Vrije Universiteit Amsterdam

Bachelor Thesis

RC Modeler: Modeling Thermal Behavior of Processors in Discrete Event-Based Data Center Simulators

Author: Zohaib Zaheer (2735075)

1st supervisor: Tiziano De Matteis
daily supervisor: Dante Niewenhuis
2nd reader: Alexandru Iosup

A thesis submitted in fulfillment of the requirements for the VU Bachelor of Science degree in Computer Science

July 18, 2024

Abstract

As data centers become increasingly vital to modern society, their energy consumption continues to increase, with up to 40% of this energy used for cooling. Therefore, efficient thermal management is crucial not only for reducing operational costs but also for minimizing environmental impact. This thesis focuses on understanding and modeling the thermal profiles of processors in data centers to enhance cooling efficiency.

We evaluate current industry methods and tools for modeling processor thermal characteristics and introduce our own modeling tool based on Resistance-Capacitance (RC) circuit principles. This tool predicts processor die temperatures from power dissipation data. We validate our tool by integrating a software prototype into the OpenDC discrete event data center simulator, achieving an average error rate of 16%, which is within accepted industry standards. Our modeling tool aims to facilitate more effective cooling optimization in data centers, contributing to significant energy savings and environmental benefits.

The final implementation of the model can be found on GitHub at AtLarge/opendc

Contents

1 Introduction **1**

1.1 Problem Statement . . . 3
1.2 Research Questions . . . 4
1.3 Research Methodology . . . 5
1.4 Thesis Contributions . . . 6
1.5 Thesis Structure . . . 7
1.6 Plagiarism Declaration . . . 7

2 Simulation and the Role of Power in Thermal **9**

2.1 Simulation . . . 9
2.2 Power . . . 11
2.3 Electrical Energy to Thermal Energy . . . 13
2.4 Thermal Energy to Temperature . . . 14
2.5 Processor Temperature and Server Temperature . . . 15
2.6 Conclusion . . . 16

3 Thermal Modeling **17**

3.1 Challenges in Estimating Temperature . . . 17
3.2 Manufacturer Approach . . . 18
3.3 Machine Learning Approach . . . 19
3.4 Physics Approach . . . 20
3.5 Conclusion . . . 22

4 Design of an RC-Model **23**

4.1 RC Circuit . . . 23
4.2 Static Power Model . . . 25
4.3 Thermal Model . . . 27
4.4 Conclusion . . . 28

CONTENTS

5 Evaluation of Thermal Simulation **31**
5.1 A Primer on OpenDC . . . 31
5.2 Implementation . . . 32
5.3 Dataset . . . 33
5.4 Experimental Setup . . . 34
5.5 Results . . . 34
5.6 Conclusion . . . 39

6 Conclusion **41**
6.1 Answering Research Questions . . . 41
6.2 Limitations and Future Work . . . 42

7 Reproducibility **45**
7.1 Abstract . . . 45
7.2 Artifact check-list (meta-information) . . . 45
7.3 Description . . . 46
7.4 Installation . . . 46
7.5 Evaluation and expected results . . . 46

References **49**

1

Introduction

Cloud computing, in the last two decades, has grown to become the predominant paradigm in modern computing (1). Figure 1.1 illustrates this trend by showing the increase in the total energy use year over year of data centers in The Netherlands, as reported by the *Centraal Bureau voor de Statistiek* (CBS). Many new applications, whether on web or mobile, leverage diverse cloud technologies to deliver their services. This ubiquity necessitates highly available cloud servers, with leading cloud providers now promising up-time guarantees as high as 99% (2). However, the pursuit of uninterrupted service has forced many cloud providers to adopt a strategy of overprovisioning their physical resources in an effort to meet the needs of their clients (3). This approach, while addressing availability concerns, entails a significant drawback, namely: resource wastage. Consequently, cloud providers incur fiscal losses, which are subsequently transferred to clients, while also contributing to an unnecessarily inflated carbon footprint of data centers (1).

Numerous strategies have been devised to address data center inefficiency, including the implementation of sophisticated scheduling algorithms (5), dynamically managing server utilization by transitioning idle servers to low power modes (6), and leveraging ambient temperatures for server cooling (7). Testing all these strategies on multiple data center topologies in the real world is not always feasible due to cost and time constraints. Therefore, one of the best methods to test these strategies before real-world implementation is data center simulation. Data center simulation involves modeling different workloads on various data center topologies to predict peak data center utilization (8, 9). We further explain simulation as well as its advantages and use cases in the Background Chapter.

One of the primary causes of data center inefficiencies is the amount of energy used for cooling. During the operation of servers in data centers, a substantial amount of heat is generated, leading to the necessity of cooling solutions to ensure both optimal operation and

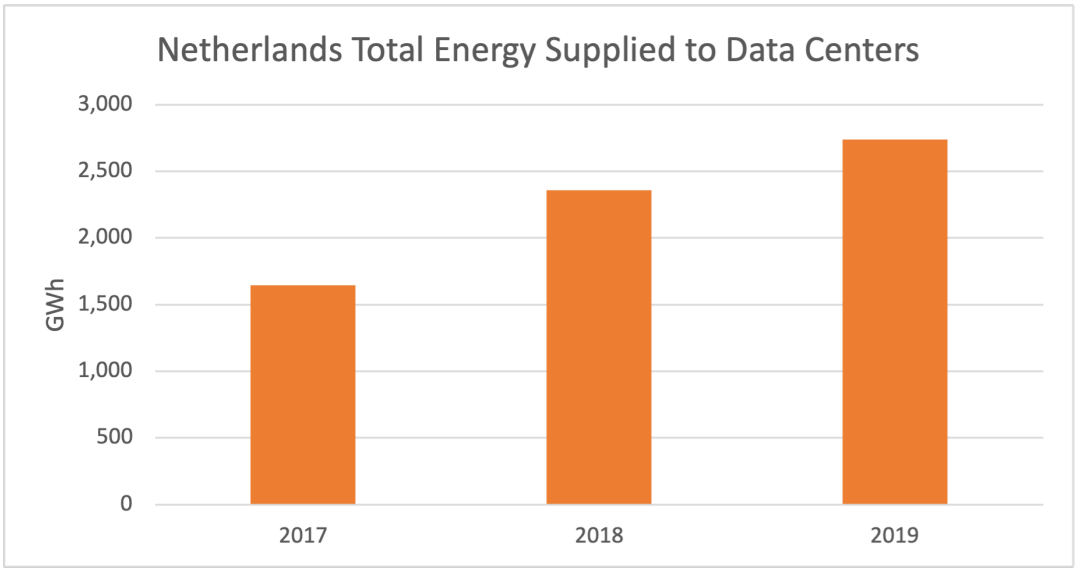

Figure 1.1: Data center energy consumption growth (4)

longevity of the processing units. The cooling infrastructure in these facilities can consume up to 40% of the total power consumption of the data center (10) and is therefore one of the biggest causes of inefficiency in the data center. As such, simulating the cooling and thermal cost associated with computational workflows in a data center can be quintessential to researchers and data center designers.

The urgency of optimizing the thermal profile of data centers is driven by two primary factors. *First*, the amount of energy used by data centers has been growing every year for the last two decades, leading to heightened environmental concerns (3). As illustrated in 1.1, the energy demand for data centers has continued to grow in the past and is forecasted to do so in the coming decade (4). This implies that the carbon footprint and negative environmental impact of these data centers are also likely to increase(11). Given that cooling costs are one of the largest contributors to data center energy costs and, consequently, the carbon footprint, it is no surprise that effective thermal modeling is crucial for lowering energy costs and the carbon footprint .

Second, with the advent of Large Language Models (LLMs) and other generative artificial intelligence (AI) models, the demand for data centers is expected to grow exponentially (11). Training and running these models at scale requires more computation than current data center providers can deliver, leading to a surge in the establishment of new data centers and a corresponding increase in energy consumption as new actors set up more facilities to train and run these models (12). Given the significant processing requirements

of these new AI technologies, the cooling costs for data centers and their subsequent carbon footprint are also poised to grow exponentially. In light of these emerging factors, the need for improved thermal models has become more critical than ever. Effective thermal models can facilitate better thermal management in data centers, paving the way for more efficient cooling strategies and lower overall energy use.

Furthermore, designing a model capable of simulating the thermal characteristics of a processor - i.e., at the node level - offers several additional benefits. For example, the industry-wide server utilization rate is only 20-40%, partly because higher utilization tends to lead to thermal throttling and, over time, hardware degradation (13). Such a model can enable processors to be pushed further by ensuring they do not overheat or throttle while also extending the lifespan of processors, resulting in a reduction in electronic waste.

Therefore, as climate change worsens and new technologies likely to exacerbate its effects emerge, the urgent need for a tool that can accurately model the thermal characteristics of processors at the node level becomes more critical. An instrument that can fulfill that role would help realize all these benefits whilst allowing researchers to build on top of it to reduce the overall energy use of data centers.

1.1 Problem Statement

We identify and aim to address third key problems that emerge when attempting to design a generalizable model for data center thermal characteristics.

First, there is a **lack of published knowledge about generalizable thermal models for data centers at the node level**. Despite the significant impact that processor thermal modeling has on the overall energy use and hardware longevity in data centers, there is limited research on modeling thermal characteristics at the node level. Most published studies focus on modeling the thermal characteristics of entire computer rooms in data centers using Computational Fluid Dynamics (CFD) (14, 15, 16) or even machine learning (17). When studies do model at the node level, they often rely on physical measurements for specific processors rather than developing a simple, generalizable model applicable to any processor. The only study we found that models a generic processor also requires initial knowledge of certain physical values (18). Furthermore, many studies addressing thermal management in data centers do so with the objective of improving cooling efficiency rather than solely modeling thermal properties (19). This makes it challenging to find literature that focuses exclusively on thermal models within the data center context.

Second, we observe the **need for an instrument facilitating thermal modeling at the node level in data centers**. Typically, CPU temperature analysis in data center simulation is conducted through mathematical or machine learning models, which often rely on previous knowledge of the physical properties of the processor (19). Meanwhile, most available tools focus on thermal modeling at the room level rather than providing measures for CPU temperatures (14). As a result, there is a scarcity of open-source tools capable of detailing the thermal characteristics of a processor under discrete event simulation conditions. This limitation makes it challenging to determine node-level temperatures, which is crucial for identifying whether specific nodes are throttling or if there are hotspots causing heating effects at certain nodes.

Third, and finally, we identify the **need for a comprehensive evaluation of the validity and accuracy of thermal models based on reference data from real data centers**. Many node-level models in the literature have not been extensively tested under real-world conditions, integrated into larger frameworks, or evaluated over extended periods (19, 20). Consequently, there is uncertainty about whether these predominantly mathematical models can effectively scale to accommodate diverse data types, maintain accuracy over time, or adapt to continuously evolving real-world conditions. Therefore, we advocate for comprehensive real-world testing within the context of a larger simulation framework. This approach would allow these models to be rigorously tested against multiple real-world scenarios and their functionality evaluated within the broader context of data center operations.

1.2 Research Questions

RQ1: How to assess the current state-of-the-art of thermal models for processors in data centers?

