig. 3.2 (b)), besides, some sensor was placed into full metallic air conditioning pipes and structure. Figure 3.2 shows two pictures of the WSN deployment.

Abbreviation	Definition
CRAC	Computer Room Air Conditioning unit
CRAC+DFC	CRAC unit with Direct Free Cooling
TW	Time Window
TWN	Time Window Number
TDCTW	Total Data Collection Time Window
LTWN	Last Time Window Number
TWG	Time Window Group
Temp(SensorID,TWN)	Data set of temperatures that was collected by *sensorID* in *TWN*.
Pow(lineID,TWN)	Data set of power consumption that was collected by meter from lineID in *TWN*.
CC(x,y)	Correlation Coefficient of x and y
Pair	One Sensor and One Power Line
SST	Statistical Significance Test
SSTMask	Statistical Significance Test Mask
meanCC	Mean Correlation Coefficient

Table 3.1: Table of Abbreviations and Definitions.

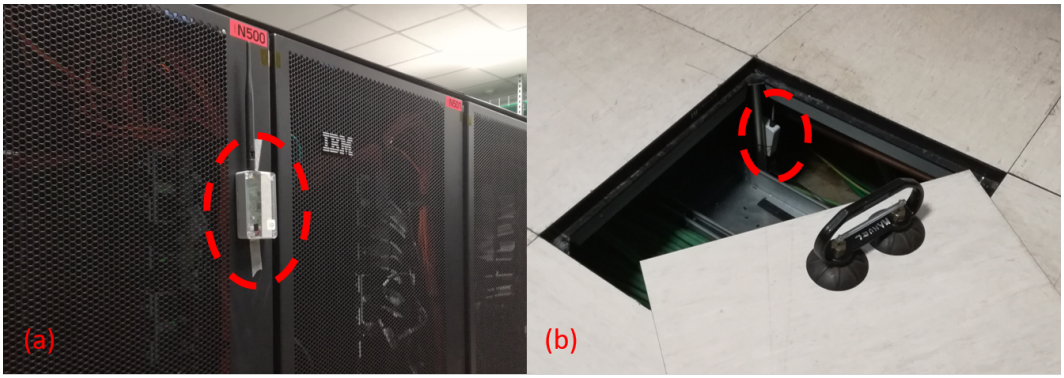


Figure 3.2: Sensor Node Deployment. (a) a Hallway Positioning Is Proposed, (b) the Sensor Node Is Under the Datacenter Floor.

Data Collection

Six main electrical distribution lines ("A1 a", "A1 b", "A3 a", "A3 b", "Galileo a", "Galileo b") feed the three main computing clusters (Galileo, Marconi A1, and A3). Each rack of clusters is bi-powered by branch "a" and branch "b". Servers of cluster Marconi A1, and A3, are single-fed, and inside each rack half of the servers have powered by branch "a" and another half by "b".
The switchboards meters monitor the power consumption of each cluster's power rail/branch. They are sampled by Lebert SiteScanner and periodically collected on a per hour average in the ExaMon database. The room temperature values monitored by the wireless sensor network are also available in the ExaMon database but at a higher sampling rate (every 30 seconds). Since the Galileo cluster production

started from 12-03-2018, we focused our analysis on the last two months (starting from the 20-03-2018 until the 17-5-2018) of our WSN installation to cover a period with minor infrastructure works in the monitored room. We extract data from ExaMon and process it by Python scripts. Figure 3.3 illustrates the electrical power consumption of three clusters. The x-axis reports the different days of study, and the left y-axis the power consumption of clusters by each of the branches (red line "a", blue line "b", and the yellow line is the absolute difference between "a" and "b". In the studied period, the A3 cluster, with average power dissipation 483.47 KW, consumed more than the two other clusters. The A1 cluster, on average, consumed 434.30 KW while the Galileo consumed, on average, 68 KW. The heat produced by the Galileo cluster is lower than the other two clusters (Table 3.2).
In the HPC room N, there are thousands of on-board sensors on the nodes. Although these sensors collect precise information about the proximity of the HPC nodes, they are too many to monitor at high speed, and they do not cover the different areas of the room, such as the subfloor, top of the racks, and CRAC units. In this study, we leveraged the proposed WSN.
Room N is cooled by the Direct Expansion (DX) Air-conditioning system with 14 CRAC units. In DX Air-conditioning, the air used for cooling the room is directly passed over the cooling coil. Five of these CRAC units support the Direct Free Cooling (DFC) system that is referred by the CRAC+DFC in this thesis. DFC system is designed to reduce energy dissipation and improve the carbon footprint by utilizing the external cold air for cooling the room. In this case, the DFC system starts to work when the outdoor temperature is lower than 18^oC. Without the DFC system, the CRAC units work in standard air recirculation mode with refrigeration based cooling. By combining compressors with the DFC system, we can reduce the compressor's operation. For instance, the room was cooled by the DFC system for 5064 hours during the year 2018. There are two cages on the clusters of Galileo and A1 to shield the cold part of these two clusters. Generally, the airflow moves under the raised floor and gets to the loaded areas; then, above the raised floor, the hot air returns to the CRAC units. Also, there is a water cooling system for RDHX on cluster A3, with the chiller loop (cold loop) temperature around 12^oC to 17^oC, and RDHX loop (hot loop) temperature around 23^oC to 30^oC. The RDHX device is placed in front of the hot outlet airflow of the compute node. During operation, the compute node's hot airflow is forced through the RDHX device by the compute node fans and exchanges heat from the hot air to circulating water from a chiller. Thus, the compute node outlet air temperature reduces before its discharge into the datacenter.

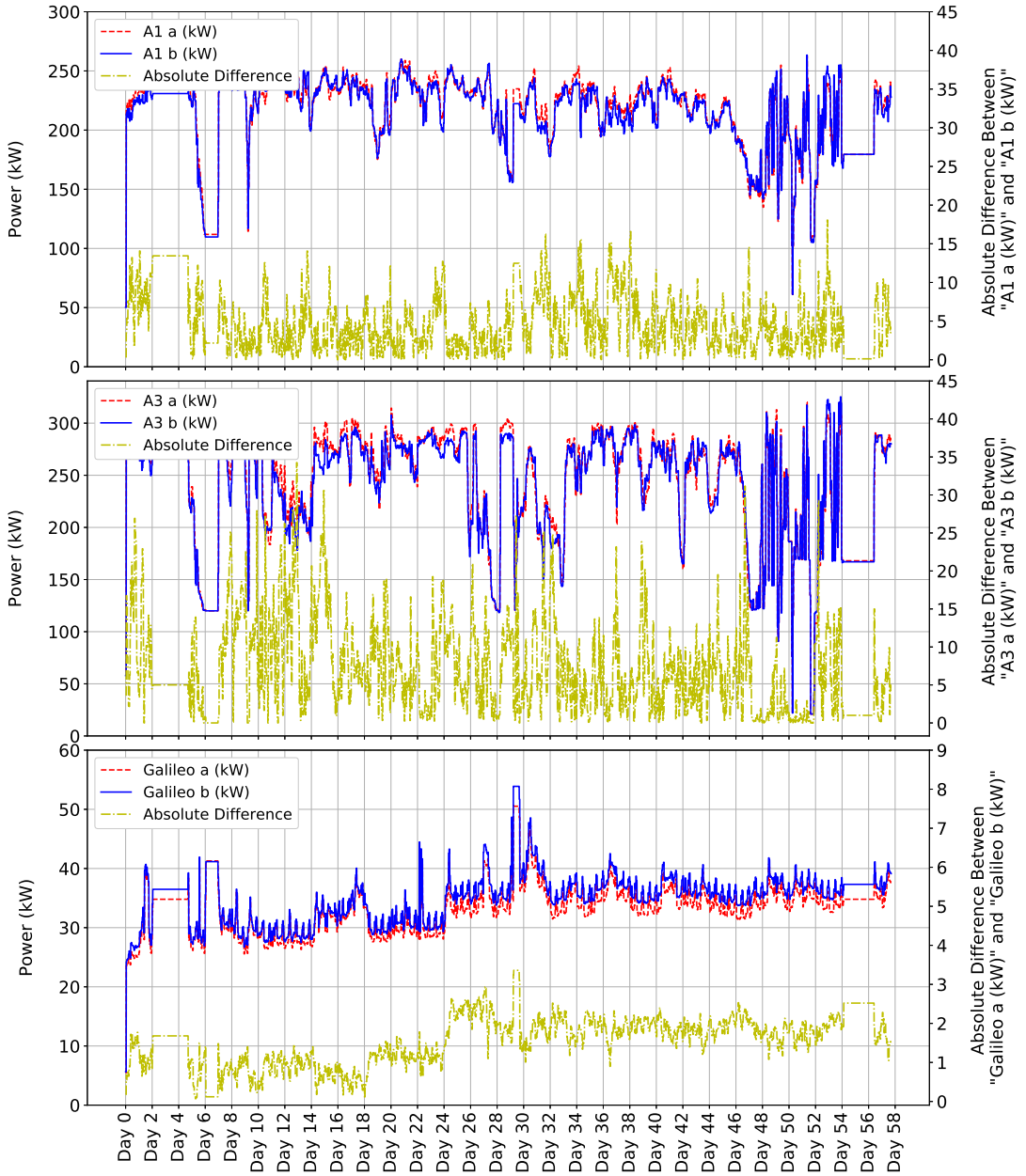


Figure 3.3: Electrical Power Consumption of Clusters (Branch a and b) and Absolute Difference Between Two Branches.

	A1 (KW)	A3 (KW)	Galileo (KW)
mean	434.30	483.47	68.00
std	63.61	107.98	7.48
min	121.06	42.00	50.97
max	526.22	645.42	95.83

Table 3.2: Power Consumption of Different Clusters of Room N.

Data Validation

The collected data were validated with statistical hypothesis testing [61, 62]. The statistical hypothesis testing checks if the observed result is more unusual than the result that can be produced by chance. As a null hypothesis, we assume that the results happened by chance; in other words, a null hypothesis states that no statistical significance exists in a set of a given study. If the occurrence of the given null hypothesis is unlikely, a result has statistical significance. The $p - value$ shows the probability of the occurrence of results, given a chance model (null hypothesis) as unusual as the observed results. The study can define the significance level by α; it is the probability of the study accepting the null hypothesis. The significance level for a study typically is set to 5% or much lower, depending on the field of study. The study is statistically significant, when $p - value < \alpha$ [61, 62]. For all Pearson's correlation coefficients calculated in this chapter , the statistically significant test with the significance level of $\alpha = 0.05$ has been computed the results (correlation coefficients) which could not pass the significant test are omitted from the study.

3.3.2 Experimental Results

In this section, the methodology for thermal and power study of the real in-production HPC system are represented, data are collected utilizing the combination of WSN and one of the stat-of-the-art data collection systems (ExaMon) for spatial monitoring of physical parameters (i.e., temperature) of a datacenter room in a real Tier 0 datacenter. In this context, (a) The average temperature and Pearson's correlation coefficient are computed to explain the impact of the sensors' placement in sensing the room's cooling system heterogeneity. (b) The time-series dataset is split into smaller periods, i.e., different subsets that have varying durations from one day to two weeks are created to examine if it is more relevant to consider the dataset as a unique dataframe; alternatively, is it better to split in chunks due to the several not monitored events (maintenance, racks substitution, meteorological changes) that may have potentially modified the room temperature? (c) The effect of time granularity of collected data (that is critical in optimizing the battery lifetime of sensor nodes) is analyzed. (d) On the room temperature map, the correlation of different sensors (which, considering the location of the sensors, they

can be representative of sources of heat and cold) is reported.

Thermal Study

This section analyzes the correlation between the different WSN sensors and their placement in the datacenter room. The experimental setting consists of 14 wireless sensors distributed to cover each cold and hot aisles, subfloor, cages, and CRAC. With this in mind, it could expect that the spatial proximity of the sensors correlates with the sensor's data. However, as we will discuss in this section, forced airflow and physical barriers create heterogeneity in the sensors' readings. The statistical characteristics of the measurements: average, standard deviation, and cross-correlation, are analyzed. These parameters are used to cluster the sensors according to their measurements similarity and compare these results with the spatial location of the sensors in the datacenter room and floorplan. The monitored datacenter takes advantage of free-cooling; thus, its thermal system depends on the computational load and the outdoor ambient temperature fluctuation and phases.

The figure 3.4 reports the average temperature for each sensor during the study period. The x-axis is the sensor's name, the left y-axis shows the mean temperature, and the right y-axis shows the standard deviation. We can divide the sensors into two groups based on their average temperature (**hot-sensors** {s5, s11, s12, s6, s3, s7, s8} and **cold-sensors** {s10, s14, s13, s2, s4, s1, s9 }). The cold zones of the room include the subfloor, inside of the cages, and cold aisle with an active subfloor. The hot zones of the room consist of the top of the room and hot aisles. In the center of the room, sensor s10, which is located in the subfloor under the cold aisle of cluster A3 with an average temperature of 13.45^oC is the coldest sensor. The sensor s8 in the same aisle but in the top of the cluster with 27.88^oC is the hottest sensor. Sensor s9 measurements are proximate to the average temperature of the room.

Figure 3.5 shows the normalized Euclidean distances (normalized on the maximum value) of sensors from all the CRAC units. The x-axis is the name of the sensors, and the y-axis is the normalized distance. Sensor s10 in the center of the room has the lowest distance from all the CRAC units, and s11, on average, is the farthest sensor from the CRAC units.

The following matrix in figure 3.6 shows Pearson's correlation coefficient between measured temperatures, which includes all the sensors and outdoor ambient temperature. Ambient temperature is represented in the matrix by "temp". All the data that are used for the study passed the statistical significance test.

By filtering out the correlation coefficient greater than or equal to 0.5 in matrix Figure 3.6, the correlation coefficient graph (Figure 3.7) is generated, which visualizes the correlation of measured temperatures by the different sensors. The green arrows illustrate the direct correlation, and the blue ones show the inverse

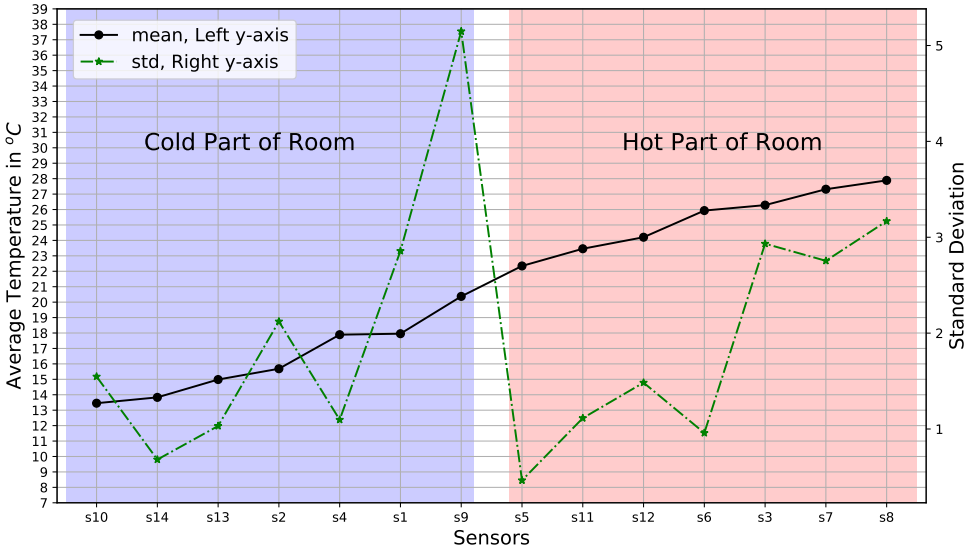


Figure 3.4: Average Temperature and Standard Deviation from 20-03-2018 to 17-05-2018.

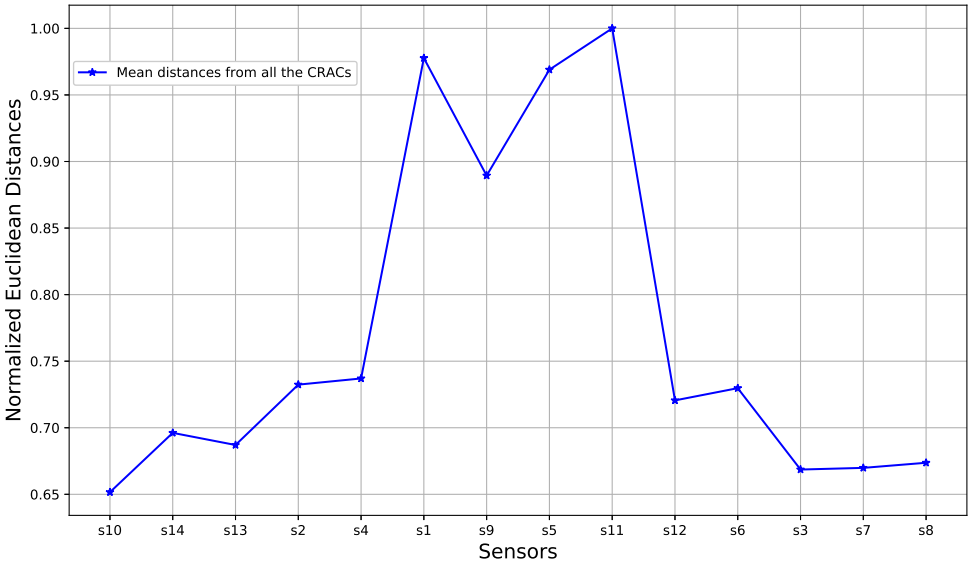


Figure 3.5: Normalized Euclidean Distances of Sensors From all the CRAC Units.

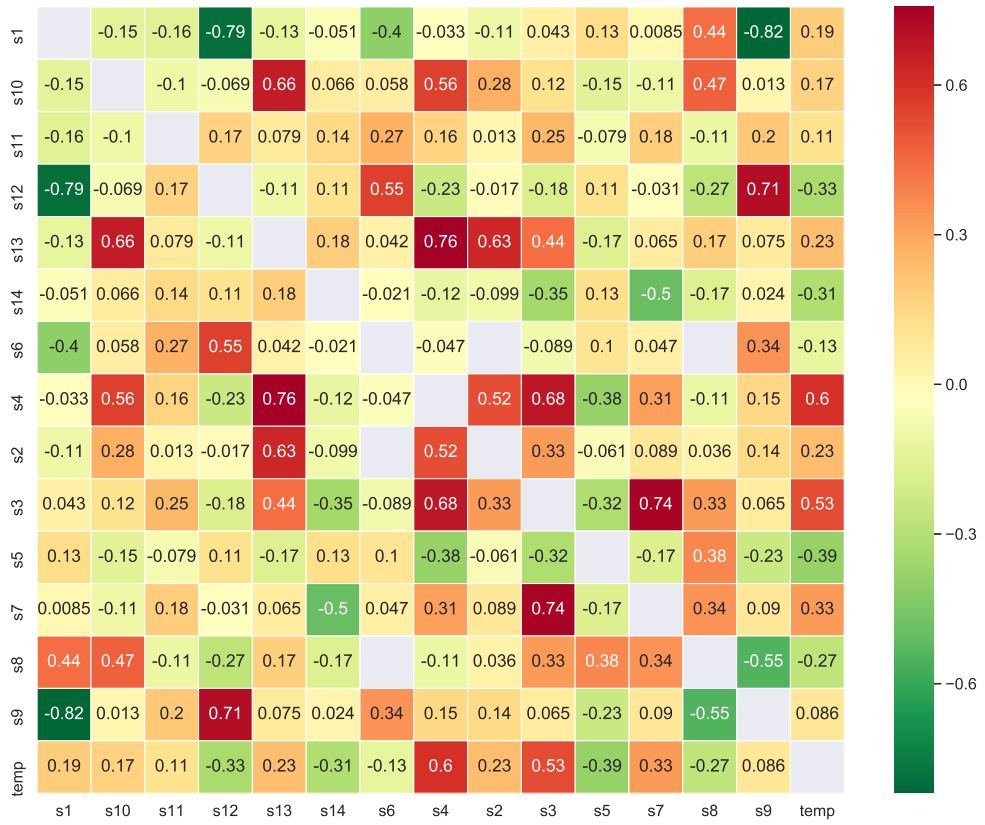


Figure 3.6: Pearson's Correlation Coefficient of Measured Temperatures.

correlation.

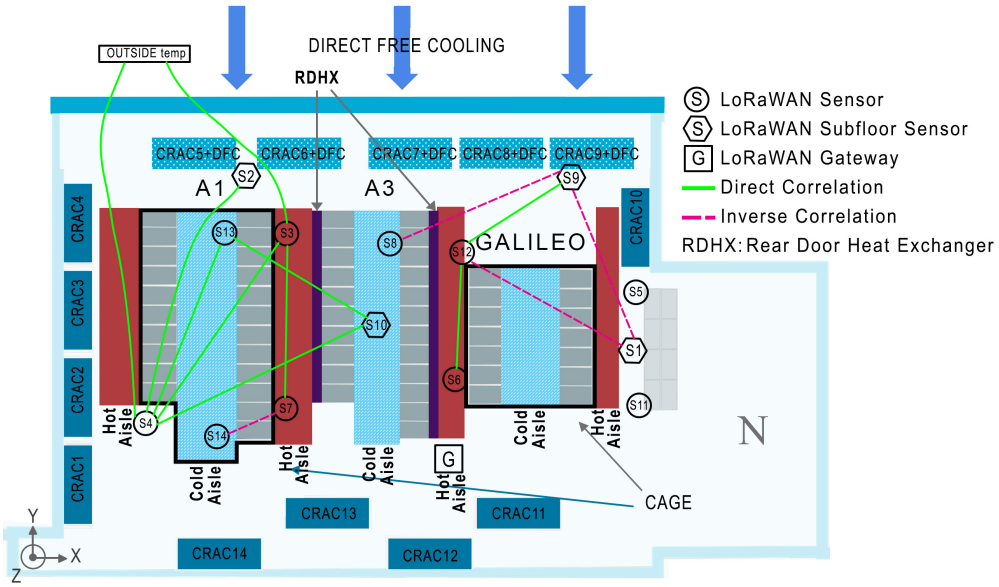


Figure 3.7: Datacenter Plot and Correlation Coefficient Graph of Sensors and Outdoor Ambient Temperature. Each Sensor Is Identified by Its Corresponding Number.

Figure 3.7 reports that there are four thermal zones in the room: in the subfloor, in the left and right parts of the room, and in the vertical direction. Subfloor sensors (s2, s9, s10, s1) do not correlate with each other except s1 and s9, which have an inverse correlation. The room's left side's sensors correlate more with each other than sensors of the right part of the room, and the same situation is valid for the sensors of the right part of the room, which are more correlated with each other. In the center of the room, the hottest sensor s8 and coldest s10 are in the same aisle but with different heights. These sensors do not correlate; therefore, there is a vertical thermal detachment.

Cold Sensors: generally, the subfloor, bottom of the cold aisle of clusters, and inside the cage are the cold zones of the HPC room. As evident in figure 3.4, sensors s2, s9, s1, s10, s14, s13, s4, which are located in specified parts of the room, measure on average colder temperatures than other sensors.
Sensors, s14, and s13 are in the same cold aisle inside the cage, but they do not strongly correlate with each other due to the following reasons. First, sensors s13 and s14 sense different sources of cold air. Sensor s13 is closer to the CRAC units with the DFC system (CRAC+DFC), while s14 is adjacent to the CRAC units without DFC. Second, the active subfloor creates multiple small air curtains. These air curtains create thermal barriers in the horizontal (Y-axis) direction.

Sensor s10 is located in the subfloor in the center of the room with the lowest normalized Euclidean distance (of 0.65) to all the CRAC units; therefore, it receives cold air from the different CRAC units. With an average temperature of 13.45^oC, it is the coldest sensor in the room. Sensor s10 correlates with s13 (inside the cage) and s4 (outside the cage). Sensors s2 and s13 are proximate to the same CRAC+DFC units (CRAC5+DFC, CRAC6+DFC). Sensor s2 in the subfloor, on average, measures 0.7^oC higher temperature than s13, which is inside the cage. This means that the *inside cage can be colder than the subfloor*. Sensor s13, which is inside the cage, correlates with s10. Sensor s10 is located in the coldest part of the room in the center and subfloor, receives the cold air of all CRAC units. The correlation of s13 and s10 means that the *inside of the cage receives the cold air of all CRAC units*. It is clear that sensor s2, which is close to the CRAC+DFC unit, should perceive a higher temperature than s13, which measures the temperature of the CRAC units with DFC and without DFC.

Sensor s4, on average, is located far from the heat sources. It is highly correlated with different sensors (s2, s13, s3, s10); all of these sensors are in the left part of the room. Sensor s9 approximately measures the average temperature of all sensors.

Hot Sensors: As mentioned before, the monitored room N hosts three clusters: Marconi A1, A3, and Galileo. Marconi A3 consumes on average 483.47 KW of electrical power, Marconi A1 of 434.30 KW, and Galileo of 68.00 KW during the study period. Sensors s5, s11, s12, s6, s3, s7, s8 are in the hot part of the room. Pairs of sensors (s3,s7), (s12,s6), and (s5,s11) have a comparable situation of the viewpoint of location in the hot aisles of clusters. The pair (s3,s7) is between A1 and A3 clusters that consume more electrical power than the Galileo, so they experience hotter temperatures than the pair (s12,s6), which is between the A3 and Galileo. On average, the pair (s5,s11) is colder than the other two pairs, as it is far from A1 and A3 and closer to Galileo. Sensors of each pair have cross-correlate except the s5, s11.

Sensors s8, s10 are in the cold aisle of A3 in different elevations. The cold aisle of the A3 cluster is equipped with an active subfloor, and it does not have a cage. Sensors s8 on top of racks and s10 in subfloor respectively perceive the highest and lowest average temperature of 27.9^oC and 13.45^oC. Therefore, thermal heterogeneity of the center of the room with, on average, 14.43^oC thermal variation is significant. Although these two sensors are close together, they do not correlate. Sensor s10 measures the cold air of the CRAC units; meanwhile, sensor s8 perceives cluster dissipated heat. Sensor s8 has an inverse correlation with the s9, which means that when the temperature of the s8 increases, the temperature of s9 decreases. Since s8 is affected by the dissipated heat, the CRAC units are activated when the dissipated heat at the top of cluster A3 increases. Activation of the CRAC units reduces the temperature of the subfloor where s9 is placed. This is

the cause of the negative correlation between s8 and s9.

Negative Correlation: The following pairs of sensors show a negative correlation: (s1,s9), (s1,s12), (s9, s8), and (s14, s7), i.e., the temperature of the one sensor in the pair increases, while the other pair experiences a reduction in the temperature. Sensor s1 is located in the subfloor, neither in front of the CRAC nor under the active subfloor. It measures the lateral air of one CRAC unit; it seems that this CRAC unit, which is close to the s1, has a different duty cycle than the CRAC units close to the s9, so they have an inverse correlation. Sensors s1 and s9 are located in the subfloor. Therefore, they perceive more the temperature of CRAC units. Sensors s12 and s8 are at the top of clusters, measuring the clusters' dissipated heat. Consequently, when the temperature of the room's hot part increases due to a rise in dissipated heat of clusters, CRAC units are activated and reduce the temperature of nearby and cold parts of the room. This creates an inverse correlation between the room's hot zones and some parts of the cold zones. The pair (s9, s8) that measures the temperature of the dissipated heat of clusters and cold air of CRAC units have an inverse correlation. The same phenomenon is valid for s7 and s14; s7 is in the hot part of cluster A1, and s14 is inside the cage, which is the cold part of the same cluster.

As a summary of this analysis, we can conclude that in thermal modeling and monitoring *in a datacenter room, the air volume should be divided into zones separated not only by hot, cold aisles and cages - which is expected - but as well as by vertical airflow barriers generated by the active subfloor and RDHX. In addition, the subfloor temperature cannot be assumed homogeneous.*

Thermal and Power Consumption Study

This section investigated the impact of not monitored events (i.e., maintenance, racks substitution, and meteorological changes that occurred during the study period, which may have changed the room temperatures and powers dependencies) by splitting the dataset into chunks of data. Then temporal granularity of the data collection by the WSN is studied, which plays a significant role in battery lifetime and dataset size. Temporal granularity has several consequences in different steps, such as data communication and transmission, data storing, and processing. Finally, correlations of the sensors with the power consumption of clusters are analyzed.

Time Window for Correlation Coefficient This test analyzes the impact of monitoring period duration on the correlation between the clusters' power consumption and the temperature (monitored through the wireless sensor network). Indeed, during the two-month trial, several not monitored events, like maintenance, racks substitution, and seasonal changes, happened and could have temporally biased some sensor measurements. Thus, the two-months period is divided into

subsets, calling their length as Time Window (TW). In this analysis, several TWs, ranging from 24 to 1400 hours are considered (all aligned at the 00:00:00 a.m.). The set of TWs ($\{24h, 48h, ..., 1400h\}$) is named TWG. $TDCTW$ represents the Total Data Collection Time Window, which is the study period. Time Window Numbers (TWN) is the number of subsets that it be obtained by $ceil(\frac{\texttt{TDCTW}}{\texttt{TW}})$ or $\lceil\frac{\texttt{TDCTW}}{\texttt{TW}}\rceil$. The data collection rate of clusters' power consumption is lower than thermal data, so to have the same granularity in the dataset for both types of monitoring data, the upsampling of the clusters' power consumption (by replication of the same data) is employed. The power consumption and temperature data are time-aligned, and there is no time lead or lag between the two types of data. As early introduced, to analyze the impact of the length of the considered period, w.r.t the relevance of the room temperature and the power consumption, the correlation between all the possible pairs of sensors (temperature and power) for any time windows is computed, i.e., for each pair and for each time window, a correlation coefficient (CC) is computed. Among these values the mean value (Mean Correlation Coefficient)($meanCC$) is calculated.

Algorithm 1 identifies the TW values, which leads to a higher correlation between the thermal sensors values and the cluster's power consumption. It creates a matrix in which rows are pairs, and columns are different Time Window Numbers TWN. Then for each pair in different subsets, the Pearson correlation coefficient and significance test are calculated. For example, for $TW = 24$ hours for one pair (temperature sensor s1 and "power A1 a"), it computes 49 Pearson correlation coefficients and around half of this for $TW = 48$ and so on.

Next, as it is shown in Eq. 3.1 with the average for each element of row TWN, the mean correlation coefficient vector is computed. Finally, the mean correlation coefficient ($meanCC$) (Eq. 3.2) for each member of TWG is computed. Results are illustrated in the bar-chart in Figure 3.8. The maximum mean CC=0.224 obtain in TW of 216h, and it has no significant difference with CC=0.221 that we use all the dataset as one time window. *Generally, in this study, the CC for larger TW is higher than the short TW.*

$$\begin{bmatrix} cc_{1,1} & cc_{1,2} & cc_{1,3} & \dots & cc_{1,LTWN} \\ cc_{2,1} & cc_{2,2} & cc_{2,3} & \dots & cc_{2,LTWN} \\ & \vdots & \vdots & \ddots & \vdots \\ cc_{84,1} & cc_{84,2} & cc_{84,3} & \dots & cc_{84,LTWN} \end{bmatrix} \xrightarrow[\text{axis=1}]{\text{Maen}} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{84} \end{bmatrix} \tag{3.1}$$

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{84} \end{bmatrix} \xrightarrow[\text{axis=0}]{\text{Mean}} \begin{bmatrix} MeanCC \end{bmatrix} \tag{3.2}$$

Algorithm 1 Time Window Algorithm

procedure TIME WINDOW
  $TWG \leftarrow \{24, 48, 72, 96, 120, 144, 168, 192, 216, 240, 264, 288, 312, 336, 1400\}$
for all TW on TWG **do**
  $LTWN \leftarrow \lceil \frac{TDCTW}{TW} \rceil$
for all $Pairs$ **do**
for all TWN on$\{1, 2, 3, ..., LTWN\}$ **do**
  $CCmatrix$[Index of $Pair$, TWN] $\leftarrow$ *CC(Temp(SensorID,TWN), Pow(lineID,TWN))*
  $SSTMask$[Index of $Pair$, TWN] $\leftarrow$ *SST(Temp(SensorID,TWN), Pow(lineID,TWN))*
  end for
  end for
  $CCmatrix \leftarrow$ Hadamard or element-wise product of $CCmatrix$ and $SSTMask$
  $MeanCCmatrix$[Index of $Pair, 1$] $\leftarrow mean(CCmatrix$[Index of $Pair$, TWN], $axis = 1)$
  $MeanCC$[Index of TW in TWG] $\leftarrow mean(MeanCCmatrix$[Index of $Pair, 1$], $axis = 0)$
  end for
end procedure

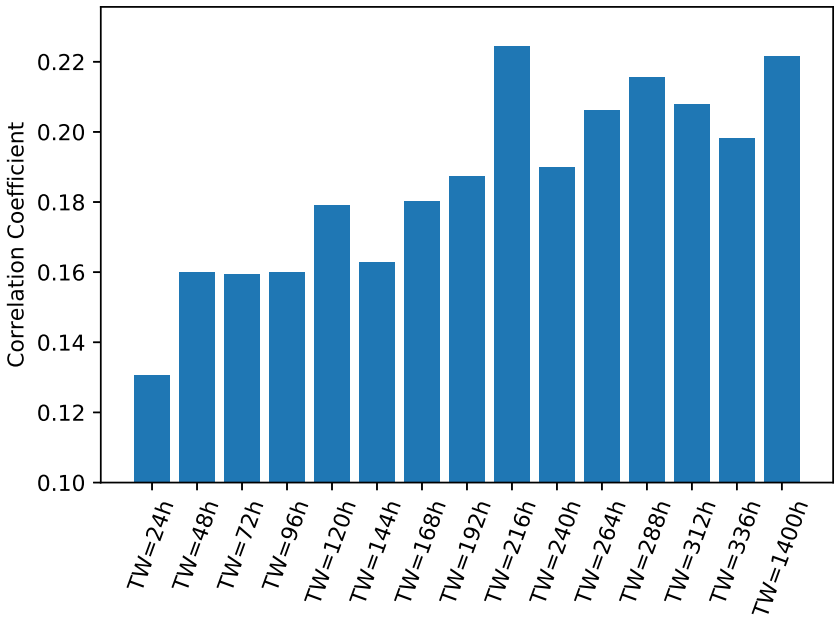


Figure 3.8: Bar Chart Mean Correlation Coefficient (*meanCC*) of All Pairs in Different Time Windows (*TW*s)

Data Aggregation/Reduction Several reasons, such as the monitoring issues,

a considerable volume of collected data, or even Big Data issues, different sampling rates of different parameters that need to be aligned, can motivate data aggregation or reduction. KairosDB (NoSQL time-series database) has different data aggregation methods such as averaging, maximum, minimum, last, median [63]. This study has two types of data, temperatures, and powers. The temperature has a finer granularity (120 samples per hour) than power (One sample per hour); therefore, to granularity alignment of the temperature and power data, the number of temperature samples per hour utilizing aggregation methods is reduced from 120 to 1 in this part of the study. (In the previous parts of the study with upsampling, the granularity of two types of data is adjusted). There are many possibilities for data aggregation from 120 to 1 per hour, e.g., using the last sample of each hour or the average of the last 5 samples of each hour. For the aggregation of the temperature data, the average of the last 1, 5, 10, 15, 30 minutes of each hour and finally the average of all 120 samples for one hour are examined. For example, figure 3.9 illustrate different correlation coefficients of "s1" and "A1 a" in different TW with the different averaging periods for data reduction. It is explicit that the different averaging period has no significant consequence on the correlation coefficient results. So the data aggregation in the range of one hour has no significant impact on data analysis. This result is because the average of standard deviations (std) for each hour does not change more than 0.4 degrees in the total data collection time window ($TDCTW$). Therefore, it is possible to reduce the data collection rate of sensors from 120 to 1 sample per hour. This will increase the battery lifetime without significant degradation in the quality of the study. We selected the last-minute sample for further study.

Figure 3.10 reports the average Pearson's correlation coefficient matrix of different pairs of clusters power consumptions and temperature sensors through TW=216h.

Cold Sensors: sensors s10, s2, s9, s1, s14, s13, s4, in the cold parts of the room generally have an inverse correlation with power consumption, which means that these parts are more affected by CRAC units than the direct effects of power consumption. With an increase in power consumption, the temperature of the hot part of the room increase, and consequently, CRAC units start, so the cold zones of the room due to the CRAC unit's activity experiences a reduction in the temperature. Sensor s1 is an exception; it directly correlates with clusters A3. This sensor also has a different pattern in correlation with other sensors that are studied in the thermal study part 3.3.2.

