UNIVERSITY OF SALERNO

Department of Computer Science

Master Degree in Computer Science

THESIS IN INFORMATION SECURITY

Exploring the correlation between PUE variation and cyberattacks: impact on Data Centers Security

SUPERVISORS

Prof. **Michele Mastroianni**

Prof. **Francesco Palmieri**

CANDIDATE

Vincenzo Emanuele Martone

Academic Year 2022-2023

Abstract

The aim of this research study is to investigate how fluctuations in Power Usage Effectiveness *PUE* values may help in the detection of cyberattacks, considering the potential impact on Data Centers security. By analyzing real-world data gathered through a specific simulation tool and modeling two attack scenarios, namely a Denial of Service (*DoS*) attack and a cooling system attack, this research aims to provide valuable insights into the relationship between *PUE* variation and attacks detection, determining whether the implementation of a detection algorithm based on *PUE* monitoring is feasible.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

In this chapter an overview about the problem addressed by this research study will be provided through the explanation of the motivations behind the study and the introduction of all the elements necessary to understand the scope of this work.

1.1 Overview and objectives

This research work aims to analyze and evaluate the utilization of the Data Center energy efficiency parameter *PUE (Power Usage Effectiveness)* for real-world cyberattacks detection. In particular, two attack scenarios will be studied and simulated, namely *DoS (Denial of Service)* and cooling system attack, in order to establish if there is a significant variation in the *PUE* parameter that leads to their detection.

1.1.1 The problem of energy consumption in Data Centers

Before discussing the problem addressed in this research work, it is worth to discuss the issue of energy consumption in Data Centers. An article published in 2020 [1] stated that in 2018, it was estimated that the electricity consumption of Data Centers accounted for 1% of the global electricity consumption, and the growing trend in consumption needs monitoring aimed at limiting the impact on global energy demand. Among the strategies mentioned to counteract this phenomenon, there is the promotion of energy efficiency standards for servers, storage devices, and networking equipment, as well as for benchmarking these devices. Finally, there is a strong encouragement for continuous reduction of *PUE*, as it is indicative of greater energy efficiency (although comparing two Data Center using *PUE* is not straightforward). A concerning reason for which it is absolutely important to take action is related to greenhouse gas emissions. A 2018 study [2] stated that, estimating a future annual growth in emissions ranging from 5.6% to 6.9%, the contribution of *ICT* will exceed 14% of the global greenhouse gas emissions of 2016 by 2040. Overall, developing an energy-based detection technique results effective to encourage the measurement of the energy consumption of the Data Center, spreading an 'energy culture' which is useful for reducing operational costs and emissions.

1.1.2 PUE definition

PUE (Power Usage Effectiveness) is a Data Center efficiency parameter introduced by a non-profit consortium called *The Green Grid* in 2007 [3]. It is defined as the ratio of total facility energy to *IT* equipment energy (equation 1.1.1):

$$PUE = \frac{TotalFacilityEnergy}{ITEquipmentEnergy} \tag{1.1.1}$$

Total Facility Energy is defined as the energy consumed by the whole Data Center (including *IT* equipment, power delivery components, cooling system components and Data Center lighting), while *IT Equipment Energy* is defined as the energy consumed to run the facility's *IT* infrastructure. *PUE* value can range from 1 to infinity. A *PUE* value of 1 represents an ideal scenario where all the power consumed is used by the *IT* equipment, making it highly efficient. As the *PUE* value increases above 1, it indicates that a greater portion of the total

power is consumed by *non-IT* equipment, which reduces the overall energy efficiency of the Data Center. Since this parameter provides an insight of the Data Center efficiency, describing how much energy is used by the *IT* equipment, it has been recognized globally as the industry's most used infrastructure efficiency metric for Data Centers. However, *PUE* depends on many attributes such as Data Center design and implementation: for this reason it is difficult to compare Data Centers based on public *PUE* reports. For this purpose, *The Green Grid* has provided a set of guidelines for organizations about making public claims regarding Data Centers *PUE* in order to make this parameter suitable for comparisons.

1.1.3 PUE calculation

Avelar et al. [3] provides a three-level approach for measuring *PUE*. Each level provides a certain level of detail in *PUE* measurement, considering additional measurement points in order to provide further insight into Data Center infrastructure components' energy consumption. There are three main parameters that vary depending on the chosen level of measurement, namely: *IT Equipment Energy*, *Total Facility Energy* and *Measurement Interval*.

Level 1: Basic

With a level 1 measurement, the parameters are calculated as follows:

- ***IT* Equipment Energy**: it is measured at the output of the *UPS (Uninterruptible Power Supply)* equipment;
- **Total Facility Energy**: it is measured from the utility service entrance that supplies power to all the equipment within the Data Center;
- **Measurement Interval**: power measurements are performed once a month.

Level 2: Intermediate

With a level 2 measurement, the parameters are calculated as follows:

- ***IT* Equipment Energy**: it is measured at the output of the *PDUs (Power Distribution Unit)* equipment;
- **Total Facility Energy**: it is measured from the utility service entrance that supplies power to all the equipment within the Data Center;
- **Measurement Interval**: power measurements are performed once a day.

Level 3: Advanced

With a level 3 measurement, the parameters are calculated as follows:

- ***IT* Equipment Energy**: it is measured at each component of the Data Center excluding *non-IT* loads;
- **Total Facility Energy**: it is measured from the utility service entrance that supplies power to all the equipment within the Data Center;
- **Measurement Interval**: power measurements are performed at least every 15 minutes.

1.1.4 Denial of Service

DoS (Denial of Service) attack is a cyberattack that aims to make a resource unavailable to its users, usually performed either by flooding the hosts that provide a specific service until they are unable to respond to their legitimate users or by exploiting a vulnerability of the target. There are various *DoS* attack techniques that have been identified and categorized during the last years, such as *SYN Flood* and *Smurf* [4]. Nowadays, the most commonly used technique is the *Distributed Denial of Service (DDoS)* attack, which is a variation of *DoS* where the attack is carried out by multiple machines that are under the control of the attacker who is usually able to take control of several machines by exploiting security weaknesses [4]. Since *DoS* leads to damage in terms of time, money and reputation for an organization, implementing network monitoring mechanism aimed at the detection of *DoS* attacks is crucial.

1.1.5 Cooling system attack

Cooling system attacks pose severe security risks to Data Centers. Over the years, several authors have focused their attention on this specific kind of attacks, analyzing various aspects such as the threat model and the impact of thermal attacks. *Zhihui Shao et al.* [5] described the attacker's capabilities in a thermal attack scenario, indicating that the attacker runs power-intensive applications to increase server power consumption, resulting in a single server equipped with multiple CPUs and GPUs consuming a large amount of energy. These authors also divided the impact of thermal attacks in two main categories, namely:

- **Performance degradation**: when the server temperature exceeds a certain threshold, a thermal emergency occurs. In this scenario the server runs in a low power state mode to avoid the temperature from rising further, which could lead to hardware damage. It is evident that this remediation strategy results in performance degradation;
- **System outage**: when a thermal emergency occurs and the server temperature keeps rising despite the load capping, the system automatically shuts down to prevent hardware damage. The consequences of a system outage depend on the nature of the application and may become catastrophic in the case of latency-critical applications.

The aforementioned work also explores thermal attack strategies and their feasibility aiming to describe a real-world attack scenario and highlight the motivations behind thermal attacks.

1.2 Motivations

There are two main reasons that led to the study of techniques based on *PUE* monitoring for cyberattacks detection. The first motivation concerns the ease of *PUE* calculation that does not require specific equipment apart from wattmeters that should be placed at various points within the Data Center, depending on the level of granularity desired in *PUE* calculation (as described in section 1.1.3). The second motivation is closely related to the cooling system attack scenario as it highlights the reasons why it is essential to be concerned about potential attacks on the Data Centers' management system. In recent years, several research studies have explored various methodologies to integrate Cloud and IoT solutions in the context of Data Centers monitoring and management. In 2016 *Q Liu et al.* [6] proposed an air conditioning system that includes both IoT sensors and cloud-based systems for Data Center management. Moreover, in 2020 *Ramphela et al.* [7] proposed an integrated monitoring system for Data Centers based on the development of various subsystems, such as an embedded system that includes sensors that gather monitoring data. In this context, the pervasive usage of IoT and Cloud technologies exposes Data Centers to additional risks. As reported in the research work by *Francesca Meneghello et al.* [8], the main risk associated with the IoT devices comes from the lack of the implementation of security mechanisms especially in cheaper devices that are widely spread. An unauthorized access to the Data Center's control system could lead to the modification of the cooling system's operational parameters. In cases where such violation may go unnoticed by the Intrusion Detection or Intrusion Prevention System, the proposed approach based on monitoring energy parameters can be a viable option for detecting such unauthorized access.

1.3 Results

The achieved results suggest that the correlation between *PUE* fluctuations and the presence of cyberattacks is not negligible in both simulated scenarios. In particular, in the *DoS* scenario, there is a substantial variation in *PUE* up to a certain load threshold (approximately 80%), while in the cooling system attack scenario, the *PUE* variation is independent of the Data Center load, making the proposed approach particularly suitable for the detection of this specific attack. A detailed analysis of the results achieved is presented in Chapter 4, where the entire research work process is comprehensively discussed.

1.4 Structure of the Thesis

The Thesis is structured as follows:

- **Chapter 2 (State of the Art):** this chapter provides an overview about the currently available techniques of *DoS* and cooling system attack detection. Furthermore, in

order to choose a viable platform to perform Data Center simulations, several Cloud simulators will be analyzed and compared;

- **Chapter 3 (GreenCloud Simulator Overview):** this chapter explores in-depth the *GreenCloud* simulator as it has been chosen as the reference tool for the simulations;
- **Chapter 4 (Data Center Design):** this chapter describes the architecture, the energy model, the *IT* capacity and the power and cooling parameters of the virtual Data Center used for the performed simulations;
- **Chapter 5 (Research Study):** this chapter illustrates the research study process, starting from the changes implemented in *GreenCloud* in order to make it suitable for this work and then describing the performed simulations and the *PUE* calculation. The chapter concludes with a discussion on the achieved results, providing an idea of how the variation of *PUE* can be a suitable parameter for attacks detection;
- **Chapter 6 (Conclusions and Future Work):** this chapter summarizes the research study and discusses potential future works.

CHAPTER 2

State of the art

In this chapter an overview about the currently available techniques of *DoS* and cooling system attack detection will be provided. Furthermore, in order to choose a viable platform to perform Data Center simulations, several Cloud simulators will be analyzed and compared.

2.1 Denial of Service detection techniques

Since Denial of Service attacks have always been a crucial threat to Data Centers security, over the years researchers carried out several studies about *DoS* detection techniques. Most of these techniques rely on the use of Intrusion Detection Systems *(IDS)* implementing detection mechanisms mostly based on Artificial Intelligence techniques. This section summarizes some of the research works about this topic available in literature:

- *Wei Zhou et al.* [9] proposed a method to detect Application Layer Distributed Denial of Service (*AL-DDoS*) attacks in heavy backbone traffic. This approach models the traffic in real-time and, through an examination of the entropy of AL-DDoS attacks and flash crowds (i.e. the legitimate traffic), it is capable to distinguish between these two cases and detect real *AL-DDoS* attacks. The authors tested this algorithm on real data and claimed it is effective to identify and stop attack sources. However they mentioned the need to improve the reaction rate leaving it as a possible future development;

- *JaehakYu et al.* [10] proposed a lightweight detection algorithm for *DoS* that collects statistical information from *Simple Network Management Protocol (SNMP)* agents instead of analyzing raw packet data. This approach, combined with the machine learning technique based on a *Support Vector Machine (SVM)* for attack classification, resulted in rapid detection and high accuracy;

- *Ming-hui Yang et al.* [11] improved *SVM* detection technique, introducing an approach that combined *SVM* and the wavelet kernel function (i.e. a multidimensional wavelet function that can approximate arbitrary nonlinear functions [12]) theory, achieving an improvement of 4% compared to the traditional *SVM* approach;

- *T. Spyridopoulos et al.* [13] proposed a game-theoretic approach to *DDoS* attack detection where the *DDoS* attack is modeled as a one-shot non-cooperative, zero-sum game. By analyzing several parameters such as the cost to perform the attack, the number of attacking nodes and malicious traffic probability distributions, authors were able to identify the only optimal strategy available to the defender under the hypothesis that the attacker is a rational player;

- *Uğur Akyazı et al.* [14] capitalized on the similarity between the architecture of *IDS* and the Biological Immune Systems, developing an Artificial Immune System as a method of anomaly-based *IDS*. Their algorithm achieved high accuracy and precision under certain parameters and experimental conditions described in the forementioned paper;

- *Wanchun Dou et al.* [15] proposed a real-time Confidence-Based Filtering method as a *DDoS* defending approach. This approach collects packets from non-attacking periods in order to generate a nominal traffic profile that is used to calculate the score of packets

in the attack period. This score indicates whether to discard or not a certain packet. Authors claimed that the proposed algorithm has a high scoring speed, a small storage requirement and an acceptable filtering accuracy;

- *Mehdi Barati et al.* [16] proposed a detection architecture that uses a Genetic Algorithm for feature selection and an Artificial Neural Network for attack detection. High accuracy and deniable false alarm have been achieved by this algorithm.

2.2 Intrusion Detection systems

Cooling system violations often occur through an unauthorized access to the system. In the context of Data Centers, the utilization of network and system monitoring devices, known as *'Intrusion Detection Systems'*, is highly common. Implementation of these devices is based on various techniques, mainly artificial intelligence, in order to detect intrusions within a system. Based on the device where detection takes place, it is possible to distinguish between two categories of *IDS*, namely:

- **Host-based *IDS*:** detection mechanisms are implemented on each host by monitoring several parameters such as system integrity, logs and timestamps. If an intrusion is detected the host user and the central server are alerted and the suspicious activity is blocked [17];

- **Network-based *IDS*:** detection mechanisms are implemented through a group of sensors that monitor packets over the network. Violations are detected by recognizing intrusion patterns in the captured packets [17].

IDS can be further categorized according to the employed detection technique. *Khraisat et al.* [18] published a review of types of *IDS* and detection techniques, where they described two main groups of *IDS*:

- **Signature-based Intrusion Detection System (*SIDS*):** this type of *IDS* is based on an intrusion signature database that is compared to the current traffic pattern in order to trigger an alarm if a pattern match occurs. Due to its operational methodology, it becomes clear that this type of *IDS* performs exceptionally well when it comes to identify known intrusions but it is not capable to identify zero-day attacks;

- **Anomaly-based Intrusion Detection System (*AIDS*):** this type of *IDS* overcomes the main limit of *SIDS* since it builds a nominal behavior of a system through various techniques such as machine learning models, statistical approach, pattern matching, data mining, biological models and clustering. Any deviation from this nominal behavior is identified as an intrusion and triggers an alarm. The nominal behavior is created through a training phase using the normal traffic profile. Subsequently a testing phase

is carried out using a different dataset in order to assess how well the constructed model is able to identify unkown intrusions. Since this system is not based on a database of known signatures, it is able to identify zero-day attacks.