This research question delves into the various thermal models that exist to simulate and predict heat generation and dissipation in CPUs. By investigating the literature, the aim is to identify the existing models used in the industry and evaluate their strengths and limitations. Furthermore, this question aims to investigate the general problem statement of this thesis: the increasing importance of thermal management in ensuring the reliability, efficiency, and longevity of CPUs in data centers. As processors continue to become more powerful and compact, effective thermal modeling is essential to prevent overheating, optimize performance, and design advanced cooling solutions.

RQ2: How to design a thermal model to simulate the thermal characteristics and temperature of processors in data centers?

This research question delves into the complex field of thermal modeling. Understanding the thermal characteristics of processors in real-world scenarios involves various methods, many of which rely on physical measurements. We do not have the luxury of taking real-world measurements and, therefore, must rely on existing models and techniques for estimating temperature in circuits. However, simulating these properties in a generalized model for complex circuits, such as modern processors, presents a significant challenge due to the low-level nature of thermal behavior.

RQ3: How to evaluate and validate thermal models for processors in data centers?

Finally, understanding the precision of these models is essential for their practical application in managing CPU, and data center thermal profiles. The question aims to assess how well these models can predict actual thermal behavior under various operating conditions when tested within a larger data center simulator framework.

1.3 Research Methodology

To answer **RQ1**, we employ a short literature survey of the field to find the different thermal modeling techniques widely used by researchers. Due to the scope and time constraints of this bachelor's thesis, a comprehensive literature survey is not feasible. We also look to literature from other fields such as engineering to gain a more holistic understanding of the field. After identifying studies that provide a foundational understanding of thermal modeling for data centers and processors, we will evaluate which of these models is the best candidate for a generalizable, quantitative approach.

To answer **RQ2**, we propose a Resistance-Capacitance (RC) mathematical model from existing literature. This model is more generalizable than models that rely on taking physical temperature measurements of processors and then modeling the relationship between those measurements and other inputs to the processors, such as machine learning approaches. We then define the use cases for such a model and formulate the requirements for the design of the model based on the use cases. Our approach differs from others in the field, who only use mathematical models, because we employ discrete event simulation for data center processes. Therefore, our design has to be compatible with other data center simulator modules. The use of discrete event simulation enables our design to be used

inside a wider data center framework, allowing for the thermal model to be used in more complex and long-term analysis of the thermal characteristics of data centers as a whole. However, this also introduces additional challenges such as interoperability, human computer interaction, and overall validity. Our design builds on top of OpenDC (8), a discrete even simulator built for data centers. This approach of designing a standalone model first then integrating it with an existing simulator has two benefits: first, it allows our model to benefit from all future updates to OpenDC and, second, it allows for the model to be used by the wider community without OpenDC.

To address **RQ3**, we will implement a prototype of our thermal model into the broader framework of OpenDC, ensuring its interoperability with other modules. We will then perform an experimental analysis of our model based on the output generated by OpenDC, making iterative adjustments to our prototype as necessary based on these results. This analysis will involve simulating several traces from a reference data center and then using these traces to validate our model. Validation will be achieved by manually inspecting the temperature values and conducting an empirical comparison between the temperature and heat dissipation simulated by OpenDC using our design against the real-world data from the traces.

1.4 Thesis Contributions

We first list the **conceptual contributions** of this thesis:

1. **Analysis of the field:** We have collected and created a list of thermal modeling techniques for data centers that could be used as a primer for future researchers. Moreover, researchers working on cooling models in the future may use this work to create their own thermal models to facilitate their cooling research.

2. **Design of RC-Model:** We have developed a design for a simplified thermal model that can be generalized to various processors. The primary contribution of this design is its dual capability to function as a standalone model and to integrate seamlessly within a larger discrete event data center simulator. This flexibility distinguishes our model from others in the field and offers significant benefits to the wider community.

3. **Evaluation of RC-Model Design:** Using our implementation in OpenDC, we can evaluate this model against reference data from real-world data centers. Unlike most existing evaluations, which do not test the model against varying real-world scenarios

or over extended periods, our approach incorporates both of these methods. This makes our evaluation a unique contribution to the field.

4. **Analysis using the RC-Model:** Our implementation within the larger OpenDC framework allows for the analysis of thermal characteristics in contemporary data centers, accommodating extensive setup diversity and various workloads. This analysis is unique in the field and provides valuable insights which may have significant societal impact.

Regarding the **technical contributions**, this thesis enhances the functionality of OpenDC by incorporating the capability to simulate the thermal characteristics of processors. This extension not only serves the current and future user base of OpenDC, including researchers and students, but also potentially benefits data center designers by providing a tool to evaluate the thermal characteristics of their facilities. The code for OpenDC, with the extended functionality, can be found at https://github.com/atlarge-research/opendc

1.5 Thesis Structure

This thesis follows a linear structure, with the first half focused on theoretical processes and the second half delving into practical elements. In Chapter 2, we provide the necessary background to understand the thesis, discussing discrete event simulation and the role of power in temperature. Chapter 3 explores the challenges in modeling the thermal properties of processors and the approaches to overcome these challenges, addressing **RQ1**. In Chapter 4, we address **RQ2** by detailing our design, including the various models and equations that underpin our tool. Chapter 5 answers **RQ3**, describing the evaluation of our design and its effectiveness. Finally, Chapter 6 provides the final answers for our three research question followed by an analysis of the limitations, and the future work needed to fill the gaps, of our tool and simulation.

1.6 Plagiarism Declaration

I confirm that this thesis work is my own work, is not copied from any other source (person, Internet, or machine), and has not been submitted elsewhere for assessment.

2

Simulation and the Role of Power in Thermal

This chapter outlines the fundamental ideas necessary to understand this thesis. The first part explains the concept of discrete event simulation, its applications, and the advantages and disadvantages of employing simulation compared to other methods. The second part explores the different elements that make up power consumption, with subsequent sections explaining the role of power in thermal profile.

2.1 Simulation

In this section, we describe what discrete event simulation is, why it is cost-efficient, fast, and fairly accurate compared to other predictive methods, and why, due to these advantages, it has become a prominent tool in predictive computing.

Simulation is the process through which engineers model real-world situations and scenarios in a virtual environment to study their behavior under various conditions (9). Simulation models are diverse and employ various techniques to achieve their objectives. Generally, three characteristics are used to distinguish different types of simulation models (21):

1. **Static vs. Dynamic:** Static models represent systems at a single point in time, while dynamic models simulate systems over a period of time.

2. **Continuous vs. Discrete:** Continuous models change states continuously, whereas discrete models change states at distinct time steps.

3. **Stochastic vs Deterministic:** Stochastic models incorporate randomness through probability distributions, while deterministic models produce the same output for a given input consistently.

In this work, we consider only *discrete-event simulation*. This approach represents the operation of a system as a sequence of events occurring over time, with the assumption that no changes occur between events. This allows for direct progression from one event to the next, in contrast to continuous models. Discrete-event simulation is widely used to model cloud and datacenter operations due to the immense scale and complexity of datacenters and the long-running nature of experiments.

2.1.1 Advantages and Disadvantages of Simulation

Simulation may not always be the appropriate tool in all cases. This section discusses the reasons why simulation works for data centers while providing context as to when it also does not.

Comparison with real-world testing: Simulation has clear advantages over real-world physical experimentation. Simulation allows researchers to study the behavior of data centers in a much simpler and cost-efficient way. For example, a workload run on OpenDC by researchers took 112 CPU hours while a similar workload on physical hardware would take up to 86 billion CPU hours (8). Not only does this show how powerful a tool simulation can be, saving not only the cost in dollars of the physical hardware but also the cost in time, it also illustrates how simulation can allow researchers to conduct experiments that would have otherwise not been feasible. This, of course, goes to show the clear advantage of simulation over real-world testing.

Comparison with mathematical models: Mathematical models are designed to simulate systems in which the relationship between inputs and outputs can be predicted by a few variables. They are often defined in simple terms, such as linear, logarithmic, or cubic functions, which makes them rigid. Although these models provide a solid understanding of a system – assuming the model is accurate – they usually cannot handle the multitude of factors present in large, complex, real-world datasets (9). Therefore, simulation offers greater capabilities than mathematical models due to their inability to account for the complexity and randomness inherent in data centers (9). Even if we could develop a comprehensive mathematical model that accounts for all significant variables in data center

processes, it would still be incapable of incorporating randomness, as its relationships are statically defined. This is where simulation clearly surpasses mathematical models.

Furthermore, even if a mathematical model could predict system behavior accurately, it might not achieve the same level of precision as a simulation. This is because simulations can run experiments multiple times, with each iteration yielding slightly different results. This allows real-world conditions to be better represented over thousands of iterations, leading to more accurate overall results (9). In contrast, once the relationship is known, mathematical models lead to the same result every time.

Simulation also has disadvantages, of course. In contrast to real-world testing on physical hardware, simulation may be faster and more cost-efficient, but it will always be less accurate (8). Similarly, whilst simulation may be more accurate than mathematical models, it is also less interpretable. The complexity of simulations makes it difficult to discern which variables are causing specific effects in outcomes (8). However, the disadvantages do not prevent simulation from being used in many fields, which is why it has become a more prominent tool in all forms of science (22) (23) (24).

2.2 Power

As high as 99% of the energy provided to a processor is converted into thermal energy, which is dissipated as heat by the processor (25). Therefore, before we provide the background on how a processor generates and dissipates heat, it is first salient to understand how it consumes the electrical energy, i.e. power, that is provided to it in Watts.

There are three main components to CPU power consumption (26):

1. **Dynamic Power** is the power that we think of when we think of the power being "consumed" by the processor. It is the power used to toggle the logic gates inside the processor when it performs computations and runs workflows. Dynamic power is generally calculated via well-defined mathematical models, thus, it is not highly relevant to this work. Moreover, the simulation tool we aim to utilize to validate our model, OpenDC, already has dynamic power models that we can use to aid our thermal calculations.

2. **Idle Power** refers to the power consumption of a processor when it is not actively performing computations. It is the base power consumption of the processor. During this state, processors typically enter a low-power mode where logic gates are not toggled actively, reducing energy consumption while remaining in an operational

state ready to receive new instructions. Idle power values are predetermined by manufacturers and are specified in processor data sheets (27).

3. **Static Power** is the most complicated of the three components to understand as it pertains to the power consumed by the microscopic transistors in modern CPUs when they are in an "off-state" but the power supply is still available. This power is considered "lost" as static or leakage power. Although in architectures larger than 45 nanometers, leakage is not a significant issue, as modern processors continue to shrink beyond 45 nm, leakage can account for 30% to 50% of the total power consumption in a processor (26). Therefore, despite the complexity in accurately calculating static power, the calculation of this component is very critical for our work. Although well-defined models exist for calculating static power (28), due to limitations posed by the requirements – generalizability and interoperability with a discrete event simulator – of our tool, most of these models need to be altered to work for this work.

In summary, understanding how a processor uses and dissipates power is crucial to understanding its thermal characteristics. The three main components – dynamic power, idle power, and static power – each play a distinct role in the overall power profile. Dynamic power is associated with computational activities, idle power is baseline consumption during inactivity, and static power, though more complex to model, becomes increasingly important as transistor sizes decrease. While dynamic and idle power are relatively easy to simulate, accurately modeling static power is essential for precise temperature predictions, especially as technology progresses towards smaller architectures. By analyzing these power components, we lay the groundwork for comprehending processor thermal characteristics, which in turn is used to estimate temperature.

