Final Year Project

Data Centre Analysis Tool

Nicholas Hamm

Student ID: 19439854

A thesis submitted in part fulfilment of the degree of

BSc. (Hons.) in Computer Science

Supervisor: Professor Damian Dalton

UCD School of Computer Science

University College Dublin

May 21, 2023

Table of Contents

Abstract

The aim of this project is to highlight the significant energy consumption of data centres (DC) and develop a dashboard to analysis the cost of their carbon footprint and electricity consumption. With the emergence of new data centres, both the construction and maintenance of these sites has had an environmental impact globally. Without major implementation of renewable energy, these sites are a continuous drain on the electrical grid daily due to the global populations reliance for data centres that store an ever-expanding sea of data. Data centres are not restricted to a large, centralized collection of servers, but can be found in the form of small to medium sized sites such as at a small company or university. This dashboard is developed as a web application to provide administrators of smaller scale DCs with a toolkit that will improve their insight of not only their hourly electricity consumption but the hourly carbon emissions as well. The dashboard utilises the capabilities of the Papillon software in conjunction with Django to calculate the varying hourly tariffs (both carbon and cost tariffs) of hosts running in the DC, offering greater financial and ecological understanding.

Chapter 1: Project Specification

The Data Centre Analysis Tool intends to assist administrators of a smaller-scale DC (data centre) to determine their economic and environmental impact, by implementing analytical insights. The analysis provided by this tool aims to identify more favorable windows for operations to execute. The under-utilisation of servers is a common occurrence in DCs as is evidenced by research conducted by Koomey, Jonathan, and Jon Taylor [1]. This study revealed that of the 16,000-server sample from from five DCs reviewed, 30% of were in a state of 'comatose', performing no meaningful operations over a six-month period. No meaningful operations are classified as a computer using less than 3% of it's central processing unit (CPU) capacity; hence these computers are referred to as zombie servers.

As DCs equate to roughly 1-2% of total electricity consumed globally, zombie servers are a significant drain of electricity resources. Pawlish, Varde and Robila [2] evaluated that the maintenance of the DC in Montclair State University accounted for 2,412,756 kWh in 2010 alone. They also discovered that running these same servers released almost 1,500 metric tons of carbon, enough emissions to completely fill an Olympic size pool. DC administrators have the opportunity to drastically reduce the consumption of economic and environmental resources through better management of existing DC hardware.

1.1 Core Objectives

- Track temporal energy consumption of IT assets/processes in a small DC
- Tag processes based on their mobility across the server population and flexibility of their execution time
- Profile processes in terms of their energy consumption, carbon footprint and financial energy cost. Accounting for electrical tariff rates and carbon intensity
- Create a dashboard interface to allow process to be moved to alternative times, subject to their tags, and calculate the impact on the DC

1.2 Advanced Objectives

- Replace the manual approval of tag assignment with automated optimisation solution techniques such as simulation annealing
- Allow for the analysis of more efficient servers and the impact on the DC

Chapter 2: Related Work and Ideas

This chapter outlines the challenges posed by DCs' energy consumption. An extensive literature review indicates that DCs have a profound impact on current society. The overarching theme of this research was exploration of the inefficiencies that occur during the operation of DCs. This chapter aims to discuss an explanation of the wastefulness of DCs, the impact of their carbon footprints and how this could be reduced.

2.1 What is a Data Centre?

The Oxford Dictionary defines a DC "as a large network of computer servers, typically used by organizations for storing, processing or sharing large amounts of data" [3]. DCs are the evolution of distributed desktop computing which emerged during the 1980s, succeeding the large-scale on-site mainframe computers that dominated the 1960s. According to Lee [4], the fundamental premise of sharing data to a client terminal from the central computer remains the same today, but dedicated communication lines have been replaced as result of the developments in network technology. DCs are comprised of three essential elements; the network infrastructure connects the end user with the DC's services, the storage infrastructure houses the data and the computing resources which are the applications that regulate the system [5].

As discussed in Chapter 1, DCs can differ greatly in size and scale. DCs are classified into 4 different tiers, commencing with Tier I DCs, simple systems administrated by small organisation with limited resources, and progressing to the Tier IV DCs of large multi-national companies that are fault tolerant as well as redundancy capable [6]. The rate of expansion and construction of DCs is expanding rapidly and will continue to grow in the foreseeable future as our reliance on data in all sectors of society intensifies. A further growth in the region of four fold is estimated by 2035, implying the worldwide figure for DCs would be 32 million helping the 'Edge' to handle the Internet of Things (IoT).

2.1.1 Financial Expenditure

The financial investment in order to maintain a DC is usually analysed over the life span of that DC. DCs are a considerable financial expense as such companies and organisations expect a corresponding return from this investment. An investigation into the Total Cost of Ownership (TCO) for a DC has proven that this sum is difficult to comprehensively analyse and predict. This is due to metrics traditional for example, per square foot of DC or per kW of DC, that do not factor in the meaningful work executed. Rasmussen [7] concluded that calculating the TCO for a single DC's physical infrastructure should be conducted on a per-rack basis, normalizing the measurement of TCO. A compelling amount of data is required as a result. Factors including the capital, engineering, installation, and operating cost are the primary costs associated with establishing and maintaining a DC. Fig 2.1 depicts the financial break down of a DC's lifespan, typically 10 years. Approximately half of the expenditure during it's lifespan per rack is capital expense, and the other half is operating expense.

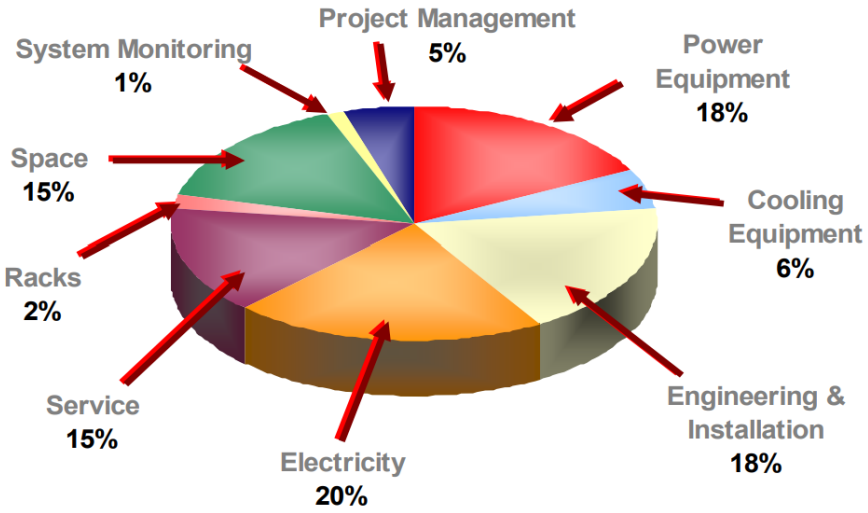

Figure 2.1: The financial break down of a DC over it's lifespan [7]

2.1.2 Measuring the Efficiency of a Data Centre

Research conducted by Malone and Belady [8] in 2006 introduced three new metrics that are frequently used in measuring the efficiency of a DC.

$$PUE = \frac{TotalFacilityPower}{ITEquipmentPower} \tag{2.1}$$

$$DCiE = \frac{1}{PUE} = \frac{ITEquipmentPower}{TotalFacilityPower} \tag{2.2}$$

$$EAC = PUE \cdot \frac{ThreeYearITEquipmentPowerCost}{AccusitonCost} \tag{2.3}$$

The Power Usage Effectiveness (PUE) compares the utilization of the total power available for a DC. The lower the PUE measurement the better in terms of efficiency. By improving the power distribution of a DC, more power can be used for IT equipment such as servers. Data Center Infrastructure Efficiency (DCiE) is the reciprocal of PUE, hence the higher the DCiE the better in terms of efficiency. Both PUE and DCiE identify the utilization of total energy consumed by the DC. The Energy-to-Acquisition Cost (EAC) correlates the cost of operating for the average life of a product to the cost of the actual hardware. Due to the progression and the variety of different hardware components, it is more difficult to provide an accurate and comparable EAC. A PUE of 1 would be an ideal number but this is theoretically impossible as server always need some energy consumption to support the IT equipment [9]. Google hyper scale DCs report a PUE of 1.1 [10]. The large power overheads associated with DCs are due to the energy required to cool the servers. Reducing the aggregate power consumption will inadvertently reduce the cooling demands in the DC. A major flaw with the PUE metric is it's reliance on the recording of multiple parameters resulting in a large amount of data for the final computation.

PUE	DCE	Level of Efficiency
3.0	33%	Very Inefficient
2.5	40%	Inefficient
2.0	50%	Average
1.5	67%	Efficient
1.2	83%	Very Efficient

Figure 2.2: PUE efficiency values [11]

2.1.3 Evolution: Cloud Computing and Edge

Cloud computing refers to delivery of computing services like storage or databases, over the internet. The term 'cloud' has permeated into our everyday vocabulary in recent years. This availability of data has propelled us into the age of the cloud DCs. However, the function of cloud DCs extend beyond storing data. Advancements in both computational and network infrastructure allow applications to be run on a server in a DC and be retrieved by a client. The cloud and edge an interlinked. Edge computing is a distributed IT architecture in which client data is processed at the perimeter of the network, close to the original source. Data is collected and transferred to the cloud, where computing is centralized. By processing data closer to it's source, both speeds and volumes are increased. Cisco compiled a report across 13 countries that concluded that 92% of organisations used two or more public cloud providers in 2022, on top of the 82% of organisations that already accommodate a hybrid cloud approach [5].

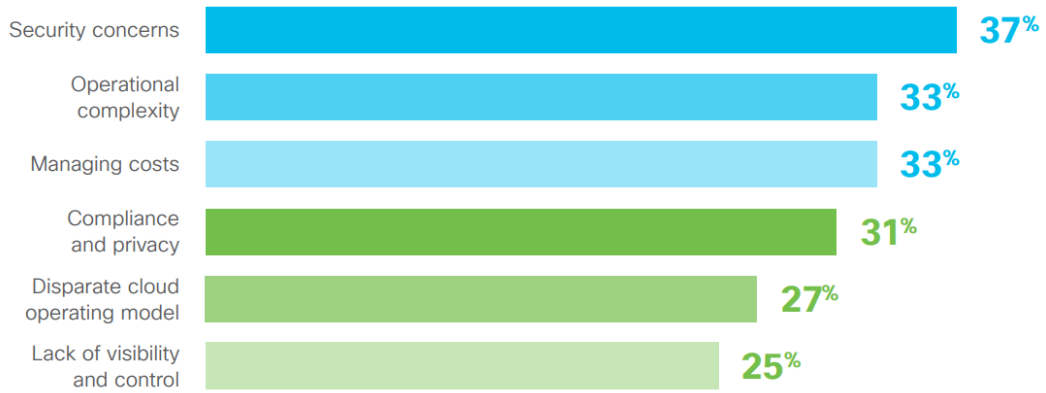

Figure 2.3: Main challenges faced by Cloud Computing [12]

Cloud computing has dominated IT discourse in the past decade, investments and research interest in edge computing has expanded extensively in recent years [13]. The shift towards cloud computing hasn't been seamless, facing many challenge including data loss, vendor lock-in, service outage or even security threats. This is reflected in the report by Cisco in 2022 [12] which identified security as the main concern for employers when migrating to a cloud DC.

2.2 Improving the Efficiency of a Data Centre

This section endeavours to explore different methods of increasing the overall efficiency of a DC. The information reviewed in this chapter correlates to the second and third core objectives of this project, assigning flexibility tags and rescheduling server applications. Analysing the extent to which a server is being utilised can facilitate the reallocation of resources to increase efficiency.

2.2.1 Zombie/Comatose Servers

As discussed in the introduction, zombie servers or servers in a state of comatose are a prevalent issue for DCs, hindering their efficiency. A server is classified as being in a state of comatose if it operates at 3% or less of it's CPU capacity over a six-month period. Comatose servers are constantly draining power from the grid while performing no meaningful computation and as a result emitting carbon into the atmosphere. The issue of server comatose is not exclusive to smaller DCs. However, it is more common and problematic due to the limited resources of smaller-scale operations. Research [1][14] demonstrates that enterprise DCs displayed very low server utilization and high numbers of comatose servers. Koomey and Taylor [1] sampled 16,000 servers and concluded that between 25% to 33% of a DC's assets are inhibited by comatose servers, resulting in zero financial return. Many DC administrators are unaware of the under-utilization of servers they control. As of September 2021, there were 7.2 million DCs globally [15] and the vast majority, roughly 33%, are in the United States. This figure also includes the 600 hyper scale DCs of companies like Amazon, Microsoft and Google [10].

The large sample size of the survey implies that a similar trend of servers in a state of comatose would be replicated on global scale. Simple measures can be introduced by administrators to reduce energy and economic consumption. Barclays discovered they had about 9,000 comatose servers which they removed in 2013, eliminating the 2.5 megawatts of power that they were consuming [16]. The tool this author will develop aims to allow administrators to rearrange the architecture of their servers in addition to providing in depth analysis of electrical usage and carbon emissions, eliminating redundancy in their DCs. Comatose servers identified in the DC can be decommissioned or the load from a neighbouring server can be shared.

2.2.2 Virtualisation

Another method of increasing efficiency of a DC is to deploy virtualisation on servers. Virtualisation is implemented by using software rather than a physical computer or server to complete tasks and deploy applications [2]. A server can have multiple virtual machines (VM) running on it, each performing their own operations. Servers can be virtualised in three main ways; software VMs, hardware VMs and virtual Operating Systems. The diagram below demonstrates their differences, Fig 2.4 [17]. VM enables service-oriented architectures, isolated secure systems and flexible deployment. The efficiency of the DC grows as the revenue savings generated by reducing hardware is enhanced by the remaining active servers reaching their full potential [17].

