Universidad Politécnica de Madrid

Escuela Técnica Superior
de Ingenieros de Telecomunicación

Máster Universitario en Ingeniería
de Telecomunicación

Trabajo Fin de Máster

Design and Implementation of Energy-Conscious Optimizations through Deep Reinforcement Learning for Edge Computing Scenarios

Sergio Pérez Morillo

2020

Máster Universitario en Ingeniería de Telecomunicación

Trabajo Fin de Máster

Título: Design and Implementation of Energy-Conscious Optimizations through Deep Reinforcement Learning for Edge Computing Scenarios

Autor: D. Sergio Pérez Morillo

Tutor: Dña. Patricia Arroba García

Departamento: Departamento de Ingeniería Electrónica

Miembros del Tribunal

Presidente: D.

Vocal: D.

Secretario: D.

Suplente: D.

Los miembros del tribunal arriba nombrados acuerdan otorgar la calificación de: .....

Madrid, a de de 2020

Universidad Politécnica de Madrid

Escuela Técnica Superior
de Ingenieros de Telecomunicación

Máster Universitario en Ingeniería
de Telecomunicación

Trabajo Fin de Máster

Design and Implementation of Energy-Conscious Optimizations through Deep Reinforcement Learning for Edge Computing Scenarios

Sergio Pérez Morillo

2020

Resumen

La demanda actual de aplicaciones basadas en el internet de las cosas (IoT), la inteligencia artificial (IA), y el *Big Data* está creciendo más rápido que nunca. Estas aplicaciones tienen el potencial de transformar muchos ámbitos de la sociedad. Sin embargo, tienen unas necesidades específicas que requieren de una revisión del contexto tecnológico actual. Hasta ahora, el paradigma de computación reinante ha sido *Cloud Computing*. En este, las instalaciones se concentran en áreas remotas. Nuevos servicios con limitaciones críticas de latencia y ancho de banda como la conducción autónoma sufrirán en una red cada vez más saturada.

Un nuevo paradigma surge para complementar el anterior, la computación de borde o *Edge Computing*, el cual acerca los recursos a los usuarios finales. Este consigue aliviar la saturación en centros *Cloud* desplegando *Edge Data Centers* (EDCs) en ciudades. De esta forma, se fomenta un modelo más sostenible, escalable y flexible. No obstante, plantea otros problemas. Los EDCs deberían ocupar el menor espacio posible ya que estarán en áreas densamente pobladas. Su despliegue consta de numerosos y más pequeños centros cuyo coste es vital para poder escalar la solución. Las ciudades tienen una alta demanda de electricidad, por lo que, la eficiencia energética es crítica. Por último, los EDCs deben ofrecer soluciones a medida a los usuarios finales y sus aplicaciones. Centrarse en aplicaciones específicas se traduciría en mejores servicios y más rentables.

Esta trabajo aborda estos desafíos optimizando escenarios de *Edge Computing* desde dos frentes. Por un lado, optimizamos los sistemas de refrigeración. Los gastos de estos sistemas en centros de datos representan el 40% de la factura total [1]. Usando refrigeración por inmersión en dos fases, este consumo podría reducirse a un 1% o 2% [2]. Esto tiene implicaciones espaciales también. Comparado con sistemas tradicionales, este tipo de refrigeración reduce diez veces el espacio necesario. Por otro lado, también optimizamos la asignación de recursos mediante modelos de IA para mayor eficiencia energética. Por poner un ejemplo, los centros de Google han reducido su consumo total de energía en un 15% usando estas teconologías [3].

Para ello, en este proyecto se han definido varios elementos que componen los escenarios de *Edge Computing* y se han desarrollado estrategias energéticas para optimizarlos. Estos elementos incluyen (i) una aplicación basada en un sistema de asistencia a la conducción; (ii) dos tipos de EDC, uno refrigerado por aire y otro por inmersión en dos fases, utilizando datos de prototipos reales; (iii) un gestor de asignación de recursos encargado de optimizar basado en *Deep Reinforcement Learning*, una tecnología que ayuda a resolver entornos complejos y dinámicos. El simulador de *Edge Computing* empleado para demostrar el potencial de nuestras estrategias es Mercury. Este ha sido validado ya por la comunidad científica [4].

Palabras Clave

Computación de Borde, Optimización energética, Refrigeración por Inmersión en Dos Fases, Aprendizaje por Refuerzo Profundo, Asignación de Recursos

Summary

Today's demand for applications involving emerging technologies such as the Internet of Things (IoT), Artificial Intelligence (AI), and Big Data is growing faster than ever before. These data-intensive applications hold the potential for a better future by transforming all aspects of society. However, they come at a cost. Up until now, the reigning computing paradigm has been Cloud Computing (CC). CC facilities concentrate in large and remote areas. Novel services with critical latency and bandwidth constraints such as autonomous driving, remote healthcare, or cloud gaming would suffer under an increasingly saturated network.

Inevitably, a new paradigm arose to complement the former one, Edge Computing (EC). EC brings computing facilities closer to end-users for the offloading of CC tasks in Edge Data Centers (EDCs) deployed in cities. Thus, helping to foster a more sustainable, scalable, and flexible model. Nevertheless, EC raises other concerns. EDCs should occupy as little space as possible since cities are densely populated. Unlike in CC, there are many rather small facilities. Their economic cost is vital to scale EC solutions. High-density urban areas have a higher electricity demand than large and isolated regions. Hence, energy efficiency becomes a critical aspect. Finally, EDCs need to offer tailor-made solutions to end-users and their applications. Focusing on specific applications in an area would translate into better and more cost-effective services.

This MSc thesis seeks to address these challenges by optimizing EC scenarios in two ways. On the one hand, the optimization of cooling systems. Cooling energy expenses in data centers represent, on average, 40% of the total bill [1]. By using two-phase immersion cooling, the cooling energy consumption could drop to a mere 1% or 2% [2]. It does not only have energy implications but space ones too. Compared to traditional systems, two-phase immersion cooling reduces the physical footprint by a factor of ten. On the other hand, the optimization of resource allocation by employing AI data-driven models. For instance, by applying these algorithms, Google's data centers have reduced its total energy consumption by 15% [3].

To this end, this project has devised the elements that comprise EC scenarios and developed energy-aware strategies to optimize them. These elements include (i) a data-intensive application based on Advanced Driver Assistance Systems (ADAS) and suited for EC; (ii) two types of EDC, one air-cooled and another immersion-cooled, using data traces of either an air-based or a two-phase immersion cooling prototype; (iii) a resource allocation manager in charge of optimizing the scenarios using Deep Reinforcement Learning (DRL), an AI-based technology for solving complex and dynamic environments. The simulation tool employed is Mercury, a novel 5G EC simulator already validated in scientific journals [4].

Keywords

Edge Computing, Energy-Aware Optimization,Two-Phase Immersion Cooling, Deep Reinforcement Learning, Resource Allocation

This project has been partially supported by the Centre for the Development of Industrial Technology (CDTI) and State R&D Program Oriented to the Challenges of the Society (*Retos Colaboración 2017*) under contracts IDI-20171194 and RTC-2017-6090-3 from the Spanish Ministry of Science and Innovation.

Contents

Summary **i**

List of Acronyms **ix**

1 Introduction and Objectives **1**

1.1 Smart Cities, Growth and Concerns 1

1.2 Cloud Computing, the Current Paradigm 3

1.3 Edge Computing, a Novel Approach 4

1.4 Energy Optimization in Edge Data Centers 5

1.5 MSc Thesis Motivation and Objectives 8

2 Literature Review **11**

2.1 Cooling Systems in Edge Data Centers 11

2.1.1 Cooling System Overview 11

2.1.2 Two-Phase Immersion Cooling Systems 14

2.2 Resource Allocation in Edge Data Centers 17

2.2.1 Current Optimization Strategies 17

2.2.2 Deep Reinforcement Learning 19

3 The Edge Computing Scenario **25**

3.1 Mercury, a 5G Edge Federation Simulator 25

3.2 Use Case: Advanced Driver Assistance Systems 27

4 Edge Data Center Modeling **31**

4.1 Air-Based Edge Data Center 31

4.1.1 IT Equipment 32

4.1.2 Cooling System 34
4.2 Immersion-Based Edge Data Center 35
4.2.1 IT Equipment 35
4.2.2 Cooling System 38

5 Energy-Conscious Optimization **41**
5.1 Resource Allocation Manager 41
5.1.1 Actor-Critic Theory 42
5.1.2 Actor-Critic Implementation 44
5.2 Edge Computing Scenario Configuration 53
5.2.1 General Considerations 53
5.2.2 Air-Based Scenario 56
5.2.3 Immersion-Based Scenario 57
5.2.4 Heterogeneous Scenario 59

6 Results **61**
6.1 Air-Based Scenario 62
6.2 Immersion-Based Scenario 66
6.3 Heterogeneous Scenario 70

7 Reflection on Research **77**
7.1 Conclusions 77
7.2 Future Work 81

Bibliography **94**

A Ethic, Economic, Social and Environmental Aspects **97**
A.1 Introduction 97
A.2 Description of Relevant Impacts Related to the Project 98
A.3 Detailed Analysis of Some of the Relevant Impacts 98
A.4 Conclusions 99

B Budget **101**

B.1 Budget of Material Execution 101

B.2 General Expenses and Industrial Benefits 102

B.3 Total Budget 102

C WILO IPL Pump Datasheet **103**

List of Acronyms

A2C Advantage Actor-Critic.

A3C Asynchronous Advantage Actor-Critic.

ADAS Advanced Driver Assistance Systems.

AI Artificial Intelligence.

ANN Artificial Neural Network.

AP Access Point.

AR Augmented Reality.

CACS Cold Aisle Containment System.

CC Cloud Computing.

CFC Chlorofluorocarbons.

CNN Convolutional Neural Network.

COP coefficient of performance.

CPU Central Processing Unit.

CRAC Computer Room Air Conditioning.

CRAH Computer Room Air Handler.

CV Computer Vision.

DEVS Discrete Event System Specification.

DL Deep Learning.

DNN Deep Neural Network.

DQN Deep Q-Network.

DRL Deep Reinforcement Learning.

EC Edge Computing.

EDA Exploratory Data Analysis.

EDC Edge Data Center.

ETSI European Telecommunications Standards Institute.

FaaS Function as a Service.

FC Fog Computing.

FNN FeedForward Neural Network.

GPU Graphics Processing Unit.

GWP Global Warming Potential.

HACS Hot Aisle Containment System.

ICT Information and Communications Technology.

IoT Internet of Things.

IT Information Technology.

LSTM Long Short-Term Memory.

M&S&0 Modeling, Simulation, and Optimization.

MDP Markov Decision Process.

MEC Mobile Edge computing.

ML Machine Learning.

MP Markov Process.

MSD Mean Square Deviation.

NRMSD Normalized Root Mean Square Deviation.

OCP Open Compute Project.

ODP Ozone Depletion Potential.

PID Proportional–Integral–Derivative.

PU Processing Unit.

PUE Power Usage Effectiveness.

QoE Quality of Experience.

QoS Quality of Service.

R^2 Coefficient of determination.

RAM Random-Access Memory.

RAN Radio Access Network.

RL Reinforcement Learning.

RMSD Root Mean Square Deviation.

RNN Recurrent Neural Network.

SDN Software-Defined Network.

SLA Service Level Agreement.

SOTA State of the Art.

TD Temporal Difference.

UE User Equipment.

VEC Vehicular Edge Computing.

VM Virtual Machine.