2.2.1 Static Power

Various sources, due to the engineering, physics, and chemistry of the materials within a CPU, contribute to static power. We will look at one illustrative example to understand how leakage can occur and how it is related to temperature: *subthreshold leakage*. This kind of leakage occurs when small amounts of current can still pass through a transistor even when it is supposed to be in an off state because the barriers designed to block the flow of electrons are not completely effective at a microscopic size (26). Furthermore, when the temperature of a CPU is high, the electrons in the current have more energy to overcome the barrier of the transistor, resulting in higher leakage. The low level nature of

static power paired with the limitation that most data center simulators do not simulate architecture at a level this low results in static power being extremely difficult to simulate.

Consequently, whilst many equations exist to simulate this leakage in circuits and processors that directly take into account the leakage sources, they require knowledge about the physical properties – number of transistors, their size, etc. – of the processor that we are simulating (28). This poses a new problem as processor manufacturers typically do not publish these values. However, manufacturers do release data on the leakage they observed on various input and output pins of their processors (27). While these measurements do not fully capture the entire leakage current in a CPU, they can form a basic leakage indication metric across the processor that may be used in models.

Additionally, another significant challenge is that static power increases exponentially with temperature (28). Without a comprehensive understanding of how a processor works, simulating the effects of increasing temperature on static power is highly complex (20). This complexity is exacerbated by our decision to incorporate static power into our temperature calculations, as it creates a circular dependency, posing a Catch-22 problem regarding which parameter should be addressed first. However, (28) and (20) suggest that, within the normal operating temperatures of a processor, a linear model is capable of effectively estimating static leakage. The results in these studies are promising for our approach which is discussed in Chapter 4.

2.3 Electrical Energy to Thermal Energy

Having discussed power in the previous section, it is now pertinent to discuss the intertwined relationship between power and thermal output. As described above, a processor converts all its electrical energy to thermal energy and dissipates it as heat (25). This phenomenon occurs for several reasons, once again owing to the chemical and physical properties of the materials and processes involved. We will describe two of these properties as illustrative examples:

1. **Electrical Resistance:** This is the most fundamental form of heat generation in processors, or any circuit for that matter. Silicon, the material used in most modern processors, is a semiconductor which has a naturally occurring resistance to the flow of electricity through it. The discussion about why or how conductors or semiconductors generate heat through resistance is outside the scope of this thesis, it is sufficient to just know that they do generate heat when current is passed through them. This

resistance results in the charge carrier, silicon in this case, rising in temperature in a phenomenon known as Joule Heating or Joule Effect (29).

2. **Transistor Switching:** When a transistor switches between the on state and off state or vice versa, it passes through a transition stage during which a small amount of current leaks and results in power dissipation (this is similar to the leakage we discussed earlier). This power is dissipated as heat energy. With these switches happening billions of times per second, a significant amount of energy is consumed and dissipated as heat (30).

Therefore, by the law of conservation of energy – which states that *the total energy of an isolated system remains constant* – the energy input to a processor, i.e. electrical energy, is equal to the energy output which, in the case of processors, will be heat energy. Consequently, by simulating the total power usage of a processor, it is possible to estimate the temperature of the processor at a given time as a function of power (25).

2.4 Thermal Energy to Temperature

Having discussed the role that electrical power consumption plays in the thermal output of a processor in the previous sections, we will now discuss temperature – a metric is more interesting to this work and one that is significantly more difficult to estimate. We will begin by describing the differences between thermal energy – which we discussed in the previous sections – and temperature.

Thermal energy is defined as *the total internal energy of a system* (31). Since processors receive only electrical energy as input, according to the law of conservation of energy, the thermal energy dissipated by processors equals the total electrical energy input, as that is the *total energy of the system.* This dissipated thermal energy is commonly measured in watts. While the watt is typically associated with electricity, from an engineering standpoint, it represents the rate of change of energy over time. In the case of processors, it signifies the rate at which electrical energy is converted into heat energy. Therefore, the thermal energy output of a processor is synonymous with the heat energy dissipated, measured in watts, once the electrical energy is transformed into thermal energy. This measurement underscores the direct relationship between electrical power input and thermal energy dissipation in processors.

Conversely, temperature is measured in degrees Celsius and is defined as the *average kinetic energy of particles in a system*, making it a measure of the average thermal energy

in that system (31). In the real world, temperature is recorded by observing changes in the physical properties of materials that respond to variations in thermal energy in their environment.

For processors, temperature is measured using a special type of circuit known as a *thermocouple*, which is embedded within the processor (27). *Thermocouples* are highly sensitive to changes in heat and generate a measurable voltage difference in response to these changes (32). The processor uses this voltage difference to estimate its temperature. Most processors are equipped with multiple thermocouples distributed across the die, enabling them to accurately measure temperature across the entire surface and identify any hotspots (27). This temperature data is utilized by the processor's thermal management system to regulate and manage thermal output, preventing thermal throttling – a process where the processor reduces its clock speed to avoid overheating – and extending the processor's lifespan.

2.5 Processor Temperature and Server Temperature

To end the background discussion on thermal, it is essential to clarify why we have chosen to simulate processor temperature only and not other aspects of the server. The primary rationale is that within the server case, the processor is responsible for a significant portion of the heat dissipated (19). Given the limited time and scope of a bachelor's thesis, focusing on the primary source of heat in servers is a sensible approach.

Secondary reasons for this choice are practical considerations due to limitations of data center simulators. OpenDC, the discrete event simulation framework we intend to utilize to validate our tool, does not simulate the physical location of modules inside a server case, making it nearly impossible to model heat interactions between these modules. Additionally, modeling airflow inside a server case is exceedingly complex and beyond the scope of this thesis. Another limitation is that OpenDC does not simulate the server cases themselves nor their location relative to other cases and racks, rendering simulation of the server case's heat profile unnecessary. OpenDC does, however, simulate the operation of the processor itself comprehensively, thus focusing on a single processor node is more feasible. This narrow focus allows us to keep the scope manageable and eventually extend the work here to other modules in the future.

2.6 Conclusion

In this chapter, we have provided the foundational concepts necessary for understanding this thesis. We began by explaining discrete event simulation, its applications, and its advantages and disadvantages compared to other predictive methods, highlighting its cost-efficiency, speed, and accuracy for modeling data centers. We then explored the different elements of power consumption in processors, emphasizing their roles in thermal profiles. We discussed how processors convert electrical energy into thermal energy and the mechanisms behind this conversion, such as electrical resistance and transistor switching. Finally, we addressed the complexities of estimating temperature from thermal energy and clarified why we focus on simulating processor temperature rather than other server components. This comprehensive overview sets the stage for the subsequent chapters, where we delve deeper into our chosen modeling approach and its implementation.

3

Thermal Modeling

This chapter primarily addresses **RQ1** by outlining the various methods used in the industry for modeling and simulating the thermal characteristics of data centers at the node level. We explain the advantages and disadvantages of each model and discuss why we chose to proceed with the Resistance-Capacitance (RC) model.

3.1 Challenges in Estimating Temperature

Before we describe the different methods used in industry for thermal simulation and temperature prediction, we believe it is pertinent to highlight why exactly thermal modeling is so difficult. The following three challenges are the most important that we faced when designing our own model.

Complexity of Modern Processor Circuits: With the increasing demands placed on modern processors, their designs have become more complex each year. According to Moore's Law, the number of transistors in processors has doubled every two years, leading to architectures that are now as small as 3 nanometers (33). This miniaturization has allowed for greater compute power from the same chip size but has also resulted in highly complex circuit designs. However, the complexity is not solely due to smaller transistors. Requirements for higher clock speeds, parallelization, power efficiency, memory optimization, and even machine learning have driven manufacturers to design separate cores within the same processor, each specializing in different tasks (33, 34). All these intricacies of modern processor designs make them and their internal interactions very challenging to model.

High-level Design of Data Center Simulators: Many data center simulators operate at a high level. They simulate interactions between different components of a data center

at the server or room level. These simulators focus on the overall performance of tasks in an abstract and aggregated manner, rather than on detailed metrics for each individual component (8). This approach allows them to generate information such as total power consumption for the entire data center, which is typically more valuable to researchers and designers. However, because of this focus, these simulators lack the low-level details, such as static power, needed to accurately simulate the temperature of processors.

Generalizability: As already stated, modern processors are complex circuits, and their thermal characteristics depend on very low-level values. Since no two processor models are identical, the ways they consume and dissipate energy also differ (27, 35). This is because the equations governing the flow of heat energy and resistance in a charge carrier require specific information about the material, such as specific heat capacity and size. Thus, Each processor has its own heat dissipation model, usually defined by the manufacturer. Consequently, creating a generalizable model that works for most, if not all, processors poses a unique challenge.

Therefore, it stands to reason that developing a solution that accounts for processor complexity, integrates into the larger framework of a data center simulator, and is generalizable necessitates a complex approach. In the next few sections, we will discuss the three main approaches available in the literature and contextualize them within the scope of these challenges and our research criteria.

3.2 Manufacturer Approach

This approach uses a linear relationship defined by the manufacturer for their processors' thermal output. This linear relationship, published in processor data sheets, serves as a guide for heat sink manufacturers (27, 35, 36).

This linear relationship follows the familiar straight line equation:

$$y = mx + c \tag{3.1}$$

where y is the temperature in degrees Celsius and it either represents T_{case}, the processor's case (also known as the Integrated Heat Spreader (IHS)) or the the processor die, T_{die}. x is the dynamic power consumption of the processor in Watts. m is the slope, or gradient, of the linear function and is defined by the manufacturer at the factory based on real world measurements. c is the y-intercept value which shows the temperature of the processor at idle power consumption and it is also defined by real world measurements (27, 35).

Despite its simplicity, it is worth mentioning because it is indeed a viable The manufacturer method for temperature simulation is arguably the most accurate because it uses data directly from the manufacturer. However, this approach lacks generalizability. Each processor model requires a specific linear relationship provided by the manufacturer, meaning users must input a unique equation for each different processor they wish to simulate. This makes the manufacturer method the least generalizable of the three approaches discussed in this section.

Despite its high accuracy, we opted not to use the manufacturer method for our tool. Instead, we prioritized versatility. Our tool is designed to simulate a wide range of processors, including generic models with standard values, giving users the flexibility to design data centers that meet their specific needs. By focusing on generalizability, we can simplify the temperature calculation process and enhance usability. In the following sections, we will explore more sophisticated methods that achieve generalizability without significantly compromising accuracy.

3.3 Machine Learning Approach

Machine learning is another approach we identified during our brief survey of the field. This approach involves training a machine learning model to predict the temperature of a processor at a given time, t. In this section, we will present two illustrative examples of this approach in the literature and explain why we have decided not to use it for our own design.

The first example is an experiment titled "Exploring the Utility of Graph Methods in HPC Thermal Modeling" (17). The researchers in this experiment use publicly available data from the Marconi100 supercomputer to train two different kinds of models, *Ridge regression* and *Dense Neural Network (DNN)*, to predict temperature at the node-level. A third model is trained for room-level temperature prediction, but that is irrelevant to this thesis as we are only concerned with processor temperature. Table 3.1 shows a list of predictors this study uses for both models and it highlights the data needed to train the models.

