Monitoring, Modelling and Identification of Data Center Servers

Author:
Martin Eriksson

Supervisors:
Damiano Varagnolo
Arash Mousavi
Riccardo Lucchese

Academic year 2016/2017

Acknowledgments

I would like to thank my advisors Damiano Varagnolo, Arash Mousavi, and Riccardo Lucchese for giving me the opportunity to work with you and your expert advise throughout the thesis. I would also like to thank Filip Blylod and all the staff at Swedish Institute of Computer Science ICE (SICS) for supporting this work with their time and technical expertise.

Abstract

Energy efficient control of server rooms in modern datacenters can help reducing the energy usage of this fast growing industry. Efficient control, however, cannot be achieved without: *i)* continuously monitoring in real-time the behaviour of the basic thermal nodes within these infrastructures, i.e., the servers; *ii)* analyzing the acquired data to model the thermal dynamics within the data center. Accurate data and accurate models are indeed instrumental for implementing efficient datacenters cooling strategies. In this thesis we focus on Open Compute Servers, a class of servers designed in an open-source fashion and used by big players as Facebook. We thus propose a set of appropriate methods for collecting real-time data from these platforms and a dedicated thermal model describing the thermal dynamics of the CPUs and RAMs of these servers as a function of both controllable and non-controllable inputs (e.g., the CPU utilization levels and the air mass flow of the server's fans). We also identify this model from real data and provide the results so to be reusable by other researchers.

Contents

Chapter 1

Introduction

Datacenters are prominent infrastructures of the current data-driven societies, and they are becoming more numerous, bigger in size, and more complex in terms of the number and behavior of their internal components. Datacenters consume considerable amounts of energy: in 2013 datacenters consumed an average power of 11.8GW in Europe alone. This is approximately 3% of the total electrical power produced across the continent and with this a yearly overall value of 38.6 million tonnes of CO^2. One strive is thus to try to make them as energy-efficient as possible. By implementing the best practices in datacenters energy savings of 15,500GWh per year are predicted in the EU. This is approximately equivalent to the energy consumed by 1 million European households yearly, 1.1 billion euro in electricity costs saved in the EU and 5.4 million tonnes less of CO^2 emissions [42].

Although improving the energy efficiency of data centers can be achieved starting from different perspectives, modeling and control play a vital role in addressing this important issue. For instance, as mentioned in [30], three of the best practices in attaining energy efficiency in datacenters are *Optimize Air Management*, *Design Efficient Air Handling*, and *Improve Humidification Systems and Controls*. Since all of these three best practices require appropriate modeling and efficient control, here we focus on how to improve thermal cooling in datacenters (known to account for between $40\% - 50\%$, depending on the studies, of the total energy usage in data centers [23, 25]).

On the other hand, the architecture of the thermal control strategies in datacenters typically comprises three different layers: server level, rack level and datacenter lever. Here we focus on the lowest one, and explicitly consider the servers designed in the Open Compute Project (OCP) [13]. This choice reflects our belief that working with an open source hardware platform will be beneficial since our results can provide: *i)* collaborative opportunities to researchers and developers around the globe; *ii)* information for developers in the OCP program to enhance the efficiency of the current servers; *iii)* a verifiable methodology exploitable by designers of other types of servers during the design stage.

1.1 Introduction to data centers

A data center is a building specified for hosting servers (i.e., computer systems that run software applications that provide services for other programs or devices), and their peripherals (see an example in Figure 1.1). These software applications can be anything form running financial transactions or simulations to cloud based services, like social networking or cloud storage services.

Figure 1.1: TierPoint datacenters, Dallas [1].

Data centers are the pillar of today's The Information Technology system (IT) society, since technologies like banks, hospitals, governments, everything that is IT-based relies on data centers. As the mobile and the internet community is growing for each day, also the dependency of data center storage is growing too. Data centers are not only vital in an IT-based view, they are also very beneficial from economical standpoints. For example, the establishment of Facebook's data center in Luleå contributed in 2012 to approximately 1.5% of the local areas entire economy, and is forecasted to generate around 9 billion SEK of revenues, and engage 4,500 full-time jobs over the course of ten years nationwide [54]. As this example shows, the data centers industry has become a billion scaled market with enormous economical benefits.

1.2 The structure of data centers

There are many different types of data centers build for a lot of different applications. The simplest way of categorising them is by size:

- **mini\small size data centers:** these systems employ a hundred racks or less, and are normally used for smaller businesses, like experimentation facilities (e.g., the small 10-racks data center module hosted by Swedish Institute of Computer Science ICE (SICS) in Luleå [18], Figure 1.2, or modular server containers, e.g., the Sun Microsystems Modular Datacenter shown in Figure 1.3);
- **mid size data centers:** these systems employ between a hundred to a thousand racks, and are typically used for medium size businesses;
- **large size data centers:** these systems host more a thousands racks, and are normally used by huge corporations like Microsoft, Google and Facebook. These premises may be very wide; for example, the Facebook's data center in Luleå, Sweden is shown in Figure 1.5.

The infrastructure within a data center can be structured into three main components:

- IT, analyzed in Section 1.2.1;

Figure 1.2: The SICS datacenter, Luleå (courtesy of Pär Bäckström).

Figure 1.3: A Sun Microsystems Modular Datacenter [2].

DATA CENTER SIZE			
Size Metric	Rack Yield	Compute Space	
		SQFT (ft^2)	SQM (m^2)
Mega	>= 9,001	>= 225,001	>= 22,501
Massive	3,001 – 9,000	75,001 – 225,000	7,501 – 22,500
Large	801 – 3,000	20,001 – 75,000	2,001 – 7,500
Medium	201 – 800	5,001 – 20,000	501 – 2,000
Small	11 – 200	251 – 5,000	26 – 500
Mini	1 – 10	1 – 250	1 – 25

Figure 1.4: Table summarizing the categorization of different data centers by size as proposed in [6].

Figure 1.5: Aerial view of the Facebook datacenter in Luleå [5].

- The Cooling Technologies system (CT), analyzed in Section 1.2.2;
- The power distribution system (PD), analyzed in Section 1.2.3.

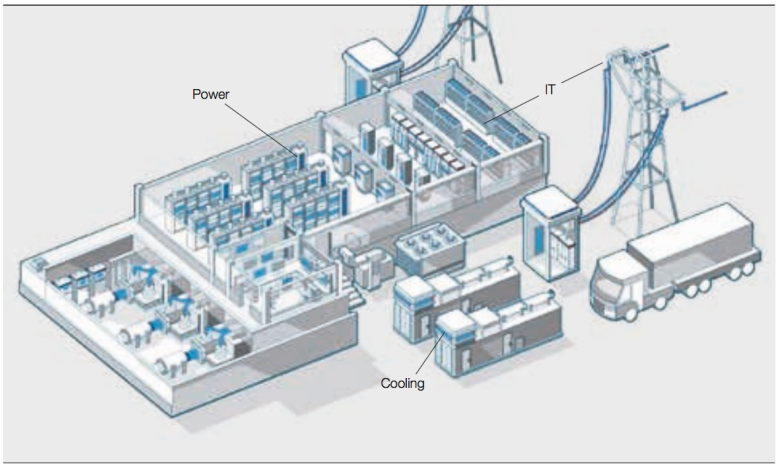

Figure 1.6: Physical layout of a generic data center [19].

1.2.1 The Information Technology system

The IT system comprises mainly the computer systems, i.e., the servers, the storage devices, and the network hardware. The IT system is the main one within a data center, since it provides the services that should be provided by the data center itself (typically a large variety of software applications, like virtualization, databases, web hosting, operating systems, and clouds computing). The IT components uses a huge amount of energy and nearly all of this energy is

eventually transformed into heat. All this heat needs then to be rejected; this is where the CT system (which is explained in details in Section 1.2.2) comes in.

1.2.2 The Cooling Technologies system

As said in the previous subsection, the IT systems produce a huge amount of heat that needs to be removed. There are many different architectures for cooling data centers, but the main ones are Chilled water systems, Computer Room Air Conditioning (CRAC) systems, Rear door cooling and Liquid cooling systems. All these types of strategies are described in details in the following subsections.

1.2.2.1 Chilled water system

A chilled water system (also called a Computer Room Air Handling (CRAH) system) is a combination of a CRAH joined together with a water chiller (see Figure 1.7 for a schematic view). The CRAH cools the air inside the IT room by drawing the warm air across chilled water coils. The heated water circulates in the coils of a water chiller which is removing the heat with the help of a cooling tower. The cooling towers are located outside the data center and cool the circulated water by spraying it on an opportune fill (i.e., a sponge-like material). The water is thus cooled by flowing through the fill, that let some of it evaporate; this eventually gives the same cooling effect as the transpiration of sweat in a human body. Notice that the design of the CRAH system can vary depending on the type of chiller used in the system. Typical designs are water-cooled chillers, glycol-cooled chillers and air-cooled chillers [26].

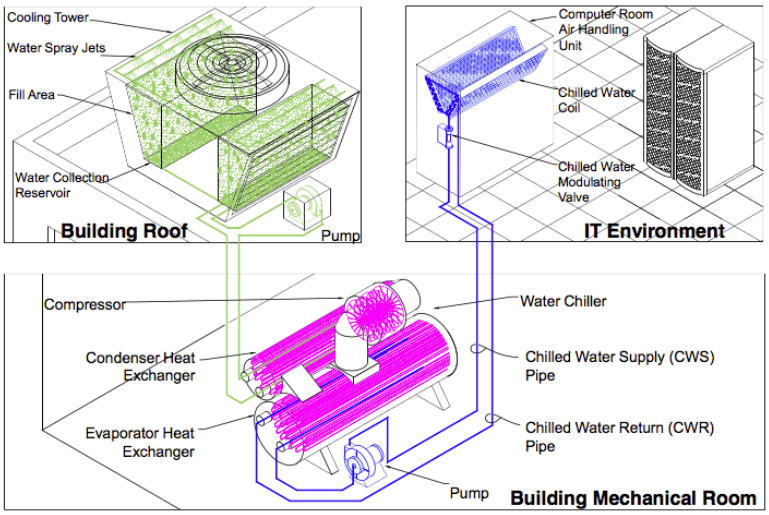

Figure 1.7: Example of a chilled water system [26].

1.2.2.2 Computer Room Air Conditioning systems

The CRAC system can be constructed in three different ways: as an air-cooled CRAC; as a glycol-cooled CRAC; or as a water-cooled CRAC (see Figure 1.8 for their schematic views). An air-cooled CRAC system is typically constructed by joining an air-cooled CRAC and a condenser together. The heat in the IT room is in this way removed by blowing air through the evaporator

coils that are inside the CRAC (normally from top to bottom). The evaporator coils are then connected by opportune refrigerant lines in a loop with the condenser. In the pipes a refrigerant is circulated with the help of a compressors. The refrigerant cools the coils inside the CRAC by evaporation and then removes the heat to the outside environment by condensation through the condenser. This is called a Direct Expansion (DX) refrigeration cycle. Glycol[1] based CRAC systems operate in a similar fashion as the air based ones, except that in this case the entire DX refrigeration cycle is contained inside the CRAC system. The heat is transported from the refrigeration cycle using a heat exchanger, that gathers the heat in the glycol liquid, and then circulates it with the help of a pump to an outside standing dry cooler. The dry cooler then dissipates heat from the glycol to the outside environment. Finally, water-cooled CRAC systems use the same principles as glycol based ones, but instead of glycol they employ water to be pumped to a cooling tower (as mentioned in Section 1.2.2.1) [26].

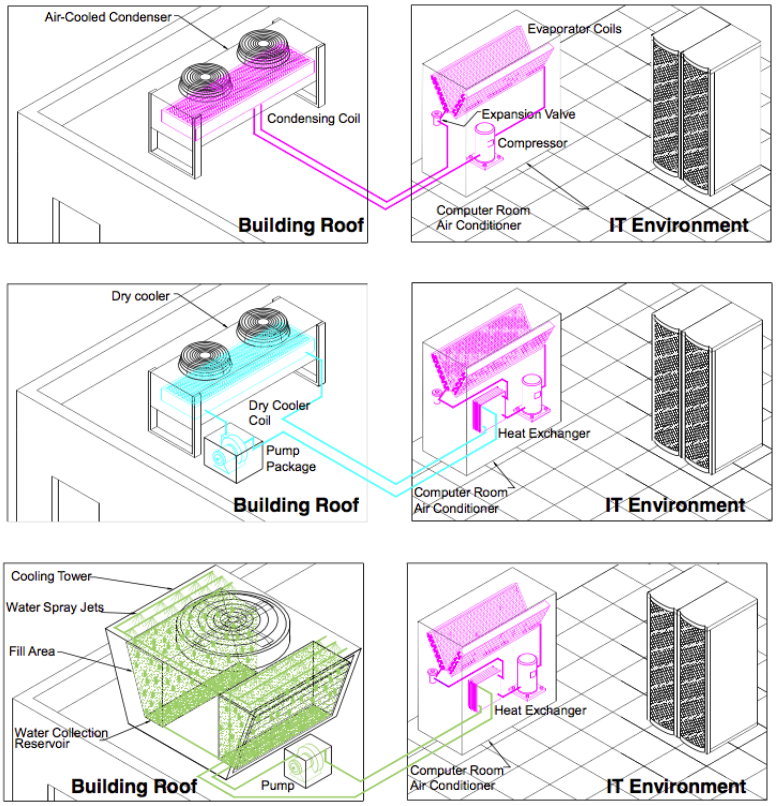

Figure 1.8: Air-cooled CRAC (above), Glycol-cooled CRAC (middle) and Water-cooled (below) CRAC. [26]

1.2.2.3 Rear door cooling system

Rear door cooling systems operate in such a way that each rack is equipped with a rear mounted cooling door. In its turn the cooling door is equipped with fans that draw the warm air from the rack towards a heat exchanger mounted on the door itself (usually so-called air-to-liquid heat

[1] A liquid blend of ethylene glycol and water.

exchangers, i.e., pipes with circulated water). This circulated water is then cooled with the help of a external cooling option, e.g, a chiller, a dry cooler or a cooling tower. Notice that this type of systems is used in combination with chilled water or CRAC-based systems.

1.2.2.4 Liquid cooling

Another cooling technique that is becoming more and more popular is to exploit liquid cooled systems. Since this thesis focuses on air cooled data centers, liquid cooling is not specifically considered here, but just mentioned for completeness.