Sensor s10 is located in the center of the room in the middle of cluster A3. It shows a strong inverse correlation with the power consumption of cluster A3 and inversely correlates with the power consumption of A1; meanwhile, it does not correlate with the power consumption of Galileo because of the low power

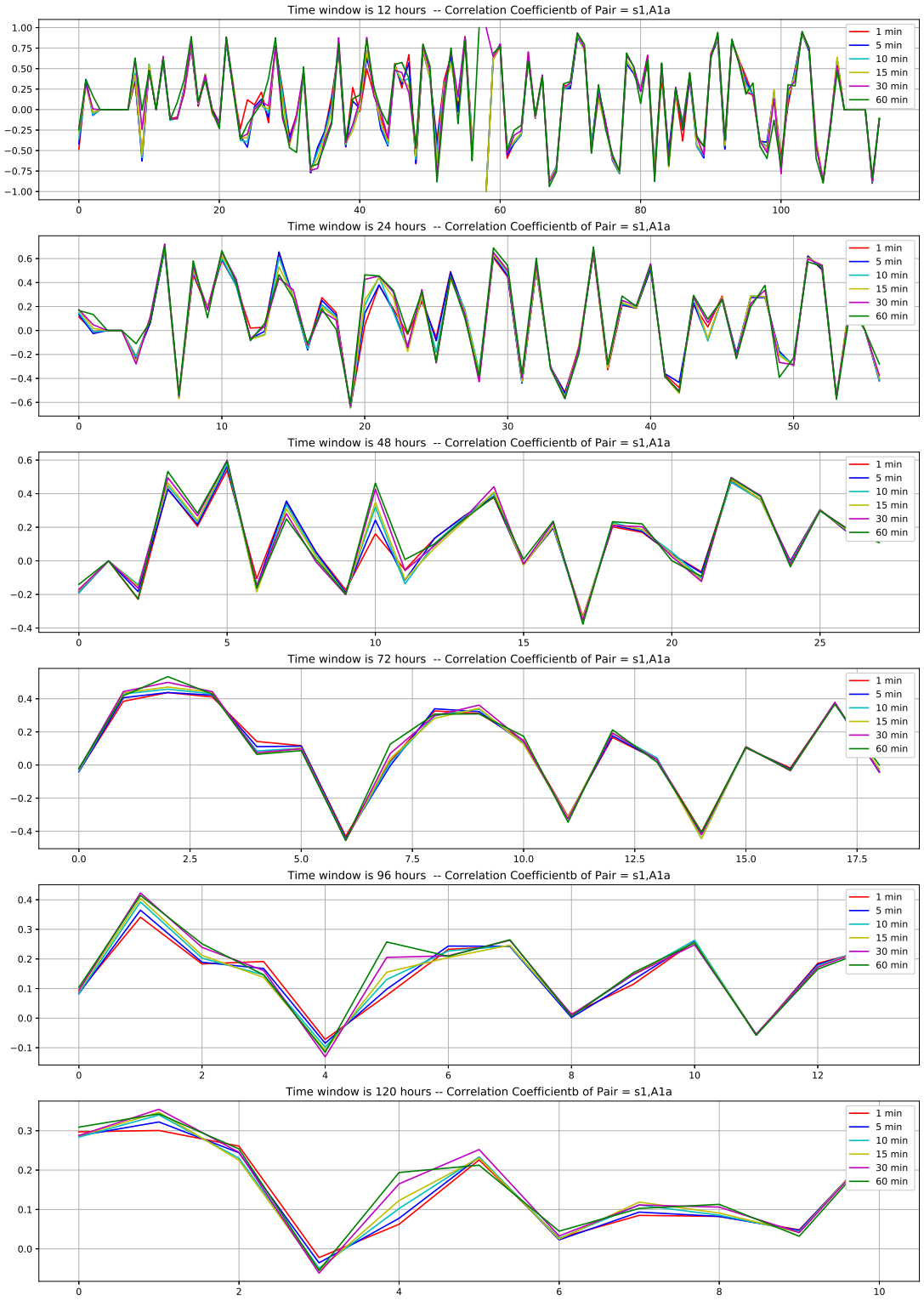


Figure 3.9: Correlation coefficients of "sensor 1" and "A1 a" at $TW = \{12, 24, 48, 72, 96, 120\}$

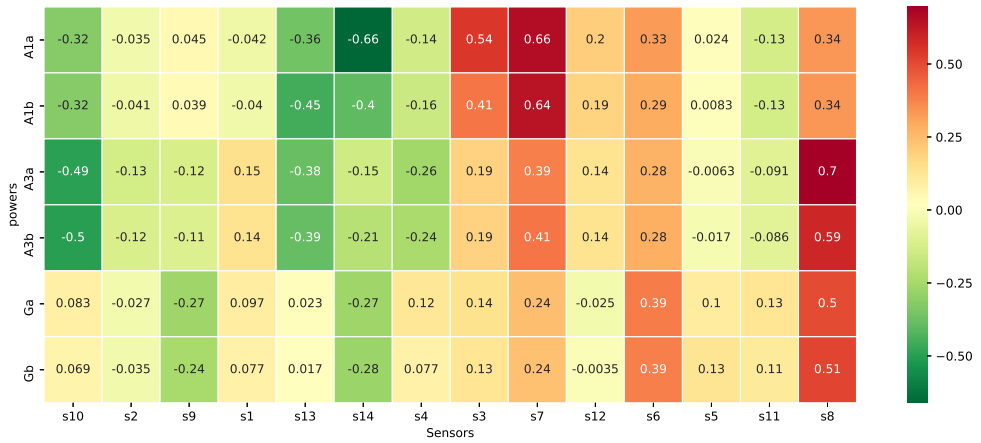


Figure 3.10: Correlation Coefficients Matrix.

consumption of this cluster. Although sensor s2 does not strongly correlate with any of the clusters' power dissipation, its negative correlation coefficient with the A3 cluster is bigger than A1 due to the cage effect.

Sensor s9 has an inverse correlation with Galileo and A3 clusters; it more inversely correlates with the Galileo cluster than A3 due to the distance. For sensor s1, A3 is farther than the Galileo cluster, but its *CC* with A3 is slightly higher than Galileo; it can be because of the high average power consumption of the A3 and the effect of the cage (Galileo has a cage). Therefore, *the effect of the cage and average power consumption can be important than the distance.*

Sensors s13 and s14 are inside the cage of cluster A1. They have an inverse correlation with the power consumption of the A1 cluster, particularly the s14 with *CC*=-0.5 correlates with the power consumption of the A1 cluster. Sensor s14 lowly inversely correlates with the A3 and Galileo clusters. Although the sensor s13 is inside the cage of the A1 cluster, it shows the same inverse correlation with the power consumed by the A1 and A3 clusters. Sensor s13 is close to CRAC5+DFC and CRAC6+DFC, so it measures the output of these CRAC units more than heat generated by the A1 cluster. These CRAC units are activated based on the heat generated by clusters A3 and A1 due to their proximity. *In clusters with a cage, the sensors located in the hot aisle (in the top of racks) highly correlate with the power consumption of these clusters. Conversely, the sensors inside the cage show an inverse correlation with the power consumption of the clusters.*

Sensor s4, located in the cold zone of the room, has a low inverse correlation with the power consumption of clusters of the center and left side of the room. It is more affected by all the cold sources, so it has an inverse *CC* with the power consumption.

Hot Sensors: Sensors s5, s11, s12, s6, s3, s7, s8 are in the hot part of the room. Sensors s3 and s7 are between cluster A1 and A3 considering that they are very close to the hot part of cluster A1; furthermore, cluster A3 have RDHX; therefore, they have a very high correlation with the power consumption of cluster A1 around 0.6, and with CC=0.2, 0.4, they correlate with the cluster A3. Due to the low power consumption of the Galileo cluster and distance and the cage of Galileo, they have a low correlation with this cluster. The sensor s7 is far from s3 from the sources of cold air, so it has a higher CC than s3.

The sensors s6 has almost the same CC with all clusters. It is more affected by the heat source than the cold. Because of the closeness, it has a bit high CC with the Gailelo cluster than others. Sensor s12 correlates with the A3 and A1 clusters. The sensor s5 lowly correlates to the Galileo cluster, and s11 has a low direct correlation with Galileo and inverse with two other clusters.

Sensor s8 in the center of the room on top of the A3 cluster has a high correlation with A3 and next Galileo and A1. It directly correlates with all of the three clusters; meanwhile, the effect of distance is evident in its CC results.

3.4 Thermal and Power Characteristic of the CINECA HPC Room F

This section investigates thermal and power consumption characteristics of HPC room F of the CINECA datacenter, which hosts the Marconi A2 Tier-0 cluster. Marconi A2 employed Lenovo NeXtScale platform, and it is based on 68-cores Intel Xeon Phi7250 (KnightLandings (KNL)) at 1.4 GHz, with many-core architecture (Intel OmniPath Cluster), provided about 250 thousand cores(68 cores/node, 244.800 cores in total) with the computational power of around 11Pflop/s. Each node has 16 GB/node MCDRAM + 96 GB/node DDR4 [12]. The CINECA datacenter features a holistic monitoring framework, namely ExaMon [7], which aggregates a wide set of telemetry data.

3.4.1 Methodology

Figure 3.11 depicts the layout of the HPC Marconi room F in CINECA. In Marconi room F, 46+1 racks (one of them is a rack of switches) are located in three rows. Each rack comprises 18 chassis in different heights, and each chassis has four computing nodes. Chassis one (C1) is in the bottom, and chassis 18 (C18) is the highest one. There are six computer room air conditioning (CRAC) units that support the two cold aisles. Racks' RDHXs are in the hot aisle. For each node and its associated components, such as voltage regulators and fans, the Intelligent Platform Management Interface (IPMI) provides remote telemetry access to the

built-in sensors [13]. The ExaMon monitoring system collects sensor data with the IPMI interface with 20 seconds sampling rate [7]. ExaMon monitored data is stored in its internal KairosDB database as time traces and remotely accessible through RESTfull APIs [7].

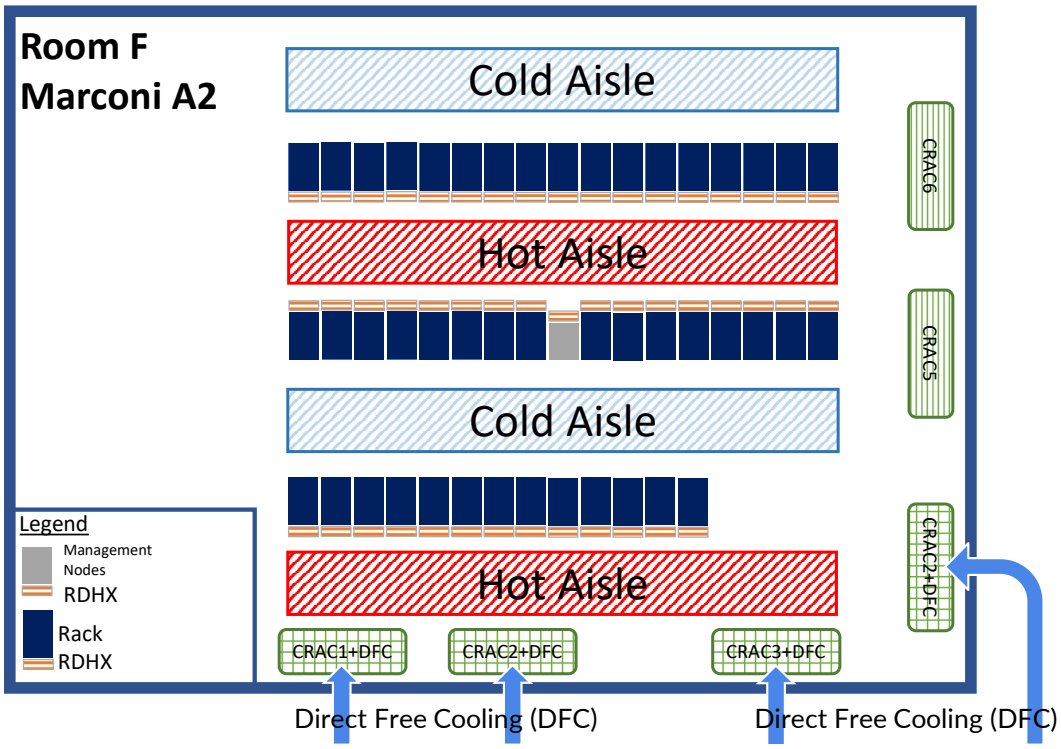
This section's analysis focuses on the following metrics: (i) inlet temperature (`BB_Inlet_Temp`) which senses the temperature of a node close to the cold aisle; (ii) outlet temperature (`Exit_Air_Temp`), which senses the temperature of a node close to the RDHX and the hot corridor. (iii) The node power, which is derived from the power measured for the two power supplies of each chassis (namely `PS1_Input_Power` and `PS2_Input_Power` metrics). The power consumption and workload are related, which is not the focus of this section. All these metrics are available in ExaMon. This study investigates the spatial and temporal heterogeneity during production and thermal hazards for the period from 2019-06-01 00:00:00 to 2019-07-01 00:00:00 over the 3312 Marconi A2 nodes. The following methodology is employed to conduct the study. The data by using the RESTfull API provided by ExaMon is extracted from KairosDB, and for data analysis and plots, the Python codes are utilized [7]. Table 3.3 summarizes the characteristics of the dataset used in this study, and the boxplot in Figure 3.12 shows the shape of the distribution of inlet and outlet temperatures and power consumption of nodes in June 2019. All the collected sensors data during June 2019 are utilized to generate these boxplots. As it is noticed, there is no overlap in the interquartile range between the inlet and outlet temperatures.

Name of Parameter	**Value**
Number of Racks	46
Number of Chassis Per Rack	18
Number of Nodes	3312
Number of Metrics	42 IPMI with KNL tag
Sampling Rate	20 Second
Period of Study	from 2019-06-01 00:00:00 to 2019-07-01 00:00:00
Thermal Emergency	2019-06-28

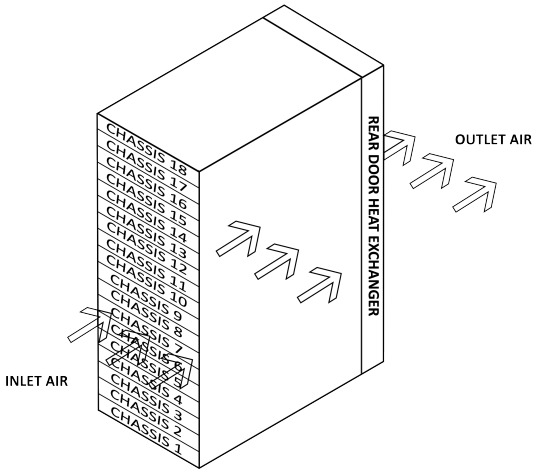
Table 3.3: Characteristics of Dataset

3.4.2 Experimental Results

To study the thermal characteristic of the Marconi A2 KNL room F the spatial and temporal aspects of temperature and power consumption of nodes in the



(a) Marconi A2 (KNL) Room F in CINECA Datacenter.



(b) Schematic of One Rack of Marconi A2.

Figure 3.11: Racks Arrangements of Marconi A2 (KNL) Room F in CINECA Datacenter.

room during June 2019 are investigated. This study contributes the 3D view of the thermal and power characteristics of the room by utilizing the heat-map of distribution of the power consumption and temperature of nodes, and also, different chassis-level analysis that represents the power consumption and thermal variation in different height of the room.

Subsection 3.4.2 analyzes the static spatial gradients present in the computing room. The analysis is conducted by averaging each metric for the entire month and studying their correlation and spatial variation. Differently, subsection 3.4.2 analyzes the temporal variations by computing the average and the min-max variation on a per-day basis for the entire computing room. Finally, subsection 3.4.2 focuses on the day for which the computing room has faced a rare cooling hazard.

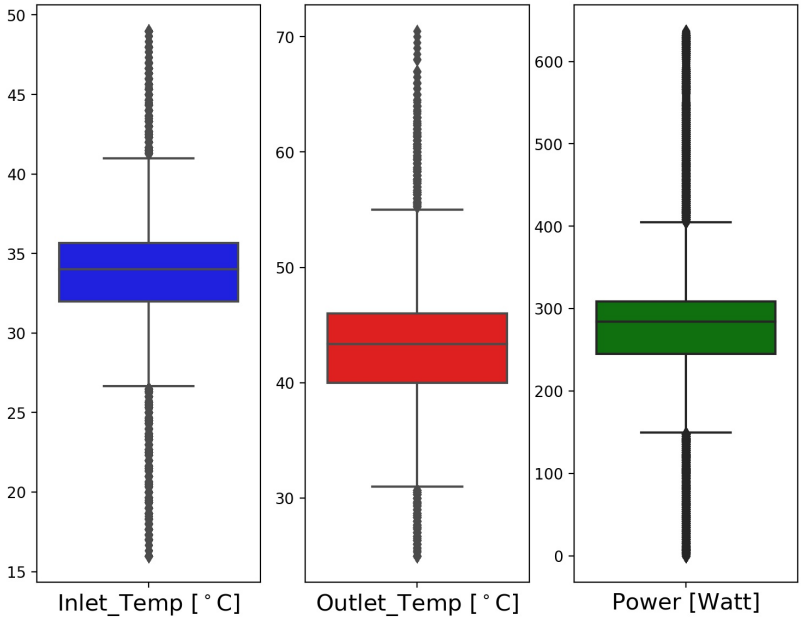


Figure 3.12: Boxplot of Inlet and Outlet Temperatures and Power Consumption of Computing Nodes in June 2019.

Spatial Study

Figure 3.13 shows the boxplot of inlet and outlet temperatures. In the x-axis, the chassis number: higher vertical position is represented by bigger chassis number, being C1 the bottom one, and C18 the top chassis in a rack. The y-axis shows the temperature in C. For each chassis number, the temperature of nodes located in

the different racks in the room still in the same chassis number and, consequently, in the same height are collected.

For each chassis number, the boxplot generated among all the nodes belonging to a given chassis number in the room ($4\ nodes\ per\ chassis \times 46 racks$) and all the samples (23.8M samples) collected in June 2019. Figure 3.14 reports the boxplot of power consumption of chassis in Watt. This plot is generated in the same approach as Figure 3.13.

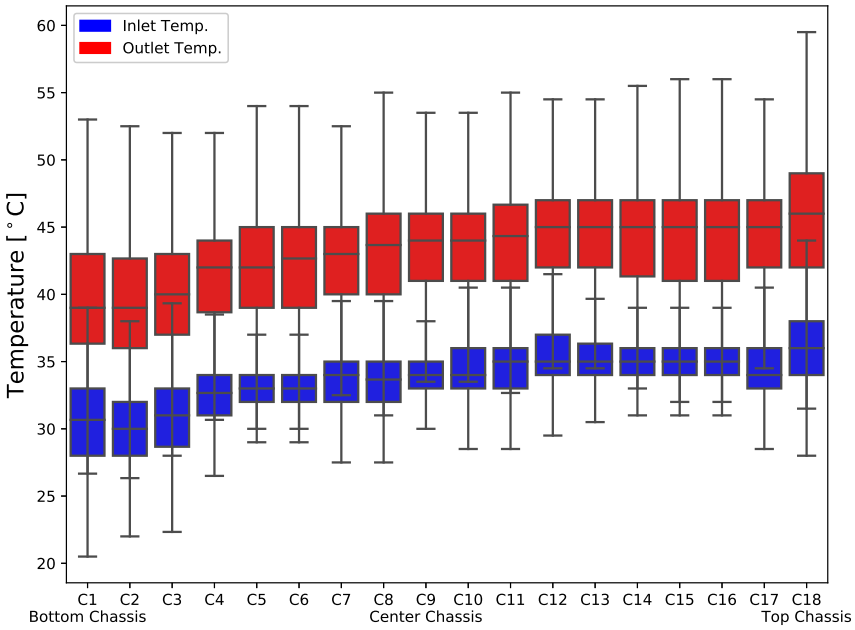


Figure 3.13: Boxplot of Inlet and Outlet Temperature of Computing Nodes in Different Chassis in June 2019.

Figure 3.13 demonstrates the presence of a vertical spatial thermal gradient. The nodes hosted in the chassis-2 of racks, on average, have a minimum inlet and outlet air temperature; therefore, these nodes, on average, are the coldest nodes in the room ($\sim 6^oC$ colder than chassis-18 ones).

Figure 3.15 illustrates the distribution of fans speed in different chassis-numbers. Measured data confirm that fans of nodes of chassis-2 work with lower speed/RPM and consume 15.8 Watt less ($\sim 6\%$) than nodes of chassis-18.

Then the thermal variation in different chassis-level is studied by averaging for the entire June the daily variation for the different analyzed metrics for different chassis levels. Figure 3.16 reports these values and indicates that chassis-2 endured maximum thermal variation, and it has experienced on average 7.3^oC thermal variations in inlet temperature and 12.1^oC in the outlet. The plot shows that the

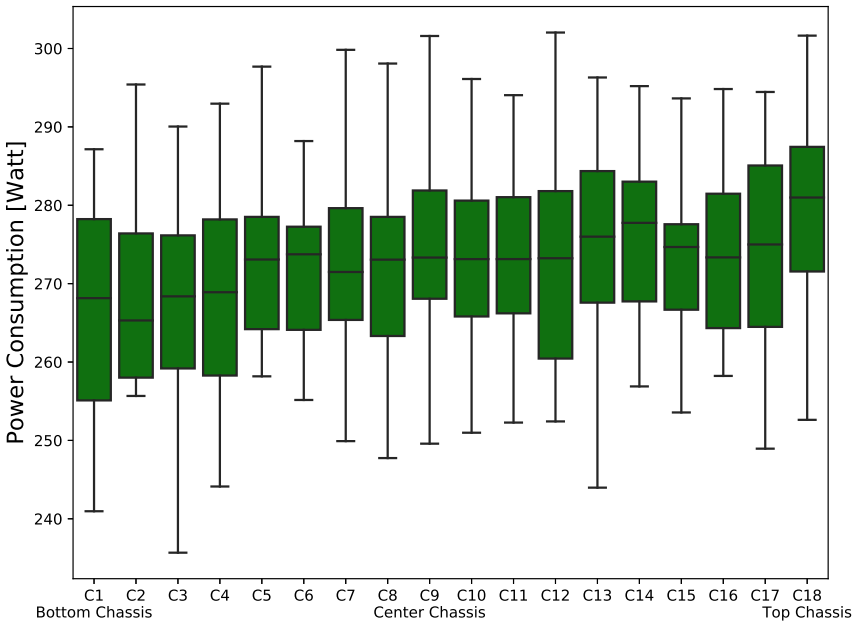


Figure 3.14: Boxplot of Power Consumption of Computing Nodes in Different Chassis in June 2019.

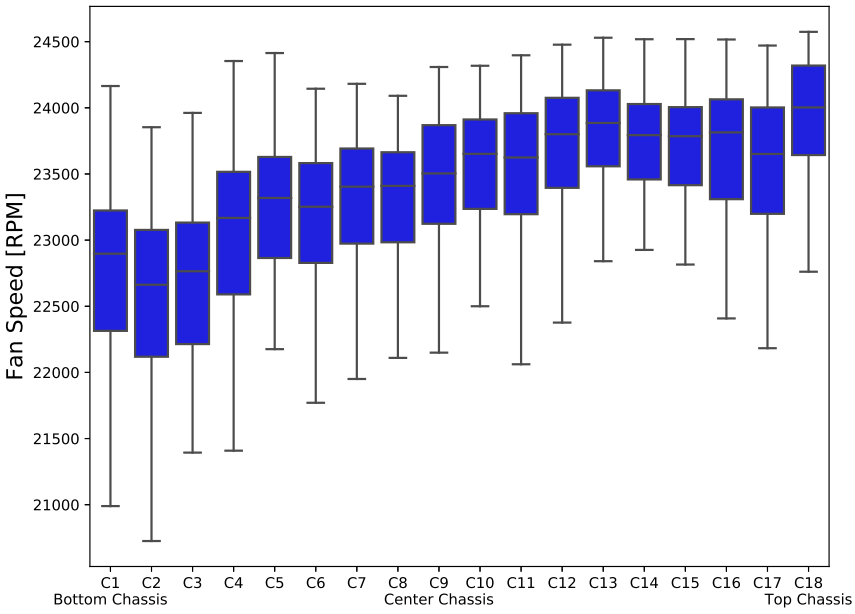


Figure 3.15: Boxplot of Fan Speed (RPM) of Computing Nodes in Different Chassis in June 2019.

thermal variation drops vertically and is more severe for the inlet temperature. This effect can be explained by the fact that the inlet temperature of the lower chassis is closer to the CRAC outlet air, which, due to free-cooling, follows the external ambient temperature and daily variations. The inlet air temperature for nodes in the higher chassis is instead affected also by the rack dissipated heat as the effect of heat recirculation. The lower variation for the outlet air w.r.t. the inlet air can be explained by the larger fan speed of the nodes in the higher chassis.

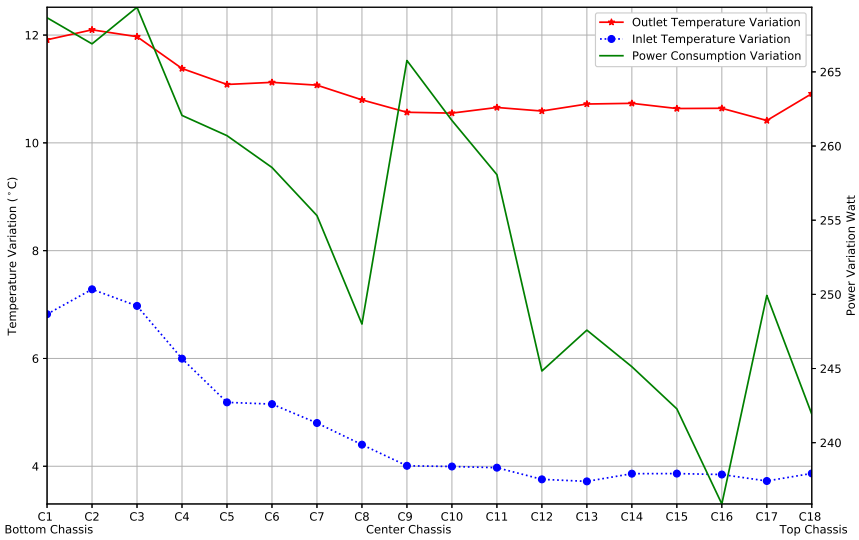
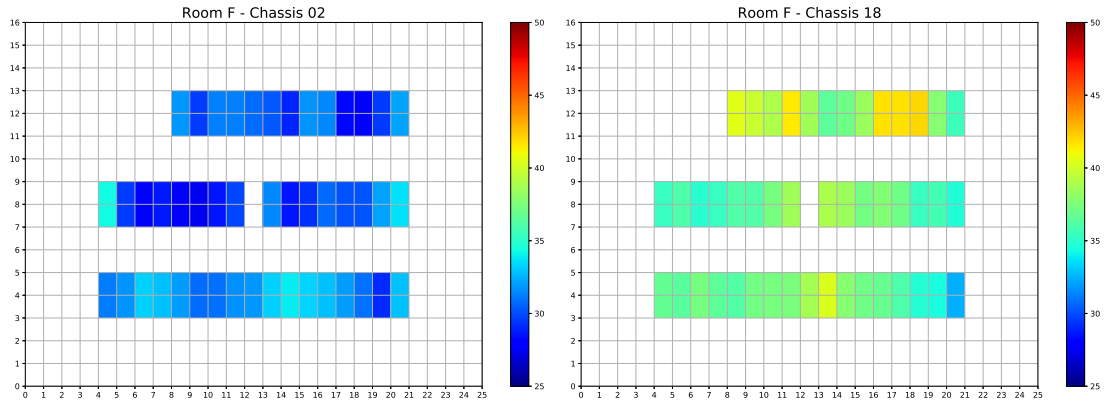


Figure 3.16: Average Inlet and Outlet Temperature Variation and Power Consumption Variation of Computing Nodes in Chassis in June 2019.

Two heat-map of room F (figure 3.17) illustrate the distribution of the average inlet temperature in different racks at two different heights (bottom and top of the racks) of the room in June. The bar-color shows the temperate in C degree. The top plot in Figure 3.17 describes the inlet thermal status of nodes at the top of the room (chassis-18), and the bottom plot in the same figure shows inlet thermal status at the bottom of the room (chassis-2). The center row of racks experiences a colder temperature for both the plots. On average, in June, the bottom nodes of each rack (chassis-2) had a maximum of 34.35^oC and a minimum of 25.46^oC as inlet air temperature for the top nodes of each rack (chassis-18) that was 42.1^oC, and 31.3^oC, respectively. Moreover, for the same height just by moving in a horizontal/plane direction for the top of the racks (chassis-18 height), the room had 10.8^oC of thermal variation, which is notable inlet temperature heterogeneity. At the bottom of the racks (chassis-2 height), the room had 2^oC lower thermal variation. It must also be noted that the two horizontal sections of the average room temperature show different hotspots locations. This result suggests that



(a) Chassis-2. (b) Chassis-18.

Figure 3.17: Average Inlet Heat Map of Marconi A2 KNL Room F on June 2019.

the horizontal heat distribution varies vertically in the room. This effect poses challenges in proactive room-level thermal management, as all the 3D thermal maps should be considered for optimizations.

Figure 3.18 reports the average power consumption of different racks in the room in June. The bar-color shows electrical power in KWatt. In the computing room, a rack's maximum average power consumption was 20.4 KWatt, the minimum of 14.0 KWatt with a standard deviation of 1.8 KWatt. The power consumption correlated to the inlet temperature with a correlation coefficient (CC) equal to 0.68. The outlet temperature correlated with the inlet temperature with CC=0.91. Finally, the outlet temperature correlated with the power consumption with CC=0.88. We can conclude that there is an intertwined dependency between node's power consumption, inlet temperature, and outlet temperature, which can be exploited for optimizing the room cooling and saving cooling energy.

Temporal Study

This subsection analyzes the temporal variations in the heat dissipation of the datacenter room.

Figure 3.19 shows in the x-axis the days of June 2019, and the y-axis reports the average inlet and outlet temperature of nodes in the room for each day in C degree. Each reported value corresponds to a day and is computed as the average among all the nodes in the room (4 nodes per chassis x 18 chassis x 46 racks) and all the samples in a day (3 samples per minutes x 1440 minutes per day).

From the plot, we can notice that all the reported metrics were relatively constant for all the days of June except the 28th, which had a thermal capacity

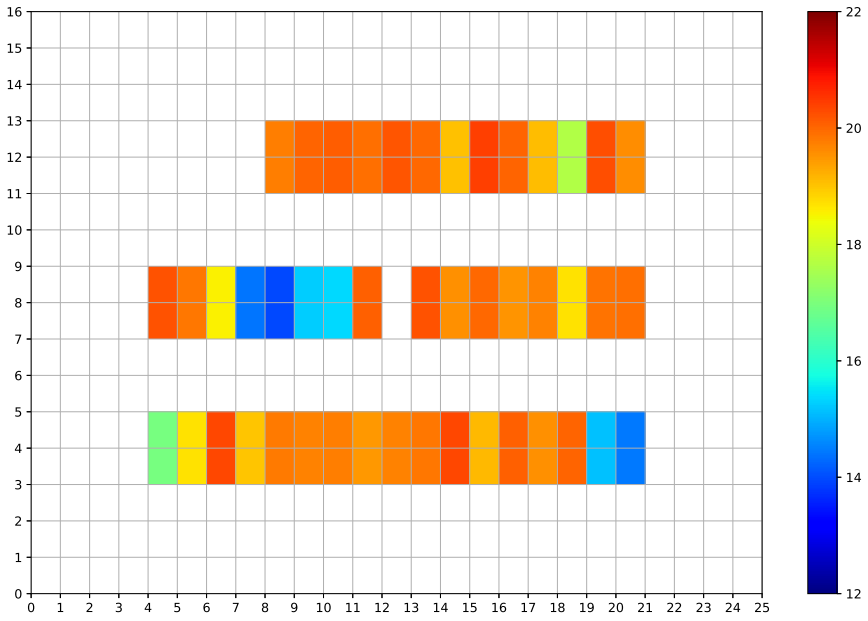


Figure 3.18: Average Power Consumption [KWatt] of Racks of Marconi A2 KNL Room F in June 2019.

failure. In addition, both the outlet and inlet temperature daily variation follows the average node power consumption. It must be noted that even if during the thermal emergency the outlet and inlet temperature increased due to the compromised RDHX cooling capacity, their average value in the day is lower due to the counteracting action taken by the system administrators that reduced the room power consumption as we will see in the following subsection. Indeed, the node average power consumption was 280 Watt on the 11th of June and 183 Watt on the 28th of June. Although with around 100 Watt reduction in the power consumption of each node, the inlet and outlet temperatures of nodes decreased, its thermal variation dramatically raised (Figure 3.20).

Measurement reveals that nodes in the thermal emergency day, on average, had 14.7^oC of thermal fluctuations in the inlet and 23.7^oC in the outlet temperatures. It must be noted that during regular days, the average thermal fluctuation (computed as the average of the daily min-max variation) is lower for the inlet temperature than the outlet temperature.

Thermal Emergency

This section analyzes the data center room's heat variation during the thermal emergency day (28th of June 2019). Figure 3.21 shows in x-axis time and left, and

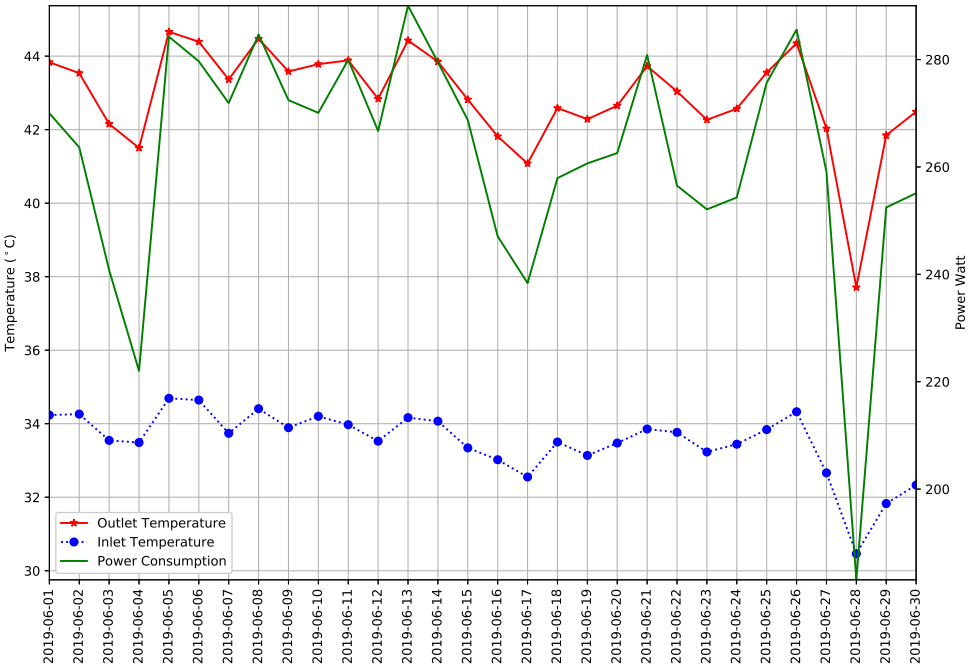


Figure 3.19: Average Inlet and Outlet Temperature and Power Consumption of Computing Nodes in Different Days of June 2019.

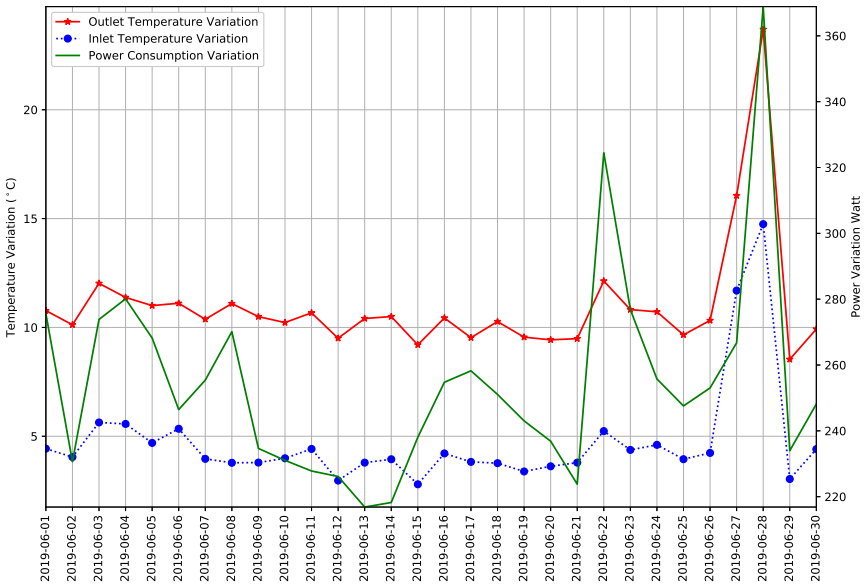


Figure 3.20: Average Inlet and Outlet Temperature Variation and Power Consumption Variation of Computing Nodes in Different Days of June 2019.

right y-axis respectively shows the temperature in C degree and power consumption in Watt. Figure 3.21 reports the inlet, outlet temperatures, and power consumption of a node during the thermal hazard day, and as it can be seen, the thermal hazard starts after 16:00 o'clock, then it reaches its' peak around 17:20 o'clock. In this period, the power consumption decreased to zero, which means the computing nodes were turned off for a while, so there is missing data for the outlet temperature. Therefore the data of five snapshots in time correspond to before, during, and after the thermal emergency were extracted. The 10:00, 12:00, and 16:00 o'clock snapshots correspond to the node's condition before the peak of thermal emergency. The 17:20 and 19:00 o'clock snapshots provide information for the peak and after the peak of the thermal emergency. Finally, the 21:00 o'clock snapshot corresponds to the recovery after the thermal emergency.

Figure 3.22 shows in the x-axis the chassis number and in the y-axis the inlet temperate in C degree. As can be noted from Figure 3.22, although generally inlet temperature increase with height, the thermal pattern of chassis in the thermal emergency period is quite different from the one during the normal conditions. Around the hazard at 17:20 and 19:00, (i) the inlet temperature increases of $\sim 5^oC$ (ii) the hotspot from chassis-18 moved to the chassis-17 and 15 also, (iii) the global minimum for temperature was not on the chassis-2, and (iv) the chassis-1 and 4 were colder than the chassis-2. The outlet temperature is reported in figure 3.23, which shows in the x-axis the chassis number and in the y-axis the outlet temperate in C degree. We can notice that the outlet temperature was colder during thermal hazard than during a typical day. This outlet temperature reduction was more prominent for the higher chassis than the lower one. For example, chassis-15 faced a 10^oC temperature reduction.

An explanation of this effect can be found in Figure 3.24, which provides the average power consumption for the nodes in different chassis during the five time-snapshot examined for the thermal emergency day. The x-axis of Figure 3.24 shows the chassis number, and the y-axis shows the average power consumption in Watt. We find that the reduction in outlet temperature correlates with a sharp decrease in power consumption from 270 Watt to 6 Watt, as is evident in Figure 3.24. Moreover, after the thermal emergency at 21:00 o'clock, the average node power has been reduced to only 150 Watt. This power reduction was due to the machine administrator intervention, which initially switched off the nodes (17:20) and then started to bring up the nodes gradually.

To finalize the study of the thermal emergency day in Figure 3.25 we report two heat-map plots of Marconi A2 room F that show the distribution of the inlet temperature in different racks at two different heights (bottom and top of the racks) of the room at 17:20 on the 28th of June 2019. The bar-colour shows the temperature in C degree. From the figure, we can notice that the center row is

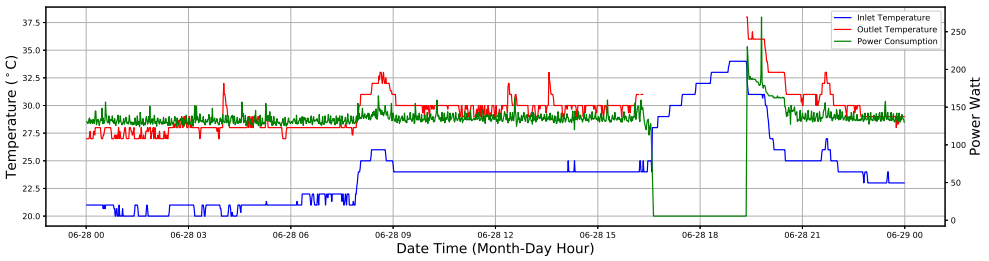


Figure 3.21: Inlet, Outlet Temperature and Power Consumption of a Computing Node on 28 June 2019 The Day of Thermal Emergency.