Their results show that simpler models are best for predicting temperature, which is also in line with other machine learning literature which typically utilizes linear models. They use *Mean Squared Error (MSE)* to report their results, which is a commonly used metric for assessing the accuracy of machine learning models. The simpler *Ridge* model had a *MSE* of just 1.103 while the more complex *DNN* had a higher *MSE* of 2.261.

name	description
pcie_avg_t-1	average outlet temperature in snapshot $t-1$
pcie_t0	outlet temperature at time instant t
total_power_avg_t-1	average node power consumption in snapshot $t-1$
total_power_avg_t0	average node power consumption in snapshot t
ambient_avg_t-1	average inlet temperature in snapshot $t-1$

Table 3.1: Description of predictors used in (17)

In the **second example**, a study titled "Machine Learning-Based Temperature Prediction for Runtime Thermal Management Across System Components" (37) also used a similar approach to the first example by training both a linear regression model and a neural network model. The models they employed were *Lasso Regression* and *Multilayer Perceptron (MLP)*. These models used both physical and non-physical features to predict temperature. The physical features included voltage and die temperature, while the non-physical features were application-level metrics, which involved learning the temperature based on the application load on the processor.

In this study, the researchers achieved an average absolute error of 2.9 degrees Celsius for the MLP and 3.8 degrees Celsius for the regression model. In percentage terms, these errors correspond to approximately 16% and 21%, respectively, based on an estimation of the total range of temperatures listed.

While both these examples showcase the usefulness and viability of machine learning as a method for temperature estimation and prediction in processors, it also, like the manufacturer approach, shows the lack of generalizability. In both case studies, prior data was needed to train the models and validate them. This prior data is not always available for a new processor – in fact, within the time constraints of this bachelor's thesis, we were unable to find more than one dataset that contained this information. Moreover, even if the data is available, a new model still needs to be trained, validated, and integrated into a larger data center simulator framework for it to be used by the users of a tool like ours. These reasons highlight our unwillingness to use machine learning methods for our tool as generalizability is of high priority to us.

3.4 Physics Approach

The physics-based approach is perhaps the most common method used in literature for estimating processor temperatures. This approach uses a series of equations to model the

entire CPU node as a unified system. It is more generalizable than the other two approaches as it treats the processor as a single component within a larger system and thus any generic processor can be part of that system. Moreover, unlike the manufacturer approach, it does not require detailed specifications of each processor, and unlike the machine learning approach, it does not need prior temperature data for each new processor.

The two studies that form the basis of our physics-based approach are (19) and (20). Both studies propose similar Resistance-Capacitance (RC) circuit models to represent the thermal characteristics of a processor and, by extension, the server. This RC model is built upon foundational models for static and dynamic power, which are further explained in Chapter 4.

It is worth noting that in both studies, the researchers aimed to measure and optimize the performance of cooling systems through their thermal solutions. In (19), the researchers developed a simplified server model for each server rack, positing that the server's internal temperature is primarily determined by the temperature generated by the processor, as the processor is the largest contributor to the heat. They used this hypothesis to create a thermal model for servers in data centers, which, combined with a cooling model, allowed them to calculate the annual energy consumption for cooling in a data center. Their simplified server models provide a strong foundation for our design while also demonstrating how a tool like ours can be extended to include cooling models with relative ease.

In (20), the researchers had a different aim. They developed power models, thermal models, and cooling models within a unified system, intending to optimize all three components together, using feedback from each other. This optimization led to a significant decrease in the overall power consumption of the processor. The methods they used to develop their power and thermal models are particularly relevant to our thesis and have inspired the design of our own model, as described in Chapter 4. Additionally, this paper highlights an important use case for our tool: the optimization of cooling and power in data centers.

However, the simplification inherent to this approach does lead to lower accuracy. This is because it cannot account for variations in the internal temperature of the processor circuit, nor is it based off real-world measured values – it simply relies on known physical laws and equations. Despite this trade-off, the reduction in accuracy is acceptable for this work, as the overall performance is comparable to the accuracy of other approaches listed in this section while being more generlizable than them. Moreover, this approach aligns better with simulation practices which typically rely on using mathematical and physics equations to model and simulate complex interactions.

Both case studies demonstrate the viability of the physics approach as a generally accurate method for modeling temperature. In these studies, the model is built first, using processor temperature data only for validation. This contrasts with the other two approaches, where processor temperature data was essential for building the model. This process highlights that the model is independent of any single processor model, making it highly generalizable and suitable for simulating any processor. Therefore, this model best meets our criteria for generalizability and is our choice for our own model.

3.5 Conclusion

In this chapter, we addressed **RQ1** by exploring various industry methods for modeling and simulating the thermal characteristics of data centers at the node level. We highlighted the challenges of thermal modeling due to the complexity of modern processors, high-level design of simulators, and the need for generalizability. We examined three main approaches: the Manufacturer Approach, the Machine Learning Approach, and the Physics Approach. While each has its strengths and weaknesses, we concluded that the Physics Approach, specifically the Resistance-Capacitance (RC) model, offers the best balance of accuracy and generalizability for our tool. This method aligns with our goal of providing a flexible, broadly applicable solution for simulating processor temperatures in data centers. The next chapter details the design of this RC-model for our thermal modeling tool, along with the equations we used.

4

Design of an RC-Model

This chapter details the design of the RC-Model in relation to our thermal modeling tool, addressing **RQ2**. We first explain the use of the RC circuit and its components, followed by a discussion of the various power models that constitute the thermal model. Additionally, we provide an in-depth explanation of the equations used to model the system, contextualizing them within the discrete event simulator framework.

4.1 RC Circuit

While a comprehensive discussion of a Resistance-Capacitance (RC) circuit is outside the scope of this thesis, it is important to provide a brief overview due to its central role in this thesis. An RC-circuit is, in its most foundational form, an electrical circuit consisting solely of capacitors and resistors connected to a power supply (38). A capacitor is an electrical device used to store electrical charge, while a resistor is any conductor that resists the flow of electrons when current flows through it (38).

Within the context of this thesis, the RC circuit is important because thermal systems can be modeled using equivalent electrical systems (39). As the researchers in (39) show, the governing equations for thermal systems and an equivalent electrical system are the same. Therefore, a solution to the electrical system is also a solution for the thermal system. It is this fact that allows us to use the RC circuit to model a CPU node, which is a thermal system, as an RC circuit and calculate the temperature of the CPU die.

To convert a thermal system to an electrical system, thermal quantities need to be translated into their electrical counterparts. Specifically, thermal resistance is represented as electrical resistance (R), heat flow as current (I), temperature as voltage (V), and thermal inertia (M)—the capacity of a material to conduct and store heat—as electrical

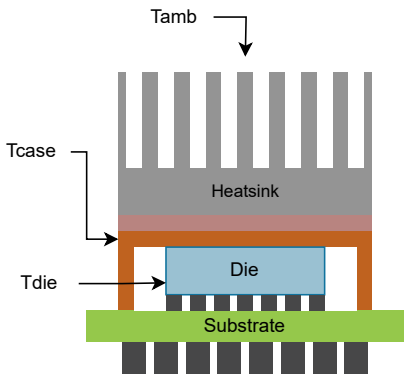

Figure 4.1: Cross sectional diagram of a CPU node (19)

capacitance (C) (39). Using these equivalents, we can construct an RC circuit to model any thermal system. This approach is demonstrated by the researchers in (39), who modeled a refrigerator room as a relatively simple RC circuit.

The reason electrical systems are used as representatives for thermal systems is that thermal systems are highly complex to solve, with the values for thermal quantities being difficult to estimate. RC circuits allow researchers to simplify a complex thermal system into a more easily analyzable electrical system. This approach simplifies the intricate heat transfer and electrical interactions within the thermal system, making the analysis computationally efficient while still providing accurate predictions (40). As the researchers in (40) demonstrate, even very complex thermal systems, such as Thermoelectric Modules (TEMs), can be effectively modeled using RC models. They show that by creating a sufficiently detailed RC model, the accuracy can be within 3% of more precise methods.

The RC circuit we use to model the CPU node in our tool's design is depicted in Figure 4.3, while Figure 4.1 shows a cross-sectional diagram of a generic thermal system for a CPU node. These diagrams illustrate how a CPU node can be converted into a series RC circuit: the three components – the die, the case, and the heat sink – are stacked on top of each other, allowing the flow of heat, or current, to occur sequentially.

As such, we model the resistance of the case (R_{case}) and the heat sink (R_{hs}) as two simple resistors in a circuit. Additionally, we represent the temperatures of the die (T_{die}), the CPU case (T_{case}), and the ambient environment (T_{amb}) as the thermal potential across the circuit. We also model the thermal capacitance of the heat sink (C_{hs}) and the CPU die (C_{die}) (20), although these two capacitors are not shown in the circuit diagram in

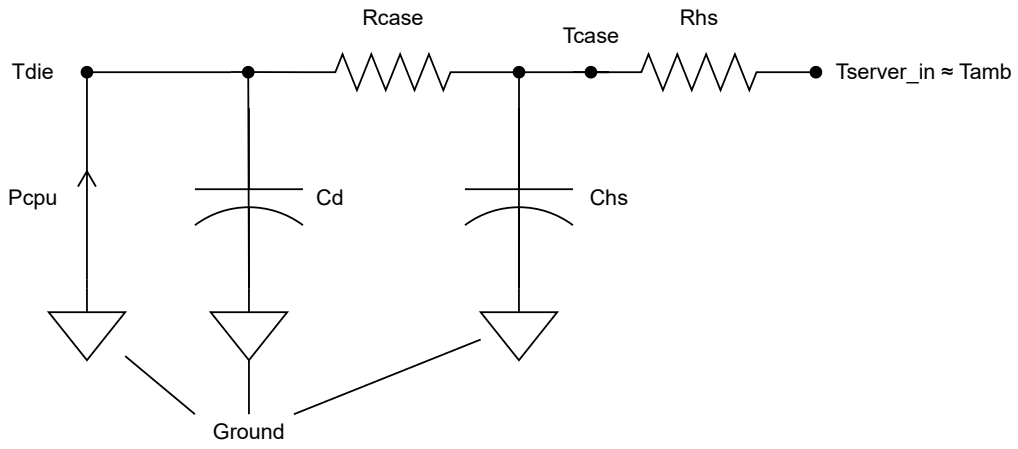

Figure 4.2: Complete RC circuit of a CPU node, showing the capacitors (20)

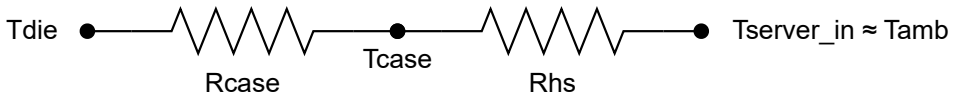

Figure 4.3: Simplified RC circuit of a CPU node, without showing the capacitors (19)

Figure 4.3. This omission is because, as we will demonstrate in the next two sections, capacitance is not relevant to our specific use case of the RC circuit. This is because the equations for calculating the temperature of the die do not involve capacitance (20). However, a complete circuit diagram is provided in Figure 4.2, illustrating what a full RC circuit diagram of a CPU circuit might look like.

4.2 Static Power Model

Having explored how an RC circuit models the thermal system of a CPU node, we now turn our attention to the power equations used to calculate the CPU's total power. These equations form the foundation for the thermal model, which is used to determine the temperature, as illustrated in Figure 4.4 and depicted in the circuit diagram in Figure 4.3.