2.3 Simulation tools comparison

Simulation tools play a critical role in various domains of cloud computing. They offer researchers and infrastructure designers a virtual environment to work in, eliminating the expenses associated with physical infrastructure. Over time, several authors have conducted comparative studies to assess different simulators and highlight their unique characteristics, assisting users in selecting the most suitable option for their specific context. In this section, several cloud simulators that have been studied by the researchers will be discussed, focusing on their main strengths, drawbacks and the problems they aim to address. Since the main focus is to identify a simulator that is well-suited for energy consumption metrics extraction, features that ensure the accuracy of the simulations will be prioritized in order to make this work relevant in real scenarios. The following argument is based on several previous surveys, namely: *N. Mansouri, et al.* [19], *P. Suryateja* [20], *D. Perez Abreu, et al.* [21], *Khaled M. Khalil, et al.* [22], *Nimisha Patel & Hiren Patel* [23].

CloudSim

When it comes to cloud simulators it is impossible to overlook *CloudSim* [24], as it is one of the most widely used *event-driven* tools among researchers. A search for the keyword "*CloudSim*" on the *Scopus* platform yields 2020 results (this search was performed in May 2023), indicating its widespread adoption in the research community. Over the years, several simulators based on *CloudSim* have been proposed. This tool, written in *Java*, is comprehensive and highly extensible. One notable feature is its virtualization engine, which allows the creation and management of virtualization services on a network node. Additionally, *CloudSim* provides the capability to allocate machine cores in two different ways: space-shared and time-shared. In space-shared allocation, each machine is divided into a set of cores, and each core is assigned to a single job until it is completed. On the other hand, in time-shared allocation multiple jobs can be assigned to a single core and each of them is executed for a certain amount of time before another job is chosen. Overall, *CloudSim* offers researchers a comprehensive and flexible platform to simulate and study virtualization and resource allocation techniques [19]. Architecture of *CloudSim* is shown in figure 2.1. It is composed by three layers, described below:

- **CloudSim core simulation engine**: initially implemented through the discrete event simulation engine *SimJava* that supports functionalities such as queuing, processing of events, creation of Cloud system entities, communication between components and

management of the simulation clock ([24]), it has been replaced with an engine that provides some advanced operations;

- ***CloudSim* simulation level**: it provides various interfaces and services that allow to model and simulate Cloud-based Data Center environments;
- **User code**: it allows to set up various simulation parameters such as number of machines, their specification, number of tasks and broker scheduling policies.

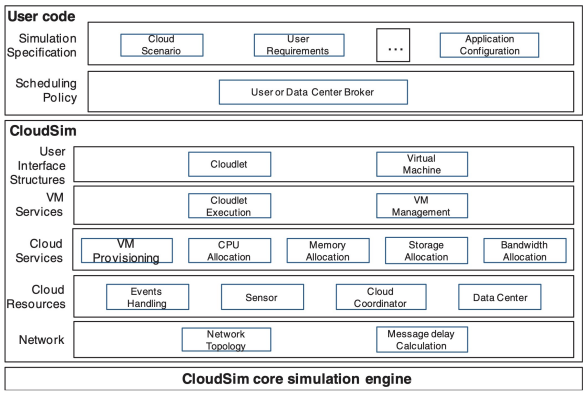

Figure 2.1: CloudSim architecture

NetworkCloudSim

NetworkCloudSim [25] implements a network layer and provides several communication models, including message-based, packet-based and flow-based models. It also offers an accurate evaluation of the scheduling of machines in the Data Center. Finally, it provides a basic energy model of the Data Center, although it does not extensively focus on energy efficiency and it does not provide a packet-level network model [19] [23].

CloudAnalyst

CloudAnalyst [26] makes simulation work easier as it provides a *GUI*. Its main feature is the ability to gather information about the geographical location of users and Data Centers. The simulator also offers a set of metrics based on response time and request processing. However, it does not implement a complete communication model based on the *TCP/IP* protocol and it provides a poor energy model [19].

EMUSim

EMUSim [27] includes an emulation and a simulation environment. The simulator is able to get profiling data about the running application behavior through the emulation

environment; these data are used to build a simulation environment. However, the emulation environment strongly limits scalability and makes the simulator not suitable for large workload scenarios [19].

CDOSim

CDOSim [28] integrates *CloudSim* simulator and *CloudMIG* framework. This simulator adopts a scaling technique that assigns a new virtual machine when CPU usage exceeds a specific threshold. It provides a set of client-centric metrics, allowing developers to compare different solutions based on various deployment parameters. Another noteworthy feature is the presence of a benchmark module that detects the impact of choosing a specific architecture on application performance. However, the communication model employed by *CDOSim* is overly simplistic and strongly limits large-scale applications [19].

TeachCloud

TeachCloud [29] is designed to be suitable for students who want to have some practical experience with Cloud Computing simulation, covering aspects such as networking, Service Level Agreement contraints, web-based applications, Service Oriented Architecture, virtualization, and so on. Through a user-friendly *GUI* it allows to design several network architectures in addition to the pre-existing ones. This simulator lacks realism in various aspects since it is mainly designed for academic purposes. For example, the simulator does not consider the possibility of faults in the Data Center, preventing developers from studying the impact of faults on application usage [19] [23].

DartCSim

DartCSim [30] provides a simple *GUI* that allows users to set various simulation parameters, such as the characteristics of the Data Center and the network topology. These parameters can be imported and exported at any level of the simulation and can be set for individual CPUs or the entire Data Center. However, this tool lacks a comprehensive energy model, which prevents developers from implementing strategies aimed at improving Data Center efficiency [19].

DartCSim+

DartCSim+ [31] aims to enhance *CloudSim* by introducing an energy model and a network model, allowing developers to design power-aware scheduling algorithms. However, this simulator does not include a cost model and lacks security features, which prevents developers from analyzing the security aspects of the Data Center [19].

ElasticSim

ElasticSim's [32] main feature is the resources runtime auto-scaling based on stochastic modeling that allows developers to design efficient scheduling algorithms. However, this simulator provides a poor energy consumption model and it cannot simulate security-related experiments [19].

FederatedCloudSim

FederatedCloudSim [33] provides developers with the capability to test several types of cloud federations; this simulator is built upon the functionalities of *CloudSim* and expands them by incorporating Service Level Agreement management, workload generation, event logging, scheduling, and brokering. While the simulator offers comprehensive functionality for modeling and simulating cloud federations, it does not provide specific insights into the energy consumption of each Data Center in the federation [19].

FTCloudSim

FTCloudSim [34] is primarily designed to simulate reliability mechanisms in cloud services, allowing developers to analyze and evaluate the performance of these mechanisms and implementing fault generation services, that enable the generation of faults based on specific probability distributions. However, one drawback of *FTCloudSim* is its simplified energy consumption model, which can impact the assessment of energy-efficient strategies or algorithms [19].

WorkflowSim

WorkflowSim [35] provides a platform for studying the performance impact of different job clustering strategies in a Data Center through the implementation of various workflow scheduling methods. However, *WorkflowSim* has limitations when it comes to simulate data-intensive applications. It does not consider the delays caused by input-output operations, which are crucial in such scenarios. Additionally, the supported fault model in *WorkflowSim* is limited, which can affect the realism of simulations. As a result, the accuracy and realism of simulations involving data-intensive applications may be compromised [19].

CloudReports

CloudReports [36] provides developers with a user-friendly *GUI* that allows them to manage various aspects of the Data Center and access detailed reports on resource utilization, virtual machine allocation, and energy consumption, allowing developers to optimize resource usage and energy efficiency. On the other hand, one limitation of *CloudReports* is

the absence of a security layer: this means that developers cannot explore and evaluate the security characteristics of the Data Center using this simulator [19].

CEPSim

CEPSim [37] models various *Complex Event Processing (CEP)* environments [38] where users can define queries using different proprietary languages and model the execution flow using a directed acyclic graph. The simulator implements various load scheduling algorithms, allowing developers to evaluate queries under different load conditions. However, one limitation of *CEPSim* is that it does not consider network consumption. As a result, the simulator may not provide a comprehensive analysis of energy consumption from a network perspective. Other factors such as network transmission impact on energy consumption are not explicitly taken into account [19].

DynamicCloudSim

DynamicCloudSim [39] is primarily concerned with addressing the instability of computing center parameters that can change during runtime. It specifically introduces failure models for task execution, allowing developers to define the failure rate when conducting experiments in a simulated environment. However, one limitation is that developers do not have the capability to calculate the energy consumption of the experiments accurately due to the restricted nature of the provided energy model; moreover, its failure model is strongly limited [19] [20].

CloudExp

CloudExp [40] provides developers with a simple *GUI* that allows easy configuration of environment parameters and monitoring of their behavior. Specifically, *CloudExp* enables the definition of a Service Level Agreement (SLA) based on parameters such as the number of users, service availability and cost, network performance, and security measures. Additionally, it establishes a specialized framework for cloud computing in mobile device scenarios. However, the energy model used by the simulator is simplistic and not suitable for analyzing energy-aware strategies [19] [22].

CM Cloud

CM Cloud [41] is able to estimate overall energy simulation expenses through a comparison between several providers such as *Google*, *Microsoft* and *Amazon*. However, this simulator lacks a complete communication model, not allowing developers to define specific traffic patterns or investigate the overall impact of the traffic generated by the hosts of the network. Finally, there is no task failure model [19].

MR-CloudSim

MR-CloudSim [42] is based on the *MapReduce* computational model [43] for big data computation, allowing developers to work with data-intensive applications. One limitation is that it does not allow developers to accurately calculate service usage expenses as it does not consider file processing time and cost [19].

UCloud

UCloud [44] is designed to address educational purposes as it simulates cloud for universities, allowing developers to evaluate several policies in the public and private clouds. However, this simulator lacks a support for security policies and a cost model [19].

CloudSimSDN

CloudSimSDN [45] is built for cloud environments based on *Software Defined Networking*, namely a programmatic approach to network management [46]. It is a scalable simulator that allows to manage energy consumption and resource policies as well as investigate several metrics such as performance. Its major drawback is that it models applications through a set of tasks and assumes long packet transmission between VMs, which can impact the granularity of application communication, making it not well-suited for works based on applications. [22] [21].

MDCSim

MDCSim [47] is a simulation platform for multilayer Data Centers analysis that allows developers to investigate applications performance under different loads and tier configurations as it has low simulation overhead. *MDCSim* is able to estimate several parameters, such as throughput, response time and power consumption, so developers can compare different energy policies. However, it lacks simulation realism as it does not provide a complete network model [19] [23].

GDCSim

GDCSim [48] is an open-source, event-based simulator written in *C, C++ and Shell*, as part of the *BlueTool* computer infrastructure project funded by *NSF*. It is designed to analyze green Data Centers as it provides a simulation environment where it is possible to analyze the energy consumption of the Data Center in a simple and accurate way. This tool also takes into account the thermal impact of the Data Center, giving developers the ability to design cooling policies and energy management strategies with a particular attention to the Computational Fluid Dynamics (CFD) that allows to characterize the thermal effects and airflow patterns. However, this simulator does not consider aspects related to the security of the Data Center.

Moreover, it does not allow parallel execution of defined experiments [19]. Architecture of *GDCSim* is shown in figure 2.2. It is composed by four modules:

1. **BlueSim Tool**: a simulation package which integrates various software for HRMs (heat recirculation matrix) array generation;
2. **Input/Output Management**: the interface between the user and the simulator. It takes the following inputs: job trace, Service Level Agreement, management schemes and queuing model;
3. **Resource Management**: module that implements the following algorithms: workload management, power management, cooling management and coordinated workload, power and cooling management;
4. **Simulator**: it provides several modules such as the queuing module, the power module, the thermodynamic module and the cooling module.

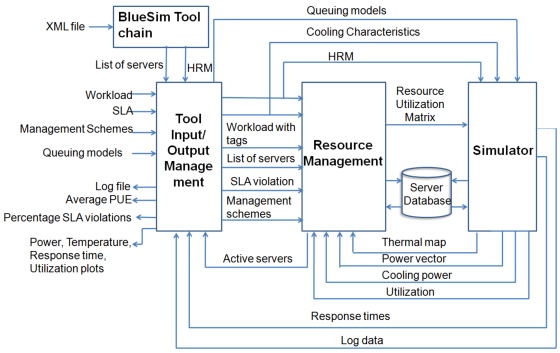

Figure 2.2: Architecture of the GDCSim simulation environment

CloudNetSim

CloudNetSim [49] models end-to-end communication between clients and servers. Its extensibility and modularity make it possible to integrate different modules and evaluate different VM deployment algorithms. This simulator provides a platform that allows developers to investigate resource management for realistic cloud applications. However, *CloudNetSim* implements a poor energy and thermal model that makes energy management algorithms implementation difficult. [19]

CloudNetSim++

CloudNetSim++ [50] is an open source simulator built on the top of *OMNET++*, written in *C++*. It introduces the concept of distributed Data Centers connected with physical network through various topologies. This simulation tool has a modular architecture that allows

researchers to explore different aspects of Data Centers and to extend network topology by adding switches at the aggregation and core levels. *CloudNetSim++* provides a platform to analyze energy consumption during the simulation and an energy-aware scheduler which supplies several techniques such as Dynamic Voltage and Frequency Scaling. *CloudNetSim++* architecture is shown in figure 2.3 and consists of five modules:

- **Pricing Policy Manager**: computes the billing cost for each user request based on the agreement;
- **Cloud Usage Monitor**: analyzes usage patterns;
- **Task Scheduling Selection Module**: determines the scheduling policy;
- **VM Manager**: Determines VM assignments according to the received SLA requests;
- **User Task Scheduler**: receives all incoming user requests and distributes them to the appropriate VMs.

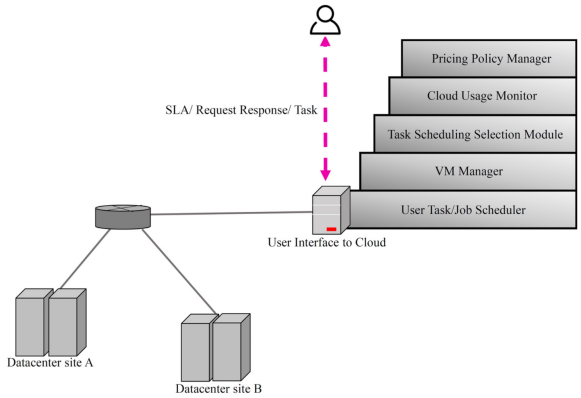

Figure 2.3: Architecture of the CloudNetSim++ simulation environment

GreenCloud

GreenCloud [51] is an open source simulator, written in *C++* and *OTcl* and built on the top of *NS2* that enables researchers to study the energy consumption of Data Centers. *GreenCloud* operates at packet level, as it simulates the behavior of individual network packets and their interactions within the TCP/IP protocol suite that is fully implemented by this simulator. *GreenCloud* aims to provide insights into the energy usage of various components within a Data Center, including servers, switches, and network links and allows users to evaluate the effectiveness of different energy-saving techniques and algorithms, by accurately modeling the energy consumption models. However, despite its utility in energy monitoring, *GreenCloud* has encountered challenges related to scalability because simulation times in *GreenCloud* tend to be relatively long. This simulator also requires significant memory resources, which can pose constraints on the size and complexity of the simulated scenarios.

These scalability limitations can make it difficult to analyze large-scale Data Center networks or evaluate energy-efficient protocols and algorithms in a timely manner. [19] The structure of *GreenCloud* simulation environment mapped onto the three-tier Data Center architecture is shown in figure 2.4.