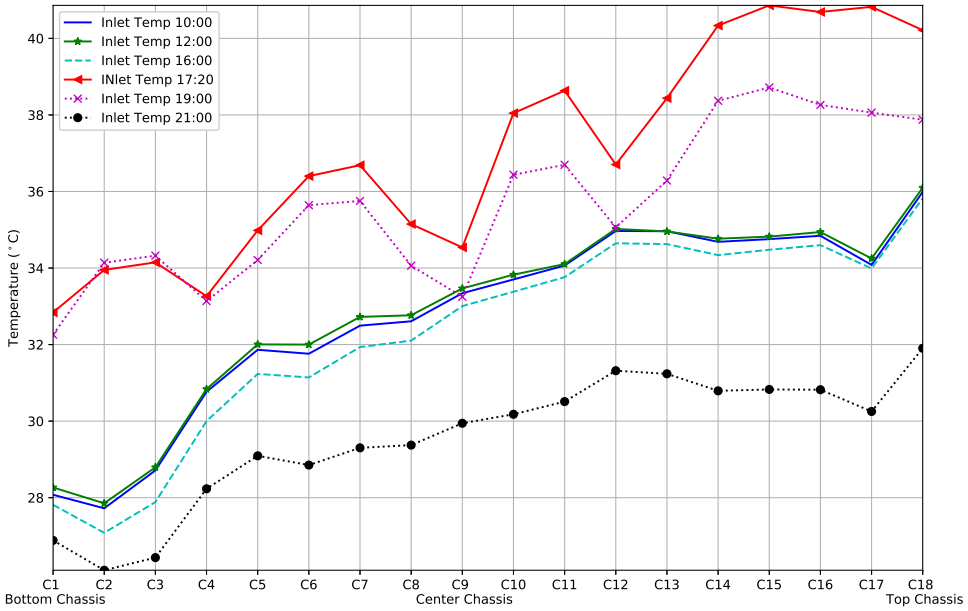


Figure 3.22: Average Inlet Temperature of Computing Nodes in Different Chassis in Different Time Instances on 28 June 2019 The Day of Thermal Emergency.

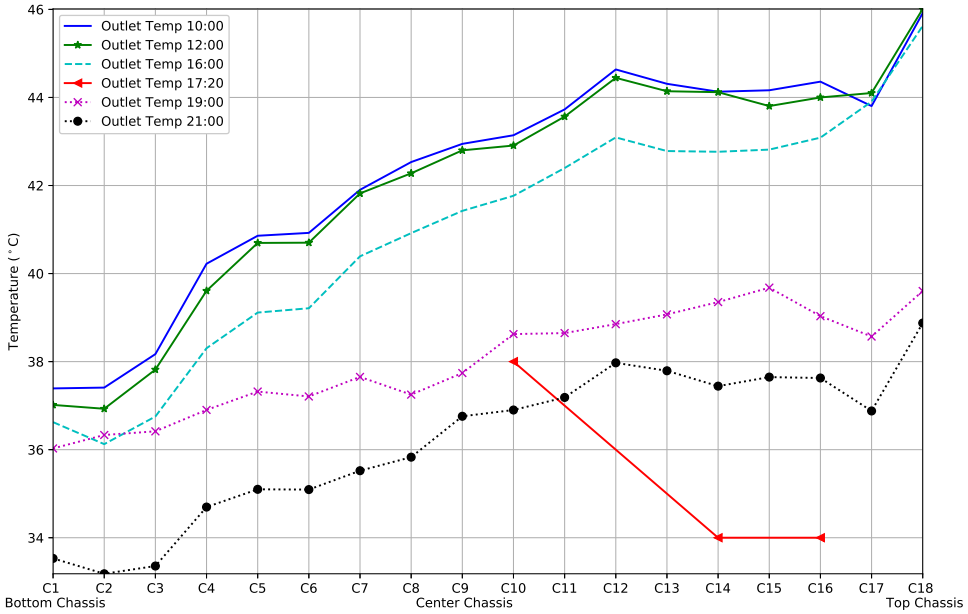


Figure 3.23: Average Outlet Temperature of Computing Nodes in Different Chassis in Different Time Instances on 28 June 2019 The Day of Thermal Emergency.

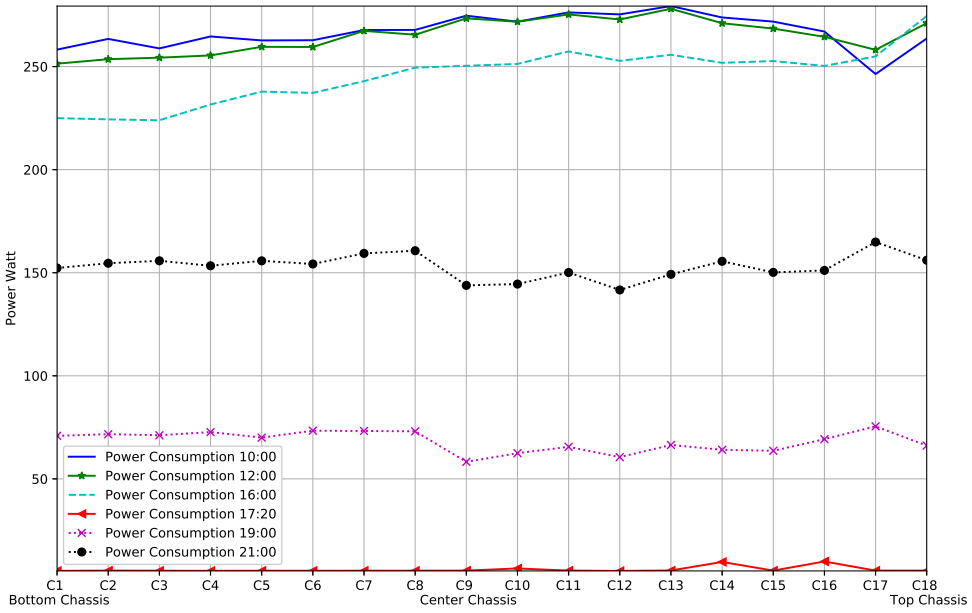
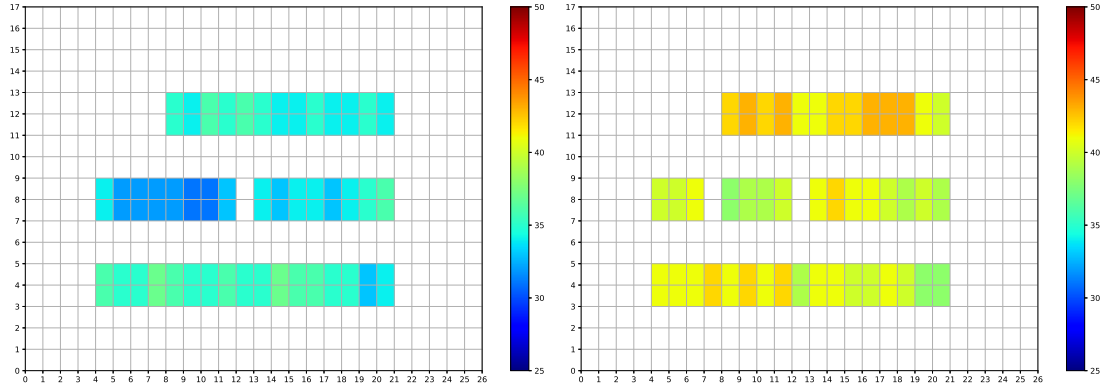


Figure 3.24: Average Power Consumption of Computing Nodes in Different Chassis in Different Time Instances on 28 June 2019 The Day of Thermal Emergency.



(a) Chassis-2. (b) Chassis-18.

Figure 3.25: Heatmap of Marconi A2 KNL Room F on 28 June 2019 at 17:20 The Day of Thermal Emergency.

almost always colder than others like the normal. At 17:20, the bottom nodes of each rack (chassis-2) had a maximum of 37^oC and a minimum of 30^oC as inlet air temperature for the top nodes of each rack (chassis-18) that was 43^oC, and 36^oC respectively. Moreover, for the same height just by moving in a horizontal/plane direction for the top of the racks (chassis-18 height), the room had 7^oC of thermal variation, which is lower than the normal condition 10.8^oC (Figure 3.17) and also in the bottom we have the same amount of variation 7^oC. Therefore the thermal heterogeneity of the room was reduced.

3.5 Summary

This chapter studies the thermal and power consumption characteristics of the two HPC rooms in the CINECA datacenter.

Considering the HPC Room N, which hosts three HPC clusters: the data collected by a WSN monitoring system in the HPC facility, which tracks the room temperature, are analyzed. The correlation between the different measured temperatures is analyzed and find that there are four thermal zones in the room: (a) Subfloor, which is a cold area. (b) The left and (c) right parts of the room that are separated by the RDHX. In the (d) vertical direction, we found that there is not a strong correlation between the top and bottom in the center of the HPC room, and the center of the room has high thermal variation. Data analysis proves that we can reduce data collection and transmission rates of two orders of magnitude. Therefore, sensors consume two times less power. With this reduction in the data collection, we need a hundred times fewer data storage capacity, and, consequently,

for data processing, it needs lower computing resources. This study can be used as a guideline for sensor placement. Finally, we can sum up; liquid cooling and cage divide the room into the different thermal zones; meanwhile, CRAC units, RDHX, and generated heat by servers create a complex thermal system. Using the internal temperature sensors and onboard sensors like IPMI combined with our WSN telemetry system, we can upgrade our system and enhance the study's preciseness. We would highlight the difficulties of doing a more precise analysis with the current WSN and then suggesting to combine it with IPMI.

Considering the HPC Room F, which hosts the Marconi A2 HPC cluster, the room's spatial and thermal heat dissipation characteristics are analyzed. The study revealed that nodes hosted in the top chassis of racks have worse thermal conditions than bottom nodes. This directly impacts the average power consumption of the nodes, which is higher for the top nodes. These nodes can consume up to 6% more power due to a higher fan speed than bottom nodes. The study of the thermal map revealed that the center row of racks in the Marconi A2 room F is colder than the other two rows; overall, this was valid for normal and thermal hazard conditions. The hotspot varies vertically during the thermal emergency condition. We can conclude that the study of the spatial and thermal heat dissipation characteristics revealed significant non-idealities and heterogeneity, which, if modeled, can be leveraged by thermal-aware job-scheduler and room-level power management runtimes.

Chapter 4

Detection and Prediction of Thermal Emergency

4.1 Overview

The ICT sector's total electricity consumption is expected to reach 20% of the world-wide demand by 2030, with data centers expected to account for one-third of that [64]. Cooling is a high cost item for datacenter operation. The power-usage efficiency ratio (PUE) expresses the additional power required by the IT for removing the heat produced by the IT power consumption. While air-cooled datacenters easily reach PUE up to 2 [64], advances in cooling technologies like direct-liquid, hot water, and free-cooling can reduce it close to almost 1 [65]. In 2016, Google announced a PUE of 1.12 [23], while in 2018, NREL achieved the world-record PUE of 1.036 by leveraging thermosyphon technology [65]. A higher than nominal coolant temperature is required to leverage free-cooling in temperate regions [32, 66], which increases the risks of thermal runaway. In the scientific computing sector, in Europe, a EuroHPC pre-exascale system costs on average ∼600K€ per day[1]. Thus each day on which the supercomputer causes to the European taxpayer a loss of ∼600k€. Whereas in the business datacenter sector, in 2016, an Amazon.com web service shortage would have cost, on average, 15M$ of revenue lost [67].

A *thermal hazard* is a dramatic increase in node temperature, which can be triggered by *i)* failures in the cooling equipment (i.e. Computer Room Air Conditioning) or *ii)* failures in the monitoring and controlling of the cooling system; this can lead to the outage of the datacenter, with severe societal and business

[1]The EuroHPC program has invested ∼650M€ in CAPEX ad OPEX for the three procured pre-exascale systems with an estimated daily average cost of ∼600k€ for a supercomputer - https://www.etp4hpc.eu/euexascale.html .

losses. Detecting thermal hazards in time is of extreme importance to avoid IT and facility equipment damage. Therefore, holistic monitoring systems are in place to monitor and visualize the datacenter state over time [7].

At the same time, progress in Deep Learning (DL) has enabled techniques for training models on large-scale time-series data. A recent DL architecture for detection of patterns in time series is the Temporal Convolutional Network (TCN) [68], which has proved able to outperform Long Short-Term Memory (LSTM) nets [69]. As such, TCNs are good candidates to perform prediction of thermal hazards, if a large data set of thermal events is available. For these reasons I choose a TCN model for our work. TCNs use dilated causal 1D-convolutions inside residual blocks. For sequence-to-sequence modeling, they can map the input series to an output with the same length. Since our TCN is a probability predictor, I equipped it with a block of dense layers at the end.

In this chapter, after an overview, state of the art, and background setup *i)* HPC room, thermal distribution during thermal hazards in a real Tier-0 datacenter is studied; under these hazards conditions, I found regularity across many nodes with a significant percentage of them having an inlet temperature above the 95% quantile. Next *ii)* a rule-based statistical approach is proposed to detect thermal hazards for an HPC room. After that *iii)* different machine and deep learning models (SVMs, SGD-classifier, LSTM, and TCN) in predicting the thermal hazard events 6 hours before they happen are investigated, which would give ample time for taking proactive countermeasures. Based on a set of experiments, I identify an optimal TCN achieving an F1-score of 0.98 in the hazard prediction for a randomly sampled. Then *iv)* different methods and tools like; samples overlap canceling, a method for dealing with an imbalanced dataset, etc., are introduced or examined. *v)* Causality is enforced between the training and test dataset to enable this framework's practical and realistic implementation in an HPC system, which leads to significant performance degradation. Then *vi)* to improve the model performance with more than 10 sets of experiments, different TCN architectures and data flows are investigated. Finally *vii)*, new approaches for defining the thresholds (used in the rule-based statistical thermal hazard labeling method) are introduced, and the TCN model performance for new sets of labels is evaluated. And at the end chapter is concluded with a summary of the chapter and motivation for the future chapter.

4.2 State of the Art

In the SoA, thermal hazards have been studied with different methodologies. [70] proposed to use simulators. [71, 72] proposed Machine Learning (ML) approaches, [73] proposed mathematical models, and finally, [74] proposed to use sensors with a

computer model to create the room's heat map or thermal evolution model. While the simulator is hard to tune to the real environment, [71] used an Artificial neural network (ANN) model trained with offline simulation data of Computational Fluid Dynamics (CFD). Compared to the CFD, the ANN model's fast response time is suitable for an online predictor. To the best of our knowledge, no one has leveraged the large data available from holistic monitoring systems to study the statistical thermal hazard distribution, or proposed a data-driven Big Data (BD) and DL model for predicting thermal hazards.

4.3 Background Setup

Our study focuses on Marconi-A2 (KNL), the largest partition of the Tier-0 cluster Marconi at the CINECA datacenter, where it was hosted in the *Marconi KNL Room* (Figure 4.6, bottom-left). In this room Marconi-A2 was composed of 3312 nodes with one 68-cores Intel Xeon Phi 7250 CPU Knights Landing (KNL) running at 1.4 GHz. Nodes had a 16 GB/node MCDRAM and a 96 GB/node DDR4. The internal network was Intel OmniPath Architecture 2:1. The cluster's peak performance was 11 PFlop/s [5].

Marconi KNL Room hosted 46 racks, plus 1 rack of switches, arranged in 3 rows; each rack had 18 stacked chassis, each with 4 nodes, totaling 3312 compute nodes. All racks had Rear Door Heat eXchangers. The room's 2 *hot aisles*, and 2 *cold aisles* were supported by 6 Computer Room Air Conditioning units.

CINECA runs a holistic monitoring framework, called EXAscale MONitoring (ExaMon) [7], scalable and capable of high-rate HPC telemetry from a wide range of heterogeneous sensors and data sources. For each cluster node and associated components, such as voltage regulators and fans, the Intelligent Platform Management Interface (IPMI) provides remote telemetry access to the built-in sensors [75]. ExaMon collects sensor data via IPMI at sampling interval 20 s [7] via an MQTT broker, and stores them using KairosDB, a specialized time-series database built on Cassandra (a NoSQL database management system), remotely accessible via RESTful APIs.

4.4 Thermal Hazard Prediction Methodology

In this section, a rule-based statistical tool for thermal hazard detection is introduced based on the statistical analysis of two real reported thermal emergencies. This tool is adopted to generate ground-truth labels of the HPC room for the whole year 2019. After some preliminary definitions and analysis related to the dataset, then a framework for thermal hazard prediction is suggested, which encompasses data

query and preprocessing, model training, and final model inference, which provides the prediction. Finally, preliminary experiments utilizing classical machine learning and deep learning tools empirically show that TCN is the best model as a brain of the framework.

4.4.1 Thermal Hazard Analysis and Labels Generation

In this section, a rule-based statistical tool for thermal hazard detection is introduced based on the statistical analysis of two real reported thermal emergencies. This tool is adopted to generate ground-truth labels of the HPC room for the whole year 2019. Based on the study in [75], CINECA Marconi KNL room had two known physical-thermal-hazard events in 2019: one on 28$^{\text{th}}$ June (peak from 16:00 to 19:00), and one on 1$^{\text{st}}$ July (peak from 14:30 to 17:00). In this chapter, physical-thermal-hazard refers to these two recorded failures of the cooling system. The distribution of temperatures during these two peaks to compare the hazard distribution with the non-hazard distribution is analyzed: the aim is to find indicators of the thermal hazards in the temperature data.

As a non-hazard distribution, we use the temperatures of the nodes in June and July, and downsampled to 1 Sample/minute: this yields $\sim$ 88k samples, large enough to be representative of the ordinary temperature distribution. In this study, two temperature metrics from ExaMon are selected: the inlet temperature `BB_Inlet_Temp` and the outlet temperature `Exit_Air_Temp`. These are the metrics most related to room temperature and were taken for every computing node.

Figure 4.1-top reports the inlet temperature distribution of one node for the three cases: non-hazard, 28$^{\text{th}}$ June hazard, and 1$^{\text{st}}$ July hazard. The dashed black line is the quantile 0.95 of the node's non-hazard distribution: as it is evident, the quantile 0.95 is a threshold that separates well the non-hazard and hazard temperatures. Figure 4.1-bottom reports the same information for the outlet temperature: hazard and non-hazard distributions overlap much more compared to inlet temperature, making it impossible to discriminate by thresholding on outlet temperature. We inspected some randomly selected nodes with this approach. We determined that the single-node quantile 0.95 of the non-hazard inlet temperature is a good parameter to discriminate between hazard and non-hazard.

Node-threshold (NT)

Based on the characterization of thermal hazards described above, this study introduces two approaches to define *node-threshold* to indicate that one node in one timestamp is in thermal stress.

1.Cooling-aware: *node-threshold* defined for each node as the 0.95 quantile of its inlet temperature distribution over the entire dataset (which covers the whole 2019

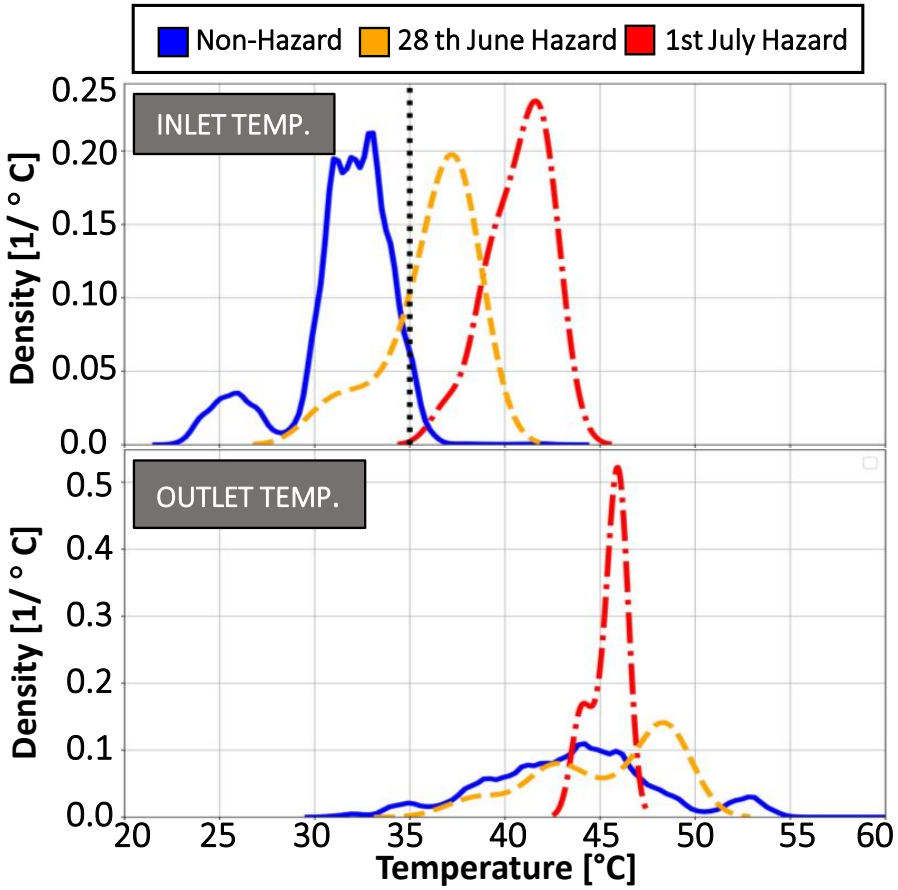


Figure 4.1: Temperature distributions for Marconi A2's node 141 in June-July 2019.

	Time	Node_1	···	Node_3311	Node_3312
6 HOURS	2019-01-25 00:00:00	30°C	...	41°C	42°C
	...	...	...	...	...
	2019-01-25 05:58:00	29°C	···	39°C	40°C
	2019-01-25 05:59:00	28°C	···	39°C	40°C

(a) Inlet Temperature dataset

	Time	Node_1	···	Node_3311	Node_3312
6 HOURS	2019-01-25 00:00:00	FALSE	···	TRUE	TRUE
	...	...	...	...	...
	2019-01-25 05:58:00	FALSE	···	FALSE	FALSE
	2019-01-25 05:59:00	FALSE	···	FALSE	FALSE

(b) True-False table

Figure 4.2: Time Windowing and Labeling.

year). So the *node-threshold* can be different for different nodes. The Cooling-aware approach uses the data of all nodes in different racks and chassis, which spread in the room so it can see the failure of the CRAC units and RDHX system.
2.Aging-aware: *node-threshold* defined for each node as the 0.95 quantile of inlet temperature distribution of all nodes in the room over the entire dataset (which covers the whole 2019 year). So the *node-threshold* is the same for all nodes. We know from the study of thermal and power characteristic of the HPC room (Chapter 3) that nodes located in the chassis at the bottom of racks perceive lower temperatures than the chassis at the top of the racks because of the raised floor. Therefore when one constant temperature for all the nodes' *node-threshold* is set, the probability of that node in the bottom of racks encountering temperature bigger than *node-threshold* is less than this probability for top nodes. So in the Aging-aware approach, we set a *node-threshold* that takes care of nodes that experience high temperature. The high temperature has an essential impact on the age of the silicon.

In this study, the Cooling-aware approach to define *node-threshold* is used. Figure 4.2(a) summarizes a 6-hour time window (TW) of the inlet temperature dataset. We applied the *node-threshold* to assign to each (node, time) cell a `True`/`False` label indicating sample-by-sample thermal trouble, as shown in Figure 4.2(b). We empirically chose TW = 6 hours.

Spatial-temporal-impact-threshold (STIT)

To assign hazard /non-hazard labels to Time Windows (TW)s (Figure 4.2(a)), not just to samples, we introduce a *spatial-temporal-impact-threshold* able to account for thermal hazards' spatial and temporal continuity. A 3312-node 6-hour TW with 1 Sample/minute amounts to $3312 \times 6 \times 60 = 1192320$ `True`/`False` values (Figure 4.2(b)). The *spatial-temporal-impact-threshold* regulates the portion of `True`'s inside the TW required to declare the room in thermal hazard. A higher quorum will select thermal hazards that are more widespread, i.e. involve more nodes for a longer time. *spatial-temporal-impact-threshold* is a general answer to the following question. How much thermal hazard spread in time and different nodes in the room (in a Time Window)?

It is essential to remark that, though based on real information extracted from the physical hazard distribution, this statistical labeling approach is artificial and must be confirmed by comparing with the ground-truth reported thermal emergencies. As made evident in Figure 4.3 (x-axis is date), if we set the *spatial-temporal-impact-threshold* to 5%, our statistical approach captures the reported ground-truth thermal emergency, while detecting additional thermal hazards, which were unnoticed by the system administrators. Indeed, these are conditions for which the compute nodes' temperatures have drastically increased without causing

immediate damage, but still possibly damaging the nodes. With our statistical labeling approach, this approach can capture these events which are unnoticed by humans.

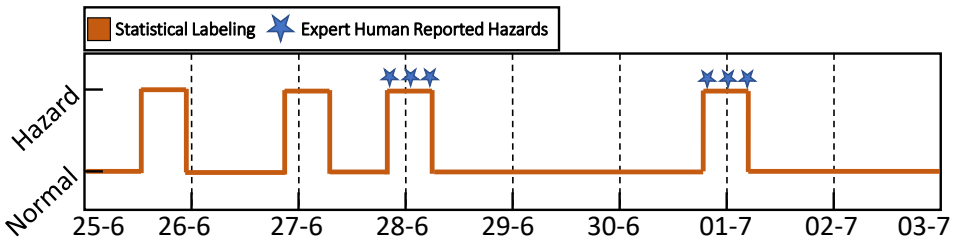


Figure 4.3: Thermal Hazard Detection

If we increased the *spatial-temporal-impact-threshold* quorum to 25%, the statistical labeling approach could only detect the second hazard, thus being to restrictive in identifying abnormal states. For the selected *spatial-temporal-impact-threshold* = 5%, the room is labeled in thermal hazard for 19.5% of the time in 2019. and, raising the threshold to 15%, the thermal hazard category reduces 3.8%, still detecting both hazards. This quantifies how rarer the extensive thermal hazards are, compared to narrow ones.

	Spatial-Temporal-Impact-Threshold		
	5%	**10%**	**15%**
Node-threshold 95%	19.5%	8.0%	3.8%

Table 4.1: Thermal Hazard Percentage.

4.4.2 Imbalanced Dataset

In general, anomalies are rare events, so the thermal hazard is a minority class in this study. There are different methods for dealing with an imbalanced dataset like (i) upsampling the minority class, downsampling the majority class, (ii) using the weight of classes in the loss function to compute the loss based on the weight of the classes [76, 77]. We examined both methods and decided to use upsampling the minority class and downsampling the majority class to reach a balanced dataset.

4.4.3 Prediction Horizon

The *Prediction Horizon* (PH) is defined the label's time distance since the last input data. For instance, if the input is the temperature of time window 00:00:00-05:59:59 and PH = 6 hours, the task is to predict the state of the time interval 06:00:00-11:59:59. For the PH = 0, it is the *detection* task. Table 4.2 reports the

time interval of state prediction or detection for different PH if the input is the temperature of time window 00:00:00-05:59:59. In particular, PH = 6 hours was chosen upon discussions with system administrators, as a tradeoff sufficient to provide enough time for the different correction actions to be taken by the system administrators. Treating PH = 6 hours as a time lag, the hazard/non-hazard binary ground-truth labels have autocorrelation 0.65 over the year 2019.

Time Interval of Input Data	Prediction Horizon	Time Interval of State Prediction or Detection	Model Type
00:00:00 to 05:59:00	0	00:00:00 to 05:59:00	Detector
00:00:00 to 05:59:00	6	06:00:00 to 11:59:00	Predictor
00:00:00 to 05:59:00	12	12:00:00 to 17:59:00	Predictor
00:00:00 to 05:59:00	18	18:00:00 to 23:59:00	Predictor
00:00:00 to 05:59:00	24	24:00:00 to 29:59:00	Predictor

Table 4.2: Prediction Horizon.

4.4.4 Last Value Predictor

Last Value Predictor (LVP): minimum baseline for any time-series task; the prediction $\hat{y}$ is simply a copy of the present observation y_{true}, with PH prediction horizon as defined in Section 4.4.5. For instance, if the prediction horizon = 6 hours, the LVP will return the room's label from 00:00:00 to 05:59:00 as a prediction to 06:00:00-11:59:00.

$$\hat{y}(t + \text{PH}) = y_{\text{true}}(t) \tag{4.1}$$

Last Value Predictor		
Time Interval of Labe that used as Prediction	Prediction Horizon	Prediction Time Interval
00:00 to 05:59	6	06:00 to 11:59
00:00 to 05:59	12	12:00 to 17:59
00:00 to 05:59	18	18:00 to 23:59
00:00 to 05:59	24	24:00 to 29:59

Table 4.3: Last Value Predictor.

The following Figure 4.4 reported the accuracy and f1 score of the LVP for different prediction horizons. There is a periodicity of every 48 hours; it is expected to be 24 hours.

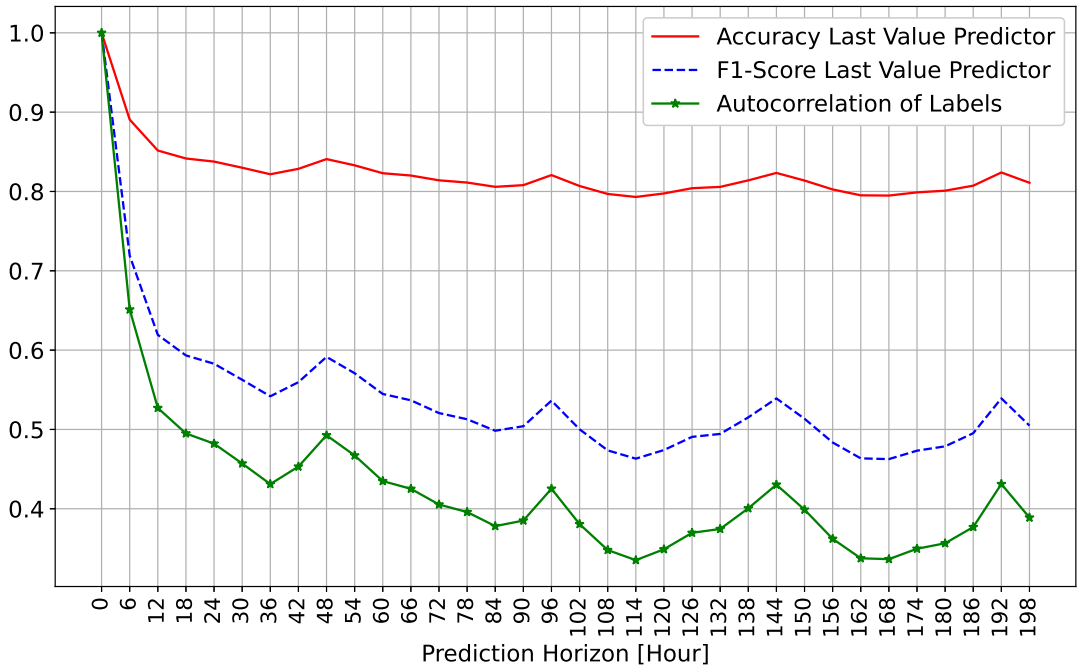


Figure 4.4: Last Value Predictor.

The average temperature of all nodes and the sum of the node labels (NOT room label) with time windowing of 6 hours are studied to check the reason for 48 periodicities.

$$MeanTemp(t) = Mean(Temp(3312Nodes, t\,to\,t+6)) \tag{4.2}$$

$$SumNodeLabel(t) = \sum(Label_{Nodes}(3312Nodes, t\,to\,t+6)) \tag{4.3}$$

Figure 4.5 shows the autocorrelation of the average temperature of all nodes in the red line and the autocorrelation of the sum of the node labels in the green line with a time window of 6 hours. The autocorrelation of the mean temperature of all nodes has a periodicity of the 24 hours the peaks are not strong but visible. After applying the Node-Threshold to the temperature and converting the temperature to the label True/False table, the pattern changed to 48 hours periodicity.

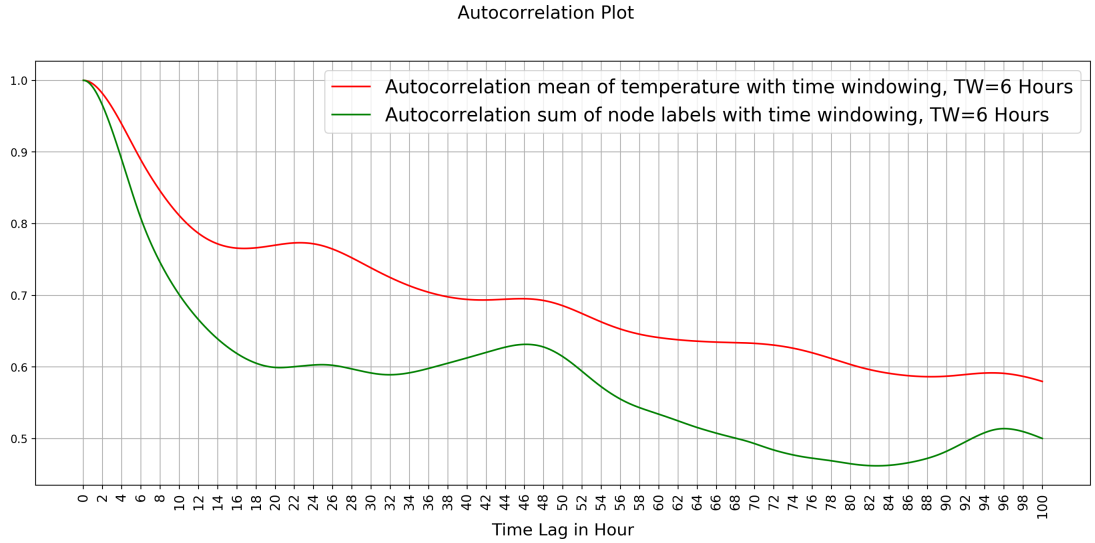


Figure 4.5: Auto correlation.

4.4.5 Thermal Hazard Prediction Framework

In this section, a framework for thermal hazard prediction is suggested, which encompasses data query and preprocessing, model training, and final model inference, which provides the prediction. The thermal hazard predictor is a model that, based on time series data of computing nodes' sensors, predicts if a thermal hazard will happen in the room in the next hours. Input data are the time series of nodes' temperature (and power consumption), and the output is a binary classification: likely forthcoming hazard or not.

As mentioned, PH = 6 hours was chosen upon discussions with system administrators as a tradeoff sufficient to provide enough time for the different correction actions to be taken by the system administrators. Treating PH = 6 hours as a time lag, the hazard/non-hazard binary ground-truth labels have autocorrelation 0.65 over the year 2019 and identifying ground-truth labels 6 hours apart as the output and target of a Last-Value Predictor (LVP) yield an F1-score of 0.72. Being the LVP, the simplest (non-)model, F1-score = 0.72 is a baseline any proposed model must be compared against.

Figure 4.6 illustrates our proposed architecture for the thermal hazard predictor, composed of three main components: the architecture for data collection, storage, based on ExaMon (section 4.3), the thermal hazard analysis including the data extraction, preprocessing (e.g., missed data handling, time alignments), label generator and data loader, and the Deep Learning (DL)-powered thermal hazard prediction system (training and inference). For the DL model used for prediction,

a Temporal Convolutional Network (TCN) is selected [68].

The TCN's input is a TW of data extracted from the database. In the off-line training stage, a large set of TWs is extracted (training set), and preprocessed to generate the ground-truth labels with the two-threshold statistical approach of Section 4.4.1. Inferences with the trained model are the predictions of thermal hazards.

Relying on ExaMon, it is possible to implement and test DL models using a very broad set of node metrics collected from sensors: the database stores hundreds of metrics, of which 42 are IPMI metrics. In this work, we focused on the nodes' inlet temperature (as motivated in Section 4.4.1).

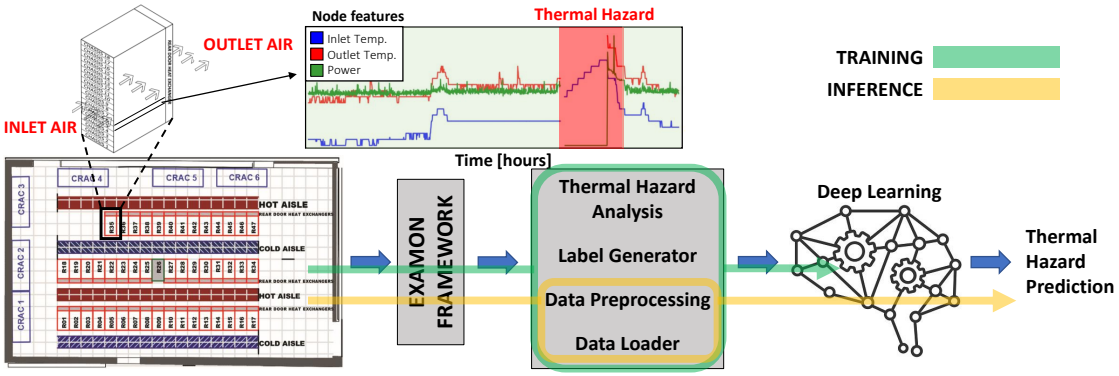


Figure 4.6: Architecture for Thermal Hazard Predictor.

4.5 Machine Learning Model Selection

In this section, to find the most suitable machine learning model for the thermal hazard prediction framework, different machine learning tools are evaluated in predicting thermal hazards in CINECA's Marconi A2 KNL Room F. We describe the dataset, introduce our Temporal Convolutional Network (TCN) topology and competitor models, and finally discuss two experiments highlighting our TCN's promising prediction skills.