In brief, there are two main techniques for liquid cooling: on-chip liquid cooling, and submerged liquid cooling. In the on-chip liquid cooling systems cold-plate heat exchangers are mounted in direct contact with the IT components and cooled by passing some liquid coolant thorough opportune micro-channels within the cold-plate. In submerged liquid cooling systems, instead, the entire server racks are submerged in a dielectric fluid inside an enclosed cabinet. This dielectric fluid then releases heat to an opportune heat exchanger, so that the overall temperature of the dielectric fluid remains low enough.

1.2.3 The power distribution system

Notice that data centers need very stable electricity supplies, thus they are typically placed in places where there is a lot of electricity available and redundancy in the distribution network. For example, Northern Sweden is a renown location for deploying data centers thanks to its modern infrastructure in delivering electrical power and to being a stable source of renewable energy [12].

The data centers subsystems related to the power distribution are Power Distribution Units (PDUs), power conditioning units, backup batteries and generators. There are different ways to construct a power distribution system inside a data center, but the infrastructure of the system mainly depends of the size, the flexibility and the mobility requirements of the data center. The power distribution infrastructures typically used today are: panelboard distributions, traditional field-wired PDUs, traditional factory-configured PDUs, floor-mounted modular power distributions, modular overheads or underfloor power busway distributions (see Figure 1.9 for a schematic summary of these strategies). These technologies are then discussed in details in the next subsections.

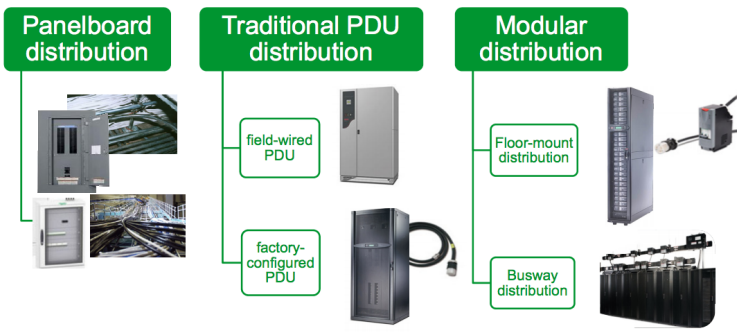

Figure 1.9: Summary of the most typical approaches to implement power distribution within a data center [52].

1.2.3.1 Panelboard distribution

Panelboard distribution units are mainly used for smaller installations, and when keeping a low initial cost is a priority over flexibility and mobility. This type of distribution is composed by wall-mounted panelboards (as in Figure 1.10) that distribute the main power feed through cable trays to the various racks in the datacenter. The cable trays are usually installed over the racks or under a raised floor. Panelboard distribution are custom designed, meaning that most of the wiring work is done on-site so to fit a specific data center with lower cost components that are easy and fast to acquire.

Figure 1.10: Examples of panelboard distribution units [52].

1.2.3.2 Traditional field-wired PDU distribution

Like the case of panelboard distribution units, traditional field-wired PDUs are mainly used for smaller installations or when initial cost is a priority over flexibility and mobility; at the same time, field-wired PDUs enable a higher degree of monitoring options than a panelboard approach. The main power feed is distributed to the PDUs, that are placed throughout the datacenter. Branch circuits from the PDUs are then distributed to the racks, either through rigid conduit below the raised floor or cable trays overhead (as shown in Figure 1.11). Just like the installation of panelboard distribution, field-wired PDU distribution are custom designed to fit a specific data center.

1.2.3.3 Factory-configured PDU distribution

Factory-configured PDU distribution installations are mainly chosen when initial cost is a priority as well as having some flexibility and mobility for easy scaling and changes. In contrast with field-wired distribution strategies, when employing factory-configured distributions most of the installation work is (as the name says) factory-configured. The PDUs are factory-assembled and pre-designed to the datacenters demands. The main power feed is distributed to standardized PDUs that are usually placed right next to the IT racks to bring the distribution closer to the load. The racks are normally placed one after the other, in succession forming a line, with a PDU on one side or either side as seen in (as shown in Figure 1.12).

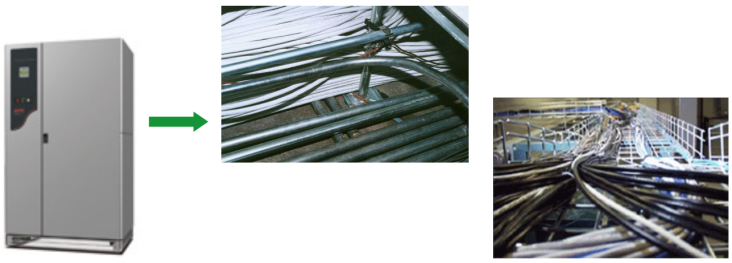

Figure 1.11: Traditional field-wired PDUs with either rigid conduit below the raised floor or cable trays overhead [52].

Figure 1.12: Example of factory-configured PDU placed in a row with the racks [52].

1.2.3.4 Modular power distribution

Modular power distribution is used when flexibility and mobility is prioritized over initial cost, and is usually more manageable and more reliable than the traditional distribution, thou the modular PDUs are build up by factory-assembled modules that can be easily installed without wire work within the data center. The two main modular power distribution systems are: *i)* overhead or underfloor modular distributions, both using plug-in units powered by busways that are placed respectively either overhead or underfloor to feed IT enclosures, and *ii)* floor-mounted modular distributions, which use branch circuit cables and distributed overhead in cable troughs to the IT enclosures. In these systems enclosures are pre-terminated with breaker modules that plug into a finger-safe backplane of a modular PDU as shown in Figure 1.13.

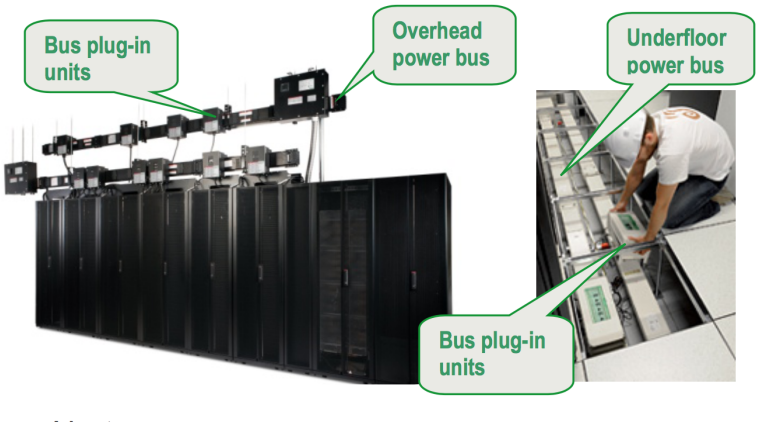

Figure 1.13: Example of a overhead or underfloor modular distribution [52].

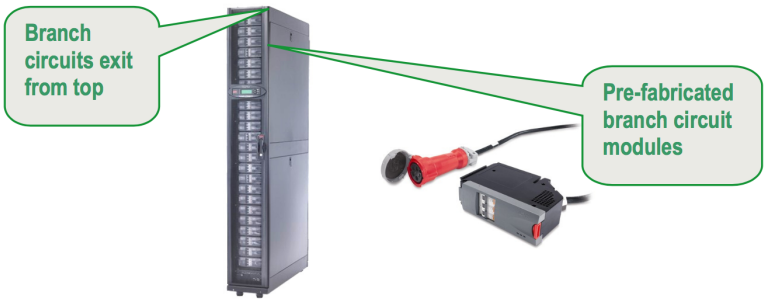

Figure 1.14: Example of a floor-mounted modular distribution [52].

1.2.4 Datacenters as energy consumers

As said in the introduction, data centers consume a great deal of energy and the main contributors are the IT and CT systems, as summarized in Figure 1.15. Data center enterprises are estimated to consume on a global scale around 120GW, which is approximately 2% of the worlds energy consumption (more than the total energy consumption of the entire country of Italy [19]).

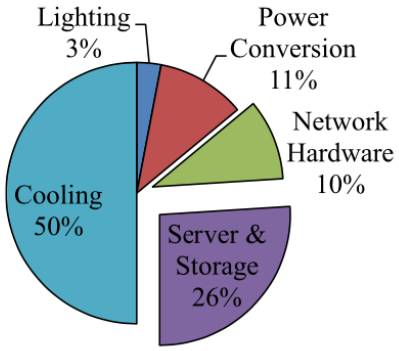

Figure 1.15: Summary of the relative sizes of the energy consumption of the various components of typical data centers [25].

Notice also that with the increase of technology, i.e., social media, the Internet of Things, 4- and 5G, and data-driven medical and biological applications, there is also a rapid growth of the production and consumption of data. According to Cisco Global Cloud Index (CGCI) *Forecastand Methodology, 2015-2020 report* [24] the annual global data center IP traffic will reach 15.3 zettabytes (ZB) in 2020, up from 4.7 ZB in 2015, for a compound annual growth rate (CAGR) of 27 percent from 2015 to 2020 (as shown in Figure 1.16). Global data-center demand will thus continue to increase, with more than 60 new large data centers expected in western Europe by 2020 [54].

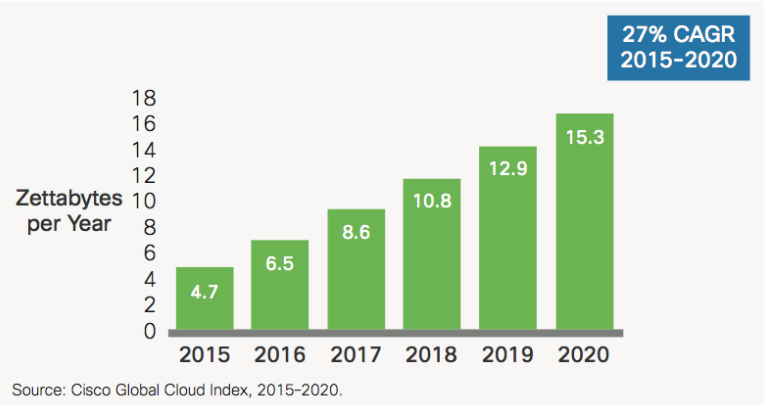

Figure 1.16: Forecasted global data center IP traffic growth [24].

1.2.5 Other ways to reduce the environmental footprint of a data center

There are several other strategies that can be implemented to reduce the environmental footprint of a data center that are outside the scope of the thesis. For completeness, a (non-exhaustive) list of actions that could be taken for improving the energy efficiency is:

- make algorithms faster/needing less CPU power;
- make CPUs more energy efficient;
- implement heat recovery/re-usage strategies (since this indirectly means to waste less energy);
- adopt liquid cooling systems (since these ease the implementation of heat recovery technologies).

1.3 Statement of contributions

In brief, in this thesis we perform three specific tasks:

1. propose an open-source software that allows the real-time monitoring of OCP servers (and provide the code in [14]);
2. propose a control-oriented model of the thermal dynamics of OCP servers and validate it using real data;
3. propose how the software and the thermal model can be used for predictive control purposes.

Notice thus that we take a control-oriented perspective: the monitoring software is specifically made for returning information useful for the online modeling and control of OCP hardware, plus the model of the thermal dynamics is explicitly made for being used in conjunction with control schemes based on Linear Quadratic Regulators (LQRs) or Model Predictive Controls (MPCs).

Notice also that the parameters defining the thermal model, learned using data collected in an experimental data center, are provided so to allow other researchers to both simulate and develop other control schemes.

1.4 Structure of the thesis

Section 2 describes the aim of this master thesis in more details and formulates the problems tackled in the thesis. Section 3 reviews the existing literature on the topic. Section 4 describes the platforms considered in this thesis from a technical perspective. Section 5 describes the software used and the suite developed for the remote monitoring of the considered servers. Section 6 describes the proposed control-oriented thermal model of Open-Compute servers and how to identify this model from real data. The section also describes the model that we identified from the data that was collected in an experimental data center. Section 7 proposes in more details how to use the results provided in this thesis for control purposes. Section 8 summarizes what has been learned from our efforts and describes some future research directions.

Chapter 2

Problem formulation

The aim of this master thesis is to provide the ancillary results that are essential to improve the thermal cooling in data centers, i.e, the CT, in the most energy efficient way, without compromising the Quality of Service (QoS). This energy efficiency can be achieved through control-oriented perspectives using model-based approaches. For example, having an accurate control-oriented model of the server enables using advanced control strategies based on LQRs or MPCs approaches. These strategies, though, require models describing the dynamics of the system. This means that to improve the thermal cooling in data centers there is a need for control-oriented models of the thermal dynamics within the data center.

This thesis deals with the problem of collecting information from the servers and processing this information to obtain these control-oriented models, to be used in the future for operating the data center in an energy-efficient way.

The thesis thus proposes a set of appropriate methods for collecting real-time data from these platforms and a dedicated thermal model describing the thermal dynamics of the Central Processing Units (CPUs) and Random Access Memorys (RAMs) of these servers as a function of both controllable and non-controllable inputs (e.g., the CPU utilization levels and the air mass flow of the server's fans). The thesis also proposes tailored identification algorithms for estimating this model from real data, plus provides the results so to be reusable by other researchers.

To achieve this, the thesis was divided into the following tasks:

- **Monitoring**: To be able to make models describing the dynamics of the system, there is the need for collecting data that capture the behaviour of the system in all its different working conditions. The data to be collected should thus allow the user to make an accurate thermal model of the system. The most important types of data are the following:
  - the temperatures of the CPUs and the RAM banks;
  - the speed of the fans;
  - the power usage of the CPUs and the RAMs.
- **Modelling**: Formulate a simple but sufficiently thermal model of the system that can be used for model-based control.
- **System identification**: Estimate the parameters of the system using the thermal model and the collected real-time data. As mentioned before, there is also a need of running the system in different conditions by stressing the server's CPUs in different ways, so to know how the system behaves at a set of different operation points.
- **Control**: How to utilize the tasks above to maximize the electric efficiency of a data center by optimizing the control using LQRs or MPCs approaches.

Chapter 3

Literature review

3.1 The structure of data centers from control perspectives

From a control perspective, the main aim of controlling a data center is to jointly control the CT together with the IT to improve the total energy efficiency of the datacenter while maintaining a specified QoS. To achieve this goal, it is beneficial to inspect the structure of data centers from control perspectives.

As mentioned before in Section 1.2, a data center can be structured into three main infrastructures, i.e., the IT system, the CT system and the PD system. Looking at the system from control organizational standpoints, the operations of a data center can be structured using a hierarchy of three different types of layers:

- Server level, described in more details in Section 3.1.1
- Rack level, described in more details in Section 3.1.2
- Data center level, described in more details in Section 3.1.3

For completeness, these layers are shown in Figure 3.1.