Estimates for servers that are Windows based architecture average capacity of 8 to 12 percent and UNIX servers utilise a higher 25 to 30 percent of their capacity without virtualisation. Virtualisation promotes increasing server utilization to sustainable levels. In 2008 the Royal Borough of Windsor and Maidenhead virtualised 184 of their physical servers [5]. Standard Life is a pension company that virtualised 65% of its physical servers, releasing almost two thirds of the floor space in the DC in addition to a reduction of £300K per annum on their electricity consumption [18].

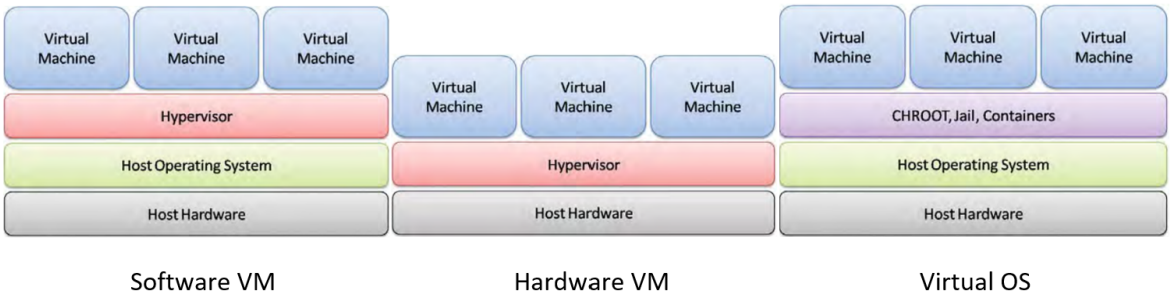

Figure 2.4: Types Virtual Machine Architecture

Multi-tenancy is a disadvantage of virtualisation. Each individual on the server is a tenant. A server with more than one tenant is considered a multi-tenancy server. Security concerns are expressed by some users who don't not want to share a physical hardware device with another. This should not have a significant impact on the development of this project as multi-tenancy is less common in small DCs.

2.2.3 Hardware Upgrades

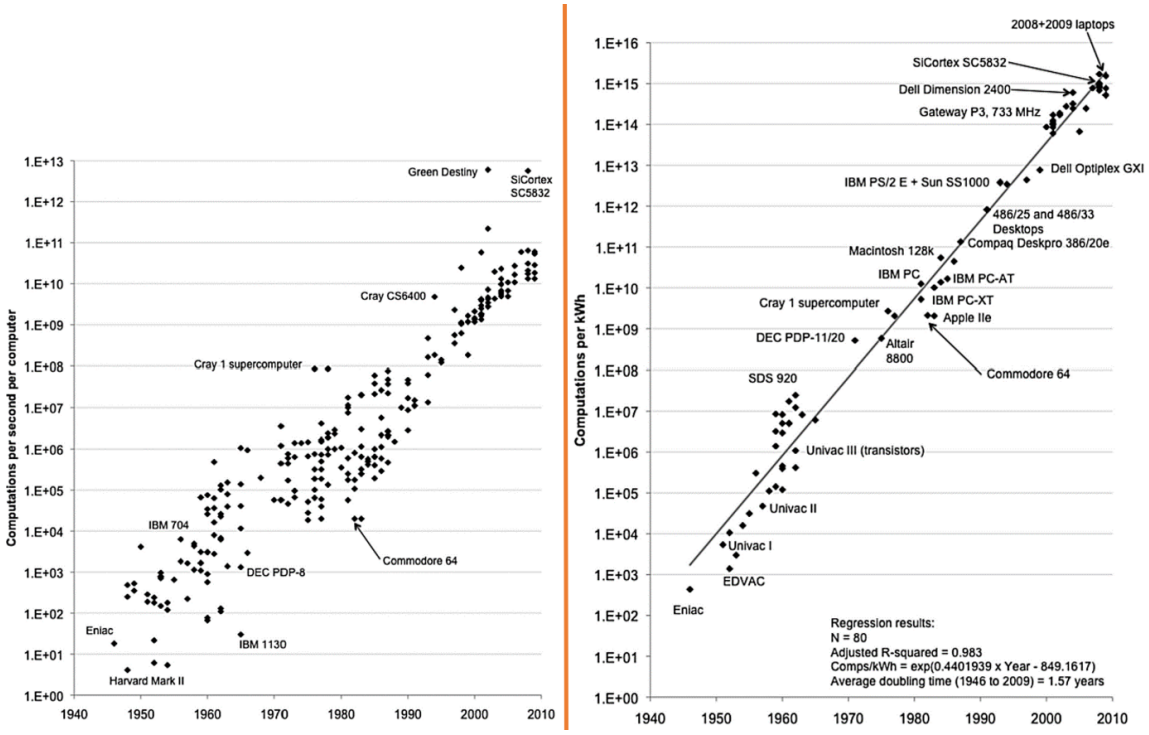

Figure 2.5: Graphs from Nordhaus research. On the left speed increase (Computations per second per computer) and the right shows power increase (Computations per kWh) [19]

A straightforward approach to improving a DC's efficiency is to upgrade the hardware that runs it's servers. Improvements in the electrical effectiveness of computation has progressed significant since the first DCs were constructed, nearly doubling every 18 months for past 60 years [19]. This substantial growth has been documented by Nordhaus [19] since the earliest computers of the 1940s to those of the 2010s, with a performance increase identified as a factor between 1.7 trillion and 76 trillion depending on the method of calculation. The graph, depicted by Fig 2.5 , illustrates this increase, including the edition of modern-day processors. This sentiment was echoed by Horowitz [20] in 2014. However, he concluded that to continue scaling computational

performance will require the design and implementation of new specialized compute engines. The graph on the right hand side, also Fig 2.5, represents the comparison in computations per kWh of computers over the span of 70 years.

Implementing technology upgrades on outdated and faulty servers as recommended significantly impact the TCO. Hardware updates may transpire as the most obvious way to improve a DC's efficiency. This may not be accessible to minor DCs due to insufficient capital to invest. A motivation for and objective of this project is to provide administrators with in-depth insights into their EAC and to identify the length of time until they would show a positive return from such a significant financial investment.

2.3 The Climate Crisis

Human activity involving the production and burning of coal, oil, and gas are generating greenhouse gas emissions at a record high. These emissions are predicted to increase, amplifying the billions of tons of CO2 that are released into the atmosphere every year. The impact of climate change is irrefutable. However, there is still time to suspend further destruction of the atmosphere. This will require fundamental transformations in the harvesting of the land, transportation of goods and powering of economies. New efficient technologies can diminish net emissions and initiate a cleaner world. Research indicates that technological solutions already exist the elimination of more than 70% of today's emissions.

2.3.1 The United Nation's Sustainability Goals

Starting in 2015, the United Nations released a framework which determined 17 Sustainable Development Goals (SDG) [21]. The UN SDG are a pledge to the entire population of the planet with the aim of developing the quality of all lives by the year 2030. They are standards that encompass ending poverty, preserving the environment, and establishing world-wide peace and prosperity. Goal 13, Climate Change, has a strong correlation to role of DCs in modern society. The Paris agreement, signed the same year and in conjunction with the SDGs, was the first ever legally binding climate change agreement. It claimed that yearly global temperature increases are not sustainable for the well-being of the planet and set a goal of limiting the escalation to 1.5°C [22]. The most recent report by the UN on the status of goal 13 stated that the average global temperature increase recorded was roughly 1.11 ± 0.13 °C compared to pre-industrial level at the beginning of 20th century [21].

To be successful, greenhouse gas emissions must reach their max before 2025 to then decline by 43% by 2030, before again declining to net zero by 2050. The report continues to outline that carbon dioxide (CO2) emissions decreased by roughly 5% two years ago before the Covid 19 pandemic which caused a surge of 6% in 2021 [21]. Considering the decrease in 2020 was the equivalent of halting almost 2 billion metric tons of CO2 being emitted, the largest decline since 2009. This is a powerful reminder that we most act before it's too late.

2.3.2 The Climate Crisis and Data Centres

The climate crisis that looms over our heads is a consequence of emitting CO2 into the atmosphere. The main culprits for the pollution of the earth's atmosphere are deforestation, cement production

and the burning of fossil fuels [23]. Without significant progress in reducing CO2 emissions, the warmer the planet will become, and the ramifications of global warming will intensify. Declarations such as the Paris Climate Agreement [22] set by the UN and the European Green Deal [24] were authorized with the hope to stop the world emission rate increasing, ultimately to level-off in 2030. However, the targets in both the 2030 Climate Target Plan [25] and the 2050 Long Term plan were reevaluated to lessen the rise in global temperatures. The major amendment shows the previous target of reducing greenhouse gas emissions by 40% altered to a new target of 55% by 2030, which the EU mentioned would require action in all sectors of the economy [25]. In 2017, it was found that the Information and Communication Technology sector (ICT) consumed 2% of all CO2 emissions worldwide [26]. An evermore alarming statistic from [27] warned that the figure of 2% could rise all the way to 14% by 2035, without immediate intervention.

2.3.3 The European Code of Conduct for Data Centres

To encourage the implementation of the 2030 Climate Target Plan in DCs, the EU commissions a release of The Code of Conduct for Data Centres [28] each year. The initiative that was originally launched in 2008 aims at promoting knowledge and awareness of the energy demand within the data centre. For 2020, the EU predicted this sector's energy consumption would continue to grow and reach 100 TWh for that year [29]. Although the Code of Conduct for Data Centres is not enforced by law and does not provide any funding, the EU endorse the following benefits to participants [28]:

1. Save money by carrying out cost effective improvements

2. The annual Data Centres Code of Conduct Awards acknowledges DCs that greatly reduce their energy consumption

3. Participants are part of a European program to reduce CO2 emissions, so should be considered a "green or environmental conscious company" as well as get free publicity about their participation from public authorities including the Commission

4. Participants will be invited to a Code of Conduct Stakeholder Forum to review progress and further develop the Code of Conduct

2.4 Summary and Conclusions

In recent years, energy efficiency has emerged as one of the most important design requirements for modern DCs, as they proceed to consume vast amounts of electrical energy. The high operating costs for the workload completed by a DC's computing resources signifies the emission of considerable volumes of CO2 into the environment, with much of this energy being wasted or underutilized by DC due to mismanagement and an overall lack of awareness. This research concludes that the pressing issue of climate change is a persuading factor in the need for change in the IT sector. Of the 8 million or so DCs worldwide, smaller DCs require a greater insight into the impact of their DC. Access to a web based tool to analyse their power and carbon consumption will facilitate administrators to rearrange and reschedule process in the appropriate windows for electrical and carbon efficiency.

Chapter 3: Overview of Approach

As stated in the Chapter 1, the overall objective of this project is to design and implement a web dashboard which will assist administrators with the daily operations of their DC. Evidently, the majority of administrators are unaware or oblivious to the DC's processes and their repercussions in terms of carbon emissions. As governing bodies implement strategies to reduce global carbon emissions inline with the 2030 Climate Plan [25], DCs are required to diminish their carbon footprint. Since large-scale multinational sites have the resources and infrastructure to adhere to this regulations, the dashboard will provide insight for minor scale DCs into the carbon emitted and their power expenditure.

3.1 Previous Work

This project is not a stand alone body of work. It will build on the work by Daniel Houlihan who completed his project, Datacentre Management Tool [30]. His undertaking saw him develop a web-based application which presented administrators of small-scale DCs with insights into the power consumption of their servers. The application provided statistics that could be illustrated through graphs depicting set audit periods, energy usage and carbon footprint of the DC. He coded this tool through Python using Django[31], a high-level web framework along side the Papillon API software[32].

3.2 Papillon

Papillon is a software tool developed by Beeyon [33] for tracking the power consumption of computer systems among other metrics detailed below in Section 3.2.1. Papillon is an acronym for **P**rofiling and **A**uditing of **P**ower **I**nformation in **L**arge and **L**ocal **O**rganisational **N**etworks. It uses no physical hardware measurement devices, instead it utilises patented modelling technology. It runs simultaneously on the server providing over 98% accurate to measure the power consumption. This also means Papillon can calculate the consumption of all processes on the server separately, something physical metering is incapable of. Each server type has its own power model which is created by Beeyon and allows for such high accuracy. The client-server hierarchy of a DC enables Papillon to deploy agents on all client servers, which communicate with the singular master server that accommodates the power models. Figure 3.1 depicts the architectural hierarchy of one master server and numerous client servers being monitored. For Papillon to function, agents are installed on all of the client servers. The agents are responsible for disclosing *'Performance Data Packets'* regarding server activity back to the master server which hosts the power models and a database.

The information returned is stored in a database which is housed on the master server. The master server then calculates and saves all metrics. Papillon is designed to support a REST based web application. Once Papillon installed, relevant data about the DC can be accessed with the API endpoint on a local network.