4.5.1 Experimental Dataset

Experiments were based on the inlet temperature time series of HPC room, which hosts 46 racks containing 18 chassis, each chassis include 4 nodes. So in total sensory data of 3312 nodes, for the whole year 2019. There were two thermal hazards on 28th June and 1st July. We have utilized the IPMI interface to data collection with a sampling rate of 20 seconds; then, data were downsampled to 1 minute in the preprocessing step.

We generated the ground-truth labels with the statistical approach described in Section 4.4.1, with *node-threshold* = 0.95 and *spatial-temporal-impact-threshold* = 0.05 as motivated. With these values, 19.5% of the data is labeled as a thermal hazard, sufficient for training our algorithms.

4.5.2 TCN and Competitor Predictors

The proposed TCN has 2 blocks: (1) a *Feature Learning Block* (14k parameters) of 7 1D-convolutional layers with average pooling; (2) *Classification Block* (173 parameters) of 4 dense layers of 15, 6, 4, 3, 2 units. All layers present the batch normalization and the ReLU activation (Figure 4.7).

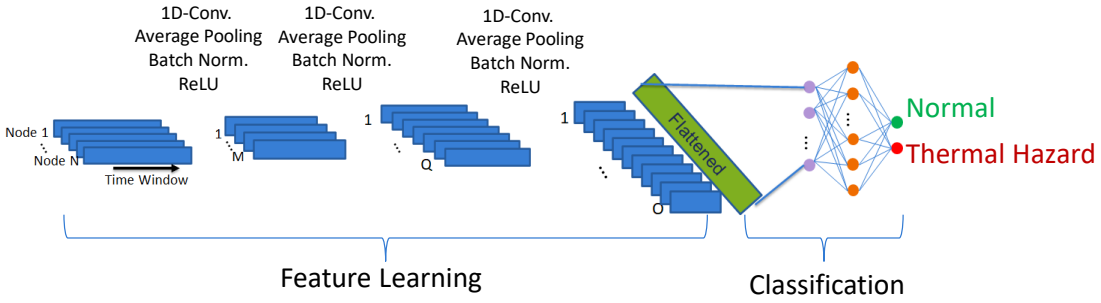


Figure 4.7: TCN Model with 1DConv. Layers.

We compare our TCN against other models[2]:

0) Last Value Predictor (LVP): minimum baseline for any time-series task; the prediction $\hat{y}$ is simply a copy of the present observation y_{true}: $\hat{y}(t + \text{PH}) = y_{\text{true}}(t)$, with PH prediction horizon as defined in Section 4.4.5.

1) Support Vector Machine (SVM): SVM with either linear or Radial Basis Function (RBF) kernels. SVMs produce decision boundaries with margins to improve generalization.

2) Stochastic Gradient Descent (SGD)-classifier: linear SVM trained with SGD instead of convex optimization, enabling larger train set size.

3) Long Short-Term Memory (LSTM): a type of Recurrent Neural Network (RNN) that learns long-term dependencies thanks to additional gates [69]. Our LSTM has 2 layers of hidden and output size 16, followed by a dense layer.

To keep the parameter space of the models small, the models were built using as input only the `BB_Inlet_Temp` temperatures of 72 nodes which composes one

[2]SVMs and SGD-classifier were implemented in Scikit-learn 0.23; LSTM was implemented in Keras 2.4; TCN was implemented in PyTorch 1.5.

rack. The rack was selected randomly in the room. We remark that all the 3312 nodes were used for generating the thermal hazard labels.

4.5.3 Experiment 1: Random Test Dataset ML-Model Selection

In the random test dataset experiment, we selected the test dataset randomly as 20% of the 2019 data, and trained all models on the remaining 80%. Table 4.4 shows the results.

The linear SVM yields F1-score 0.55, essentially random and worse than the LVP-baseline: this is due to the linear models' poorness and to the train set reduction made necessary by computational complexity. The RBF ranks better, with F1-score 0.86, which is also 0.17 above the SGD-classifier. Both DL models outperform the non-deep ones: the LSTM reaches F1-score 0.91, and our TCN ranks best, with F1-score 0.98.

So empirically is shown that DL models work better than classical machine learning tools in the thermal hazard prediction framework. And in DL models, TCN outperforms the LSTM model; therefore, we selected the TCN for continuing the study.

ML-Model	Recall	Precision	F1-score
Last value predictor	0.72	0.72	0.72
Linear SVM	0.55	0.56	0.55
RBF-SVM	0.80	0.94	0.86
SGD-classifier	0.64	0.76	0.69
LSTM	0.84	0.98	0.91
TCN	**0.97**	**0.99**	**0.98**

Table 4.4: Prediction Results

For this experiment, the TCN model (Figure 4.7) employed 1D Convolutional layers, and the input data structure is depicted in figure 4.8.

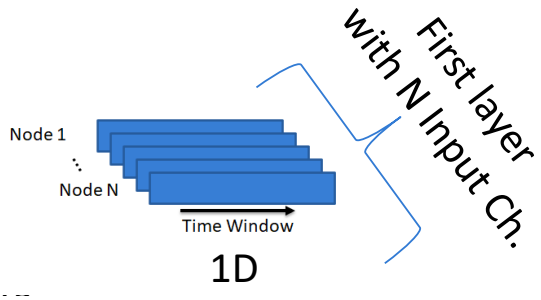


Figure 4.8: Input Data Structure of the TCN Model with 1DConv. Layers.

4.6 Experimental Results

After creating a thermal hazard framework and selecting the best ML model with preliminary experiments in this selection, the results of different experiments that are done aim to improve the prediction performance and implement it on the in-production HPC system to use in an online mode.

4.6.1 Experiment 2: Overlap Cancellation of Training and Test Dataset

One of the problems of experiment 1, which is a common issue in most studies, is that there is a lot of overlap in each successive sample, i.e., each consecutive sample of the dataset has a lot of replicated data. So if one of the two consecutive samples be in the training dataset and the other in the test dataset, due to the high overlap of the two samples model, somehow is already trained by the test sample, i.e., a test sample is somehow inside the training dataset.

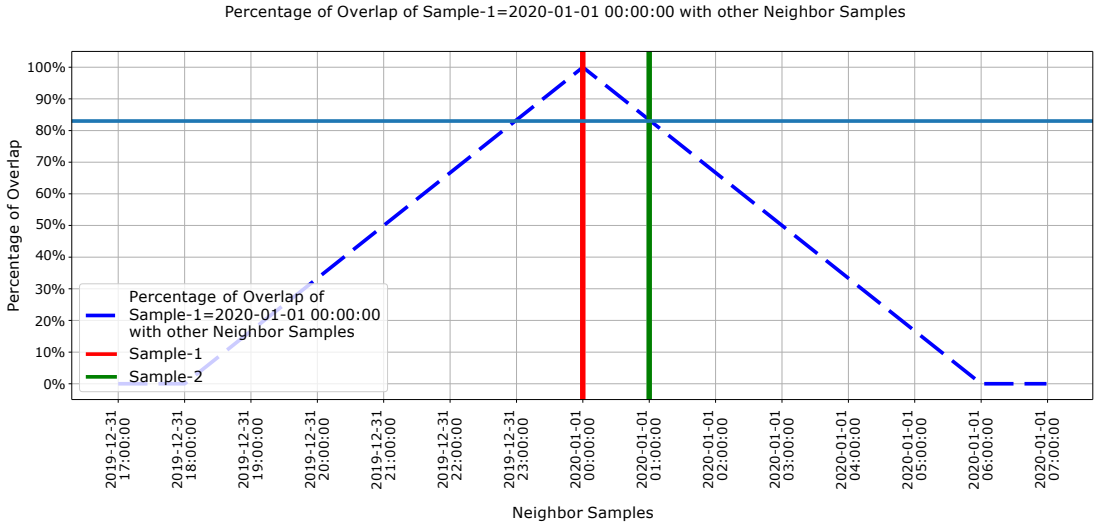
Each sample of the dataset is composed of two parts input data which is the temperature of nodes and thermal hazard label of HPC room; for instance, in sample-1 of timestamp 2020-01-01 00:00:00 (red line in figure 4.9a), the input data is the temperature of nodes in the room from 2019-12-31 18:00:00 to 2019-12-31 23:59:00, while with the prediction horizon of 6 hours, the label created based on the temperature of the nodes in the room from 2020-01-01 00:00:00 to 2020-01-01 5:59:00. Sample-2 of timestamp 2020-01-01 01:00:00 (green line in figure 4.9a), which has an input (temperature of nodes from 2019-12-31 19:00:00 to 2020-01-01 00:59:00) and a label (with the prediction horizon of 6 hours it used the temperature of nodes from 2020-01-01 01:00:00 to 2020-01-01 6:59:00 to generate the label) these two samples have around 83% same temperature data in the input. In the figure 4.9a, we report the percentage of overlap of input data of sample-1 of timestamp 2020-01-01 00:00:00 with other neighbor samples. This overlap or replicated data percentage increases with a decrease in the time distance of two samples. With a prediction horizon of 6 hours, assume sample-1 2020-01-01 00:00:00 is in the test dataset, and sample-2 2020-01-01 01:00:00 is in the training dataset considering the overlap as mentioned earlier, around 83% input data of the test sample is in the training sample, and with increase the number samples, this overlap will boost.

So we decided to cancel overlapped data from the training and test dataset and check the training results. Three overlap cancellation scenarios are studied: (a) Cancel the samples of the training dataset that, (i) temporally, are after the test samples and (ii) if they have overlap. For example, if sample-1 2020-01-01 00:00:00 is in the test dataset, the green part of the dataset will be canceled from the training dataset (figure 4.9b). Although the samples in the left part of sample-1 have an overlap with sample-1, they did not propagate the temperature of the

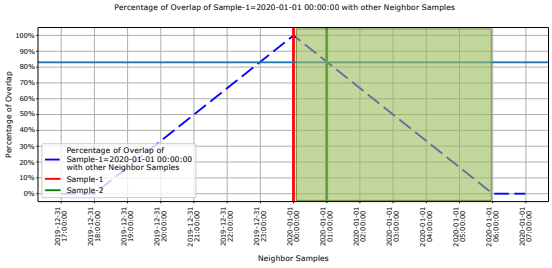
future of sample-1 (future with overlap) in the training dataset. (b) Cancel the samples of the training dataset if they have overlap with test samples (from both sides, green and blue parts figure 4.9c) (c) Cancel the samples of the training dataset if they have more than 50% overlap with test samples (from both sides, green and blue parts figure 4.9d).

Barchart 4.10 shows the accuracy and F1-score of training of model with different overlap cancellation scenarios as well as the training of the model without overlap cancellation. The last value predictor can be a baseline to compare the results. (i) The blue bars show the results of the model's training without any cancellation of the samples of the training dataset, as it's clear *there is no significant difference between the prediction and detection, and increasing of prediction horizon has no effect on the results.* This is not reasonable in a real scenario, and the increase of the prediction horizon should degrade the results. *Therefore, considering the accuracy and F1-score as performance metrics of the model for time-series data while selecting the training and test dataset randomly* (with tools like `sklearn train_test_split`) *is inadequate. The samples of time-series data are a sequence of the data that can overlap with each other, which can be a source of the error in the study.* (ii) Orange bars show the results of both sides' cancellations. *Due to this substantial reduction in the training dataset, the model's performance reduces significantly.* (iii) with the right side cancellation or cancellation of the samples with more than 50% overlap (brown, green bars), the model has reasonable results.

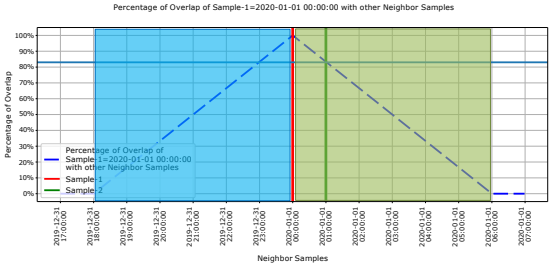
For this experiment, the TCN model (Figure 4.7) employed 1D Convolutional layers, and the input data structure is depicted in figure 4.8.



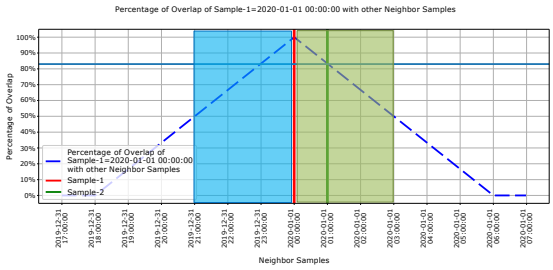
(a) Overlap Percentage of a Sample.



(b) Right Side Overlap Cancellation.



(c) Both Sides Overlap Cancellation.



(d) Overlap Cancellation of the Samples With More Than 50% Overlap.

Figure 4.9: Overlap and Overlap Cancellation.

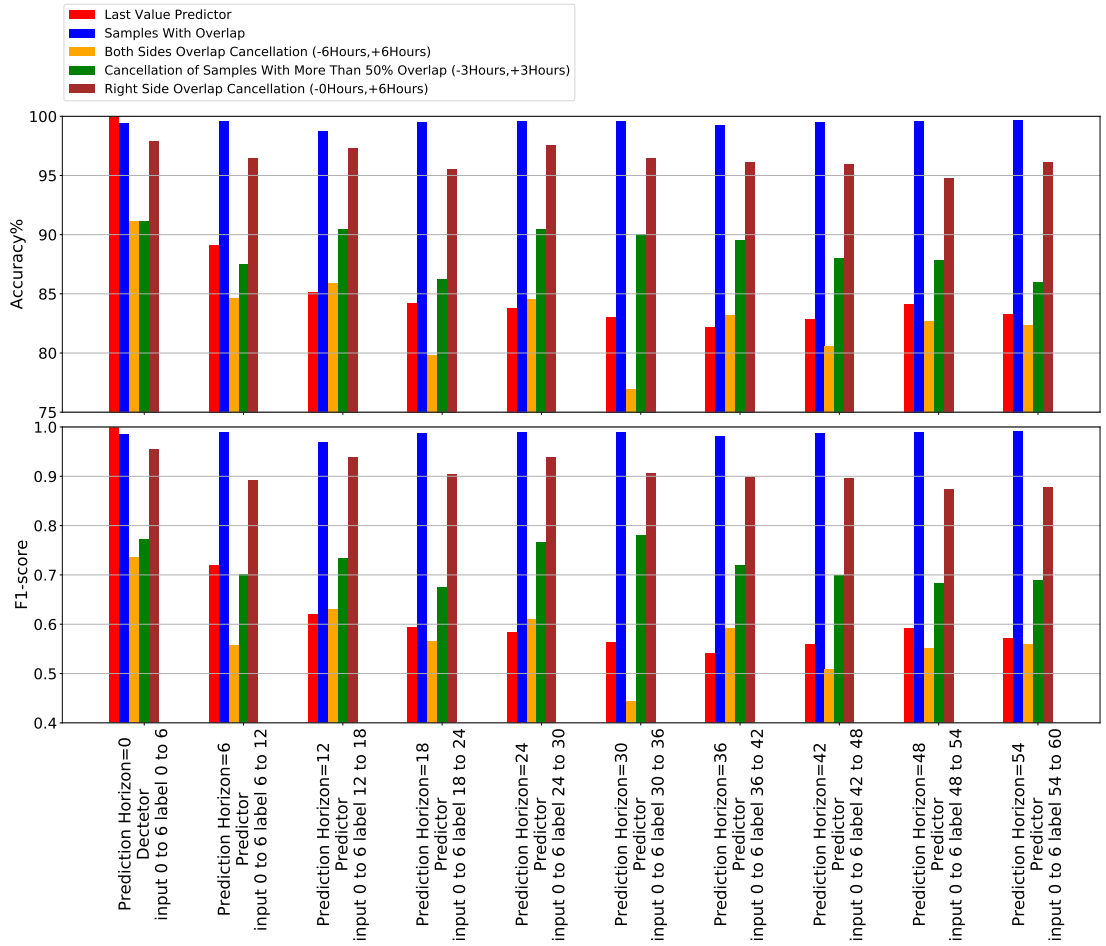


Figure 4.10: Accuracy and F1 score with and without overlap.

4.6.2 Experiment 3: Time-separate Test Dataset

The concern of randomly selecting training and test dataset is related to the fact that random selection destroys the chronological order of the training and test samples. i.e., in the test dataset, some samples are chronologically before the training samples. But in the real case implementation, the model train with data of the past to predict the future.

To simulate a real case scenario, we trained the TCN with only May 2019 data and validated the model in the first week of June 2019. Our TCN with a time-separate test set (Experiment 3) achieved an F1-score of 0.74, which is 0.24 lower than Experiment 1 with a random test set while outperforming results of overlap cancellation (Experiment 2). Such degradation is due to the random selection of the test set in Experiment 1 for which similar samples are present in

ML-Model	Recall	Precision	F1-score
TCN - Time-separate Test Set	**0.79**	0.70	**0.74**

Table 4.5: Results of Experiment 3.

the training and test set. Experiment 3 is, however, closer to the real usage of the predictive model. We suspect that the limited accuracy of Experiment 2 is caused by (1) the limited set of nodes considered for the prediction, and (2) a non-stationarity in the thermal effects that is not captured if we use only past data to predict the future.

The limited performance of Experiment 3 motivates us to try in the following experiments to mitigate causes for the performance loss and propose advanced models for achieving higher performance.

For this experiment, the TCN model (Figure 4.7) employed 1D Convolutional layers, and the input data structure is depicted in figure 4.8.

4.6.3 Experiment 4: Input Selection/Node Selection Randomly Selected 72 Nodes as Input and TCN with 1D Conv. Layers

Label Generating with a Subset of the Dataset:

The size of the TCN model (trainable parameters of the model) will be substantial if all the nodes' temperature data are used as an input of the TCN model. So the temperatures of 72 nodes (nodes one rack 4 NodesperChassis * 18 ChassisperRcak) are used as input of the ML-Models in the previous experiments. Assume the TCN model employs the identical role-based statistical approach with two thresholds (4.4.1) applied in the labeling approach to detect the thermal hazard. Considering the assumption, what will happen if it uses fewer (72 instead of 3312) nodes' data as input of the TCN model? We know that TCN does not use the rule-based statistical approach for thermal hazard prediction, so it should/could predict with acceptable performance with fewer input data. The binary labels are generated utilizing the same statistical tool 4.4.1 by a subset of the nodes regarding the earlier assumption. The following results are achieved given that the labels generated by 3312 nodes is the ground-truth label, and the labels generated with 72 nodes are detected labels. TN = 35895 , FP = 1581, FN = 2260, TP = 10837, F1-score = 0.85, Accuracy = 92%. So the labeling approach with data of 72 nodes has an 8% reduction in the accuracy of labeling the room. This experiment shows that if the TCN model just learns the role-based statistical approach in detecting the label, how much it will successfully label the room with the same algorithm but with fewer inputs. Therefore, If the TCN model uses the same rules applied

in the labeling approach in the best case situation, it will have 92% accuracy in labeling the room. We know the TCN has its own approach and may reach better accuracy ($> 92\%$) with 72 nodes. As mentioned, to control the size of the TCN model, the model's input is restricted; therefore, it is essential to select the most informative inputs. In the previous experiments, a rack was randomly selected, and the model's inputs were the nodes' temperature of the rack. The following experiments are done to find a better approach to selecting the more informative nodes, i.e., the goal is to select a subset of nodes that create labels similar to the original labels (the original labels created by all nodes 3312) as much as possible. So most informative nodes mean a subset of nodes with the lowest number of nodes creates the labels that are very similar to the room's original labels. I did some experiments to find the best method to select the nodes. Table 4.6 reports the result of different approaches. For example, I divided the nodes into three groups bottom, center, and top, and from inside each group, we randomly selected 36 nodes in total 36*3. Then I selected the same number of nodes (3*36) from the entire room completely randomly without grouping the nodes to the bottom, center, and top. Also, I selected the nodes which have the highest correlation with the rooms' label; I found that completely random selection has a high F1-score.

Node Selection Approach	**Number of Nodes**	**F1_score**	**Accuracy**	**Confusion Matrix**
From 24 racks, node 0 from chassis 1(bottom), chassis 9(center), and chassis 18(top)	72	0.85	0.92	[[35904, 1581], [2251, 10837]]
Select nodes with highest correlation with room label	72	0.77	0.89	[[36713, 772], [3846, 9242]]
From 36 racks, node 0 from chassis 1(bottom), chassis 9(center), and chassis 18(top)	108	0.86	0.92	[[35888, 1597], [1992, 11096]]
Random selection of racks, node 0 from chassis 1(bottom), chassis 9(center), and chassis 18(top)	108	0.85	0.92	[[35660, 1825], [1892, 11196]]
Divided the nodes to three groups bottom (chassis 1 to 6), center (chassis 7 to 12), and top (chassis 13 to 18), and from inside each group, randomly selected 36 nodes.	108	0.91	0.95	[[36936, 549], [1689, 11399]]
Completely Random Selection of Nodes	108	0.91	0.95	[[36936, 548], [1690, 11399]]

Table 4.6: Different Approaches for Node Selecting.

Figure 4.11 reports in the y-axis F1-score (red line) and Accuracy (blue line), while the x-axis shows the number of randomly selected nodes.

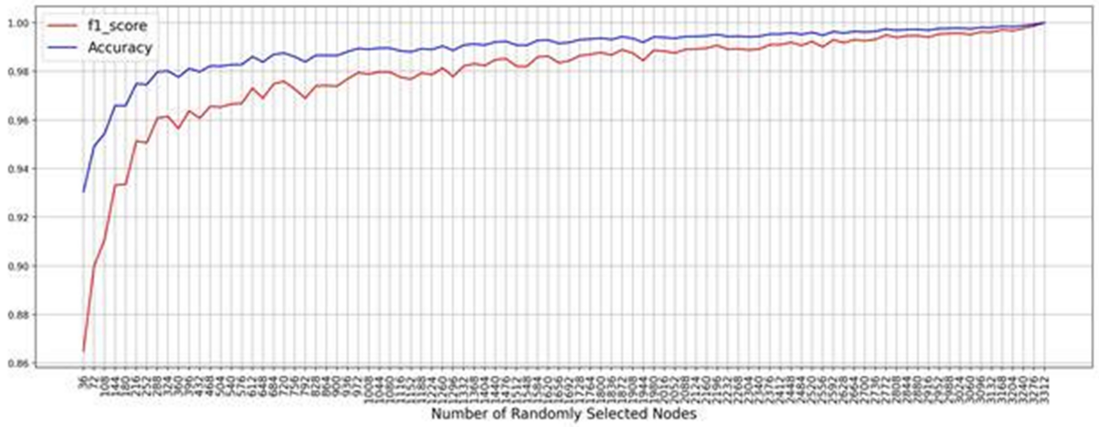


Figure 4.11: Accuracy and F1 score for Randomly Nodes Selection.

We empirically find that random selection of 72 nodes is the best method for input setting from these experiments because it collects the different nodes' temperature data which spread in different locations of the HPC room. This random selection creates a better representative of the entire room than all nodes of just one random rack.

Randomly Selected 72 Nodes as Input and TCN with 1D Conv. Layers

It has been shown that randomly selected 72 nodes from the HPC room will provide more information than 72 nodes of one rack, so the nodes that provide the TCN model's input data are selected randomly in this experiment. The labels are generated utilizing the temperature of all nodes in the HPC room. For this experiment, the TCN model (Figure 4.7) employed 1D Convolutional layers , and the input data structure is depicted in figure 4.8. TCN model trained with data of May 2019 and test is done on the data of the first week of June 2019 and table 4.7 reports the results of this experiment. Random selection of the nodes (inputs) makes a slight improvement (F1-score from 0.74 enhanced to 0.77) in the performance of the predictor TCN model.

Model	**Test Period**	**TN**	**FP**	**FN**	**TP**	**lr**	**sum**	**Accuracy%**	**Precision**	**Recall**	**F1-score**	**MCC**
Detector	First Week After Training	4934	555	1249	2983	0.01	9721	81.44	0.84	0.70	0.77	0.62
Predictor	First Week After Training	2912	2217	343	4249	0.01	9721	73.66	0.66	0.92	0.77	0.52

Table 4.7: Results of Experiment 4.

4.6.4 Experiment 5: Randomly Selected 72 Nodes as Input and TCN with 2D Conv. Layers

Experiment 4 shows that randomly selected 72 nodes from the HPC room will provide more information than 72 nodes of one rack, so the nodes that provide the TCN model's input data are selected randomly in this experiment. The labels are generated utilizing the temperature of all nodes in the HPC room. To reduce the size (trainable parameters) of the TCN model (Figure 4.7) instead of 1D Convolutional (1DConv.) layers, the TCN model employed 2DConv layers, and the input data structure is depicted in figure 4.12. With 2D Conv. layers, we have more control over the model's size. So with a lower number of parameters, it is possible to reduce the size of the training dataset, so the sampling rate of data is modified from 1 to 10 minutes. The training time will increase because of the reduction in the training dataset size. TCN model trained with data of May 2019 and test is done on the data of the first week of June 2019, etc., and table 4.8 reports the results of this experiment. This modification in the architecture of the TCN model and sampling rate indeed improved the performance of the predictor and detector models. F1-score of detector enhanced from 0.77 to 0.88 while for predictor from 0.77 to 0.82.

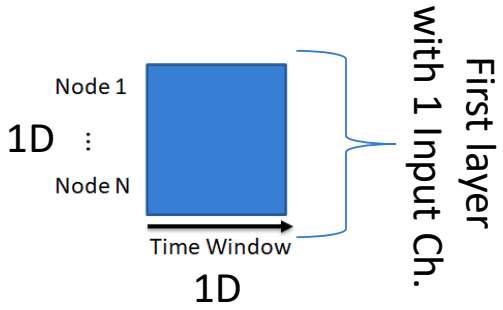


Figure 4.12: Input Data Structure of the TCN Model with 2DConv. Layers.

Model	Test Period	TN	FP	FN	TP	lr	sum	Accuracy%	Precision	Recall	F1-score	MCC
Detector	First Week After Training	277	132	18	546	0.1	973	84.58	0.81	0.97	0.88	0.69
Predictor	First Week After Training	215	190	40	528	0.01	973	76.36	0.74	0.93	0.82	0.52
Predictor	Second Week After Training	87	506	23	357	0.01	973	45.63	0.41	0.94	0.57	0.13
Predictor	Third Week After Training	248	612	31	82	0.01	973	33.92	0.12	0.73	0.20	0.01
Predictor	Fourth Week After Training	389	249	159	176	0.01	973	58.07	0.41	0.53	0.46	0.13

Table 4.8: Results of Experiment 5.

4.6.5 Experiment 6: Power Consumption (of Randomly Selected 72 Nodes) as a Second Input Channel of TCN with 2DConv Layers

Power consumption of the nodes is the primary source of the heat generation in the room, so the TCN model's performance with adding the power consumption as a new channel of the input of the TCN model in this experiment is studied. The TCN model will have two channels in input (inlet temperature and power consumption) in this experiment. Before adding the power consumption of the nodes as a second input channel, some analysis is done to find the best lead or lag between two input channels, i.e., employing the delay between the power consumption and heat-generating in feeding the input data to the TCN model aiming to improve the performance.

The medians of power consumption, inlet temperature, and outlet temperature of all nodes in each timestamp are computed then the correlation of these medians is calculated. Table 4.9 reports the correlation matrix of Tin (Inlet temperature), Tout (outlet temperature), power, and deltaT, which is the difference between the outlet and inlet temperature. Figure 4.13 shows the pairwise relationship between power and temperatures (scatter plot matrix). (i) It is clear that inlet temperature Not correlated with power consumption. It is more under the control of the CRAC units. (ii) Outlet temperature is highly correlated with inlet temperature and next power consumption. (iii) There is lower thermal variation in the inlet temperature than the outlet. (iv) deltaT (Difference of the inlet and outlet temperature) is more related to the outlet temperature and power than the inlet temperature.

	Power	**Tin**	**Tout**	**deltaT**
Power	1	0.4	0.68	0.79
Tin	0.4	1	0.89	0.27
Tout	0.68	0.89	1	0.68
deltaT	0.79	0.27	0.68	1

Table 4.9: Correlation of the Parameters.

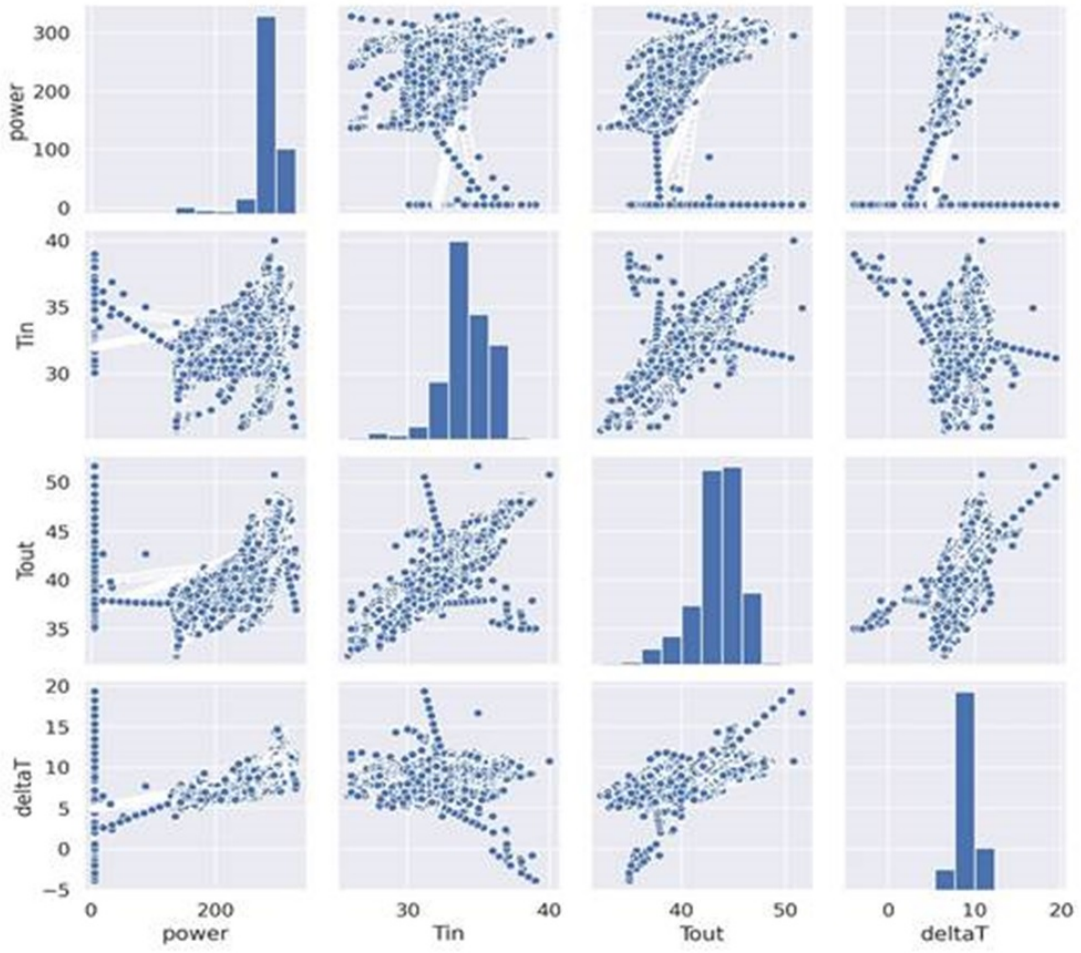


Figure 4.13: Scatter Plot Matrix of Power Consumption and Temperatures

The correlation coefficient of different parameters (power and temperatures) with different lead/lag (lead/lag in the range of 0 to 6 hours) are computed Figure 4.14, to employ the delay between the power consumption and heat-generating in feeding the input data to the TCN model aiming to improve the performance. As it is evident, the highest correlation between the different parameters is performed in zero lag. So the nodes' power consumption is added as the second input channel to the model with zero lag.

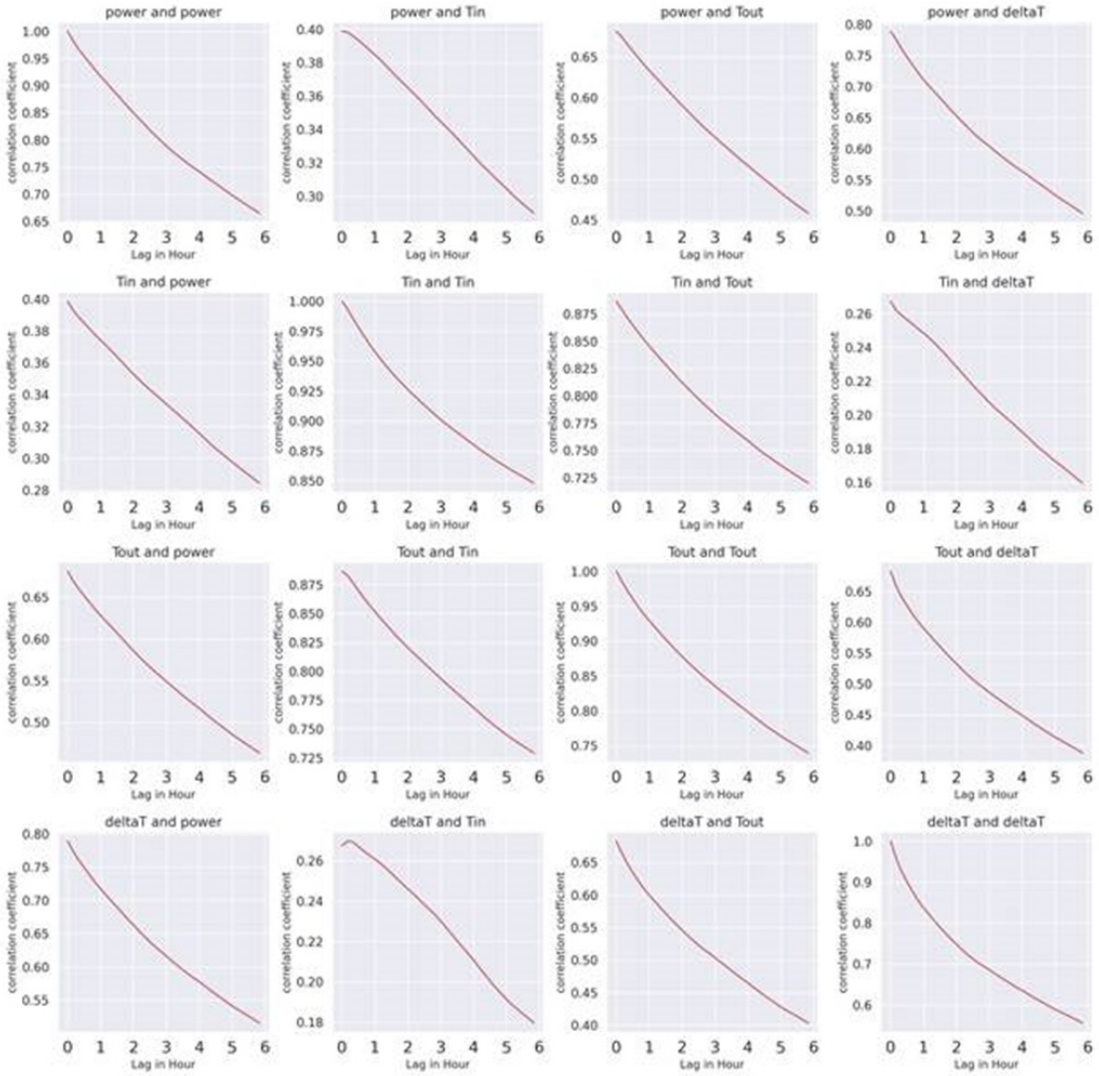


Figure 4.14: The correlation coefficient of parameters (power and temperatures) with lag/lead.

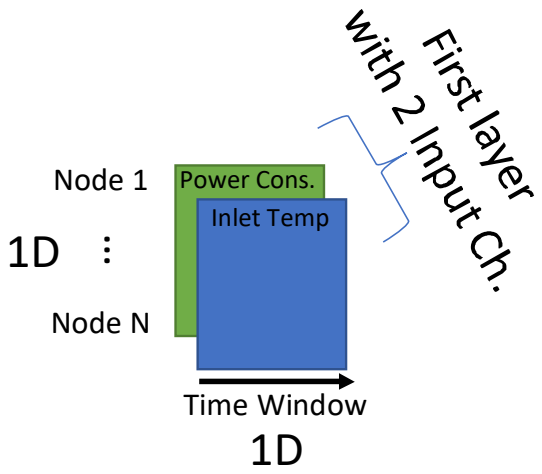


Figure 4.15: Input Data Structure of the TCN Model with 2DConv. Layers.

Power Consumption (of Randomly Selected 72 Nodes) as the Second Input Channel of TCN with 2DConv Layers

Based on the data analysis and correlation of power consumption and temperature of nodes, the power consumption (of randomly selected 72 nodes) is added without lead/lag to the TCN model as a second input channel of the TCN model, first input channel is inlet temperature of randomly selected 72 nodes. Although adding the power as the second channel to the TCN with the 2DConv layers improved the test results for the second week after training, in general, it has no significant impact on the performance of the TCN model 4.10.

Training Period	Test Period	F1-score	MCC	Accuracy	Precision	Recall
2019-01-01 to 2019-06-01	2019-06-01 to 2019-06-08	0.73	0.29	66.39	0.67	0.81
2019-01-01 to 2019-06-01	2019-06-08 to 2019-06-15	0.63	0.33	63.1	0.52	0.82

Table 4.10: Results of Experiment 6: Adding the Power Consumption as a Second Input Channel of TCN Model with 2DConv Layers.

4.6.6 Experiment 7: TCN Model with 3DConv Layers

This experiment uses the TCN model (Figure 4.7) with 3DConv layers, and the input data structure is depicted in figure 4.16. In this TCN model with 3DConv layers, 3 dimensions of the model's input are utilized to specify for 3 axes of nodes locations in the HPC room (nodes x, y, z-axis). So considering the location of the

nodes which provided monitoring data, the input data structure is created. For the time dimension of data, the input channels of the first layer are used, i.e., since the data is time-series data, each input sample of the TCN model is a sequence of the data, for each element of the sequence, one input channel is used; e.g., with a sampling rate of 1 minute, for Time Window of six hours (TW=6 Hours), 360 input channels are utilized. With this new configuration of the TCN model, it is feasible to use the inlet temperature data of all the nodes of the HPC room (*3312 nodes instead of randomly selected 72 nodes*) in the training and test dataset. The TCN model with 3DConv layers is trained with data of May 2019, and the test is done on the first week of June 2019, with a sampling rate of 10 minutes. Utilizing the 3DConv layers in the TCN model in both detection and prediction mode improved the model's performance. Table 4.11 reports results. Therefore TCN model with 3DConv layers has the highest performance utile this part of the study, and the 3D data structure of input data performs better than 1D and 2D.