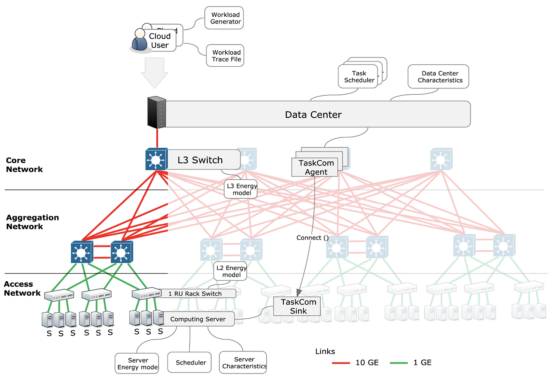

Figure 2.4: Architecture of the GreenCloud simulation environment

iCanCloud

iCanCloud [52] offers a dedicated *GUI* that allows users to configure the computing center and obtain graphical reports. *iCanCloud* supports the implementation of various brokering strategies, providing the flexibility to define different brokers connecting users and the computing center. However, this simulator does not address aspects related to the energy model and security [19].

secCloudSim

secCloudSim [53] is an extension of *iCanCloud* that implements simple security mechanisms lacking in its predecessor. Among the various security features, it includes an authentication protocol and *Access Control List (ACL)* that allows associating specific privileges with each authenticated user. *secCloudSim* provides a framework where researchers can develop security characteristics such as encryption, decryption, encapsulation, authentication, and privacy. However, the security mechanisms provided by *secCloudSim* are not very advanced, limiting researchers from studying infrastructure vulnerabilities [19].

GroudSim

GroudSim [54] is a *Java-based* simulator primarily used in the context of scientific applications. Developers can import *ASKALON* experiments [55] into this simulator to conduct simulations of real applications. However, this simulator lacks realism as it does not allow the configuration of a realistic network topology and does not scale effectively [19].

CloudSched

CloudSched [56] implements various energy-efficient algorithms and resource scheduling strategies for physical and virtual machines to avoid bottlenecks. It allows developers to define custom resource scheduling algorithms as needed. However, *CloudSched* does not consider task failures, making it unable to implement fault-tolerance strategies [19].

SimIC

SimIC [57] focuses mainly on the heterogeneity of environments in which experiments are executed. Developers can define inter-cloud scheduling algorithms based on various distributed parameters. However, the main limitations of this simulator are that it does not allow investigation of energy consumption, traffic controls, and congestions [19].

SPECI

SPECI [58] models aspects related to the scalability and performance of computing centers, allowing developers to monitor the system's behavior by varying its architecture. The simulator also enables investigation of inconsistencies that may arise in the event of failures. However, it does not model changes in the performance of virtual machines during execution, despite this factor being particularly relevant in real-world scenarios [19].

SCORE

SCORE [59] is well-suited for defining energy-aware scheduling algorithms, such as mechanisms for shutting down and powering on resources. However, it lacks a security module that would allow developers to investigate fundamental aspects of the computing center related to security [19].

GAME-SCORE

The *GAME-SCORE* simulator [60] is an extension of the *SCORE* simulator written in *Scala*, which uses a combination of discrete-event and multi-agent simulation approaches. Its primary purpose is to simulate energy-efficient IaaS in cloud environments. This simulation tool provides the flexibility to dynamically select energy-efficiency policies from a range of options, allowing for the shutdown of idle machines during runtime. As a practical example, it introduces an algorithm based on the Stackelberg Game that utilizes this capability. However, it's important to note that this simulation tool can accommodate other strategies as well. These strategies can involve the dynamic switching between various energy-efficiency policies and scheduling algorithms. The versatility of this tool enables the implementation of different approaches to optimize energy consumption in simulated environments. Architecture of *GAME-SCORE* is shown in figure 2.5. It is composed of two modules, described below:

- **Core Simulator Module**: the module responsible for executing the experiments composed of a workload generator, a core engine and a scheduling module;
- **Energy-Efficiency Module**: the module responsible for the implementation of the energy-efficiency policies.

It is additionally composed of a special module that implements the Stackelberg Game in order to dynamically switch between energy-efficiency policies.

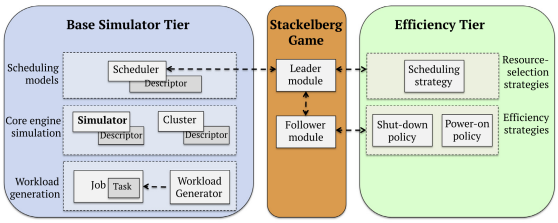

Figure 2.5: GAME-SCORE architecture

DISSECT-CF

DISSECT-CF [61] is an event-based open source simulator written in Java which aims to offer energy-aware scheduling for infrastructure clouds. In contrast to other recently developed simulators, *DISSECT-CF* takes a unique approach by separating energy modeling from resource simulation. This enables the inclusion of energy consumption that may not be directly related to the utilization of Data Center resources. By decoupling energy modeling, *DISSECT-CF* allows for more comprehensive energy and power modeling, facilitating the analysis of sophisticated energy-aware algorithms in areas such as virtual machine placement and task scheduling. Architecture of *DISSECT-CF* is shown in figure 2.6 and it is composed of five modules decribed below:

- **Event system**: this component serves as the time reference for simulations;
- **Unified resource sharing**: this subsystem establishes a flexible and lightweight foundation for sharing low-level computing resources, such as CPU and I/O;
- **Energy modeling**: *DISSECT-CF* includes components that allow simulator developers to monitor and analyze energy usage patterns of specific simulated resources, such as network links and disks;
- **Infrastructure simulation**: these components govern the behavior and interactions of various elements within the infrastructure, for example the virtual machines;
- **Infrastructure management**: this subsystem offers a user interface, and encompasses higher-level functionalities like virtual machine schedulers of infrastructure clouds.

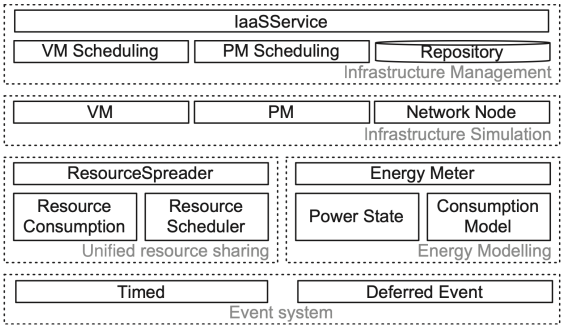

Figure 2.6: Architecture of the DISSECT-CF simulation environment

ICARO

ICARO [62] aims to analyze changes in the load of a computing center when the load structure dynamically varies at runtime [22].

SmartSim

SmartSim [63] simulates the behavior of mobile devices and resource-intensive applications [22]. With *SmartSim*, it is possible to model both system components and their behavior in terms of resource allocation and management [20].

PICS

PICS [64] is a simulator designed to evaluate the cost and performance of public *IaaS* (Infrastructure as a Service) while considering factors related to resource management and job scheduling. However, this simulator lacks a model for communication costs [20].

2.4 Choice of simulation tool

As discussed earlier, this research work requires accurate insights about the energy consumption of the Data Center components. This need led to the selection of the five tools between the ones analyzed in the previous section, that are most focused on analyzing energy consumption, namely: *CloudSim*, *DISSECT-CF*, *GreenCloud*, *GDCSim* and *GAME-SCORE*. Table 2.1 shows a summary of the selected simulators. As it can be observed, *GDCSim* is stated to be not available. According to the original paper about *GDCSim* [48], it was mentioned that the simulator would have been available on the *BlueTool* platform once it was ready, however, this platform is no longer accessible. The rest of the information has been gathered from the original papers related to the simulators, namely [45], [61], [51], [48], [65] and [60].

Table 2.1: Simulation tools summary

Simulator	Simulation type	Language	Availability
CloudSim	Event-based	Java	Open source
DISSECT-CF	Event-based	Java	Open source
GreenCloud	Packet-level	C++, oTcl	Open source
GDCSim	Event-based	C, C++, Shell	Not available
GAME-SCORE	Event-based and multi-agent	Scala, Java, Python	Open source

2.4.1 Comparison parameters

The selected simulators have been evaluated based on several parameters, including:

- **Last update**: it provides an indication of how well-supported and updated the simulator is;
- **Popularity**: it gives an idea of how popular the simulator is;
- **Availability**: it indicates the availability of the simulator;
- **Granularity**: it suggests the level of detail at which the elements of the Data Center and simulation can be defined;
- **Performance profile**: it provides an indication of the performance of the simulator.

2.4.2 Simulators evaluation

The following sections describe the simulators from the perspective of the evaluation parameters. Each feature of the simulators has been assigned a score on a scale from 1 to 5, in order to highlight strengths and weaknesses. The evaluation results are presented in the radar chart shown in figure 2.7.

Last update

CloudSim's, *GAME-SCORE*'s and *DISSECT-CF*'s last update (2020, 2018 and 2023 respectively) were gathered from the latest commits on their GitHub repositories[1]. *GreenCloud*'s latest files modification year is 2016 (the project has been download from the simulator platform[2]). Since *GDCSim* is not available, it is reasonable to assume that its last update corresponds to the year of the last paper published about it [65], which is 2014.

[1] CloudSim: https://github.com/Cloudslab/cloudsim,
GAME-SCORE: https://github.com/DamianUS/game-score
DISSECT-CF: https://github.com/kecskemeti/dissect-cf

[2] http://greencloud.gforge.uni.lu/ftp/greencloud-v2.1.2.tar.gz

Popularity

Popularity has been evaluated through the citation numbers that have been obtained from the *SCOPUS* platform:

- CloudSim: 2043 citations;
- DISSECT-CF: 23 citations;
- GreenCloud: 31 citations;
- GDCSim: 3 citations;
- GAME-SCORE: 1 citation;

Availability

Tools availability is reported in table 2.1.

Granularity

According to the original papers about the analyzed simulators:

- *CloudSim* allows to create energy-conscious provisioning policies by overriding the method *getPower()* of the abstract class *PowerModel* whose input parameter is the utilization metric for Cloud host and return parameter is the power consumption value. *CloudSim* also provides a VM Allocation controller component (*VmAllocationPolicy*) that exposes some custom methods which can be used by developers in order to implement new policies based on several optimization goals, and a VM Scheduler component which can be extended to test several allocation policies;
- *DISSECT-CF* allows developers to define various energy consumption models based on simulation entities' power state. In order to compute energy consumption accurately, this simulator uses several meters and allows developers to define an aggregation function that addresses the dependency between two metered components (e.g., a virtual machine and the physical one where it is hosted);
- *GreenCloud* fully implements the *TCP/IP* protocol. This simulator is built on top of the *NS2* simulator, allowing customization of network topology and enabling work at the packet-level to implement specific traffic patterns and model real-world scenarios. Furthermore, thanks to its open-source nature, it allows the implementation of various energy management and workload scheduling algorithms;
- *GDCSim* architecture was designed to be modular and extensible in order to easily plug new components into the simulator and to perform various analysis under different

physical configurations. For example *GDCSim* allows to replace the cooling model with a user-defined one and to add new power consumption and resource management models;

- *GAME-SCORE*'s main aim is to dynamically apply energy policies during simulations by enabling developers to dynamically choose between a catalog of energy-efficiency policies that shut-down idle machines in runtime.

Performance profile

- *CloudSim*'s authors found out that the time to instantiate an experiment setup with 1 million hosts is around 12 s. Moreover, they observed that the total memory usage never grew beyond 320 MB even for larger system sizes;
- *DISSECT-CF*'s authors stated that the simulator scales comparably to other state-of-the-art simulators, such as *CloudSim*, since it never drops below linear scaling;
- Since *GreenCloud* is a packet-level simulator, its simulation time is significantly higher compared to the other state-of-the-art simulators. [22] states that *GreenCloud* simulation time is on the order of minutes;
- The authors of *GDCSim* conducted several large-scale experiments to validate this simulator. They discovered that each *GDCSim* simulation, which involved *HRM* generation taking 775 minutes, required less than 1 minute to complete. In comparison, corresponding *CFD* simulations took 30 minutes. Therefore, despite the initial cost of generating the *HRM*, *GDCSim* outperforms *CFD* simulations in terms of runtime in the long run [66];
- This simulator is based on Google cloud facilities, so the performance could scale well with the growing of the simulated system.

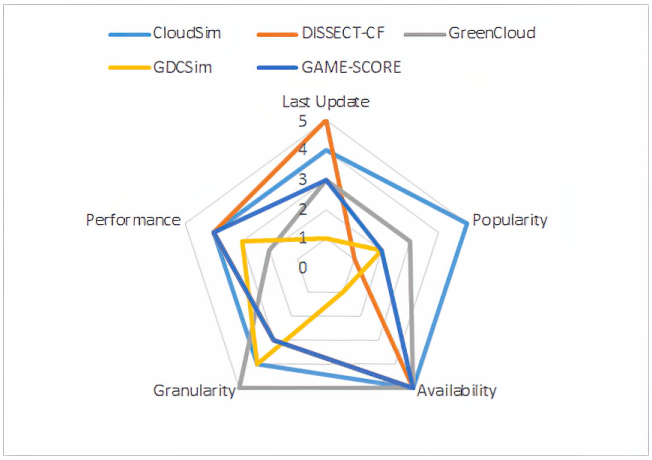

Figure 2.7: Simulators scoring

CHAPTER 3

GreenCloud Simulator Overview

In this chapter the reason for the selection of *GreenCloud* simulator as the reference tool for the simulations will be explained. Then several aspects about *GreenCloud* will be covered, including its usage, its source code and the output files of the simulations.

3.1 Selection of GreenCloud

Based on the comparisons made among the different analyzed simulators, each of them has its strengths and weaknesses. However, for the needs required in the study addressed by this thesis work, it is essential to prioritize aspects related to the energy consumption of the computing center and the accuracy of simulations. From the conducted comparisons, it is evident that *GreenCloud* is the simulator that accurately considers these aspects as it is built on top of *NS2* simulator and fully implements the *TCP/IP* protocol. Moreover, *GreenCloud* offers its users a set of features related to the simulation and to the energy consumption management. In particular, it is possible to choose between several pre-implemented Data Center architectures and energy models as well as to customize them. Furthermore *GreenCloud* provides various workload scheduling and power saving models, allowing programmers to implement new ones. Despite *GreenCloud* simulation times tend to be high, for the purposes of this study it is reasonable to prioritize granularity over performance. Therefore, the study will continue using *GreenCloud* as the reference tool for the simulations to be conducted. The following sections provide a description of various aspects of *GreenCloud* based on the work by *Kliazovich et al* [51].

3.2 GreenCloud project

In the following subsections, two aspects of the *GreenCloud* simulator project will be covered, namely the simulator source code structure and the description of the output files that are generated upon simulation completion.

3.2.1 Simulator source code

The *GreenCloud* source code is organized into three directories:*dashboard*, *ns2/greencloud*, and *scripts*:

- *dashboard* contains *HTML*, *CSS*, and *JS* code related to the simulation report page;
- *ns2/greencloud* contains *C++* code that implements the core functionalities of the simulator, the energy models of the data center components, scheduling algorithms, and energy-saving models;
- *scripts* includes various *TCL* and *Bash* scripts that facilitate the modification of simulation parameters and allows to create simulation reports.