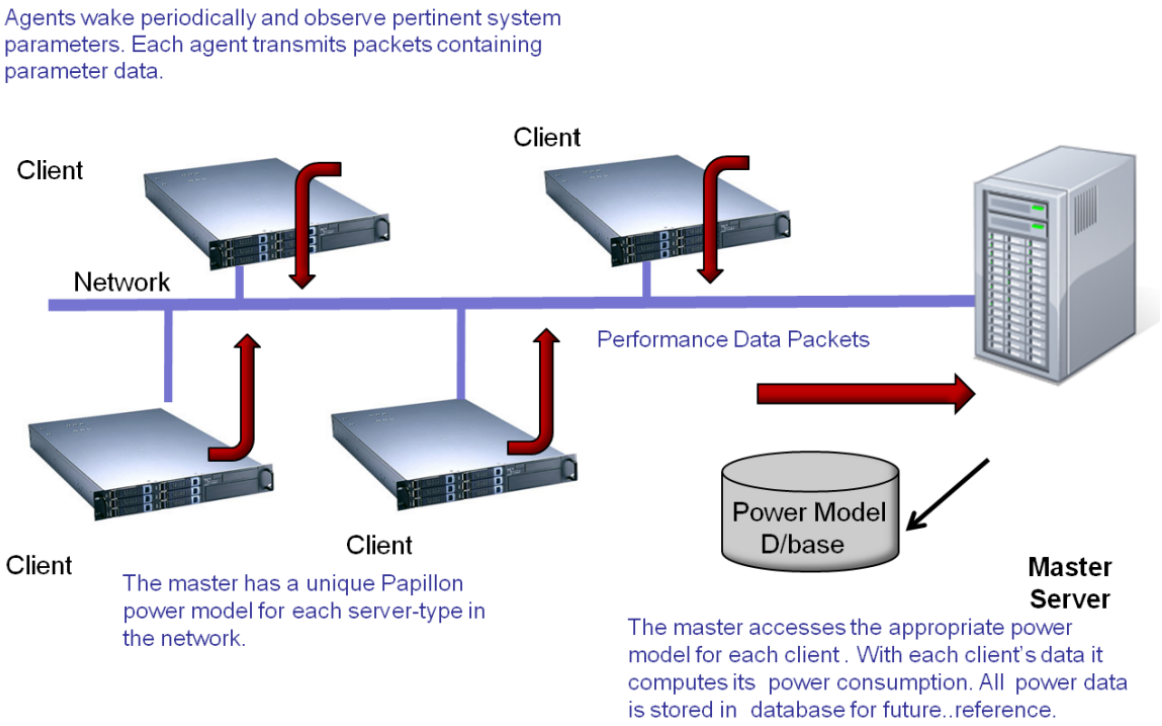

Figure 3.1: PAPILLON Architecture Overview

3.2.1 Papillon API

The usage of the Papillon API [32] is crucial for the operation of the proposed web dashboard. It provides essential information regarding available DCs, floors, racks, and hosts, which are required to initiate the mapping process. The retrieved properties are then utilized to conduct additional API calls to retrieve power and energy consumption data for each host. The DC structure is organized in a hierarchical manner, forming a tree-like structure of nodes, starting from DC to floor, rack, and host, as illustrated in Figure **??**. A DC can have multiple floors, with each floor can have multiple racks, and each rack can have multiple hosts.

As mentioned before, Papillon has been developed following the RESTful architecture style which offers several benefits. RESTful APIs are inherently scalable, allowing the system to handle multiple concurrent requests efficiently and enabling easy scaling as the number of servers or requests increases. In general they are lightweight and stateless, making them suitable for resource-constrained environments and reducing resource usage. The widespread adoption of RESTful APIs in the industry provides a supportive ecosystem of tools and frameworks, simplifying development, maintenance, and integration with other systems or third-party applications.

After installing and running Papillon on the master server, real-time power and energy data of both the master and clients can be accessed through an API endpoint on the local network. Data can be retrieved from the Papillon API through XML objects via the base URL:

http://<**MasterIPAddress**>:8080/papillonserver/rest

The **MasterIPAddress** refers to the IP address associated to the master server unique to each DC. To retrieve XML objects using the API, the request headers must include the appropriate specification. The data necessary for the dashboard can be obtained by accessing four different endpoints in Papillon.

3.2.2 Retrieving Data

In Python, the JSON (JavaScript Object Notation) import is used for data interchange format that is human-readable and easy for machines to parse. The JSON module in Python provides functions for encoding Python objects, serialization of Python objects into JSON strings and deserialization of JSON strings into Python objects. The two JSON functions used to achieve this were:

- **json.dumps():** This function takes a Python object as input and returns a JSON string representing that object. It serializes the Python object into a JSON format.
- **json.loads():** This function takes a JSON string as input and returns a Python object representing that JSON data. It deserializes the JSON string into a Python object.

By importing the JSON module in Python, these functions were utilised to easily convert data between Python objects and JSON strings. Working with the Papillon API that sends data in JSON format, meant easy conversion of the data to Python objects for processing and conversion of Python objects to JSON strings for storing data.

The first URL utilised, returns a list of all available DCs on the master IP address. This includes additional information about the DC such as its description and location, but must importantly it includes the DC's ID.

http://<**MasterIPAddress**>:8080/papillonserver/rest/datacenters

An example of XML file returned showing the DC's associated to a certain IP address :

```
<datacenters>
    <datacenter>
        <id>267</id>
        <name>dc1</name>
        <description>dc1</description>
        <latitude>0.0</latitude>
        <longitude>0.0</longitude>
    </datacenter>
    <datacenter>
        <id>268</id>
        <name>dc2 test</name>
        <description>test datacenter</description>
        <latitude>0.0</latitude>
        <longitude>0.0</longitude>
    </datacenter>
</datacenters>
```

The second URL returns a list of all available hosts on the specified IP address and DC ID. This includes information about the DC, floors, racks, and hosts. A host can be a physical server or a virtual machine with its own IP address, all this data can be retrieved from the database.

http://<**MasterIPAddress**>:8080/papillonserver/rest/datacenters/ <**DatacenterID**>/allhosts

An example of XML file returned showing the hosts information associated to a certain DC :

```
<hostExtendeds>
    <hostExtended>
        <datacenterId>267</datacenterId>
        <datacenterName>dc1</datacenterName>
        <datacenterDescription>dc1</datacenterDescription>
        <floorId>291</floorId>
        <floorName>fl1</floorName>
        <floorDescription>fl1</floorDescription>
        <rackId>294</rackId>
        <rackName>rack1</rackName>
        <rackDescription>rack1</rackDescription>
        <hostId>284</hostId>
        <hostName>vm1</hostName>
        <hostDescription>vm1</hostDescription>
        <modelGroupId>186</modelGroupId>
        <modelGroupName>PMG-Dell-PowerEdge-R711-V1.0</modelGroupName>
        <hostType>VM_SERVER</hostType>
        <IPAddress>127.0.0.1</IPAddress>
        <processorCount>4</processorCount>
        <VMCount>1</VMCount>
    </hostExtended>
    ...
</hostExtendeds>
```

The third URL returns power information for a specified host within a given time range. Since the measurement is for a specified host, data about including the floor and rack on which it is located are included in the URL. The timestamp data is required to be in UNIX format. The XML returned provides information about the amount of watt-hours the host consumed when it was active, divided into minute intervals accompanied by the corresponding timestamp.

http://<**MasterIPAddress**>:8080/papillonserver/rest/datacenters/ <**DatacenterID**>/floors/

<**FloorID**>/racks/<**RackID**>/ hosts/<**HostID**>/power?starttime=<**Start**>&endtime=<**End**>

An example of XML file returned showing the hosts power consumption in one minute intervals :

```
<powers>
    <power>
        <power>1.725646627409716</power>
        <timeStamp>1683666905</timeStamp>
    </power>
    <power>
        <power>1.7256466276286662</power>
        <timeStamp>1683666965</timeStamp>
    </power>
    <power>
        <power>1.7256466274232898</power>
        <timeStamp>1683667025</timeStamp>
    </power>
    ...
</powers>
```

Finally, the fourth URL returns activity information for a specified host within a given time range. Similar to the power URL, specific data about the host is included in the header. However the XML returned discloses more in-depth analysis. It provides three statistics for the previous minute:

- Stat1: Average CPU utilization as a percentage
- Stat2: Disk I/O (pageins & pageouts)
- Stat3: Network I/O (bytesin & bytesout)

In addition to their respective timestamp, the activity breakdown of running applications on the host system is also divulged.

http://<**MasterIPAddress**>:8080/papillonserver/rest/datacenters/ <**DatacenterID**>/floors/

<**FloorID**>/racks/<**RackID**>/ hosts/<**HostID**>/activity?starttime=<**Start**>&endtime=<**End**>

An example of XML file returned showing the hosts activity breakdown of a specified host :

```
<activities>
    <activity>
        <id>4351</id>
        <hostId>284</hostId>
        <power>1.725646627409716</power>
        <powerMode>AC_DEFAULT</powerMode>
        <stat1>0.8904076325578282</stat1>
        <stat2>344452.0</stat2>
        <stat3>1232896.0</stat3>
        <timeStamp>1683666905</timeStamp>
        <allApps>940.0</allApps>
            <apps>
                <app>
                    <appId>19356</appId>
                    <activityId>4351</activityId>
                    <name>blaster#01557905142</name>
                    <cpu>570.0</cpu>
                </app>
                ...
            </apps>
    </activity>
    ...
</activities>
```

3.3 Django

Django is a high level Python web framework that provides the swift development of secure, maintainable, and scale-able websites [31]. On top of this it is free and open source. Django was utilized by Daniel in his project to great effect.

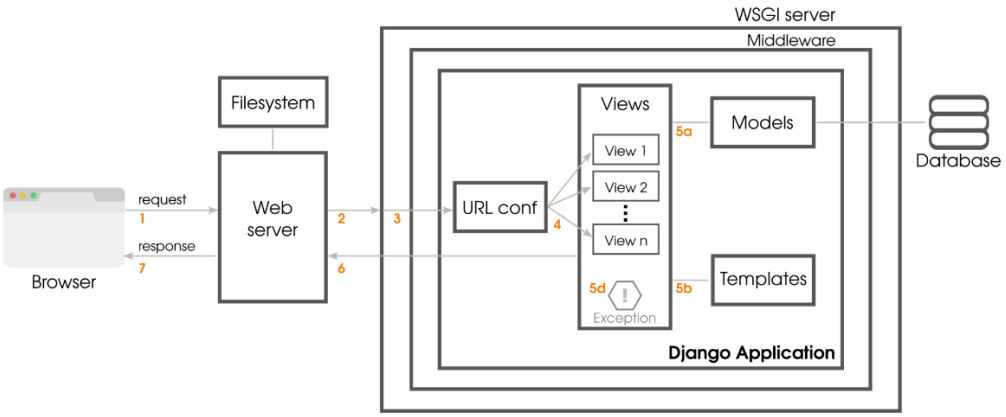

Figure 3.2: Overview of Django Architecture

Django's large feature set of more than 10,000 Django packages, the framework covers nearly all aspects needed to build a web application. Packages include APIs, content management systems, user authentication, form validation and CAPTCHA protection. It also provides an object-relational mapper (ORM) which converts traditional database structure into Python classes to make it easier to work within a fully Python environment. Django-MySQL supports the JSON data type and related functions. Django makes dynamic HTML with a built-in templating engine called the Django template language (DTL).

3.3.1 Storing Data

By default, Django offers built-in support for SQLite3. The dashboard utilises this SQLite3 database to store relevant data retrieved using the Papillon API. Once the dashboard is running alongside the DC with Papillon installed, the administrator is prompted to load a new DC. Data about the selected DC over a specified period of time including, the hosts, their tagged flexibility and hourly analysis are stored. All the configured DCs are saved to the database, regardless of whether they are the currently selected instance or located on the master IP address. The user only provides the current master IP address, the currently selected instance, and the configurations table. All other information is retrieved from the Papillon API.

3.4 Dummy Data Centre

The use of the Papillon API required a server running the software. Following the same process of last years project, creating multiple virtual machines running the Linux Mint 18 operating system with Papillon installed. One of the virtual machines was designated as the master server, responsible for storing the database and providing the API. The other virtual machines were configured with

a Papillon client software that calculated usage data and transmitted it back to the master server. This virtualized setup emulated a real DC environment with multiple hosts, analogous to physical servers. While it was feasible to scale the simulation by adding more virtual machines, practical limitations related to RAM utilization constrained the number of virtual machines. Despite these limitations, the virtualized environment was successfully operational, and allowed the objective of developing a web based dashboard that could be deployed on an a DC to be achieved.

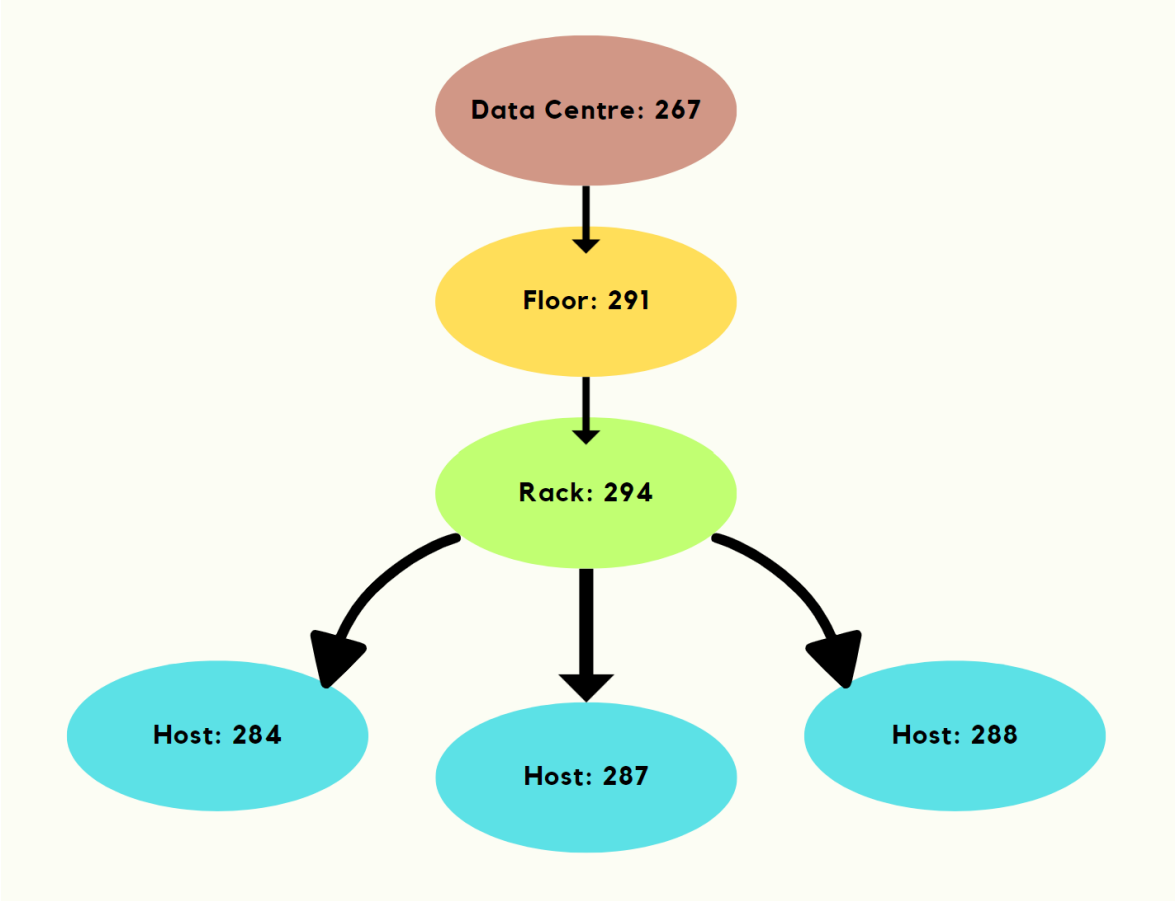

Figure 3.3: Structure of the Dummy Data Centre with 3 virtual hosts.