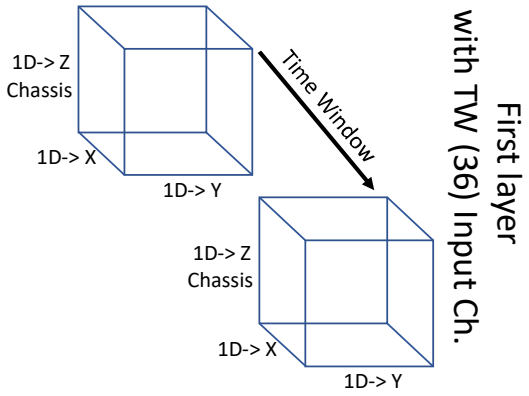


Figure 4.16: Input Data Structure of the TCN Model with 3DConv Layers.

Model	**TN**	**FP**	**FN**	**TP**	**lr**	**Accuracy%**	**Precision**	**Recall**	**F1-score**	**MCC**
Detector	360	48	14	550	0.1	93.63	0.92	0.97	0.94	0.87
Predictor	266	139	51	516	0.1	80.45	0.79	0.91	0.84	0.60

Table 4.11: Results of Experiment 7: the TCN Model with 3DConv Layers.

4.6.7 Experiment 8: Outlet Temperature of Nodes Interleaved to Inlet Dataset and Depthwise Separable Convolutions

For detection and perdition of the thermal hazard (binary labeling of the HPC room) in experiments (2, 3, 4, 5, 7), inlet temperatures of the nodes are used as input of the model. And experiment 6 empirically reveals that adding the power consumption of the nodes to the model's input as the second input channel of the TCN model did not significantly affect the performance. In this experiment, the data of all nodes in the room (3312 nodes) is used. Each node has a sensor at the inlet of the nodes, which measures inlet air temperature (that is already used in the input of TCN model); also, each node has a sensor at the outlet of the node, which measures the outlet air temperature. The outlet temperatures are augmented to the input data structure in this experiment, i.e., the outlet temperature of nodes is interleaved in the inlet temperature dataset. The TCN model with 3DConv trained with the augmented dataset for May 2019 and test is done on the data of the first week of June 2019. In this experiment, the performance of the typical TCN model and TCN model with depthwise convolution is evaluated. The experiment's results are reported in table 4.12. Interleaving the outlet temperature to the dataset of the inlet temperature reduced the performance of the TCN model. The TT in table 4.12 is training time, and the size is the number of the trainable parameters of the TCN model. Depthwise separable convolution reduces the model's size (number of trainable parameters of the model) and computation. Still, the training time increases in PyTorch implementation due to the increased number of convolutional layers (pointwise), i.e., these layers are the sequential layers, and although it reduces the number of parameters and multiplications, it increases the serial parts of the codes. So the interleaving of the outlet temperature to the input dataset and depthwise separable convolutions do not have an advantage, and for the rest of the study, as the model's input, just inlet temperature and the standard convolution are used.

Input	**Inlet Temp**		**Inlet and Outlet Temp**	
Model	**Normal Conv.**	**Depthwise Conv.**	**Normal Conv.**	**Depthwise Conv.**
Detector	F1-score=0.94 TT=1H Size=25.1K	F1-score=0.93 TT=1.5H Size=5.4K	F1-score=0.87 TT=1.5H Size=27.1K	F-score=0.92 TT=2H Size=7.4K
Predictor	F1-score=0.84 TT=1H Size=25.1K	F1-score=0.80 TT=1.5H Size=5.4K	F1-score=0.80 TT=1.5H Size=27.1K	F1-score=0.81 TT=2H Size=7.4K

Table 4.12: Results of Experiment 8: Comparison of the Results of the Standard 3DConv TCN Model and Depthwise Separable Convolutions.

4.6.8 Experiment 9: Check the Model's Performance Week by Week

The TCN model with standard 3DConv layers is trained with data of May 2019 (just inlet temperature of all nodes 3312) as a predictor, and then the test is done for subsequent weeks without retraining the model, and results are reported in table 4.13. For example, in row three of table 4.13, there is two weeks gap between the training of the model and inference. As it is clear from the results, prediction performance significantly degrades when there is a gap between training and test. The gap between the training and test is canceled by adding gap data to the training dataset to mitigate performance degradation, i.e., the training dataset increased week by week, and the TCN model is trained with data of (May + N×Weeks) and tested by the week just after the training period. Results are reported in table 4.14. Increasing the training dataset did not improve the model's performance in prediction. Some cells of table 4.14 are empty due to the division by zero error; it is impossible to calculate the associated metrics.

Test Period			Percentage of Hazard Class in Test	Accuracy	F1-score	Recall	Precision	MCC	TN	FN	FP	TP
Week	Start	End										
Week 01	2019 June 1	2019 June 7	59.42	80.45	0.84	0.91	0.79	0.60	266	51	139	516
Week 02	2019 June 8	2019 June 14	41.27	56.79	0.53	0.62	0.46	0.15	317	145	275	235
Week 03	2019 June 15	2019 June 21	11.11	88.48	-	0.00	-	-	860	112	0	0
Week 04	2019 June 22	2019 June 28	36.11	67.08	0.10	0.05	0.86	0.16	634	317	3	18
Week 05	2019 June 29	2019 July 5	4.37	95.47	0.19	0.11	0.50	0.22	923	39	5	5
Week 06	2019 July 6	2019 July 12	5.75	94.75	0.22	0.12	1.00	0.34	914	51	0	7
Week 07	2019 July 13	2019 July 19	15.77	87.04	-	0.00	-	-	846	126	0	0
Week 08	2019 July 20	2019 July 26	0.00	100.00	-	-	-	-	972	0	0	0
Week 09	2019 July 27	2019 August 2	0.00	97.74	-	-	0.00	-	950	0	22	0
Week 10	2019 August 3	2019 August 9	0.00	100.00	-	-	-	-	972	0	0	0
Week 11	2019 August 10	2019 August 16	0.00	100.00	-	-	-	-	972	0	0	0
Week 12	2019 August 17	2019 August 23	0.00	100.00	-	-	-	-	972	0	0	0
Week 13	2019 August 24	2019 August 30	0.00	100.00	-	-	-	-	972	0	0	0
Week 14	2019 August 31	2019 September 6	0.00	100.00	-	-	-	-	972	0	0	0

Table 4.13: Results of Experiment 9: Model Is Trained with Data of May 2019 and Test with Subsequent Weeks.

Test Period			Percentage of Hazard Class in Test	F1-score	Accuracy	MCC	Precision	Recall	TN	FP	FN	TP
	Start	End										
Week 01	2019 June 1	2019 June 7	59.42	0.84	80.45	0.60	0.79	0.91	266	139	51	516
Week 02	2019 June 8	2019 June 14	41.27	0.42	60.39	0.13	0.49	0.37	445	147	238	142
Week 03	2019 June 15	2019 June 21	11.11	-	87.04	-0.04	0.00	0.00	846	14	112	0
Week 04	2019 June 22	2019 June 28	36.11	0.22	59.16	-0.02	0.32	0.17	518	119	278	57
Week 05	2019 June 29	2019 July 5	4.37	0.15	90.74	0.11	0.13	0.18	874	54	36	8
Week 06	2019 July 6	2019 July 12	5.75	0.43	93.52	0.40	0.45	0.41	885	29	34	24
Week 07	2019 July 13	2019 July 19	15.77	0.04	84.77	-0.01	0.11	0.02	821	25	123	3
Week 08	2019 July 20	2019 July 26	0.00	-	100.00	-	-	-	972	0	0	0
Week 09	2019 July 27	2019 August 2	0.00	-	91.98	-	0.00	-	894	78	0	0
Week 10	2019 August 3	2019 August 9	0.00	-	100.00	-	-	-	972	0	0	0
Week 11	2019 August 10	2019 August 16	0.00	-	99.18	-	0.00	-	964	8	0	0
Week 12	2019 August 17	2019 August 23	0.00	-	99.28	-	0.00	-	965	7	0	0
Week 13	2019 August 24	2019 August 30	0.00	-	96.81	-	0.00	-	941	31	0	0
Week 14	2019 August 31	2019 September 6	0.00	-	100.00	-	-	-	972	0	0	0

Table 4.14: Results of Experiment 9: Model Is Trained with May + N×Weeks and Test with Subsequent Week.

4.6.9 Experiment 10: Cross-validation Month by Month

As shown in experiment 9, for some training and test periods, prediction results of the model are acceptable, but for other periods, it is not. So table 4.16 reported the model's month-by-month cross-validation results to see if the dataset has some lucky periods and some unlucky. The model is trained with inlet temperature data of all nodes (3312 nodes) during 2019 except one-month, test-month shown in the first column. There is no thermal hazard in March and August, so the F1-score cell in the result table 4.16 is empty due to TP=0 (accuracy is 100%). So the TCN model with 3DConv layers does not have the same performance for different months of 2019.

Test Month	F1-score	Accuracy%	MCC	TN	FP	FN	TP	Precision	Recall
January	0.81	96.53	0.79	2186	32	53	177	0.85	0.77
February	0.93	99.00	0.93	3670	8	32	287	0.97	0.90
March		100.00		4429	0	0	0		
April	0.98	99.95	0.98	4243	0	2	40	1.00	0.95
May	0.93	93.79	0.87	2387	122	153	1767	0.94	0.92
June	0.88	91.37	0.81	2613	181	189	1302	0.88	0.87
July	0.76	97.54	0.76	4148	20	89	172	0.90	0.66
August		100.00		4429	0	0	0		
September	0.80	97.69	0.80	3989	93	6	197	0.68	0.97
October	0.60	42.47		0	2548	0	1881	0.42	1.00
November	0.92	89.96	0.79	1476	304	126	2379	0.89	0.95
December	0.98	97.04	0.75	205	101	30	4086	0.98	0.99

Table 4.15: Results of Experiment 10: Cross-validation Month by Month of Detector Model.

Test Month	F1-score	Accuracy%	MCC	TN	FP	FN	TP	Precision	Recall
January	0.49	90.11	0.43	2092	126	116	114	0.48	0.50
February	0.18	89.29	0.13	3521	157	271	48	0.23	0.15
March		99.95		4427	2	0	0	0.00	
April	0.15	97.62	0.15	4174	69	33	9	0.12	0.21
May	0.85	87.47	0.75	2252	221	334	1622	0.88	0.83
June	0.62	69.57	0.38	1938	888	416	1043	0.54	0.71
July	0.21	92.57	0.18	4055	113	216	45	0.28	0.17
August		99.84		4422	7	0	0	0.00	
September	0.37	93.16	0.33	3907	155	138	85	0.35	0.38
October	0.69	68.28	0.41	1448	1092	313	1576	0.59	0.83
November	0.76	70.06	0.37	969	811	472	2033	0.71	0.81
December	0.95	91.09	0.37	143	190	204	3885	0.95	0.95

Table 4.16: Results of Experiment 10: Cross-validation Month by Month of Predictor Model.

4.6.10 Experiment 11: Decomposition of Time Series Data

As is evident from month-by-month cross-validation, the TCN model with 3DConv layers successfully predicts some months while not working for others. This

performance fluctuation can be due to the seasonality of data. The input dataset is decomposed into the trend, seasonal, and residual to deal with the seasonality of the dataset. In decomposition of the time series data, it is common to use a time-window of 12 months to check the yearly periodicity of the dataset, which requires at least data of a few years. While, in this study, just data of one year is available. So for the time-series decomposition is set time-window of one month. In Figure 4.17, the first row shows the average inlet temperature of the HPC room (original inlet temperature data), while other rows illustrate the decomposition of the data to trend, seasonality, and residual, respectively.

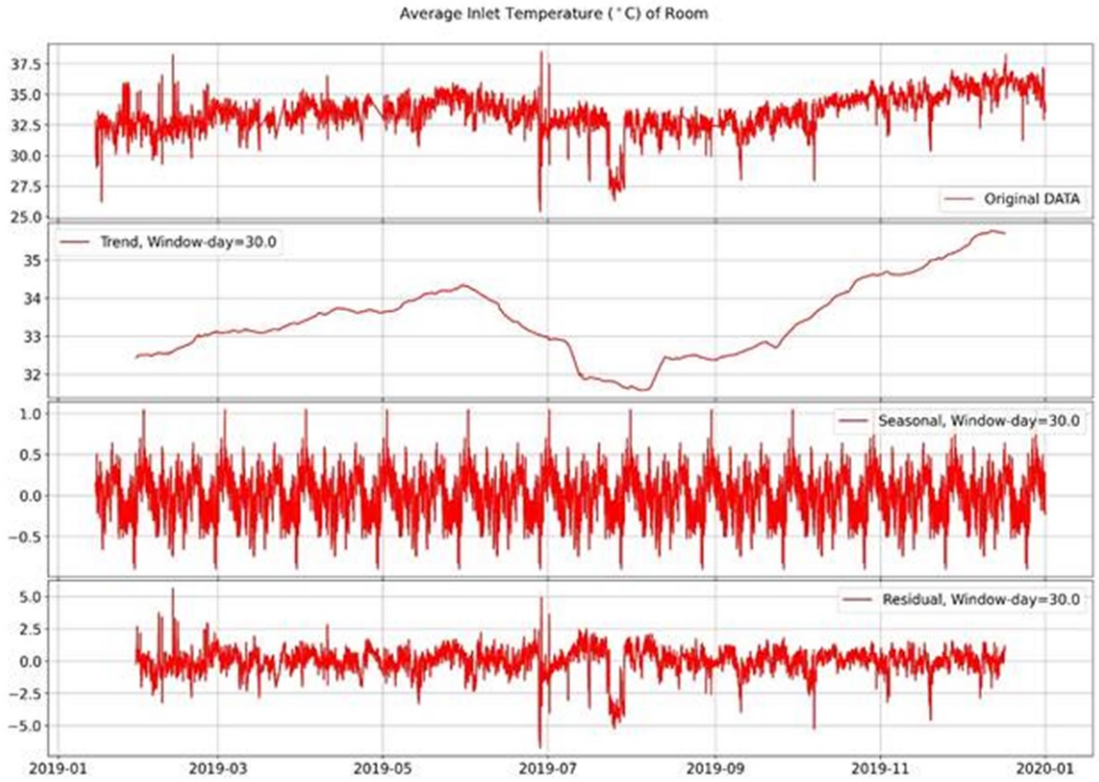


Figure 4.17: Decomposition of Time Series Data.

It is clear from the data trend (second row of 4.17) that it is possible to divide the dataset into three subsets with three trends. The model's performance is checked by training the model with transformed data (residual). The model is trained with transformed data of all nodes (3312 nodes) during 2019 except one-month, test-month shown in the first column. In table 4.17, the results are reported. The firts column shows the test month.

Predictor Decomposition of Inlet Temp Data - Month by Month Cross-validation Learning Rate=0.1, StepSize=30, Gamma=0.1, Dropout=0									
Test Month	**F1-score**	**Accuracy%**	**MCC**	**TN**	**FP**	**FN**	**TP**	**Precision**	**Recall**
1		77.43	-0.12	223	42	23	0	0.00	0.00
2	0.19	87.77	0.13	3449	229	260	59	0.20	0.18
3		97.20		4305	124	0	0	0.00	
4	0.01	91.93	-0.01	3937	306	40	2	0.01	0.05
5	0.43	65.48	0.31	2317	156	1373	583	0.79	0.30
6	0.38	68.42	0.22	2511	315	1038	421	0.57	0.29
7	0.10	87.47	0.04	3843	325	230	31	0.09	0.12
8		79.97		3542	887	0	0	0.00	
10	0.29	57.73	0.08	2170	370	1502	387	0.51	0.20
11	0.23	42.47	-0.05	1449	331	2134	371	0.53	0.15
12	0.45	34.22	0.14	177	12	1476	597	0.98	0.29

Table 4.17: Results of Experiment 11: Month by Month Cross-validation of Predictor Model Trained with Decomposed Time Series Data.

The decomposition of the time series data did not improve the model's performance (table 4.17). The proposed rule-based statistical approach for thermal hazard labeling of the HPC room utilizes one year's data; rules and thresholds are defined based on one year's data while the data has seasonal properties. Therefore proposed labeling approach has some issues due to the time-series nature of the data. This is the motivation for studying the memory-based labeling approach, which uses adjacent data to generate the label.

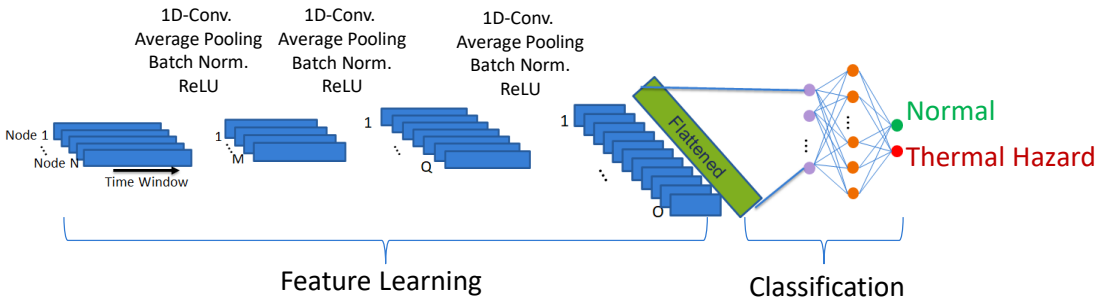
4.6.11 Experiment 12: Comparison of TCN Models with Different Convolutional Layers

Before starting the memory-based labeling, a new set of experiments is done to have a complete comparison between all TCN models and select a model for further study. In this experiment, 10% of the dataset is randomly selected as a test dataset from the whole dataset. After overlap cancelation from both sides (as mentioned in 4.6.1) of the remaining dataset, the model is trained with the remaining dataset. The results are reported in table 4.18.

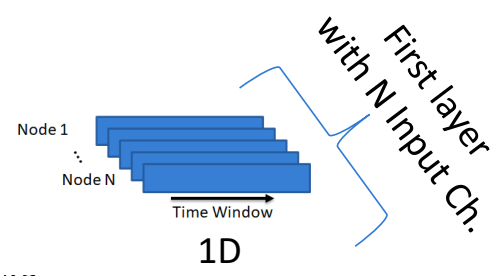
Summary of Input Data Structure (First Layer) of the TCN Model

- TCN model with 1D convolutional layers, and first layer with N Channels:
  - 1 Dimension for time
  - N_channels of the first layer = size of sensors. Sensors create a 1D-array without considering the location (x,y,z) of the sensors.
  - The drawback of this model is the size of the model: The number of weights is proportional to N_channels×kernel_size×out_channels, and this is a considerable number of weighs, in the first layer of the model.

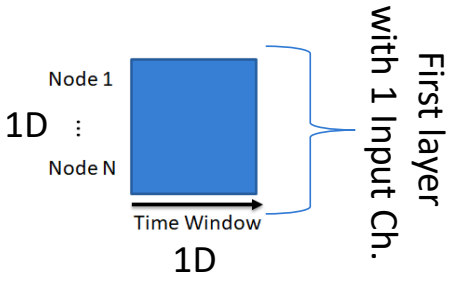
- TCN model with 2D convolutional layers, and first layer with 1 Channel:
  - 1 Dimension for time
  - 1 Dimension for sensors. Sensors create a 1D-array without considering the location (x,y,z) of the sensors
  - First layer with 1 Channel
  - More control over the number of the weights (size of the model).
  - No attention has been paid to the spatial relationship of the sensors.
- TCN model with 3D convolutional layers and first layer with 1 channel:
  - 1 Dimension for time
  - 1 Dimension for sensors. Sensors in the same height/z-axis create the 1D-array.
  - 1 Dimension for z-axis/height
- TCN model with 3D convolutional layers, and the first layer with T channels = time windows, for example, with a sampling rate of 1 minute, 360 channels:
  - 1 Dimension for sensors x-axis
  - 1 Dimension for sensors y-axis
  - 1 Dimension for sensors z-axis
  - So 3D convolutional for the 3D data of the sensors. For each timestamp/sample, it has a cubic data structure.
  - Channels of the first layer are different timestamps of Time Window = 6 Hours



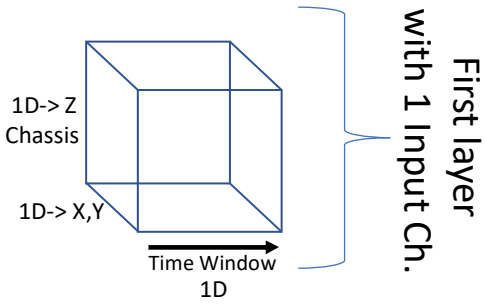
(a) TCN Model with 1DConv. Layers.



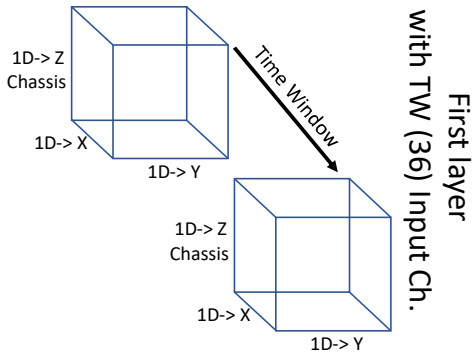
(b) Input Data Structure of the TCN Model with 1DConv. Layers.



(c) Input Data Structure of the TCN Model with 2DConv. Layers.



(d) Input Data Structure of the TCN Model with 3DConv. Layers.



(e) Input Data Structure of the TCN Model with 3DConv. Layers.

Figure 4.18: TCN Model's Architecture and Input Data Structures for Different Types of Convolutional Layser (1DConv., 2DConv., and 3DConv.).

The first three models of table 4.18 (1D3312C, 2D1C, and 3D1C) have convolution on the time dimension. The two last models (3D36C and 3D36CBig) used the first layer's input channels for the time dimension of input data. As it is clear, the best model is 3D36CBig with more than 2 million trainable parameters (weights).

In this model, the spatial relationship of data is considered. The input data is a 4D tensor, and the data's location in the tensor is related to the location of the sensors in the room. The 3D1C (row 3) has convolution in time, and the spatial relationship of data just in the z-axis is considered. This model has the lowest number of trainable parameters.

Model Name	Number of Trainable Parameters	Convolutional Layers	Number of Input Channels of First Layer	F1-score	Accuracy%	MCC	Precision	Recall
1D3312C	3,017,186	1 Dimensional 1D for Time Window	Number of Nodes = 3312	0.895	94.286	0.865	0.810	1.000
2D1C	38,315	2 Dimensional 1D for Time Window 1D for Nodes	1	0.824	91.429	0.767	0.824	0.824
3D1C	5,453	3 Dimensional 1D for Time Window 1D for Nodes 1D for Chassis (Height or z-axis)	1	0.889	94.286	0.853	0.842	0.941
3D36C	25,112	3 Dimensional 1D for x-axis of nodes 1D for y-axis of nodes 1D for z-axis of nodes	Time Windows = 36 (Sampling Rate 10 min)	0.878	92.857	0.841	0.783	1.000
3D36CBig	2,253,356	3 Dimensional 1D for x-axis of nodes 1D for y-axis of nodes 1D for z-axis of nodes	Time Windows = 36 (Sampling Rate 10 min)	0.947	97.143	0.930	0.900	1.000

Table 4.18: Results of Experiment 12: Comparison of All TCN Models.

4.6.12 Experiment 13: Memory Based Labeling

The different architectures of the TCN model and methods are investigated to improve the performance of the thermal hazard prediction of the HPC room. Although the complex architecture of TCN with 3DConv layers has an improvement in the performance of the model, the prediction results are not acceptable. The seasonality and time-series nature of the data is not considered in the proposed rule-based statistical method for thermal hazard labeling. Rules and thresholds are defined based on one year's data distribution while the data has seasonal properties; this is one of the primary sources of the problem. Based on the results of the detection model. The detection works well if it randomly selects the training and test dataset because of well coverage of all ranges of possible temperatures, but it will be biased if it does not provide the full range of data, i.e., if the training dataset does not give the full range of the temperature in training, the network does not learn the similar rules that are used to generate the label. Also, if the label is generated based on the absolute temperature threshold, even this threshold changes in a smaller period (every month); if the dataset does not give the network full coverage of the absolute threshold in the training period, the network can not learn the same rules because it gets biased. So we decided to define the regular time window to retrain the model and generate the new thresholds that we employ in the label generating. So the model will be trained based on the recent past history and have a new set of thresholds for labeling in each time window. So in each time

window, the threshold to labeling will be redefined. Defining the label based on the last week's statistics is a short-term memory extension of what we have done until now. More extended training that uses the past more samples but is less robust to the non-stationarity (need to trade-off). The small model 3D1C 5K parameter is selected to prevent overfitting, and the following experiments will be done on this model. **Memory:** *In this part of the study, memory means a period (time window) that the rule-based labeling method defines (or redefines) its Node-Threshold.*

Dataset and Label Generation:

With different memory ranges (3, 7, 14 days), and different Node-Threshold (95%, 98%, 99%) and Spatial-Temporal-Impact-Threshold (1%, 5%, 10%, 20%, 50%),the rooms labels are generated . In table 4.19, the thermal hazard's weight with different configurations is illustrated. The configurations that generate around 20% of the thermal hazard label for a whole year are selected for further study. (Red Cells).

		Memory 3 Days					Memory 7 Days					Memory 14 Days				
		Spatial Temporal Impact Threshold					Spatial Temporal Impact Threshold					Spatial Temporal Impact Threshold				
		0.01	**0.05**	**0.1**	**0.2**	**0.5**	**0.01**	**0.05**	**0.1**	**0.2**	**0.5**	**0.01**	**0.05**	**0.1**	**0.2**	**0.5**
Node Threshold	**95%**	0.89	0.62	0.4	0.17	0.02	0.87	0.54	0.31	0.12	0.02	0.84	0.52	0.31	0.14	0.03
	98%	0.82	0.47	0.26	0.1	0.01	0.75	0.35	0.16	0.07	0.01	0.69	0.34	0.18	0.07	0.03
	99%	0.78	0.4	0.21	0.08	0.01	0.66	0.26	0.12	0.05	0	0.58	0.25	0.13	0.06	0.02

Table 4.19: Weights of the thermal hazard class in the dataset with different configurations of the thresholds.

The label generated by these configurations should detect the real thermal hazards (28 June, 1 July), and all the red cells configurations are validated by real thermal hazards detection. For example, for memory=7 days Figure 4.19 shows the label around the thermal hazards:

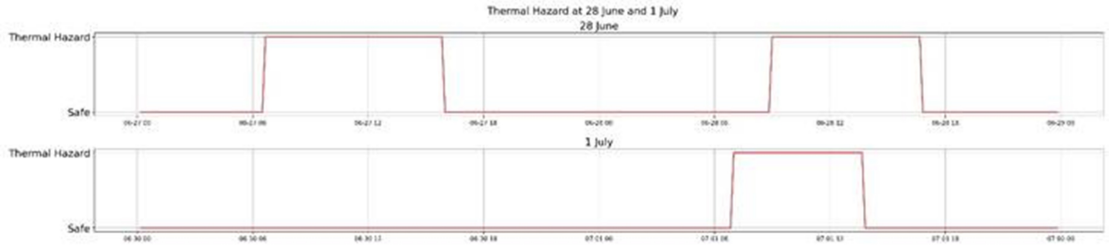


Figure 4.19: Labels generated for memory=7 Days and, Node-Threshold = 0.98, Spatial-Temporal-Impact-Threshold = 0.1.

Table 4.20 reports the monthly thermal hazard class weights. Compared with the old labeling approach, the monthly distribution of the new memory-based

labeling approach is closer to a uniform distribution. Although the 3-day memory has a better uniform distribution, 7-day memory is selected as a starting point for training the model, aiming to have enough samples in the training dataset in case of having the same duration of the training dataset and memory. With 14-days memory, thermal hazard has non-uniform distribution (October).

	Old Labeling Approach Window All Year STIT=0.05, NT=0.95	**Window=3 Days STIT=0.2, NT=0.95**	**Window=7Days STIT=0.1, NT=0.98**	**Window=14 Days STIT=0.1,NT=0.98**
January	0.08	0.12	0.21	0.06
February	0.07	0.11	0.08	0.09
March	0	0.25	0.09	0.07
April	0.01	0.08	0.08	0.12
May	0.34	0.17	0.29	0.39
June	0.25	0.1	0.05	0.06
July	0.03	0.17	0.18	0.05
August	0	0.14	0.13	0.03
September	0.02	0.2	0.15	0.1
October	0.26	0.17	0.34	0.61
November	0.44	0.24	0.24	0.28
December	0.78	0.21	0.13	0.14

Table 4.20: Monthly thermal hazard class weights with different configurations.

From table 4.18 the model 3D1C (a model with 3DConv, first layer input channel = 1, in total with 5K parameters) is selected for further study. And following experiments are done on this model.

Four Approaches for Computing the Node-threshold

As mentioned in section 4.4.1, the thermal hazard labels are generated based on the rule-based statistical method, which uses two thresholds: (i) Node-threshold (to indicate that one node in one timestamp is in thermal stress) and (ii) Spatial-temporal-impact-threshold (to account for thermal hazards' spatial and temporal continuity, it regulates the thermal hazard severity). As noted, due to the time-series nature of the dataset, it is impossible to use the distribution of the entire year to define the thresholds and rules, and it should use smaller memory for defining the rule-based statistical method for thermal hazard labeling. Therefore, in the following, four different approaches for computing, the node-thresholds are introduced. The main difference between these approaches is related to the definition of the time window (memory) that the node-threshold is computed based on the dataset of that time window. After computing node-threshold with each approach new set of labels is generated, and the performance of the TCN model (model 3D1C with 5K parameter and with 3DConv layers) trained with new labels is evaluated.

Approach 1: Node-threshold-Memory = Year2019

The first approach is the approach that was used for label generating until this part of the study, as Node-thresholds, the quantile 0.95 of nodes' inlet temperature for the whole year is computed. So Node-Threshold-Memory = Year2019. In figure 4.20, this approach is sketched. It shows the period used to compute the Node-threshold with green-box and the period when those Node-thresholds apply to generate the label in a red box. In fact, this approach has some issues due to the time-series nature of the dataset.

Figure 4.20: Node-Threshold Approach 1.

Approach 2: Node-threshold-Memory = Static-Past-Week

In this approach, the past week's data is used to compute the Node-threshold of the current week. And the shift/movement unit in the time direction to compute the Node-thresholds is one week, So each point in that week has the same threshold. In Figure 4.21, this approach is sketched. It shows the period used to compute the Node-threshold with green-box and the period when those Node-thresholds apply to generate the label in a red box. So the memory of the Node-threshold is static, which means for each week, the Node-thresholds are identical, and updates of the Node-thresholds occur week by week.

Figure 4.21: Node-Threshold Approach 2.

Approach 3: Node-threshold-Memory = Static-Current-Week

In this approach, the current week's data is used to compute the Node-threshold of the same week. And the shift/movement unit in the time direction to compute the Node-thresholds is one week, So each point in that week has the same threshold. In Figure 4.22, this approach is sketched. It shows the period used to compute the Node-threshold with green-box and the period when those Node-thresholds apply to generate the label in a red box. So Node-threshold-Memory = Static-Current-Week. So the memory of the Node-threshold is static, which means for each week, the Node-thresholds are identical, and updates of the Node-thresholds occur week by week.

Figure 4.22: Node-Threshold Approach 3.

Approach 4: Node-threshold-Memory = Dynamic-Past-Week

In this approach, the connected-last-week data is used for computing each timestamp/sample's Node-thresholds. Moreover, shift/movement unit in time direction to compute the Node-thresholds is sampling rate (10 minutes or 1 minute). The difference between this approach and the previous two approaches is that it utilizes different data for computing the Node-thresholds of two samples within one week. In contrast, the previous methods for samples of one week use the same data to compute the Node-threshold. In Figure 4.23, this approach is sketched. It shows the period used to compute the Node-threshold with green-box and the period when those Node-thresholds apply to generate the label in a red box. So Node-Threshold-Memory = Dynamic-Past-Week. Memory is dynamic, which means that for each sample, this method uses a different dataset (at least one element of the sequences is different) to compute the Node-thresholds.

Figure 4.23: Node-Threshold Approach 4.

In figure 4.24, the x-axis is DateTime, and the y-axis shows the Node-threshold (temperature in $°C$) for 4 different Node-threshold computing approaches. Each row shows one node from one random rack (3 nodes are in the same rack but at different heights -bottom, center, top chassis). As evident, Node-thresholds are constant for each week for approach 2, and approach 3 (red and blue lines), while it is dynamic for approach 4 (green line). And the black line which is constant for all year shows the approach 1.

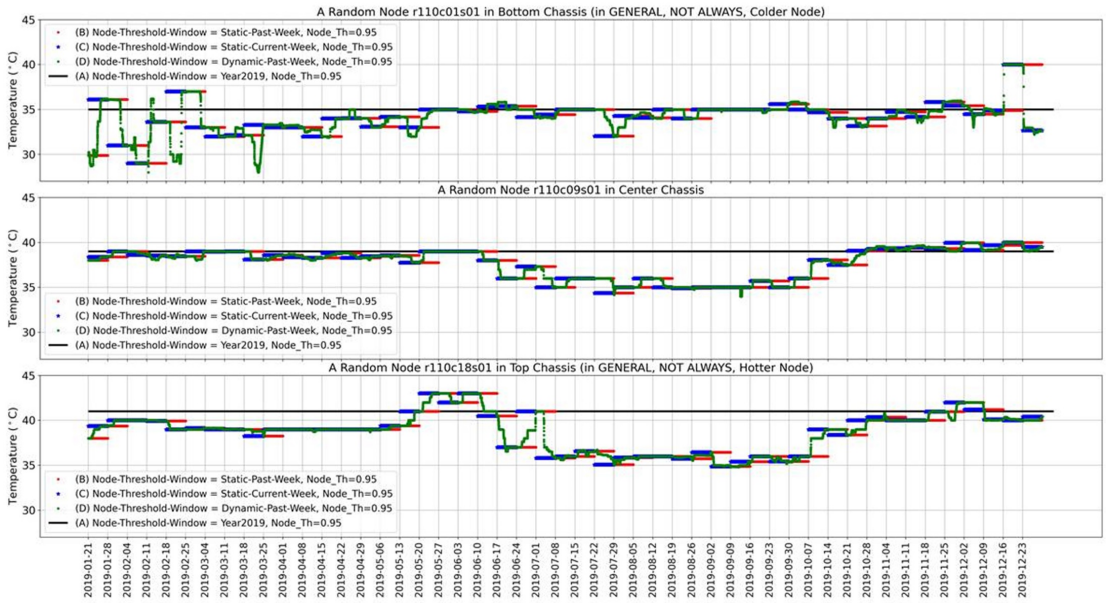


Figure 4.24: Threshold temperature of three nodes from a random rack with different labeling approaches.

Figure 4.25 reports boxplot of Node-thresholds for different weeks of the year (approach 2, 3). Each box shows the distribution of threshold temperature ($°C$) for 3312 nodes in a week. So each box is generated by utilizing 3312 temperature data (3312 nodes); for each week, there is one threshold for one node. This figure shows the variation of the thresholds for the different weeks.

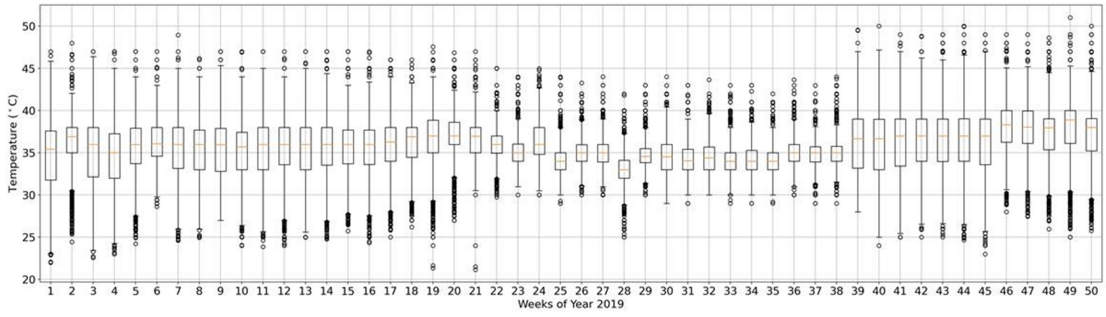


Figure 4.25: Box plot of temperature thresholds of nodes for different weeks of the year 2019.

Figure 4.26 illustrates the weekly distribution of the thermal hazards with 4 different labeling approaches. The x-axis is the date, and the y-axis shows the percentage of thermal hazards each week. For example, 10% means that

we classified the room as in thermal hazard for 10% of the time in that week. Until this part of the study, experiments employ the labeling Approach-1 (with configuration: Node-threshold-Memory = Year2019, (Node-threshold) NT=95% (Spatial-temporal-impact-threshold) STIT=5%) , which classified around 20%of times the HPC room in thermal hazard. Suppose the same thresholds are used in new approaches; it identifies rooms at thermal hazard for more than 30% of the time. Therefore thresholds should be modified for new approaches to have a reasonable percentage of the thermal hazard class in the dataset. New configuration: NT=95% and STIT=10% to have around 15% thermal hazard class in approach 4.