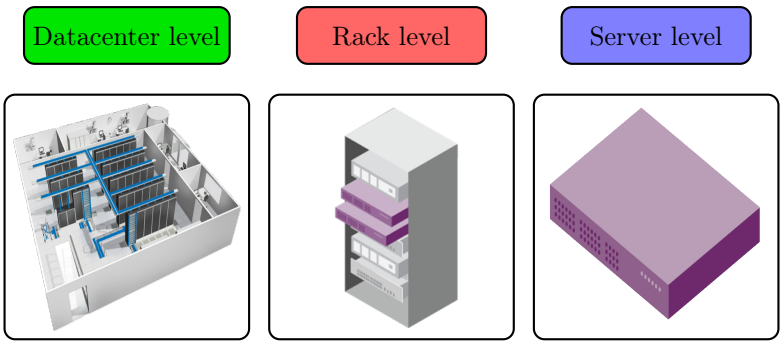

Figure 3.1: Datacenters main control organization levels: the data center level, the rack level and the server level.

3.1.1 Server level

A single server can be schematized as in Figure 3.2.

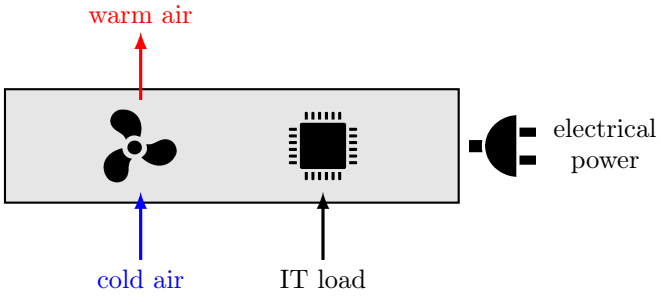

Figure 3.2: Scheme of a single server when considering it from control perspectives.

When looking at a single server from a control perspective, the main strategies through which one can improve the energy efficiency of the equipment while maintaining a specified QoS are:

1. Reduce its energy usage by adaptively turning off those components that are idle (e.g., the network devices or even whole servers) [32, 58];
2. Reduce the peak power consumption of CPUs leveraging predictive Dynamic Voltage and Frequency Scaling (DVFS) techniques (e.g., by modulating the clock frequency based on the current demands) [38, 51];
3. Optimize the flow of the coolant through the enclosure of the server so as to satisfy to given temperature constraints on the internal components while minimizing the electrical power that is spent to produce this flow (e.g., the power spent to run the fans in air-cooled servers).

Regarding the last technique, one may address the problem of controlling just the local fans of the server under consideration and consider the temperature of the inlet coolant as given [36], jointly control local fans with the infrastructural ones in free air cooling data centers [22, 39, 40], or connect the fans control problem with the one of forecasting and allocating the IT loads opportunely through predictive control strategies [45]. These approaches exploit model-based control strategies to achieve energy efficiency. Notice, however, that the thermal models considered in the papers mentioned above are often *general-purpose* and do not consider the detailed thermal structure of *specific* server platforms.

3.1.2 Rack level

A data center rack can be schematized as in Figure 3.3.

When looking at a whole rack from a control perspective, the main strategies through which one can improve the energy efficiency of the equipment while maintaining a specified QoS are:

1. Maximize the usage efficiency and minimize the energy usage by optimally schedule tasks, making active servers run at 100% and enabling ideal servers to be turned off or entering a low-power mode. A problem of this approach is often the setup time, i.e., the time required to turn a server back on and off. The two main control techniques studied in this matter is the migration of Virtual Machines (VMs) [31, 33, 37, 53, 56, 57] and optimal scheduling of the setup time [27–29, 50];

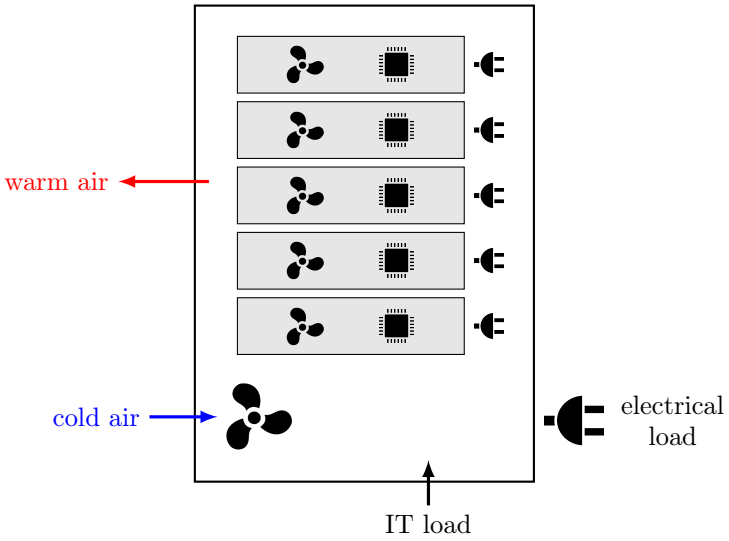

Figure 3.3: Scheme of a single rack when considering it from control perspectives.

2. Thermal aware workload placement scheduling (e.g., task scheduling that consider the thermal properties [20] or prediction model based thermal aware scheduling [35])

3. Reduce the energy consumption by implementing IT loads assignment (e.g., dynamically adjusts the IT resources, using an adaptive control system, between servers that are divided among different tiers [41]). This requires to dynamically schedule the IT loads of several servers in concert to balance the computational load across the data center's space or even multiple data centers [43];

4. Power and cooling management of server rack cabinets by implementing control strategies for the fans of the rack (e.g., a model-based systems approach that combines fan power management with conventional server power optimizations [55]).

3.1.3 Data center level

A whole data center can be schematized as in Figure 3.3.

At the broadest control perspective, controlling the whole datacenter to improve its energy efficiency is often done by splitting the data center in zones, applying different control techniques on these zones, and combining the control actions so to make the different zones act synergically. This leads to the following list of techniques:

1. Implementation of hierarchical/distributed based control strategies, i.e., controllers with specific assignments reflecting their respective level in a hierarchical fashion (e.g., time based hierarchical levels where fast dynamics of the IT system are managed at the lower levels and slower thermal dynamics of the CT system at the higher levels [44, 46, 47, 49]);

2. Energy efficient control strategies on a smart grid (e.g., interacting with the power-grid by taking advantage of time-varying electricity prices, renewable energy usage and reliable energy [48];

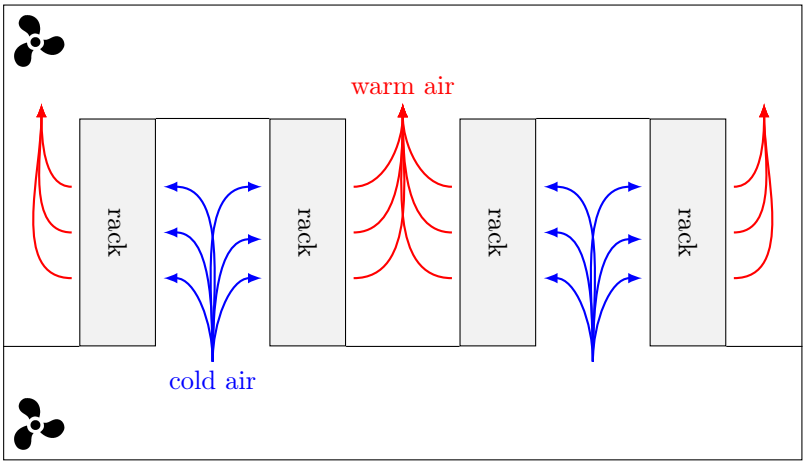

Figure 3.4: Scheme of a whole data center when considering it from control perspectives.

3. Installation of various CRAC control systems (e.g., Control adjustable CRACs that change and redirect the supply air flow and temperature depending on the condition throughout the datacenter [21]).

Notice that making a data center entirely monitorable and controllable enables better decisions and synergies. Moreover, summarizing the papers mentioned above, there exist strong IT-CT couplings, so that operating IT and CT separately is suboptimal. In other words, both *how* and *where* the workloads are executed affect the total energy consumption: indeed the way workload is executed impacts the efficiency at which IT operates, while the location where the workload is executed impacts the efficiency at which CT operates.

Often data centers are modular, and this suggests (as done in the previously mentioned papers) the implementation of hierarchical/distributed estimation and control strategies. Nonetheless this is problematic because cooling resources are usually shared, so distributed controllers need data from a whole-data center-level point of view. At the same time, data centers are also large-scale systems: the number of state variables ranges in the order of ten of thousands for a medium-size data center, since it is useful to let each CPU temperature, workload, and amount of resources used be part of the state of the system. This means that there is an intrinsic trade-off between implementing centralized or distributed control strategies.

Also, notice that every different data center has very specific thermal dynamics, and there is no general whole data center model. This complicates the development of general-purpose control strategies.

Finally, another problem is that there is no algorithm that accurately models the air flows and that is fast enough to be used for real-time control.

3.2 Monitoring for maintenance and alerts

Since 2010 the need for reliable Data center infrastructure management (DCIM) systems has continuously increased, with the expansion of datacenters demands. There are almost an endless supply of different DCIM vendors to choose from. A selective few of the main ones can be summarized as follows [7]:

- ABB Ability™;

- Nlyte Software;
- Emerson Network Power;
- Schneider Electric.

Notice that all the vendors above are famous businesses with all-included solutions. However there are also many smaller vendors with open source solutions like *Zabbix*, *Opendcim* and *Device 42*.

We also notice that in many datacenters the facility and IT are operated separately in different teams: for example the building automation system, the power supply control system and the cooling automation system are typically managed separately, often also by the different vendors who delivered the systems [3]. Doing holistic monitoring as in Figure 3.5 would instead allow for detecting more faults and managing the whole structure in a more holistic way. The problem is then that this type of monitoring requires also more complex and holistic models, and this raises the issue of how to connect the various models of the most important components into a unique model.

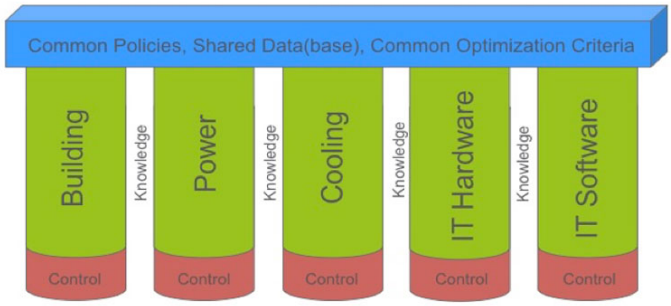

Figure 3.5: A holistic approach on automation in datacenters [3].

Chapter 4

The Platforms

4.1 The Open Compute Platform

The Open Compute Project initiative was started by Facebook in 2011, and supports the development and sharing of data center designs [13]. The project has fostered an open engineering community that aims at increasing several performance indicators of data centers and prominently their efficiency. Major data center operators (e.g., Nokia, Intel, Microsoft) collaborate within the project on many different parts composing data centers (e.g., networking infrastructure, racks, power units, etc.).

4.2 Facebook's Windmill V2 server blade

In this thesis the main focus was on a Open Compute Server (OCS) platform based server, more specifically on the OCS platform *Facebook's Server V2 Windmill* (shown in Figure 4.1 and henceforth generically referred to as the OCS blade). This server is equipped with two Intel Xeon E5-2670 CPUs and 16GiB of memory partitioned in 8 Dual In-line Memory Modules (DIMMs) banks. In the rear of the server two system fans are installed to push out the hot air and to create a air flux through the server. The specifications and schematics of the OCS blade can be seen in Table4.1 and Figure 4.1. Notice that our blade were only equip whit 4 DIMMs per socket and not the total 8. A real photo of our blade can also be seen in Figure 4.1.

The OCS blade comes from an Open Rack V1 system. The system is a custom rack that houses Open Compute Project server technologies. The Open Rack V1 system uses an all-encompassing design to accommodate compatible Open Compute Project chassis components, which include the power solution as well as input and output voltage distribution.

4.3 The Dell PowerEdge R730 server blade

As previous stated the main focus of this thesis was on the OCS blade, but because of access limitation to the OCS blade in the beginning of the thesis, a Dell server was also used. The server in question was the Dell platform *PowerEdge R730xd* (shown in Figure 4.2 and henceforth generically referred to as the Dell blade). The server is equipped with two Intel Xeon E5-2620 CPUs and 251GB of memory partitioned in 16 DIMMs banks. The specifications of the Dell blade can be seen in Table4.2. The Dell server is equipped with six system fans to cool the server

Platform	Intel Xeon Processor E5-2600 product family platform
CPUs	2 Intel Xeon Processor E5-2670
Threads	2 threads per core
Cores	8 cores per socket
DIMM Slots	Up to 16 total DIMM slots Up to 8 DIMMs per CPU Up to 2 DIMMs per channel 2 system fans

Table 4.1: Specifications of the OCS Facebook blade Version 2.0 Windmill

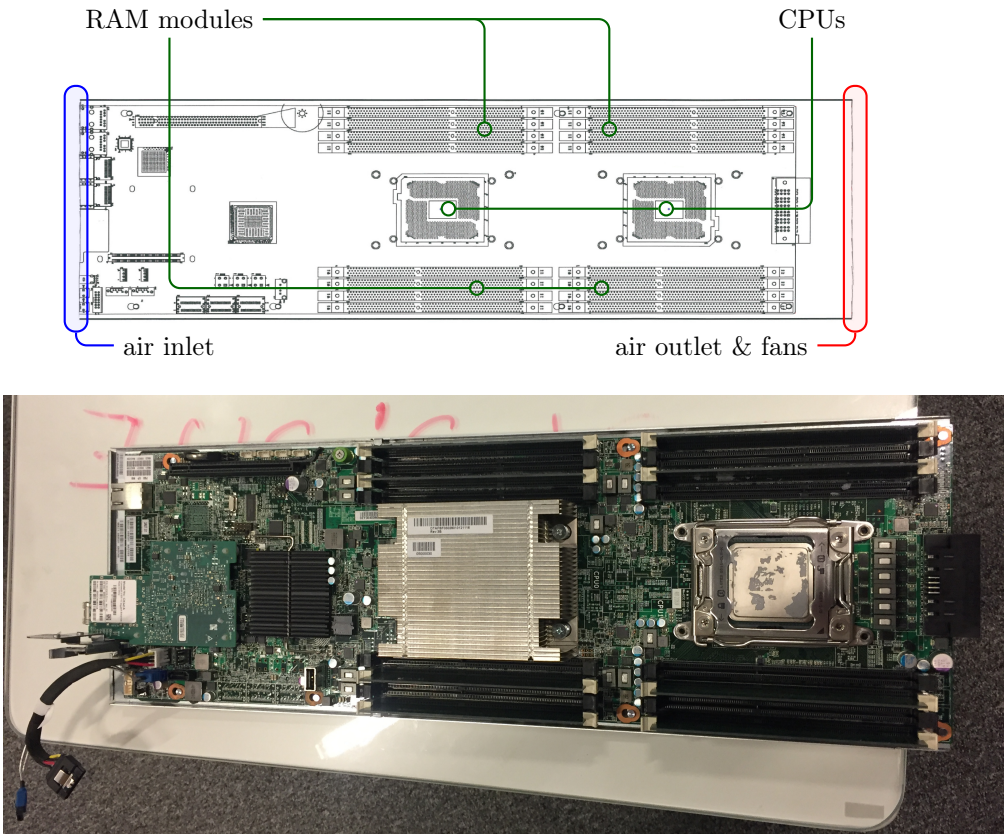

Figure 4.1: Blueprint of the OCS Facebook server V2.0 Windmill (above) and photo of one of the servers used in our experiments (below). Notice that the picture shows a server from which all the hard disks and one of the heat dissipators of the CPUs have been removed.

components. Notice that our blade were only equip whit 8 DIMMs per socket and not the total 12, hence partitioned in 16 DIMMs banks.