3.5 Calculations

To identify the severs in the DC that are running efficiently, different metrics which cover the emission of carbon dioxide and the operational costs are calculated over hourly segments. Getting a consistent measurement involved accumulating the hour-by-hour consumption of each host over a period specified by the administrator during configuration. This period can be range from a single day to the compound of entire months. During a period of analysis, for each single day the energy consumed by a host during each one of the twenty-four hour divisions is totaled. For example, the final value of the 1:00a.m - 2:00a.m segment will equal all energy consumed everyday at the specified hour of 1:00a.m - 2:00a.m. An extended time frame will provide insights to the true consumption of the hosts. In other words, a longer analysis period will cultivate even more dependable observations for the administrator.

The tariff rates are designed to be highly adjustable depending on the demands of the administrator. Due to the volatile nature of both the carbon conversion rate and the cost conversion rate for

expressing the consumed energy in terms of these metrics, the administrator can choose the high and low tariffs, the average tariff will be calculated automatically. The daily time frame is divided into two 12 hour periods of ***High-Demand*** **(08:00 - 18:00)** and ***Off-Peak*** **(19:00 - 07:00)** is set and cannot be altered.

1. Average Carbon Tariff: Represents the carbon consumption. The calculation is expressed in kg/C02, the amount of carbon dioxide in kilograms. This figure is calculated by finding the total amount of energy consumed in kilowatt-hours (KWh). Which is converted to kg/C02 using the average carbon tariff rate. This tariff, which can be adjusted by the administrators, is acquired by averaging the low and high carbon rates since each rate has an equal number of hours (12 each).

$$AverageCarbonRate = \frac{HighCarbonRate + LowCarbonRate}{2} \tag{3.1}$$

$$CarbonConversion = TotalKWh \cdot AverageCarbonRate \tag{3.2}$$

2. High Carbon Tariff: Follows the same format as the Average Carbon Tariff and the rate can be adjusted by administrators.

$$HighCarbonTariff = TotalKWh \cdot HighCarbonRate \tag{3.3}$$

3. Low Carbon Tariff: Follows the same format as the Average Carbon Tariff and the rate can be adjusted by administrators.

$$LowCarbonTariff = TotalKWh \cdot LowCarbonRate \tag{3.4}$$

4. Average Cost Tariff: Represents the electricity cost. The calculation is expressed in terms of the Euro currency. This figure is calculated by finding the total amount of energy consumed in kilowatt-hours (KWh). The total KWh are then expressed in terms of the DC's PUE and the average cost rate. Like the carbon tariffs, the cost tariffs can be adjusted by the administrators. The average cost is acquired by averaging the low and high cost rates.

$$AverageCostRate = \frac{HighCostRate + LowCostRate}{2} \tag{3.5}$$

$$EnergyCost = TotalKWh \cdot AverageCostRate \cdot PUE \tag{3.6}$$

5. High Cost Tariff: Follows the same format as the Average Cost Tariff and the rate can be adjusted by administrators.

$$HighCostTariff = TotalKWh \cdot HighCostRate \cdot PUE \tag{3.7}$$

6. Low Cost Tariff: Follows the same format as the Average Cost Tariff and the rate can be adjusted by administrators.

$$LowCostTariff = TotalKWh \cdot LowCostRate \cdot PUE \tag{3.8}$$

Chapter 4: Proposed Solution

4.1 Main Objective

The main objective of the DC management dashboard is to provide a comprehensive view of the DC's operations as regards hourly consumption. The dashboard will allow users to tag applications and servers based on their flexibility, highlighting of which are more applicable to changing circumstances. Additionally, the dashboard will provide hourly insights into the DC's operations, including hourly tariff rate changes. This information will enable users to optimize their resource usage and minimize costs. Overall, the dashboard aims to streamline DC management, improve operational efficiency, and ensure cost-effective operation of the DC.

4.2 Implementation Plan

Due to Daniels success and the observations regarding the proposed approach mentioned in the previous Chapter 3, this project uses the same web framework Django [31]. In essence, this project aims at enhancing the functions Daniel has already created. By extending the capabilities of the audit and carbon measurement functions, it will allow administrators using the tool to set hourly electricity tariff rates as well as incorporating volatility in to the carbon intensity measurement. The web application will consist of multiple tools that reflect the objectives of this project. Visually representing the analytical enhancement (Electricity and Carbon), as well as another tab to depict the flexible processes and alternative rates to execute them.

The web application will contain forms that the administrator can submit to modify the database. The main form will configure the a single DC from the available DCs including it's PUE, start time, and end time. Once configured, other forms to alter carbon and cost tariffs can be submitted. The SQLite3 database will be used as it is supported by Django. Apart from Django back-end architecture that will support the web app, front-end will be developed using HTML, JavaScript, and CSS.

4.2.1 What is a Flexible Host?

A major objective of this dashboard is to assign tags to hosts in the DC to identify their flexibility. But, what is meant by a *flexible host*? In this case, a flexible host is a process identified by the administrator as having no set operating time frame. In other words, the host can operate at only time during the 24 hours of the day. Time sensitive processes are a common occurrence during the daily functions for the majority of DCs, meaning some process have to run on a certain schedule which can overlap with high-demand hours. Ideally, for maximum financial and environmental benefits all hosts would be flexible and only operate during the off-peak period.

4.2.2 Manage Host Flexibility

Regarding the flexibility of hosts running on the DC, the dashboard will allow users to select whether or not host can be run at an alternate time in a 24 hour period. The dashboard is solely an analytical tool and doesn't have the capability to change application operating patterns. It will prove calculations based on the flexible tag assigned to each host. Financial costs and CO2 emissions of hosts the user tags as *'flexible'* will be highlighted, then allowing for the high and low tariff rates to be modified. Best and worst case scenarios will be calculated using the carbon tariff rates as well as the cost tariff rates. The dashboard will display these metrics to provide a comprehensive overview of the DC's financial and environmental impact. By analyzing these metrics, users can identify fields of interest that can be adjusted to reduce energy consumption and in turn limit emissions. Ultimately, the dashboard suggests optimizations and provides recommendations to improve efficiency by rescheduling servers.

4.2.3 Analyse Consumption

Since the dashboard aims to enhance administrators' comprehension of daily DC operations, the energy consumed will be presented in hour-by-hour segments. Not only will the energy consumption be depicted in the form of a graph but also separate graphs for the operational cost and the carbon emitted. The analysis will portray the consumption of each host associated with the configured DC. The representation of this data in graph format will assist the administrator by identifying hosts performing during peak hours, uncovering areas in the DC of potential resource waste. The graphs will have clearly defined hour segments with corresponding colouring to illustrate the contrast between high-demand and off-peak hours. Each host will be assigned a colour and this will be labeled in the graph legend.

4.2.4 Alternative to Reducing Workload

One way a DC can reduce its operational costs and lower carbon emissions is by running during low tariff hours. The electrical grid typically experiences lower demand during off-peak hours, resulting in lower electricity prices. By scheduling their operations to coincide with these off-peak hours, DCs can take advantage of the lower tariff rates, reducing their electricity consumption and lowering their carbon emissions. The same methodology can be implemented to reduce the DC's carbon emissions. During off-peak hours, there is typically lower overall energy demand as commercial and industrial activities are minimized, and residential energy usage decreases. This reduced demand means that fewer power plants and energy-intensive facilities need to be operational, resulting in lower carbon emissions.

Renewable energy sources such as wind and solar power can have fluctuating generation patterns. Since renewable energy generation produces little to no carbon emissions, increased reliance on renewable energy during off-peak hours helps reduce overall carbon emissions. DCs can also invest in their own renewable energy sources to supplement their energy needs and further reduce their carbon footprint. By combining these strategies, DCs can significantly reduce their environmental impact while also benefiting from lower energy costs.

4.2.5 Hourly Tariffs

DCs consume a vast amount of energy, and as energy prices rise, the cost of running these facilities increases accordingly. The hourly tariff rate changes can have a significant impact on the total electricity consumption cost of a DC, particularly during peak usage hours. As a result, having hourly insights into the DC's power consumption patterns can help identify opportunities to reduce energy consumption and costs. For instance, during off-peak hours, it is possible to operate the DC at a lower cost since the energy tariffs during these hours are relatively low. In contrast, during peak hours, the energy tariffs are high, making it expensive to operate the DC. As a result, the DC's administrators can optimize their energy consumption by running the DC during off-peak hours, thereby minimizing energy costs and reducing overall carbon footprint.

The dashboard developed for DC management provides real-time insights into the DC's power consumption patterns, allowing administrators to identify opportunities to reduce energy consumption during peak hours. These insights can help identify applications and servers that drain resources at unnecessary times, enabling DC administrators to optimize their energy usage and minimize costs. By running the DC during low tariff hours, administrators can reduce their overall carbon footprint while still meeting the demands of the organization. Ultimately, hourly electrical tariffs are a crucial aspect of DC management, and by using them wisely, administrators can reduce costs, minimize environmental impact, and ensure the efficient operation of their facilities.

4.2.6 Off-Peak & High-Demand Hours

Off-peak hours generally refer to periods of low electricity demand. These are typically times when energy consumption is relatively lower, such as late at night, early morning, or during weekdays when most people are not actively using electricity. Off-peak hours are characterized by a reduced load on the electrical grid as fewer appliances, businesses, and industries are consuming electricity. As regards carbon emissions, renewable energy sources generally contribute a higher proportion of the electricity supply during off-peak hours due to their relatively consistent generation patterns. Since electricity demand is lower and more renewable energy sources may be used, carbon emissions associated with electricity generation are generally lower during off-peak hours.

High-demand periods, on the other hand, occur when there is a significant increase in electricity consumption. These periods typically coincide with peak hours, which are usually during weekdays when people are at work, and in the evenings when they return home. High-demand periods are characterized by a larger number of appliances, lighting, air conditioning, heating, industrial processes, and other activities that contribute to higher electricity usage. Fossil fuel-based power plants, which may have higher carbon emissions, are often relied upon to meet the increased demand. Consequently, carbon emissions associated with electricity generation tend to be higher during high-demand periods.

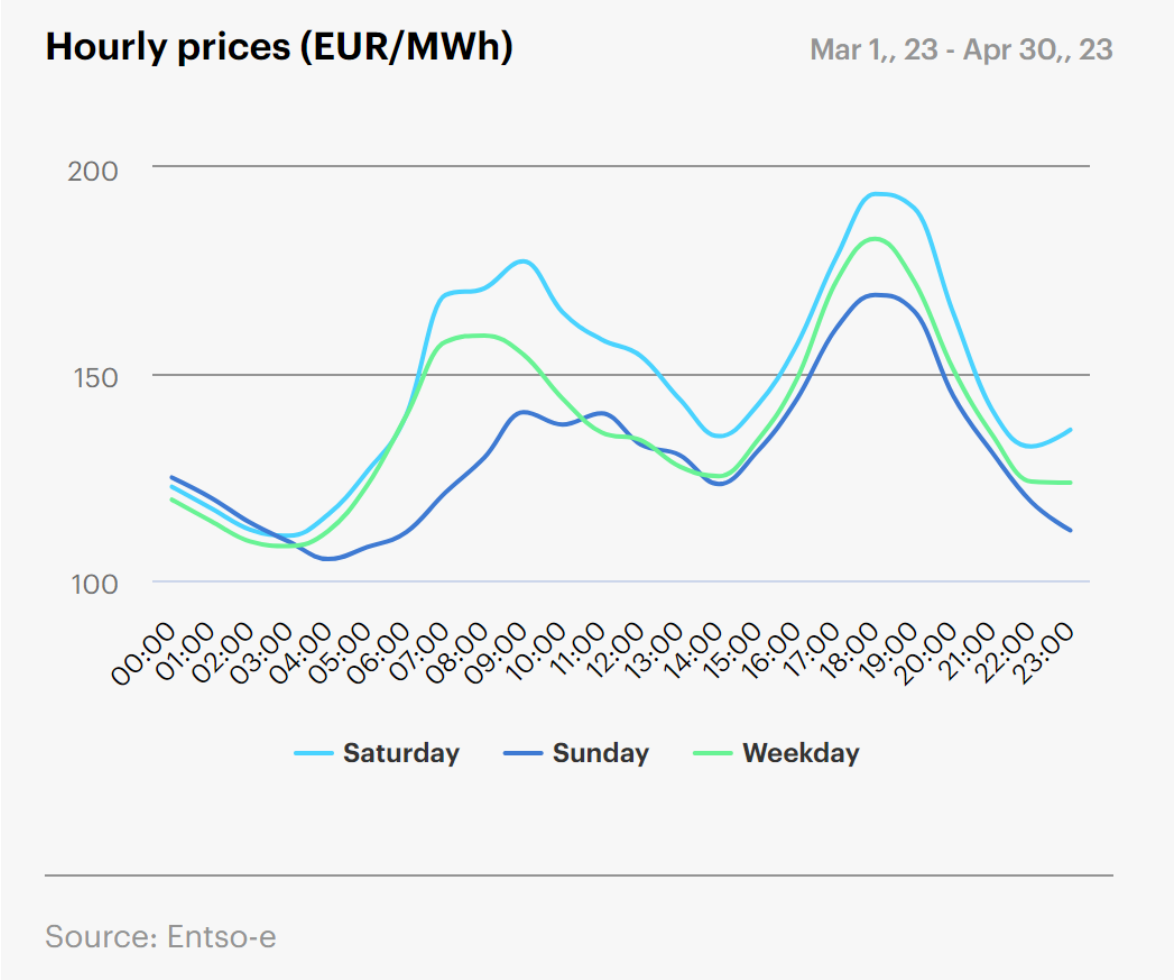

Figure 4.1: Average hourly price of electricity from the 1/03/2023 to 30/04/2023 in Ireland taken from the IEA (Real Time electricity Tracker) website [34].