Since the scripts in the *scripts* directory require access to various variables within the *C++* classes to modify and read them for constructing logs, each class associates each variable with a symbolic name using the *bind* method. This allows the scripts to access these variables. For instance, the class *DcHostClass* defined in *ns2/greencloud/dchost.cc* exposes several variables

related to a Data Center host, including the current load, current energy consumption, and the total consumed energy.

3.2.2 Simulation input parameters

The executed simulations are highly customizable by users. Below are descriptions of some of the most important parameters along with the files in which they can be modified:

- **SCHEDULER**: specifies the scheduling algorithm used. The available options are *Green RoundRobin*, *Random*, *HEROS*, *RandDENS* and *BestDENS*. This parameter can be set at line 9 of the run.sh file;
- **TOPOLOGY**: indicates the type of data center architecture. The possible choices are *three-tier*, *three-tier high-speed*, *three-tier debug*, *three-tier heterogenous debug* and *three-tier heterogenous*. Although *GreenCloud*-related literature mentions a two-tier architecture implementation, such implementation isn't found in the simulator's source code. This parameter can be set at line 10 of the run.sh file.
- **top(NCloudUser)**: specifies the number of data center users. It can be set at line 11 of the user.tcl script;
- **serv(eDVFS_enabled)** and **serv(eDNS_enabled)**: indicate the usage of *DVFS* and *DNS* in Data Center servers. These can be set at lines 19 and 20 of the setup_params.tcl script;
- **switches(eDVFS_enabled)** and **switches(eDNS_enabled)**: indicate the usage of *DVFS* and *DNS* in data center switches. These can be set at lines 28 and 29 of the setup_params.tcl script;
- **task(mips)**, **task(memory)**, **task(storage)**, **task(size)**, **task(duration)**, **task(outputsize)**, and **task(intercom)**: Indicate various properties of tasks to be executed. In particular, *task(mips)* signifies the required computation in MIPS, *task(memory)* indicates required RAM in bytes, *task(storage)* denotes required storage memory in bytes, *task(duration)* specifies the computing deadline in seconds, *task(outputsize)* represents the length of output upon task completion in bytes, and *task(intercom)* describes the size of inter-task communication. These parameters can be set at lines 67-82 of the setup_params.tcl script.

3.2.3 Simulation output files

At the end of a simulation, GreenCloud generates various report files in TR format, that provide a detailed exploration of different aspects of the conducted simulations. Specifically, the following files are produced:

- *dcLoad.tr:* a time-stamped log of the Data Center load throughout the entire simulation;

- *dcLoadMem.tr:* a time-stamped log of the Data Center memory load throughout the entire simulation;
- *dcLoadStor.tr:* a time-stamped log of the Data Center storage load throughout the entire simulation;
- *dcPower.tr:* a time-stamped log of the power consumed by servers;
- *dcServLoad.tr:* it reports the average load for each server during the simulation;
- *dcServLoadMem.tr:* it reports the average memory load for each server during the simulation;
- *dcServLoadStor.tr:* it reports the average storage load for each server during the simulation;
- *dcServTasks.tr:* it reports the average number of submitted tasks per server;
- *dcServTasksFailed.tr:* it reports the average number of failed tasks per server;
- *dcVmLoad.tr:* it reports the average load for each VM during the simulation;
- *dcVmLoadMem.tr:* it reports the average memory load for each VM during the simulation;
- *dcVmLoadStor.tr:* it reports the average storage load for each VM during the simulation;
- *dcVmTasks.tr:* it reports the average number of submitted tasks for each VM during the simulation;
- *dcVmTasksFailed.tr:* it reports the average number of failed tasks for each VM during the simulation;
- *eAccessSwitches.tr:* it reports the amount of energy consumed by each access layer switch during the simulation;
- *eAggrSwitches.tr:* it reports the amount of energy consumed by each aggregation layer switch during the simulation;
- *eCoreSwitches.tr:* it reports the amount of energy consumed by each core layer switch during the simulation;
- *energySummary.tr:* it reports the amount of energy consumed by all switches during the simulation;
- *eServers.tr:* it reports the amount of energy consumed by each server during the simulation;

- *link_C1C2_load.tr:* it reports the average load on network links connecting core switches to aggregation switches in the downlink;
- *link_C2C1_load.tr:* it reports the average load on network links connecting aggregation switches to core switches in the uplink;
- *link_C2C3_load.tr:* it reports the average load on network links connecting aggregation switches to Top-of-Rack (ToR) switches in the downlink;
- *link_C3C2_load.tr:* it reports the average load on network links connecting Top-of-Rack (ToR) switches to aggregation switches in the uplink;
- *link_C3H_load.tr:* it reports the average load on network links connecting computing servers to thier Top-of-Rack (ToR) switches in the downlink;
- *link_HC3_load_time.tr:* it reports the average load on network links connecting computing servers to thier Top-of-Rack (ToR) switches in the uplink;
- *link_HC3_load.tr:* it reports the load dynamics of a single link connecting a given server to its Top-of-Rack (ToR) switch in the uplink;
- *loadSummary.tr:* it reports the Data Center average load;
- *parameters.tr:* it contains a summary of the input simulation parameters;
- *queue_C3C2-0_pkts.tr:* it reports the queue size in packets on network links connecting Top-of-Rack (ToR) switches to aggregation switches in the uplink;
- *queue_HC3_pkts_avg.tr:* it reports the average queue size in packets on network links connecting computing servers to their Top-of-Rack (ToR) switches in the uplink;
- *queue_HC3_pkts_time.tr:* it reports the queue load dynamics in packets of a single link connecting a given server to its Top-of-Rack (ToR) switch in the uplink;
- *serv_load_time.tr:* a time-stamped log of the servers load throughout the entire simulation;
- *simulation.tr:* it reports the simulation duration;
- *taskSummary.tr:* it reports a summary of executed and failed tasks;
- *users.tr:* it reports several Data Center users statistics.

The information about these files is obtained from the descriptions provided in the *dashboard/graph-definitions.js* file. To access a more immediate summary of the conducted simulations, the *dashboard/dashboard.html* file can be viewed using a web browser. This webpage presents essential simulation details along with various charts, as illustrated in figures 3.1 and 3.2.

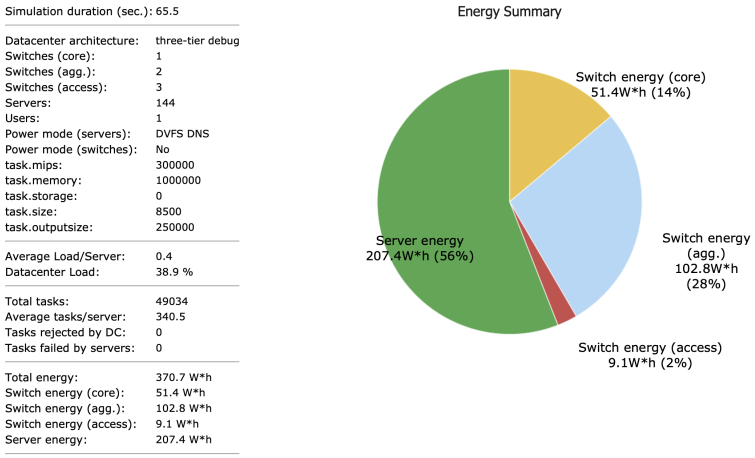

Figure 3.1: GreenCloud energy consumption chart

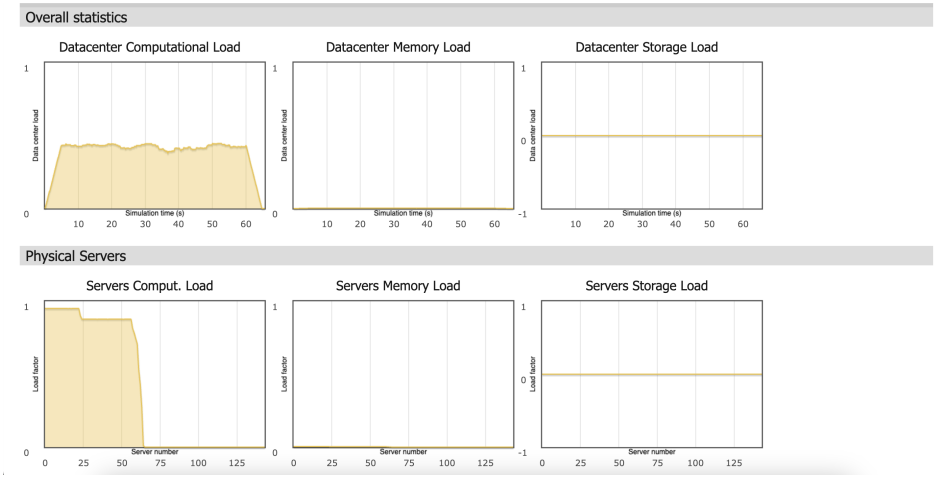

Figure 3.2: GreenCloud overall statistics charts (partial)

3.3 Available Data Center architectures

As mentioned in the previous section, *GreenCloud* provides several Data Centers architectures. The implemented architectures consist of various components, described as follows:

- **Servers:** single core nodes with a fixed processing power limit expressed in *MIPS* (million instructions per second) or *FLOPS* (floating point operations per second) that are responsible for task execution. These components are organized in racks and the architecture includes the presence of a Top-of-Rack switch that connects them to the access layer of the architecture;

- **Switches and links:** they implement the interconnection between the Servers in the Data Center. The type and the quality of these devices influences the transmission rate, anyway the costs of such devices need to be taken into account. Switches usually support either 1 *GE (Gigabit Ethernet)* or 10 *GE* as transmission rates, while links usually support 10 *Mb/s*, 100 *Mb/s*, and 1 *Gb/s* as transmission rates;

- **Workloads:** the representation of tasks to be executed that consist of two components, computational and communicational. The computational part specifies the required amount of computing resources, measured in *MIPS* or *FLOPS*, and the duration of resource allocation. On the other hand, the communicational component of the workload entails the quantity and dimensions of data transfers essential before, during, and after the workload execution.

The following subsections provide an overview of the available Data Center architectures within the *GreenCloud* simulator. Since, as mentioned in the previous chapter, this simulator is based on *NS2*, it is possible to customize the *Data Center* architecture.

Two-tier Data Center architecture

Two-tier architecture is shown in figure 3.3. This architecture consists of an *Access Network* layer where rack switches group several computing Servers through 1 *GE* links and a *Core Network* layer where L3 switches provide full mesh connectivity through 10 *GE* links. This type of architecture supports up to 5500 nodes.

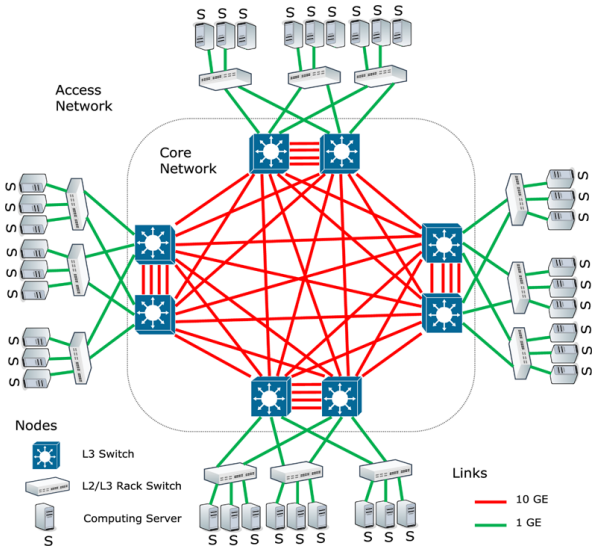

Figure 3.3: GreenCloud two-tier architecture

Three-tier Data Center architecture

Three-tier architecture is shown in figure 3.4. This architecture consists of an *Access Network* layer where rack switches group several computing Servers through 1 *GE* links, an *Aggregation Network* and a *Core Network* where L3 switches provide full mesh connectivity through 10 *GE* links. This type of architecture is the most commonly used nowadays.

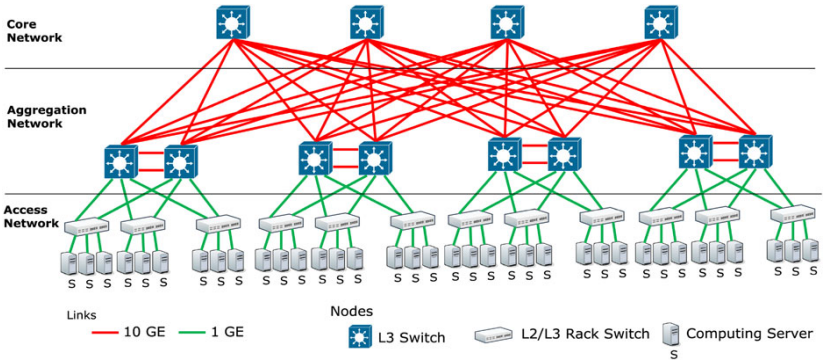

Figure 3.4: GreenCloud three-tier architecture

Three-tier high-speed Data Center architecture

Three-tier high-speed architecture is shown in figure 3.5. This architecture is analogous to the three-tier architecture except it employs 100 *GE* links in the *Aggregation Network* and *Core Network* instead of 10 *GE* ones. Moreover, it consists of only two L3 Switches in the *Core Network* instead of 4.

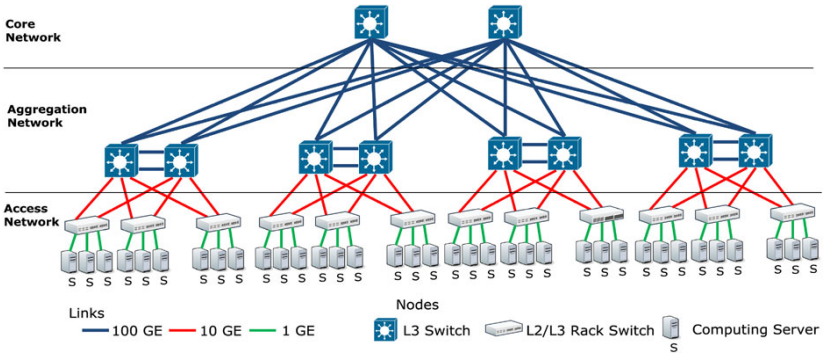

Figure 3.5: GreenCloud three-tier high-speed architecture

3.4 Available power saving models

Two power saving models are available within the *GreenCloud* simulator, namely *Dynamic Voltage and Frequency Scaling (DVFS)* which makes the energy consumption of servers proportional to their operating frequency and *Dynamic Network Shutdown (DNS)* which places idle servers into a sleep mode. The power savings achieved through these techniques are not negligible. According to [51] the use of *DVFS* results in a power savings of 4%, the use of *DNS* leads to a savings of 63%, and the combined use of both techniques results in a savings of 65%. It is evident that the most significant contribution comes from the utilization of *DNS*, as an active server consumes a considerable amount of energy that does not scale with the operating frequency.

3.5 Energy model of servers, switches, memory and disks

Since *GreenCloud* aims to be suitable for energy management, the authors provided the energy models related to the components of the *Data Center* architecture. The following subsections present the energy models described in [51] for each described component.