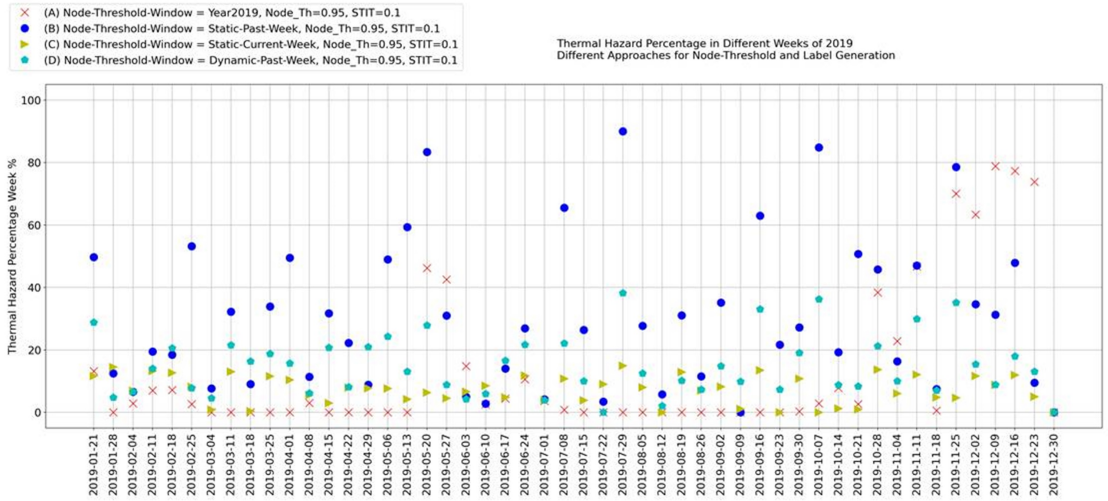


Figure 4.26: Weekly distribution of the thermal hazards with 4 different labeling approaches.

Memory Based Labeling (Approach 2: Node-Threshold-Memory = Static-Past-Week)

In this set of experiments to generate thermal hazard labels, approach 2 is employed. So Node-thresholds are updated week by week. For 10 different training and test periods, the selected model (3D1C) is trained with two different sets of hyperparameters. Then, metrics are computed by summing TP, TN, FP, FN for 20 different experiments. The results are reported in table 4.21; in the first row, the model is trained with one week's data, and the test is done in just a week after training. The train and test periods of experiments of row 1 are reported in table 4.22. Then the train and test period increase to 2 and 3 weeks ((rows 2 and 3 in table). Results are not satisfactory.

Training Dataset Duration	TN	FN	FP	TP	sum	acc%	precision	recall	f1-score	MCC	#Exp.
7 Days	12253	1231	1315	321	15120	83.16138	0.19621	0.20683	0.20138	0.107385	20
14 Days	19962	3378	4568	820	28728	72.34057	0.15219	0.195331	0.171083	0.008244	20
21 Days	16800	3294	3390	708	24192	72.37103	0.172767	0.176912	0.174815	0.008922	20

Table 4.21: Results of Experiment 13: Predictive Model with Labeling Approach 2.

	Start Train	Stop Train and Start Test	Stop Test
1	2019-01-21	2019-01-28	2019-02-04
2	2019-01-28	2019-02-04	2019-02-11
3	2019-02-04	2019-02-11	2019-02-18
4	2019-02-11	2019-02-18	2019-02-25
5	2019-02-18	2019-02-25	2019-03-04
6	2019-02-25	2019-03-04	2019-03-11
7	2019-03-04	2019-03-11	2019-03-18
8	2019-03-11	2019-03-18	2019-03-25
9	2019-03-18	2019-03-25	2019-04-01
10	2019-03-25	2019-04-01	2019-04-08

Table 4.22: Periods of Experiments.

Memory Based Labeling (Approach 3: Node-Threshold-Memory = Static-Current-Week)

In this set of experiments to generate labels, approach 3 is employed. So Node-thresholds are updated week by week. The results are summarized in table 4.23. The selected model (3D1C) is trained with different time windows (column "Training Dataset Duration" of table 4.23) for various train periods. Then, metrics are computed by summing TP, TN, FP, FN for different experiments. For example, in row 1 of table 4.23, the model is trained with one week's data, and the test is done just a week after training. Then the train duration increase to 2, 4, and 6 weeks but with the same test duration(a week just after the train). Results are not satisfactory.

Training Dataset Duration	TN	FP	FN	TP	sum	acc%	precision	recall	f1-score	MCC	#Exp.
7 Days	9488	5467	2406	2079	19440	59.5	0.3	0.5	0.3456	0.084703	20
14 Days	6098	2275	1446	873	10692	65.2	0.3	0.4	0.3194	0.094716	11
28 Days	3883	714	837	398	5832	73.4	0.4	0.3	0.3392	0.173632	6
42 Days	2566	622	458	242	3888	72.2	0.3	0.3	0.3095	0.13919	4

Table 4.23: Results of Experiment 13: Predictive Model with Labeling Approach 3.

Memory Based Labeling (Approach 4: Node-Threshold-Window = Dynamic-Past-Week)

In this set of experiments to generate labels, approach 4 is employed. So Node-thresholds are updated dynamically for each sample. The results are summarized in table 4.24. The selected model (3D1C) is trained with different time windows (column "Training Dataset Duration" of table 4.24) for various train periods. Then, metrics are computed by summing TP, TN, FP, FN for different experiments. For example, in row 1 of table 4.23, the model is trained with one week's data, and the test is done just a week after training. Then the train duration increase to 2, 4, and 6 weeks but with the same test duration(a week just after the train). Results are not satisfactory.

Training Dataset Duration	TN	FP	FN	TP	sum	acc%	precision	recall	f1-score	MCC	#Exp.
7 Days	5396	605	689	114	6804	80.982	0.16	0.1	0.1	0.0432	7
28 Days	3483	397	853	127	4860	74.28	0.24	0.1	0.2	0.0353	5

Table 4.24: Results of Experiment 13: Predictive Model with Labeling Approach 4.

4.7 Summary

This section suggests a framework for thermal hazard prediction, which encompasses data query and preprocessing, model training, and final model inference, which provides the prediction. The thermal hazard predictor is a model that, based on time series data of computing nodes' sensors, predicts if a thermal hazard will happen in the room in the next hours. Input data are the time series of nodes' temperature, and the output is a binary classification: likely forthcoming hazard or not. The dataset does not contain any labels, so this study used statistical analysis of real thermal hazard data from the CINECA Marconi KNL (largest HPC cluster of CINECA at 2019) Room F to characterize thermal hazards in the HPC room. Then based on this analysis, a rule-based statistical method was defined to create labels. The proposed rule-based statistical method is composed of two thresholds (i) Node-threshold (to indicate that one node in one timestamp is in thermal stress) and (ii) Spatial-temporal-impact-threshold (to account for thermal hazards' spatial and temporal continuity, it regulates the thermal hazard severity). Different classical machine learning and DL tools were investigated and empirically shown that the proposed thermal hazard predictor, namely a Temporal Convolutional Network (TCN), outperformed non-deep models and LSTM. Some techniques are introduced/examined to deal with issues like; samples-overlapping of time series data or imbalanced datasets. I showed that thermal hazard prediction

has many challenges in real case scenario implementation. Although the TCN model works well in the research phase (selecting the test dataset randomly like what is common in most research and papers), it will have substantial performance degradation in real implementation. This study investigates enough complex TCN models with different convolutional layers, and input data flow to improve the model's performance. The memory-based approaches for labeling the thermal hazard were investigated. During this study, we had meetings with one of the most powerful HPC cluster's sys-admin (Marconi A2 HPC clsuter of CINECA ranked 21th in June 2019 [2] Top500 list) to understand the situation better and find a solution to implement this thermal hazard prediction framework in a real in-production large-scale HPC cluster. Based on the study results, I find that due to the dataset's complexity, the monitoring signal's dynamism, manual update of the cooling setpoints, activation of the free cooling system, etc. it is essential to use a more sophisticated anomaly detection method (or thermal hazard detection method), i.e., a rule-based statistical method with just node level data is insufficient for thermal hazard perdition for real in-production HPC rooms, and I should add the metrics of HPC room level facilities like RDHX, CRAC unit, etc. to the dataset and improve the anomaly detection approach. The study results motivated us to collect essential metrics of the Marconi100 HPC cluster and Room F from April of 2021 and utilize this big dataset to develop a sophisticated anomaly detection tool in the next chapter of the thesis.

Chapter 5

Thermal Anomaly Detection

5.1 Overview

Anomaly detection is an important research topic and is widely applied in diverse fields, predictive maintenance in the industry [78], security [79, 80], fault detection in HPC systems, and Datacenters [81, 82], finance [83], sensor networks [84] and the internet of things [85], etc. Although anomalies in HPC systems, like other domains, are very rare events, anomaly detection is vital due to the significant harmful consequence of anomalies. In this study, an anomaly is a suspicious pattern in the monitoring signals of the HPC room, which can initiate due to; an inappropriate working of the cooling system or subsystem, or inconsistency between the different cooling systems in the HPC room, or abnormal computing demand, or extreme hotspot during the summer that can affect the capacity of cooling systems, or fast variation of some monitoring signals that could not support by other signals, or it can be complex temporal and/or spatial relation of the different monitoring signals which is not clear for human expert but can be identified by machine learning approaches. The severity of the anomaly can be different. It can be due to just some transient variation of the monitoring signals, which has no significant thermal effect at the node level and the temperature of the node level and room level is under control, or it can be very severe, which create a thermal hazard and even outage of part or whole of the computing capacity of the HPC system. In order to the non-conservative operation of the HPC system, different thresholds that have a critical role in defining the anomaly should adjust accurately.

In this study of anomaly detection, the monitoring data of the real HPC Room at CINECA, which hosts Marconi 100 (One of the most powerful computing systems worldwide, ranked 9th in the TOP500 list in June 2020 [3]) is adopted. This study is done based on real data analysis of in-production HPC cluster and HPC room facilities (CRAC units, RDHX, etc.) and never used any synthetic data or artificial

anomalies. To collect the monitoring data, we used a holistic monitoring system, ExaMon (2.3), one of the state-of-the-art HPC monitoring systems developed by other members of our group at the University of Bologna. To anomaly detection in the monitoring data of computing nodes and HPC facilities, I utilized two tools: 1- Rule-based Statistical Method (Flags) and 2- Semi-supervised Machine-Learning-based Method (Autoencoder). The rule-based statistical method consists of a set of events, which hereby I refer to as flags (in total, 281 flags for one rack with 20 computing nodes and the room's facilities) that can identify the samples with abnormal patterns or variations of the monitored signals. For the ML-based methods, I focused on two semi-supervised deep-learning approaches, the Multilayer Perceptron Autoencoder (MLP-AE) and Long Short-Term Memory Autoencoder (LSTM-AE), which reconstructs the input at the autoencoder's output. Different configurations for the training dataset to find a suitable subset of the dataset for the training of the autoencoder are investigated. Anomalies are detected by comparing the reconstruction error with a threshold. I defined the threshold based on the statistical distribution of the training dataset. Finally, the performance of the introduced approach and tools in anomaly detection at room level and subsystem level by a detailed study of monitoring signals is verified. The labeling results are validated with real/physical failure on 28-07-2021.

5.2 State of the Art

By approaching exascale computing systems [86], the importance of anomaly detection research topics in HPC systems increases [87]. In the HPC system, anomalies reduce the performance and increase the cost by affecting the computing capacity and energy of HPC systems. Anomalies are reported due to network contention [88], shared resources contention [89, 90], hardware-level problems [91], memory [92], CPU [93], and cooling system failure [75, 94]. Some researchers used the rule-based analysis to define the anomalies; researchers manually, or based on the statistical analysis or recommendations, set thresholds for system metrics [95, 96]. The monitoring data of the system and component is investigated to find the correlation between the different problems (like detecting I/O congestion and out-of-memory) and causes by other studies [92,93]. Although rule-based analysis is easy to implement, due to the size of the monitoring data of HPC systems, rule-based analysis or manual root cause analysis is an inefficient approach, so the ML-based approaches are widely used by researchers for anomaly detection [87, 97–99]. Based on the ML-based approach proposed in [100], the authors introduced an end-to-end machine learning framework in paper [87] that diagnoses performance anomalies on compute nodes at node-level and job-level. The authors of [101] used Long short-term memory (LSTM) neural network to detect running applications with

suspicious behavior to increase the system's efficiency. In [102], the authors present an ML-based predictor framework for real-time node failures. They used log collections of 4 HPC systems to offline training to extract the failures patterns. Authors in the series of studies provided [81, 98, 103, 104] fault classification and anomalies detection; first introduced supervised methods which mostly learn trivial correlations (i.e., idleness equals to failure) without anticipation capability [103]. Then they proposed a semi-supervised method. Using the only semi-supervised method has suboptimal performance (high number of false positives) [98, 104]. At [81], they propose combining a semi-supervised and a supervised model, in which both models are accurate (with an F-score around 0.86). This approach can anticipate anomalies around 1 hour before the system administrator registers the anomaly.

Most studies investigate the anomalies employing one of the statistical rule-based or ML-based methods for anomaly detection at the application or node levels without considering the room level facilities, which can create severer anomalies than the application and node levels. This study employed a combination of statistical rule-based and deep learning methods on a big dataset, composed of node-level metrics as well as room-level facilities metrics (like two different sophisticated cooling systems metrics, total power consumption metrics of different parts of HPC room collected from Modbus, etc.) to anomaly detection at room-level, node-level, system-level, and subsystem-level. Anomalies are infrequent, so some studies employed synthetic anomalies at the test state and out of production HPC. In this study, all the data is collected from in production HPC cluster (one of the most powerful computing systems worldwide), and, finally, the study and approaches are validated with real physical failure. So, to the best of our knowledge, this is the first time that a study employed both statistical rule-based and ML-based (semi-supervised) tools on different levels monitoring data of in production HPC cluster (collected by one of the states of the art holistic monitoring system at the different levels node, room facilities, etc. which developed in our group in University of Bologna by other members of the group) to anomaly study at different levels of node, system, subsystem, and HPC room. This study included a detailed study of the real thermal failure, which caused the outage of half of the computing nodes of the HPC cluster.

5.3 Dataset

This study is done on the monitoring data of the Room-F of the CINECA datacenter, which hosts the Marconi100 (More Information 2.2.3) cluster. The Marconi100 is a Tier-0 cluster ranked 9th (June 2020 [3]) in the list of the most powerful supercomputers worldwide [3]. In figure 5.1, the schematic of the HPC room's

facilities and a rack is depicted. Different metrics like inlet, PCIe, CPU [0,1], and GPU [0,1,2,3] temperatures, fan speed, power supply are studied for the rack nodes. The racks are equipped with RDHX, and the metrics: water flow rate, inlet, and outlet water temperature, position of the three-way valve, and delta temperature of the water are studied. Moreover, there are 6 CRAC units in the room, metrics like compressor utilization, free cooling, free cooling valve open position, fan speed, return, and supply air temperature are studied. From the Modbus, we extracted the metrics: total power consumption of ICT, total power consumption of RDHX pumps, total power consumption of chillers, total power consumption of CRAC units. In total, for one rack with 20 nodes and room facilities, 242 metrics are collected. The data collection period starts from 2021-04-08 ends on 2021-08-21.

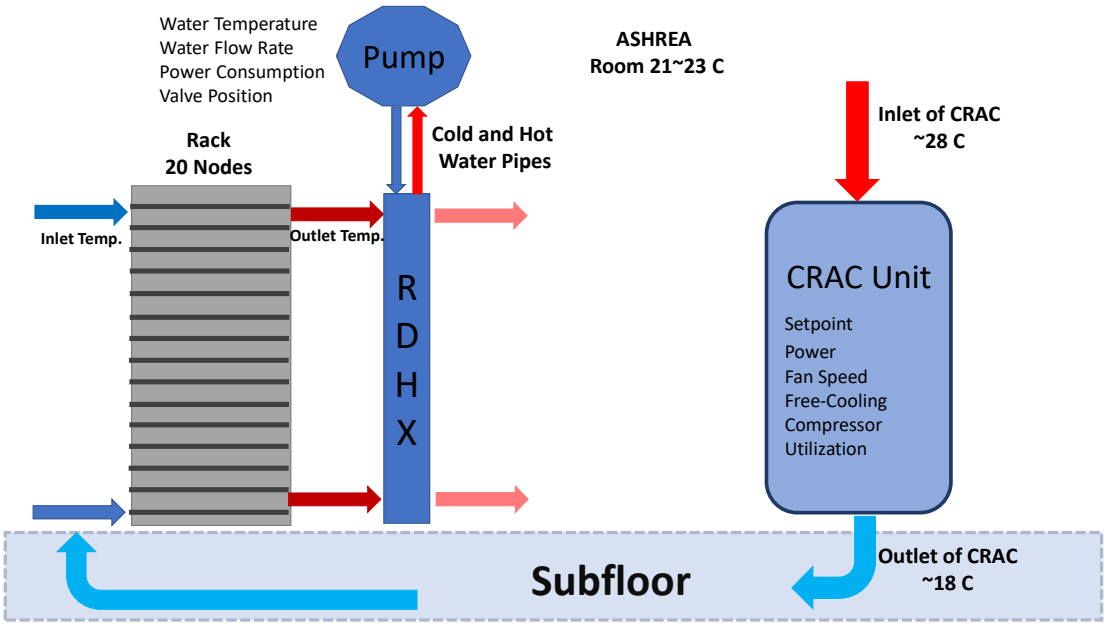


Figure 5.1: Schematic of the HPC Room's Facilities and a Rack.

5.4 Rule-based Statistical Method (Flags)

Figure 5.2 shows the two monitoring signals blue line on the right y-axis shows the total power consumption of the chillers, while the red line in the left y-axis displays one random node's inlet temperature. The green zone demonstrates part of the signal that we know cluster is in normal production; in contrast, the red zone is the failure zone. The red line (Inlet temperature of a node) in the abnormal zone, the red zone, reaches a very high value compared to the normal zone; this is the (1) **Constraint Violations** condition for this signal. Moreover, the blue line (Total

power consumption of chillers) has a high variation in a small time interval in the abnormal zone (② **High Derivative**). Considering these two abnormal patterns of the monitoring signal in the definitions of the flags, we used High Derivative and Constraint Violations widely to find the anomalous and suspicious patterns. A set of flags is defined for all the critical metrics of computing nodes of one rack and room's facilities (CRAC Units, RDHX, Modbus, etc.), as mentioned in detail in the 5.3.

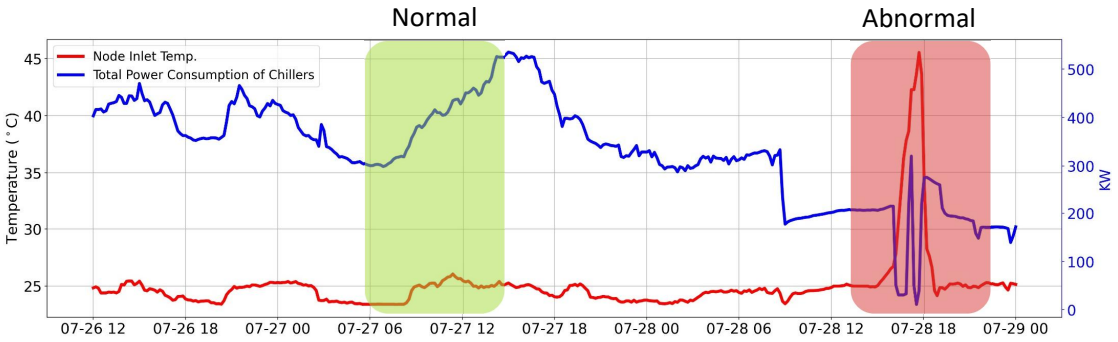


Figure 5.2: Comparison of Normal and Abnormal Signals.

Two main groups of flags: ① **Constraints violation**; $\mathcal{M}(t) > threshold$ or $\mathcal{M}(t) < threshold$, $\mathcal{M}(t)$ is a metric, and t shows time. and ② **High derivative**; $\mathcal{M}(t) - \mathcal{M}(t-1) > threshold$ or $\mathcal{M}(t) - \mathcal{M}(t-1) < threshold$. Group ① has three subgroups: ⓐ Cooling shortage, indicating a part of the cooling system reached its max or close to the maximum capacity ($\mathcal{M}(t) > threshold$), or it is due to the failure of one part ($\mathcal{M}(t) < threshold$). ⓑ Thermal/ASHRAE, which shows CPU/GPU or inlet temperature of the node, violated the ASHRAE recommendations. ⓒ The computing load that shows, based on the history, the Rack/Room consumes more than usual, reaches its maximum computing capacity. In group ②, there are flags due to the high variation of the signals, for example, a high derivative of the power consumption or temperature. In total, 281 flags for 242 metrics are defined.

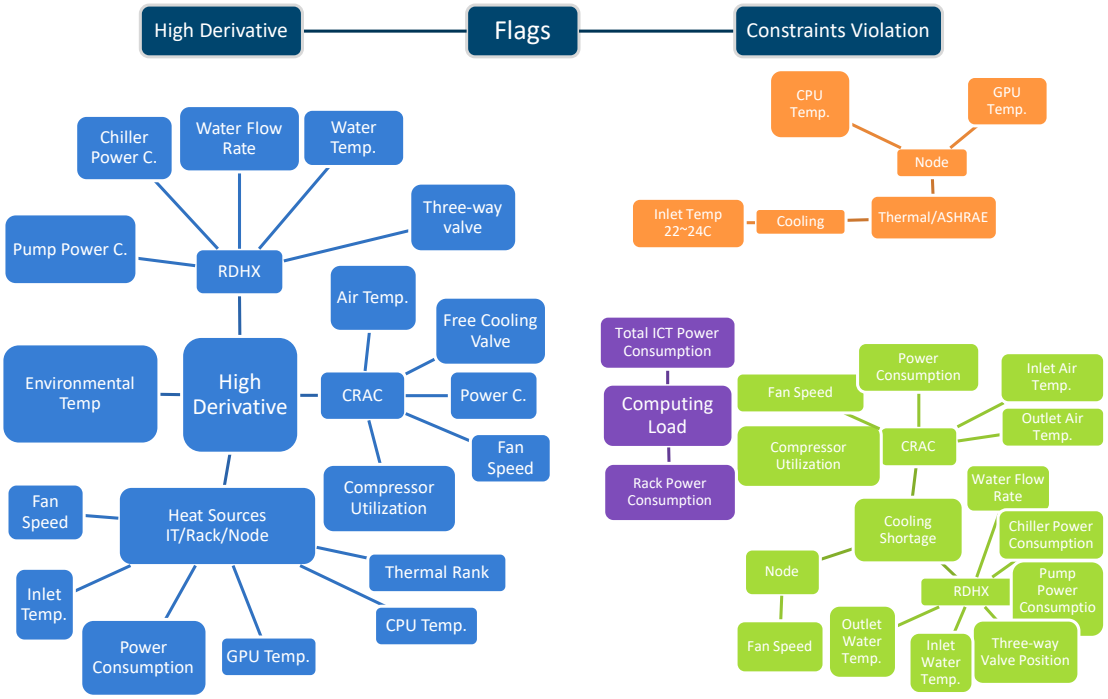


Figure 5.3: Different parts of flags set.

5.4.1 Mathematical Definition of the Flags

In this section, the mathematical formula of the flags is presented. Figure 5.3 shows the different parts of the set of flags. Each rack of Marconi-100 has 20 chassis, and each chassis host one node. From chassis 1 to 20 from bottom to top. So for Marconi-100, we can use the chassis temperature and node temperature, interchangeably since each chassis host one node. In the following, the ***thresholds*** are different for each formula, and we set them based on the recommendations like ASHRAE or data analysis. For metrics with no recommendation for inequality like $\mathcal{M} > threshold$, we used a quantile of 0.99 of parameter, and for inequality like $\mathcal{M} < threshold$ quantile of 0.01.

In the following equations, $\mathcal{M}(t)$ is a metric, and t shows time. $\mathcal{M}(t)$ can be power consumption of node, chiller, CRAC unit, pumps or temperature of GPU, CPU, PCIe, Inlet, water of RDHX, Air of CRAC, or the fan speed of node, CRAC units or compressor utilization of CRAC units, the position of the valve of RDHX, CRAC units, water flow rate, etc. Each rack has 20 nodes/chassis; each node experiences a different inlet, CPU core, and GPU temperatures. ${}_{rack}\mathcal{M}_{max}(t)$ and ${}_{rack}\mathcal{M}_{min}(t)$ show the maximum and minimum value measured for a metric by these 20 nodes of the rack at time t. Flag 3.1 and 3.1 check if a metric experience

is higher and lower than a threshold. This is a constraint violation check flag.

$$_{rack}\mathcal{M}_{max}(t) > _{major}threshold \tag{5.1}$$

$$_{rack}\mathcal{M}_{min}(t) < _{minor}threshold \tag{5.2}$$

The Flag 5.3 checks the maximum heterogeneity of measured value by a metric at one timestamp for the nodes of a rack.

$$_{rack}\mathcal{M}_{max}(t) - _{rack}\mathcal{M}_{min}(t) > threshold \tag{5.3}$$

Flag 5.4 and 5.5 examine the rack's maximum and minimum value variation for a metric, respectively.

$$|_{rack}\mathcal{M}_{max}(t) - _{rack}\mathcal{M}_{max}(t-1)| > threshold \tag{5.4}$$

$$|_{rack}\mathcal{M}_{min}(t) - _{rack}\mathcal{M}_{min}(t-1)| > threshold \tag{5.5}$$

Flag 5.6 controls the number of items of a metric that violate the threshold. For example how many GPUs experience high temperature in the rack.

$$\sum_{i=1}^{20*C} (_{rack}\mathcal{M}_i(t) > threshold) \tag{5.6}$$

Flag 5.7, 5.8, and 5.9 how many items of a metric experience abnormal variation and C for CPU, GPU, inlet, and PCIe temperature is 2, 4, 1, and 1, respectively.

$$\sum_{i=1}^{20*C} (|_{rack}\mathcal{M}_i(t) - _{rack}\mathcal{M}_i(t-1)| > threshold) \tag{5.7}$$

$$\sum_{i=1}^{20*C} (_{rack}\mathcal{M}_i(t) - _{rack}\mathcal{M}_i(t-1) > _{+}threshold) \tag{5.8}$$

$$\sum_{i=1}^{20*C} (_{rack}\mathcal{M}_i(t) - _{rack}\mathcal{M}_i(t-1) < _{-}threshold) \tag{5.9}$$

Flags 5.10, 5.11, 5.12, and 5.13 check the metrics' constraint violation and abnormal variation (except the node metrics).

$$_{major}\mathcal{M}(t) > threshold \tag{5.10}$$

$$_{minor}\mathcal{M}(t) < threshold \tag{5.11}$$

$$\mathcal{M}(t) - \mathcal{M}(t-1) > {}_{+}threshold \tag{5.12}$$

$$\mathcal{M}(t) - \mathcal{M}(t-1) < {}_{-}threshold \tag{5.13}$$

Flags 5.14 check the number of metric items that experience abnormal value based on their own history. There is a difference between this flag 5.14 and flag 5.6 which checks the number of the items of metric which violate a defined threshold for all of the items, i.e., in flag 5.14, GPU-1 has a threshold based on the history of just GPU-1. However, flag 5.6 has a fixed value as a threshold for all GPUs of the rack, which can be based on the ASHRAE recommendation or history of all the GPUs in the rack.

$$\sum_{i=1}^{20*C} ({}_{rack}\mathcal{M}_i(t) > {}_{c}thresholds) \tag{5.14}$$

Flag 5.15 checks the number of nodes that are in an odd situation due to the abnormal value of a metric, while the flag 5.16 controls the number of nodes that have strange variations in a metric.

$$\sum_{i=1}^{20} (\sum_{c=1}^{C} ({}_{rack}\mathcal{M}_{i,c}(t) > {}_{c}thresholds) \geq 1) \tag{5.15}$$

$$\sum_{i=1}^{20} (\sum_{c=1}^{C} (|{}_{rack}\mathcal{M}_c(t) - {}_{rack}\mathcal{M}_c(t-1)| > {}_{c}thresholds) \geq 1) \tag{5.16}$$

Variation of Coldest Chassis at a Rack: Subscript i shows chassis/node number. $\mathcal{C}^{inlet}(t)$ shows chassis-number of coldest chassis at time t, based on the inlet temperature.

$$|{}_{rack}\mathcal{C}^{inlet}(t) - {}_{rack}\mathcal{C}^{inlet}(t-1)| > threshold \tag{5.17}$$

Thermal Rank of Chassis ${}_{node}\mathcal{R}_i^{inlet}(t)$***:*** Index of the chassis/node in a sorted list of chassis/node based on its inlet temperature at time t. For example, in Marconi 100 $chassis-7$ of rack-5 at 2021-02-05 15:50:00 is the coldest chassis, so its thermal rank is one at that time ${}_{node}\mathcal{R}_7^{inlet}(2021/02/01-15:10:00) = 1$.

$$\sum_{i=1}^{20} |{}_{node}\mathcal{R}_i^{inlet}(t) - {}_{node}\mathcal{R}_i^{inlet}(t-1)| > threshold \tag{5.18}$$

In general, this flag can detect a situation that there is switching in the thermal rank of most of the chassis of the one rack, which mostly appears when chassis temperatures of a rack quickly change from compact/dense to widespread pattern or vice versa.

5.4.2 Initial Labeling of Samples Utilizing the Abnormality Level (Sum of Flags)

Dataset is created by nodes' metrics and room facilities metrics, but it does not contain any normal or abnormal label to distinguish between the normal or abnormal samples. There are some reports related to the anomaly/failure that the experts of CINECA provided. But these reports are very rare and just for situations where the bad side effects of the anomaly are evident, and it caused a reduction of computing capacity or even an outage of the cluster. Some conditions or abnormalities restrict the effective utilization of resources in HPC systems. Although these anomalies degrade the performance of HPC clusters, those are not effortlessly noticeable for human experts. In general, these anomalies can affect energy-to-solution, time-to-solution, again of the nodes, etc. In this study, to find the suspicious patterns accurately in the monitoring signals of the HPC room (Marconi 100), the flags are introduced, and to have an initial label for each sample of the dataset the sum of the raised flags at each timestamp (sample) is utilized.

The set of flags defined in this study can detect many suspicious patterns in monitoring signals (especially related to the thermal and power characteristic of the HPC cluster). The main weakness of the flags and generally most of the rule-based methods is that these methods could not consider the complicated correlation of the signals in finding the anomalies or suspicious patterns of the monitoring signals. Each of the individual signals may represent a normal pattern for a period, but the correlation of the signals creates an anomaly or vice versa, an individual signal represents a suspicious pattern, but it is in a normal condition regarding the whole monitoring signal. If an ML-based approach for anomaly detection design and train correctly can solve this weakness of the flags.

Utilizing flags makes it potential to identify some abnormal patterns; therefore, if a sample has zero raised flags, it is more probable that this sample is a normal sample. So for this part of the study, I classify the samples with zero raised flags as normal or non-anomaly samples based on the flags. It is difficult or impracticable to find a solid threshold for classifying samples as abnormal based on the raised flags at each timestamp. Therefore, a threshold of 25 for the sum of raised flags is defined to classify a sample as an abnormal sample by statistical analysis of samples, i.e., this definition of the thresholds for normal and abnormal samples is very conservative and classifies almost 14%of samples as normal and 4% as abnormal and majority of the samples classified as grey samples (Tabel 5.1).

Figure 5.4 shows the sum of flags with the blue line on the left y-axis and the moving average (with a window of one week) of the sum of flags with the orange line on the right y-axis, two thresholds, divided the samples of the dataset into three regions. There is a peak in the moving average of the sum of flags after 2021-07-22, which is related to the real physical failure at 2021-07-28. These labels

are not final, but they can assist in finding a subset of the dataset that is normal or close to normal samples.

These labels are not final labels; they are just for a select part of the dataset to train the LSTM-AE. The flags could not understand the relation of different parameters (how different parameters connected to each other), but LSTM-AE expectedly should know about it.

Initial Label	Definition	Percentage of Dataset
Normal	$\sum Flags = 0$	13.93%
Grey	$1 \leq \sum Flags \leq 25$	81.83%
Abnormal	$\sum Flags > 25$	4.24%

Table 5.1: Definition of Initial Labels Based on the Flags and Percentage of Dataset.

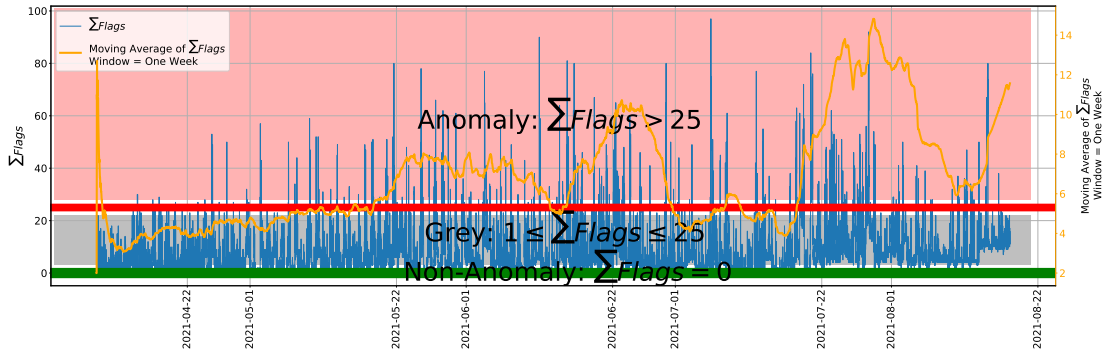


Figure 5.4: Sum of the Flags and Initial Labeling.

5.5 Autoencoder

Autoencoder is a sort of Artificial Neural Network (ANN) model (Figure 5.5) composed of three components, *Encoder*, *Code*, and *Decoder*, which reconstruct the input at the output of the model, and each of these parts can compose of multiple hidden layers. $Encoder(Input)$ maps the input to the *code* layer, and *Decoder* maps the code layer to the output of the autoencoder. Equation 5.20 shows the *error* of the input from the reconstructed input ($\widehat{Input}$) at output of autoencoder.

In the training step, by minimizing the error, the autoencoder train to reconstruct the input at the output, and in the test step, the reconstruction error shows the performance of the autoencoder in reconstruction. The training of the autoencoder can be in a supervised, semi-supervised or unsupervised manner. The reconstruction error in anomaly detection can use to classify the samples as normal and abnormal by comparing with predefined threshold if the well-trained

autoencoder reconstructs the sample with an error lower than the threshold; this sample is normal; otherwise, it is abnormal [105].

$$\widehat{Input} = Decoder(Encoder(Input)) \tag{5.19}$$

$$error = Error(Input, \widehat{Input}) \tag{5.20}$$

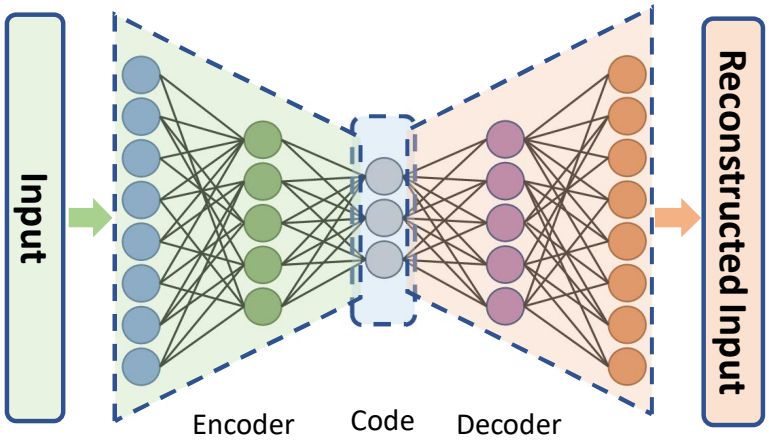


Figure 5.5: Autoencoder.

The sum of flags is utilized for selecting a normal subset of the original dataset to ensure that the autoencoder trained with almost normal samples and learned the normal property of the data. In this study, two types of the autoencoder are investigated (i) MLP-AE, which is composed of the Multilayer Perceptron (MLP), and (ii) LSTM-AE, which is composed of the Long short-term memory (LSTM) layers. In this study, the input of the autoencoder is the Marconi 100 thermal, power, and cooling parameters, and the same parameters should be reconstructed at the output. This model should learn useful properties of data and how different parameters are related to each other, and the temporal aspect of data.

(i) MLP-AE: This model is composed of the Multilayer Perceptron (MLP) and can learn the normal relation of different input parameters, which is essential. However, this model cannot learn the temporal relation of the data. Flags work based on the derivative of the parameters; it means flags somehow can see some temporal characteristic of the dataset. **(ii) LSTM-AE:** Long short-term memory (LSTM) autoencoder, is composed of LSTM layers, a type of Recurrent Neural Network (RNN) that learns long-term dependencies thanks to additional gates [69]; therefore, this model can learn the temporal characteristic of the time-series data.

5.5.1 Autoencoder Model and Training Dataset Configuration Selection

From the dataset, the first two months are selected to do some preliminary experiments. In the first row of figure 5.6 graphically illustrated, that selected dataset is divided into three subsets based on the sum of flags: samples with zero raised flags as normal or non-anomaly ($\sum Flags = 0$), samples with more than 25 raised flags as the anomaly ($\sum Flags > 25$), and samples between normal and abnormal as a grey dataset ($1 \leq \sum Flags \leq 25$). In these experiments, MLP-AE and LSTM-AE are evaluated; meanwhile, the effect of mixing some parts of the grey dataset into the training dataset on the performance of models is investigated. Each row of figure 5.6 shows the different configurations for the training dataset. The Autoencoder (AE) models are trained with 75% of the normal dataset or 75% of the normal dataset + parts of the grey dataset. Adding some parts of the grey dataset can improve the performance of the autoencoder because it increases the size of the training dataset and it can help to generalization of the model, and the labels generated by the sum of the flags are not the final labels, and a lot of the normal samples are inside the grey dataset. The remaining 25% of the normal dataset and the whole abnormal dataset are used for the test dataset. The test dataset is fixed in all the experiments. In training, samples dont have any labels, and it is somehow unsupervised learning.

Figure 5.6: Different Configurations of the Train and Test Dataset.

The reconstruction error of the trained autoencoders on the test dataset is computed. If the trained model can reconstruct the input sample with low error, it means the autoencoder identified that sample as normal, but if it reconstructs with high error, it means it detects some anomaly at that sample. And as mentioned, the test is done on the trained model with 25% of the normal dataset, and usually but not always, the autoencoder should reconstruct this part with low reconstruction error (the low error will be defined). Moreover, the test dataset contains whole abnormal (based on the sum of flags) samples, and it is expected that the autoencoder reconstructs these samples with high reconstruction error (the high error will be defined). Figure 5.7 reports the reconstruction error of MLP-AE and LSTM-AE of the test dataset for 6 different configurations of the training dataset. The outliers are invisible to be a more readable plot. The red and blue boxes show the reconstruction error of the abnormal and normal parts of the test dataset, respectively. The goal is to discriminate between the normal and abnormal samples of the test dataset based on the reconstruction error of the autoencoder. Consequently, to distinguish between the normal and anomaly samples, these two boxes should have low overlap i.e., if the boxplot of the reconstruction error of the normal and abnormal datasets have high overlap, then the distribution of the error of normal and abnormal samples will have high overlap, and classification of the samples based on the reconstruction error will be impracticable. As shown in figure 5.7, the normal and abnormal boxplots in MLP-AE have a higher overlap than LSTM-AE, so the LSTM-AE in classifying samples will be more effective. This effectiveness is related to the fact that LSTM can learn about the time-series characteristic of the dataset. So empirically is shown that the LSTM-AE outperforms the MLP-AE in anomaly detection, so the LSTM-AE is selected to continue the research.