Platform	Intel Xeon Processor E5-2600 product family platform
CPUs	2 Intel Xeon CPU E5-2620 series
Threads	2 threads per core
Cores	6 cores per socket
DIMM Slots	Up to 24 total DIMM slots Up to 12 DIMMs per CPU Up to 2 DIMMs per channel 6 system fans

Table 4.2: Specifications of the Dell PowerEdge R730xd blade

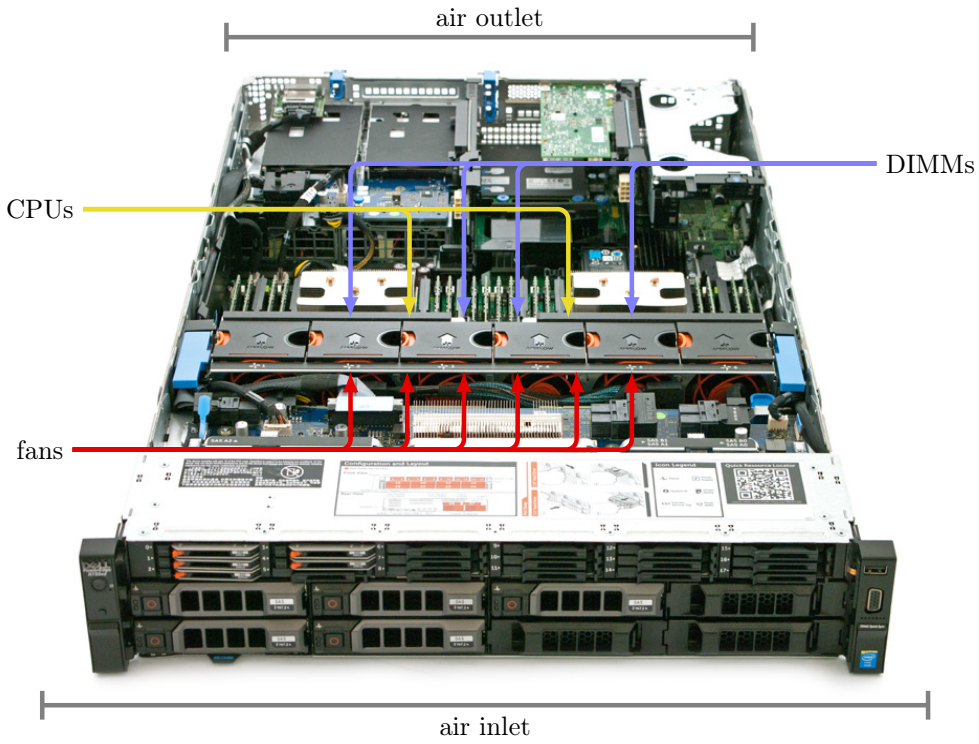

Figure 4.2: The Dell blade layout inside SICS ICE datacenter [4].

4.4 Limits and constraints

As for the structural limits of the considered platforms, we notice that:

- the rotational speeds of the fans are limited, so that the control values u are constrained in the hyperrectangle defined by the extreme points $u_{\min}, u_{\max}$;
- the temperatures of the IT components shall be kept below some specified safe limits, that implies that there exist state constraints of the kind $x^c \preceq x^c_{\max}$;
- the sensors in the two servers used, i.e., the OCS blade and the Dell blade, have quite limited resolution and sensitivity for the purpose of this thesis. More accurate thermal information would help to a more superior estimation of the servers unknown parameters.

Chapter 5

Monitoring Dataservers

This section describes the approach that we developed for the remote monitoring of the thermal status of a generic OCP server remotely through Python scripting. Our aim is indeed to be able to gather information remotely on the thermal state of a server while it is operating (instrumental for both monitoring its performance and controlling the associated air outlet fans) by interfacing with the firmware of the server and its operating system.

More specifically the developed software suite gathers information on the fans, on the CPUs and on the RAM modules. Since the contributions of the other IT components to the thermal properties of this type of servers are negligible, we indeed neglected them. In practice, thus, the software monitors:

- the temperatures of the CPUs and the RAM banks;
- the speed of the fans, since these affect the convective thermal exchanges of the heat sinks of the components;
- the CPUs computational loads as an approximate indication of their power usage.

Notice that one would be interested also in measuring the power usage of the RAM modules; Doing this monitoring by software, using e.g. performance counters, is however a difficult task, given the volatility of these memories. This task is still currently subject to on-going research. Here, instead, we approximate the average power consumption of the RAM banks by the instantaneous amount of resident memory. This ansatz is further discuss when presenting the validation results in Section 6.7.

The developed remote monitoring suite has been developed laddering on two existing open technologies, i.e., Intelligent Platform Management Interface (IPMI) (discussed in Section 5.2) and Simple Network Management Protocol (SNMP) (discussed in Section 5.1). The structure of the developed suite is instead discussed in Section 5.3.

5.1 Simple Network Management Protocol (SNMP)

SNMP [16] is an application layer protocol built on the Transmission Control Protocol - Internet Protocol (TCP-IP) suite, and is used for monitoring and managing network devices such as routers, switches, servers, etc. The SNMP network is build up by a managing computer (manager Section 5.1.1) that queries information from the network devices through a deamon (agent Section 5.1.2) installed on the network device.

5.1.1 Manager

The manager or managers are the main computers that send request to the agents in the network. This computers usually runs some kind of managing software (Network management station (NMS)). We used the standard NMS called Net-SNMP Command Line Applications provided in the Net-SNMP suite [11]. This make it possible to send specific SNMP commands directly from the command line on the manager.

5.1.2 Agent

The agent is a managing deamon installed on the network device that is being monitored. It is to the agent that the manager sends its requests to queries information about the local environment of the network device. The agent used is the standard deamon in the Net-SNMP suite. The agent stores the information about the local environment of the network device on a shared database (Management Information Base (MIB)). This database is shared between the agent and the Manager.

5.1.3 Management Information Base (MIB)

MIB is a database used for managing the entities in a communication network. Most often associated with the SNMP, the term is also used more generically in contexts such as in OSI/ISO Network management model. While intended to refer to the complete collection of management information available on an entity, it is often used to refer to a particular subset, more correctly referred to as MIB-module.

5.2 The Intelligent Platform Management Interface (IPMI) specifications

IPMI [10] is a communication protocol broadly used for monitoring and management assignments led and provided by Intel. More specifically IPMI is a collection of network interface specifications that defines how a Baseboard Management Controller (BMC) (an embedded micro-controller located on the motherboard of the server dedicated to handle all the IPMI communications) can exchange local information over the network.

Interestingly there exists a plethora of different user-space implementations of the IPMI protocol; in our case we exploited *FreeIPMI* [9], a set of IPMI utilities and libraries that provides a higher-level IPMI language so to simplify its usage. Moreover, to work properly, IPMI requires opportune drivers and managing tools to be operating in the server. The developed software suite has been tested using the default Linux driver *OpenIPMI* [15], but has been built so to be compliant with standard IPMI commands. Notice also that *FreeIPMI* has been chosen over other alternatives (such as, e.g., *IPMItool* 5.2.2) since it allows higher rate of completed IPMI queries per second, something that allows to better capture transient thermal behaviours and thus to obtain better thermal models.

5.2.1 OpenIPMI

OpenIPMI is the most know Linux driver for IPMI, that both contains a full-function IPMI device driver and a library that supplies a higer-level output, that make easier human understanding.

5.2.2 IPMItool

IPMItool is on of many utilities for managing IPMI on capable devices. The tool have a command-line based interface and can interact both locally and remotely over LAN.

5.2.3 FreeIPMI

FreeIPMI provides in-band and out-of-band IPMI software based on the IPMI v1.5/2.0 specification. The IPMI specification defines a set of interfaces for platform management and is implemented by a number vendors for system management. The features of IPMI that most users will be interested in are sensor monitoring, system event monitoring, power control, and serial-over-LAN (SOL).

5.3 *CISSI*: A control and system identification framework aimed at server blades

The aim is to help achieve energy savings in data centers through better thermal control schemes. The first contribution is thus a software suite called *CISSI* that allows to easily and reliably collect information from OCP server to be used either offline (for data-driven thermal modelling) or online (for computing feedback signals in control algorithms).

Since the IPMI and SNMP tools described above are more general-purpose software, the choice has been to develop a dedicated tool. The provided *CISSI* suite implements thus three main tasks:

- monitoring and fetching (potentially also remotely) thermal information from OCP servers;
- stressing the server's CPUs (for experiment design purposes);
- actuating the server's fans (for controlling purposes, so to ease the prototyping and testing of different thermal control schemes).

From logical standpoints the suite is composed by three modules, named *fetch*, *stress*, and *control*, interacting among them and with the hardware of the server as described in Figure 5.1.

In more details,

the *fetch* module is a Python script collecting information from the native sensors in an OCP server using the tools *ipmi-sensors* and *snmpwalk*. The collected data comprises information on the fan speeds, the CPUs temperatures, the inlet and outlet air temperatures, the temperatures of the RAM modules, and the CPUs computational loads. The data is then available to both socket communications or (as shown in Figure 5.1) for saving in an external database. The code provided in [14] currently allows to save NumPy `.npz` or Matlab `.mat` files;

the *stress* module is a Python script that randomly changes the loads of the CPUs with an either predefined or random sampling period that uses the Linux-based stress test `stress-ng` [17];

the *control* module is a Python script sends Pulse-Width Modulation (PWM) signals to the fans over the network through the tool *Ipmi-raw* from the suite *FreeIPMI*. Notice that this control strategy requires using the OCP Facebook Server Fan Speed Control Interface (FSC) tools [8], and that –despite having been implemented in Python– it can actually

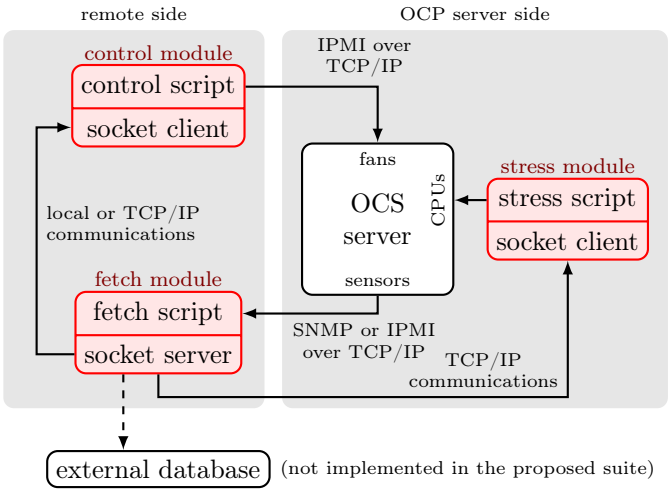

Figure 5.1: Overview of the logical structure of the *CISSI* suite. The *fetch* module, potentially residing on an other computer, fetches the raw data from the sensors, processes it, and propagates the information to who requests it through an opportune socket (also potentially to other external databases). The *control* module, that also may reside remotely, gets the processed information and uses it to compute a control action for the fans that is then communicated back to the server through opportune IPMI commands. The *stress* module, residing instead locally in the inspected OCP server, directly affects the CPUs loads through opportune Python scripts.

be implemented in other languages by exploiting the provided Application Programming Interface (API) of the provided *fetch* module as soon as one connects to its socket stream.

CISSI is a software that is developed to aid the aim of energy saving in data centres. There are two main requirements on the design of *CISSI*:

- be able to collect the necessary information to do a thermal model of the OCS blade from a system identification standpoint.
- enable a continuous input stream that should be used as feedback in a predictive control purpose.

CISSI is a framework that replies at the previous requirements by implementing three main tasks:

- Monitoring and fetching thermal information of server blades.
- Stressing the CPUs of server blades.
- Act as intercommunication between server fans and actuator.

The first two tasks is the ones that allows system identification of temperature dynamics models of the server blades. The third task for the framework is to act as an intercommunication between server fans and actuator, which allows a real-life control system where the server fans are controlled in a feedback loop as shown in Figure 5.1.

CISSI is composed of three main parts as can be see in Figure 5.1:

- **fetch**: Data acquisition module, analyzed in 5.3.1
- **stress**: CPU stressing module, analyzed in 5.3.2
- **control**: Server control loop module, analyzed in 5.3.3

This modules are designed to work in two main modes:

- Data gathering mode: This mode aims to solve the first two main tasks to allows system identification of temperature dynamics models of the server blades. This mode runs both the *fetch* and the *stress* module at the same time. Were the *stress* module randomly changes the loads of the CPUs, while the *stress* module synchronously gathers the thermal information of the server.
- On-line control mode: This mode aims to solve the third main task to act as an intercommunication between server fans and actuator.

5.3.1 Data acquisition module

It is a script written in python, designed to collects thermal information from the server such as:

- Fan speeds
- CPU temperature
- Inlet- and exhaust temperature
- DIMM temperature
- CPUs computational loads

It does so using two standard interfaces IPMI and SNMP that are discussed in Section5.2 and Section5.1. It then streams the data to a client connected to the server socket as shown in Figure 5.1. The software can also, instead of continuously stream the data to a socket, save the data in NumPy .npz file or a MATLAB .mat file. This data can later be used as the input in a system identification perspective. The main loop of the module rendered in pseudo-code can be seen in Algorithm 1.