List of Figures

1.1 IoT investment growth in the past three years. 2
1.2 Expected economic impact by segment in IoT by 2025 ($ trillions). . 2
1.3 Top IoT concerns in 2018 and 2019. 3
1.4 Edge Computing and Cloud Computing stack. 4
1.5 ICT energy consumption forecast. 6

2.1 Two-phase cooling tank design. 15
2.2 Air-based and two-phase immersion cooling space comparison. . . . 16
2.3 Novec 7100's heat transfer capacity as a function of surface temperature. 17
2.4 Framework of Reinforcement Learning. 20
2.5 Simplified taxonomy of Reinforcement Learning. 21
2.6 Framework of Deep Reinforcement Learning. 22

3.1 Fog Computing model of Mercury. 26
3.2 Devised ADAS scenario. 27
3.3 Satellite view and UE location in the San Francisco Bay Area. . . . 28
3.4 APs' and EDCs' location. 28

4.1 Correlation heatmap of the air-based dataset. 32
4.2 Air-based GPU power model's predictions. 33
4.3 Air-based data center layout for the cooling model. 35
4.4 Two-phase immersion cooling prototype. 36
4.5 Two-phase GPU power model's loss curves and predictions. 37
4.6 Pump's power consumption as a function of flow rate. 39
4.7 Simplified heat transfer scenario. 39

5.1 Two-headed FeedForward Neural Network. 46
5.2 Return without (left) and with (right) prior reward (top) standardization. 48
5.3 Evaluation metrics with rolling averages during model training. . . . 52
5.4 Evaluation metrics for trained models. 53
5.5 Air-based GPU power model with maximum clock frequencies. . . . 56
5.6 PUE subject to IT inlet temperature. 57
5.7 Immersion-based GPU power model with maximum clock frequencies. 57
5.8 Real-sized two-phase immersion cooling tank. 58
5.9 Immersion power consumption subject to IT demand 59

6.1 Air-based best scenario energy performance comparison. 63
6.2 Air-based best scenario delay performance comparison. 64
6.3 Air-based best scenario energy performance comparison (hot standby). 65
6.4 Air-based best scenario delay performance comparison (hot standby). 66
6.5 Immersion-based best scenario performance comparison. 67
6.6 Immersion-based best scenario performance comparison (hot standby). 69
6.7 Heterogeneous I best scenario performance comparison. 71
6.8 Heterogeneous I best scenario performance comparison (hot standby). 72
6.9 Heterogeneous II best scenario performance comparison. 74
6.10 Heterogeneous II best scenario performance comparison (hot standby). 75

List of Tables

2.1 Cooling System Overview. 12
2.2 Cooling System Operating Temperatures. 12

4.1 RX580 air-based power model's hyperparameters. 34
4.2 RX580 air-based power model's evaluation metrics. 34
4.3 RX570 immersion-based power model's hyperparameters. 38
4.4 RX570 immersion-based power model's evaluation metrics. 38

5.1 Two-headed FeedForward Neural Network's hyperparameters. 47
5.2 Model's training hyperparameters. 50
5.3 Configuration of Mercury's version, elements, and simulation time. . 54
5.4 Configuration of GPU resource allocation. 55
5.5 Configuration of the ADAS session. 55
5.6 Configuration of Processing Units. 56

6.1 Air-based scenario energy results. 62
6.2 Air-based scenario delay results. 63
6.3 Air-based scenario energy results (hot standby). 64
6.4 Air-based scenario delay results (hot standby). 65
6.5 Immersion-based scenario energy results. 66
6.6 Immersion-based scenario energy results with (hot standby). 68
6.7 Heterogeneous I scenario energy results. 70
6.8 Heterogeneous I scenario energy results (hot standby). 71
6.9 Heterogeneous II scenario energy results. 73
6.10 Heterogeneous II scenario energy results (hot standby). 74

B.1 Budget of material execution. 101
B.2 General expenses and industrial benefits. 102
B.3 Total budget. 102

List of Algorithms

5.1 Resource allocation manager's A2C training 51

Chapter 1

Introduction and Objectives

1.1 Smart Cities, Growth and Concerns

Nowadays, technological breakthroughs such as the Internet of Things (IoT), Big Data, Artificial Intelligence (AI), and 5G usher the world into a new way of life. Innovative services and applications that originated from these advances facilitate the establishment of Smart Cities. This novel conception of modern cities has the potential to empower society in truly diverse sectors. R. Sánchez-Corcuera et al. [5] identify plenty of examples. There are thriving solutions in business-related domains (e.g., agriculture, logistics), citizen-related domains (e.g., education, healthcare), environment-related domains (e.g., waste management) and goverment-related domains (e.g., E-government).

From an economic standpoint, there has also been a growing interest in the past few years. According to ARM's 2020 IoT business Index [6], surveyed companies had substantial progress in IoT adoption from 2017 to the present. The companies that increased investment in this area have surged to a 82% from 62% in 2017. As shown in Figure 1.1, one in five companies has incremented their budget by 50%. On the other hand, AI and IoT combined solutions are gaining momentum. 26% of survey respondents integrate these two technologies in their plans altogether, while 64% of them concur that IoT data improves the value of their AI solutions.

Another report from Gartner [7] estimates that the IoT market will reach 5.8 billion devices this year, which entails a 21% increase from last year. The sector of automation will have the most considerable growth rate at 42%, followed by automotive and healthcare (31% and 29%, respectively). However, the larger sectors will be utilities driven by smart metering and physical security driven by intruder detection systems.

Not only that, but McKinsey [8] also says IoT investments are set to grow 13.6% per year through 2022. As for devices, they expect annual growth of 30% in wide-range devices and 20% in short-range devices, and both fueled by 5G capabilities. In a previous report from them [9], they conclude that IoT sector's potential value will lie between $4 and $11 trillions by 2025 (Figure 1.2).

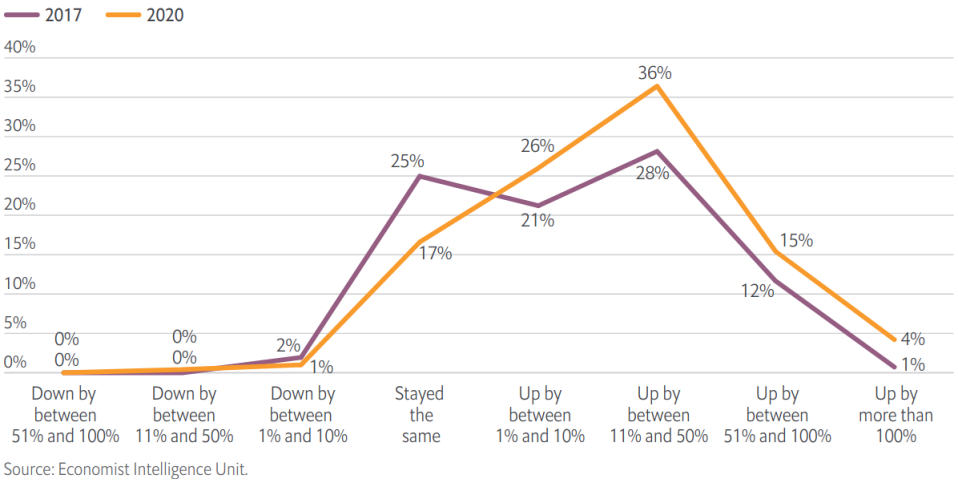


Figure 1.1: IoT investment growth in the past three years.

From this brief analysis, it is clear from both a social and economic point of view that there is a vast potential in the sector. However, some concerns surrounds these solutions too. In a report by the Business Insider Intelligence team in early 2020 [10], they surveyed to find significant concerns of a handful of top executives in the field (Figure 1.3). They found that security was at the top by far, but the highest growing issues were privacy (related to security) and performance. The latter is part of the motivation of this research and lead us to the next Section. In addition, the same report estimates that the number of IoT devices will grow from 10 billion in 2018 to 64 billion in 2025. With these numbers in mind, is the current computing paradigm ready to sustain and scale this ever-growing number of devices with an even more extensive range of applications? Or will the performance ultimately suffer under these circumstances?

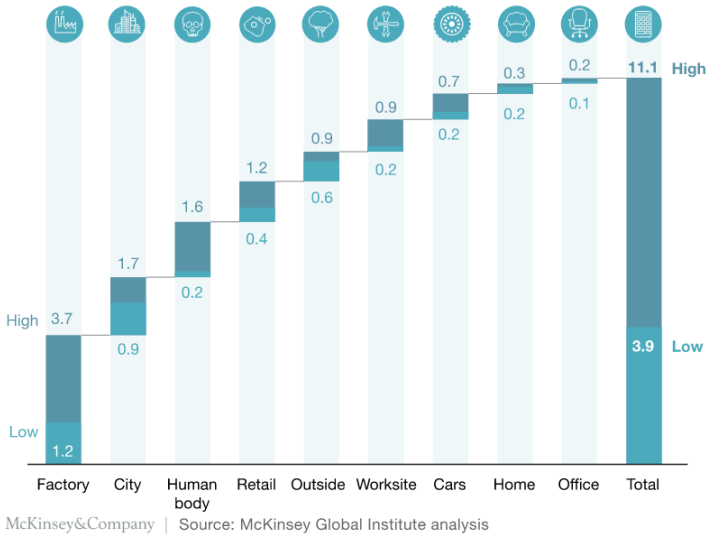


Figure 1.2: Expected economic impact by segment in IoT by 2025 ($ trillions).

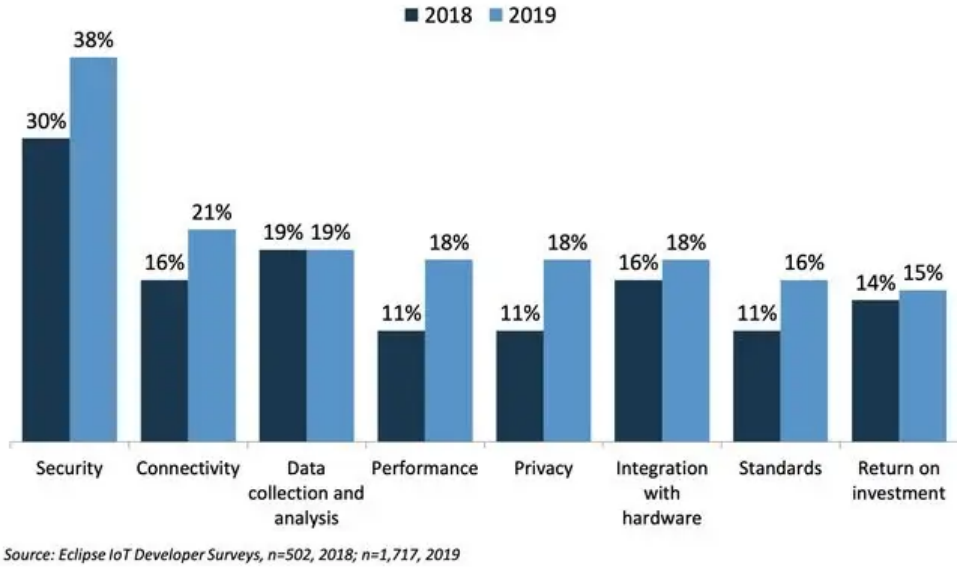


Figure 1.3: Top IoT concerns in 2018 and 2019.

1.2 Cloud Computing, the Current Paradigm

So far, the reigning computing paradigm has been Cloud Computing (CC). These past few years proved successful, and thus enabled the first wave of novel applications and services. Its core concept relies on a massive centralization of computing resources, storage, and control in large-sized data centers. Among other reasons, their size makes their deployment unfeasible in cities where most of the data provenance is located. Companies tend to build these infrastructures far away from them, prioritizing other factors like land cost or climatology.

While this limitation might not be a problem for some applications, those with critical latency and bandwidth constraints will undoubtedly experience performance issues. Take, for instance, 5G-Powered Remote Robotic Surgery. Last year the first surgery of this kind took place in China [11]. The main reason that made it possible was the low latency provided by a 5G network. Swedish telecommunication multinational Ericsson clearly describes what it takes to make this service go mainstream [12]:

- **Latency** is critical. It must be one millisecond or less for feasible haptic feedback (the technology behind it).
- **High throughput capacity** is necessary to transfer high-quality video streams, which impacts Quality of Experience (QoE).
- Connection downtime (**availability**) and packet loss (**reliability**) must be minimal to guarantee the maximum Quality of Service (QoS).
- **Security** is fundamental since breaches might be lethal costing numerous human lives for this kind of application.

In an increasingly saturated network, CC solutions will likely fail to meet the requirements previously listed. Another application with similar requisites, though in a completely different sector, is Advanced Driver Assistance Systems (ADAS).

ADAS applications lie between conventional and autonomous vehicles. Their data sources are high-resolution cameras placed around the vehicle. According to NVIDIA [13], cars that feature ten of these cameras produce, on average, two megapixels per second. Moreover, those using Deep Learning (DL) applications may reach 250 trillion of operations per second.

Similar examples are Cloud Gaming, Augmented Reality (AR), and AI-based virtual assistants. Therefore, it is not only a matter of latency, like in the remote surgery example. Some services also need enough computing resources to work. Thus, a novel approach is required to face these challenges.

1.3 Edge Computing, a Novel Approach

Edge Computing (EC) emerges as an alternative computing paradigm that rather complements CC. Unlike CC, EC relies on bringing computing resources closer to data sources [14]. Edge Data Centers (EDCs) are placed at the edge of the network, where data sources concentrate, offloading tasks from cloud data centers.

These smaller data centers have enough computing resources to process small-to-medium workloads. EDCs work as a gateway connecting cloud servers to edge devices in the event of more demanding tasks. Figure 1.4 depicts the stack. Gartner predicts that by 2022, 75% of enterprise-generated data will be created and processed outside of centralized cloud centers, a massive leap from just 10% in 2018 [15].

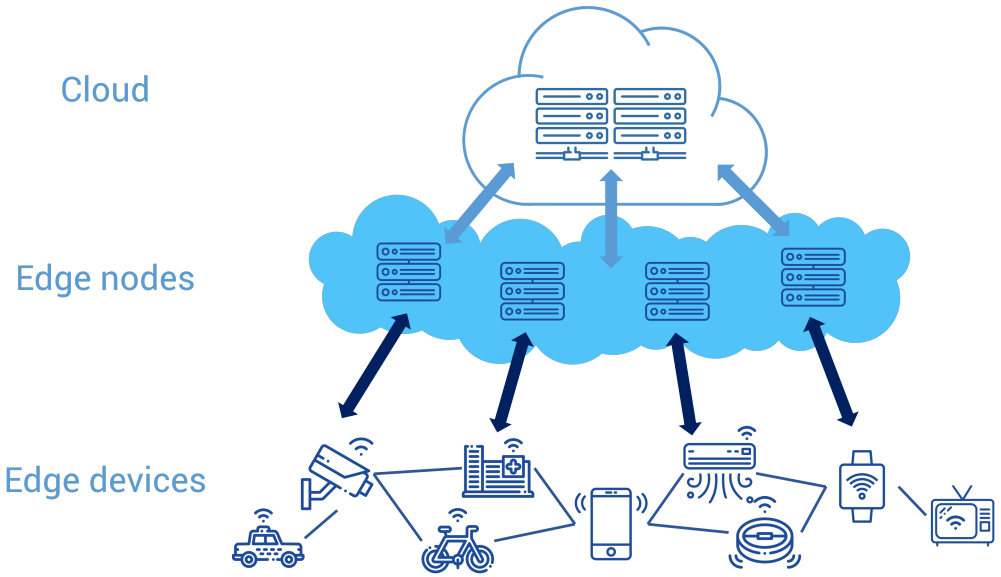


Figure 1.4: Edge Computing and Cloud Computing stack.