Servers

Assuming that f is the operating frequency of the CPU, P_fixed is the fixed energy consumption that does not scale with f and P_f is the CPU-dependent energy consumption, the total energy consumption of a server is calculated as follows (equation 3.5.1):

$$P_{server} = P_{fixed} + P_f \cdot f^3 \tag{3.5.1}$$

Switches

The power consumption model for switches is illustrated in [67] and is described by the following equation (equation 3.5.2):

$$P_{switch} = P_{chassis} + n_c P_{linecard} + \sum_{r=1}^{R} n_p^r P_p^r u_p^r, \tag{3.5.2}$$

where $P_{chassis}$ is the energy consumed by the switch hardware, n_c is the number of line cards, n_p^r is the number of ports running at rate r, $P_{linecard}$ is the energy consumed by a linecard, P_p^r is the energy consumed by a port running at rate r and u_p^r represents a port utilization, defined as follows (equation 3.5.3):

$$u_p = \frac{1}{T}\int_t^{t+T} \frac{B_p(t)}{C_p} dt = \frac{1}{T \cdot C_p}\int_t^{t+T} B_p(t) dt, \tag{3.5.3}$$

where $B_p(t)$ is an instantaneous throughput at the port's link at the time t, C_p is the link capacity, and T is a measurement interval.

Memory and disks

GreenCloud implements a simple energy model for memory and disks as it allows to choose between the idle and the maximum power of these components resulting respectively in the minimum and the maximum consumption.

3.6 Workload scheduling algorithms

The choice of the scheduling algorithm significantly affects the energy consumption of the data center. As previously mentioned, *GreenCloud* provides various scheduling algorithms

and, due to its open-source nature, allows developers to create new ones. This section provides an overview of the algorithms available within the simulator and conducts a comparison among them.

3.6.1 Algorithms description

The *GreenCloud* simulator provides the following scheduling algorithms:

- **Green:** this energy-aware scheduler groups the workload onto the fewest number of computing servers possible;
- **RoundRobin:** it cyclically distributes the tasks across a set of servers for a fixed time slice. Once the time elapses, this scheduler moves to the next task;
- **Random:** it allocates tasks by randomly choosing servers;
- **RAND-DENS and BEST-DENS:** they aim to reduce the energy usage of the Data Center by choosing the most suitable computing resources for tasks execution. This selection is based on two parameters, namely the load level and the communication potential of Data Center components, defined as the available end-to-end bandwidth that specific servers or groups of servers can access through the Data Center [68];
- **HEROS:** it allocates tasks to a server based on a score computed through a decision function based on two subfunctions, namely the *server selection function* and the *communication potential function*. The server selection function is based on the idea that the power usage of different hardware can vary a lot in terms of how much energy it uses and how it behaves, while the communication potential function is similar to the *DENS* communication potential function but it uses the actual link load instead of the queue buffer size [69].

3.6.2 Algorithms comparison

In order to evaluate the performance of these algorithms, several simulations were performed using *GreenCloud*. These simulations model the scenario where a specific amount of computational work is requested by a certain number of users and fulfilled by a specific number of servers within a given timeframe. In particular, all simulations have been executed with the same parameters described below:

- **Architecture:** a three-tier debug architecture was utilized, differing only from the three-tier architecture described in Section 3.3 in terms of the number of components comprising the data center. Specifically, 144 servers, 3 switches at the access layer, 2 switches at the aggregation layer, and one switch at the core layer were employed;
- **Number of users:** due to simulation time constraints, only two users were instantiated;

- **Processing power (MIPS):** the total processing power required by the tasks to be executed amounts to 600000 *MIPS*;
- **Simulation time:** the simulation models a scenario in which the servers process client requests for 60 seconds;
- **Adopted power saving models:** the power saving model adopted for switches is *DVFS*, while for servers both *DVFS* and *DNS* were employed.

The reports provided by *GreenCloud* regarding these simulations can be found on the *GitHub* repository related to this research work, where each directory contains the files generated for each scheduling algorithm. Furthermore, an *xlsx* file has been generated, containing a time-stamped log of the overall energy consumption and *PUE* of the Data Center throughout the entire simulation, for each described scheduling algorithm. In order to calculate the time-stamped log of the energy consumption, it was necessary to make specific modifications to the simulator code, which will be adequately detailed in Section 4.1. Additionally, computing the Power Usage Effectiveness (PUE) involved the utilization of an external tool, discussed further in Subsection 4.2.4. In order to provide an immediate understanding of the behavior of these aforementioned algorithms, three charts (one for energy consumption, one for *PUE* and one that summarizes the total energy consumed during the simulation) have been produced (figures 3.6, 3.7 and 3.8).

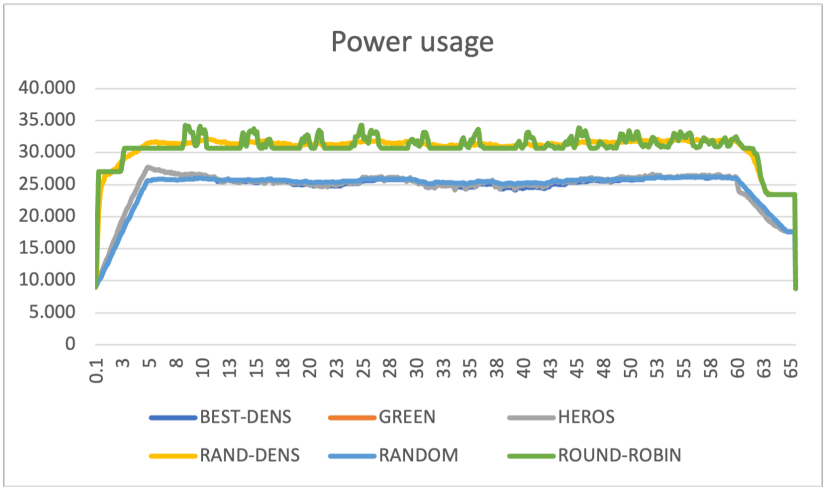

Figure 3.6: Scheduling algorithms energy consumption

As indicated by the charts, it is evident that the algorithms that perform better are *BEST-DENS*, *GREEN*, and *HEROS* as they manage to achieve an energy-saving of approximately 22% compared to the other three algorithms.

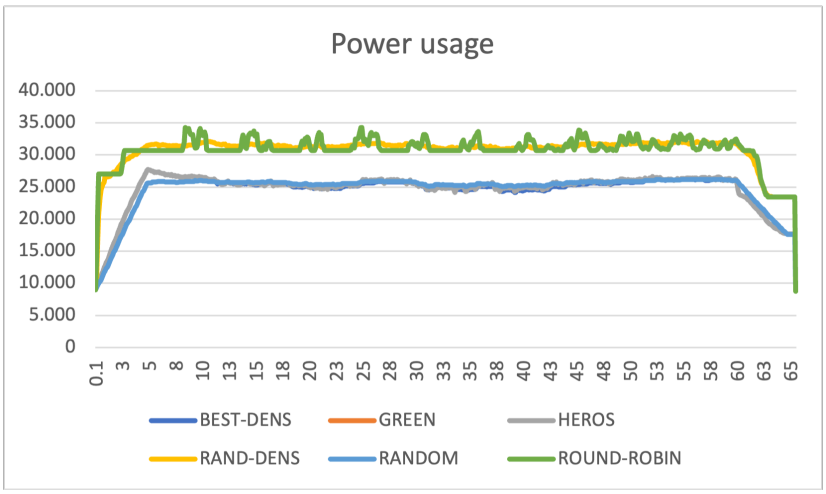

Figure 3.7: Scheduling algorithms PUE

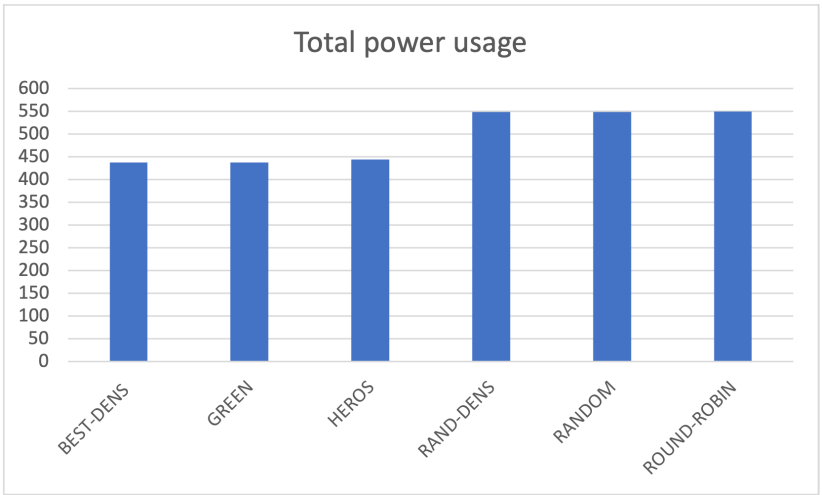

Figure 3.8: Scheduling algorithms PUE

CHAPTER 4

Research Study

In this chapter the research study process will be illustrated, starting from the changes implemented in GreenCloud in order to make it suitable for this work and then describing the performed simulations and the *PUE* calculation. Finally, the chapter ends with a discussion of the achieved results, providing an idea about how the variation of *PUE* can be a suitable parameter for attacks detection.

4.1 Changes implemented in GreenCloud

The modifications made to the *GreenCloud* simulator's code aim to achieve a time-stamped log of the instantaneous overall energy consumption of the Data Center. While exploring the log files, it became apparent that such a report was missing. Therefore, it was necessary to construct it using the data provided by the simulator. As discussed in Subsection 3.2.1, in order to enable scripts to access the variables provided by C++ classes and create simulation reports, a binding between these variables and symbolic names within the classes is necessary. Since the *SwitchEnergyModel* class defined in the file ns2/greencloud/switchenergymodel.cc and responsible for modeling switch energy consumption, does not make the *eCurrentRate_* variable, representing instantaneous switch energy consumption, available to scripts, the initial modification made was to bind the *eCurrentRate_* variable to the symbolic name *eCurrentRate_*. Once that was done, within the *record_graphs* procedure (called every 100 milliseconds by the simulator to update data in report files) defined in the scripts/record.tcl script, an instantaneous energy consumption calculation for Data Center components was implemented. This was achieved using the *eCurrentRate_* variables related to servers and switches (lines 41-74). Given that this calculation occurs every 100 milliseconds, an instruction was added (line 79) to store the instantaneous energy consumption and its corresponding timestamp in a specifically created file. This log file is fundamental for the conducted research study as it provides an insight of the consumed energy throughout the whole simulation and allows to calculate *PUE* at specific moments facilitating the study of its fluctuations.

4.2 Data Center Design

In order to perform extract accurate data from simulations, it is necessary to understand various aspects of the Data Center where these simulations are carried out. The goal of the following subsections is to describe its key aspects, such as architecture and component specifications. Furthermore, aspects related to configuring power and cooling parameters for the Data Center will be addressed, and the methods used to estimate the maximum Data Center capacity will be illustrated.

4.2.1 Architecture

One of the most popular architectures in the field of Data Centers is the three-tier architecture, explained in Section 3.3. *GreenCloud* offers this architecture, comprising 1536 servers, 64 switches at the access layer, 16 switches at the aggregation layer, and 8 switches at the core layer. However, this type of architecture comes with significant overhead for simulations, since, as confirmed by the experiments conducted, the simulation of a 30-minute scenario can take up to 24 hours to complete. In order to maintain reasonable simulation times, the "debug" variant of this architecture was employed. This variant comprises 144 servers, 3

switches at the access layer, 2 switches at the aggregation layer, and 1 switch at the core layer.

4.2.2 Components specifications

Inspecting *GreenCloud*'s source code reveals that servers in the three-tier debug architecture feature a 4-core CPU, 8GB RAM, and a 500GB hard drive. Each CPU core provides a computing power of 150015 *MIPS* resulting in a total of 600060 *MIPS* per server that consumes 201 W under maximum load and 50% of the maximum (100.5 W) during idle periods. Furthermore, the energy profile of the core and aggregation layer switches is characterized by a power consumption of 1558 W for chassis, 1212 W for line cards, and 27 W for ports, while for the access layer switches, the power consumption amounts to 146 W for chassis and 0.42 W for ports. Finally, in the access layer, 1 GE links are employed, while in the aggregation and core layers, 10 GE links are used. These informations have been gathered from the source code of the *GreenCloud* simulator, available in its modified version as described in Section 4.1, on the *GitHub* repository related to this research work.

4.2.3 IT capacity calculation

The calculation of the energy required by the Data Center was carried out by considering the maximum consumption of its components. In particular, as described in Section 4.2.2, the maximum consumption of a single server is 201 W when its CPU operates at 100% load. Since the architecture consists of 144 servers, the maximum amount of energy consumed by the servers totals 28944 W. Concerning the switches, on the other hand, the documentation of the simulator and the source code did not provide a clear indication of their maximum achievable energy consumption. To overcome this issue, multiple simulations were carried out with all energy-saving mechanisms disabled, ensuring that switches operated at their highest power levels. These simulations demonstrated that core and aggregation layer switches consume 2824 W, while access layer switches consume 166 W. Therefore, the 3 switches in the access layer collectively consume 498 W, the 2 switches in the aggregation layer consume 5648 W, and the single switch in the core layer consumes 2824 W, resulting in a total of 8970 W consumed by all switches in the architecture. By summing up the consumption of servers and switches, the total power consumption amounts to 37914 W.

4.2.4 Power and cooling parameters

Since *GreenCloud* does not incorporate the *PUE* calculation feature, an online tool provided by *Schneider Electric SE* was employed for this purpose. This tool requires several details related to the cooling system for *PUE* calculation, configured as follows:

- **Data Center IT Capacity:** as mentioned in Section 4.2.3, the total energy consumption of the Data Center is approximately 37914 W. Designing the Data Center to accommodate

and cool an energy load 20% higher than the maximum capacity, an IT capacity of 50 kW was set;

- **Electricity Cost per kWh:** since this research study is not aimed at investigating Data Centers costs, this parameter is not relevant;
- **UPS System:** typical;
- **Power Redundancy:** dual path power;
- **Cooling System:** air cooled;
- **Chiller:** N/A;
- **Air Distribution:** perimeter cooling;
- **CRAC/CRAH Redundancy:** single path CRAC/CRAH;
- **Heat Rejection Redundancy:** single path heat rejection;
- **Water-side Economizer Time:** N/A;
- **Design Details:** Standby Generator, Optimized Rack Layout, Blanking Panels, UPS in Eco Mode, Energy Efficient Lighting, Coordinated CRAC/CRAH.

4.3 Attack scenario modeling

Chapter 1 has described the attack scenarios studied from a theoretical point of view and has analyzed real-world implications and threats for a Data Center. The aim of this section is to provide an overview about how the described attack scenarios have been modeled within the context of the performed simulations.

4.3.1 DoS scenario

The *DoS* scenario was simulated by dynamically increasing the load during the simulation. Specifically, the expected load was set at 40% of the available *MIPS* in the Data Center, while anything beyond 80% was considered indicative of a *DoS* attack.