In brief, the desired execution time t and the sampling period T are set and also the desire to use IPMI or SNMP. Both interface can set if wanted. The module fetch the server information and posts the data to a socket. The queries are done using the tools `ipmi-sensors` and `snmpwalk`.

The API of the data sharing mechanism over the socket is to simply send tagged data. Where variable descriptors and values are comma separated. For example if the two variables a,b with values 3,4 were obtained by `snmpwalk` and then send the socket client receives "SNMP:a,3,b,4".

5.3.2 CPU stressing module

It is a script written in python designed to randomly stress the CPUs of a server blade, with a set sampling period, and then collect and saves the sequence. It does so by using the Linux based stress test `stress-ng`.

The main loop of the module rendered in pseudo-code in Algorithm 2.

In brief, the desired execution time t and the sampling period T are set. The script then randomly stresses the CPUs with `stress-ng` and saves the stressing schedule.

Algorithm 1 Data acquisition component of *CISSI*

Input: The execution time t and the sampling period T, use_snmp, use_ipmi, "output file"
Output: SNMP and/or IPMI query via Socket or output file, i.e CPU temperature, Fan speed, Inlet- and Exhaust temperature, etc.

```
Connecting to socket client
while cur_time < t do
    for t = 0, T_1, T_2, T_3, ... do
        if use_snmp == True then                    ▷ SNMP triggerd in input
            X_j(t) ← SNMP query
        if use_ipmi == True then                    ▷ IPMI triggerd in input
            X_i(t) ← IPMI query
        Send X_j(t), X_i(t) to socket client
if "output_file" then
    Save output_file.m ← X_j(t), X_i(t)
```

Algorithm 2 CPU stressing component of *CISSI*

Input: The execution time t and the sampling period T, "output file"
Output: CPU socket loads

```
Read topology of server, i.e. number of sockets and associated cores per socket
Create pseudo random binary signal
while cur_time < t do
    for t = 0, T_1, T_2, T_3, ... do
        for socketid in nr_sockets do
            if cur_load ≠ desired_load then
                socketid ← desired_load
if "output_file" then
    Save output_file.m ← CPUsocketloads
```

5.3.3 Server control module

The primary task of this module is to be able to control the fans of a server. This can be done in two different ways, namely:

- Software based, i.e., by sending control signals over the network;
- Hardware based, i.e., by using a dedicated fan controller.

The strategy implemented in this thesis is a software based one. More precisely the chosen strategy is to use the `Ipmi-raw` tool from `FreeIPMI` to communicate with the FSC tools [8] of an OCP Facebook Server; the FSC software then instructs the BMC of the OCP server to send an appropriate PWM signal to the fans (in practice, thus, commanding their rotational speed).

The server control module is thus a Python script that sends the desired PWM control signals to the fans over the network through *Ipmi-raw* to first FSC and then to the BMC. Despite having been implemented in Python, it can actually be implemented in other languages by exploiting the provided API of the provided *fetch* module as soon as one connects to its socket stream. In brief, the module would connect to the socket stream from the fetch module 5.3.1. The module would then compute the control action according to the model and post the control action to the server fans via the FSC.

Remark 1 The server control module were not entirely completed due to lack of support of the FSC on the used OCS blade.

5.3.4 Thermal information collected from *CISSI*

In this section the thermal information collected from *CISSI*, when in data gathering mode, are analyzed in more details. All of the figures below are collected during the same run of the data gathering mode. In Figure 5.2 it is shown that the two fans, which are controlled by the native control systems in the data center, seem to act synchronously, i.e, both the fan speeds are affected similarly by the stressing of both the CPUs (compare this to the dynamics of the temperatures of the two CPUs, shown in Figure 5.3). Comparing the dynamics of the temperature of the CPUs and the DIMMs, plotted in Figures 5.3 and 5.4 respectively, we also notice that the temperatures of the DIMMs are not as directly affected by the *stress* module as the temperatures of the CPUs. A direct reason for this is that the *stress* module built in this thesis focuses on stressing the CPUs, not the DIMMs. A more direct stressing of the DIMMs could have been implemented in the module, but this was however not prioritised and then completed for lack of time (and is thus kept as a future work). Figure 5.5 shows the inlet- and exhaust temperature of the server. The inlet temperature was controlled by acting on the CRACs of the datacenter by the native control systems and was, as seen in the figure, almost entirely constant during the test period. Through the figures below it can be seen that the exhaust temperature is instead affected by a combination of all the temperature changes in the server. Figure 5.6 finally shows the computational load of the two CPUs in the OCP server during the experiment.

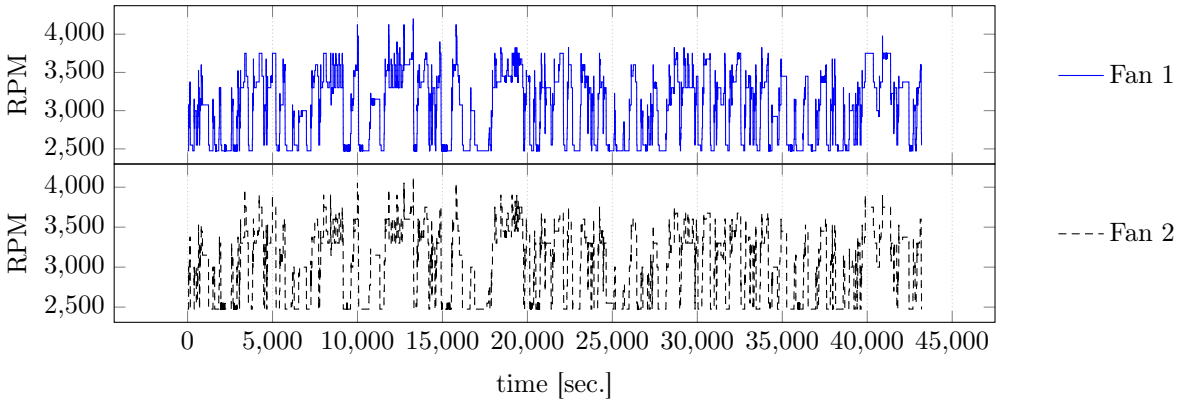

Figure 5.2: The fan speeds of the two fans in the OCP server during our experiment.

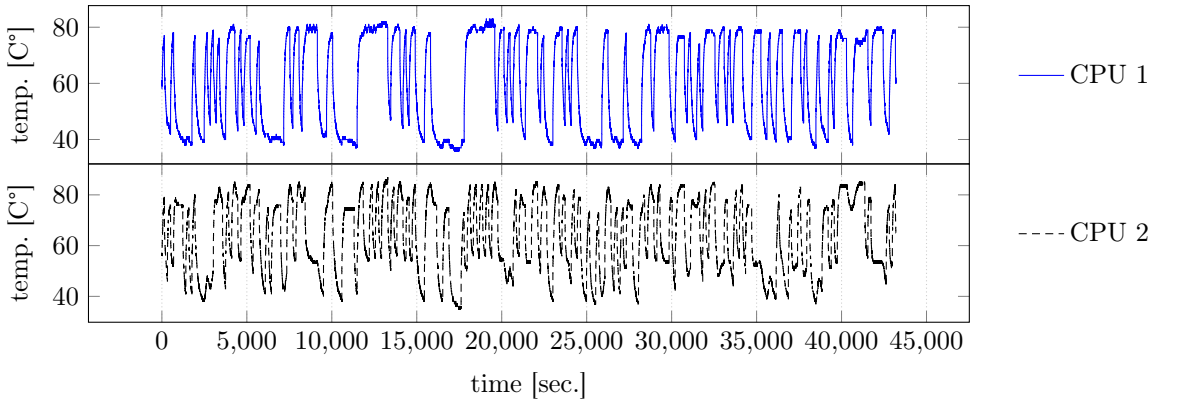

Figure 5.3: The temperature of the two CPUs in the OCP server during our experiment.

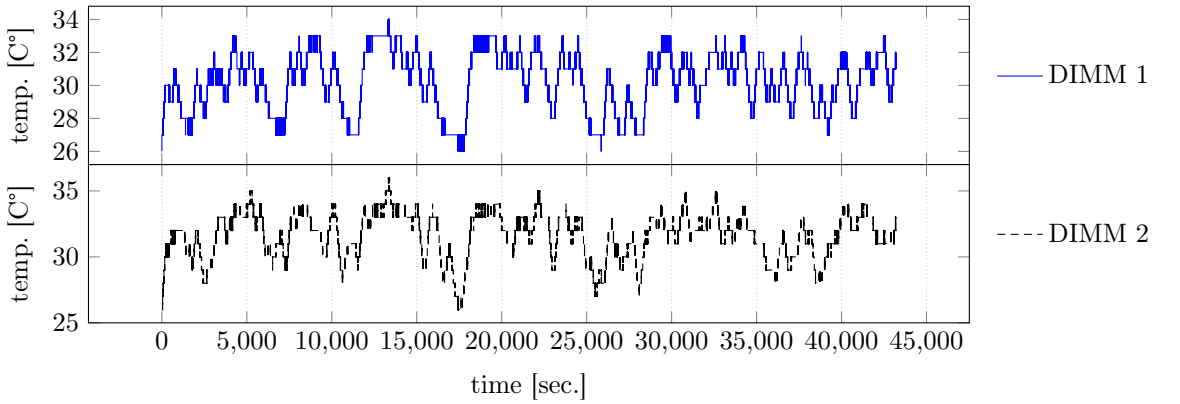

Figure 5.4: The temperature of the two DIMMs in the OCP server during our experiment.

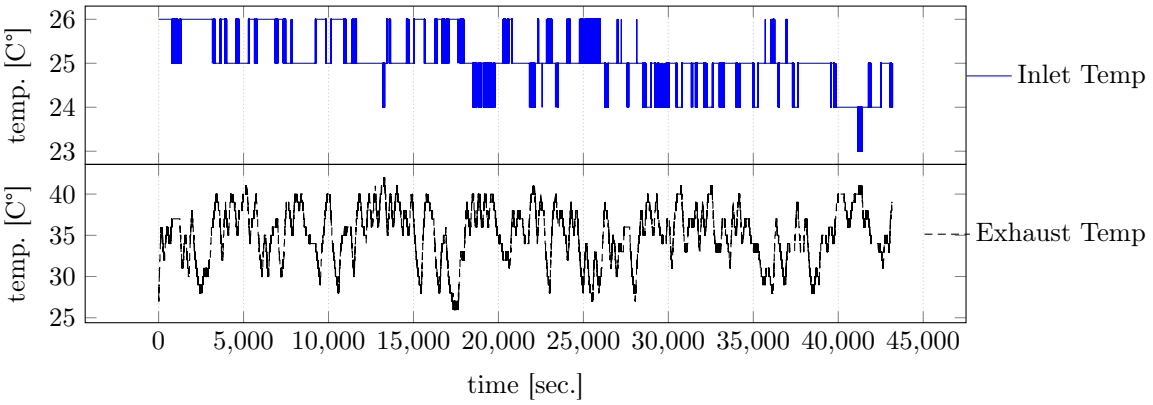

Figure 5.5: The inlet- and the exhaust temperature of the OCP server during our experiment.

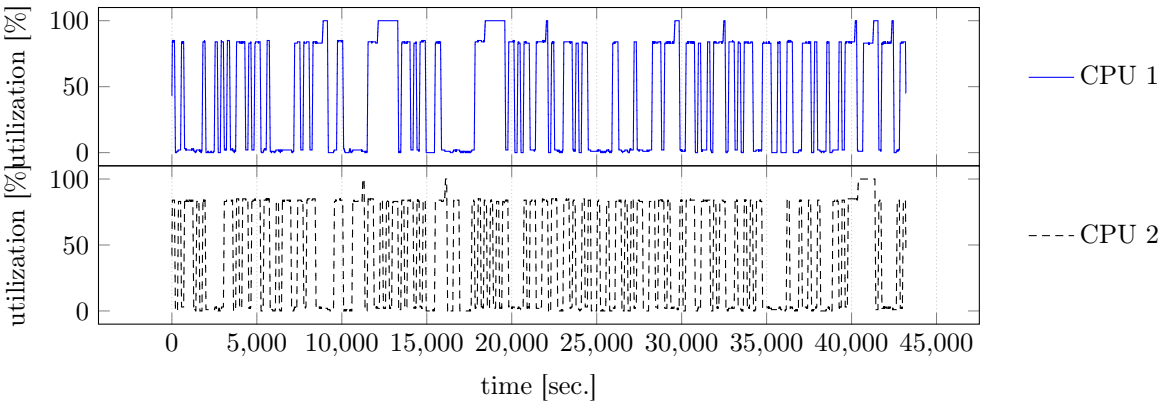

Figure 5.6: The computational load of the two CPUs in the OCP server during our experiment.

Chapter 6

Thermodynamical modelling

6.1 Modelling open compute servers

The goal is to derive a discrete time control-oriented thermal model that captures the thermal dynamics within the server enclosure with a formalism that can be used for designing online thermal control strategies. We thus disregard modelling the statistical properties of IT loads, and focus instead on the most important components of the system from a thermodynamics perspective, i.e., the CPUs, the RAMs, and the fans forcing air flows within the enclosure.

We thus divide the thermal model as follows:

- as for the *air flow model*, we consider a static linear model that accounts for potential mixing effects among the individual fluxes imposed by the fans;
- as for the *temperatures of each single electronic component*, we assume that the dynamics follow a first order model.

Moreover, given the structure of the enclosure of the server, we assume that there are no air recirculation effects, so that the thermal network can be schematized as in Figure 6.1.

Referring to Figure 6.1, for convenience we divide the server in three zones numbered consequentially while traveling from the air inlet to the outlet and indicated with the index $i = 1, 2, 3$. The first zone comprises thus the first CPU and the first two DIMM modules; the second zone comprises the remaining CPU and DIMM modules; the third zone instead goes from the end of these components to the server's fans.

Importantly, we assume that the air flows within the enclosure of the server are static functions of the flows induced by the fans. We motivate this simplification through considerations on the time scales of the various dynamics: in other words, *i)* the dynamics of the air flow is much faster than the dynamics of the temperatures of the electronic components; *ii)* given these fast dynamics and the fact that our goal is to control the speed of the fans so to cool the components, variations in the flow due to its turbulent behavior are absorbed by considering only its average behaviour [34, Chapp. 10 and 11]. In the following subsections we will thus first list the variables involved in our model, then detail the models of the air flows and of the temperatures of the components independently, and then combine these sub-models and the notation into a unique overarching model.