This section is divided into two parts: the first part presents the dynamic power model used in our tool, and the second part explains the static power model. *Note that idle power, the third element of power, is defined by the manufacturer and, thus, does not require us to model it.*

Dynamic Power: To calculate dynamic power draw for a processor, we use a linear

model to calculate the current power draw based on utilization. This model is already implemented in OpenDC, see Chapter 5 for more information on the implementation.

This model provides us with the dynamic power draw at a given time, t, based on the current utilization. Despite its simplicity, the model is accurate within normal working conditions, as demonstrated during the evaluation in the Evaluation chapter. We prefer this model over more complex ones due to its simplicity and ease of implementation, allowing us to concentrate on the primary goal of this thesis: thermal modeling. The time constraints of a bachelor's thesis do not permit the implementation of a sophisticated power model, but this should certainly be a focus for future research.

Static Power: We had to design and implement our own static power model as it does not currently exist in OpenDC. The equation used to calculate static power is the familiar power equation:

$$P_{static} = I_{IL} \cdot V_{CCIN} \tag{4.1}$$

Where, P_{static} is the total static power, I_{IL} is the total internal leakage or static current, and V_{CCIN} is the supply voltage as defined by the manufacturer (36).

For calculating the leakage current, I_{IL}, we use the approach proposed in (28). In this study, the researchers tested a method for temperature-dependent integrated circuit static power estimation, allowing them to model static power as a linear function of temperature. They demonstrate that, within normal temperature ranges, leakage current is linear rather than exponential, making it easy to predict. However, to apply their approach exactly, we would need to know the minimum and maximum temperatures of each processor in advance. However, this requirement conflicts with our goal of creating a generalized model – a model which requires as little prior data as possible. Moreover, calculating temperature is also one of our goals, therefore we do not necessarily have access to this information in advance. This is the catch-22 problem defined earlier.

Consequently, we need a novel approach to develop a solution that works within the confines of our tool. As such, inspired by this case study, and, considering that our working temperature is already a direct linear function of dynamic power draw, we build a leakage current model as a linear function of dynamic power draw. This approach aligns with our generlizability requirement and simplifies the modeling process.

Therefore, the linear equation used for calculating leakage current is:

$$I_{leakage} = \left(\frac{I_{ILmax} - I_{ILmin}}{P_{max} - P_{min}} \right) \cdot (P_{dynamic} - P_{min}) + I_{ILmin} \tag{4.2}$$

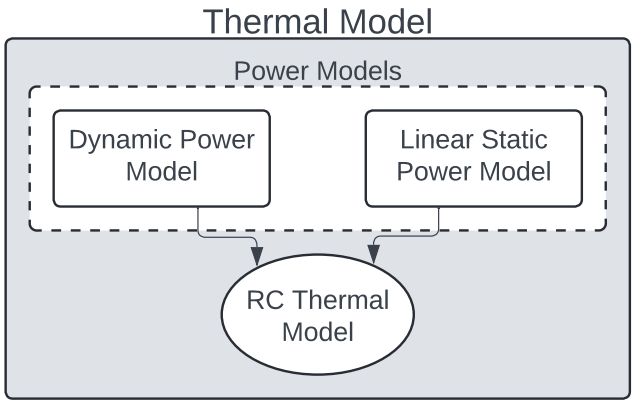

Figure 4.4: Thermal and power modeling framework

Where, $I_{leakage}$ is the leakage current at a given time, t. I_{ILmax} and I_{ILmin} are the maximum and minimum internal leakage, respectively. P_{max} and P_{min} are the maximum and minimum dynamic power draw, respectively, as defined by the manufacturer.

It is also worth noting that for the minimum and maximum leakage currents, we use the values recorded and published by the manufacturer at different input and output signals (36). These values only provide a crude estimation of the total leakage across the processor but, despite their limitations, they represent the best and most generalizable estimates available, as manufacturers typically do not publish comprehensive leakage data for their processors. Furthermore, calculating internal leakage current based on the equations presented in (28) is not feasible. These equations require knowledge of the processor's physical quantities, which are usually protected as trade secrets and intellectual property of the manufacturer.

4.3 Thermal Model

This section explains how we integrate all the concepts from the previous sections on RC circuits and power models into one cohesive thermal model. This unified model will enable us to calculate the temperature of the die effectively.

The thermal model, depicted in Figure 4.4, integrates the different power models to establish a thermal equation. Consequently, the first equation that determines the thermal characteristics of the processor is for the total power dissipated by the CPU. This equation, shown in equation 4.3, defines the total power as the sum of all three power components.

$$P_{CPU} = P_{dynamic} + P_{idle} + P_{static} \tag{4.3}$$

The next two important equations define the total resistance, R_{TOT}, in our circuit diagram shown in Figure 4.3. The first method, described in equation 4.4, involves summing the resistances of the two resistors since this RC circuit is a series circuit. The second method, presented in equation 4.5, calculates R_{TOT} by dividing the thermal potential difference across the circuit by the total power from the CPU.

$$R_{TOT} = R_{hs} + R_{case} \tag{4.4}$$

$$R_{TOT} = \frac{T_{die} - T_{amb}}{P_{CPU}} \tag{4.5}$$

Equation 4.6 unifies all the systems of equations. By substituting all the other equations into this one. We make T_{die} the subject which results in a governing equation for the temperature of the die.

$$T_{die} = T_{ambient} + P_{CPU} \cdot (R_{hs} + R_{case}) \tag{4.6}$$

These concepts form the foundation of a comprehensive thermal modeling tool. Through a series of equations, we establish a framework capable of accurately predicting the temperature of the CPU die. This unified model, validated by the equations presented, forms the core of our thermal modeling tool and sets the stage for further evaluation and application in discrete data center simulations.

4.4 Conclusion

This chapter's contributions to this these are threefold. **First**, we detailed the design of the RC-Model in relation to our thermal modeling tool. We began by explaining the principles and components of the RC circuit and its relevance in modeling thermal systems. By translating thermal quantities into their electrical equivalents, we demonstrated how the RC circuit can effectively represent a CPU node's thermal characteristics.

Second, we delved into the power models essential to the thermal model, outlining both dynamic and static power calculations. The linear models used for dynamic power draw and leakage current provided a balance between simplicity and accuracy, aligning with the constraints and goals of this thesis.

Third, and finally, we integrated these concepts into a comprehensive thermal model, thus answering **RQ2**. By developing a series of equations, we created a tool that accurately predicts the temperature of the CPU die. This unified model is central to our thesis and paves the way for further evaluation and practical application in data center simulations.

5

Evaluation of Thermal Simulation

This chapter addresses **RQ3** by detailing the methodology used to evaluate the RC model and presenting the results of our experiments. We start with an overview of OpenDC, the discrete event data simulator used for testing our tool, and explain how we extended it to integrate the RC model. Next, we describe the dataset employed for validation, followed by a discussion of the experimental setup and the results obtained from our experiments.

5.1 A Primer on OpenDC

As stated in the methodology, in this thesis, we evaluate our thermal model using the OpenDC framework. This section, therefore, briefly introduces OpenDC, a discrete event data center simulator, and its relevant modules. Moreover, here, we will only discuss modules related to the power and thermal operations of OpenDC. A detailed discussion of all OpenDC modules is outside the scope of this work but can be found at *OpenDC 2.0* (8).

OpenDC is an open source discrete event data center simulator, written in Java and Kotlin, developed for research and educational purposes by the @Large research group at the Vrije Universiteit, Amsterdam (8). It is capable of modeling all the major operational layers of many cloud system providers, from data center infrastructure and virtualization to resource management and scheduling. Additionally, it supports various configurations of hardware – including different processors, memory, storage, network – and workloads, which can include machine learning tasks, serverless tasks, and other major cloud services.

Users interact with OpenDC by defining two main components of the input: a workload and a topology. We use these modules to define our own experiments later in this document, therefore it is important to briefly discuss each of the modules here:

1. **Workload:** A workload is the trace of all the actual tasks that need to be simulated on OpenDC. A workload requires two traces: *meta* and *trace*. The *meta* file provides an overview of the requirements for the overall workload. The *trace* file provides the demand over time for each server. These trace files allow OpenDC to simulate the servers, the timestamps, and the demand over time for each server based on what the companion values were in the real-world data. More information can be found in the OpenDC Documentation.

2. **Topology:** The topology is where a user defines the components inside the data center. As stated in the official OpenDC documentation, *"A topology consists of one or more clusters. Each cluster consists of at least one host on which jobs can be executed. Each host consists of one or more CPUs, a memory unit and a power model."* A cluster is an organizational module inside OpenDC, allowing a user to define to multiple different kinds of clusters within one data center, each with its own characteristics, allowing for a single workload to run on different topologies in one simulation. A host, also sometimes referred to as a node, is a single server inside a cluster which can contain many individual processors on which the workload is eventually run.

5.2 Implementation

Having provided a high-level overview of user interactions with OpenDC, we will now focus on the specific modules within OpenDC that are relevant to our work. In this study, we extend the functionality of OpenDC's Power Supply Unit (PSU) module. Since we are adding thermal functionality to OpenDC and, as discussed in Chapter 2, thermal properties are directly related to power, the PSU module gives us access to all the necessary power-related values, such as voltage, current, and dynamic power draw. This access is crucial for implementing the equations of our unified thermal model.

The PSU module in OpenDC supplies power to the simulated machine and is a unique component of each machine. This allows us to extend only this module to estimate the physical properties of the machine, without also modifying the machine's functionality. In fact, even in the real world, processors use the voltage and current values from the PSU to calculate their own temperature (27).

Moreover, we use the **Dynamic Power** model provided within OpenDC to calculate the dynamic power for our thermal model. While OpenDC contains many different models

for power calculation, we use a linear model to calculate the current power draw based on utilization. The linear power draw model follows the familiar $y = mx + c$ format.

Therefore, the equation used in this work for calculating dynamic power draw in OpenDC is:

$$P_{dynamic} = P_{idle} + (P_{max} - P_{idle}) \cdot U \tag{5.1}$$

Where $P_{dynamic}$ is the dynamic power draw of the CPU, P_{idle} and P_{max} is the idle and maximum power of the CPU, respectively, as defined by the manufacturer (36)(35), and U is the active utilization of the CPU.

Finally, as previously mentioned, the user provides OpenDC with the values for maximum power and minimum power. Our tool utilizes these values to calculate the total energy dissipated by the CPU using the power models. This total energy dissipated is then fed into the thermal model, which updates the machine temperature each time a cycle is completed in OpenDC. The data reader module within OpenDC then reads this updated machine temperature and outputs it along with other node-level metrics.

5.3 Dataset

We needed data from a data center that included both temperature and power consumption metrics for each CPU, as well as other necessary metrics to create a workload trace in OpenDC. We found this data from the Marconi 100 (M100) supercomputer. The M100 is a tier-0 supercomputer designed and hosted by CINECA, the High Performance Computing (HPC) center in Italy. Tier-0 represents the highest level of computing performance in the European HPC ecosystem, designed to operate at the largest scales and tackle the most complex and demanding computational challenges. Each node of the M100 is equipped with two Intel Xeon 8160 processors, each containing 24 cores.