4.3.2 Cooling system attack scenario

There are several attack scenarios targeting the cooling system of the Data Center; however, for this research work, focus has been placed on a specific one. In Data Center design, various cooling solutions can be chosen, including perimeter cooling and close-coupled cooling. In the former, a computer room air conditioner *(CRAC)* cools the entire room, while in the latter, the air conditioning units are placed as close as possible to the heat source. The primary

advantage of the close-coupled approach is that cooling occurs precisely where it is needed, preventing the formation of hot spots [70]. Considering that Data Centers typically employ a combination of both cooling approaches, a potential attack could target the close-coupled cooling system. Since the tool used for Data Center configuration allows analysis of scenarios with both close-coupled and perimeter cooling setups, it is feasible to pratically simulate such a scenario.

4.4 PUE formula approximation

As previously mentioned, an online calculator provided by Schneider Electric SE was used for calculating the PUE and configuring the Data Center. Due to the absence of an exact formula for PUE calculation, a curve fitting technique was employed. To derive the formula for PUE calculation, it is necessary to consider two scenarios: one where the Data Center configuration aligns with what is described in the Subsection 4.2.4, and another where the cooling system attack described in paragraph 4.5.4 occurs. In the first case, after configuring the tool with the parameters specified in Subsection 4.2.4, 17 points were selected on the graph representing various load intensities (and consequently different power values in kW), from which the PUE was extrapolated. Subsequently, Excel's curve fitting feature was utilized, using the "Power" function, as it best matched the PUE's shape and aligned with the collected data points. The resulting formula obtained was: $y = 725.31x^{-0.575}$ and the corresponding chart is illustrated in figure 4.1.

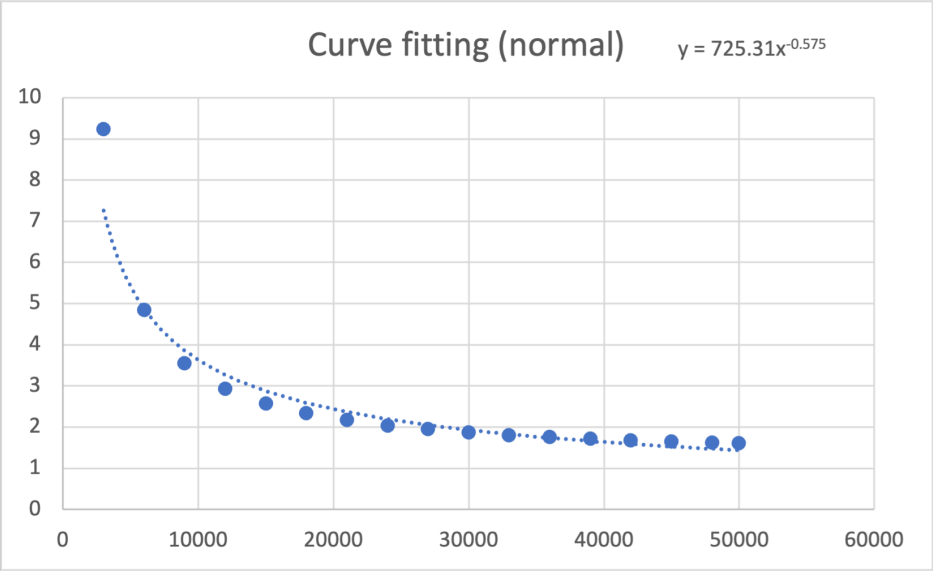

Figure 4.1: Curve fitting (normal Data Center parameters)

On the other hand, in the second case, after setting the *air distribution* parameter to *perimeter cooling*, the same procedure was followed as described earlier, resulting in the following formula: $y = 1337.4x^{-0.623}$ and the corresponding chart is illustrated in figure 4.2.

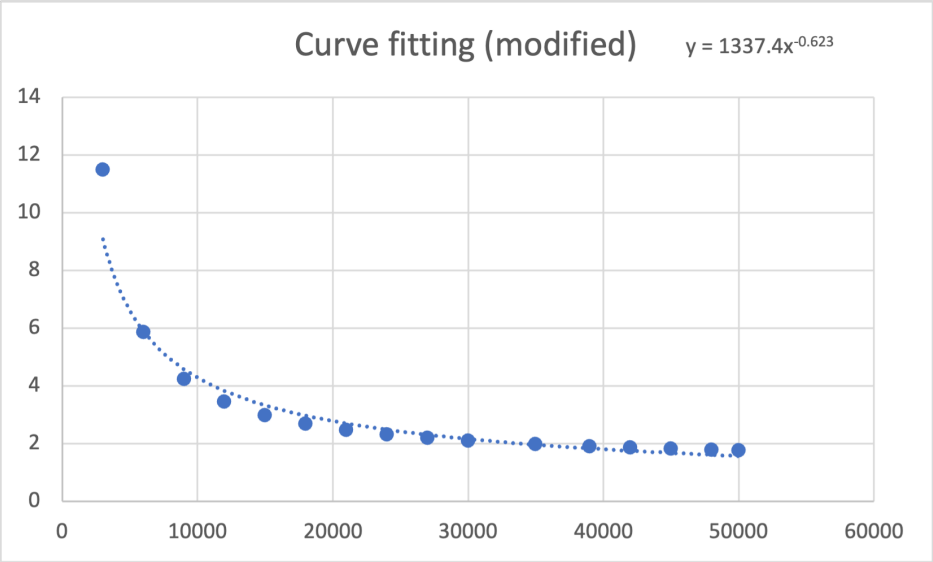

Figure 4.2: Curve fitting (modified Data Center parameters)

4.5 Executed simulations

In order to examine the variation of PUE across several scenarios, multiple simulations under various conditions have been carried out. The simulations results, organized as described in Section 3.2.3, are accessible in the simulations_reports directory of the research work's repository. This directory comprises individual subdirectories corresponding to each conducted simulation. The following subsections provide descriptions of the individual conducted simulations. For each simulation a *PUE* chart has been obtained using the formulas previously derived.

4.5.1 Normal load simulation

The first simulation has been performed keeping a constant load of about 40% for the whole simulation time. This scenario is purely theoretical and does not account for load fluctuations, but it provides an insight into the *PUE* values throughout the normal load periods. Figure 4.3 shows *PUE* variation throughout the simulation. It is apparent that after a first period of about 5 seconds, during which the Data Center becomes fully operational, it remains constant for the whole simulation time and its value is around **2.32**.

4.5.2 Fast DoS scenario simulation

This simulation has been performed by modeling a scenario in which the load rapidly increases from a normal condition of 40% to reach 100%. Specifically, the load variation was managed as follows: at the beginning of the simulation, a single user was instantiated, requiring computational power around 40% of the total capacity. At time 20.0, an additional user was added, increasing the computational power demand to 80%. At time 30.0, a third

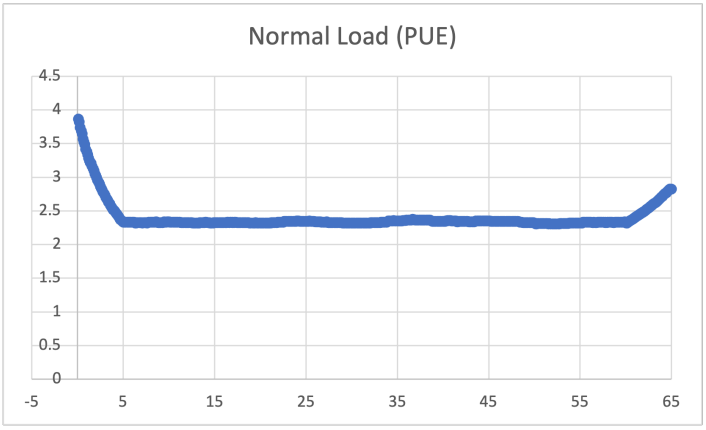

Figure 4.3: Normal Load (PUE)

user was added, utilizing the full computational power. Subsequently, at time 40.0, the last user was added. Figure 4.4 shows *PUE* variation throughout the simulation. It is apparent that as long as the Data Center load has not reached 100%, the variation in *PUE* is noticeable as there is a **23% decrease** when the second user joins the Data Center (from 2.33 to 1.80) and a **6.66% decrease** when the third user joins the Data Center (from 1.80 to 1.68); however, after reaching the peak, this variation disappears entirely.

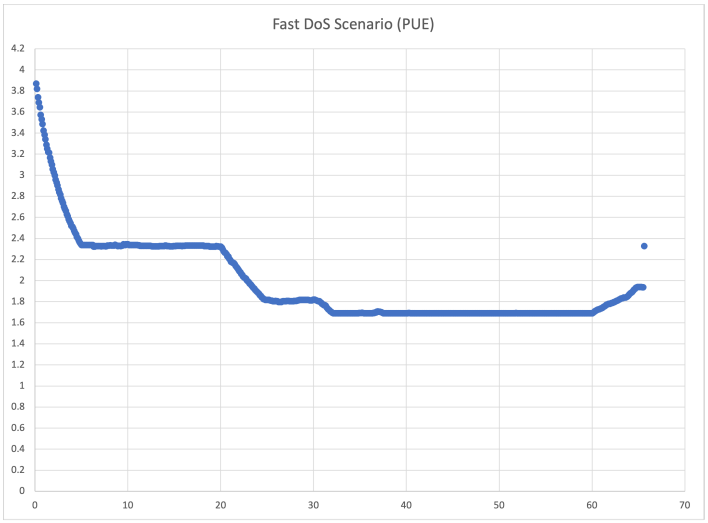

Figure 4.4: Fast Denial of Service Scenario (PUE)

4.5.3 Slow DoS scenario simulation

This simulation has been performed by modeling a scenario in which the load slowly increases from a condition of 10% to reach 100%. Specifically, the load variation was managed as follows: at the beginning of the simulation, a single user was instantiated, requiring computational power around 10% of the total capacity and then every 60 seconds a new user was instantiated for a total of 11 users throughout the simulation that lasted for 660 seconds. Figure 4.5 shows *PUE* variation throughout the simulation. It is evident that each

load variation is definitely less observable compared to the fast *DoS* scenario and is more pronounced as long as the Data Center load has not reached 60% (around the time 360.0). *PUE* variation ranges between 12% and 4%.

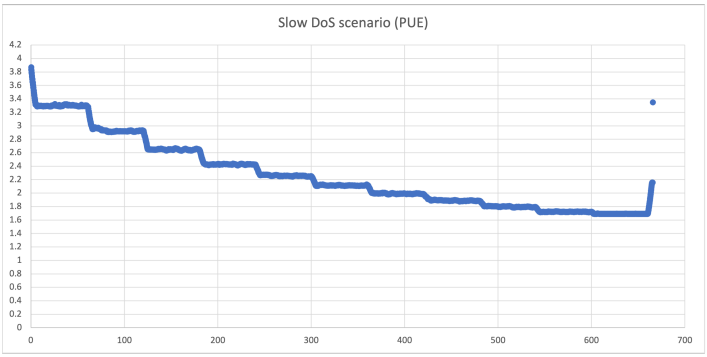

Figure 4.5: Slow Denial of Service Scenario (PUE)

4.5.4 Cooling system attack scenario simulation

In order to simulate the scenario where the cooling system is attacked as outlined in subsection 4.5.4, data from the previously conducted simulations were utilized. Specifically, the data from the silent *DoS* scenario were used as it covers all the workload conditions. The *PUE* was calculated using both the formula applicable under normal conditions and the formula applicable under the cooling system attack scenario. The variation in *PUE* was then observed. Figure 4.6 illustrates the comparison of *PUE* trends throughout the whole simulation, showing the difference between the Data Center when it has not experienced a cooling system attack and when it has. *PUE* value variation ranges **between 17.4% and 10.5%**.

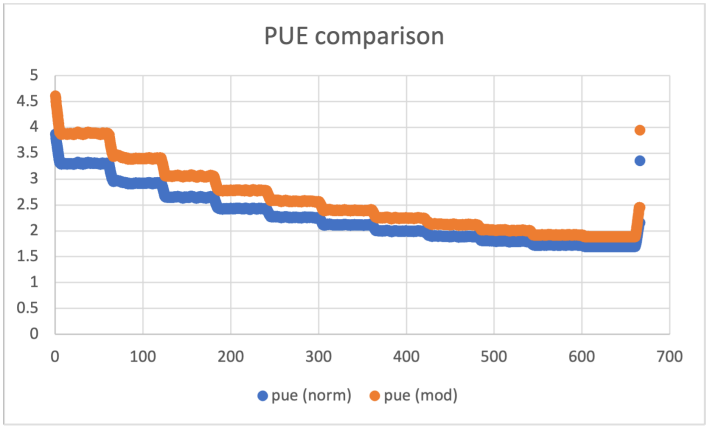

Figure 4.6: PUE comparison

4.6 Results

Once the simulations are done and the graphs showing changes in *PUE* are obtained, it is important to look at the data to figure out if the *PUE* parameter is really helpful to detect the studied attacks. The following subsections aim to provide both reflections about the collected data placing them in real-world contexts and a comparison with the current state of the art will be carried out to contextualize the proposed approach within existing detection techniques.

4.6.1 PUE utilization in DoS scenarios

The simulation conducted to model the fast *DoS* scenario, reveals a noticeable percentage change in *PUE*, which is observable up to a certain point, and then gradually diminishes until it disappears completely. The reason behind this phenomenon can be attributed to the shape of the *PUE* curve illustrated in figure 4.7 (gathered from the *Schneider Electric SE* tool). After reaching 55% of the maximum supported capacity by the Data Center, it becomes less steep compared to the portion ranging from 0% to 55%. It then further flattens after reaching 75%, stabilizing at approximately 1.70. This explains the sudden variation in *PUE* during the time the Data Center takes to become operational. The collected data suggests that the time it takes for the Data Center to start accommodating a sudden drastic load change (from 40% to 80% of total capacity) is around 5 seconds. During this period, there is an increase in the overall energy consumption of about 11000 W (23% *PUE* decreasement). However, the scenario is quite different when transitioning from 80% to full resource saturation. In that case, the change is about 5000 W in 5 seconds (6.66% *PUE decreasement*), and it eventually disappears with further computational resource demands. Furthermore, in the case of a slow DoS situation, the PUE variation becomes less noticeable. In the analyzed scenario, the maximum observable percentage variation is 12% when the load goes from 10% to 20%. However, as the load increases, the variation becomes lower due to the reasons discussed earlier. Therefore, the PUE can definitely be used as a parameter for detecting rapid and noisy DoS attacks, up to the point of reaching the peak supported load. There is no need to use static values or thresholds; it is enough to observe a significant *PUE* variation within a certain time interval. In the studied case, this interval is 5 seconds, as mentioned before, which is the time the Data Center takes to accommodate load changes (independent of simulation time), while the significance of the *PUE* variation needs to be established based on the context. A non-trivial problem lies in distinguishing between a *DoS* attack and an increase in load due to legitimate traffic that results in a significant *PUE* variation. The distinction between these two scenarios cannot be solely accomplished through monitoring the *PUE*, but requires additional methodologies, such as a profile of energy consumption during different times of the day and year. Based on these profiles, the percentage variation threshold is dynamically adjusted. To understand the approach based on variable thresholds,

it is possible to consider the following practical scenario. Let's consider a data center that supports real-time communication activities (including meetings) of a company operational from 9 to 18. Let's suppose that after performing a profiling activity, it becomes evident that the majority of traffic is concentrated between 9:30 and 12:30, and between 15:00 and 17:30. Therefore, it is reasonable to expect a sharp decrease in PUE around 9:30 and 15:00. Hence, a variation recorded around those times is presumably not indicative of a DoS attack. Let's assume that generally, during active hours, around 45% of the available computational resources are utilized on average. In this case, it is reasonable to consider that a potential PUE variation around or exceeding 20% could be a symptom of a DoS attack. Finally, suppose that during the night and holidays, the utilization of computing resources hovers around 5%. In that scenario, even a 10% PUE variation could signify a DoS attack, as it would mean reaching 13% of the computing resources' utilization in the data center, namely more than double the observed load. It is important to take into account that the numbers mentioned are purely indicative and come from results obtained through simulations conducted on a simplified model of a data center. To implement such a system in a real-world scenario, it is necessary to consider data obtained from that specific real system.