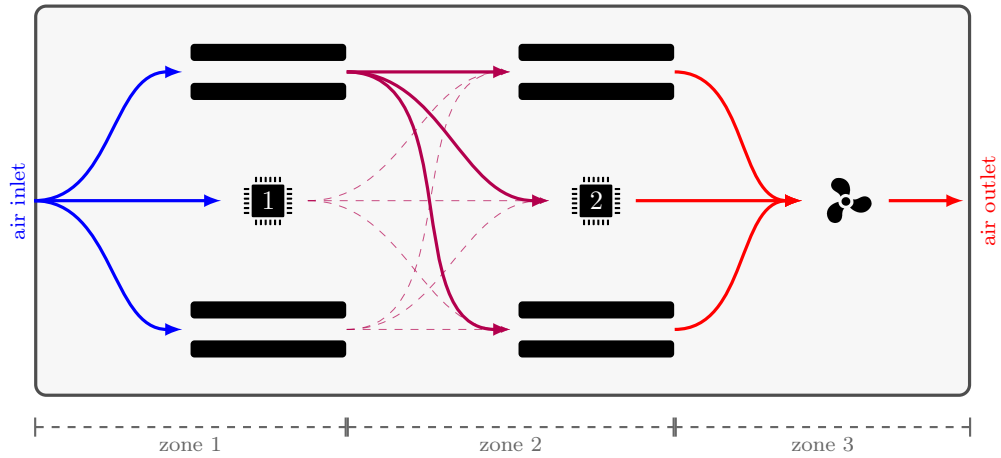

Figure 6.1: Graphical representation of the thermal network corresponding to the OCP Facebook Server V2.0 Windmill. The arrows indicate the modelled air fluxes among the various IT components (with some fluxes from the first to the second column of components dashed for graphical clarity).

6.2 Notation

We thus consider the following variables:

- as *states*, the temperatures of the heat-dissipating IT components (x^c_{ij} in Figure 6.2), the temperatures of the various air flows *just before hitting the relative target components* (x^f_{ij} in Figure 6.2), and the temperature of the air at the outlet (x^{out} in Figure 6.2). Notice that the temperatures of the air flows in zone 1 are equal to the temperature of the air inlet, and that the temperature of the air flow in zone 3 is equal to the temperature of the air outlet;

- as *exogenous inputs*, the average electrical power dissipated by each IT component (p_{ij} in Figure 6.2) and the temperature of the air inlet (x^{in} in Figure 6.2;

- as *controllable inputs*, the total air mass flow produced by the fans at the outlet of the server (u in Figure 6.2), assumed to be equal also to the total air mass flow at the air inlet.

6.3 Modelling the air flows

To model the air flows within the enclosure of the server we use a set of auxiliary variables that will not be present in the final thermal model but that are useful to construct it. More specifically these variables ideally capture *the intensities of the air flows among the various components within the server*. The notation used is $f_{i,j\to k}$ with i indicating the zone of the server and j and k respectively the source and the destination (e.g., $f_{1,\text{in}\to 2}$ indicates the intensity of the flow from the inlet to the first CPU, $f_{2,1\to 3}$ indicates the intensity of the flow from the top DIMM in zone 1 to the bottom DIMM in zone 2, and $f_{3,3\to 2}$ indicates the intensity of the flow from the bottom DIMM in zone 2 to the fans in zone 3).

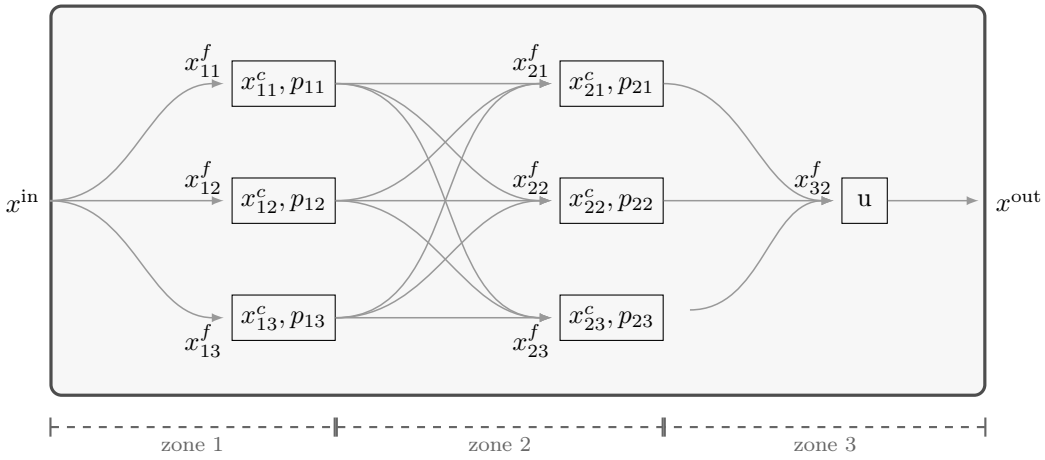

Figure 6.2: Graphical summary of the notation used to model the thermal dynamics of the OCP Facebook Server V2.0 Windmill. Notice that $x^f_{11} = x^f_{12} = x^f_{13} = x^{\text{in}}$ and that $x^f_{32} = x^{\text{out}}$.

Considering the previously posed assumption that the flow in and the flow out have equal intensity and formulating a model that guarantees the physical prior of *mass conservation* leads then to the linear models for zone 1 and 3

$$f_{1,\text{in}\to j} = \lambda_{1,\text{in}\to j} u, \qquad \sum_{j=1}^{3} \lambda_{1,\text{in}\to j} = 1, \tag{6.1}$$

$$f_{3,j\to\text{out}} = \lambda_{1,j\to\text{out}} u, \qquad \sum_{j=1}^{3} \lambda_{1,j\to\text{out}} = 1. \tag{6.2}$$

As for zone 2, notice that it is convenient to add the further auxiliary variables $f_{2,j}$, $j = 1, 2, 3$, ideally capturing the intensity of the flow hitting component j, i.e.,

$$f_{2,j} = \sum_{k=1}^{3} \lambda_{2,k\to j} f_{1,\text{in}\to k}. \tag{6.3}$$

Notice that our assumption of conservation of mass imposes then the constraint

$$\sum_{j=1}^{3} \sum_{k=1}^{3} \lambda_{2,k\to j} = 1. \tag{6.4}$$

Notice also that the various $\lambda_\star$ are non-negative constants. Moreover, assuming that the flows do not mix while passing through the various components, it holds that

$$f_{3,j\to\text{out}} = f_{2,j}. \tag{6.5}$$

As for modelling the temperatures of the various air flows, we approximate the temperature of the flow just before each IT component (x^f_{ij} in Figure 6.2) as the weighted average of the temperatures of the incident flows under the simplifying assumptions of perfect flow mixing and heat energy conservation. Since we also ignore recirculation effects, this implies that:

- as for the first zone, the temperatures of the flows (just before hitting the first components) is x^{in};
- as for the second zone, the temperatures of the flows (just before hitting the relative components) is

$$x_{2j}^f = \frac{\sum_{k=1}^3 \lambda_{2,k\to j} f_{1,\text{in}\to k}\Big(x^{\text{in}} + \frac{h_{1k}}{c_p}\left(x_{1k}^c - x^{\text{in}}\right)\Big)}{f_{2,j}} \tag{6.6}$$

  where c_p is the heat capacity of air at the constant pressure of 1 atmosphere, the quantity $x^{\text{in}} + \frac{h_{1k}}{c_p}\left(x_{1k}^c - x^{\text{in}}\right)$ stands for the temperature of the air flow *just after* the k-th component in the first zone, and where the multiplication of each term by $\lambda_{2,k\to j} f_{1,\text{in}\to k}$ and the division by $f_{2,j}$ are needed for normalization purposes. Notice that here the unknown parameter h_{1k} approximately captures the thermal convection effects happening on the components in the first zone. More specifically, h_{1k} subsumes in a scalar parameter the heat capacity of the air[1] and the heat capacity and thermal conductivity of the component in the first zone and k-th row (cf. (6.8) and see the comments thereafter for more details);
- as for the third zone, similarly as before the temperatures of the flow reaching the fans is

$$x_{32}^f = \frac{1}{u}\sum_{k=1}^3 \lambda_{3,k\to\text{out}} f_{2,j}\Big(x_{2k}^f + \frac{h_{2k}}{c_p}\left(x_{2k}^c - x_{2k}^f\right)\Big) \tag{6.7}$$

  where the structure follows closely that one of (6.6).

6.4 Thermal components modelling

We model the dynamics of the generic j-th component in the generic i-th zone through the classical Newton's laws for thermodynamics, i.e.,

$$\dot{x}_{ij}^c = \underbrace{-h_{ij} f_{i,j}\left(x_{ij}^c - x_{ij}^f\right)}_{\text{convection}} + \underbrace{\begin{bmatrix} R_{(ij)} & \rho_{ij}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}^c \\ x^{\text{in}}\end{bmatrix}}_{\text{conduction}} + \underbrace{b_{ij} p_{ij}}_{\text{el. power}} \tag{6.8}$$

with $i = 1,2,3$ the index of the zone in the server, $j = 1,2,3$ the component index, $\boldsymbol{x}^c$ an opportune column-vectorization of all the various scalars x_{ij}^c, and $R_{(ij)}$ a row vector of conduction parameters with the same cardinality as $\boldsymbol{x}^c$. In (6.8) we highlight three specific main contributions:

a convection term that expresses the rate at which heat is transferred between the electronic component and the air flow crossing it. As said before, h_{ij} describes the average heat transfer coefficient appearing in Newton's law of cooling. Notice that the rate of this heat exchange is, as expected, proportional to the mass of the air flow crossing the component (i.e., f_{ij}), that in its turn is a function of the controllable input u;

a conduction term that expresses the rate at which heat is exchanged through conduction among neighboring components and potential parasitic losses to the environment through the thermal resistance ρ_{ij} between the component and a fictitious environmental node;

a self-heating term that expresses the rate at which electrical power flowing through the electrical component is converted into heat.

[1] We note that the heat capacity at constant pressure of air is nearly constant in a neighbourhood of atmospheric pressure.

6.5 The complete model

For the electronic components in the first zone in Figure 6.1 the complete thermal model can be obtained by inserting $f_{1,j} = f_{1,\text{in}\to j} = \lambda_{1,\text{in}\to j} u$ and $x^f_{1j} = x^{\text{in}}$ in (6.8), leading thus to

$$\begin{aligned} \dot{x}^c_{1j} &= -h_{1j}\lambda_{1,\text{in}\to j} u \left(x^c_{1j} - x^{\text{in}}\right) + R_{(1j)}\boldsymbol{x}^c + \rho_{1j} x^{\text{in}} + b_{1j} p_{1j} \\ &=: \Psi_{1j}\left(\boldsymbol{x}^c, u, \boldsymbol{p}, x^{\text{in}} \,;\, \boldsymbol{\theta}\right) \end{aligned} \tag{6.9}$$

where $\boldsymbol{\theta}$ is a vector collecting all the parameters defining the model.

As for the components in the second zone in Figure 6.1, instead, the model can be obtained by inserting (6.3) and (6.6) in (6.8) plus simplifying the resulting equation, i.e.,

$$\begin{aligned} \dot{x}^c_{2j} &= -h_{2j} \sum_{k=1}^{3} \left(\lambda_{2,k\to j}\lambda_{1,\text{in}\to k} u\Big(x^c_{2j} - \big(x^{\text{in}} + \frac{h_{1k}}{c_p}\left(x^c_{1k} - x^{\text{in}}\right)\big)\Big)\right) \\ &\quad + R_{(2j)}\boldsymbol{x}^c + \rho_{2j} x^{\text{in}} + b_{2j} p_{2j} \\ &=: \Psi_{2j}\left(\boldsymbol{x}^c, u, \boldsymbol{p}, x^{\text{in}} \,;\, \boldsymbol{\theta}\right). \end{aligned} \tag{6.10}$$

As for the temperature at the air outlet, since there is no dynamics there, the equation is simply given by (6.7), i.e., $x^{\text{out}} = x^f_{32}$.

Assume then that the power consumption of the components is piecewise constant, and that the controllable inputs u and x^{in} are zero-order held. Discretizing either (6.9) or (6.10) using Euler's rule with a sample interval of length Δ yields thus to

$$x^c_{ij}(t+1) = x^c_{ij}(t) + \Delta\Psi_{ij}\left(\boldsymbol{x}^c(t), u(t), \boldsymbol{p}(t), x^{\text{in}}(t) \,;\, \boldsymbol{\theta}\right) \tag{6.11}$$

for $i = 1, 2$ and $j = 1, 2, 3$, and where t is used to index the various time instants. For completeness, we let the right-hand side of (6.7) to be equal to $\Psi_{32}\left(\boldsymbol{x}^c(t), u(t), \boldsymbol{p}(t), x^{\text{in}}(t) \,;\, \boldsymbol{\theta}\right)$, so that we can complete the set of equations (6.11) with

$$x^{\text{out}}(t) = \Psi_{32}\left(\boldsymbol{x}^c(t), u(t), \boldsymbol{p}(t), x^{\text{in}}(t) \,;\, \boldsymbol{\theta}\right). \tag{6.12}$$

6.6 Using CISSI for system identification purposes

Assume to have collected through the scripts described in Section 5 a dataset composed by measurements of $\boldsymbol{x}^c(t), u(t), \boldsymbol{p}(t), x^{\text{in}}(t), x^{\text{out}}(t)$ for $t = 1, \ldots, N$. With this information it is then possible to estimate the parameters defining (6.11) and (6.12) through a plethora of different system identification approaches.