Source: Alibaba Cloud [16].

Some evidence is found in the Literature about the benefits of EC. S. Yi et al. [17] devise an application for face recognition in which they compared response time for CC and EC solutions. The latter reduced the response time from 900 to 169 ms, more than five times faster. K. Ha et al. [18] used a solution similar to EDCs, Cloudlets, to test wearable cognitive assistance's performance. Not only was the delay reduced between 20 and 200 ms, but energy consumption was also reduced at least by 30%. It is worth noting that EC is location-aware, unlike CC, hence it is more efficient handling mobility of devices, which is an essential characteristic in many Smart City applications.

R. Buyya et al. [19] identify four advantages that are related to the requirements mentioned in Section 1.2. These advantages are: reduction of network congestion as EC avoids centralization; latency plummets as EC is closer to the data origin; security improves as EDCs can further process information utilizing safer methods; and reliability improves as EC distributed architecture avoids potential central outages.

Although EC offers many opportunities, it is not without its challenges. Its requirements are: minimum space since EDCs are placed in urban areas with high land prices; minimum cost since an EDC federation might feature many facilities; efficient energy management since energy demand is more intense in urban areas (this will also avoid potential grid issues and save costs); and user-centered design since EDC traffic might be entirely dedicated to a single service from a specific type of source. An EC system with the right design will foster a more scalable, sustainable, and flexible model.

1.4 Energy Optimization in Edge Data Centers

Regardless of the computing paradigm, the ever-rising demand for Information and Communications Technology (ICT) applications have made its energy consumption soared in recent years. This trend is not expected to end in the foreseeable future. An article published in Nature [20] suggests that by 2030 the total electricity demand of ICT may reach 20.9% of the global market, being data centers the fastest growing sector (Figure 1.5).

Interestingly enough, ICT companies accounted for more than half of renewable energy purchases already in 2016. In a report by the European Commission [1], they state that ICTs energy consumption got to 7% in 2018 and expect this number to reach 13% by 2030. They also claim that 1.3% of electricity demand and 2% of global CO_2 emissions came exclusively from data center operations in 2018.

Furthermore, the price of electricity has increased considerably over the last decade [21]. Between 2007 and 2017, EU citizens have seen their bills increase by 23%, and in Spain alone, this figure has soared to 47%. If EC seeks to close the gap between sources and resources everywhere, the energy-efficient operation of its EDCs is essential.

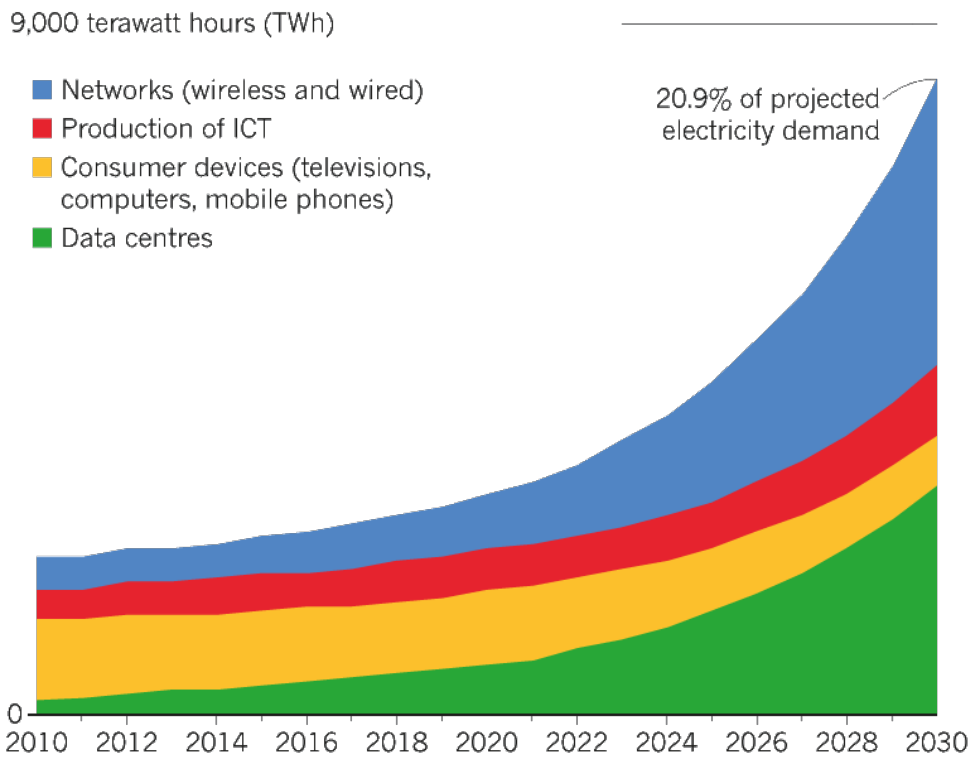


Figure 1.5: ICT energy consumption forecast.

Source: Nature [20].

The concept of EDCs is rather recent. So, the following analysis is based on conventional data centers since both draw many similarities. In data centers [1], there exist two main energy contributors, Information Technology (IT) demand and cooling equipment. On average, the latter accounts for 40% and oscillates between 24% and 61%. There is room for improvement in this area.

On the other hand, it is critical to allocate efficiently IT demand and monitor temperatures to avoid overheating or overcooling issues [22]. Power consumption is closely related to temperature. A decrement of just one degree Celsius in data center's rooms may translate to savings of 7.08% in energy consumption [23].

Thus, power optimization in data centers, and hence in EDC, is two-fold. As for EDC cooling strategies, there exists three predominant groups: air-based, water-based, and dielectric-liquid-based. Here is a short description of each of them (Chapter 2 cover this topic in more depth):

- **Air-based cooling systems**: The most widespread systems today. Their biggest advantage is the harnessing of favorable weather conditions. Though, in EC, this is not possible in all scenarios. Compared to the others, these solutions consume more energy and need more space (vital for EC). Current strategies: hot and cold aisle (with or without containment), and free cooling.

- **Water-based cooling systems**: Water systems solve some issues of the former such as corrosion and vibration. It also has a higher heat transfer coefficient, although lower than dielectric-liquid systems. Potential leakages are fatal in these solutions. Current strategies: CRAH units, in-row cooling, active and passive rear door cooling, and direct water cooling.

- **Dielectric-liquid-based cooling systems**: The highest heat transfer coefficient by far. It requires less space than air solutions and is safer than water solutions. However, most of them are still in development. Their potential is unmatched, specially immersion-based architectures. Current strategies: Indirect cooling, sealed equipment, one-phase, and two-phase immersion cooling.

Regarding the second type of optimization in EDCs, resource allocation strategies, there is plenty of examples in the Literature (more on them in Chapter 2). Most approaches include three steps: a monitoring system to gather data from different sources such as temperature, frequency, or humidity; the implementation of a model that utilizes these data inputs; and dynamic and optimal allocation based on the outputs of the model. Modeling may go two ways, theory-driven, and data-driven. The former has been predominant until the last decade or so, with the surge of AI and DL.

In data centers, perhaps the most common theory-driven models make use of control theory for dynamic systems [24]. Popular choices often include Proportional–Integral–Derivative (PID) controllers [25]. However, in large-scale data centers, this approach might not be the most suitable since State of the Art (SOTA) data-driven solutions capture better the ever-changing and extremely volatile nature of these large infrastructures [26]. As reported by Google [3], the use of AI data-driven solutions in their data centers has yielded cooling and total energy savings of 40% and 15%, respectively. This reasoning could also be applied to EDCs, as their conditions are similar, although one concentrates its processing power in a single place, and the other shares it among many.

This research tries to leverage the latest advances in AI data-driven modeling. Their sudden success is no coincidence. DL models have already achieved outstanding results [27] in genuinely different areas such as computer vision, search engines, health, robotics, autonomous driving, or industrial control. Artificial Neural Networks (ANNs) are the basic DL algorithm. Unlike other Machine Learning (ML) algorithms, ANNs scale particularly well with large sets of data.

They are not a novel idea. ANNs were designed back in the late 1940s. The exponential progress in hardware systems (e.g., Graphics Processing Units (GPUs)) this past decade has helped to unlock their true potential. ML and DL solutions have three ways of training algorithms: supervised, unsupervised, and Reinforcement Learning (RL). The most suitable approach for dynamic systems like resource allocation in EDCs is RL, which tries to optimize a cumulative reward over time by taking different actions in an environment. On the other hand, supervised learning is also useful for modeling EDC behavior in simulated scenarios.

1.5 MSc Thesis Motivation and Objectives

To the best of our knowledge, there is no a single project in the Literature that combines both SOTA cooling system (two-phase immersion cooling) and resource allocation (AI and DL modeling) optimizations in the data center sector, let alone in EC. Some projects focus on the former [28] and others on the latter [26], but none of them take advantage of both. Moreover, most EC works center around the deployment of small EC nodes instead of EDCs with enough computing resources for more demanding tasks like ADAS. Our research fills this gap.

The final goal of our research is the design and implementation of energy-efficient deployments of EDC federations in EC scenarios. It is achieved through the strategies mentioned above, two-phase immersion cooling systems and the most-up-to-date data-driven modeling of IT demand allocation. So far, during my tenure in this project, I have been involved in two publications:

- In the first publication [29], we created our first two-phase immersion cooling prototype with a cluster of Raspberry Pi 3B [30] running data analytics workloads and using the dielectric liquid Novec 7100 [31] as coolant. We modeled the temperature as a time series forecasting problem utilizing Recurrent Neural Networks (RNNs).
- In the second publication [32], we devised an air-based system. This time, we used a potential EC hardware unit, a Sapphire Pulse Radeon RX 580 GPU [33]. We utilized an ADAS application as workload and real traffic demand data from New York City to emulate a real EC scenario. Instead of temperature, we modeled power consumption. The model and workload of this publication are used again in Section 4 along with an immersion-based GPU system.

In this MSc thesis, we want to go a step beyond and join the modeling of EDCs equipment from our previous work with energy-conscious optimizations of dynamic EC scenarios through RL and a simulation tool for EC applications, Mercury. This simulator has already been tested and validated by the research community [4, 34]. The goal of the MSc thesis is the following:

- The simulation of the most realistic EC scenario possible using a real use case like ADAS. This will be achieved by leveraging the potential of Mercury.
- The modeling of actual parts of an EDC such as the processing units (i.e., GPUs) or the power consumption of the cooling system. We will use DL models for doing so.
- The study of energy optimizations in EC scenarios using RL algorithms, in particular its DL variant, Deep Reinforcement Learning (DRL).

It is worth noting that we are also working on a real-sized prototype with commercial partners. Many of the efforts here and previous works are linked to the

successful design and implementation of this prototype. This research and Mercury's have been partially supported by the Centre for the Development of Industrial Technology (IDI-20171194) and State R&D Program Oriented to the Challenges of the Society (*Retos Colaboración 2017*) (RTC-2017-6090-3). The remainder of the MSc thesis is divided as follows:

- In Chapter 2, an analysis of SOTA solutions is performed for both EDC cooling systems and resource allocation strategies. Moreover, a brief introduction to DRL is also provided.
- In Chapter 3, a brief explanation of the EC scenario is given, covering both the simulator, Mercury, and the case of use, a data-intensive ADAS application.
- In Chapter 4, the modeling of EDC components for this use case is carried out. There are power consumption models of an air-based and an immersion-based EDC. The models encompass the two primary energy sources in data centers, IT equipment and cooling. They both are based on real hardware.
- In Chapter 5, the design of energy optimization strategies for this case of use is devised. From DRL algorithms, agents, and reward functions to the specific choices for each type of EDC implemented.
- In Chapter 6, a comprehensive analysis of the results obtained using the designed energy optimizations and EC scenarios.
- In Chapter 7, a brief reflection on the MSc thesis and future steps that lay ahead.

Chapter 2

Literature Review

This Chapter covers SOTA solutions found in the Literature in both cooling systems and resource allocation in data centers that are relevant for EC scenarios. It puts a particular emphasis on technologies utilized in the following Chapters.

Section 2.1 presents an overview of different kinds of cooling systems used in data centers currently. It also explains in more detail the newest methods based on two-phase immersion cooling, from the concept, advantages, market solutions to its much-needed partnership with predictive modeling. Section 2.2 introduces current strategies for optimizing resource allocation. Moreover, it provides an introduction to RL and DRL with examples of use cases in EC applications.

2.1 Cooling Systems in Edge Data Centers

This Section gives a comprehensive review of different cooling systems in data centers on the market with a specific focus on two-phase immersion cooling, the tested method in this MSc thesis. Tables 2.1 and 2.2 summarized the content of the Section. Note that for the latter, higher operating temperatures mean higher potential energy savings.