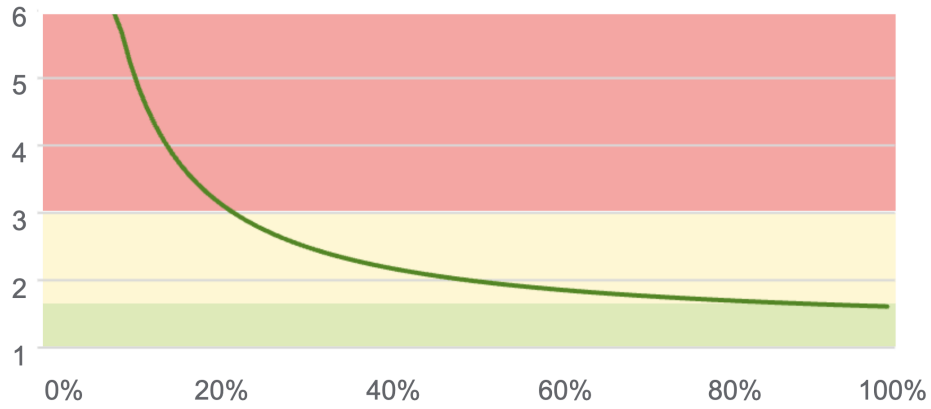

Figure 4.7: PUE curve

4.6.2 PUE utilization in cooling system attack scenarios

As highlighted by the results obtained and described in subsection 4.5.4, unlike the case of the *DoS* scenario, the difference in *PUE* variation is well observable regardless of the load conditions. Once again, the reason behind this phenomenon can be attributed to the nature of the curve describing the *PUE*, which, in the case of an attack on the close-coupled cooling system, is shifted. Figure 4.8 illustrates the curve obtained through curve fitting for the scenario where no attack is occurring and the curve obtained through curve fitting for the scenario with an attack on the cooling system, presented on the same graph. Although the two curves seem to overlap at some point, the study has shown that even under maximum load conditions a difference in *PUE* of 10% is observable. Unlike the *DoS* case,

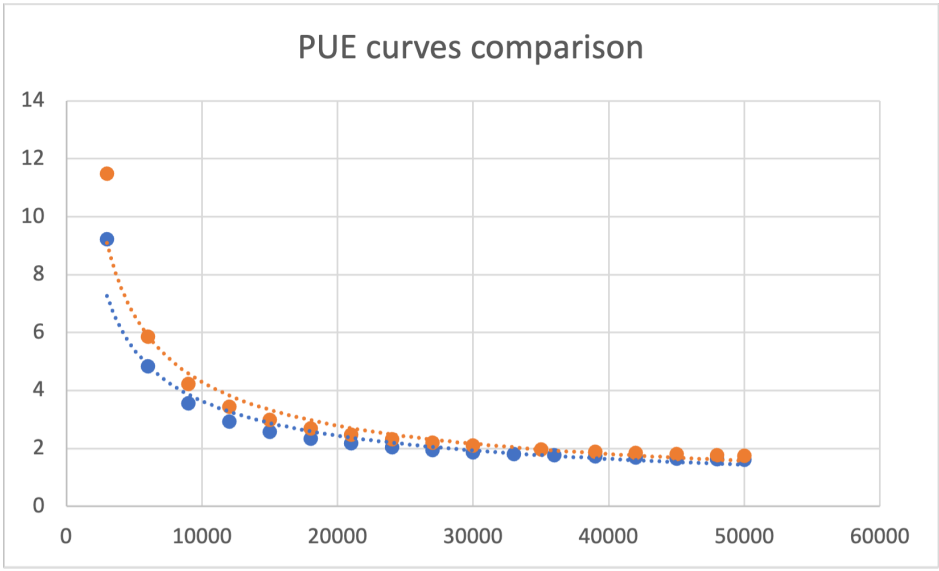

Figure 4.8: PUE curves comparison

where detecting the attack involves monitoring a shift on the same curve that tends to flatten out, in this scenario, the problem is reduced to determining which curve the points observed during runtime belong to. If they belong to the curve that describes the scenario of an attack at the cooling system, it is reasonable to assume that an attack is occurring. Referring to the real-world scenario described earlier, an attack that disables the close-coupled cooling system can be detected at any time of the day, even during peak operational hours of the Data Center. Unlike the previous case, in this scenario, it's not just the *PUE* variation itself that needs to be observed but a variation while keeping energy consumption constant. Therefore, if the *PUE* is expected to have a certain value around a specific level of energy consumption, a significant change in it (which depends on load conditions) under the same energy consumption conditions can be indicative of an attack. Highlighting once again, the numbers mentioned are approximate and derived from simulations on a simplified Data Center model. To deploy this system in a real-world scenario, specific data from the actual Data Center must be considered. The scatter graph shown in figure 4.9 displays the percentage change in PUE between the two scenarios, starting from a 10% load and reaching the maximum load. It is noticeable that this variation ranges between 10% and 18%, making the *PUE* a valid parameter for attack detection.

4.6.3 Comparison with the State of the Art

After explaining the limits within which *PUE* results effective to detect the described attacks, it is useful to contextualize these approaches within the current state of the art solutions. These approaches do not need specific equipment for *PUE* calculation; a wattmeter and one of the *PUE* calculation techniques explained in Subsection 1.1.3 are enough. In comparison to conventional *DoS* detection methods outlined in Section 2.1, this detection

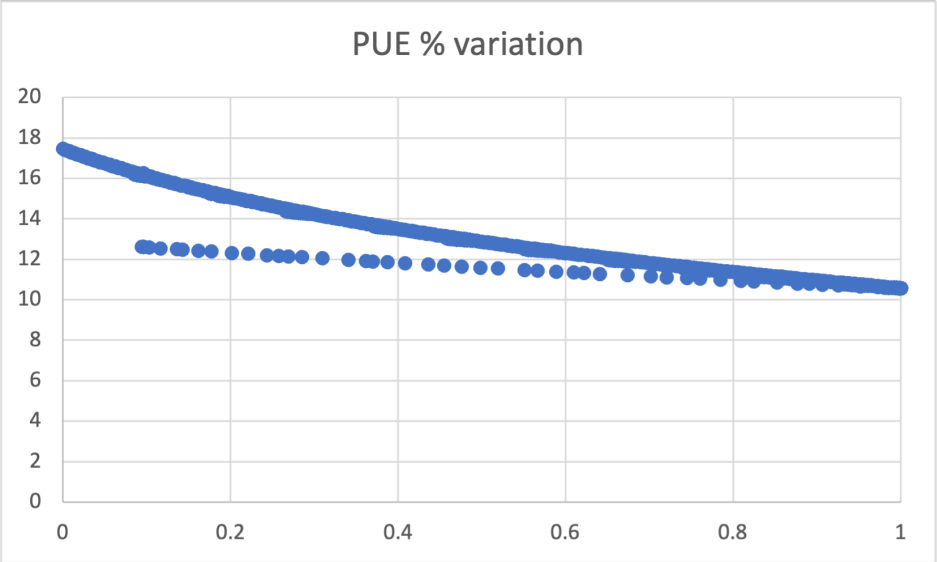

Figure 4.9: PUE percentage variation

technique operates without the need to understand or inspect network traffic. For this reason, it does not require artificial intelligence techniques or models training in order to work. However, as explained before, this approach has inherent limitations, especially during peak load moments when distinguishing between an attack and a normal load increase can become difficult. Therefore, as highlighted before, it is crucial to complement this technique with a comprehensive consumption profile. In general, the proposed technique can be used in synergy with existing state of the art solutions, functioning as an early warning system for potential *DoS* attacks. In cases where the detection threshold is not sufficiently high, a traffic inspection technique should be used. On the other hand, as emphasized earlier in section 2.2, an attack on the cooling system typically involves an intrusion into the system, and the conventional method to detect it is through intrusion detection. In the state of the art, there are various intrusion detection techniques, some of which were illustrated in the dedicated section on the state of the art (section 2.2). However, the approach proposed in this research work aims to detect the cooling system attack through the identification of a situation symptomatic of this specific kind of attack, so it remains effective even if a potential intrusion is not detected by the *IDS/IPS*. A practical scenario in which this could occur is when unauthorized access happens through the theft of a device belonging to an authorized user. In such a case, it becomes challenging to detect the violation. Similar to the technique for detecting *DoS* attacks, this method also does not require traffic inspection as it relies solely on *PUE* calculation. Moreover, this technique can be used in synergy with other classical techniques employed in the state of the art, serving as an alarm not only for cooling system attacks but also for potential intrusions, which would then be further investigated by traditional *IDS/IPS*. It is important to observe that in this context, detection may not necessarily indicate an attack but could be a symptom of a fault in the cooling system. In

such circumstances, distinguishing between the two cases can be difficult as the conditions under which attacks and faults occur are nearly identical. However, this indistinguishability does not pose a significant issue, as both candidate situations are anomalous and require immediate attention.

CHAPTER 5

Conclusions and Future Work

This chapter summarizes the research study and discusses potential future works.

5.1 Conclusions

The idea behind this research work originated from a reflection about the usage of the *PUE* parameter which is mainly used by companies as an indicator of their data centers' efficiency. The aim was to investigate the relationship between the variation of this parameter and the presence of a cyberattack, providing a practical importance to a parameter that is primarily used in a commercial and promotional way. After studying the *PUE* metric, through the exploration of its introduction, usage and calculation, two attack scenarios to be evaluated, namely the *DoS* and the cooling system attack, were chosen. Subsequently, an investigation about the techniques employed in the state of the art to detect the chosen attacks was carried out in order to contextualize the proposed approach into the modern techniques. Since this research required data about the energy consumption of Data Centers under various load and cooling conditions, a close attention was payed to the selection of a Data Center simulator that was able to provide accurate energy reports. The *GreenCloud* simulator has been chosen as the reference platform for the simulations that were carried out in order to gather the data that revealed a correlation between the *PUE* variation and the presence of the above-mentioned attacks. The results can be summarized as follows:

- *DoS* attack detection through *PUE* variation monitoring requires a workload profile in order to distinguish between an expected load increase and an attack. Depending on the workload level the *PUE* variation can be more or less observable. When the load goes from 40% to 80% a *PUE* variation of 23% is observable, while when the load goes from 80% to 100% only a *PUE* variation of 6.66% can be observed. Overall, the *DoS* detection through *PUE* variation monitoring is possible since the *PUE* variation is not negligible; nevertheless such a detection algorithm, whose implementation goes beyond the scope of this research work, should be implemented in order to validate this methodology;

- Cooling system attacks detection through *PUE* variation monitoring is definitely more feasible than the *DoS* attack detection as this problem is reduced to determining which *PUE* curve the points observed during runtime belong to, so it only requires the calculation of *PUE* curves under normal and altered conditions. Overall, there is a strong correlation between *PUE* variation and cooling system attack and this approach can also be employed to detect faults in the cooling system since they occur under the same conditions as the attacks. Therefore, in the case of detection, it is important to distinguish between an attack and a fault, even though both situations require urgent intervention.

It should be clear that the goal of this research work is not to provide the implementation of a detection algorithm based on *PUE* variation monitoring. However, an eventual detection algorithm based on *PUE* monitoring should be integrated into a more complex security

ecosystem in order to bring better results. This leads to the consideration that the aim of this work is to propose an approach that could bring to the implementation of a system that integrates with the current technologies. The choice of the parameter to monitor for attack detection fell on *PUE* as it is easy to calculate, due to the cost-effectiveness of the tools used for its computation. Moreover, this provides the basis to design and implement an architecture specialized in monitoring the energy consumption of the Data Center that, regardless of the specific issue addressed in this research, results effective to spread an 'energy culture' which is useful for reducing operational costs and emissions in Data Centers.

5.2 Future Work

As highlighted on multiple occasions, this research work was conducted under certain assumptions and simplifications, some of which were useful to focus only on relevant issues, while others could be overcome to obtain data more in line with real scenarios. Three future work ideas are described below:

- Servers should not be modeled as single-core machines but as multi-core machines;
- The linear model for energy consumption is simplified, and more complex models in line with those used by modern processors should be considered;
- *GreenCloud* lacks a module for managing the thermal aspects of the data center and for PUE calculation. Therefore, it would be useful to implement a module for handling these aspects. However, this is not straightforward, as it would require knowledge of the conditioning models, which is a complex field.

Thanks to the open-source nature of *GreenCloud*, it is possible to make these modifications and additions to the simulator. Finally, as mentioned before, this research work opens the door for the implementation of attack detection algorithms based on *PUE* variation monitoring that could be used to improve Data Centers security. The implementation of these algorithms could provide insights about the potential and limitations of the proposed approach through a real-world testing within actual security ecosystems, helping to understand the practical benefits of this methodology.