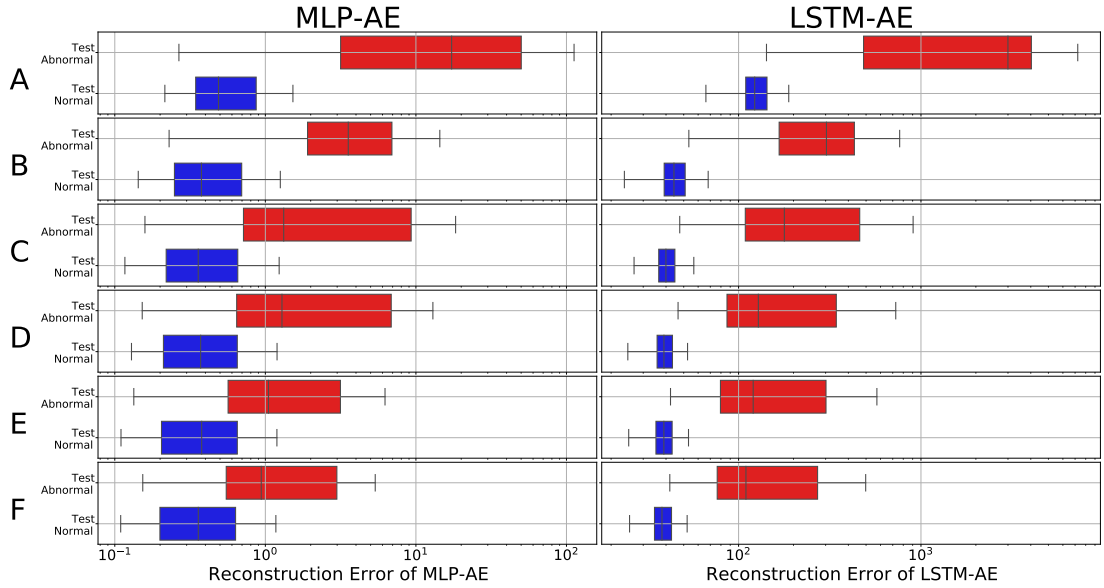


Figure 5.7: Reconstruction error of MLP-AE and LSTM-AE for the test dataset for six different configurations of the training dataset.

Reconstruction Error Threshold: To binary classification (Normal, Abnormal) of the samples utilizing the reconstruction error of the autoencoder, a reconstruction error threshold is required. So a reconstruction error threshold should be defined to convert the error by comparing it with this threshold; if the error is bigger than the threshold, the autoencoder identifies the sample as an anomaly; else, it is identified as non-anomaly. In this step of the study, quantile 0.99 of the error of the training dataset as a threshold is defined.

Assume the labels created utilizing the sum of the flags are ground truth labels by comparison of these labels with classification results based on the reconstruction error of the autoencoder (as detected labels); the results in table 5.2 are achieved. As mentioned before from boxplots in figure 5.7 evident that the LSTM-AE is more effective than MLP-AE, and also in all of the F1-score results of experiments reported in table 5.2 except configuration-A, LSTM-AE outperforms the MLP-AE.

	MLP-AE				LSTM-AE			
	Accuracy Test Normal	Accuracy Abnormal	Average Accuracy	Average F1 score	Accuracy Test Normal	Accuracy Abnormal	Average Accuracy	Average F1 score
A	0.97	0.8	0.88	0.88	0.61	1	0.83	0.86
B	0.99	0.74	0.85	0.85	1	0.95	0.97	0.97
C	1	0.36	0.64	0.53	1	0.87	0.93	0.93
D	1	0.32	0.61	0.49	1	0.77	0.88	0.87
E	1	0.29	0.59	0.44	1	0.68	0.83	0.81
F	1	0.29	0.6	0.45	1	0.6	0.78	0.75

Table 5.2: MLP-AE and LSTM-AE performance results with six different configurations of the training dataset.

Analyzing the results of the F1-score table 5.2 and figure 5.7, the configuration B, C, or D for the training dataset can be a good candidate. Figure 5.8 shows boxplots of reconstruction error of the LSTM-AE for the training and test datasets for configuration of B, C, D. The red dashed line shows the quantile of 0.99 of the training dataset; it somehow can classify the normal and abnormal dataset. Our approach for configuration B identifies 28% of the grey dataset as anomalies, and for C, D respectively, 9% and 4%. Therefore, I selected configuration C for continuing the study, which identified a lower percentage of anomalies than B; although configuration B has the highest F1-score, C has an acceptable F1-score.

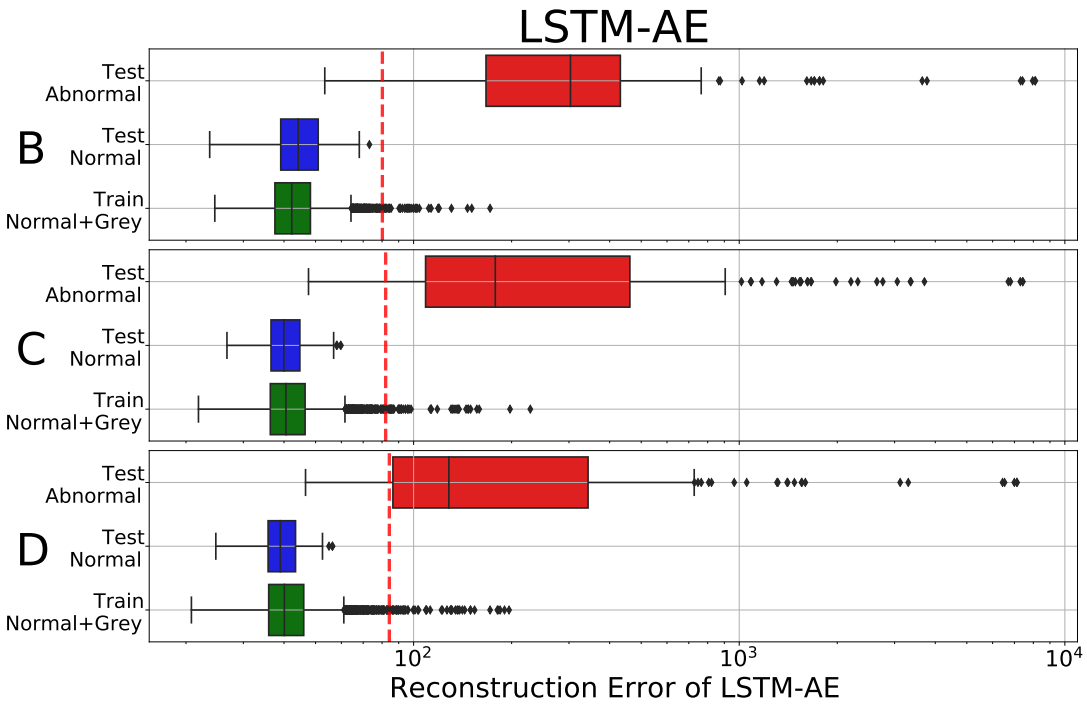


Figure 5.8: Reconstruction error of LSTM-AE for configuration of B, C, D of training dataset.

In this experiment, the training and test datasets are randomly selected. However, in a real case scenario with time-series data, the data have chronological order, and randomly selecting the test and training dataset destroys this order. This was just a preliminary experiment with the limit data part of the dataset to select the training dataset configuration and, in future steps of the study, the configuration C for training dataset without destroying the order of time-series data will be utilized.

5.6 Experimental Results

In this section, the experiment results are reported for different experiments reported. Based on the results of section 5.5.1, configuration c for the training dataset is selected, which means that adding the samples with less than 10 raised flags is a suitable configuration for the training dataset. Nine different periods from the dataset are selected for training the LSTM-AE; while the test is done for a week just after training, training periods are reported in table 5.3.

Experiment	Start Train	Stop Train
1	2021-06-15	2021-07-15
2	2021-04-08	2021-07-01
3	2021-04-08	2021-07-15
4	2021-04-08	2021-05-22
5	2021-04-08	2021-07-22
6	2021-06-22	2021-07-22
7	2021-04-08	2021-07-27
8	2021-04-08	2021-08-02
9	2021-04-08	2021-08-18

Table 5.3: Experiments Training Periods.

5.6.1 Reconstruction Error Threshold

As mentioned in 5.3, the sample classification is done by comparing the reconstruction error of each sample by *Reconstruction Error Threshold*. Defining the reconstruction error threshold is of utmost importance. This threshold regulates the anomalies ratio; a low threshold creates a high rate of anomalies and consequently a high number of False Positives, which can lead to a conservative operation of the HPC cluster, while a high threshold can generate False Negatives, which can have drastic harmful consequences. Here four different configurations to define the *Reconstruction Error Threshold* are introduced. After training the autoencoder, the reconstruction error of all samples is extracted by running the whole dataset to the trained autoencoder (inference). Then as schematics of four configurations depicted in figure 5.9, different parts of the training samples are used to define the *Reconstruction Error Threshold*.

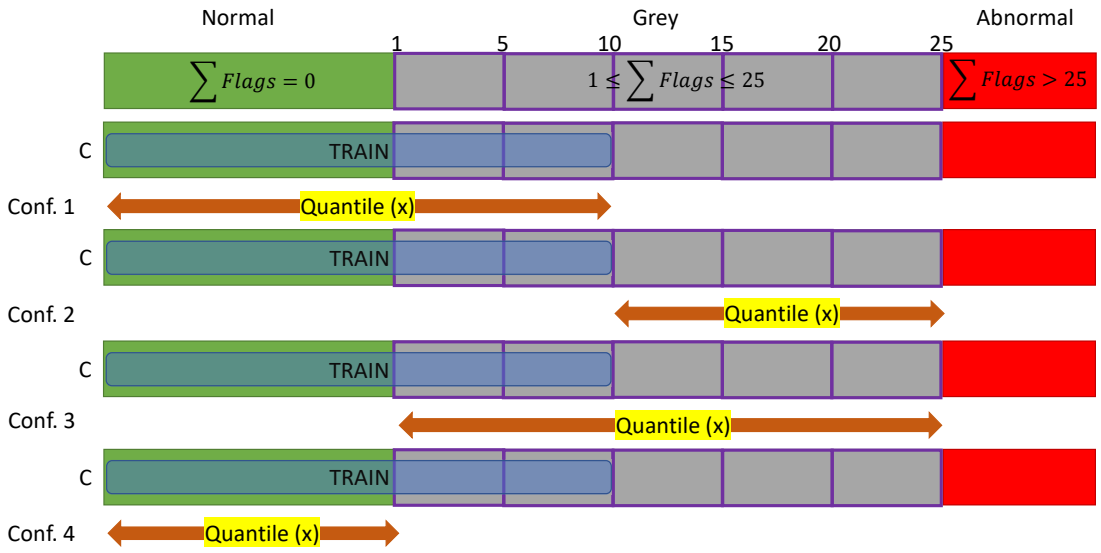


Figure 5.9: Schematics of four different configurations for computing the *Reconstruction Error Threshold*.

Figure 5.10 reports the average percentage of the anomaly for test weeks of nine experiments detected by the LSTM-AE. The x-axis shows the quantile, which starts from the median and reaches the maximum (quantile 1), and the y-axis shows the average of anomalies for test weeks identified by the autoencoder. Each line shows one configuration for defining the *Reconstruction Error Threshold*. Even with the maximum error threshold, which can be achieved by setting the quantile to 1, there is a boundary of 4% for a minimum of average anomalies for the test weeks since the test week of three of experiments contains the real thermal failure, so physically, there are anomalies in the HPC room. As evident in figure x, Conf. 2: $10 \leq \sum Flags \leq 25$ with the lowest percentage of anomalies for different quantiles provided better control on the percentage of anomalies. So it is a suitable candidate to define the error threshold. However, its capability of detecting the real failure on 28-07-2021 should be checked before finalizing this approach to compute the error threshold. In the following, it will confirm that the trained LSTM-AE with configuration C for the training dataset and computing Reconstruction Error Threshold with approach Conf. 2: $10 \leq \sum Flags \leq 25$ can detect real physical failure (very severe) accurately as well as anomalies with low severity.

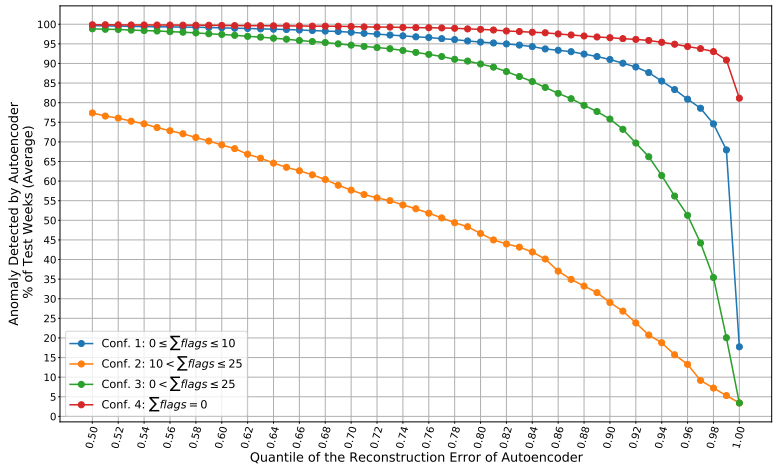


Figure 5.10: The average percentage of the anomaly for test weeks, utilizing different configurations as error threshold (5.9).

Some of the primary results of the trained LSTM-AE with configuration C (for the training dataset) and computing *Reconstruction Error Threshold* with approach Conf. 2: $10 \leq \sum Flags \leq 25$ are illustrated in figure 5.11. The x-axis is the date, and the dashed red line shows around the real thermal failure at 28-07-2021. The first, second, and third-row show the sum of flags, reconstruction error of the LSTM-AE for different experiments, and label generated by the sum of flags (given that zero flags mean normal and more than 25 means abnormal and between 1 and 25 grey zone), respectively, and all remaining rows show the waveform of the label generated by computing *Reconstruction Error Threshold* by approach Conf. 2: $10 \leq \sum Flags \leq 25$ for different experiments. Two dashed black lines show the training period of each experiment, and the dashed green line shows the end of the test week, which starts just after training and lasts for one week. This figure shows all the ranges of the dataset, and it is hard to read, so the zoom-in version around the real failure is shown in figure 5.12.

As it is evident in figure 5.12, (i) *sum of flags has a maximum value at real failure*, (ii) *all nine experiments have their peak of reconstruction error at the real failure*, and (iii) *all of the experiments can detect the real failure*. Experiments 5, 6, and 7 are more important than others because the distance between the training and real failure is short, and the test period includes the real failure. These three experiments can identify the real failure with an acceptable percentage of anomalies.

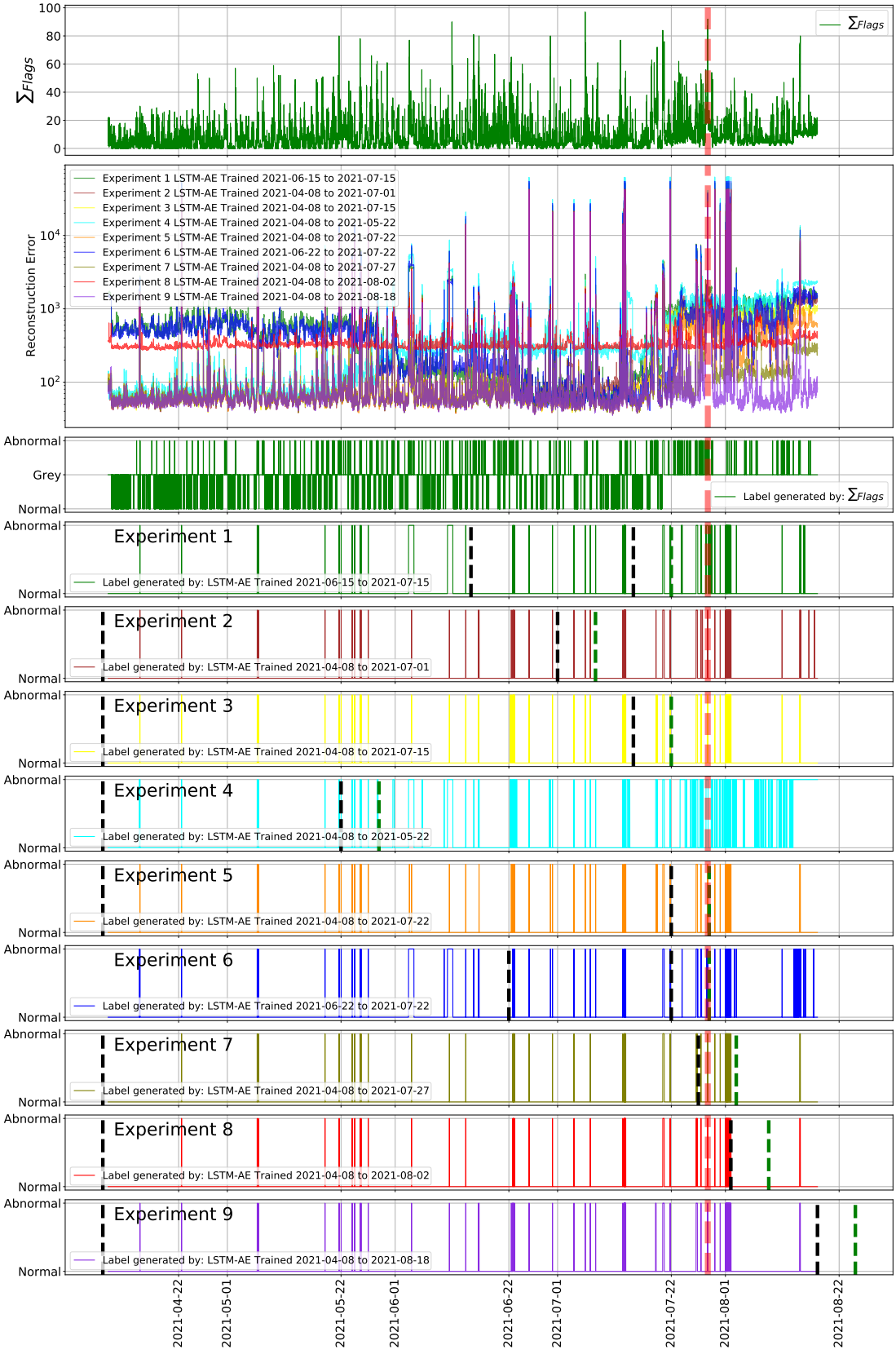


Figure 5.11: Results of the 9 different experiments with computing error threshold with approach Conf. 2: $10 \leq \sum Flags \leq 25$.

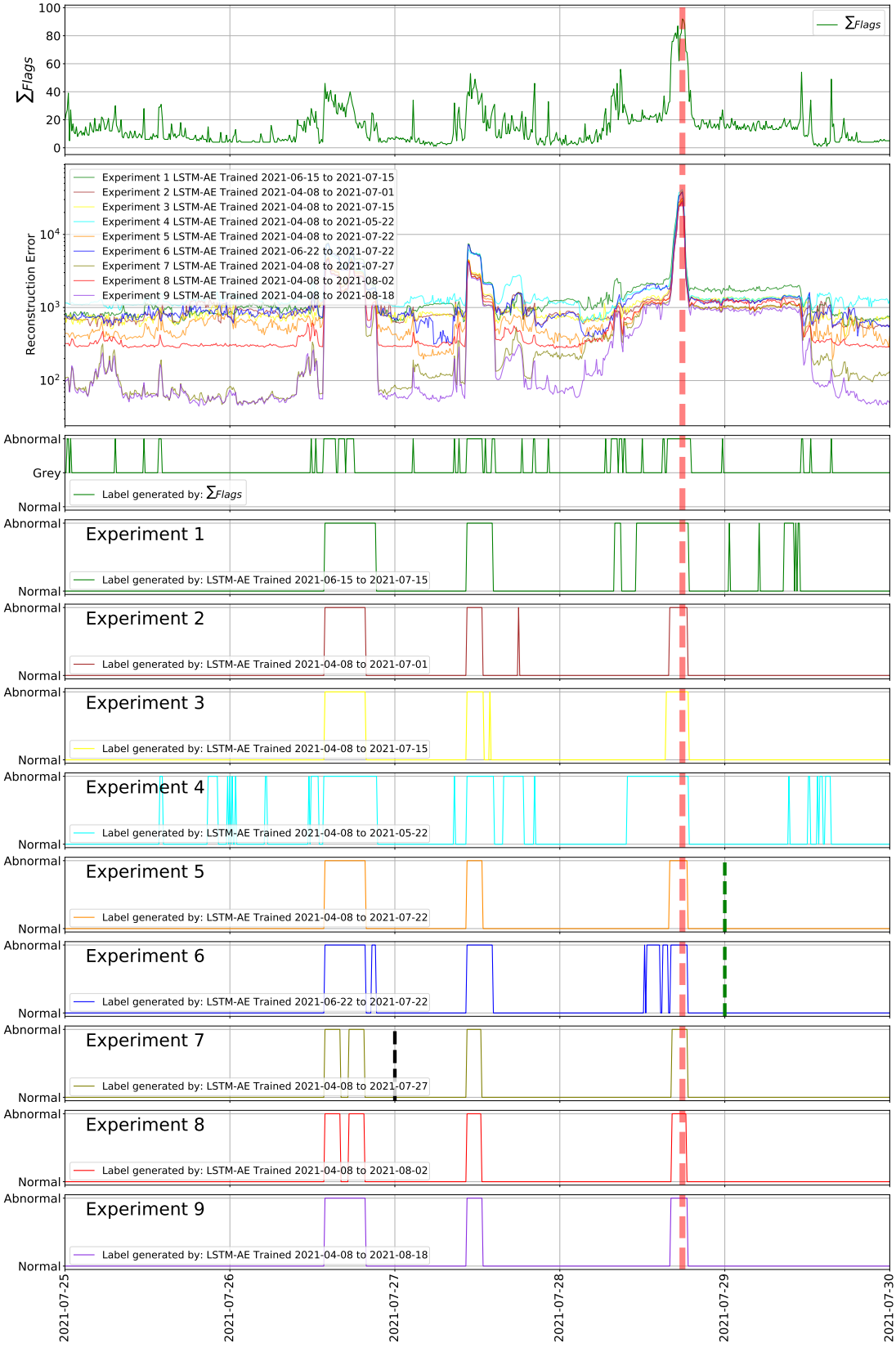


Figure 5.12: Results of the 9 different experiments with computing error threshold with approach Conf. 2: $10 \leq \sum Flags \leq 25$ (Zoom in).

5.6.2 Detailed Study of Real Physical Failure

In this section, by a detailed study of the monitoring signals at three important points around the real failure 28-07-2021, the performance of the autoencoder for detecting the anomaly is evaluated. The generated labels of experiments 5, 6, and 7 are very similar; as experiment 7 has the lowest distance between the training period and real failure, this experiment is selected to study the results further. Figure 5.13 shows the sum of flags, reconstruction error at first and second-row respectively. The third row shows the label generated by the sum of the flags with the red line and autoencoder with the blue line. Three zones (A, B, and C) around the real failure are interesting for study. Point A identified by both of the tools, the sum of the flags and autoencoder as an anomaly. And point B, identified as non-anomaly by the autoencoder, but it has more than 25 flags, which means that the sum of flags sees this point as an abnormal point. Finally, point C is a real failure and identified correctly by both sum of flags and autoencoder as an anomaly.

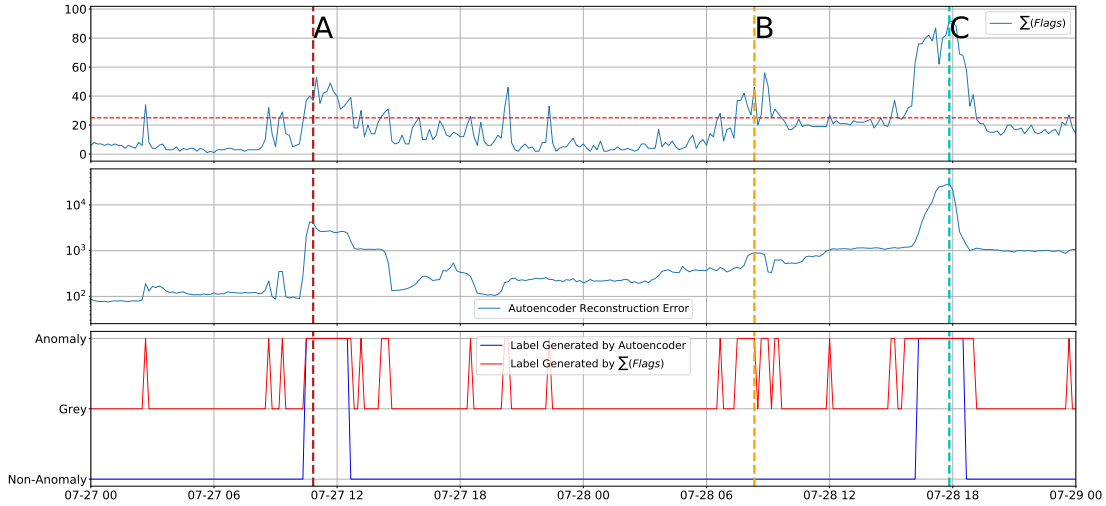


Figure 5.13: Labels of three interesting points nearby real failure.

To understand the reasons behind the identifying of these points as anomalies by the autoencoder. A more detailed study for these three points is done by generating the line plots of all sensors' signals, which are summarized in the two figures 5.14 and 5.15, and heatmap 5.16 summarized the location of the issues for both of sum of flags and autoencoder. In these two figures, the colored lines show the value of each sensor, and the black dashed line shows the average value of parameters in each row.

Figure 5.14 shows the signals of different metrics/parameters/sensors of the computing nodes of rack 205. It illustrates the CPU, GPU, PCIe, Inlet temperature

and fans speed, and finally power consumption of the nodes of one rack at room F (Rack 205) respectively in row 1 to row 6. While figure 5.14 shows the node level metrics of the HPC room, figure 5.15 shows room level metrics, especially the cooling system characteristic of the HPC room. Figure 5.15 from the first row to last row respectively shows: (i) total power consumption of the ICT devices, (ii) CRAC units: total power consumption of the CRAC units, fans speed of the CRAC units, compressors utilization of the CRAC units, Free cooling valve open position of CRAC units, outlet, inlet temperature of the CRAC units, (iii) RDHX: total power consumption of the chillers, total pumps power consumption, inlet, outlet temperature of the water, the position of three-ways valve, delta temperature of outlet and inlet water temperature, (iv) ambient temperature (temperature of outside).

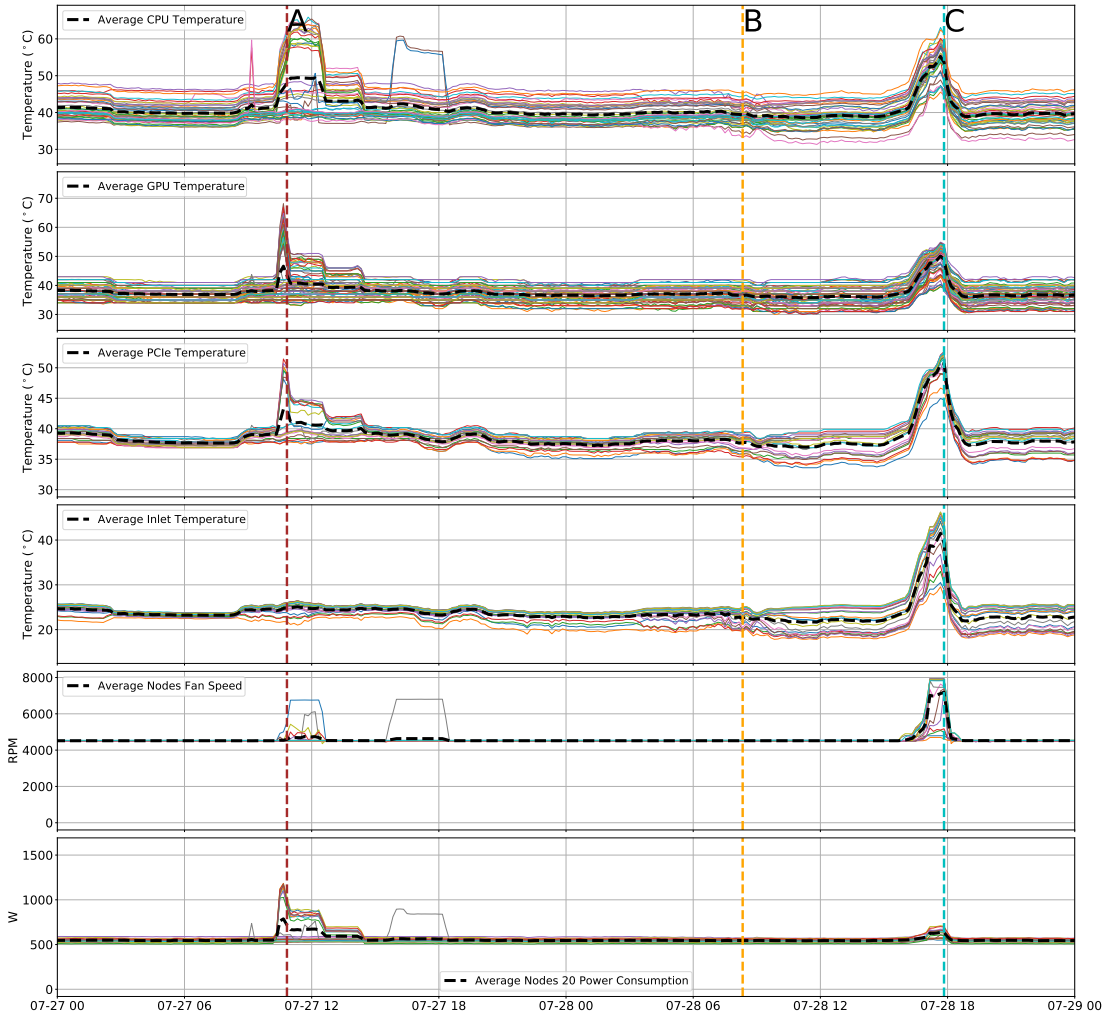


Figure 5.14: Nodes parameters of rack 205.

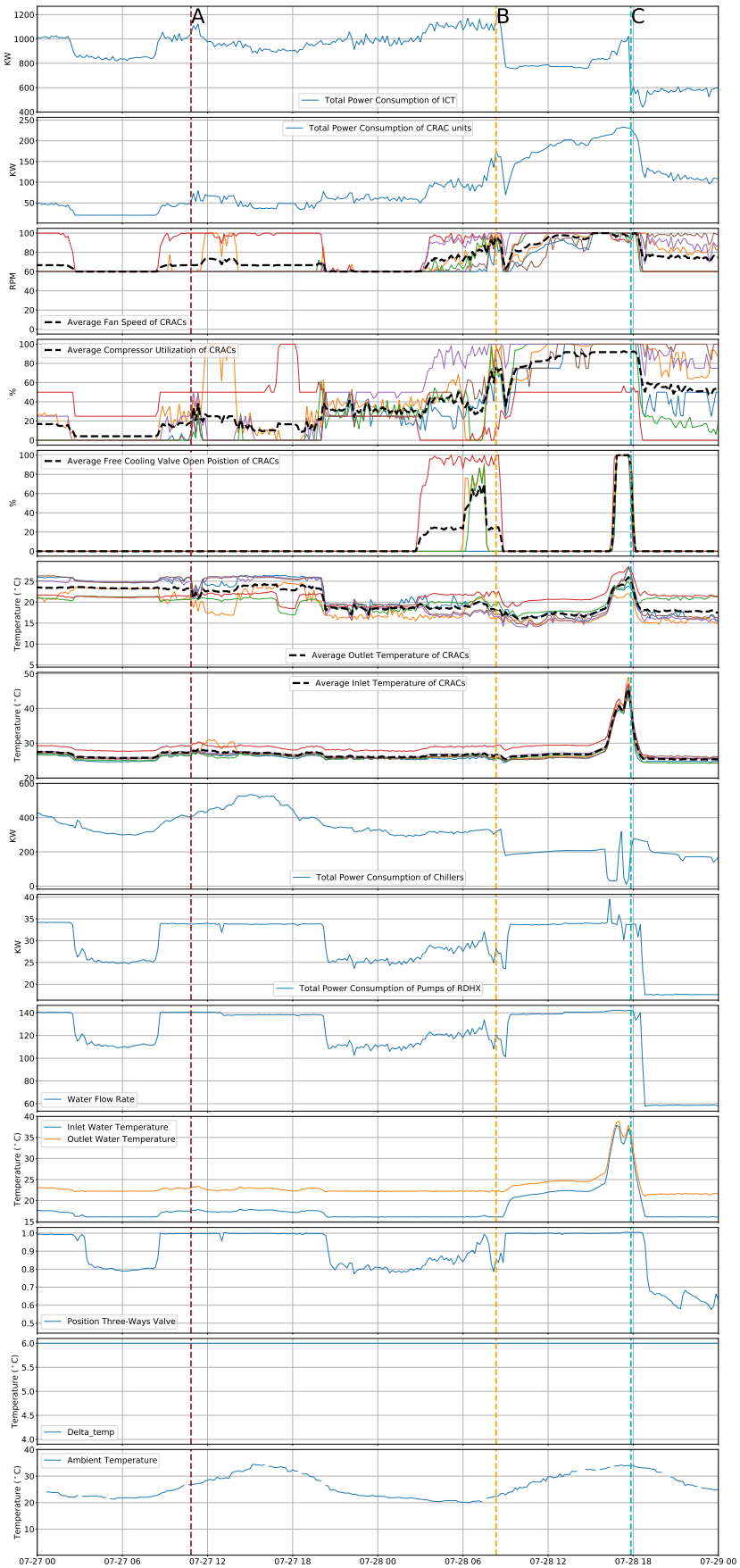


Figure 5.15: Room level parameters, Cooling systems parameters.

Considering point A, which is identified by both the sum of flags and autoencoder as an anomaly: While nodes experience the normal inlet temperature, the inside temperature (CPU, GPU, and PCIe) is high. So the room temperature is normal, and the cooling system operates correctly. Before point A the power consumptions of the nodes start to increase due to the computing demands (GPU and CPU), which turns into the high temperature inside the nodes, and then quickly, the computing loads are reduced. Meanwhile, the fans of nodes increase the speed, and it seems that after point A the high power consumption of nodes related to the fans rather than the computing and there is a peak of total power consumption of ICT just after point A I think it is due to increase in fan speeds of the nodes. Although there is some fluctuation in the compression utilization, and it reduces the outlet temperature of the CRAC units, this is not enough to change the inlet temperature of the nodes. The anomaly label of point A was related more to some nodes' computing load, and the cooling system's reaction was not fast enough to support this quick increase in the computing demand or power consumption, which turned into the nodes as a high temperature of nodes.

So while nodes' inlet temperatures are normal, computing loads are high and reaction of the cooling systems are not fast enough to support computing load which turns into high temperature at nodes level, and autoencoder correctly detects this as an anomaly due to the high temperature of nodes.

Considering point B, which is identified by the autoencoder as non-anomaly but the sum of flags detects it as abnormal: The node-level parameters of this point, like temperature, fan speed, and power consumption of nodes, are completely normal. In room-level parameters before this point, the free cooling activated (first two CRAC units out of four units then three out of four) and this is the primary source of signal fluctuations of the other parts of the two cooling systems. Activating the free cooling has caused (i) an increase in the power consumption of the RDHX, which means the water cooling system works more, and also (ii) an increase in power consumption (fans speed and compressors utilization) of CRAC units. This situation is controlled by deactivating the free cooling as well as a reduction in computing load of the room, and as it is explicit, it is successful, and there is no rise in the node level temperature.

So node level parameters are normal, and activation of free cooling is the primary source of signals' fluctuations of cooling systems, and flags identify these signals' fluctuations as a suspicious pattern while autoencoder correctly detects this point as normal because all systems are under control.

Considering point C, which is a real thermal failure and identified by both of sum of flags and autoencoder as an anomaly: All the node level parameters like temperature and fans speed of the nodes are high, and nodes experience high inlet temperature, so the cooling systems are in trouble. After

reduction of point B (some parameters like total power consumption of the ICT and CRAC units), continuously the power consumption of the CRAC units is increased, and it reached its peak at C. Before C, the free cooling activated for four out of four CRAC units meanwhile by activating of free cooling the power consumptions of the chillers of the RDHX reduced, and in the same time, the computing load increased these three action 1- increasing the computing load 2- activation of free cooling and 3- reduction in chillers cooling capacity, create thermal emergency which cause an increase in the temperature of the room and temperature of the inlet and outlet water of the RDHX and inlet and outlet temperature of the CRAC units which turn into thermal emergency in the cores of nodes and it creates out of control situation in node level and room level. The autoencoder and flags correctly identified these problems and labeled the dataset as an anomaly.

So three actions create a thermal emergency; 1- increasing the computing load, 2- activation of free cooling, and 3- reduction in RDHX cooling capacity. Which increase: 1- room temperature, 2-inlet and outlet water temperature of RDHX, and 3- inlet and outlet temperature of CRAC units, which leads to out-of-control conditions in node level and room level.

5.6.3 Locations of Anomalies

Heatmap in figure 5.16 shows the severity and zone of issues/anomalies that each of the autoencoder and sum flags identified in three points around real failure. It is composed of two main columns; the first column from the left shows the results of the autoencoder and the sum of flags independently, while the second column shows the aggregated results of both tools. Also, this figure has two rows, and the top row in the y-axis shows different parts of the HPC room: room level facilities metrics and node-level metrics, so the first row shows the zone of detected issues/anomalies, but the second row shows total severity of anomalies in three points. Annotation of the top columns is a normalized number, while the annotation of the bottom columns is the sum of the metrics identified as anomalies.