In the context of the dashboard, the daily time frame for analysis will be divided into two 12 hour periods of ***High-Demand*** **(08:00 - 18:00)** and ***Off-Peak*** **(19:00 - 07:00)**, as mentioned in Section 3.5.

4.3 Testing

The implementation of unit tests focuses on testing individual services within the dashboard. Services represent the functions utilized by views to perform operations such as creating, updating, and deleting objects in the database via API access and data manipulation. These unit tests aim to validate the functionality of each method, regardless of the application's current state.

A key principle followed in the testing approach is the "zero, one, many" rule, which applies where applicable. In the context of unit testing, the "zero" scenario refers to testing the method with null or empty input. This helps verify if the method handles such cases gracefully and avoids any unexpected errors or exceptions. The "one" scenario involves testing the method with a single input instance. This is done to validate the functionality when dealing with a single data element, ensuring that the method performs as expected in this specific context. The "many" scenario involves testing the method with multiple input instances. This allows for assessing the behavior and performance of the method when processing a collection of data or handling repetitive

operations. It helps identify any potential issues related to scalability, performance degradation, or incorrect handling of multiple instances. By covering these three scenarios, the correctness and robustness of the method across various input cases is approved. It helps ensure that the method behaves consistently and correctly regardless of the specific input values or number of instances encountered.

Unit tests were also conducted on database models to ensure the accuracy of data types and attributes. Additionally, data input forms undergo unit testing to ensure the proper sanitation of data which the user might insert. By adhering to this rigorous unit testing approach, the system can verify the correctness and robustness of individual services, models, and forms, thereby enhancing the overall quality and reliability of the application.

Chapter 5: Dashboard Design & Results

5.1 Dashboard Overview

The dashboard content is divided between three separate pages under the following URL headers:

1. dashboard/ : Home - Configure and load DCs
2. dashboard/tags/ : Tags - Assign the flexibility of DC's hosts
3. dashboard/analysis/ : Analysis - Hourly breakdown of each hosts' energy, cost and carbon footprint

5.2 Home Page

On the homepage, users can view the list of configured DCs and available DCs. Additionally, they can change the master IP address, configure a new DC, update an existing DC, and delete an existing DC. Figure 5.1 represents the initial page users experience when they first run the dashboard. For a more detailed image of the homepage connected to the Papillon master server, see Figure 5.4 on page 29.

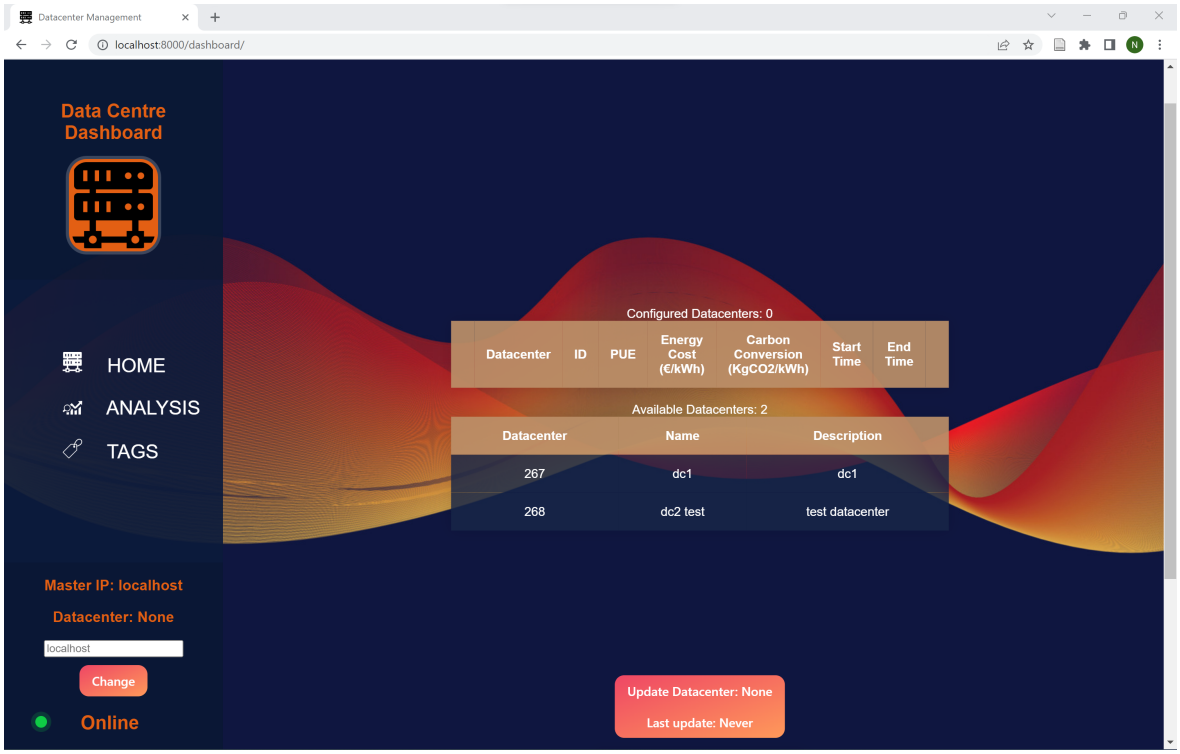

Figure 5.1: Initial Home Page when no DCs are configured

The first table displays the configured DCs, including the date and PUE values that the user inputted during the configuration process. The carbon conversion rate and the cost conversion rate are also displayed and are set to 1.1 as default values. Users can configure any number of DCs, and the currently selected DC is highlighted in grey. The currently selected DC is also displayed on the bottom left-hand side of all pages inside the navigation bar. The second table displays the available DCs on the Papillon master, making it possible to set up multiple DCs from the same master server with a different duration or PUE.

5.2.1 Configuring

The master IP address is required to locate all DCs hosted on the master server. By default, the master IP address is set to "localhost" but can be changed by the user at any time, see Section 5.2.3. Once these DCs are found, another API call is made for each of them to obtain the layout of each DC. Hosts, racks, floors, and even DCs can be added to Papillon running on a single IP address at any point in time, so this process is done automatically (before configuring) to capture any changes.

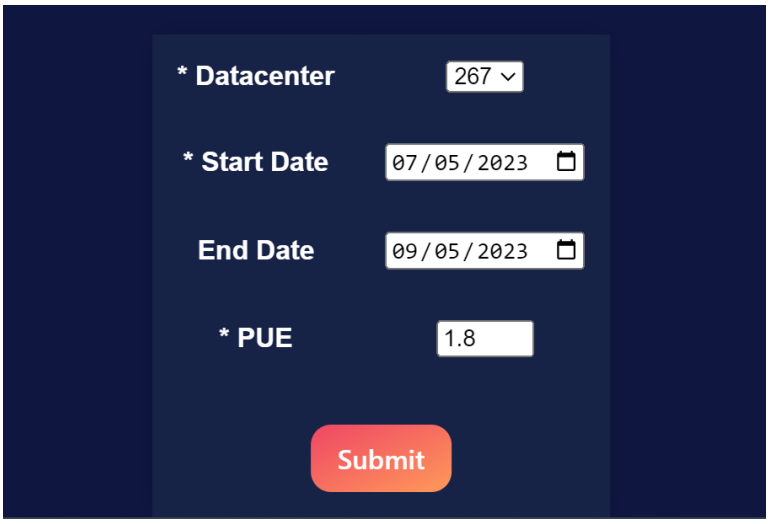

Figure 5.2: Form to configure new DC on the Home Page.

The database stores the DC, floor, rack, and host objects, containing descriptive attributes such as ID, name, and description. However, power and energy data has not been obtained at this stage. When the administrator initiates the configuration of a DC, the DC's details are utilized to establish API calls to the Papillon master. The necessary variables for these calls include the Master IP address, DC ID, Floor ID, Rack ID, Host ID, Start time, and End time. Except for the start and end time, these variables are retrieved by iterating through the tree structure and constructing a URL for each host node. The URL incorporates the host's ID along with the parent's values (floor ID, rack ID, DC ID). The start and end time are user-specified via the configuration form, shown by Figure 5.2. These API calls are made twice for each host in the DC, once to obtain power data and once for activity data. By analyzing the activity data, the CPU utilization can be determined, while the power metrics allow for the calculation of energy, cost and carbon consumption.

The activity data retrieved provides a high level of granularity. To compute the average CPU utilization over the specified period, the 'stat1' values are summed and divided by the number of server responses. The average value is then updated in the corresponding host object. This method

was designed by Daniel Houlihan[30] and provides insight into the under-utilisation of hosts in the DC. By including it in this dashboard, users are able to understand which servers' CPU capacity is:

- Under strain
- Not being fully capitalised
- In a state of *Comatose (CPU usage less than 3%)*

5.2.2 Update Button

The update button allows the administrator to revise the time frame of the currently selected DC. It re-adjusts the DC's end time to became the exact time the button is clicked by the user. This eliminates the need for the administrator to reenter data or even configure the same DC as a new instance with a different end time.

5.2.3 Changing IP Address

In the context of Papillon, the IP address of the master node plays a significant role in the system's configuration. By default, the master node's IP address is set to "localhost," which refers to the current machine. However, it is possible to modify this IP address to accommodate scenarios where multiple master nodes exist within the same network. Figure 5.3 illustrates the interface where the IP address of the master node can be adjusted.

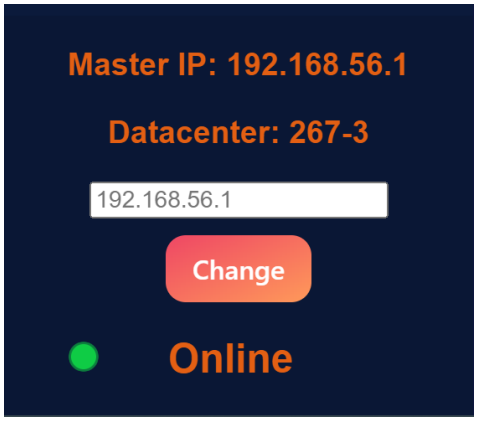

Figure 5.3: Status Indicator in the navigation bar

Changing the master IP address does not result in the removal of any previously configured DCs. Therefore, users have the flexibility to switch back and forth between different master IP addresses without losing their configured DCs. This feature proves useful in situations where there is a need to manage multiple Papillon instances within the same network environment. The master IP address serves as a crucial component in making API calls to Papillon for retrieving server information. When the master IP address is accessible and hosts a functioning Papillon master, a green light indicator signifies that the master is online and responsive. Conversely, if the entered IP address does not correspond to a running Papillon master or the master server fails to respond, a red light indicator denotes that the master is offline. The administrator can also easily identify which configured DC is currently selected in addition to the IP address. The status of the dashboard is displayed on all pages for convenience and coherence.