2.1.1 Cooling System Overview

Historically, most data centers have built air-based cooling systems to refrigerate their rooms. EDCs are no exception. Huawei [35] and Rittal [36] already offer EDCs with air-based cooling methods. Hot Aisle / Cold Aisle [37] strategies are the foundation of this type of solution. It consists in lining up the data center's racks to face cold air intakes and hot air exhausts alternatively, and hence, forming cold and hot rows. CRAC units cool down hot air from hot aisles which comes from ceiling vents. Then, they reintroduce cold air into cold aisles using ventilation ducts placed under the room's floor. Its major drawback is the creation of unwanted hotspots and airflows. Even the most meticulous layout suffers from these issues.

Table 2.1: Cooling System Overview.

Cooling System	Advantages	Disadvantages
Air-based Cooling	Most widespread systems that can harness climatology.	High energy and space costs.
Hot Aisle / Cold Aisle	Aisle distribution of separated hot and cold airflows.	Prone to undesired hotspots and turbulence.
Hot Aisle Containment	Less cold air volume with similar efficiency.	High energy consumption. In case of overheating, lack of cold air to evacuate the system.
Cold Aisle Containment	Large cold air volume to avoid overheating. Lower energy costs than HACS.	Higher deployment costs than HACS.
Free Cooling	Outdoor heat exchange. Massive energy savings.	Needs favorable climatology and another cooling system for safety.
Water-based Cooling	Solves air problems like corrosion and vibration. Higher heat transfer coefficient.	Lower heat transfer coefficient than dielectric-liquid-based. Leakages are lethal.
CRAH Units	More efficient CRAC-like water-based system.	High energy consumption.
In-row Cooling	60% energy reduction compared to air solutions.	Needs traditional data center designs, thus limiting efficiency.
Passive Rear Door	Remove heat from the source using water.	Relies on server's fans.
Active Rear Door	Less space than in-row. Dynamic cooling.	Not cost efficient for most companies.
Direct water cooling	Ultra-low temperature water not needed.	High economic costs. Needs specific IT equipment.
Dielectric-liquid-based Cooling	Much more safe and energy efficient than water systems.	Still under development. Not widely used.
Indirect Cooling	Like in water-based, but no lethal leakages.	Does not leverage the heat transfer potential of the liquid.
Enclosed Chassis	More maneuverable system in direct contact with liquids.	Needs specific IT equipment.
One-phase Mineral Oil Immersion	Harness real heat transfer potential in open baths. Low pressures.	Needs low oil temperatures using an external circuit. Not easy to manage. Not clean.
One-phase Engineered Fluid Immersion	More clean, maneuverable, and better heat transfer than oil.	Needs low coolant temperatures using an external circuit.
Two-phase Engineered Fluid Immersion	Passive method that saves up to 95% of energy compared to air solutions.	Potential high pressures in closed baths. Still in development.

Table 2.2: Cooling System Operating Temperatures.

Cooling System	Temperature Range
Air-based Cooling (CRAC)	18°C - 27°C
Water-based Cooling	07°C - 40°C
One-phase Immersion Cooling	15°C - 38°C
Two-phase Immersion Cooling	15°C - 65°C

Cold Aisle Containment System (CACS) and HACS [38] alleviate them. CACS insulates cold aisles to avoid mixing the two air sources. This containment results in less needed cold air volume and higher stable temperatures, therefore energy and costs savings. Though, slower respose to overheating issues. HACS insulates hot aisles instead. It is the preferred option over CACS as it saves up 43% cooling energy costs [38]. However, the infrastructure for its deployment has higher prices.

The last air-based option, Free Cooling, offers the best energy efficiency. Free Cooling leverages favorable climatology of cold locations to cool down the data center's equipment. Cooling costs drop drastically since much of the cooling infrastructure is gone. Big ICT companies like Facebook tend to locate their largest cloud centers in these areas, achieving mind-boggling Power Usage Effectiveness (PUE) factors [39]. PUE is the ratio between the total and IT energy costs in a data center. However, this is not viable for EDCs as location-dependent solutions limit the reach of EC. Moreover, reports are suggesting that Free Cooling solutions are prone to failures, hurting efficiency [40].

Air-based system's drawbacks outweigh the advantages [41], more so in EC as Free Cooling is not feasible. Air flows lead to corrosion, vibration, and turbulence. Hence, potential failures. Air-cooling infrastructure is not space-efficient, critical in EC. Furthermore, the air heat transfer coefficient is the lowest among coolants in data centers. On the other hand, water-based cooling systems solve these problems and have better transfer characteristics, thus offering higher energy savings. It might save around 50% of energy costs compared to an air-based system [42]. Although it has a fatal flaw, leakages. Water leakages in a data center may be lethal, even more in EDCs, since they are placed in densely populated areas.

CRAH units [43] are the first approach to water-based cooling. They serve the same purpose as CRAC. However, CRAH's mechanism differs from its air-based counterpart. While the latter uses mechanical refrigeration, the former utilizes a cooling coil that is filled with water. In-row cooling [44] units have a high power density per rack that may save up to 60% compared to air solution [42]. Nevertheless, they need a conventional row configuration that leads to inefficient use of space.

Passive rear door cooling exchangers [45] remove heat from the source but rely on IT equipment's fans to push the airflow. They can remove up to 30 kW per rack in optimal situations, but it might be better to aim for lower levels to be cost-effective [46]. Active rear door cooling exchangers functions as the passive ones but have fans, valves, and programable controllers to avoid relying on server's fans. However, this technology seems only viable for companies that possess dense racks of at least 10 kW [47]. Lastly, direct water cooling [48] offers the maximum power density per rack among water solutions, amounting to 80 kW in some commercial products. It also works at higher operating temperatures, thus saving energy costs. The problem lies in the need of specific hardware, which limits its repercussion.

Dielectric-liquid-based cooling is on the rise. Its insane potential has put it into the spotlight. The most efficient solutions (two-phase immersion cooling) have more than six times the power density per rack of air solutions and a whopping

95% of energy reduction in cooling expenses [2]. Dielectric-liquid solutions vary widely. Indirect cooling resembles water solutions as it evades direct contact with the equipment. Unlike those strategies, leakages are not such a thread anymore. However, indirect cooling does not take advantage of the heat transfer potential that dielectric liquids have. Enclosed chassis cooling [49] aims to seal equipment in liquid-tight casings. This time, the hardware is in direct contact with the liquid. It is either pumped or circulated through rack's servers. Its primary disadvantage is that not all devices fit this design,. Hence cases usually cannot be repurposed.

The last dielectric-liquid cooling subgroup and the most energy-efficient is immersion-based. There are two main methods. First, one-phase immersion cooling [50] submerges the equipment in either an open or semi-open bath. The coolant remains in liquid-phase. This method leverages the heat transfer potential that the liquid has but needs an extensive recirculation system to cool it down, therefore worsening energy and space efficiency. Furthermore, the coolant needs to work at low temperatures to avoid evaporation, hence extra energy needed for its correct functioning.

This solution uses two types of coolants, mineral oils and engineered fluids [51]. The first ones are more widely used due to their market availability. These chemicals are petroleum-based distillates (mainly composed of alkanes and cycloalkanes) that are usually further processed with a hydrotreatment. These compounds present many disadvantages. Even with the special treatment that adds resistance, oils are still flammable. Their viscosity hurts the heat transfer capabilities and an efficient recirculation through the cooling system. Mineral oils have high Global Warming Potential (GWP) (i.e., significant toxicity) and are not entirely biodegradable, just around 30%-40%. They are also not clean, thus hindering maintenance operations by data center operators. On the other hand, engineered fluids are specifically designed for tasks such as cooling. Even though they tend to be more expensive, they are usually non-flammable, cleaner, safer to operate, almost entirely biodegradable (+90%). Moreover, they also have low toxicity, a low GWP, and higher heat transfer coefficients.

In the next Section, two-phase immersion cooling, the most energy-efficient cooling method today and the one used in this MSc thesis, is described in more detail than the rest along with the selected coolant, the engineered fluid Novec 7100 [31].

2.1.2 Two-Phase Immersion Cooling Systems

Two-phase immersion cooling is the most promising method of all. Its advantages are endless. As stated before [2], it can reduce cooling energy consumption by 95%. While air solutions have a power density between 4 and 40 kW per rack, two-phase immersion solutions can get to 250 kW per rack. It yields PUE values of 1.02-1.03, close to the global minimum, and only achieved before with free cooling solutions of large ICT enterprises. The advantage over the air method is that it is geographically agnostic, vital for EC solutions. This strategy is also space-efficient as it has ten times less physical footprint than air solutions (100 kW/m^2 compared to 10 kW/m^2). This space reduction is also huge for the

deployment of EDCs. Moreover, it does not need specific a casing or equipment. Thus it can adapt better than other water or dielectric liquid solutions.

The coolant's recirculating method is the key to both outstanding energy and space efficiencies [52]. It aims to passively (i.e., with no energy consumption) flow the engineered fluid by boiling and condensing it. During operation, the electronic equipment immersed in a close bath evaporates the coolant. Then, the coolant's vapor rises to the top where a heat exchanger condenses it back to liquid-phase.

The heat exchanger placed inside the tank circulates water and transfers it to a dry cooler where is cooled down. This part is the only source of energy consumption in this method, and it is close to zero since operating temperatures of both water and coolant are the highest among cooling solutions. Figure 2.1 depicts the tank in more detail.

Some critical considerations [53] for tank's design are: welded metal with powder coating as primary material; perforations for power, data, and heat exchager's pipes above the liquid line; hollow o-rings or inflatable gaskets for the lid; good insulation to minimize heat losses; and a system with a mechanical and pressure switch, a bellow and a valve for venting and pressure control. This last consideration is the primary disadvantage of this method. High pressures due to gas expansion in power demanding periods cause deterioration of the tank's insulation system. This problem leads to potential gas leakages and significant coolant losses. Nevertheless, a proper depressurization and monitoring system makes these issues unlikely to happen.

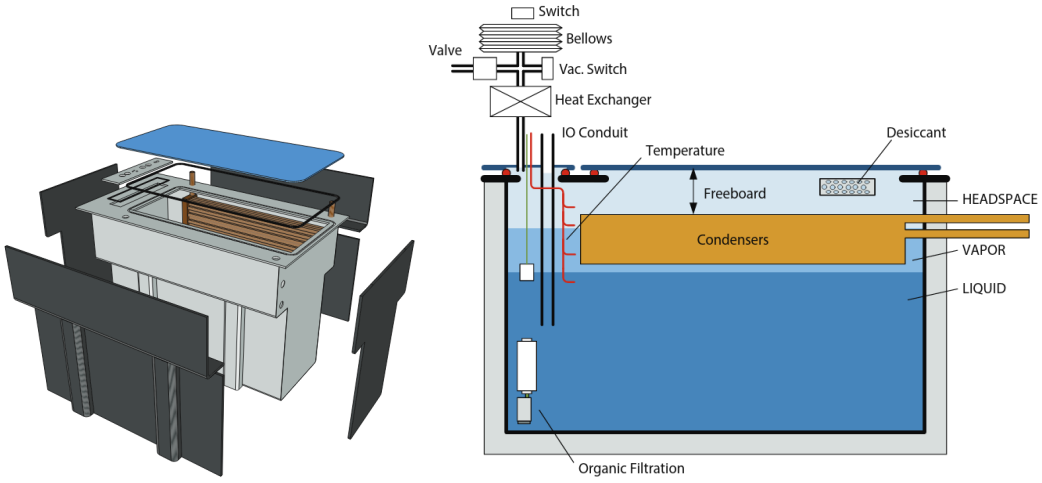


Figure 2.1: Two-phase cooling tank design.

Source: 3M [53].

Two-phase immersion cooling is still a work in progress. Just a handful of companies have venture to develop a market-ready product. Bitfury and its wholly-owned subsidiary Allied Control [54] developed in 2014 the award-winning DataTank, a container unit with six two-phase immersion cooled racks of 200-240 kW each for up to 1.4 MW in total. It has a cost of less than one dollar

per watt and is designed for 19-inch rails. They claim to achieve a PUE of 1.01, exceeding expectations. Figure 2.2 depicts how much space is saved using DataTank, comparing it to a conventional hot aisle/cold aisle configuration. While this product is admirable, it seems that it has only been used for mining cryptocurrency since its release. In any case, this data-intensive application is the perfect match for the novel hyper-efficient cooling system.

Another solution, though still a prototype, is the Open Compute Project (OCP)-compliant two-phase immersion cooling tank by Wiwynn [55]. The tank has a thermal design power of 100 kW, although it was tested using 60 OCP hardware nodes of 1 kW each. It achieves a PUE of 1.02. The tank was first showcased in the OCP 2019 Summit with no further news to this day. The most glaring flaw for both solutions is the underused and hidden heat transfer potential of the engineered fluid. To find out why, the coolant must be explained first.

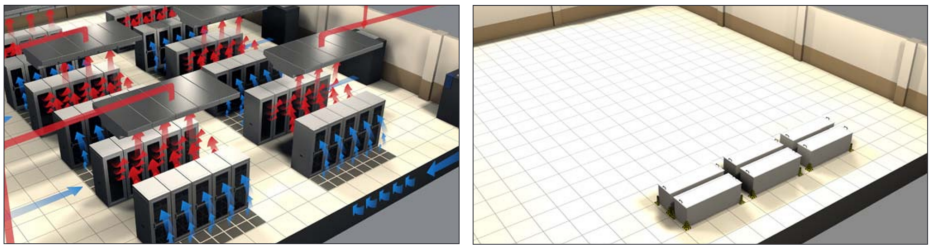


Figure 2.2: Air-based and two-phase immersion cooling space comparison.

Source: Allied Control [54].