Here we consider the standard strategy of estimating the unknown parameters as that ones that maximize the predictive capabilities of the model assuming the dataset to be affected by Gaussian measurement noises, and thus consider the various costs

$$J_{ij}\left(\boldsymbol{\theta}\right) := \sum_{t=1}^{N-1} \left(x^c_{ij}(t+1) - x^c_{ij}(t) - \Delta\Psi_{ij}\left(t \,;\, \boldsymbol{\theta}\right)\right)^2 \tag{6.13}$$

for $i = 1, 2, 3$ and $j = 1, 2, 3$ (notice that the notation for Ψ_{ij} has been simplified, omitting its dependence on $\boldsymbol{x}^c(t), u(t), \boldsymbol{p}(t), x^{\text{in}}(t)$ for simplicity) that then leads to the Prediction Error

Method (PEM) estimation strategy

$$\begin{aligned} \boldsymbol{\theta}^* = \quad & \arg\min_{\boldsymbol{\theta}\in\Theta} \sum_{i,j} J_{ij}\left(\boldsymbol{\theta}\right) \\ \text{s.t.} \quad & \textstyle\sum_{j=1}^{3} \lambda_{1,\text{in}\to j} = 1 \\ & \textstyle\sum_{k=1}^{3} \lambda_{2,k\to j} = 1 \quad j = 1,2,3 \\ & \textstyle\sum_{j=1}^{3} \lambda_{3,j\to\text{out}} = 1 \end{aligned} \tag{6.14}$$

with the hypothesis space Θ defined opportunely to guarantee the physical meaningfulness of the various estimated parameters.

6.7 Experiments on real systems

The experimental part of this work has been performed on the experimental datacenter SICS-ICE located in Luleå, Sweden [18]. The SICS-ICE datacenter runs a datacenter module dedicated to a Open Rack V1 systems, housing OCP *Facebook Server V2 Windmill* servers, storages and PDUs. As mentioned in Section 5, the experimental data has thus been gathered on this OCS platform during February and March 2017.

More precisely, the experiments have been performed running the *stress* module (cf. Figure 5.1) so that both the CPUs and the DIMMs followed a PWM signal with random durations of the various periods (cf. Figure 6.3 for a typical excitation pattern). Importantly, during these stress tests both the fans actuation signals u and the inlet temperature x^{in} were controlled by the native control systems in the datacenter (see Figure 6.4 for the values of these signals relative to PWM patterns shown in Figure 6.3).

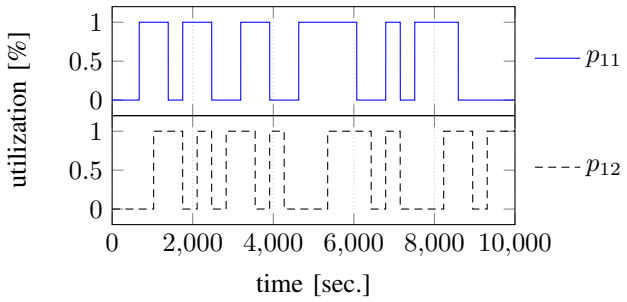

Figure 6.3: Typical pattern of the activation levels of the CPUs and the DIMMs requested by the *stress* module described in Section 5.3 and used in our system identification procedure.

As intuition states, following the procedure outlined in Section 6.6 we then get a different set of parameters every time we collect a different dataset. Despite these numerical changes, we report a specific set obtained using: *i)* a sampling period of 1 second; *ii)* a dataset comprising samples for three hours (i.e., 10800 samples); *iii)* a system excitation pattern like the one in Figure 6.3. We indeed noticed that models obtained using training sets as above have practically equivalent generalization capabilities on generic test sets. Moreover increasing the number of samples in the datasets (i.e., running longer training phases) leads to negligible improvements.

Importantly, in our settings we were endowed of servers not having temperature sensors for the lower DIMM modules (i.e., it was not possible to collect x^c_{13} and x^c_{23}). The following results

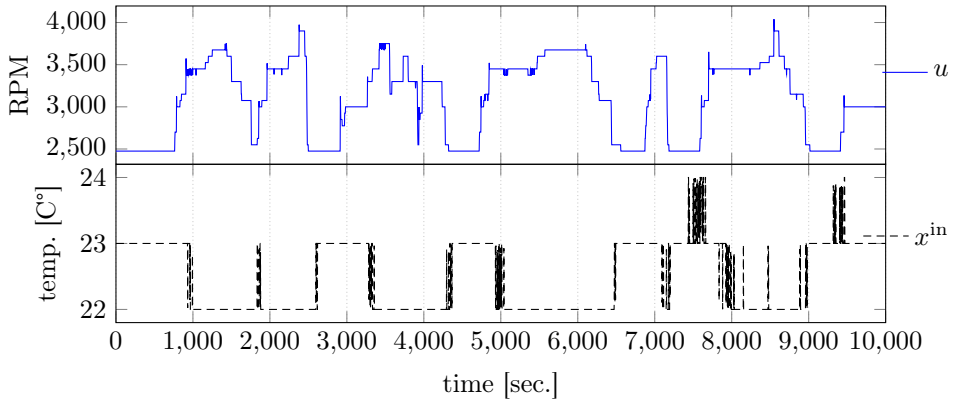

Figure 6.4: The fans actuation signal u and the inlet temperature x^{in} during the executions of the PWM patterns shown in Figure 6.3.

thus correspond to a simplified server where the contributions from these modules were ignored (i.e., where all the relative parameters λ's have been set to zero).

The following Table 6.1 thus reports the suggested parameters for this reduced model. To complement this table, our approach is to compute the approximation capabilities of the model through computing the quantities

$$\widehat{x}^c_{ij}(t+1) = \widehat{x}^c_{ij}(t) + \Delta\Psi_{ij}\left(\widehat{\boldsymbol{x}}^c(t), u(t), \boldsymbol{p}(t), x^{\text{in}}(t)\ ;\ \widehat{\boldsymbol{\theta}}\right) \tag{6.15}$$

initialized with $\widehat{\boldsymbol{x}}^c(0) = \boldsymbol{x}^c(0)$. Figure 6.5 presents the evolutions of typical realizations of $\widehat{\boldsymbol{x}}^c(t)$, $\boldsymbol{x}^c(t)$ and $\widehat{\boldsymbol{x}}^c(t) - \boldsymbol{x}^c(t)$. In other words, this figure compares (6.11) against (6.15), i.e., compares a typical evolution of the real system against the simulated one obtained through the identified parameters $\widehat{\boldsymbol{\theta}}$ reported in Table 6.1.

Figure 6.5 implicitly says that the model has good approximation capabilities and is able to reproduce the most important features of the true thermal process. We can nonetheless draw the following conclusions:

1. RAM modules tend to be associated to smaller simulation errors than CPUs in absolute values, but not in relative ones;

2. the parameters of the components relative to the first zone tend to be identified better than the ones relative to the second zone. Potentially, this is due to the combination of two facts: *i)* the dynamics of the air fluxes in the first zone tend to be simpler than the ones in the second zone; this implies that identifying what happens in the second zone is more complicated; *ii)* additional uncertainty may be due to the fact that the amount of information used to identify the parameters in the second zone is smaller than the amount of information used for the first zone: indeed in the latter we know x^{in}, i.e., we have measurements for the temperature of the flux hitting the elements in the first zone. For the second zone these measurements are substituted by estimates, that are naturally more uncertain than the measurements.

For completeness we report in Table 6.2 the normalized Mean Squared Errors (MSEs) of the

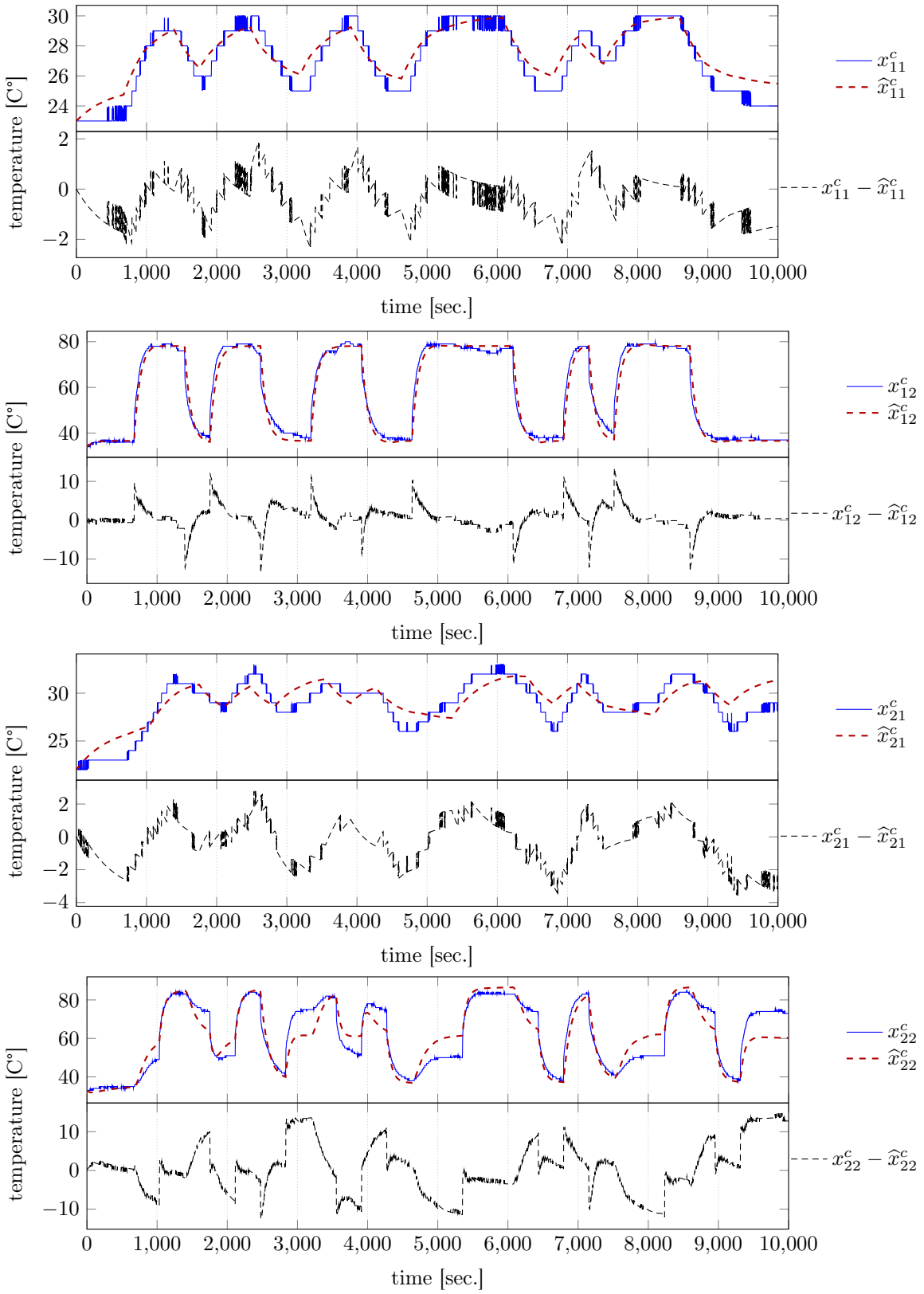

Figure 6.5: Comparison of the evolution of the real system again the simulator obtained through the identified parameters $\widehat{\boldsymbol{\theta}}$ reported in Table 6.1.

h_{11}	0.00000		$R_{(12),11}$	0.00013
h_{12}	0.00068		$R_{(21),11}$	0.00000
h_{21}	0.00000		$R_{(22),11}$	0.00000
h_{22}	0.00009		$R_{(11),12}$	0.00493
$\widetilde{h}_{11}$	359.83539		$R_{(21),12}$	0.00000
$\widetilde{h}_{12}$	906.39287		$R_{(22),12}$	0.00150
$\widetilde{h}_{21}$	1459.44161		$R_{(11),21}$	0.00000
$\widetilde{h}_{22}$	1459.44161		$R_{(12),21}$	0.00019
λ_{11}	0.99291		$R_{(22),21}$	0.00000
λ_{12}	0.00709		$R_{(11),22}$	0.00001
λ_{211}	0.47267		$R_{(12),22}$	0.00001
λ_{212}	0.06944		$R_{(21),22}$	0.00234
λ_{221}	0.52733		ρ_{11}	0.00000
λ_{222}	0.93056		ρ_{12}	0.00000
λ_{31}	0.50000		ρ_{21}	0.00000
λ_{32}	0.50000		ρ_{22}	0.00001
b_{11}	0.01167		b_{21}	0.01246
b_{12}	0.68022		b_{22}	0.83668

Table 6.1: Summary of the parameters identified from real data and proposed to be used for modelling the OCP servers under consideration.

errors plotted in Figure 6.5, i.e., the values of the quantities.

$$\frac{\left\|\widehat{x}_{ij}^{c} - x_{ij}^{c}\right\|^{2}}{\left\|x_{ij}^{c}\right\|^{2}} \tag{6.16}$$

for the test set plotted in Figure 6.5.

component indexes	*MSE*
11	0.00141
12	0.00278
21	0.00343
22	0.01240

Table 6.2: Normalized MSEs of the errors plotted in Figure 6.5.

Chapter 7

Minimum cost MPC fan control

The purpose of this chapter is to show how to use the models obtained in the thesis for practical purposes, so to complete and summarize the whole logical flow of improving the thermal cooling in a data centers by looking at it from a control-oriented perspective.

To exemplify how our models can be used to develop control schemes, we here provide an example of how to build a MPC scheme. Notice that all ideas and problem formulations used in this example are taken from another article, namely [36]. See the referenced article for a more thorough discussion on the usage of MPC strategies in data centers applications.

The problem is then to control the fans by solving the on-the-fly MPC problem defined in Equation 7.2, given the observations that: *i)* the rotational speeds of the fan are limited, so that the control values u are constrained in the hyper-rectangle defined by the extreme points $u_{\min}, u_{\max} \in \mathbb{R}^m_{\geq 0}$; *ii)* the temperatures of the IT components shall be kept below some specified safe limits, that implies that there exist state constraints of the kind $\boldsymbol{x}^c \preceq \boldsymbol{x}^c_{\max} \in \mathbb{R}^n_{\geq 0}$; *iii)* the concept of minimizing the cooling provisioning can be translated into minimizing the sum of the power consumption of the individual fans while guaranteeing the temperatures of the IT components to be within their limits. In first approximation, then, the power necessary to produce a given air mass flow is proportional to the product of the generated pressure drop and the mass flow itself. For the fan at time t, the latter product is then proportional to the cubic power of the control value $u(t)$. Assume that the dynamics of the system can be written as

$$\boldsymbol{x}^c(t+1) = \boldsymbol{\Psi}^{\Delta}\left(\boldsymbol{x}^c(t), \boldsymbol{u}(t), \boldsymbol{p}(t), x^{\text{in}}\right) \tag{7.1}$$

with $\boldsymbol{\Psi}$ defined by the superposition of the various (6.12) for the various i and j. Then the control problem can be thus stated as

$$\begin{aligned}
&\min_{u(0),\ldots,u(T-1)} \sum_{t=0}^{T-1} u(t)^3 \\
&\text{subject to:} \\
&\quad \boldsymbol{x}^c(0) = \boldsymbol{x}^c_0 \\
&\quad u_{\min} \preceq u(t) \preceq u_{\max} \\
&\quad \boldsymbol{x}^c(t+1) \preceq \boldsymbol{x}^c_{\max} \\
&\quad \boldsymbol{x}^c(t+1) = \boldsymbol{\Psi}^{\Delta}\left(\boldsymbol{x}^c(t), u(t), \boldsymbol{p}(t), x^i\right)
\end{aligned} \tag{7.2}$$

Notice that the electrical power consumption of the n IT components over the future horizon is unknown, i.e., at each time t the values of $\boldsymbol{p}(t), \boldsymbol{p}(t+1), \ldots, \boldsymbol{p}(t+H-1)$ are unknown. It

is thus possible to use the following heuristic: consider that the worst case temperature and cost scenario in (7.2) occurs when the server runs at its full computational load, i.e., when $\boldsymbol{p}(t) = \boldsymbol{p}_{\max}, t = 0, 1, \ldots, H-1$. One may then use as a forecast of the future power consumption $\boldsymbol{p}(t), \boldsymbol{p}(t+1), \ldots$ the worst-case power consumption. Of course this strategy may lead to over-cooling. An other strategy is to use, as forecasts of the future computational loads, the same computational load that is currently seen (a.k.a. *persistence* forecaster).