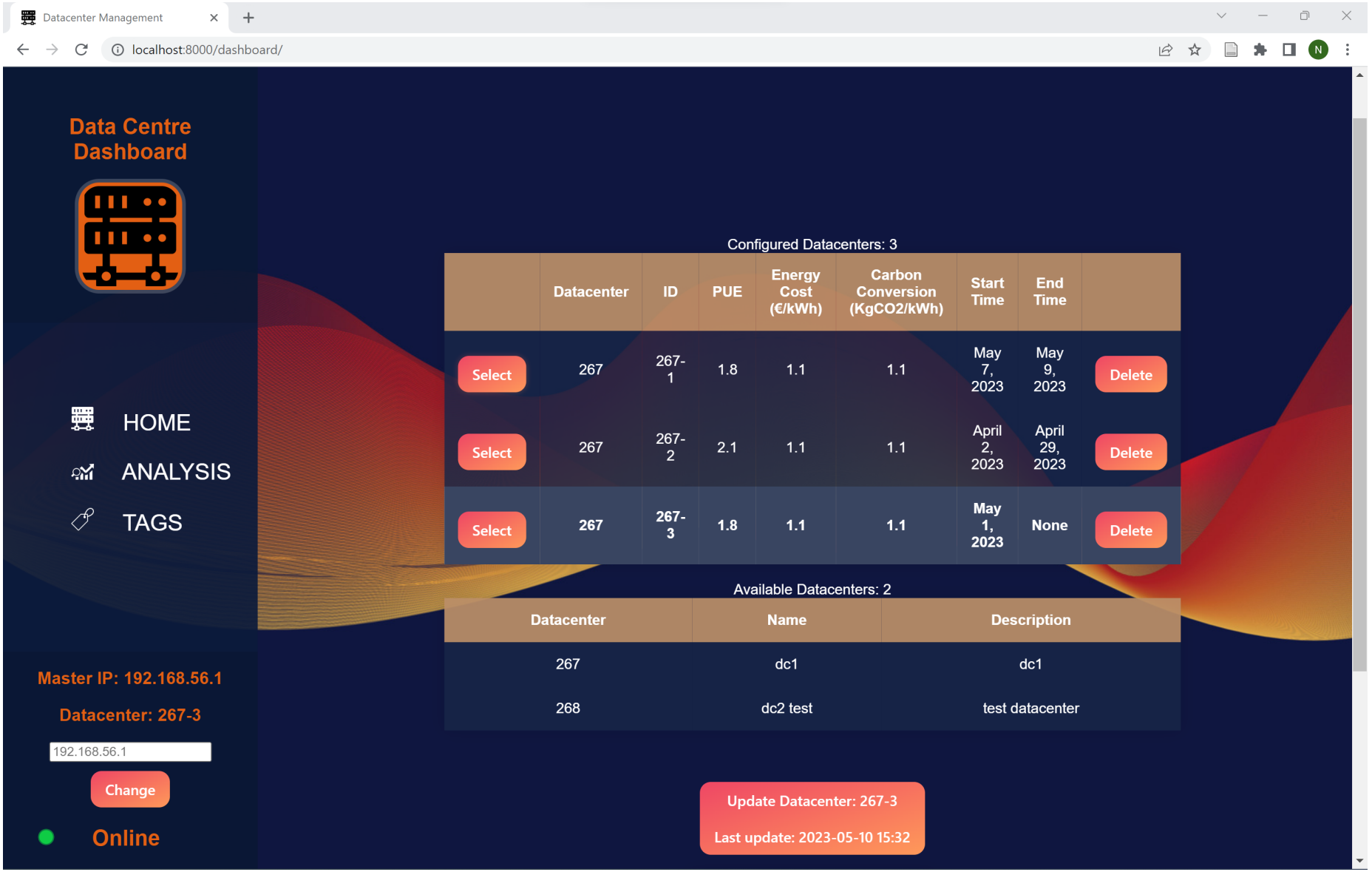

Figure 5.4: Home Page with configured DCs

5.3 Analysis Page

The Analysis page depicts three graphs concerning the energy, cost and carbon associated with the configured DC. The data represents the accumulation of the aforementioned metrics from the start time to end time segregated into hourly intervals. The user can choose to view either the *Energy, Carbon* or *Cost* graph using the tabs located on top of the graph. The graphs are plotted using **mathplotlib** in Python as it is a library I am experienced using. When a DC is configured, the graphs are plotted immediately and saved as an image. To save this image into the SQLite database, it is encoded using the **base64** module.

This encoding converts binary image data into a text format that consists of ASCII characters. This makes it suitable for transmitting images through text-based protocols such as HTTP, which typically only support text data. Encoding the graph images into base64 allows for their transmission as part of Django API requests to be display on the web page templates. However, one drawback to saving the graphs as images is the fact that they aren't dynamic. This required me to included a refresh button for when the administrate alters the tariff rates, as the graphs will need to be re-plotted and the updated graph image re-encoded to represent the change to the data.

Regarding the design of the graphs, all are bar charts and have the same horizontal axis (x-axis) divided into one hour segments of a 24 hour period. Colouring to the background of the graph illustrates the *High-Demand* period in a red tint, whereas the *Off-Peak* period has a green tint. Each bar in the graph represents a different host which is identified and labeled with it's corresponding colour in the legend, located in the top right.

5.3.1 Refresh Button

The cost and carbon graphs have been adjusted to accommodate their respective high and low tariff rates. These rates can be altered from the Tags page, however, for the change to appear in the charts they must first be refreshed. As mentioned above, the graphs must be re-plotted to represent this change.

5.3.2 Energy Graph

The energy graph is the basis from which all other graphs are calculated from. It represents the total number of kilowatt-hours (KWh) consumed in 24 hour segments of the configured DC. Each bar in the graph represents a different host, Figure 5.5 shows the three hosts and their IDs; 284, 287 and 288.

The code snippet below demonstrates how the raw JSON data is store using a Pandas dataframe table. An outer for loop that iterates over each host in the configured DC, makes a call to the Papillon API to get the host's power usage. The API response is converted into JSON format which obliges a more streamline assimilation with other Python objects. Calculating the total power consumed is performed using another for loop which affixes the power over a single hour into a list named *energy*.

This snippet describes how the *energy* list is totaled and stored in a Python dictionary, with the key value assigned the relevant hour segment. This dictionary is than stored in a Pandas dataframe of 24 rows representing each hour segment and the a single column representing a host. The column is titled by its corresponding *'hostid'* and its respective totals stored in the rows beneath. The column is divided by the value *GRAPH_DIVIDE* which simply converts the power total from megawatt-hours to kilowatt-hours. The host dataframe is added to a list of all dataframes from where they will later be merged.

```
1    hourlyResult = {key: sum(values) for key, values in energy.items()}
2    df = pd.DataFrame(hourlyResult.items(), columns=['hour', str(host['hostid'])])
3    df[str(host['hostid'])] = df[str(host['hostid'])]/GRAPH_DIVIDE
4    df_list.append(df)
```

Listing 5.1: Storing data from the Papillon API as Pandas dataframe divided into 24 hour segments extracted from the service/analysisService.py file

The resulting graph is depicted in Figure 5.5 on the following page 32. Since the Dummy Data Centre was running continuously over a three day period for the example DC, the energy consumed per hour remained relatively consistent. Including *Host 284* which was running a more strenuous process compared to the other hosts, the total KWh ranged from 0.052 to 0.023.

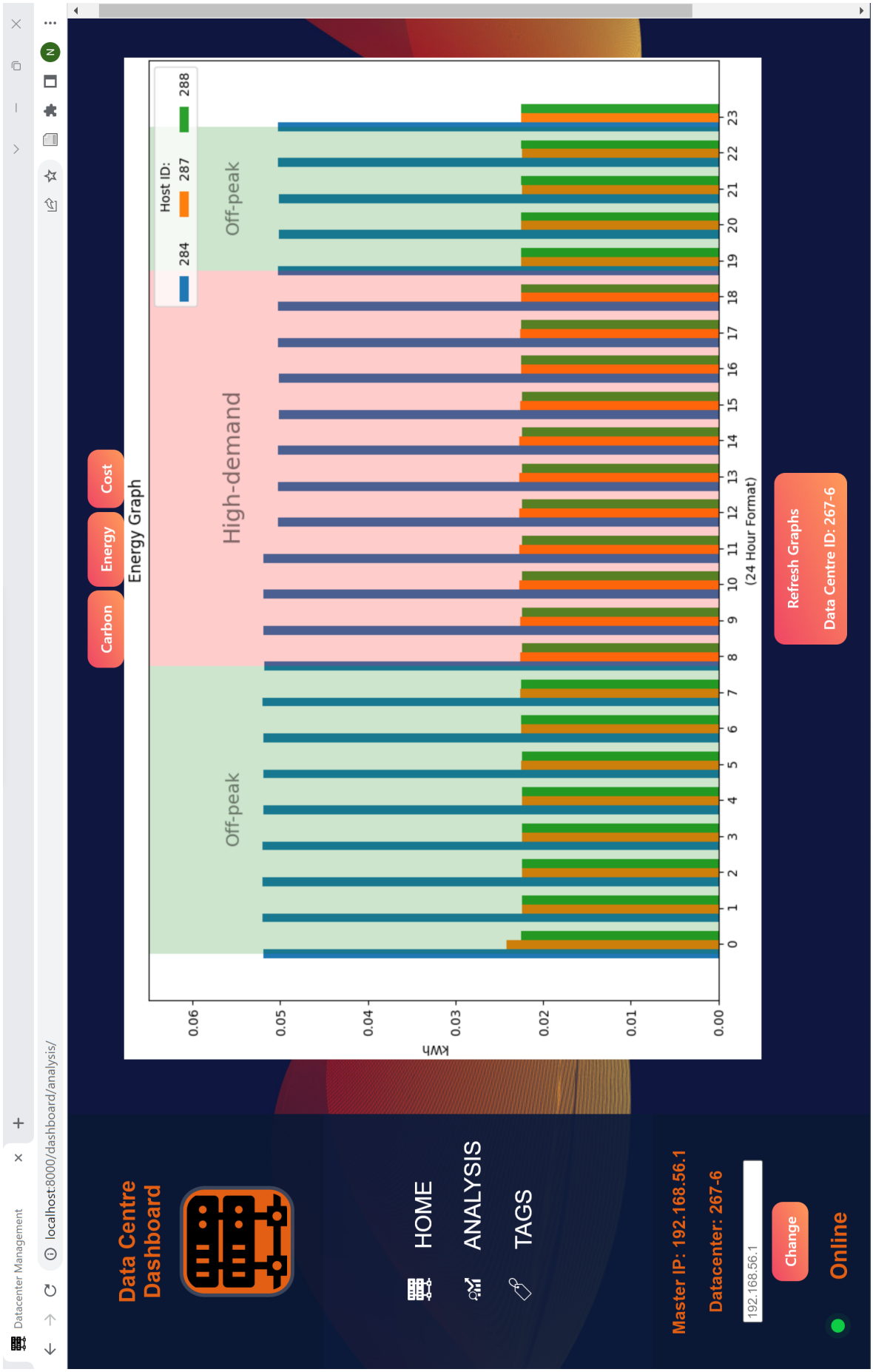

Figure 5.5: Energy Consumption Graph from Analysis Page

5.3.3 Cost Graph

The cost graph is derived from the energy graph by applying calculations that convert KWh to €. It represents the total number of Euros (€) consumed in 24 hour segments. As before, each bar in the graph represents a different host, Figure 5.6 on the following page 34 shows the three hosts IDs; 284, 287 and 288.

The code snippet below demonstrates how the energy dataframe is converted to represent it's cost equivalent. A table containing all the hosts and their respective hourly segment totals is passed in to the function below. This function returns a table which will be plotted as the cost graph. The conversion is calculated by determining whether the the hour segment is the during the high-demand or the off-peak tariff, with the relevant tariff being applied. The PUE assigned to the DC by the administrator during configuration is also applied to adjust for overall efficiency.

```
def cost_estimate(table):
    try:
        temp = table.copy()
    except: return

    pue = services.get_pue()
    high = services.get_high_cost_tariff()
    low = services.get_low_cost_tariff()

    for col in temp.columns:
        for row in temp.index:
            if row < HIGH_TARIFF_START or row > HIGH_TARIFF_END:
                mult = low
            else:
                mult = high
            temp.at[row, col] = temp.at[row, col] * mult * pue

    return temp
```

Listing 5.2: Function that converts the Dataframe of energy consumed to it's cost counterpart before it's plotted as a graph from the services/analysisService.py file

The resulting graph is depicted in Figure 5.6 on the following page 34. The variation in the high tariff rate and low tariff rate is very apparent. For *Host 284*, roughly €0.0003 is saved per hour during the off-peak hours. The other two host also see a saving of €0.0002. Even though these figures are negligible and may even seem insignificant, the impact in the context of a DC that operates with more than 3 hosts longer than a 3 day period could be imperative.