Both products and this research use Novec engineered fluids by 3M. These compounds are used in many applications such as precise cleaning of electronic components, fire extinction, and of course, coolant for immersion cooling. As stated in the Subsection above, these liquids are non-flammable and clean. They also have low GWP, low toxicity, and zero Ozone Depletion Potential (ODP).

In our case, the chosen coolant is 3M's Novec 7100 [31]. It is a hydrofluoroether with a boiling point of 61°C, albeit a boiling point range between 50 °C and 99 °C. This dynamic feature is stressed in Figure 2.3. It shows the heat transfer capacity as a function of the surface temperature. There are six distinct zones, and most solutions work in the first two and the beginning of the third due to ignorant or conservative views. This fact is the primary reason why these solutions are underachieving. However, the true potential of the coolant is way up high at the end of the third zone.

Trying to work at maximum efficiency is challenging as surpassing that range (fourth zone) would cause a sharp drop in performance. In addition, the last three zones also have larger gas volumes making the coolant unstable. For this reason, predictive and proactive resource allocation is vital in EC scenarios that utilize two-phase immersion cooling. The efficient use of the power grid in cities is crucial. This issue leads us to the next Subection, in which resource allocation strategies are discussed. To the best of our knowledge, no other project has ever attempted to exploit this property before.

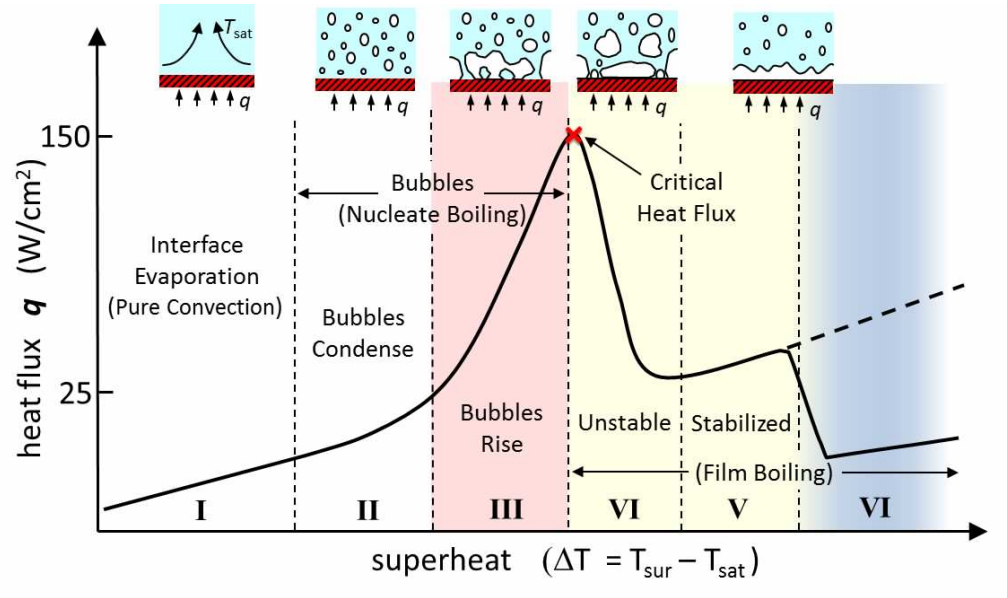


Figure 2.3: Novec 7100's heat transfer capacity as a function of surface temperature.

Source: 3M (directly to us).

2.2 Resource Allocation in Edge Data Centers

This Section analyzes modern resource allocation strategies in data centers. The principal focus was energy-aware EC solutions that included EDCs with GPU hardware, as this research aims for, but these projects scarce. Instead, this review widen that perspective by looking at other solutions that partially cover our goal. Then, a short introduction is given on the fundamentals of the selected optimization algorithm, DRL, describing its two comprising parts, DL and RL, along with SOTA solutions that employ it in data centers.

2.2.1 Current Optimization Strategies

Energy-aware optimization is a recurrent research topic, more so in a sector as data centers. Hence, many different approaches to solve it are found in the Literature. P. Arroba [23] proposes a taxonomy at different abstraction levels and scopes within a data center. The author classifies them as (i) data center scope — resource manager & scheduling, application, and middleware, (ii) server scope — run-time system, compiler, and architecture, and (iii) hardware scope — circuit, logic, and technology. This MSc thesis lies within the first scope and abstraction level (i.e., resource manager & scheduling). However, the taxonomy is for CC. Our research seeks the optimization of an EDC federation instead of a single large CC center.

These are not the only scopes drawn in this taxonomy. Perhaps, the most important division in energy-conscious optimization is between power-aware and

thermal-aware solutions. As for the first option, B. Li et al. [56] propose EnaCloud, a resource manager that aims to balance workloads in data centers at 90%, yielding up to 13% in energy savings. L. Liu et al. [57] devise GreenCloud, another resource manager with online monitoring, which uses a heuristic method to allocate Virtual Machines (VMs). The authors achieve savings of 27% in their best solution. A. Beloglazov et al. [58] alleviate performance declines due to overloading by migrating VMs again with heuristics. They show an energy reduction of up to 77% compared to static approaches for just a 5.4% of Service Level Agreement (SLA) violations.

As for thermal-aware solutions, A. Beloglazov et al. [59] present a resource manager that attempts to optimize host temperature, network, and utilization and to minimize VM migrations. They claim to achieve 83% of energy savings with this technique. Q. Tang et al. [60] utilize peak inlet temperature for balancing tasks. Their model helps to decrease cooling energy consumption by 30% by increasing the inlet temperature by 3°C. T. Mukherjee et al. [61] aim to reduce thermal cross-interference using genetic algorithms. This approach yields 40% in savings compared to their baseline. L. Wang et al. [62] develop an algorithm for thermal-aware scheduling. It accomplish a 12% reduction in cooling energy consumption, roughly 6% of the total electricity bill.

Apart from this dichotomy, another is based on the primary energy sources in data centers, IT, and cooling. Some works [63, 64, 65] in the Literature envision holistic energy-aware optimizations that encompass both IT and cooling power consumption. However, most projects usually choose one or the other. Furthermore, none of them utilize two-phase immersion cooling. Our research harnesses both energy sources with the most advanced cooling systems to maximize potential savings and to study their relation better.

So far, the cited works focus on conventional or CC data centers since there is more research about them than in EC as they have been around more time. However, there are also some recent projects related to EC. Most of them are centered around Mobile Edge computing (MEC), a term coined by the European Telecommunications Standards Institute (ETSI) for describing EC applications that employ wireless technologies such as 4G and 5G [66]. Y. Mao et al. [67] propose a green MEC system that utilizes energy harvesting devices. The authors use a Lyapunov optimization-based dynamic computation offloading algorithm for achieving the online offloading of tasks.

F. Wange er al. [68] optimize the energy consumption of Access Points (APs) using latency constraints of each user's computing task. K. Zhang et al. [69] devise energy-aware optimization combining computing and file transmission power costs for 5G heterogeneous networks. Y. Wang et al. [70] propose an algorithm that minimizes both the energy and latency of mobile devices by using dynamic voltage scaling. D. Puthal et al. [71] add an authentication scheme to their optimizations for balancing workloads and improving the security in EDCs.

The crucial difference that separates the cited projects on EC and, this research apart is the IT equipment of the EDCs (those works usually name them cloudlets or MEC servers). Unlike this MSc thesis, none of them use GPU-based

architectures as computing resources. GPUs has proven far superior capabilities to process demanding task such as Deep Neural Networks (DNNs) [72], videogames, or virtual reality applications compared to Central Processing Units (CPUs).

As explained in Chapter 1, applications like ADAS need enough resources to compute DL-based applications, hence GPU-based EDCs are necessary for EC scenarios. It seems that there is no project with these processing units in EC. However, there are plenty of examples of modeling GPU devices. J. L. Greathouse et al. [73] and R. Ge et al. [74] focus on the study of energy consumption variations of frequency changes. On the other hand, J. Lim et al. [75] and R. Ge et al. [76] study the effect of GPU's performance counters instead.

2.2.2 Deep Reinforcement Learning

The interest in DRL has exploded over the past few years in the AI community. The main driver has been several breakthroughs such as beating the world champion of Go [77] (a game in which super-human performance was deemed impossible), beating professional players of highly-complex videogames like DOTA or Starcraft [78], or solving the Rubik's cube with a robotic arm [79]. DRL is a vast topic that requires entire books to cover. This Subsection aims to introduce fundamental and useful concepts for the development of this work. There are other excellent resources to learn more about the subject [80, 81, 82].

The foundation of DRL is RL. Figure 2.4 depicts RL's framework. Here, an **agent** interacts with an **environment** repeatedly in discrete timesteps. Each timestep takes an **action** based on the **state** of the environment to maximize the cumulative **reward**. The words in bold are the base of any RL algorithm. Formally speaking, this can be described as a Markov Decision Process (MDP). As in any other Markov Process (MP), it obeys the Markov property (i.e., a process depends only on the present state). It is formed by a 5-tuple, $< S, A, P, R, \gamma >$, where:

- S is a set of states of the environment. In practice, the state space can be continuous or discrete.
- A is the set of actions the agent may take. In practice, the action space can be continuous or discrete.
- $P : S \times A \rightarrow \mathcal{P}(S)$ is the transition probability matrix. Each value is a probability of going from a state s to another s' after taking an action a.
- $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function to maximize. In a timestep, it is expressed as $r_t = R(s_t, a_t, s_{t+1})$.
- $\gamma \in (0, 1)$ is the discount factor for balancing the importance of immediate and future rewards, and converging them to finite values.

A sequence of state-action pairs is called a trajectory or episode, τ. Thus, the goal of the agent is to maximize the reward over it, the return. There exist two

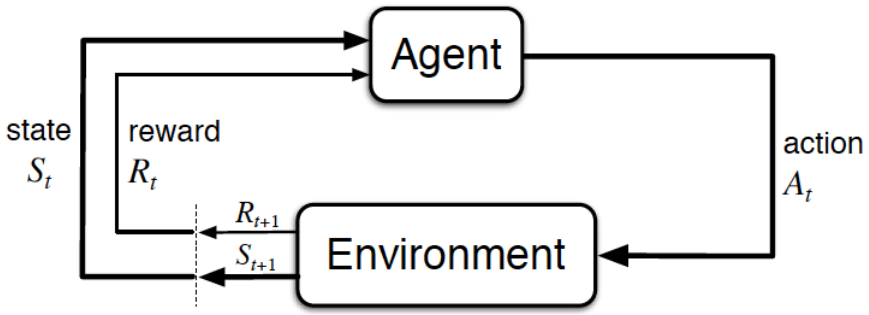


Figure 2.4: Framework of Reinforcement Learning.

Source: I. Galatzer-Levy et al. [83].

main types of return, discounted and undiscounted. In this project, the former is applied. It can be expressed as:

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \tag{2.1}$$

The agents must act accordingly to a policy to try to maximize the reward. A policy is a rule that the agent obeys to take an action based on a state. A stochastic policy is expressed as:

$$a_t \sim \pi(\cdot \mid s_t) \tag{2.2}$$

Nevertheless, there are also non-stochastic policies. Figure 2.5 shows a simplified taxonomy of RL methods. These strategies are fundamental to classify RL algorithms. Here is a short description of each of them:

- **Value-Based**: The value function of a state, $V^\pi(s)$, or the action-value function of a state-action pair, $Q^\pi(s, a)$, is learned based on the expected return. The policy is derived from these functions. They obey Bellman equations. Conventional models: Monte-Carlo, Q-Learning, or Deep Q-Networks (DQNs), the most well-known DRL model.

$$V^\pi(s) = \underset{\tau \sim \pi}{E}[R(\tau) \mid s_0 = s] \tag{2.3}$$

$$Q^\pi(s, a) = \underset{\tau \sim \pi}{E}[R(\tau) \mid s_0 = s, a_0 = a] \tag{2.4}$$

- **Policy-Based**: Instead of learning value functions, the policy is directly learned by mapping states to actions, $\pi_\theta(s, a)$. It has better convergence and better exploration properties, albeit higher variance. A usual way to fit the parametrized policy function is to perform gradient ascent. This branch is called policy gradient. Conventional methods: softmax policy, REINFORCE, actor-critic (though, they also use value functions).

$$\pi_\theta(s, a) = \mathbb{P}[a \mid s, \theta] \tag{2.5}$$

- **Actor-Critic**: It combines both policy-based (the actor), and value-based (the critic) methods. This MSc thesis employs one of them, a synchronously parallelized Advantage Actor-Critic (A2C).
- **Model-Free**: the model of the environment is not accessible to the agent (i.e., how the state transitions and rewards are computed). This method is far more widespread than its counterpart as it is agnostic to the problem.
- **Model-Based**: The agent can access to the environment's model. This helps the agent to plan ahead. However, most times, this information is not available.