This dataset, gathered by researchers in (41), contains temperature data for each core per CPU, providing even more detail than we need. We simplify this data by averaging the temperatures of the individual cores to obtain an average temperature for each CPU. We then take the average of the two CPUs in each node to determine an average temperature for the whole node. This abstraction allows us to effectively utilize OpenDC to validate our tool, as OpenDC does not simulate individual processors within a node but rather treats each node as a single processor. Additionally, we calculate the total node power by averaging the total power consumption of both CPUs. Finally, this dataset contains discrete event readings at 20-second intervals, making it ideal for testing in a discrete event data center simulator.

5.4 Experimental Setup

We conducted six experiments, each simulating a unique trace on OpenDC for one of the six available nodes. For our bachelor's thesis, we limited our study to these six nodes due to the time-intensive nature of creating traces for the workloads from the M100 data. Despite this limitation, we believe that the varying workloads used in these experiments adequately test the accuracy of our tool.

Table 5.1 outlines the topology used in all six of our experimental setups within OpenDC. This topology replicates the node structure of the M100 supercomputer, as previously described. It consists of a single cluster containing two CPUs, each with 24 cores, for a total of 48 cores, matching the Xeon processors in the M100. We assume that the M100 runs its processors at 2.1 GHz, which is the standard configuration listed on their website (42), and that there is no overclocking for these workloads, so we define the core speed as 2100 MHz. Additionally, we collectively define the memory and power of the two CPUs to create a single node in OpenDC with characteristics similar to the combined attributes of the two CPUs.

CPU	
Count	2
Core Count	48
Core Speed	2100 MHz
Memory	
Size	1.4 Gigabytes
Power Model	
Model Type	Linear
Power	400 W
Max Power	350 W
Idle Power	50 W

Table 5.1: CPU, Memory, and Power Model Specifications of the topology simulated in OpenDC

5.5 Results

In this section, we explain the results of the six experiments we conducted and assess the validity of the RC model.

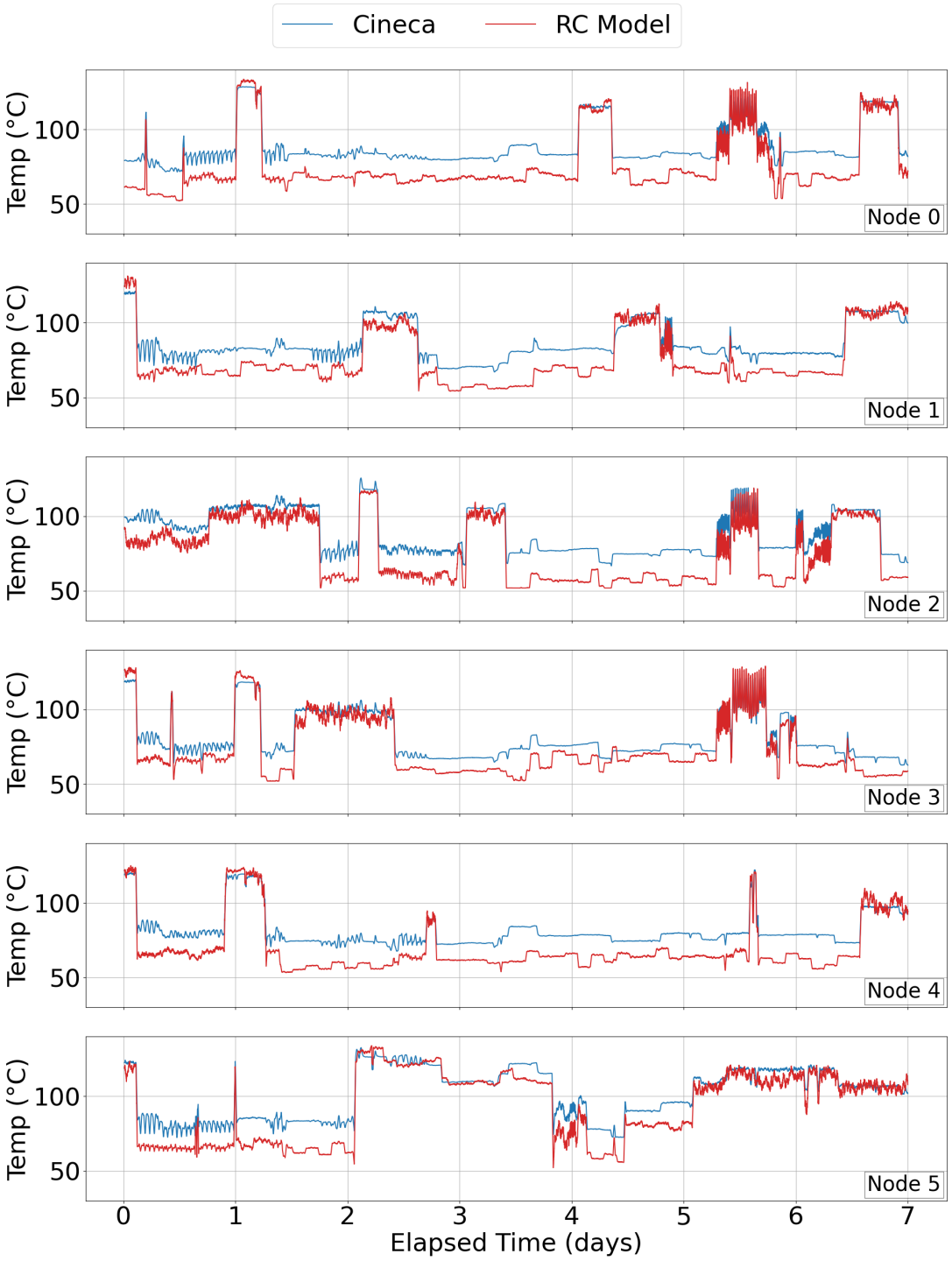

Figure 5.1: Comparison of real-world temperature data (Cineca) and RC Model predictions across six nodes over a 7 day period. The plots illustrate general alignment in temperature trends while also showing the consistent underestimation of temperature by the RC Model.

The graphs in Figure 5.1 compare real-world temperature data from Cineca with simulated temperature outputs from the RC Model across six nodes over a seven-day period. Given the large dataset, which includes over 30,000 temperature measurements for both real and simulated data, we applied a smoothing technique to enhance clarity. Each point in the graphs represents a windowed average of 50 measurements, which helps to highlight underlying trends by reducing the noise that would otherwise obscure the general trend due to rapid fluctuations in temperature from one measurement to the next.

These graphs reveals that both the Cineca data and the RC Model exhibit similar patterns of temperature fluctuations, indicating that the RC Model effectively captures the general thermal behavior and trends of the nodes. However, there are discrepancies in the amplitude and timing of temperature changes, with significant deviations around the 100-hour and 125-hour marks across several nodes. In these instances, the RC Model either overestimates or underestimates the temperature compared to the Cineca data.

A detailed analysis of each node reveals specific strengths and weaknesses of the RC Model. Overall, the RC Model tends to underestimate temperature values, as visible in the error histograms in Figure 5.2. Nodes 0, 2, and 3 exhibit significant oscillations in the RC Model, especially around the 125-150 hour mark. This suggests that using only power to directly inform the temperature of a CPU may not fully capture the transition stages of temperature; as power changes rapidly, so does the RC Model's prediction. Although nodes 1 and 4 show less oscillation, this is due to consistent power draw throughout the workload. This indicates that when power draw and temperature are consistent, our model performs well. However, when they change rapidly, the model struggles to respond accurately. This may require future work to introduce a dampening factor to reduce oscillations.

Node 5 is the best prediction from our RC Model, though it still suffers from underestimation and rapid oscillations. The likely reason for Node 5's accuracy is that the temperature is not consistent, suggesting that our model is more closely able to follow changing temperatures than it is consistent temperatures. This shows that the RC Model can adapt better to varying conditions but needs refinement for better accuracy in stable conditions.

We also plotted histograms of temperature errors between the Cineca data and the RC Model across six nodes, as seen in Figure 5.2. These histograms provide key insights into the model's performance, helping us understand whether the model systematically overestimates or underestimates temperature values.

Each histogram has a central peak around zero, indicating that, on average, the RC Model predictions are reasonably close to the Cineca measurements. However, the spread

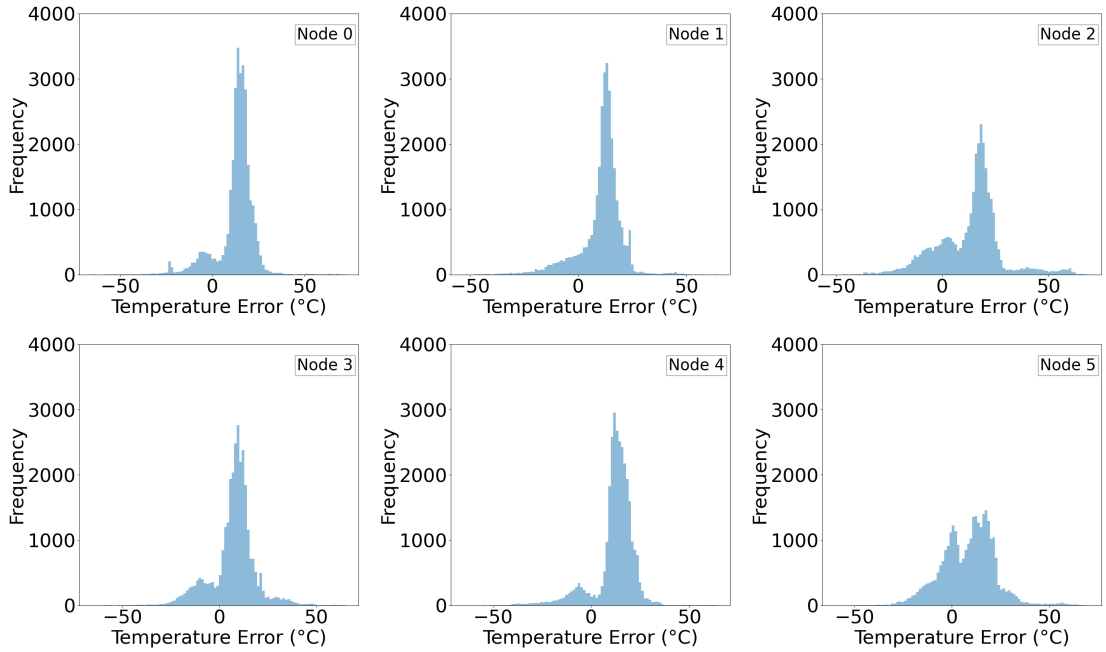

Figure 5.2: Histograms of temperature errors between Cineca data and RC Model predictions across six nodes, showing error distributions and model performance variability. Nodes 1, 3, and 4 exhibit narrower error spreads and better predictive accuracy, while Nodes 2 and 5 display wider spreads and more pronounced deviations, indicating areas for model improvement.

of errors varies across nodes. Nodes 0, 1, 3, and 4 display relatively symmetric error distributions with narrower spreads, suggesting that the RC Model consistently underestimates temperature for these nodes, with errors mostly within the range of -20°C to 40°C.