Acknowledgements

Bibliography

[1] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey, "Recalibrating global data center energy-use estimates," *Science*, vol. 367, no. 6481, pp. 984–986, 2020. (Citato a pagina 2)

[2] L. Belkhir and A. Elmeligi, "Assessing ict global emissions footprint: Trends to 2040 & recommendations," *Journal of cleaner production*, vol. 177, pp. 448–463, 2018. (Citato a pagina 2)

[3] V. Avelar, D. Azevedo, A. French, and E. N. Power, "Pue: a comprehensive examination of the metric," *White paper*, vol. 49, 2012. (Citato alle pagine 2 e 3)

[4] "Understanding denial-of-service attacks," Feb 2021. (Citato a pagina 4)

[5] Z. Shao, M. A. Islam, and S. Ren, "Heat behind the meter: A hidden threat of thermal attacks in edge colocation data centers," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 318–331, 2021. (Citato a pagina 4)

[6] Q. Liu, Y. Ma, M. Alhussein, Y. Zhang, and L. Peng, "Green data center with iot sensing and cloud-assisted smart temperature control system," *Computer Networks*, vol. 101, pp. 104–112, 2016. (Citato a pagina 5)

[7] M. K. J. Ramphela, P. A. Owolawi, T. Mapayi, and G. Aiyetoro, "Internet of things (iot) integrated data center infrastructure monitoring system," in *2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, pp. 1–6, IEEE, 2020. (Citato a pagina 5)

[8] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019. (Citato a pagina 5)

[9] W. Zhou, W. Jia, S. Wen, Y. Xiang, and W. Zhou, "Detection and defense of application-layer ddos attacks in backbone web traffic," *Future Generation Computer Systems*, vol. 38, pp. 36–46, 2014. (Citato a pagina 8)

[10] J. Yu, H. Lee, M.-S. Kim, and D. Park, "Traffic flooding attack detection with snmp mib using svm," *Computer Communications*, vol. 31, no. 17, pp. 4212–4219, 2008. (Citato a pagina 8)

[11] M.-h. YANG and R.-c. WANG, "Ddos detection based on wavelet kernel support vector machine," *The Journal of China Universities of Posts and Telecommunications*, vol. 15, no. 3, pp. 59–94, 2008. (Citato a pagina 8)

[12] L. Zhang, W. Zhou, and L. Jiao, "Wavelet support vector machine," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 34–39, 2004. (Citato a pagina 8)

[13] T. Spyridopoulos, G. Karanikas, T. Tryfonas, and G. Oikonomou, "A game theoretic defence framework against dos/ddos cyber attacks," *Computers & Security*, vol. 38, pp. 39–50, 2013. (Citato a pagina 8)

[14] U. Akyazı and A. Ş. Uyar, "Detection of ddos attacks via an artificial immune system-inspired multiobjective evolutionary algorithm," in *Applications of Evolutionary Computation: EvoApplications 2010: EvoCOMNET, EvoENVIRONMENT, EvoFIN, EvoMUSART, and EvoTRANSLOG, Istanbul, Turkey, April 7-9, 2010, Proceedings, Part II*, pp. 1–10, Springer, 2010. (Citato a pagina 8)

[15] W. Dou, Q. Chen, and J. Chen, "A confidence-based filtering method for ddos attack defense in cloud environment," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1838–1850, 2013. (Citato a pagina 8)

[16] M. Barati, A. Abdullah, N. I. Udzir, R. Mahmod, and N. Mustapha, "Distributed denial of service detection using hybrid machine learning technique," in *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*, pp. 268–273, IEEE, 2014. (Citato a pagina 9)

[17] N. S. Sulaiman, A. Nasir, W. R. W. Othman, S. F. A. Wahab, N. S. Aziz, A. Yacob, and N. Samsudin, "Intrusion detection system techniques: a review," in *Journal of Physics: Conference Series*, vol. 1874, p. 012042, IOP Publishing, 2021. (Citato a pagina 9)

[18] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019. (Citato a pagina 9)

[19] N. Mansouri, R. Ghafari, and B. M. H. Zade, "Cloud computing simulators: A comprehensive review," *Simulation Modelling Practice and Theory*, vol. 104, p. 102144, 2020. (Citato alle pagine 10, 11, 12, 13, 14, 15, 16, 18 e 19)

[20] P. S. Suryateja, "A comparative analysis of cloud simulators.," *International Journal of Modern Education & Computer Science*, vol. 8, no. 4, 2016. (Citato alle pagine 10, 14 e 21)

[21] D. P. Abreu, K. Velasquez, M. Curado, and E. Monteiro, "A comparative analysis of simulators for the cloud to fog continuum," *Simulation Modelling Practice and Theory*, vol. 101, p. 102029, 2020. (Citato alle pagine 10 e 15)

[22] K. M. Khalil, M. Abdel-Aziz, T. T. Nazmy, and A.-B. M. Salem, "Cloud simulators–an evaluation study," *International Journal Information Models and Analyses*, vol. 6, no. 1, 2017. (Citato alle pagine 10, 14, 15, 21 e 24)

[23] N. Patel and H. Patel, "A comprehensive assessment and comparative analysis of simulations tools for cloud computing," *Int J Eng Comput Sci*, vol. 5, no. 11, pp. 18972–18978, 2016. (Citato alle pagine 10, 11, 12 e 15)

[24] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011. (Citato alle pagine 10 e 11)

[25] S. K. Garg and R. Buyya, "Networkcloudsim: Modelling parallel applications in cloud simulations," in *2011 Fourth IEEE International Conference on Utility and Cloud Computing*, pp. 105–113, IEEE, 2011. (Citato a pagina 11)

[26] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *2010 24th IEEE international conference on advanced information networking and applications*, pp. 446–452, IEEE, 2010. (Citato a pagina 11)

[27] R. N. Calheiros, M. A. Netto, C. A. De Rose, and R. Buyya, "Emusim: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications," *Software: Practice and Experience*, vol. 43, no. 5, pp. 595–612, 2013. (Citato a pagina 11)

[28] F. Fittkau, S. Frey, and W. Hasselbring, "Cdosim: Simulating cloud deployment options for software migration support," in *2012 IEEE 6th International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA)*, pp. 37–46, IEEE, 2012. (Citato a pagina 12)

[29] Y. Jararweh, Z. Alshara, M. Jarrah, M. Kharbutli, and M. N. Alsaleh, "Teachcloud: a cloud computing educational toolkit," *International Journal of Cloud Computing 1*, vol. 2, no. 2-3, pp. 237–257, 2013. (Citato a pagina 12)

[30] X. Li, X. Jiang, P. Huang, and K. Ye, "Dartcsim: An enhanced user-friendly cloud simulation system based on cloudsim with better performance," in *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, vol. 1, pp. 392–396, IEEE, 2012. (Citato a pagina 12)

[31] X. Li, X. Jiang, K. Ye, and P. Huang, "Dartcsim+: Enhanced cloudsim with the power and network models integrated," in *2013 IEEE Sixth International Conference on Cloud Computing*, pp. 644–651, IEEE, 2013. (Citato a pagina 12)

[32] Z. Cai, Q. Li, and X. Li, "Elasticsim: A toolkit for simulating workflows with cloud resource runtime auto-scaling and stochastic task execution times," *Journal of Grid Computing*, vol. 15, pp. 257–272, 2017. (Citato a pagina 13)

[33] A. Kohne, M. Spohr, L. Nagel, and O. Spinczyk, "Federatedcloudsim: a sla-aware federated cloud simulation framework," in *Proceedings of the 2nd International Workshop on CrossCloud Systems*, pp. 1–5, 2014. (Citato a pagina 13)

[34] A. Zhou, S. Wang, Q. Sun, H. Zou, and F. Yang, "Ftcloudsim: A simulation tool for cloud service reliability enhancement mechanisms," in *Proceedings Demo & Poster Track of ACM/IFIP/USENIX International Middleware Conference*, pp. 1–2, 2013. (Citato a pagina 13)

[35] W. Chen and E. Deelman, "Workflowsim: A toolkit for simulating scientific workflows in distributed environments," in *2012 IEEE 8th international conference on E-science*, pp. 1–8, IEEE, 2012. (Citato a pagina 13)

[36] T. Teixeira Sá, R. N. Calheiros, and D. G. Gomes, "Cloudreports: an extensible simulation tool for energy-aware cloud computing environments," *Cloud computing: Challenges, limitations and R&D solutions*, pp. 127–142, 2014. (Citato a pagina 13)

[37] W. A. Higashino, M. A. Capretz, and L. F. Bittencourt, "Cepsim: a simulator for cloud-based complex event processing," in *2015 IEEE International Congress on Big Data*, pp. 182–190, IEEE, 2015. (Citato a pagina 14)

[38] D. C. Luckham and B. Frasca, "Complex event processing in distributed systems," *Computer Systems Laboratory Technical Report CSL-TR-98-754. Stanford University, Stanford*, vol. 28, p. 16, 1998. (Citato a pagina 14)

[39] M. Bux and U. Leser, "Dynamiccloudsim: Simulating heterogeneity in computational clouds," in *Proceedings of the 2nd acm sigmod workshop on scalable workflow execution engines and technologies*, pp. 1–12, 2013. (Citato a pagina 14)

[40] Y. Jararweh, M. Jarrah, Z. Alshara, M. N. Alsaleh, M. Al-Ayyoub, *et al.*, "Cloudexp: A comprehensive cloud computing experimental framework," *Simulation Modelling Practice and Theory*, vol. 49, pp. 180–192, 2014. (Citato a pagina 14)

[41] D. C. Alves, B. G. Batista, D. M. Leite Filho, M. L. Peixoto, S. Reiff-Marganiec, and B. T. Kuehne, "Cm cloud simulator: a cost model simulator module for cloudsim," in *2016 IEEE World Congress on Services (SERVICES)*, pp. 99–102, IEEE, 2016. (Citato a pagina 14)

[42] J. Jung and H. Kim, "Mr-cloudsim: Designing and implementing mapreduce computing model on cloudsim," in *2012 International Conference on ICT Convergence (ICTC)*, pp. 504–509, IEEE, 2012. (Citato a pagina 15)

[43] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008. (Citato a pagina 15)

[44] M. H. Sqalli, M. Al-Saeedi, F. Binbeshr, and M. Siddiqui, "Ucloud: A simulated hybrid cloud for a university environment," in *2012 IEEE 1ST International Conference on Cloud Networking (CLOUDNET)*, pp. 170–172, IEEE, 2012. (Citato a pagina 15)

[45] J. Son, A. V. Dastjerdi, R. N. Calheiros, X. Ji, Y. Yoon, and R. Buyya, "Cloudsimsdn: Modeling and simulation of software-defined cloud data centers," in *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 475–484, IEEE, 2015. (Citato alle pagine 15 e 21)

[46] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (sdn): a survey," *Security and communication networks*, vol. 9, no. 18, pp. 5803–5833, 2016. (Citato a pagina 15)

[47] S.-H. Lim, B. Sharma, G. Nam, E. K. Kim, and C. R. Das, "Mdcsim: A multi-tier data center simulation, platform," in *2009 IEEE International Conference on Cluster Computing and Workshops*, pp. 1–9, IEEE, 2009. (Citato a pagina 15)

[48] S. K. Gupta, R. R. Gilbert, A. Banerjee, Z. Abbasi, T. Mukherjee, and G. Varsamopoulos, "Gdcsim: A tool for analyzing green data center design and resource management techniques," in *2011 International Green Computing Conference and Workshops*, pp. 1–8, IEEE, 2011. (Citato alle pagine 15 e 21)

[49] T. Cucinotta and A. Santogidis, "Cloudnetsim-simulation of real-time cloud computing applications," in *Proceedings of the 4th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, 2013. (Citato a pagina 16)

[50] A. W. Malik, K. Bilal, K. Aziz, D. Kliazovich, N. Ghani, S. U. Khan, and R. Buyya, "Cloudnetsim++: A toolkit for data center simulations in omnet++," in *2014 11th Annual High Capacity Optical Networks and Emerging/Enabling Technologies (Photonics for Energy)*, pp. 104–108, IEEE, 2014. (Citato a pagina 16)

[51] D. Kliazovich, P. Bouvry, and S. U. Khan, "Greencloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, vol. 62, pp. 1263–1283, 2012. (Citato alle pagine 17, 21, 26, 32 e 33)

[52] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente, "icancloud: A flexible and scalable cloud infrastructure simulator," *Journal of Grid Computing*, vol. 10, pp. 185–209, 2012. (Citato a pagina 18)

[53] U. U. Rehman, A. Ali, and Z. Anwar, "seccloudsim: secure cloud simulator," in *2014 12th International Conference on Frontiers of Information Technology*, pp. 208–213, IEEE, 2014. (Citato a pagina 18)

[54] S. Ostermann, K. Plankensteiner, R. Prodan, and T. Fahringer, "Groudsim: An event-based simulation framework for computational grids and clouds," in *Euro-Par 2010 Parallel Processing Workshops: HeteroPar, HPCC, HiBB, CoreGrid, UCHPC, HPCF, PROPER, CCPI, VHPC, Ischia, Italy, August 31–September 3, 2010, Revised Selected Papers 16*, pp. 305–313, Springer, 2011. (Citato a pagina 18)

[55] T. Fahringer, A. Jugravu, S. Pllana, R. Prodan, C. Seragiotto Jr, and H.-L. Truong, "Askalon: a tool set for cluster and grid computing," *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2-4, pp. 143–169, 2005. (Citato a pagina 18)

[56] W. Tian, Y. Zhao, M. Xu, Y. Zhong, and X. Sun, "A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 153–161, 2013. (Citato a pagina 19)

[57] S. Sotiriadis, N. Bessis, N. Antonopoulos, and A. Anjum, "Simic: Designing a new inter-cloud simulation platform for integrating large-scale resource management," in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 90–97, IEEE, 2013. (Citato a pagina 19)

[58] I. Sriram, "Speci, a simulation tool exploring cloud-scale data centres," in *Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings 1*, pp. 381–392, Springer, 2009. (Citato a pagina 19)

[59] D. Fernández-Cerero, A. Fernández-Montes, A. Jakóbik, J. Kołodziej, and M. Toro, "Score: Simulator for cloud optimization of resources and energy consumption," *Simulation Modelling Practice and Theory*, vol. 82, pp. 160–173, 2018. (Citato a pagina 19)

[60] D. Fernández-Cerero, A. Jakóbik, A. Fernández-Montes, and J. Kołodziej, "Game-score: Game-based energy-aware cloud scheduler and simulator for computational clouds," *Simulation Modelling Practice and Theory*, vol. 93, pp. 3–20, 2019. (Citato alle pagine 19 e 21)

[61] G. Kecskemeti, "Dissect-cf: a simulator to foster energy-aware scheduling in infrastructure clouds," *Simulation Modelling Practice and Theory*, vol. 58, pp. 188–218, 2015. (Citato alle pagine 20 e 21)

[62] C. Badii, P. Bellini, I. Bruno, D. Cenni, R. Mariucci, and P. Nesi, "Icaro cloud simulator exploiting knowledge base," *Simulation Modelling Practice and Theory*, vol. 62, pp. 1–13, 2016. (Citato a pagina 21)

[63] M. Shiraz, A. Gani, R. H. Khokhar, and E. Ahmed, "An extendable simulation framework for modeling application processing potentials of smart mobile devices for mobile cloud computing," in *2012 10th International Conference on Frontiers of Information Technology*, pp. 331–336, IEEE, 2012. (Citato a pagina 21)

[64] I. K. Kim, W. Wang, and M. Humphrey, "Pics: A public iaas cloud simulator," in *2015 IEEE 8th International Conference on Cloud Computing*, pp. 211–220, IEEE, 2015. (Citato a pagina 21)

[65] S. K. S. Gupta, A. Banerjee, Z. Abbasi, G. Varsamopoulos, M. Jonas, J. Ferguson, R. R. Gilbert, and T. Mukherjee, "Gdcsim: A simulator for green data center design and analysis," *ACM Trans. Model. Comput. Simul.*, vol. 24, jan 2014. (Citato alle pagine 21 e 22)

[66] S. K. Gupta, A. Banerjee, Z. Abbasi, G. Varsamopoulos, M. Jonas, J. Ferguson, R. R. Gilbert, and T. Mukherjee, "Gdcsim: A simulator for green data center design and analysis," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 24, no. 1, pp. 1–27, 2014. (Citato a pagina 24)

[67] A. T. Makaratzis, K. M. Giannoutakis, and D. Tzovaras, "Energy modeling in cloud simulation frameworks," *Future Generation Computer Systems*, vol. 79, pp. 715–725, 2018. (Citato a pagina 33)

[68] D. Kliazovich, P. Bouvry, and S. U. Khan, "Dens: data center energy-efficient network-aware scheduling," *Cluster computing*, vol. 16, pp. 65–75, 2013. (Citato a pagina 34)

[69] M. Guzek, D. Kliazovich, and P. Bouvry, "Heros: Energy-efficient load balancing for heterogeneous data centers," in *2015 IEEE 8th International Conference on Cloud Computing*, pp. 742–749, IEEE, 2015. (Citato a pagina 34)

[70] "Choosing close-coupled it cooling solutions," *Anixter.com*, 2017. (Citato a pagina 41)