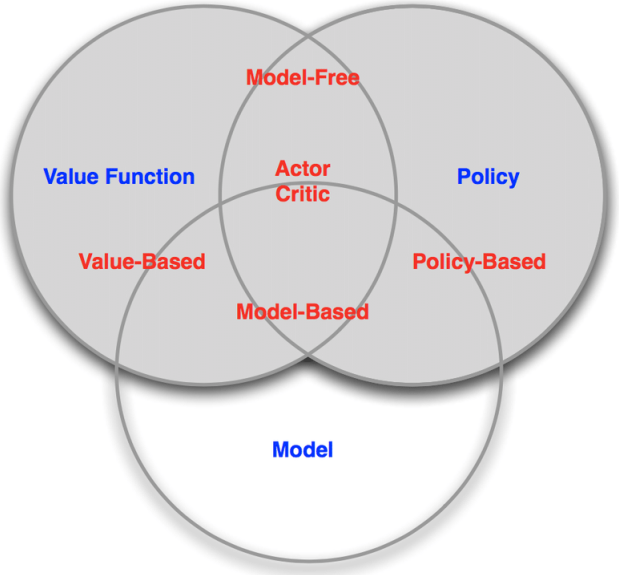


Figure 2.5: Simplified taxonomy of Reinforcement Learning.

Source: Medium [84].

Aside from this taxonomy, there are other considerations and dilemmas in RL algorithms that should be taken into account when designing and implementing these models:

- **Off-Policy vs. On-Policy**: In value-based methods, on-policy models update Q-values using the same policy they are following. On the other hand, off-policy models update accordingly to a different policy from the one they follow (e.g., the *greedy policy* is used in Q-learning). Policy-based methods are inherently on-policy models.
- **Online Learning vs. Offline Learning**: Although this dilemma is the same in supervised or unsupervised learning, it is also necessary to decide whether the RL algorithm should update as new data are gathered during an episode or once it ends.

- **Exploration vs. Exploitation**: Perhaps, the biggest dilemma in RL. On the one hand, an agent may want to exploit the known states that give decent rewards. On the other hand, it might also want to discover new states were the reward might be higher, albeit riskier as it may end up in a worse situation. There is plenty of research on this matter, primarily in exploring, as vanilla RL algorithms tend to get stuck in local minima.

So far, this introduction has covered conventional RL, but DRL differs from it by applying DL models. These DL models are utilized to estimate V-values and Q-values (value-based), and actions (policy-based). Hence, DNNs are function approximators in which the input is the state, and the output is either the actions or the values. DL works much better with high non-linear dimensional spaces than regular ML models. Thus, it yields far superior results in highly-complex problems. In DRL, the benefits are better appreciated with high-dimensional and continuous state and action spaces, and with unstructured data (e.g., images, audio, and text). Figure 2.6 depicts an updated RL framework for DRL.

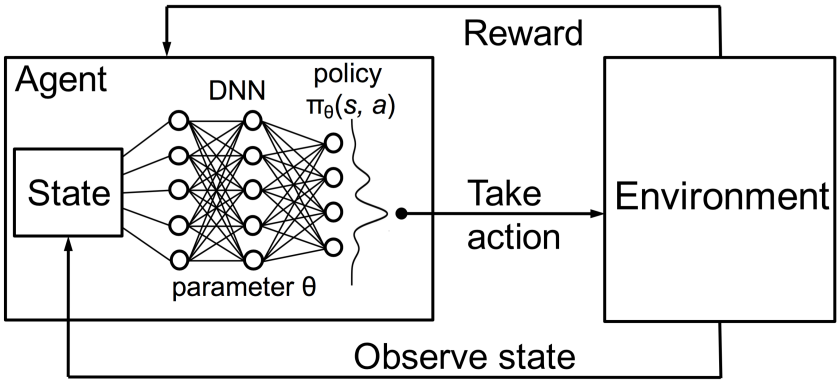


Figure 2.6: Framework of Deep Reinforcement Learning.

Source: S. Greenet al. [85].

Regarding EC, there exist some works that apply DRL techniques to solve optimization problems in MEC applications. J. Xu et al. [86] propose an RL algorithm to minimize delay and energy cost by offloading tasks to an energy harvesting MEC server. They mix online and offline learning to speed up training. X. Chen et al. [87] optimize a densely populated MEC scenario for offloading strategies based on task and energy queue state, and channel qualities between User Equipment (UE) and APs. They devise a double DQN to solve it.

D. Zheng et al. [88] propose a model-free DQN-based resource manager for EC applications, though it is not energy-aware as the only inputs are utilization and distance of edge servers. Z. Ning et al. [89] optimize a Vehicular Edge Computing (VEC) application using a two-way algorithm, being one of them a double DQN. VEC is a branch of EC that focuses only on applications for vehicles. They achieve a 90% execution-time reduction for the non-DRL-based part and 15% for the DRL-based part (though, this model was compared to an already simpler DQN model).

What makes our research and this MSc thesis different from these papers is the optimization objective and the EC scenario. On the one hand, their optimization focuses just on the utilization in MEC servers. The only contemplated energy costs are derived from latency issues due to saturation. Instead, our research provides a holistic view of the energy consumed in EDCs, both in IT equipment and cooling. On the other hand, their concept of edge resource infrastructure is limited to independent MEC servers, often with very limited CPU hardware. In our case, the EDCs are much more equipped with multiple GPUs each.

For these two reasons, our research envisions a more realistic perspective of the energy problem. These works usually employ unspecified EC applications to which they assign a utilization factor. On the contrary, this MSc thesis uses real and specific IT hardware and cooling power models based on tested utilization factors and a real and specific EC application. Hence, this work presents a comprehensive energy-conscious approach to EC.

Chapter 3

The Edge Computing Scenario

This Chapter introduces the EC scenario to be optimized. To this end, the EC simulator employed and the case of use are described. Section 3.1 covers Mercury, a 5G edge federation simulation tool with myriads of options, many of them related to energy consumption in EC. This section aims to explain the simulator's general idea and the parts needed to develop this MSc thesis. There are already excellent resources that delve into the possibilities of Mercury [4, 34]. Section 3.2 describes the EC case of use in which the optimizations are applied. It is a realistic ADAS application based on DL workloads.

From this Chapter onwards, we present our work. The key contributions are the design and implementation of (i) a DL-based ADAS application for an EC scenario (Section 3.2); (ii) power consumption models of air-based and immersion-based EDCs' IT equipment and cooling systems (Chapter 4); and (iii) energy-conscious optimizations using the abovementioned elements and DRL (Chapter 5).

3.1 Mercury, a 5G Edge Federation Simulator

Mercury is a Modeling, Simulation, and Optimization (M&S&0) framework. It provides a tool for the study of the dimensioning and operation of EC scenarios. Note that the authors use the term Fog Computing (FC) instead of EC. Although these concepts are different, they often overlap. Thus, they are frequently swapped in the Literature. It is a thoroughly verified system utilizing the Discrete Event System Specification (DEVS) mathematical formalism. Its key features are:

1. Wide variety of options to implement EC use cases.
2. Support for the mobility of edge devices.
3. Real-time 5G-based data stream with latency analysis.
4. Power consumption models of EC network's elements.
5. The option to implement resource management to optimize EC scenarios.

Figure 3.1 depicts the architecture of Mercury. The scenario is based on a single Radio Access Network (RAN). Currently, CC is not included, so the core network tasks are just network management and access control. The IoT devices are formally known as the UE. They run applications that need to be offloaded to EDCs of the edge federation. The offloading model is akin to the Function as a Service (FaaS) concept. For this purpose, resources are reserved temporarily, and only when UE requests it, creating a session. The means from which the UE connects to the RAN are the APs. Then, the data-stream created in a session is transferred to an EDC. In these infrastructures, the session is stored and computed with one of their Processing Units (PUs). The UE interacts always with the nearest AP. On the other hand, the AP does so with the EDC assigned to it by the core network. This procedure is performed by the Software-Defined Network (SDN) controller since it also receives the status of the EDCs.

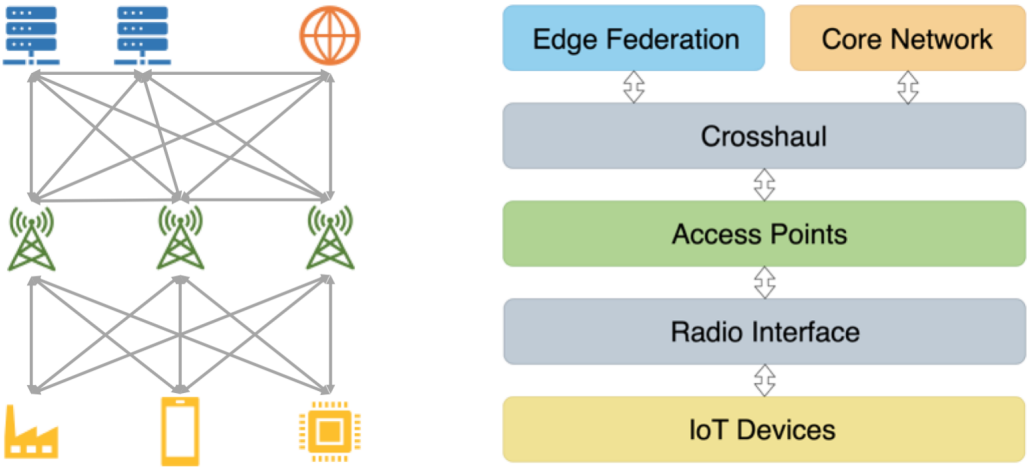


Figure 3.1: Fog Computing model of Mercury.

Source: R. Cardenas et al. [4].

One of Mercury's key features is its multiple customization options for defining the EC scenario. This opportunity allows us to model it in the most realistic way possible. In this MSc thesis, the EC application, EDCs, and SDN controller strategy are devised. The first one is introduced in the next Section. It is a data-intensive ADAS service that employs Convolutional Neural Networks (CNNs). The modeling of EDC's PUs (i.e., the IT equipment) and the cooling system are covered in Chapter 4. There are two EDC models, one air-based and another immersion-based. The SDN's allocation strategy for connecting APs and EDCs is included in Chapter 5. This strategy utilizes the DRL algorithm.

Apart from these three network elements, Mercury features many other customization options. Some of them are tweaked, and others are not in the following Section and Chapters. Those not mentioned in this MSc thesis are set to their value in a previous and tested use case [4].

3.2 Use Case: Advanced Driver Assistance Systems

The conceived case of use employs a data-intensive ADAS application well suited for a VEC scenario. It was first proposed in our previous publication [32] and further tested in Mercury [4]. The concept of offloading ADAS and autonomous driving services from vehicles is nothing new. For instance, Amazon offers CC solutions to host DL frameworks for ADAS [90].

In this use case, each vehicle represents a UE that utilizes a DL-based ADAS service. This service alerts drivers when they lose concentration on the road. A CNN-based algorithm estimates the driver's head position using video footage. If drivers spend an excessive amount of time looking to the sides, the algorithm warns them. Hence it aims to improve road safety.

The selected ADAS dataset (K. Diaz-Chito et al. [91]) fulfills the specifications mentioned for this practical EC application. It contains SD images (i.e., 640x480 pixels) of several drivers with different facial features. Each driver also has multiple images for each head position (left, right, and frontal). The photos are tagged with the driver's head position. Figure 3.2 shows the use case scenario so far.

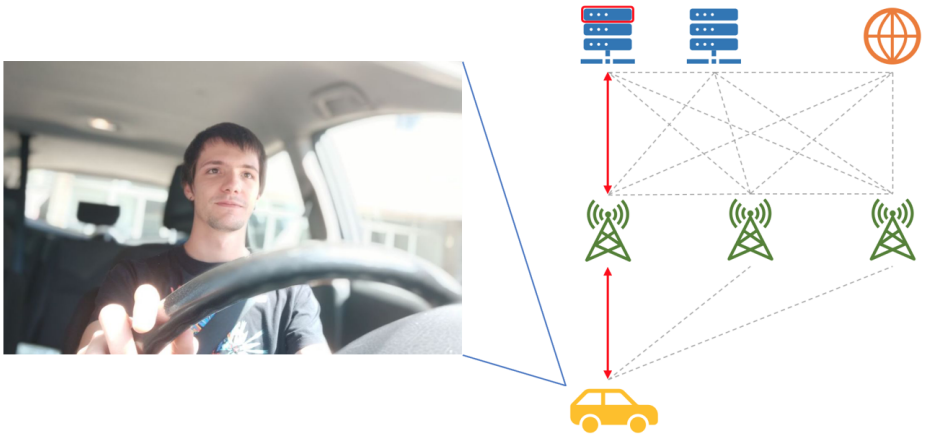


Figure 3.2: Devised ADAS scenario.

Source: R. Cardenas et al. [4].

The session created by the UE is for training the algorithm, instead of making predictions. The training of DL-based algorithms is known for being computationally expensive. Therefore, leveraging the EDC's resources reduces economic costs in the vehicle's hardware as predicting needs much less processing power. Moreover, the quality of the estimations improves as the algorithm updates more frequently with minimal delay thanks to the EDC federation layout.

Aside from the application, Mercury needs traces of the UEs, the vehicles, to work. To achieve the most realistic simulation possible, the case of use features real mobility traces (M. Piorkowski et al. [92]) from taxis in San Francisco Bay Area, California, USA. These data have already been processed for their use in Mercury [4]. To this end, the authors extracted traces from 100 taxis of the day with the highest

traffic flow. Then from these data, they only kept the ten busiest minutes from the peak hour. Finally, an interpolation to obtain traces every ten seconds is computed. Figure 3.3 depicts the satellite view of the area and locations of the UE.