As reported in the bottom left subplot of figure 5.16 in point A, which is identified by both tools as an anomaly, the autoencoder recognizes 73 out of the 241 sub anomalies in different zones of the HPC room; meanwhile, there are 37 raised flags out of 281 possible flags in this point. For point B, although there are 46 raised flags which is more than point A, the autoencoder detected 25 metrics with high reconstruction error, and finally, it identified this point as non-anomaly. And for point C, which is the real/physical thermal failure, the autoencoder identified 204 out of 241 metrics in trouble, and the sum of flags experienced maximum raised flags in this point 92 out of 281. As reported in figure 5.16, at point A, with 110 anomalies identified by both tools, the temperature of node level for a few hours is high, and autoencoder identified some issues in node level temperatures like CPU,

GPU, and PCIe and also the power consumption of nodes, and in the room level facilities, it discovers some issues mostly on water cooling system (RDHX). In point B, the autoencoder recognizes the node level metrics as almost normal, but it sees some issues in the cooling system and total power consumption (as is also evident in the first two rows of figure 5.15). For point C, autoencoder is detected in almost all parts of the system.

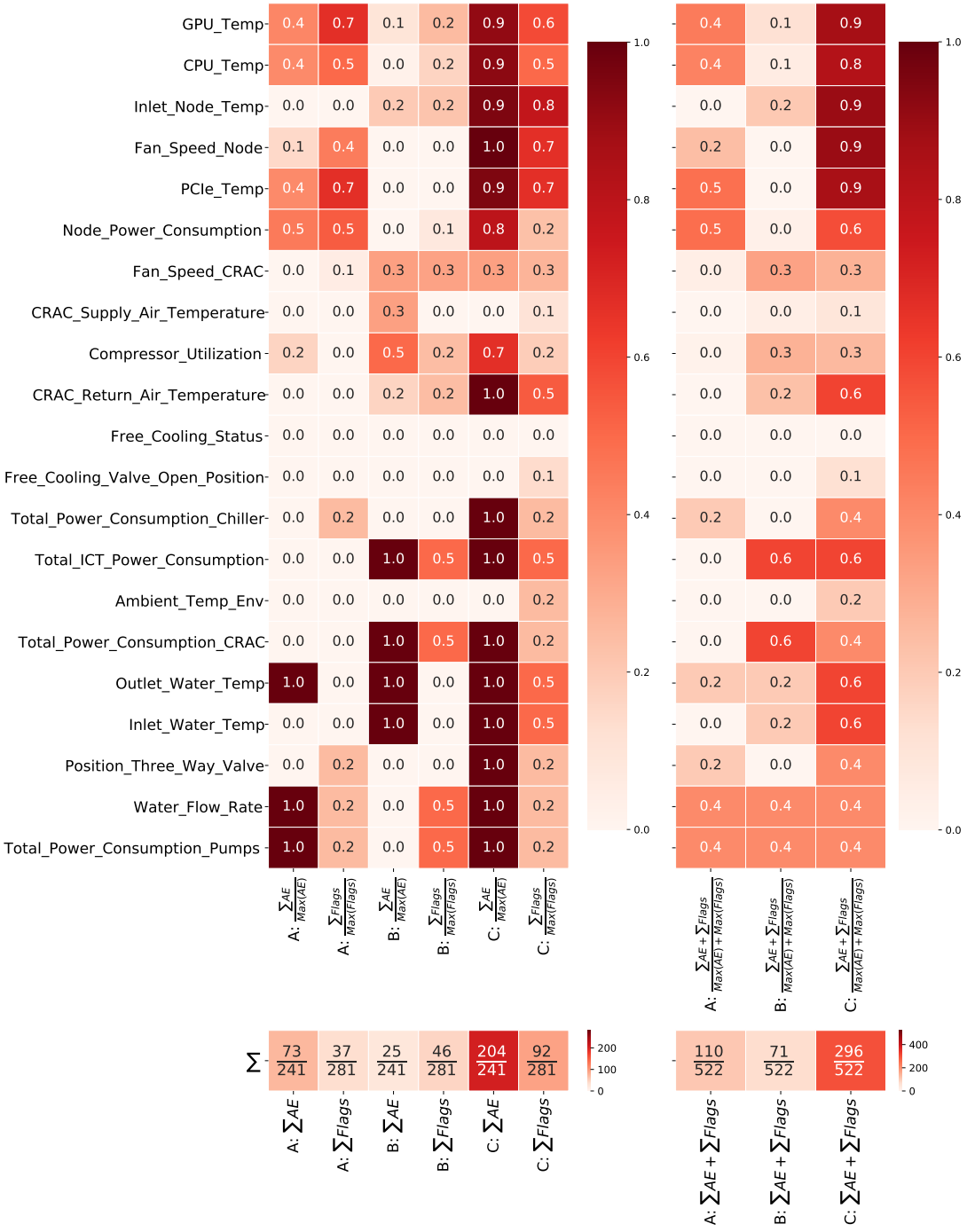


Figure 5.16: Severity and zone of the anomaly in the HPC room.

5.7 Summary

This study employs monitoring signals of the in-production HPC cluster and HPC room facilities for anomaly detection in the HPC room. Two sets of methods for anomaly detection are proposed. (i) A set of rule-based statistical methods (flags) that explore different metrics at the HPC room, system, sub-system, and node level to find abnormal patterns. (ii) Semi-supervised ML-based approaches for anomaly detection; using merely flags in anomaly detection of the HPC room is inadequate due to the flags' weakness in analyzing the complicated correlation of the signals in finding the anomalies or suspicious patterns. Indeed, I propose a methodology that uses the flags to select suitable subsets of the dataset to train the semi-supervised ML-based approaches. I tested two different semi-supervised approaches: an long short-term memory autoencoder (LSTM-AE) and an Multilayer perceptron autoencoder (MLP-AE). Empirically with a set of experiments, I demonstrated that the LSTM-AE outperforms the MLP-AE in anomaly detection. Different approaches for defining the thresholds (essential in rule-based and ML-based methods to transform floating numbers to binary classes) are investigated accurately. Finally, all the steps and approaches are validated by a detailed study of the real thermal failure and illustrated that LSTM-AE could identify the anomalies if it trains with an appropriate part of the monitoring dataset collected with a telemetry system in the ExaMon database.

Chapter 6

Conclusion

After an introduction (chapter 1) in chapter 2, I provided preliminary definitions and a brief description of the HPC system and HPC room facilities. Some technical characteristics of Marconi A1, Marconi A2, Marconi A3, Galileo, and Marconi 100, which are located in CINECA HPC rooms N and F, were illustrated. I explained the cooling systems of CINECA HPC rooms: CRAC units (+Direct Free Cooling) and water cooling systems (RDHX). CINECA is equipped with ExaMon (Exascale Monitoring), a state-of-the-art datacenter monitoring system. This chapter had a brief overview of ExaMon.

In chapter 3 I studied the thermal and power consumption characteristics of two HPC rooms in the CINECA datacenter.

Considering the HPC Room N, which hosts three HPC clusters: the data collected by a WSN monitoring system in the HPC facility, which tracks the room temperature, are analyzed. The correlation between the different measured temperatures is analyzed, and I find that there are four thermal zones in the room: (a) Subfloor, which is a cold area. (b) The left and (c) right parts of the room that are separated by the RDHX. In the (d) vertical direction, I found that there is not a strong correlation between the top and bottom in the center of the HPC room, and the center of the room has high thermal variation. With data analysis, I prove that it is possible to reduce data collection and transmission rates of two orders of magnitude. Therefore, sensors consume two times less power. With this reduction in the data collection, it needs a hundred times less data storage capacity, and, consequently, for data processing, it needs lower computing resources. This study can be used as a guideline for sensor placement. Liquid cooling and cage divide the room into the different thermal zones; meanwhile, CRAC units, RDHX, and generated heat by servers create a complex thermal system. Using the internal temperature sensors and onboard sensors like IPMI combined with our WSN telemetry system, we can upgrade our system and enhance the study's preciseness. I would highlight the difficulties of doing a more precise analysis with

the current WSN and then suggesting to combine it with IPMI.

Considering the HPC Room F, which hosts the Marconi A2 HPC cluster (Marconi A2 closed in January 2020 and was replaced with Marconi 100): the room's spatial and thermal heat dissipation characteristics are analyzed. The study revealed that nodes hosted in the top chassis of racks have worse thermal conditions than bottom nodes. This directly impacts the average power consumption of the nodes, which is higher for the top nodes. These nodes can consume up to 6% more power due to a higher fan speed than bottom nodes. The study of the thermal map revealed that the center row of racks in the Marconi A2 room F is colder than the other two rows; overall, this was valid for normal and thermal hazard conditions. The hotspot varies vertically during the thermal emergency condition. I can conclude that the study of the spatial and thermal heat dissipation characteristics revealed significant non-idealities and heterogeneity, which, if modeled, can be leveraged by thermal-aware job-scheduler and room-level power management runtimes.

In chapter 4 I suggested a framework for thermal hazard prediction, which encompasses data query and preprocessing, model training, and final model inference, which provides the prediction. The thermal hazard predictor is a model that, based on time series data of computing nodes' sensors, predicts if a thermal hazard will happen in the room in the next hours. Input data are the time series of nodes' temperature, and the output is a binary classification: likely forthcoming hazard or not. The dataset does not contain any labels, so I used statistical analysis of real thermal hazard data of the Marconi A2 KNL (largest HPC cluster of CINECA at 2019) to characterize thermal hazards in the HPC room. Then based on this analysis, I defined rule-based statistical method to create labels. Different classical machine learning and DL tools were investigated and empirically shown that the proposed thermal hazard predictor, namely a Temporal Convolutional Network (TCN), outperformed non-deep models and LSTM. Some techniques are introduced/examined to deal with issues like; samples-overlapping of time series data or imbalanced datasets.

I showed that thermal hazard prediction has many challenges in real case scenario implementation. Although the TCN model works well in the research phase F1-score of 0.98 (selecting the test dataset randomly like what is common in most research and papers), it will have substantial performance degradation in real implementation (i.e., F1-score reduce from 0.98 to $\sim$0.74). This study investigates enough complex TCN models with different convolutional layers, and input data flow to improve the model's performance. The memory-based approaches for labeling the thermal hazard were investigated. During this study, we had meetings with one of the most powerful HPC cluster's sys-admin (CINECA) to understand the situation better and find a solution to implement this thermal hazard prediction framework

in a real in-production large-scale HPC cluster. Based on the study results, I find that due to the dataset's complexity, the monitoring signal's dynamism, manual update of the cooling setpoints, activation of the free cooling system, etc. it is essential to use a more sophisticated anomaly detection method (or thermal hazard detection method), i.e., a rule-based statistical method with just node level data is insufficient for thermal hazard perdition for real in-production HPC rooms, and I should add the metrics of HPC room level facilities like RDHX, CRAC unit, etc. to the dataset and improve the anomaly detection approach. The study results motivated us to collect essential metrics of the Marconi100 HPC cluster and Room F from April of 2021 and utilize this big dataset to develop a complex anomaly detection tool in the next chapter of the thesis.

In chapter 5 for anomaly detection in the HPC room I employed monitoring signals of the in-production HPC cluster and HPC room facilities. I proposed two sets of methods for anomaly detection. (i) A set of rule-based statistical methods (flags) that explore different metrics at the HPC room, system, subsystem, and node level to find abnormal patterns. (ii) Semi-supervised ML-based approaches for anomaly detection; using merely flags in anomaly detection of the HPC room is inadequate due to the flags' weakness in analyzing the complicated correlation of the signals in finding the anomalies or suspicious patterns. Indeed, I propose a methodology that uses the flags to select suitable subsets of the dataset to train the semi-supervised ML-based approaches. I tested two different semi-supervised approaches: an long short-term memory autoencoder (LSTM-AE) and an Multilayer perceptron autoencoder (MLP-AE). Empirically with a set of experiments, I demonstrated that the LSTM-AE outperforms the MLP-AE in anomaly detection. Different approaches for defining the thresholds (essential in rule-based and ML-based methods to transform floating numbers to binary classes) are investigated accurately. Finally, all the steps and approaches are validated by a detailed study of the real thermal failure and illustrated that LSTM-AE could identify the thermal anomalies if it trains with an appropriate part of the monitoring dataset, collected with a telemetry system in the ExaMon database.

Acronyms

AC Air Conditioning. 19

ACU Air Conditioning Unit. 22

AE Autoencoder. 109

AI Artificial Intelligence. I, 1

ANN Artificial Neural Network. 57, 107

API Application Programming Interface. 11, 13, 15, 41

BD Big Data. 57

CAPEX Capital Expenditures. 2

CC Correlation Coefficient. 25, 35, 39, 40, 47

CFD Computational Fluid Dynamics. 57

CPU Central Processing Unit. 18, 99, 101–103, 118, 122, 123

CRAC Computer Room Air Conditioning. V, 7–9, 11, 12, 17, 19–22, 24–26, 29, 30, 32–34, 37, 39, 40, 46, 54, 60, 76, 97, 98, 101–103, 119, 122, 123, 127, 129

CRAH Computer Room Air Handler. 19, 22

DC Direct Current. 23

DDR4 Double Data Rate 4. 8, 40

DFC Direct Free Cooling. 7–9, 12, 17, 25, 26, 32, 33

DL Deep Learning. I, 4, 56, 57, 64, 65, 67, 96, 128

DTM Dynamic Thermal Management. 22

DX Direct Expansion. 7, 12, 26

FN False Negative. 72, 94–96

FP False Positive. 72, 94–96

GPU Graphics Processing Unit. 9, 18, 101–103, 105, 118, 122, 124

HPC High Performance Computing. I, V, VII, 1–4, 6–8, 11, 12, 17–21, 26, 28, 32, 40, 53, 54, 56–58, 60, 65, 68, 74, 75, 79–81, 84, 85, 88, 94, 96–101, 106, 113, 114, 119, 123, 125–129

ICT Information and Communication Technologies. 101, 119, 122, 123

IoT Internet of Things. 16, 19

IPMI Intelligent Platform Management Interface. 9, 11, 13, 40, 41, 54, 57, 65, 127, 128

IT Information Technology. I, 2

KNL Knights Landing. II, V, VI, 8, 10, 40, 41, 47, 48, 53, 58, 65, 96, 128

LoRa Long Range. 19, 23

LoRaWAN Long Range Wide-Area Network. 16, 24

LSTM Long Short-Term Memory. 56, 66, 67, 96, 99, 108, 110, 128

LSTM-AE Long Short-Term Memory Autoencoder. VII, IX, 99, 107–115, 126, 129

LTWN Last Time Window Number. 25

LVP Last-Value Predictor. 62, 64, 66, 67

MCC Matthews Correlation Coefficient. 74, 75, 79, 80, 82, 83, 85, 88, 95, 96

MCDRAM Multi-Channel Dynamic Random-Access Memory. 8, 40

meanCC Mean Correlation Coefficient. 25, 35, 36

ML Machine Learning. 56, 68, 99, 100, 106, 126, 129

MLP Multilayer Perceptron. 108

MLP-AE Multilayer Perceptron Autoencoder. VII, IX, 99, 108–112, 126, 129

NT Node-Threshold. 90, 94

OPEX Operating Expenses. 2

PCIe Peripheral Component Interconnect Express. 101, 103, 118, 122, 124

PH Prediction Horizon. 61, 62, 64

PMU Performance Monitoring Unit. 13

PUE Power Usage Effectiveness. 22, 55

RAM Random Access Memory. 9

RBF Radial Basis Function. 66, 67

RDHX Rear Door Heat Exchanger. V, 7–9, 11, 12, 17, 19, 20, 26, 34, 40, 41, 48, 53, 54, 60, 97, 98, 101–103, 119, 122–124, 127, 129

RNN Recurrent Neural Network. 66, 108

ROI Return on Investment. 18

RPM Rotations per Minute. V, 44, 45

SGD Stochastic Gradient Descent. 56, 66, 67

SST Statistical Significance Test. 25

SSTMask Statistical Significance Test Mask. 25

std Standard Deviations. 37

STIT Spatial-Temporal-Impact-Threshold. 90, 94

SVM Support Vector Machine. 56, 66, 67

TCN Temporal Convolutional Network. III, VI–VIII, 56, 58, 65–67, 69, 71–77, 79–83, 85–88, 90, 96, 97, 128

TDCTW Total Data Collection Time Window. 25, 35, 37

TN True Negative. 72, 94–96

TP True Positive. 72, 83, 94–96

TT Training Time. 81

TW Time Window. 25, 35–38, 60, 65, 80

TWG Time Window Group. 25, 35

TWN Time Window Number. 25, 35

WSN Wireless Sensor Network. 19, 23, 24, 26, 28, 29, 34, 53, 54, 127, 128

Bibliography

[1] The 58th edition of the top500 list., NOVEMBER, 2021. https://www.top500.org/.

[2] The 53th edition of the top500 list., JUNE 2019. https://www.top500.org/.

[3] The 55th edition of the top500 list., JUNE 2020. https://www.top500.org/.

[4] The 53th edition of the top500 list., JUNE 2019. https://www.top500.org/.

[5] Ufficio Tecnico. Technical documents, Sep 2019. https://www.cineca.it/.

[6] Andrea Borghesi, Andrea Bartolini, Michela Milano, and Luca Benini. Pricing schemes for energy-efficient hpc systems: Design and exploration. *The International Journal of High Performance Computing Applications*, 33(4):716–734, 2019.

[7] Andrea Bartolini, Francesco Beneventi, Andrea Borghesi, Daniele Cesarini, Antonio Libri, Luca Benini, and Carlo Cavazzoni. Paving the way toward energy-aware and automated datacentre. In *Proceedings of the 48th International Conference on Parallel Processing: Workshops*, ICPP 2019, pages 8:1–8:8, New York, NY, USA, 2019. ACM.

[8] ACM. Getting started with hpc, 2021.

[9] Christian Conficoni, Andrea Bartolini, Andrea Tilli, Carlo Cavazzoni, and Luca Benini. Integrated energy-aware management of supercomputer hybrid cooling systems. *IEEE Transactions on Industrial Informatics*, 12(4):1299–1311, 2016.

[10] Organization. https://www.cineca.it/en/about-us/organization.

[11] Alessio Netti, Michael Ott, Carla Guillen, Daniele Tafani, and Martin Schulz. Operational data analytics in practice: Experiences from design to deployment in production hpc environments. *arXiv preprint arXiv:2106.14423*, 2021.

[12] E. Rossi. Marconi-a2 (knl), 2017.

[13] *Intel Server Board S2600IP and Workstation Board W2600CR Technical Product Specification*. October 2013.

[14] 2021 Created by Elda Rossi, last modified by Francesco Cola on Nov 25. UG3.1: MARCONI UserGuide. `https://wiki.u-gov.it/confluence/display/SCAIUS/UG3.1%3A+MARCONI+UserGuide`, 2018. Accessed: 2021-12-27.

[15] 2021 Created by D. Guida, last modified by Neva Besker on Mar 10. UG3.3: GALILEO UserGuide. `https://wiki.u-gov.it/confluence/display/SCAIUS/UG3.3%3A+GALILEO+UserGuide`, 2018. Accessed: 2021-12-27.

[16] Andrea Bartolini, Andrea Borghesi, Antonio Libri, Francesco Beneventi, Daniele Gregori, Simone Tinti, Cosimo Gianfreda, and Piero Altoè. The d.a.v.i.d.e. big-data-powered fine-grain power and performance monitoring support. In *Proceedings of the 15th ACM International Conference on Computing Frontiers*, CF '18, page 303–308, New York, NY, USA, 2018. Association for Computing Machinery.

[17] Andrea Bartolini, Francesco Beneventi, Andrea Borghesi, Daniele Cesarini, Antonio Libri, Luca Benini, and Carlo Cavazzoni. Paving the way toward energy-aware and automated datacentre. In *Proceedings of the 48th International Conference on Parallel Processing: Workshops*, ICPP 2019, New York, NY, USA, 2019. Association for Computing Machinery.

[18] MultiTech. Programmable gateway for the internet of things, sept 2019. https://www.multitech.com/brands/multiconnect-conduit.

[19] Andrea Bartolini, Christian Conficoni, Roberto Diversi, Andrea Tilli, and Luca Benini. Multiscale thermal management of computing systems-the multitherman approach. *IFAC PapersOnLine*, 50(1):6709–6716, 2017.

[20] ETP4HPC. Strategic research agenda, 2017.

[21] Hadi Esmaeilzadeh, Emily Blem, Renee St Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark silicon and the end of multicore scaling. In *38th Annual International Symposium on Computer Architecture (ISCA 2011)*, pages 365–376, June 2011.

[22] Jim Rogers. Ornl's warm water hpc facilities and control systems, 2019.

[23] Jim Gao and Ratnesh Jamidar. Machine learning applications for data center optimization. 2014.

[24] Madhurina Pore, Zahra Abbasi, Sandeep K. S. Gupta, and Georgios Varsamopoulos. *Techniques to Achieve Energy Proportionality in Data Centers: A Survey*, pages 109–162. Springer New York, New York, NY, 2015.

[25] Chayan Nadjahi, Hasna Louahlia, and Stéphane Lemasson. A review of thermal management and innovative cooling strategies for data center. *Sustainable Computing: Informatics and Systems*, 19:14–28, 2018.

[26] Li Li, Wenli Zheng, Xiaodong Wang, and Xiaorui Wang. Data center power minimization with placement optimization of liquid-cooled servers and free air cooling. *Sustainable Computing: Informatics and Systems*, 11:3–15, 2016.

[27] Mohsen Seyedkazemi Ardebili, Carlo Cavazzoni, Luca Benini, and Andrea Bartolini. Thermal characterization of a tier0 datacenter room in normal and thermal emergency conditions. In *International Conference on High Performance Computing in Science and Engineering*, pages 1–16. Springer, 2019.

[28] M. Ot, T. Wilde, and H. Ruber. Roi and tco analysis of the first production level installation of adsorption chillers in a data center. In *16th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm 2017)*, pages 981–986, May 2017.

[29] Hayk Shoukourian, Torsten Wilde, Herbert Huber, and Arndt Bode. Analysis of the efficiency characteristics of the first High-Temperature Direct Liquid Cooled Petascale supercomputer and its cooling infrastructure. *Journal of Parallel and Distributed Computing*, 107:87–100, September 2017.

[30] Alexander Moskovsky, Egor Druzhinin, Alexey Shmelev, Vladimir Mironov, and Andrey Semin. Server level liquid cooling: Do higher system temperatures improve energy efficiency? *Supercomput. Front. Innov.: Int. J.*, 3(1):67–74, January 2016.

[31] C. Conficoni, A. Bartolini, A. Tilli, C. Cavazzoni, and L. Benini. Integrated energy-aware management of supercomputer hybrid cooling systems. *IEEE Transactions on Industrial Informatics*, 12(4):1299–1311, Aug 2016.

[32] Christian Conficoni, Andrea Bartolini, Andrea Tilli, Carlo Cavazzoni, and Luca Benini. Hpc cooling: A flexible modeling tool for effective design and management. *IEEE Transactions on Sustainable Computing*, 2018.

[33] Jungsoo Kim, Martino Ruggiero, and David Atienza Alonso. Free cooling-aware dynamic power management for green datacenters. In *Proceedings of the*

ACM/IEEE 2012 International Conference on High Performance Computing and Simulation (HPCS), volume 1, pages 140–146. IEEE Press, 2012.

[34] Domenico Balsamo, Danilo Porcarelli, Luca Benini, and Brunelli Davide. A new non-invasive voltage measurement method for wireless analysis of electrical parameters and power quality. In *SENSORS, 2013 IEEE*, pages 1–4. IEEE, 2013.

[35] Christos Stergiou, Kostas E Psannis, Brij B Gupta, and Yutaka Ishibashi. Security, privacy & efficiency of sustainable cloud computing for big data & iot. *Sustainable Computing: Informatics and Systems*, 19:174–184, 2018.

[36] Danilo Porcarelli, Davide Brunelli, and Luca Benini. Clamp-and-forget: A self-sustainable non-invasive wireless sensor node for smart metering applications. *Microelectronics Journal*, 45(12):1671–1678, 2014.

[37] LoRa Alliance. Lora™ modulation basics, 2015.

[38] Francesco Beneventi, Andrea Bartolini, Carlo Cavazzoni, and Luca Benini. Cooling-aware node-level task allocation for next-generation green hpc systems. In *High Performance Computing & Simulation (HPCS), 2016 International Conference on*, pages 690–696. IEEE, 2016.

[39] Ayse K Coskun, José L Ayala, David Atienza, and Tajana Simunic Rosing. Modeling and dynamic management of 3d multicore systems with liquid cooling. In *2009 17th IFIP International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 35–40. IEEE, 2009.

[40] R. Diversi, A. Tilli, A. Bartolini, F. Beneventi, and L. Benini. Bias-compensated least squares identification of distributed thermal models for many-core systems-on-chip. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(9):2663–2676, Sept 2014.

[41] Jungsoo Kim, Mohamed M Sabry, Martino Ruggiero, and David Atienza. Power-thermal modeling and control of energy-efficient servers and datacenters. In *Handbook on data centers*, pages 857–913. Springer, 2015.

[42] Francesco Fraternali, Andrea Bartolini, Carlo Cavazzoni, Giampietro Tecchiolli, and Luca Benini. Quantifying the impact of variability on the energy efficiency for a next-generation ultra-green supercomputer. In *Proceedings of the 2014 international symposium on Low power electronics and design*, pages 295–298. ACM, 2014.

[43] Aniruddha Marathe, Yijia Zhang, Grayson Blanks, Nirmal Kumbhare, Ghaleb Abdulla, and Barry Rountree. An Empirical Survey of Performance and Energy Efficiency Variation on Intel Processors. In *Proceedings of the 5th International Workshop on Energy Efficient Supercomputing*, E2SC'17, pages 9:1–9:8, New York, NY, USA, 2017. ACM. event-place: Denver, CO, USA.

[44] Eduard Oro, Victor Depoorter, Albert Garcia, and Jaume Salom. Energy efficiency and renewable energy integration in data centres. strategies and modelling review. *Renewable and Sustainable Energy Reviews*, 42:429–445, 2015.

[45] Rajarshi Das, Jeffrey O Kephart, Jonathan Lenchner, and Hendrik Hamann. Utility-function-driven energy-efficient cooling in data centers. In *Proceedings of the 7th international conference on Autonomic computing*, pages 61–70. ACM, 2010.

[46] Donghwa Shin, Sung Woo Chung, Eui-Young Chung, and Naehyuck Chang. Energy-optimal dynamic thermal management: Computation and cooling power co-optimization. *IEEE Transactions on Industrial Informatics*, 6(3):340–351, 2010.

[47] Luca Parolini, Bruno Sinopoli, Bruce H Krogh, and Zhikui Wang. A cyber–physical systems approach to data center modeling and control for energy efficiency. *Proceedings of the IEEE*, 100(1):254–268, 2011.

[48] Rongliang Zhou, Zhikui Wang, Cullen E Bash, Alan McReynolds, Christopher Hoover, Rocky Shih, Niru Kumari, and Ratnesh K Sharma. A holistic and optimal approach for data center cooling management. In *Proceedings of the 2011 American Control Conference*, pages 1346–1351. IEEE, 2011.

[49] N. Ahuja, C. W. Rego, S. Ahuja, Shen Zhou, and S. Shrivastava. Real time monitoring and availability of server airflow for efficient data center cooling. In *29th IEEE Semiconductor Thermal Measurement and Management Symposium*, pages 243–247, March 2013.

[50] R. K. Sharma, C. E. Bash, C. D. Patel, R. J. Friedrich, and J. S. Chase. Balance of power: dynamic thermal management for internet data centers. *IEEE Internet Computing*, 9(1):42–49, Jan 2005.

[51] Eduard Oró, Albert Garcia, and Jaume Salom. Experimental and numerical analysis of the air management in a data centre in Spain. *Energy and Buildings*, 116:553–561, 2016.

[52] Nosayba El-Sayed, Ioan A Stefanovici, George Amvrosiadis, Andy A Hwang, and Bianca Schroeder. Temperature management in data centers: why some (might) like it hot. *ACM SIGMETRICS Performance Evaluation Review*, 40(1):163–174, 2012.

[53] Y. Fulpagare, Y. Joshi, and A. Bhargav. Rack level forecasting model of data center. In *2017 16th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, pages 824–829, May 2017.

[54] Muhammad Tayyab Chaudhry, M Hasan Jamal, Zeeshan Gillani, Waqas Anwar, and Muhammad Salman Khan. Sustainable Computing: Informatics and Systems Thermal-benchmarking for cloud hosting green data centers. *Sustainable Computing: Informatics and Systems*, 25:100357, 2020.

[55] Q. Fang, J. Wang, Q. Gong, and M. Song. Thermal-aware energy management of an hpc data center via two-time-scale control. *IEEE Transactions on Industrial Informatics*, 13(5):2260–2269, Oct 2017.

[56] S. A. Bermudez, H. F. Hamann, L. J. Klein, F. J. Marianno, and A. Claassen. Optimal and distributed automatic discrete control of air conditioning units in data centers. In *2015 31st Thermal Measurement, Modeling Management Symposium (SEMI-THERM)*, pages 13–18, March 2015.

[57] Jetmir Haxhibeqiri, Abdulkadir Karaagac, Floris Van den Abeele, Wout Joseph, Ingrid Moerman, and Jeroen Hoebeke. Lora indoor coverage and performance in an industrial environment: Case study. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2017.

[58] Qiang Liu, Yujun Ma, Musaed Alhussein, Yin Zhang, and Limei Peng. Green data center with iot sensing and cloud-assisted smart temperature control system. *Computer Networks*, 101:104–112, 2016.

[59] Michael G Rodriguez, Luis E Ortiz Uriarte, Yi Jia, Kazutomo Yoshii, Robert Ross, and Peter H Beckman. Wireless sensor network for data-center environmental monitoring. In *2011 Fifth International Conference on Sensing Technology*, pages 533–537. IEEE, 2011.

[60] The 51st edition of the top500 list., June 2018. https://www.top500.org/.

[61] P. Bruce and A. Bruce. *Practical Statistics for Data Scientists*. O'Reilly, 1005 Gravenstein Highway North, Sebastopol, CA 95472., May 2017.

[62] Brian Beers. P-value definition, Apr 2019. http://www.investopedia.com.

[63] Brian Hawkins. Kairosdb, fast time series database on cassandra, jan 2017. http://kairosdb.github.io.

[64] Nicola Jones. How to stop data centres from gobbling up the world's electricity. *Nature*, 561(7722):163–167, 2018.

[65] NREL.

[66] Hayk Shoukourian and Dieter Kranzlmüller. Forecasting power-efficiency related key performance indicators for modern data centers using LSTMs. *Future Generation Computer Systems*, 112:362–382, November 2020.

[67] John L. Hennessy and David A. Patterson. *Computer Architecture, Sixth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 6th edition, 2017.

[68] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv:1803.01271*, 2018.

[69] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[70] Jinkyun Cho, Taesub Lim, and Byungseon Sean Kim. Measurements and predictions of the air distribution systems in high compute density (internet) data centers. *Energy and buildings*, 41(10):1107–1115, 2009.

[71] Jayati Athavale, Yogendra Joshi, and Minami Yoda. Artificial neural network based prediction of temperature and flow profile in data centers. In *2018 17th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, pages 871–880. IEEE, 2018.

[72] M. Marwah, R. Sharma, and C. Bash. Thermal anomaly prediction in data centers. In *2010 12th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, pages 1–7, 2010.

[73] Lizhe Wang, Gregor Von Laszewski, Jai Dayal, Xi He, Andrew J Younge, and Thomas R Furlani. Towards thermal aware workload scheduling in a data center. In *2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, pages 116–122. IEEE, 2009.

[74] Qinghui Tang, Tridib Mukherjee, Sandeep KS Gupta, and Phil Cayton. Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters. In *International Conference on Intelligent Sensing and Information Processing*. IEEE, 2006.

[75] Mohsen Seyedkazemi Ardebili, Carlo Cavazzoni, Luca Benini, and Andrea Bartolini. Thermal characterization of a tier0 datacenter room in normal and thermal emergency conditions. In *Proceedings of High Performance Computing in Science and Engineering 2019*.

[76] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[77] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[78] Ki Bum Lee, Sejune Cheon, and Chang Ouk Kim. A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes. *IEEE Transactions on Semiconductor Manufacturing*, 30(2):135–142, 2017.

[79] Giulia Moschini, Régis Houssou, Jérôme Bovay, and Stephan Robert-Nicoud. Anomaly and fraud detection in credit card transactions using the arima model. In *Engineering Proceedings*, volume 5, page 56. Multidisciplinary Digital Publishing Institute, 2021.

[80] Tara Salman, Deval Bhamare, Aiman Erbad, Raj Jain, and Mohammed Samaka. Machine learning for anomaly detection and categorization in multi-cloud environments. In *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pages 97–103. IEEE, 2017.

[81] Andrea Borghesi, Martin Molan, Michela Milano, and Andrea Bartolini. Anomaly detection and anticipation in high performance computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 2021.

[82] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3009–3017, 2019.

[83] Jerry G Thomas, Sudhir P Mudur, and Nematollaah Shiri. Detecting anomalous behaviour from textual content in financial records. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 373–377. IEEE, 2019.

[84] Ioannis Ch Paschalidis and Yin Chen. Statistical anomaly detection with sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 7(2):1–23, 2010.

[85] Jorge Ortiz, Catherine Crawford, and Franck Le. Devicemien: network device behavior modeling for identifying unknown iot devices. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, pages 106–117, 2019.

[86] Ecp: Exascale computing project.

[87] Burak Aksar, Benjamin Schwaller, Omar Aaziz, Vitus J Leung, Jim Brandt, Manuel Egele, and Ayse K Coskun. E2ewatch: An end-to-end anomaly diagnosis framework for production hpc systems. In *European Conference on Parallel Processing*, pages 70–85. Springer, 2021.

[88] Abhinav Bhatele, Jayaraman J Thiagarajan, Taylor Groves, Rushil Anirudh, Staci A Smith, Brandon Cook, and David K Lowenthal. The case of performance variability on dragonfly-based systems. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 896–905. IEEE, 2020.

[89] Abhinav Bhatele, Kathryn Mohror, Steven H Langer, and Katherine E Isaacs. There goes the neighborhood: performance degradation due to nearby jobs. In *SC'13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–12. IEEE, 2013.

[90] Matthieu Dorier, Gabriel Antoniu, Rob Ross, Dries Kimpe, and Shadi Ibrahim. Calciom: Mitigating i/o interference in hpc systems through cross-application coordination. In *2014 IEEE 28th international parallel and distributed processing symposium*, pages 155–164. IEEE, 2014.

[91] Aniruddha Marathe, Yijia Zhang, Grayson Blanks, Nirmal Kumbhare, Ghaleb Abdulla, and Barry Rountree. An empirical survey of performance and energy efficiency variation on intel processors. In *Proceedings of the 5th International Workshop on Energy Efficient Supercomputing*, pages 1–8, 2017.

[92] Anthony Agelastos, Benjamin Allan, Jim Brandt, Ann Gentile, Sophia Lefantzi, Steve Monk, Jeff Ogden, Mahesh Rajan, and Joel Stevenson. Toward rapid understanding of production hpc applications and systems. In *2015 IEEE International Conference on Cluster Computing*, pages 464–473. IEEE, 2015.

[93] James M Brandt, David DeBonis, Ann C Gentile, Jim Lujan, Cindy Martin, David J Martinez, Stephen Lecler Olivier, Kevin Pedretti, Narate Taerat, and Ron Velarde. Enabling advanced operational analysis through multi-subsystem data integration on trinity. Technical report, Sandia National Lab.(SNL-CA), Livermore, CA (United States); Sandia National ..., 2015.

[94] Mohsen Seyedkazemi Ardebili, Marcello Zanghieri, Alessio Burrello, Francesco Beneventi, Andrea Acquaviva, Luca Benini, and Andrea Bartolini. Prediction of thermal hazards in a real datacenter room using temporal convolutional networks. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1256–1259. IEEE, 2021.

[95] Rafiul Ahad, Eric Chan, and Adriano Santos. Toward autonomic cloud: Automatic anomaly detection and resolution. In *2015 International Conference on Cloud and Autonomic Computing*, pages 200–203. IEEE, 2015.

[96] Hiranya Jayathilaka, Chandra Krintz, and Rich Wolski. Performance monitoring and root cause analysis for cloud-hosted web applications. In *Proceedings of the 26th International Conference on World Wide Web*, pages 469–478, 2017.

[97] Behnaz Arzani, Selim Ciraci, Boon Thau Loo, Assaf Schuster, and Geoff Outhred. Taking the blame game out of data centers operations with netpoirot. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, page 440–453, New York, NY, USA, 2016. Association for Computing Machinery.

[98] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems. *Engineering Applications of Artificial Intelligence*, 85:634–644, 2019.

[99] Bruno L Dalmazo, João P Vilela, Paulo Simoes, and Marilia Curado. Expedite feature extraction for enhanced cloud anomaly detection. In *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, pages 1215–1220. IEEE, 2016.

[100] Ozan Tuncer, Emre Ates, Yijia Zhang, Ata Turk, Jim Brandt, Vitus J Leung, Manuel Egele, and Ayse K Coskun. Online diagnosis of performance variation in hpc systems using machine learning. *IEEE Transactions on Parallel and Distributed Systems*, 30(4):883–896, 2018.

[101] Denis Shaykhislamov and Vadim Voevodin. An approach for dynamic detection of inefficient supercomputer applications. *Procedia Computer Science*, 136:35–43, 2018.

[102] Anwesha Das, Frank Mueller, and Barry Rountree. Aarohi: Making real-time node failure prediction feasible. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1092–1101. IEEE, 2020.

[103] Alessio Netti, Zeynep Kiziltan, Ozalp Babaoglu, Alina Sîrbu, Andrea Bartolini, and Andrea Borghesi. A machine learning approach to online fault classification in hpc systems. *Future Generation Computer Systems*, 110:1009–1022, 2020.

[104] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. Anomaly detection using autoencoders in high performance computing systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9428–9433, 2019.

[105] Burak Aksar, Yijia Zhang, Emre Ates, Benjamin Schwaller, Omar Aaziz, Vitus J Leung, Jim Brandt, Manuel Egele, and Ayse K Coskun. Proctor: A semi-supervised performance anomaly diagnosis framework for production hpc systems. In *International Conference on High Performance Computing*, pages 195–214. Springer, 2021.