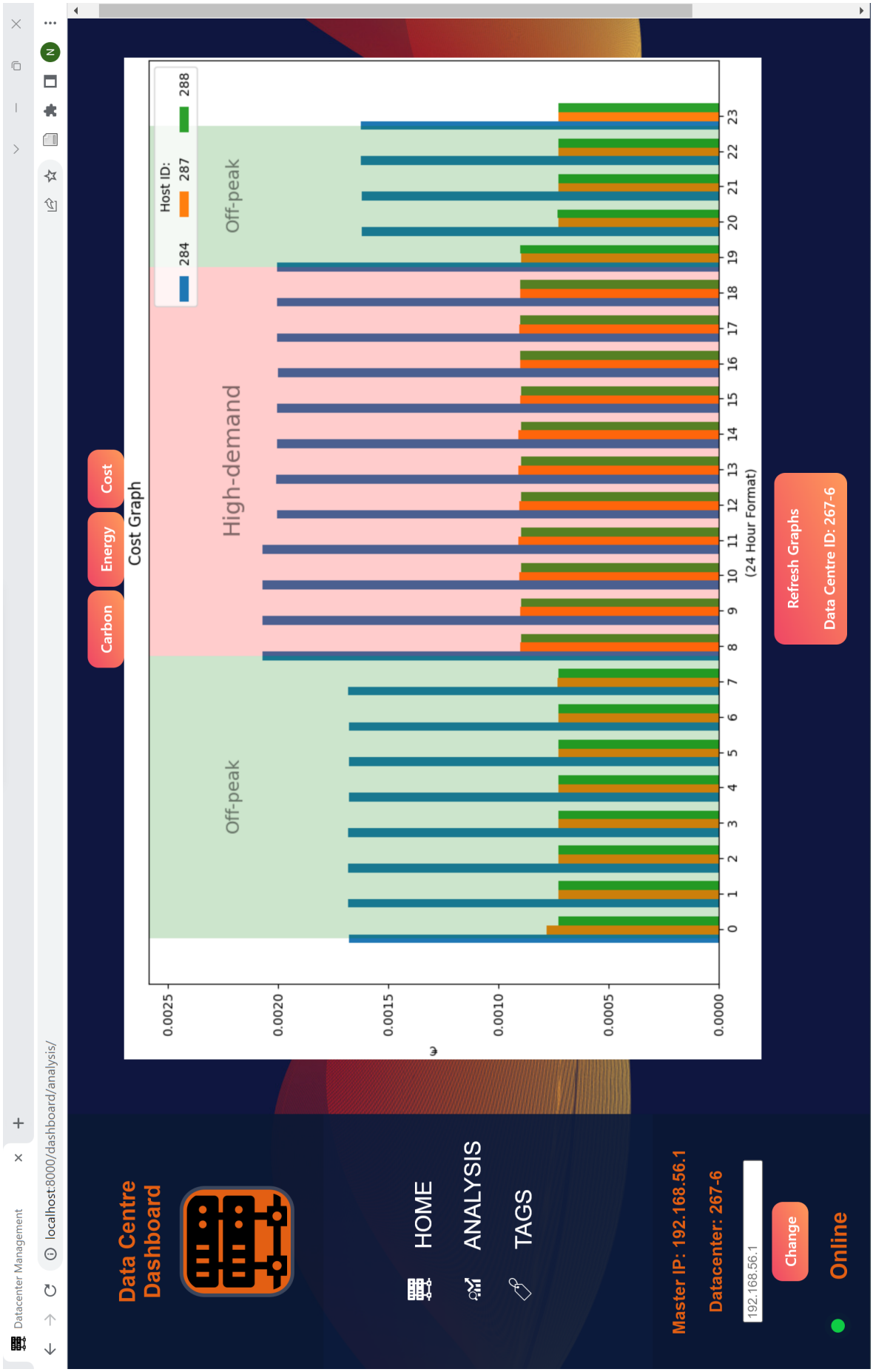

Figure 5.6: Cost Graph from Analysis Page

5.3.4 Carbon Graph

The carbon graph, like the cost graph, is also derived from the energy graph by applying calculations that convert KWh to kgC02. It represents the amount of kilograms of carbon dioxide (kgC02) consumed in 24 hour segments. Again, each bar in the graph represents a different host and it's respective carbon footprint, Figure 5.7 on the following page 36 shows the three hosts IDs; 284, 287 and 288.

The code snippet below demonstrates how the energy dataframe is converted to represent it's carbon equivalent, almost identical to the cost conversion. Apart from the different tariffs to the cost graph, PUE isn't applied to the carbon graph. Once again, a table containing all the hosts and their respective hourly segment totals is passed in to the function below. The conversion is calculated by determining whether the the hour segment is the during the high-demand or the off-peak tariff, with the relevant carbon tariff being applied.

```
1  def cost_estimate(table):
2      try:
3          temp = table.copy()
4      except: return
5
6      pue = services.get_pue()
7      high = services.get_high_cost_tariff()
8      low = services.get_low_cost_tariff()
9
10     for col in temp.columns:
11         for row in temp.index:
12             if row < HIGH_TARIFF_START or row > HIGH_TARIFF_END:
13                 mult = low
14             else:
15                 mult = high
16             temp.at[row, col] = temp.at[row, col] * mult * pue
17
18     return temp
```

Listing 5.3: Function that converts the Dataframe of energy consumed to it's carbon emissions counterpart before it's plotted as a graph from the services/analysisService.py file

The resulting graph is depicted in Figure 5.7 on the following page 36. The variation in the high tariff rate and low tariff rate is very apparent. For all host approximately half the carbon is emitted per hour during the off-peak hours. As mentioned in Section 4.2.6, additional renewable energy generated during this period could have caused the variation from the high carbon and low carbon tariff. Carbon intensity is very volatile, especially in Ireland, allowing for high and low rates provides administrators with a more in-depth understanding of the carbon consumption originating from their DC.

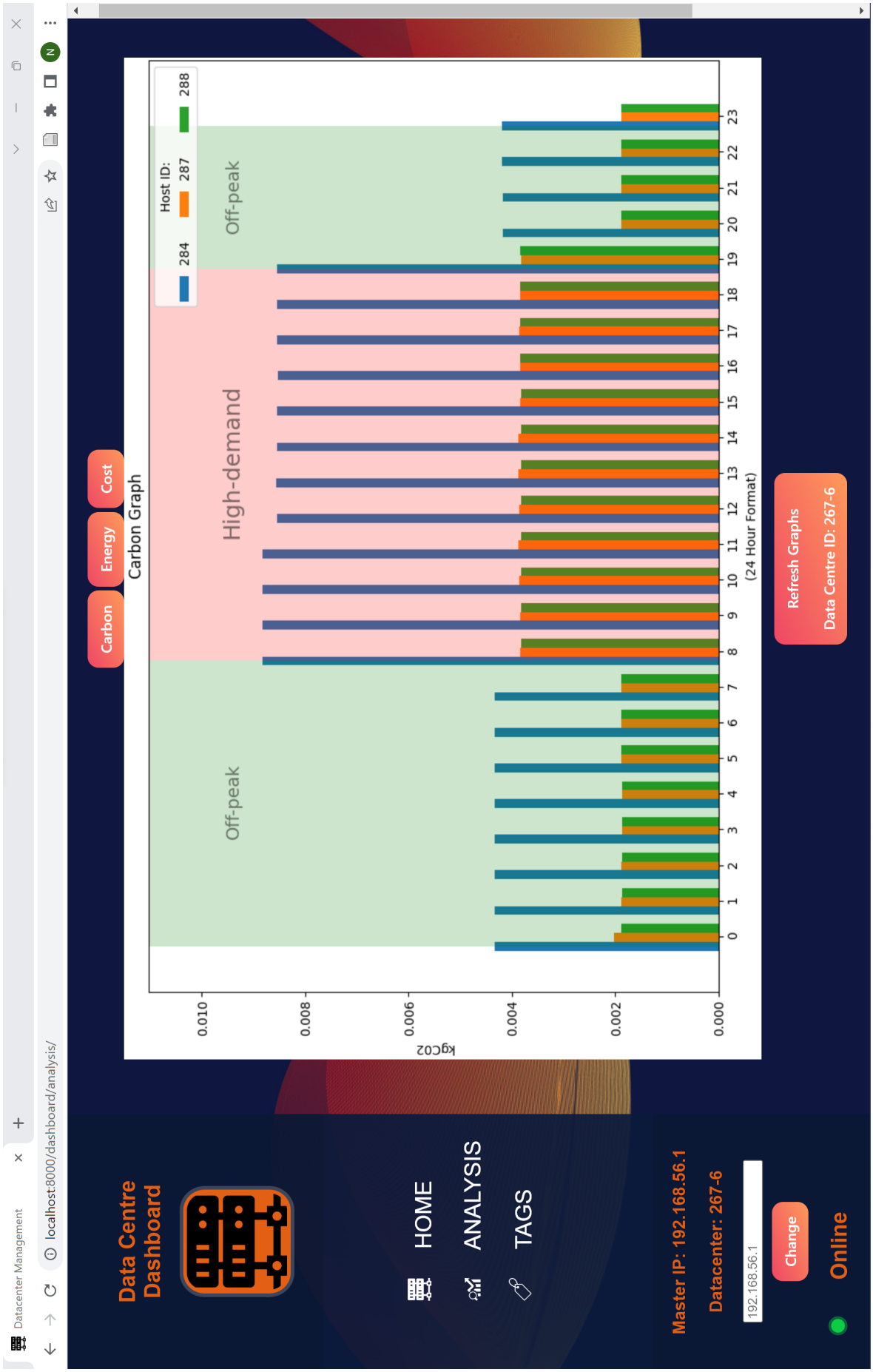

Figure 5.7: Carbon Consumption Graph from Analysis Page

5.4 Tags Page

The Tags page illustrates all the hosts of the configured DC in the first table. The administrator can choose to tag hosts based on their flexibility, depicted on page 39. The table also express useful metrics of the host such as the average cost conversion, average carbon conversion (12 hours at the high rate as well as 12 hours at the low rate), and the CPU utilisation. The carbon conversion rate and the cost conversion rate are both pres-set to a high of 1.2 and a low of 1.0 as default values. However, the screenshot of the Tag page below has edited tariff rates as follows:

- High Carbon Tariff = 0.309 kg/C02
- Low Carbon Tariff = 0.277 kg/C02
- High Cost Tariff = € 0.090
- Low Cost Tariff = € 0.160

Utilising these tariff rates, the best and worst case scenarios are calculated and portrayed in the second table. The best case scenario represents the optimal hours to run the host, in other words, the total KWh consumed by the host is converted using only the low tariff. The worst case scenario is the most unfavorable hours, calculating using only the high tariff. The potential saving calculation is the difference between the corresponding best and worst case calculations.

5.4.1 Adjusting Tariffs

The tariff rates grant the user the ability to experiment with varied high and low values for both the cost and carbon conversion of the configured DC. These values can be adjusted be entering desired rates in the input fields located in the top right corner of the page. The input fields have

Figure 5.8: Input error Tags Page

error checking to revoke unrealistic or impossible values. The data sanitation class which is in form file, is where the input data is sent to be purified. If the data doesn't meet the sanitation requirements it is rejected as is evident from Figure 5.8. Examples of such requirements are that the high tariff must be a larger value to the low tariff, and the number of decimal places is limited to 3 for the carbon tariff and 2 for the cost tariff.

5.4.2 Flexible Hosts

Hosts that are assigned the *flexible* tag are listed in the bottom table of the Tags page. From here the administrator can easily identify both the potential cost and carbon emission savings. Apart

from the adjustable rates, the best and worst case scenarios are quite rigid. They only account for the pre-determined off-peak and high-demand hours which can't be altered by the user. This is an area which I would hope to improve in future work, by offering more controller to administrator allowing them to change the peak hours and vice versa.

5.4.3 Changing Flexibility

Each host has a Boolean field that represents whether or not it is flexible. This can simple be modified b the user by clicking the **Change** button located on the right side of the first table. It's flexibility is clearly labelled as *Flexible* with a green background cell, or *Not Flexible* with a red background cell. The code snippet below demonstrates how this functionality works. The arguments which identify a specific host are passed into the method. Instead of using a pair of if statements to find other whether the flexible Boolean is true or false, it is inverted to represent whatever state it wasn't in last. The new value is updated and returned to the specified host's table in the database.

```
1 def update_hosts_flexibility(master, sub_id, floor, rack, host):
2
3     host = Host.objects.filter(masterip=master).filter(sub_id = sub_id).filter(
4         floorid=floor).filter(rackid=rack).filter(hostid=host)
5     opposite = host.values().get()['flexible']
6     if not host.exists(): return Host.DoesNotExist
7     try:
8         host.update(flexible=not opposite)
9     except: return
```

Listing 5.4: Function which inverts the current value of a host's flexibility from the services/tagServices.py

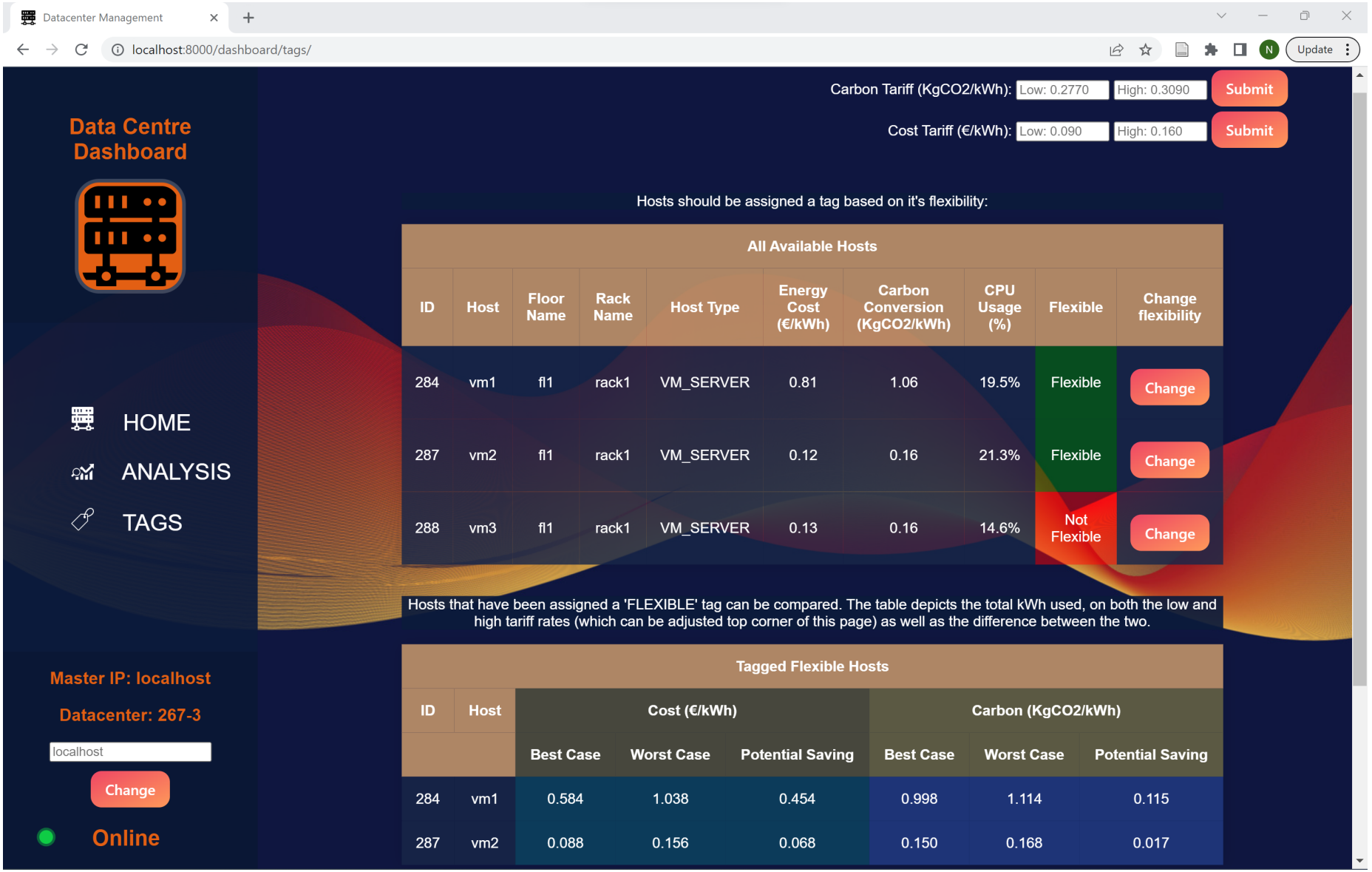

Figure 5.9: Tags Page

Chapter 6: Improvements & Future Work

This chapter presents a comprehensive exploration of potential avenues for future research and development within the project's defined scope. The suggestions for future work encompass a wide range, spanning from minor enhancements to significant and ambitious proposals. The chapter will conclude by assessing the achievements of the current project and evaluating the extent to which the initial objectives have been accomplished.