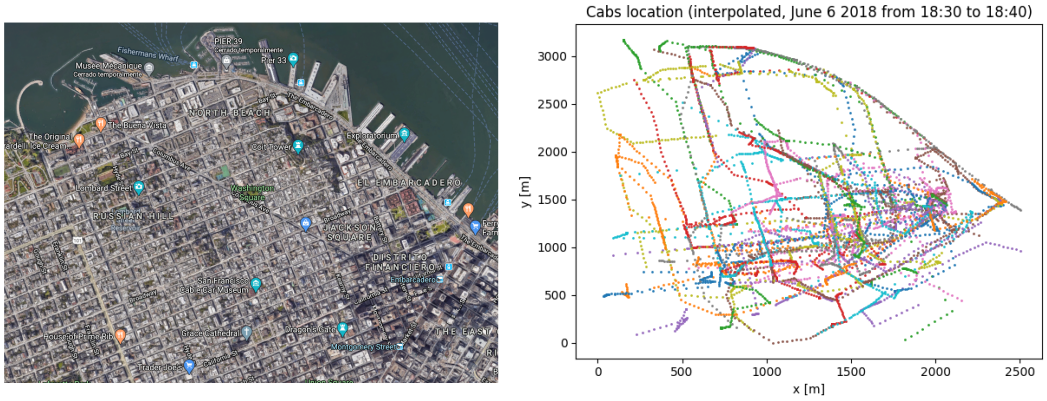


Figure 3.3: Satellite view and UE location in the San Francisco Bay Area.

Source: Google and R. Cardenas et al. [4].

As mentioned in the previous Subsection, UE creates a session that is conveyed to one of the EDCs' PUs via the APs. Hence, APs and EDCs need to be defined. This MSc thesis again mimics an already tested configuration [4]. In this layout, there are 3 EDCs and 10 APs. The authors propose a modified K-means algorithm to lay both the APs and EDCs in the most efficient fashion location-wise. Figure 3.4 shows the final results. Note that the stars are the APs, and the circles are the EDCs.

Figure 3.4: APs' and EDCs' location.

Source: R. Cardenas et al. [4].

The Figure also depicts the EDCs' area of influence using the only SDN controller strategy implemented so far. The location-aware algorithm assigns EDCs to the closest AP. However, it lacks any notion of consumed energy in the network. This MSc thesis proposes a DRL algorithm capable of learning energy-aware distributions of the UE's sessions. The current location-aware strategy serves as a baseline for our energy-aware optimizations.

Chapter 4

Edge Data Center Modeling

This Chapter covers the design and implementation of EDCs for the EC scenario presented in Chapter 3. The power consumption models are added to Mercury for optimizing the use case in Chapter 5.

There are two types of EDC, one based on an air system and another based on a two-phase immersion system. For each of them, the IT equipment and cooling system are modeled with data of real hardware to achieve the most realistic simulations. Section 4.1 presents the air-based EDC while Section 4.2 introduces the immersion-based one. The main software tools utilized in the modeling are:

- **Programming Language**: Python 3 [93].
- **IDE**: Jupyter [94], and VS Code [95].
- **DL Platform**: ROCm [96], and PlaidML [97].
- **Modeling**: TensorFlow [98], and Keras [99].
- **GPU Monitoring**: Collectd [100], Apache Kafka [101] and Cassandra [102].
- **Data Processing**: NumPy [103], SciPy [104], and Pandas [105].
- **Data Visualization**: Matplotlib [106], Seaborn [107], and Plotly [108].

4.1 Air-Based Edge Data Center

This Section focuses on the air-based EDC power models for IT and cooling. In this data center, the PUs are Sapphire Pulse Radeon RX 580 GPUs [33]. The model includes a FeedForward Neural Network (FNN). The cooling system is based on a Hot Aisle/Cold Aisle configuration devised by J. Moore et al. [109]. For each EDC part, its architecture and power model is presented.

4.1.1 IT Equipment

The model of the air-based EDC's PUs has already been validated by the research community in one of our previous publications [32]. Besides, this model has already been tested in Mercury [4]. As mentioned before, the hardware utilized is a Sapphire Pulse Radeon RX 580 GPU [33]. The model's goal is to parametrize GPU's energy behavior using the devised ADAS application in Section 3.2. This solution has three well-distinguished parts: the ADAS application that runs in the GPU, the monitoring system that collects GPU's data during runtime, and the GPU's power model that fits the collected GPU data.

The first part, the ADAS application, is a classification task using the ADAS dataset described in Section 3.2 and a CNN algorithm. The CNN model is inspired by VGG16 [110], a well-known model in Computer Vision (CV), and was used for the CIFAR-10 dataset [111]. It is a reduced version with just two VGG16-like convolutional blocks and one fully connected layer instead of the original five blocks and two layers. Reducing the model complexity and size enable us to run DL models in an AMD GPU, which were the only available and do not notably support DL applications. Moreover, at that moment, the only option was to utilize the DL backend PlaidML [97]. This limited software ran into many memory problems (namely, segmentation faults) and lacked the support to fix them. Nevertheless, some tweaks solve the issues to achieve an interestingly enough model.

The second part, the GPU's monitoring system, is comprised of a Collectd [100] script to gather the data, a Kafka [101] topic to convey them, and a Cassandra [102] database to store them finally. During the ADAS workload's execution, the monitoring system runs in parallel, collecting data of temperature, power, clock, memory clock, voltage, utilization, memory utilization, and number of parallel CNN models or sessions. During monitorization, the ADAS application is executed for each possible combination of the clock (eight frequencies), memory clock (three frequencies), and parallel sessions (up to 5 parallel CNNs). The final dataset included 6,420 traces.

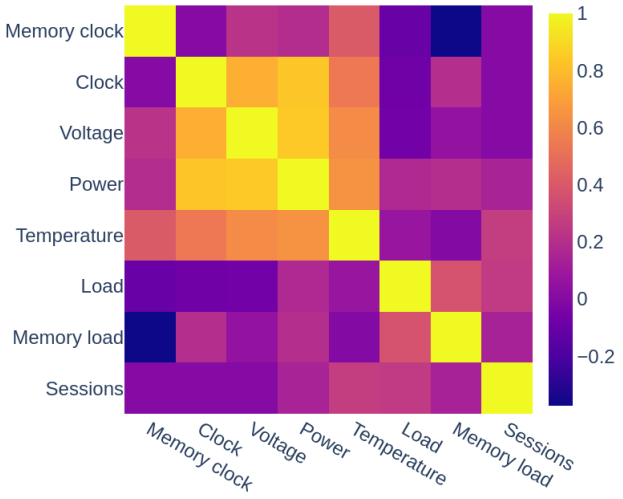


Figure 4.1: Correlation heatmap of the air-based dataset.

An Exploratory Data Analysis (EDA) of the variables is carried out to gain insights and clean the dataset. Figure 4.1 shows a correlation heatmap from this EDA. Here, it is clear that power consumption is highly correlated to temperature, voltage, and the clock. On the other hand, variables related to the workload intensity, such as utilization (load in the Figure), and parallel sessions are practically uncorrelated.

The third and final part, the GPU's power model, is a rather small FNN model. The model's inputs are the clock, memory clock, and the number of parallel sessions, while the model's output is the consumed power. Its low complexity enhances the model's speed, which also translates into quicker simulations in Mercury. During training, special care was taken tuning the hyperparameters to avoid overfitting and underfitting. Table 4.1 shows the model's hyperparameters. Figure 4.2 depicts some predictions using instances from the test set.

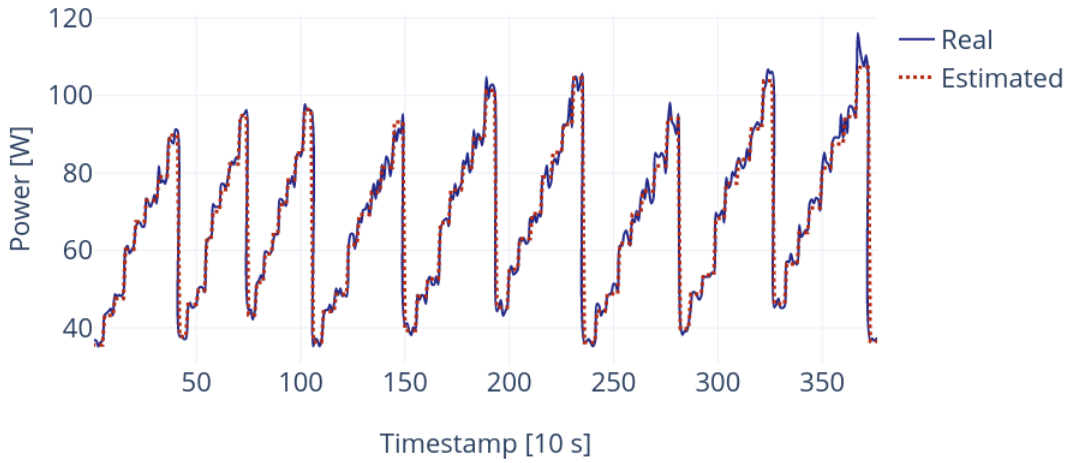


Figure 4.2: Air-based GPU power model's predictions.

Source: S. Pérez et al. [32].

For assessing the model, the Normalized Root Mean Square Deviation (NRMSD) (Equation 4.1) and Coefficient of determination (R^2) (Equation 4.2) were employed. Both are commonly-used metrics in regression tasks. The NRMSD metric provides a way to compare datasets with different scales by normalizing the Root Mean Square Deviation (RMSD). The RMSD is a well-known metric that penalizes higher errors more. The R^2 metric measures the proportion of variance that can be predicted from the dependent variable using the independent variable(s). Higher coefficients mean better results. Table 4.2 shows the final evaluation metrics. The model shows great results.

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(P_i - \hat{P}_i)^2} \quad NRMSD = \frac{RMSD}{P_{max} - P_{min}} \cdot 100\% \tag{4.1}$$

where:

P_i = real power consumption sample [W]
$\hat{P}_i$ = predicted power consumption sample [W]
P_{max}/P_{min} = maximum/minimum power consumption [W]

$$R^2 = \frac{\sum_{i=1}^{N}(P_i - \hat{P}_i)^2}{\sum_{i=1}^{N}(\bar{P}_i - P_i)^2} \cdot 100\% \tag{4.2}$$

where:

P_i = real power consumption sample [W]
$\hat{P}_i$ = predicted power consumption sample [W]
$\bar{P}_i$ = mean power consumption [W]

Table 4.1: RX580 air-based power model's hyperparameters.

Hyperparameter	Value
Hidden layers	x2 Feedforward
Neurons	32, 12
Train-Val-Test split	70%, 20%, 10%
Scaling method	Min-Max
Scaling range	[-1,1]
Batch size	5
Epochs	140
Loss function	Mean Square Deviation (MSD)
Activation function	ReLU
Optimizer	Nadam
Learning rate	0.002

Table 4.2: RX580 air-based power model's evaluation metrics.

Metric	Value
$NRMSD$	2.45%
R^2	99.01%

4.1.2 Cooling System

The EDC's cooling energy model is based on work from J. Moore et al. [109]. Their project models a data center's room with 28 racks distributed in four rows of seven racks. To cool them, it has 4 CRAC units, one on each side of the room. Hence, it envisions the most conventional cooling strategy, Hot Aisle/Cold Aisle. Figure 4.3 depicts the layout.

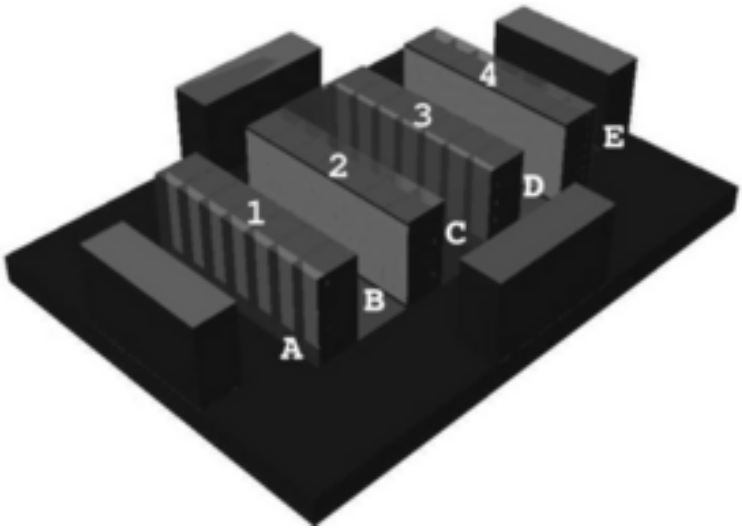


Figure 4.3: Air-based data center layout for the cooling model.

Source: J. Moore et al. [109].

The model's parameters are just the IT power and the inlet temperature. The authors employ the concept of coefficient of performance (COP) for their model. COP is often used to measure the efficiency of CRAC units' periodic cooling cycle [23]. Equation 4.4 shows the final model. In Chapter 5, a brief analysis of the model's quality comparing it to a real air-based SOTA solution is performed.

$$COP = 0.0068 \cdot T_{inlet}^2 + 0.0008 \cdot T_{inlet} + 0.458 \tag{4.3}$$

$$P_{cooling} = \frac{P_{IT}}{COP} \tag{4.4}$$

4.2 Immersion-Based Edge Data Center

This Section explains the modeling of a two-phase immersion cooling EDC. As in the air model, the data center consists of the IT equipment and the cooling system. This time, the PUs are Sapphire Pulse Radeon RX 570 GPUs [112], from the same commercial series as the RX580 model. The real pump installed in our two-phase immersion cooling tank is employed for the cooling model. The tank is developed jointly by our research team and commercial partners[1].