Chapter 8

Conclusions and future works

8.1 Conclusions

There are several control-oriented strategies using model-based approaches that can, at least nominally, improve the thermal cooling in data centers with respect to non-model-based ones. To be able to implement these strategies, though, one needs accurate descriptions of the dynamics of the system. These models should thus capture how the various components inside a thermal network affect each other while keeping the model sufficiently simple to be used for control purposes.

A key part to obtain accurate control-oriented models is then to obtain accurate data-driven estimates of the unknown descriptive parameters. This, in its turn, implies the need for collecting measurements from all the potential states where the system can be. In other words, there is the need for applying inputs to the system excites it, in the sense that these inputs should steer the system away from its natural equilibria. In data centers, this can be done by controlling opportunely the IT loads of the various servers and then by measuring the thermal status of these servers.

This thesis meets the needs above by presenting a software strategy for remotely monitoring and controlling OCP servers. It also provides a control-oriented linear model of the thermal dynamics of the system and identifies this linear model from real data. Furthermore the thesis provide a set of parameters to the community so that everybody willing to do so can: *i)* construct a simple simulator of the thermal behaviour of a server on which to perform tests *in-silico*; *ii)* build opportune model-based control strategies for the cooling of servers in a data center.

The experiments performed in this thesis show a promising potential to obtain a good control-oriented model of single servers that is able to reproduce the most important features of the true thermal process. The results provide thus a basis for future researchers to further develop their strategies.

8.2 Future works

Starting from the results presented above we foresee these distinct branches of future works:

Modelling, in the sense of improving or comparing the obtained models to ameliorate their predictive capabilities. In more details, the identified thermal dynamics have good approximation capabilities. They are however based on very simplified assumptions of linearity and perfect air mixing (cf. the introduction of Section 6). It may be that using non-linear

terms, or combining the model with results from opportune Computational Fluid Dynamics (CFD) tools, may lead to improved results;

Monitoring, in the sense of continuously comparing the results provided by the model with the measurements returned by the system to check if something is behaving in a non-expected way. This would solve the practical problem of running sanity checks of the hardware and instead signal faults when they occur;

Controlling, in the sense of building upon this thesis as a basis for model-based control strategies, e.g., such as LQRs or MPCs approaches.

At higher hierarchical layers, there is also the need for improving the software to handle multiple servers or a hole rack. More precisely the software suite *CISSI* can be improved:

1. in the *stress* module by advancing its capabilities of stressing of the CPUs by making it able to individually stress the DIMMs modules;
2. by implementing in *CISSI* a Graphical user interface (GUI) so to make it more user friendly (also by means of a tool that can be installed and set-up automatically directly from an application distribution repository);
3. by completing the *control* module by testing it on a OCS blade that support the FSC software.

Notice that another approach that was considered when developing the *control* module was to build a hardware-based controller of the server's fans. This approach was however not pursued because a software based solution would be more beneficial in an economical and resource perspective. Unfortunately during the development of the software solution it was discovered that the FSC was not supported by the OCS blade used in this project, so that, a-posteriori, following the hardware solution might have been beneficial. The development of a hardware based controller has therefore been started as a separate project within SICS.

Bibliography

[1] datacenterdynamics, http://www.datacenterdynamics.com/.

[2] dchuddle, http://www.dchuddle.com/.

[3] Sics, https://www.sics.se/media/news/a-holistic-approach-on-automation-in-data-centers.

[4] Storagereview, http://www.storagereview.com/dell_poweredge_13g_r730xd_review.

[5] Cloudberry datacenters,http://www.cloudberry-datacenters.com, 2017.

[6] Data center knowledge,http://www.datacenterknowledge.com/archives/2014/10/15/how-is-a-mega-data-center-different-from-a-massive-one/, 2017.

[7] Data center knowledge,http://www.datacenterknowledge.com/archives/2016/10/19/here-are-the-top-dcim-software-vendors-according-to-gartner/, 2017.

[8] Facebook server fan speed control interface, http://www.opencompute.org/wiki/server/specsanddesigns, 2017.

[9] Free ipmi, https://www.gnu.org/software/freeipmi/, 2017.

[10] Intelligent platform management interface, http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-v2-rev1-1-spec-errata-6-markup.html?wapkw=ipmi, 2017.

[11] Net-snmp, http://www.net-snmp.org/, 2017.

[12] The node pole, http://thenodepole.com/, 2017.

[13] Open compute project, http://opencompute.org/, 2017.

[14] Open compute server remote monitoring and control, https://github.com/lulea-datacenter-control/open-compute-server-remote-monitoring-and-control/, 2017.

[15] Open ipmi, http://openipmi.sourceforge.net/, 2017.

[16] Simple network management protocol, https://tools.ietf.org/html/rfc1067, 2017.

[17] Stress-ng, http://kernel.ubuntu.com/ cking/stress-ng/, 2017.

[18] Swedish institute of computer science, infrastructure and cloud datacenter test environment, https://www.sics.se/media/news/sics-ice-data-center-in-lulea, 2017.

[19] ABB. ABB Review - Datacenters. *The corporate technical journal*, pages 1–84, 2013.

[20] Zahra Abbasi, Georgios Varsamopoulos, and Sandeep K. S. Gupta. Thermal aware server provisioning and workload distribution for internet data centers. *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing - HPDC '10*, page 130, 2010.

[21] C.E. Bash, C.D. Patel, and R.K. Sharma. Dynamic Thermal Management of Air Cooled Data Centers. *Thermal and Thermomechanical Proceedings 10th Intersociety Conference on Phenomena in Electronics Systems, 2006. ITHERM 2006.*, pages 445–452, 2006.

[22] Yulia Berezovskaya, Arash Mousavi, Valeriy Vyatkin, Xiaojing Zhang, and Tor Björn Minde. Improvement of energy efficiency in data centers via flexible humidity control. In *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*, pages 5585–5590. IEEE, 2016.

[23] Alfonso Capozzoli and Giulio Primiceri. Cooling Systems in Data Centers: State of Art and Emerging Technologies. *Energy Procedia*, pages 484–493, 2015.

[24] Cisco. Cisco Global Cloud Index : Forecast and Methodology , 20152020. *White Paper*, pages 1 – 29, 2016.

[25] M Dayarathna, Y Wen, and R Fan. Data Center Energy Consumption Modeling: A Survey. *IEEE Communications Surveys & Tutorials*, 18(1):1–794, 2016.

[26] Tony Evans. The Different Technologies for Cooling Data Centers. *White Paper 59*, 2:1–16, 2012.

[27] Anshul Gandhi, Sherwin Doroudi, Mor Harchol-Balter, and Alan Scheller-Wolf. *Exact analysis of the M/M/k/setup class of Markov chains via recursive renewal reward*, volume 77. 2014.

[28] Anshul Gandhi, Mor Harchol-Balter, and Ivo Adan. Server farms with setup costs. *Performance Evaluation*, 67(11):1123–1138, 2010.

[29] Anshul Gandhi, Mor Harchol-Balter, Rajarshi Das, and Charles Lefurgy. Optimal power allocation in server farms. *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems - SIGMETRICS '09*, page 157, 2009.

[30] Steve Greenberg, Evan Mills, Bill Tschudi, Peter Rumsey, and Bruce Myatt. Best practices for data centers: Lessons learned from benchmarking 22 data centers. *Proceedings of the ACEEE Summer Study on Energy Efficiency in Buildings in Asilomar, CA. ACEEE, August*, 3:76–87, 2006.

[31] Z Han, H Tan, G Chen, R Wang, Y Chen, and F C M Lau. Dynamic virtual machine management via approximate Markov decision process. *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, 2016.

[32] Han-Peng Jiang, David Chuck, and Wei-Mei Chen. Energy-Aware Data Center Networks. *Journal of Network and Computer Applications*, 2016.

[33] Dong Ki Kang, Fawaz Al-Hazemi, Seong Hwan Kim, Min Chen, Limei Peng, and Chan Hyun Youn. Adaptive VM Management with Two Phase Power Consumption Cost Models in Cloud Datacenter. *Mobile Networks and Applications*, 21(5):793–805, 2016.

[34] Hassan K Khalil. *Noninear Systems.* Prentice-Hall, New Jersey, 1996.

[35] Lei Li, Chieh-Jan Mike Liang, Jie Liu, Suman Nath, Andreas Terzis, and Christos Faloutsos. ThermoCast : A Cyber-Physical Forecasting Model for Data Centers Categories and Subject Descriptors. *KDD'11: 17th annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1370–1378, 2011.

[36] Riccardo Lucchese, Jesper Olsson, Anna-Lena Ljung, Winston Garcia-Gabin, and Damiano Varagnolo. Energy savings in data centers: A framework for modelling and control of servers' cooling. *IFAC World Congress*, 2017.

[37] Mohammad Masdari, Sayyid Shahab Nabavi, and Vafa Ahmadi. An overview of virtual machine placement schemes in cloud computing. *J. Netw. Comput. Appl.*, 66(C):106–127, May 2016.

[38] Sparsh Mittal. Power Management Techniques for Data Centers: A Survey. *arXiv preprint arXiv:1404.6681*, 2014.

[39] Arash Mousavi, Valeriy Vyatkin, Yulia Berezovskaya, and Xiaojing Zhang. Cyber-physical design of data centers cooling systems automation. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 3, pages 254–260. IEEE, 2015.

[40] Arash Mousavi, Valeriy Vyatkin, Yulia Berezovskaya, and Xiaojing Zhang. Towards energy smart data centers: Simulation of server room cooling system. In *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*, pages 1–6. IEEE, 2015.

[41] Pradeep Padala, Kang G. Shin, Xiaoyun Zhu Mustafa, Uysal Zhikui Wang, and Sharad Singhal Arif. Adaptive control of virtualized resources in utility computing environments. *In Proceedings of the European Conference on Computer Systems*, pages 289–302, 2007.

[42] Pan European Datacenter Academy Project (PEDCA). PEDCA Final Report Summary. Technical report, 2014.

[43] Alessandro Vittorio Papadopoulos, Cristian Klein, Martina Maggio, Jonas Dürango, Manfred Dellkrantz, Francisco Hernández-Rodriguez, Erik Elmroth, and Karl-Erik Årzén. Control-based load-balancing techniques: Analysis and performance evaluation via a randomized optimization approach. *Control Engineering Practice*, 52:24–34, 2016.

[44] L Parolini, E Garone, B Sinopoli, and B H Krogh. A hierarchical approach to energy management in data centers. *2010 49th IEEE Conference on Decision and Control, CDC 2010*, pages 1065–1070, 2010.

[45] L Parolini, B Sinopoli, B H Krogh, and Z K Wang. A cyber-physical systems approach to data center modeling and control for energy efficiency. *Proceedings of the IEEE*, 100(1):254–268, 2012.

[46] Luca Parolini. Models and Control Strategies for Data Center Energy Efficiency. 2012.

[47] Luca Parolini, Niraj Tolia, Bruno Sinopoli, and Bruce H Krogh. A cyber-physical systems approach to energy management in data centers. *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, pages 168–177, 2010.

[48] Luca Parolini, Niraj Tolia, Bruno Sinopoli, and Bruce H Krogh. A cyber-physical systems approach to energy management in data centers. *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, pages 168–177, 2010.

[49] Luca Parolini, Niraj Tolia, Bruno Sinopoli, and Bruce H Krogh. A Cyber-Physical Systems Approach to Energy Management in Data Centers. *1st IC-CPS*, pages 168–177, 2011.

[50] Tuan Phung-Duc. Server farms with batch arrival and staggered setup. In *Proceedings of the Fifth Symposium on Information and Communication Technology*, SoICT '14, pages 240–247, New York, NY, USA, 2014. ACM.

[51] V K Mohan Raj and R Shriram. Power management in virtualized datacenter a survey. *Journal of Network and Computer Applications*, 2016.

[52] Neil Rasmussen and Wendy Torell. Comparing Data Center Power Distribution Architectures. *White Paper*, 3:1–14, 2014.

[53] Shuxiu Road. A Load Balancing Aware Virtual Machine Live Migration Algorithm Chengjiang Liu 1 1. (266):370–373, 2016.

[54] The Boston Consulting Group. Digital Infrastructure and Economic Development: An impact assessment of Facebook's data center in Northern Sweden. Technical report, The Boston Consulting Group, 2014.

[55] Niraj Tolia, Zhikui Wang, Parthasarathy Ranganathan, Cullen Bash, Manish Marwah, and Xiaoyun Zhu. Unified thermal and power management in server enclosures. *InterPACK09*, 2009.

[56] X. Wang and Y. Wang. Coordinating power control and performance management for virtualized server clusters. *IEEE Transactions on Parallel and Distributed Systems*, 22(2):245–259, Feb 2011.

[57] Xiaoqi Yin and Bruno Sinopoli. Adaptive Robust Optimization for Coordinated Capacity and Load Control in Data Centers. *Conference on Decision and Control*, pages 5674–5679, 2014.

[58] Hao Zhu, Xiangke Liao, Cees de Laat, and Paola Grosso. Joint flow routing-scheduling for energy efficient software defined data center networks. *Journal of Network and Computer Applications*, pages 1–15, 2016.