Nodes 2 and 5, on the other hand, exhibit wider spreads and more pronounced skewness, indicating larger deviations from the Cineca data. The presence of multiple peaks, particularly in Node 5, suggests that the RC Model struggles when the temperature is not consistent and has to predict frequent highs and lows. As shown in Figure 5.1, both these nodes, especially Node 5, experience significant temperature fluctuations throughout the workload. This constant change leads to the model underestimating and overestimating temperatures almost equally.

Despite this, the main issue for the RC Model is the systematic underestimation. We believe this is, at least in part, caused by the underlying power model in OpenDC which also underestimates dynamic power draw, as shown in Figure 5.3. However, it is unclear if this is the only cause as the overall error spread of the dynamic power model is not as wide as that of the RC model. Thus, while there is a need to test other power models, it

is also necessary to experiment with the other parameters of the thermal model, such as testing with other heat sink values.

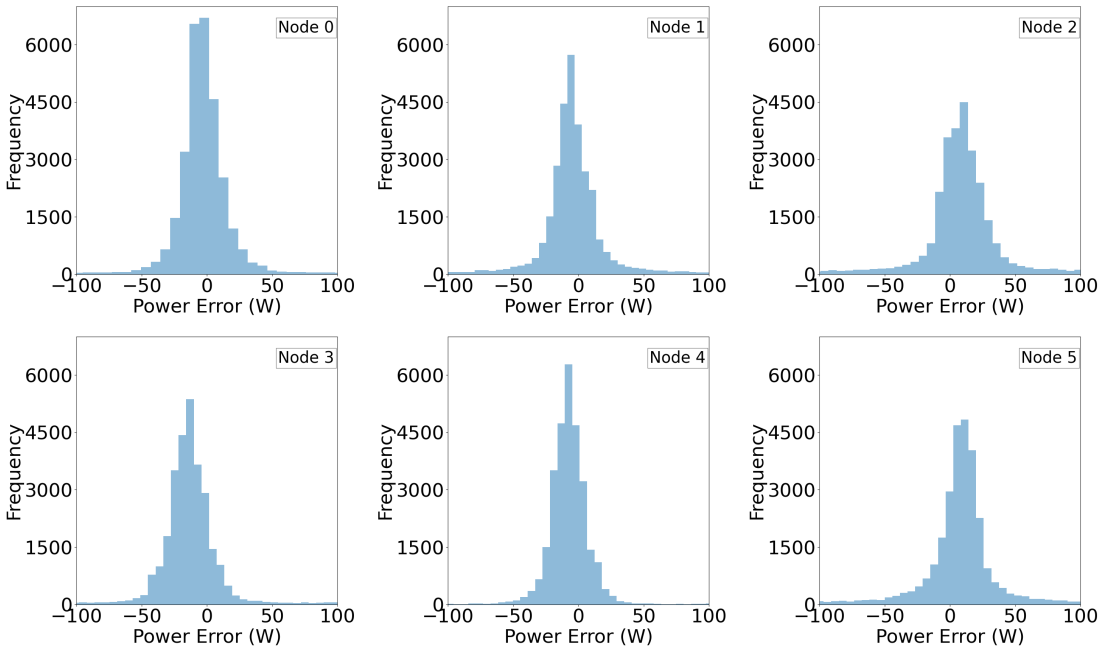

Figure 5.3: Histograms of power errors between Cineca data and RC Model predictions across six nodes, showing error distributions and model performance variability. All the nodes here resemble the behavior of the temperature error histograms, showing the direct effect of power on temperature in our RC-Model.

Moreover, to quantify the performance of the RC Model, we conducted a Mean Absolute Percentage Error (MAPE) analysis. The results are summarized in Table 5.2. MAPE is a common measure of forecasting accuracy (43). It is a relative error measure that uses absolute values to prevent positive and negative errors from canceling each other out. It is calculated using Equation 5.2 (43).

$$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - P_t}{A_t}\right| \times 100 \tag{5.2}$$

Where, n is the total number of temperature observations, A_t is the actual value of temperature observed at time t, P_t is the simulated (or predicted) temperature value.

The MAPE values represent the average percentage errors for each node, with an overall average of 16.5%. These node-specific MAPE values range from 13.6% to 19.9%, indicating where the RC Model performs well and where significant discrepancies exist. The lowest MAPE value is 13.6% for node 5, which, as shown in Figure 5.1, had the least discrepancy

Node	Temperature	Dynamic Power
0	17.6	13.4
1	15.7	14.3
2	19.9	29.3
3	14.1	27.8
4	18.3	18.7
5	13.6	15.6
Avg.	16.5	19.8

Table 5.2: Mean Absolute Percentage Error (MAPE) Results for Temperature and Dynamic Power Across 6 Simulated Nodes

between simulation and real-world data. Conversely, node 2 has the highest MAPE value of 19.9%, likely due to rapid oscillations occurring in short bursts, resulting in consistent errors.

These quantitative assessments highlight the efficacy of the RC model as the MAPE values are within the percentage error range of other thermal modeling case studies discussed in Chapter 3 (17, 19, 37). However, they also suggest that incorporating a dampening factor into our model could theoretically reduce oscillations and decrease the overall error caused by rapid changes in power draw. Moreover, the power MAPE also further reinforces the need to test with other underlying power models in OpenDC.

These findings suggest that while the RC Model is effective in simulating general thermal trends, further refinement is needed to enhance its accuracy in capturing the exact amplitude and timing of temperature fluctuations. Future work should focus on calibrating the RC Model parameters and incorporating dynamic adjustment mechanisms to improve the fidelity of the simulations.

5.6 Conclusion

In this chapter, we presented our methodology and results for evaluating the RC model by implementing it within the OpenDC data center simulator framework. We provided an overview of OpenDC and detailed our extensions to its Power Supply Unit (PSU) module to incorporate thermal functionality for each simulated machine. The results from our experiments, comparing the simulated data to real-world data from the M100 supercomputer, demonstrated that our tool effectively predicts processor temperatures at runtime with an average MAPE of 16.5%, confirming the validity and generalizability of the RC model for temperature estimation in data centers.

6

Conclusion

In this final chapter, we offer an overview of the answers to our three research questions. We first show how we identified the state-of-the art methods for thermal modeling in Chapter 3, then, how we designed our own thermal model and tool based on these approaches in Chapter 4, and, finally, how we validated our tool in Chapter 5. We conclude this chapter, and this thesis, with a a discussion on the limitations of this work as well as the future work needed to improve the limitations.

6.1 Answering Research Questions

RQ1: How to assess the current state-of-the-art of thermal models for processors in data centers? We conducted a brief literature review to identify the best methods for modeling the thermal characteristics of processors. Three different approaches, along with their respective case studies, are presented in Chapter 3. While time constraints inherent to a bachelor's thesis prevented us from conducting a comprehensive review, we believe we have presented a representative sample of the available methods.

RQ2: How to design a thermal model to simulate the thermal characteristics and temperature of processors in data centers? We have designed our own model for simulating the thermal characteristics and temperature of processors in data centers, which we present in Chapter 4. Our model relies on the fact that the properties of thermal systems and their governing equations are similar to the equivalent electrical system. Which means that a thermal system can be translated into an electrical system, enabling us to create a Resistance-Capacitance (RC) circuit to model the thermal system of an entire CPU node. We chose this approach for

its superior generalizability compared to the other two methods we found. This model is then converted to a software tool for external users to interact with, either independently or via an integration with a larger data center simulator framework.

RQ3: How to evaluate and validate a thermal model for processors in data centers? We validated our model by implementing it into OpenDC, a discrete event simulator for data centers, as presented in Chapter 5. This integration allowed us to leverage OpenDC's workload simulation capabilities to simulate a trace from the real world and predict the runtime temperature of the processor via our tool. We showed that our tool is able to follow the temperature trend very closely, with an average MAPE of 16.5%, demonstrating its accuracy and thus validating it.

6.2 Limitations and Future Work

Our implementation has various limitations that will be a focus of future improvements. We list these limitations here, along with future work that could be done to improve our tool.

Dynamic Power Model: The dynamic power model used in our tool is a linear power model. We chose this model because it is the simplest to implement, allowing us to focus more on the thermal model implementation. However, it is not the most accurate model and ignores readily available metrics in OpenDC that could be used to create a better model. The most widely used model for dynamic power consumption, as found in our review, is presented in (19, 20). For future work, this power model should be implemented as part of our tool. Given the significant role dynamic power plays in temperature, incorporating this model could greatly improve the accuracy of our RC model.

Static Current Estimation: For static current estimation, we use the values provided by the manufacturer for leakage at different input and output signals on the processor. This leakage, however, is not representative of the total leakage in a processor. As mentioned in Chapter 2 and shown in (28), calculating static current is extremely difficult and usually requires specific physical information about the processor. Due to time constraints, we had no option but to use the crude estimates provided by the manufacturer. Future work should focus on developing a generalizable solution for calculating static current. As processor architectures continue to become smaller, accurately modeling static current will be crucial for overall power consumption estimation and, consequently, for improving our thermal tool.

Static Power Testing: The static power model we developed is a novel approach to static power estimation, designed specifically to align with our research criteria and integrate with our tool. However, we were unable to test our static power model due to time constraints. Consequently, it is important not only to refine the estimation of leakage current but also to find a dataset that includes static power values for processors during runtime to validate our static power model against those values. Therefore, future work should focus on testing the static power models within OpenDC, enabling other researchers and OpenDC's user base to utilize them effectively.

Validation with other Processor Models: As discussed in Chapter 5, we validated our tool using only one type of processor, the Intel Xeon 8160 (35, 36). While our evaluation methodology allows us to make reasonable claims of generalizability – demonstrated by our model's performance on six different nodes and six different workloads and the fact that we did not consider this processor's specific architecture when designing our tool – to truly prove the model's generalizability, we need to validate it using datasets from other processor models as well. Unfortunately, due to the scope of this bachelor's thesis as well as the limited availability of datasets for other processor models, we only focused on one processor model. Future work should focus on validating the RC model's results with other processors.

Cooling Models and Thermal Management: Due to the complexity of creating a thermal modeling tool, combined with time constraints and the scope of a bachelor's thesis, we were unable to develop a thermal management system that could fully leverage our RC model. Since thermal management systems consume up to 40% of the electricity supplied to data centers, simulating them is essential to understand the total power consumption of a data center. OpenDC's user base would also greatly benefit from the implementation of cooling models as companions to the RC thermal model. Therefore, future work should focus on creating and implementing cooling models in OpenDC and further validating the RC model with these cooling models in place.

7

Reproducibility

7.1 Abstract

The document outlines the process for accessing and using the RC Model software prototype implemented in OpenDC. It also includes step-by-step instructions on how to run the RC Model within OpenDC using the same traces from the M100 that were used in our experiments. Following these instructions will enable you to replicate our results.

7.2 Artifact check-list (meta-information)

- **Program: OpenDC**
- **Compilation: OpenJDK Amazon Corretto 19**
- **Model: Resistance-Capacitance (RC) Circuit Model**
- **Data set: M100 (41)**
- **Metrics: Mean Absolute Percentage Error**
- **Output: Simulated temperature of nodes**
- **Experiments: 6 traces, simulated on OpenDC**
- **How much time is needed to prepare workflow (approximately)?: $<$ 5 minutes**
- **How much time is needed to complete experiments (approximately)?: $<$ 5 minutes**
- **Publicly available?: Yes**
- **Code licenses (if publicly available)?: MIT License**

7.3 Description

7.3.1 How to access

We have created a GitHub repository for the express purpose of recreating our experiments. That repository can be found on GitHub at RCModel. This repository also contains a README file with instructions on how to use the repository to recreate the experiments we conducted and plot the results like we did. This repository does not contain the actual source code for the RC Model, that can be found on the official OpenDC repository at AtLarge/opendc.