6.1 Incomplete Advanced Objectives

The ambitious advanced objectives designated at the onset of this project were unfortunately not achieved. These objectives were:

- Replace the manual approval of tag assignment with automated optimisation solution techniques such as simulation annealing
- Allow for the analysis of more efficient servers and the impact on the DC

The development timeline and project schedule did not have allow sufficient time to implement and test the advanced objectives. These objectives would have require additional research, complex algorithm implementation, and thorough testing to ensure their effectiveness and reliability. Given the limited time available, it was not feasible for me to complete these objectives within the project's condensed time frame.

6.2 Peak Hours as Adjustable Variables

The dashboard has a daily time frame which is divided into two 12 hour periods of High-Demand (08:00 - 18:00) and Off-Peak (19:00 - 07:00) is set and cannot be altered. This is definitely a field that could be reevaluated to introduce some customisation from the administrator. Ideally, the high-demand and off-peak times would have been adjustable in this dashboard and was something I initially planned to achieve. However, adapting the peak period as a variable posed significant technical challenges that were difficult to overcome within the project's scope. Implementation implied even more intricate calculations, data modeling, and integration with the existing dashboard. Addressing these challenges required more time, resources, and expertise than I initially anticipated.

6.3 Testing On An Existing DC

Testing this project on a larger-sized DC provides several advantages over testing on a dummy DC. A larger DC represents a more realistic and complex environment compared to the dummy DC. It reflects the challenges and intricacies of managing a substantial DC with a diverse range of servers, infrastructure, and workloads. Testing in such an environment allows for a more accurate assessment of the dashboard's performance and effectiveness.

A larger DC would also enable the evaluation of the dashboard's scalability. It would allow for testing the dashboard's ability to handle a larger volume of data, workloads, and resources. This assessment is crucial to ensure that the dashboard can scale appropriately and continue to deliver accurate results and recommendations as the DC grows. It will also be exposed greater variability in server configurations, energy consumption patterns, and operational characteristics. Testing the dashboard on an existing DC should uncover potential challenges and variations that may arise in the future.

6.4 Conclusion

The project has successfully achieved its objective of an informative dashboard aimed at reducing the carbon footprint and operational costs of a small DC. The analysis conducted on the Dummy DC has demonstrated the application's effectiveness in identifying inefficient servers, analysing the temporal consumption in hourly segments, and provide insights to potentially altering hosts' flexibility. Users can configure multiple instances of a DC, allowing for overlapping periods.

The development process exposed me to new technologies like Django and Papillon, which compelled me to establish alien method to adhere to industry-standard design practices. This approach has resulted in a stable and reliable application that can be deployed in operational DCs.

The application provides administrators with valuable statistics on carbon dioxide emissions. It estimates the carbon footprint of each hour over a certain period, adjusting the high and low rates. Real-time graphs display these carbon emissions estimates, offering a comprehensive overview of the entire DC. These features empower DC administrators to effectively reduce the carbon footprint of their facilities.

Similarly, the application provides metrics related to server costs. It calculates the best and worst case scenario costs, comparing and contrasting adjustable high and low financial conversion rates. These metrics offer insights into the least and most expensive hosts, as well as identifying areas of potential saving. Real-time operational cost graphs illustrate the estimated costs of varying tariffs.

However, due to time constraints, the advanced objectives could not be fully realized. The implementation of an automated optimisation solution for tagging processes, which would have provided more accurate and informative carbon footprint and emissions data, was not feasible. The other advanced objective of introducing a metric that would compare the current servers with a more efficient counterparts wasn't achieved either. Additionally, the application was tested only using the limited number of hosts in the Dummy DC, preventing analysis on a larger DC. Nonetheless, the results obtained from the DC analysis remain valid and informative. It is important to note that a larger DC would have yielded more comprehensive insights into the system.

Acknowledgments

I would like to express my sincere appreciation to Professor Damian Dalton for his invaluable guidance and support throughout this project. His expertise and insightful feedback have greatly contributed to the quality of this work. Additionally, his perspective has vastly broadened my understanding of the subject matter.

I extend my gratitude to Abhay Vadher, a postdoc who works in UCD alongside Damian, for his assistance in understanding and implementing the Papillon software. His expertise was instrumental in setting up the Papillon Dummy Data Center as well as providing advice on its application for this project.

Finally, I would like to thank my family and friends for their unwavering support and encouragement throughout my academic journey. Their love and understanding have been a constant source of inspiration.

I am indebted to all the individuals and organizations mentioned above for their contributions, without which this project would not have been possible.

Project Workplan

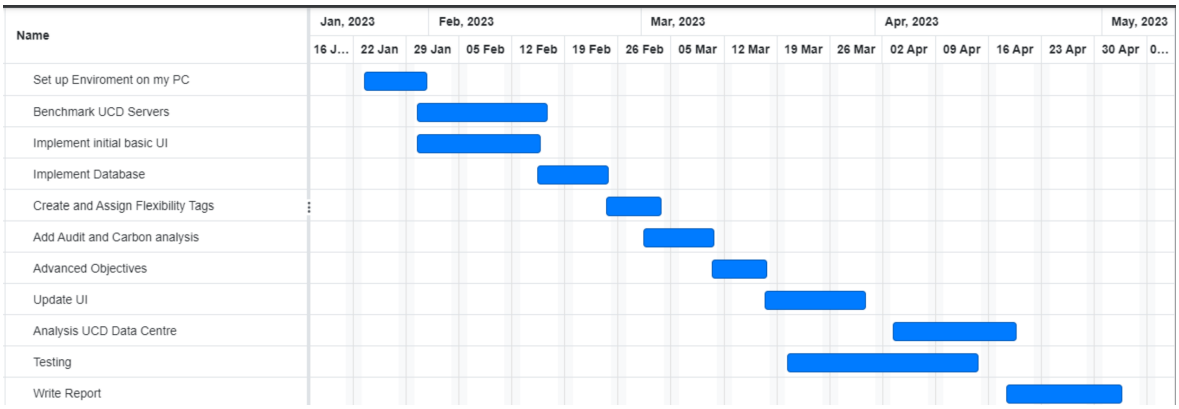

Figure 6.1: Gantt Work Flow Plan

This project plan is roughly divided into two main sections. A development section and a research/testing section. It includes a period of time March allocated to attempt the advanced goals I outlined in the Chapter 1. Ideally, the majority of the work load will be completed by March, which will allow the completion of a case study on the DC in the School of Computer Science in UCD in April.

Setting Up Environment.

- Setting up a dummy DC
- Getting familiar with Papillon

Implement UI.

- Setting up initial User Interface with Django
- Setting up models and forms with SQLite database

Core Objectives.

- Code audit and flexibility functions

Advanced Objectives.

- Code advanced objectives, time permitting

Finalize UI.

- Incorporate my code with the initial UI

Testing and Research,

Bibliography

1. Koomey, J. & Taylor, J. Zombie/comatose servers redux. *Report by Koomey Analytics and Anthesis* (2017).
2. Pawlish, M., Varde, A. S. & Robila, S. A. *Cloud computing for environment-friendly data centers* in *Proceedings of the fourth international workshop on Cloud data management* (2012), 43–48.
3. *Oxford Dictionaries - Data Centre* https://www.oxfordlearnersdictionaries.com/definition/english/data-centre..
4. Lee, G. *Cloud networking: Understanding cloud-based data center networks* (Morgan Kaufmann, 2014).
5. *What is a data center?* 2022. https://www.cisco.com/c/en/us/solutions/data-center-virtualization/what-is-a-data-center.html.
6. Dunlap, S. *Data Center Tiers: What are they and why do they matter?* July 2021. https://www.impactmybiz.com/blog/data-center-tiers-explained/.
7. Rasmussen, N. Determining total cost of ownership for data center and network room infrastructure. *Relatório técnico, Schneider Electric, Paris* **8** (2011).
8. Malone, C. & Belady, C. Metrics to characterize data center & IT equipment energy use, in 2006 Digital Power Forum. *Richardson, TX, USA2006* (2006).
9. Brady, G. A., Kapur, N., Summers, J. L. & Thompson, H. M. A case study and critical assessment in calculating power usage effectiveness for a data centre. *Energy Conversion and Management* **76,** 155–161 (2013).
10. *What is a Hyperscale Data Center?* https://www.vertiv.com/en-emea/about/news-and-insights/articles/educational-articles/what-is-a-hyperscale-data-center/.
11. Uddin, M., Shah, A., Alsaqour, R. & Memon, J. Measuring efficiency of tier level data centers to implement green energy efficient data centers. *Middle-East Journal of Scientific Research* **15,** 200–207 (2013).
12. *Cisco 2022 global hybrid cloud trends report* Sept. 2022. https://www.cisco.com/c/en/us/solutions/hybrid-cloud/2022-trends-report-cte.html?ccid=cc002960&oid=rptdnc029203#~summary.
13. Satyanarayanan, M. The Emergence of Edge Computing. *Computer* **50,** 30–39 (2017).
14. Koomey, J., Taylor, J., *et al.* New data supports finding that 30 percent of servers are 'Comatose', indicating that nearly a third of capital in enterprise data centers is wasted. *TSO logic* (2015).
15. Person, I. B. Top 10 countries with the most Data Centres. *Data Centre Magazine*. https://datacentremagazine.com/top10/top-10-countries-most-data-centres (Sept. 2021).
16. Sverdlik, Y. *Barclays 'fires' 9,000 idle servers from data centers* May 2014. https://www.datacenterknowledge.com/archives/2014/05/12/barclays-fires-9000-idle-servers-data-centers.
17. Daniels, J. Server virtualization architecture and implementation. *XRDS: Crossroads, The ACM Magazine for Students* **16,** 8–12 (2009).

18. Pretorius, M., Ghassemian, M. & Ierotheou, C. *An investigation into energy efficiency of data centre virtualisation* in *2010 international conference on P2P, parallel, grid, cloud and internet computing* (2010), 157–163.
19. Nordhaus, W. D. Two Centuries of Productivity Growth in Computing. *The Journal of Economic History* **67,** 128–159 (2007).
20. Horowitz, M. *Computing's energy problem (and what we can do about it). In 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* in *IEEE, feb* (2014).
21. UN. *The Sustainable Development Goals Report* (July 2022). https://unstats.un.org/sdgs/report/2022/.
22. *The Paris Agreement* https://www.un.org/en/climatechange/paris-agreement.
23. Pierrehumbert, R. There is no Plan B for dealing with the climate crisis. *Bulletin of the Atomic Scientists* **75,** 215–221. eprint: https://doi.org/10.1080/00963402.2019.1654255. https://doi.org/10.1080/00963402.2019.1654255 (2019).
24. *Delivering the European Green Deal* Aug. 2022. https://ec.europa.eu/info/strategy/priorities-2019-2024/european-green-deal/delivering-european-green-deal_en.
25. EU. *2030 Climate Target Plan* https://climate.ec.europa.eu/eu-action/european-green-deal/2030-climate-target-plan_en.
26. Avgerinou, M., Bertoldi, P. & Castellazzi, L. Trends in Data Centre Energy Consumption under the European Code of Conduct for Data Centre Energy Efficiency. *Energies* **10,** 1470. ISSN: 1996-1073. http://dx.doi.org/10.3390/en10101470 (Sept. 2017).
27. Project, T. S. *"Lean ICT: Towards Digital Sobriety": Report* May 2020. https://theshiftproject.org/en/article/lean-ict-our-new-report/.
28. Acton, M. *The EU Code of Conduct on Data Centres* Mar. 2022. https://e3p.jrc.ec.europa.eu/publications/2022-best-practice-guidelines-eu-code-conduct-data-centre-energy-efficiency.
29. EU. *Energy Efficiency in Data Centres* https://joint-research-centre.ec.europa.eu/energy-efficiency/energy-efficiency-products/code-conduct-ict/code-conduct-energy-efficiency-data-centres_en.
30. Houlihan, D. *Datacentre Budget Tool,* 2021. https://csgitlab.ucd.ie/danielhoulihan/fyp_datacenter_management.
31. *Django Software Foundation* https://www.djangoproject.com/foundation/.
32. Dalton D. Vadher, A. *Papillion API Reference,* 2016.
33. *Beeyon,* 2023. https://www.beeyon.com/.
34. *IEA - Real-Time Electricity Tracker,* 2023. https://www.iea.org/data-and-statistics/data-tools/real-time-electricity-tracker?category=price&from=2023-3-1&to=2023-4-30&tracker=true&country=IRL&fuel=Renewables.