7.3.2 Datasets

The dataset used to validate the RC Model comes from the M100 supercomputer and is available at (41). A subset of this dataset, which we used in our experiments, is included in the RCModel repository to facilitate easy replication of the experiments. Additionally, we created traces from this dataset, which are necessary for simulation on OpenDC. These traces are also available in the RCModel repository.

7.4 Installation

OpenDC itself is publicly available and can be installed from the OpenDC repository. There are no requirements for running OpenDC locally except a valid JDK version. We used Amazon Corretto 19.0.2, which can be downloaded for free from Amazon Corretto or from within the Jet Brains IntelliJ IDE.

To run the experiments, no formal installation is needed. Simply follow these steps:

1. Clone the repository from RCModel.
2. Install the libraries listed in the requirements file.
3. Run the main script.

Additionally, we recommend using Python 3.9, as that is the version we used.

7.5 Evaluation and expected results

Once the experiments repository is running on the local machine, the expected results should match those presented in this thesis. Specifically, you should obtain, per node:

1. Six line plots, each showing the difference between real-world temperatures and simulated temperatures for one node.
2. Six histograms that display the error distributions for temperature.
3. Six histograms that show the error distributions for power draw.
4. A table listing the Mean Absolute Percentage Error (MAPE) values for all six nodes, covering both temperature and power draw.

From the line plots, it should be immediately visible that the RC Model is able to follow the temperature trend accurately, despite frequently underestimating the temperature. The histograms should clearly show a systematic bias towards underestimation, which is attributable to the underestimation of power draw within OpenDC. Lastly, the MAPE values should demonstrate that the general accuracy of the model is within 20%.

References

[1] Khosrow Ebrahimi, Gerard F. Jones, and Amy S. Fleischer. **A review of data center cooling technology, operating conditions and the corresponding low-grade waste heat recovery opportunities**. *Renewable and Sustainable Energy Reviews*, **31**:622–638, 2014. 1

[2] Microsoft Azure. **Service Level Agreement for Microsoft misc Services**. https://www.azure.cn/en-us/support/sla/summary, 09 2020. Accessed: 2024-05-26. 1

[3] Gunnar Schomaker, Stefan Janacek, and Daniel Schlitt. **The Energy Demand of Data Centers**. In Lorenz M. Hilty and Bernard Aebischer, editors, *ICT Innovations for Sustainability*, pages 113–124, Cham, 2015. Springer International Publishing. 1, 2

[4] CBS. **Electricity supplied to data centres 2017-2019**, 2020. Accessed: 2024-06-11. 2

[5] Jiangtao Zhang, Hejiao Huang, and Xuan Wang. **Resource provision algorithms in cloud computing: A survey**. *J. Netw. Comput. Appl.*, **64**:23–42, 2016. 1

[6] S. Ricciardi, D. Careglio, G. Santos-Boada, J. Solé-Pareta, Ugo Fiore, and F. Palmieri. **Saving Energy in Data Center Infrastructures**. *2011 First International Conference on Data Compression, Communications and Processing*, pages 265–270, 2011. 1

[7] Hafiz M. Daraghmeh and Chi-Chuan Wang. **A review of current status of free cooling in datacenters**. *Applied Thermal Engineering*, **114**:1224–1239, 2017. 1

[8] Fabian Mastenbroek, Georgios Andreadis, Soufiane Jounaid, Wenchen Lai, Jacob Burley, Jaro Bosch, Erwin Van Eyk, Laurens Versluis, Vincent Van Beek, and Alexandru Iosup. **OpenDC 2.0: Convenient modeling and simulation of emerging technologies in cloud datacenters**. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 455–464. IEEE, 2021. 1, 6, 10, 11, 18, 31

[9] Kristin L. Sainani. **What is Computer Simulation?** *PM R*, **7**, 2015. 1, 9, 10, 11

[10] Ali Habibi Khalaj and Saman K Halgamuge. **A Review on efficient thermal management of air-and liquid-cooled data centers: From chip to the cooling system**. *Applied energy*, **205**:1165–1188, 2017. 2

[11] Ahmad Faiz, Sotaro Kaneda, Ruhan Wang, Rita Osi, Prateek Sharma, Fan Chen, and Lei Jiang. **LLMCarbon: Modeling the end-to-end Carbon Footprint of Large Language Models**, 2024. 2

[12] Agam Shah. **Making sense of Sam Altman's $7 trillion AI chips gambit**, 2024. 2

[13] Deborah Schalm. **Granulate Issues Findings from State of Cloud Computing Survey Highlighting Underutilization of IT Infrastructure**, March 24 2021. Accessed: 2024-07-12. 3

[14] Umesh Singh, Amarendra Singh, S Parvez, and Anand Sivasubramaniam. **CFD-Based Operational Thermal Efficiency Improvement of a Production Data Center.** In *SustainIT*, 2010. 3, 4

[15] Baptiste Durand-Estebe, Cédric Le Bot, Jean Nicolas Mancos, and Eric Arquis. **Data center optimization using PID regulation in CFD simulations**. *Energy and Buildings*, **66**:154–164, 2013. 3

[16] Waleed A. Abdelmaksoud, H. Ezzat Khalifa, Thong Q. Dang, Roger R. Schmidt, and Madhusudan Iyengar. **Improved CFD modeling of a small data center test cell**. In *2010 12th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, pages 1–9, 2010. 3

[17] Bruno Guindani, Martin Molan, Andrea Bartolini, and Luca Benini. **Exploring the Utility of Graph Methods in HPC Thermal Modeling**. In *Companion of the 15th ACM/SPEC International Conference on Performance Engineering*, pages 106–111, 2024. 3, 19, 20, 39

[18] Baver Ozceylan, Boudewijn R Haverkort, Maurits De Graaf, and Marco ET Gerards. **A generic processor temperature estimation method**. In *2019 25th International Workshop on Thermal Investigations of ICs and Systems (THERMINIC)*, pages 1–6. IEEE, 2019. 3

[19] Sang-Woo Ham, Min-Hwi Kim, Byung-Nam Choi, and Jae-Weon Jeong. **Simplified server model to simulate data center cooling energy consumption**. *Energy and Buildings*, **86**:328–339, 2015. 3, 4, 15, 21, 24, 25, 39, 42

[20] 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. 4, 13, 21, 24, 25, 42

[21] Fabian Mastenbroek. *Radice: Data-driven Risk Analysis of Sustainable Cloud Infrastructure using Simulation*. Master's thesis, Delft University of Technology, 2022. 9

[22] S. Hoban, G. Bertorelle, and O. Gaggiotti. **Computer simulations: tools for population and evolutionary genetics**. *Nature Reviews Genetics*, **13**:110–122, 2012. 11

[23] Mercedes-AMG Petronas Formula One Team. **How Does F1 Simulation Work**, 2024. Accessed: 2024-06-11. 11

[24] Hannes Sakulin. *Design and Simulation of the First Level Global Muon Trigger for the CMS Experiment at CERN*. PhD thesis, Vienna U., 2002. Presented on 2002. 11

[25] Ivan Lavrov. **Temperature of the Central Processing Unit**. *Undergraduate Journal of Mathematical Modeling: One + Two*, **7**(1):Article 3, 2016. 11, 13, 14

[26] Hameedah Sultan, Gayathri Ananthanarayanan, and Smruti R Sarangi. **Processor power estimation techniques: a survey**. *International Journal of High Performance Systems Architecture*, **5**(2):93–114, 2014. 11, 12

[27] Intel Corporation. **Intel Core i7-900 Desktop Processor Series Datasheet**, 2010. Accessed: 2024-05-26. 12, 13, 15, 18, 32

[28] Yongpan Liu, Robert P Dick, Li Shang, and Huazhong Yang. **Accurate temperature-dependent integrated circuit leakage power estimation is easy**. In *2007 Design, Automation & Test in Europe Conference & Exhibition*, pages 1–6. IEEE, 2007. 12, 13, 26, 27, 42

[29] COMSOL. **The Joule Heating Effect**, 2017. Accessed: 2024-06-11. 14

[30] Neil Storey. *Electronics, A Systems Approach*. Pearson Education Limited, 2017. 14

[31] Gaelen Erickson and Andree Tiberghien. **Heat and Temperature**. In Rosalind Driver, Edith Guesne, and Andree Teberghien, editors, *Children's Ideas in Science*, pages 52–54. McGraw-Hill Education, London, United Kingdom, 1985. 14, 15

[32] Bob Perrin. **The Basics of Thermocouples**. *Circuit Cellar*, 1999. 15

[33] Michelle Ehrhardt. **Apple M1 Chip: Specs, Performance, Everything We Know**. *Tom's Hardware*, 2021. Accessed: 2024-07-12. 17

[34] Danijela Jakimovska, Aristotel Tentov, Goran Jakimovski, Sashka Gjorgjievska, and Maja Malenko. **Modern processor architectures overview**. In *XVIII ICEST Conference, Bulgaria*, pages 194–197, 2012. 17

[35] Intel Corporation. **Second Generation Intel® Xeon® Scalable Processors: Thermal/Mechanical Specifications and Design Guide**, April 2019. Document Number: 338747-001US. 18, 33, 43

[36] Intel Corporation. **Second Generation Intel® Xeon® Scalable Processors: Electrical Datasheet Volume One**, April 2019. Document Number: 338845-001US. 18, 26, 27, 33, 43

[37] Kaicheng Zhang, Akhil Guliani, Seda Ogrenci-Memik, Gokhan Memik, Kazutomo Yoshii, Rajesh Sankaran, and Pete Beckman. **Machine learning-based temperature prediction for runtime thermal management across system components**. *IEEE Transactions on parallel and distributed systems*, **29**(2):405–419, 2017. 20, 39

[38] Samuel J. Ling, William Moebs, and Jeff Sanny. *University Physics Volume 2*. OpenStax, 2016. 23

[39] Mohammad Ali Fayazbakhsh, Farshid Bagheri, and Majid Bahrami. **A Resistance–Capacitance Model for Real-Time Calculation of Cooling Load in HVAC-R Systems**. *Journal of Thermal Science and Engineering Applications*, **7**, 12 2015. 23, 24

[40] Alvaro Martinez. **Resistance-capacitance thermal models as alternatives to finite-element numerical models in the simulation of thermoelectric modules for electric power generation**. *Energy Conversion and Management*, **292**, 2023. 24

[41] Andrea Borghesi, Carlo Di Santi, Matteo Molan, et al. **M100 ExaData: a data collection campaign on the CINECA's Marconi100 Tier-0 supercomputer**. *Scientific Data*, **10**(1):288, 2023. 33, 45, 46

[42] CINECA. **Marconi 100**, 2012. Accessed: 2024-07-01. 34

[43] Arnaud de Myttenaere, Boris Golden, Bénédicte Le Grand, and Fabrice Rossi. **Mean Absolute Percentage Error for regression models**. *Neurocomputing*, **192**:38–48, 2016. Advances in artificial neural networks, machine learning and computational intelligence. 38