4.2.1 IT Equipment

As mentioned in the introduction, the modeled GPU is now the Sapphire Pulse Radeon RX 570. The characteristics are akin to the RX580 model, albeit a bit less

[1]Contract: IDI-20171194 supported by CDTI. Participants: 3M España S.A., UPM.

processing and consumed power. While the method to obtain the GPU's power model is practically the same as in the air-based EDC, the GPU's infrastructure is completely overhauled to fit into a two-immersion cooling system. Figure 4.4 shows the designed prototype that was used to produce the GPU's power model.

Figure 4.4: Two-phase immersion cooling prototype.

This prototype features a heat exchanger mechanism, unlike in previous publications [29]. The system circulates cold water that passes through two liquid cooling blocks. On the surface of both blocks, the gas originated from the coolant's evaporation during runtime is condensed. The main GPU's chip has a graphite and copper sheet mechanically attached to distribute the heat from the primary source better.

The use of a thermal camera revealed that there were no gas leakages around the sealing and that the coolant can reach more than 61°C (i.e., the boiling point). This was further proved during the workload's execution by the two-phase periodic cycle that the coolant goes through.

As mentioned in Section 4.1, the GPU's power model is a three-part solution: the ADAS application, the monitoring system, and the power model itself. The first

part was partially changed to enhance the support for DL applications in this AMD GPU. As stated before, AMD lacks the support for DL tasks that other companies like NVIDIA have.

The best option is to employ TensorFlow [98] with ROCm [96] as the backend. TensorFlow is often referred to as the most widely-used DL tool. To this end, a new computer was needed that included at least a fourth-generation Intel Core CPU or above and a PCI Express 3.0 x16 connection.

Moreover, after tweaking both TensorFlow and ROCm configurations, the highest number of parallel ADAS session before running into memory problems was four instead of five as in the air model. The computer needed more than 16GB of Random-Access Memory (RAM) to surpass these sessions. A memory was purchased, but due to the current pandemic, it was not possible to test it with the new RAM. Nonetheless, a dataset with up to four parallel sessions was obtained, which is used in this MSc Thesis.

The GPU's power model follows the same strategy as in the air-based model. Table 4.3 and Table 4.4 shows the model's hyperparameters and the final results, respectively. Compared to the air-based model, the results are practically the same. Figure 4.5 depicts the model's training and validation curves (left) and the power consumption predictions using some samples from the test set.

It presents a stable training without overfitting and overfitting issues. As the model's inputs are the combination of three discrete variables, the model's outputs are also discrete. Bear in mind that in Chapter 2, there is plenty of discussion about the benefits of working at the ideal temperature range. So far, our single-GPU architecture is not able to reflect this behavior. This potential enhancement is part of our future directions discussed in Chapter 7.

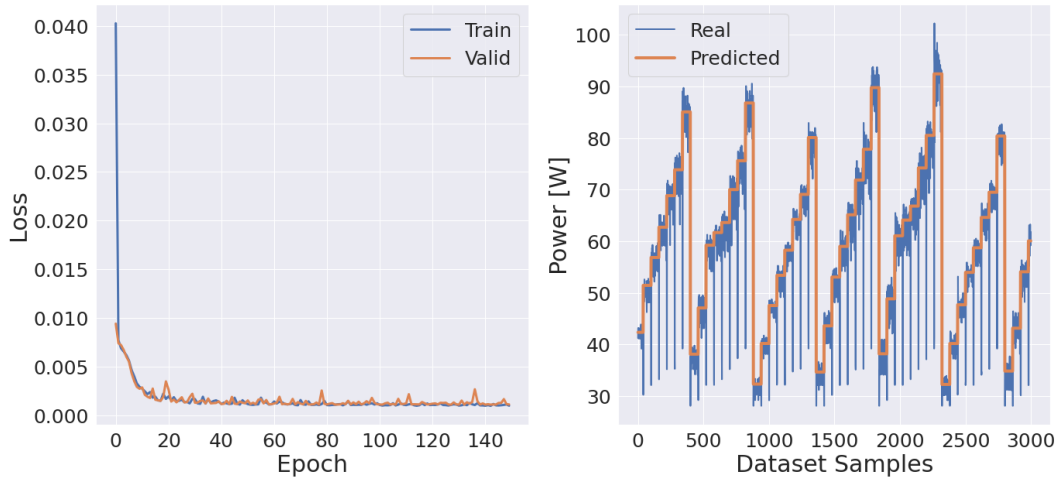


Figure 4.5: Two-phase GPU power model's loss curves and predictions.

Table 4.3: RX570 immersion-based power model's hyperparameters.

Hyperparameter	Value
Hidden layers	x1 Feedforward
Neurons	512
Train-Val-Test split	70%, 20%, 10%
Scaling method	Min-Max
Scaling range	[0,1]
Batch size	128
Epochs	150
Loss function	MSD
Activation function	ReLU
Optimizer	Nadam
Learning rate	0.001

Table 4.4: RX570 immersion-based power model's evaluation metrics.

Metric	Value
$NRMSD$	3.15%
R^2	97.97%

4.2.2 Cooling System

As discussed in Chapter 2, two-phase cooling systems consume very little energy. The only significant element that needs consideration is the pump that recirculates the heat exchanger's water. In this MSc thesis, the pump installed in our real EDC prototype, developed by one of our partners, is modeled. The pump's power model is based on the datasheet provided by our partner. The information can be found in Annex C. There is a caveat, it is written in Spanish.

In this documentation, the two bottom figures depict the pump's shaft power (P_s) and efficiency (η_m) as a function of the flow rate. The pump's power consumption (i.e., motor power, P_m) is derived from these two variables, as in Equation 4.5 [113].

$$P_m = \frac{P_s}{\eta_m} \tag{4.5}$$

Figure 4.6 shows the resulting motor power for the pump's range of flow rates. Pumps usually achieve their highest efficiency around the middle of the flow range, coinciding with the impeller at its full size [114]. Pumps are meant to work at maximum efficiency, not just to save energy, but also to avoid potential damage because of saturation.

The last step is to compute the water flow rate as a function of the heat dissipation and the temperature difference between input and output in the two-phase immersion sealed tank. Figure 4.7 depicts this simplified scenario. The heat transfer, Q, is the heat generated on the GPU's chips, thus the IT power consumption. Since the consumed energy depends on the IT demand, the only variable of the flow rate, for optimizing the pump's power consumption, is the temperature difference. Equations 4.6 through 4.8 show how to compute the flow rate through a circular pipe as a function of the temperature difference in the measurement units used in the commercial pump [115, 116].

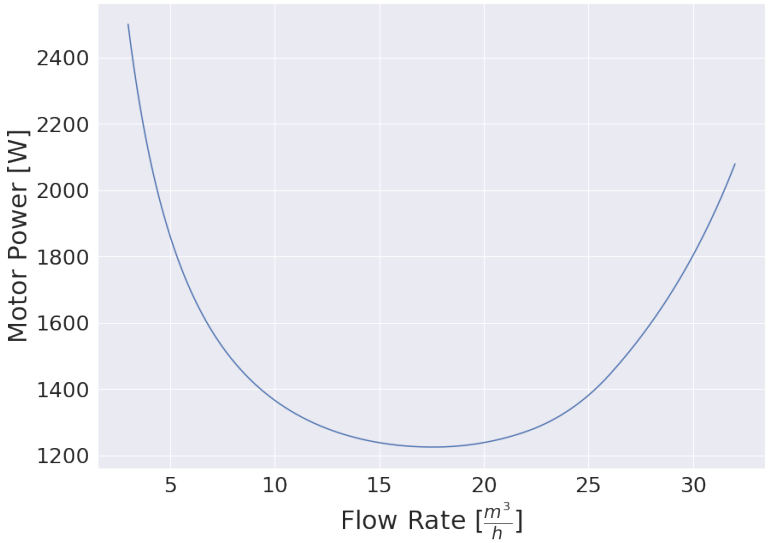


Figure 4.6: Pump's power consumption as a function of flow rate.

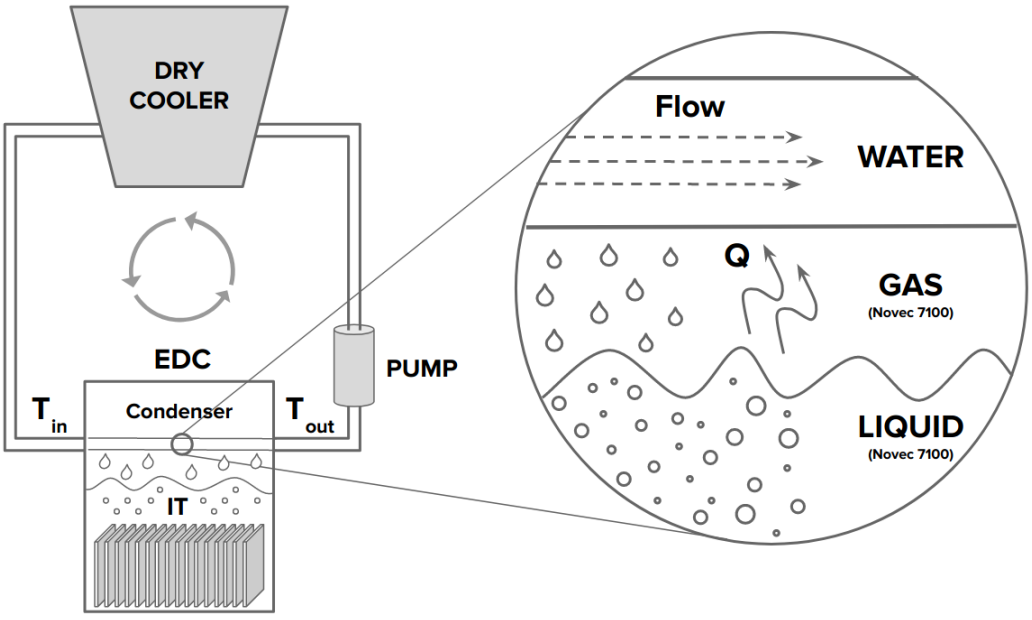


Figure 4.7: Simplified heat transfer scenario.

$$Q = f_m \cdot C_p \cdot \Delta T \tag{4.6}$$

where:

Q = power dissipation by the IT equipment [W or J/s]
f_m = mass flow rate [g/s]
C_p = specific heat capacity [4.18 J/g°K]
$\Delta T = T_{out} - T_{in}$. Temperature difference [°K]

$$f = \frac{f_m}{\rho} \tag{4.7}$$

where:

f = flow rate [cm^3/s]
f_m = mass flow rate [g/s]
ρ = density of water [1 g/cm^3]

$$f = \frac{Q}{277.78 \cdot \rho \cdot C_p \cdot \Delta T} \tag{4.8}$$

where:

f = flow rate [m^3/h]
Q = power dissipation by the IT equipment [W or J/s]
ρ = density of water [1 g/cm^3]
C_p = specific heat capacity [4.18 J/g°K]
ΔT = $T_{in} - T_{out}$. Temperature difference [°K]
$1\, m^3/h = 277.78\, cm^3/s$

Chapter 5

Energy-Conscious Optimization

This Chapter covers the design and implementation of the energy-conscious optimizations. To this purpose, the SDN controller is devised as a DNN that works as the policy of a DRL agent. This strategy is tested in different configurations of Mercury with the designed EDCs in Chapter 4.

Section 5.1 presents the resource allocation manager (i.e., the SDN controller). It features an introduction to the optimization algorithm, a synchronously parallelized A2C, and how it is applied to this case of use. Section 5.2 describes different EC scenario configurations to be optimized using the SDN controller, the ADAS use case, and the two EDC types. The primary software tools employed in the optimization are:

- **Programming Language**: Python 3 [93].
- **IDE**: Jupyter [94], and VS Code [95].
- **EC Simulation**: Mercury [4, 34].
- **DRL Modeling**: PyTorch [117].
- **Data Processing**: NumPy [103], and Pandas [105].
- **Data Visualization**: Matplotlib [106], and Seaborn [107].

5.1 Resource Allocation Manager

As stated before, this Section covers the SDN controller that manages the connections between APs and EDCs, hence the allocation of computing resources. The selected DRL algorithm to implement the energy-aware SDN controller is a synchronously parallelized A2C. It is a widely-used RL model from the actor-critic family. The algorithm's theory, practical implementation, DL-based architecture, and reward function are discussed below.

5.1.1 Actor-Critic Theory

The chosen family of RL algorithms is the actor-critic. These models combine both policy-based (the actor) and value-based (the critic) methods. In DRL Literature, they often antagonize with Q-learning as both are the foundation of the most advanced algorithms in the field. There is a trade-off between the two approaches. Among many aspects, the former inherently is more exploration-efficient and has better convergence properties (since they partially are policy-based), while the latter is more sample-efficient [118].

This MSc thesis utilizes actor-critic algorithms for several reasons. First, Mercury produces new data samples frequently, so being sample-efficient is less of a concern. Second, better exploration helps to speed up the process. Third, in some DRL baseline problems such as Atari games, actor-critic solutions have proved quicker [118]. Fourth, most of the reviewed research in EC that leverages DRL uses Q-learning, as it is usually a more popular option to start. This master's thesis provides the perfect opportunity to try a different approach. Finally, fifth, the implementation is more straightforward than Q-learning, helping avoid potential code-related issues.

Broadly speaking, in actor-critic algorithms, the actor selects the action to take at each timestep, while the critic assesses how good the selection was. These methods are also classified as policy gradients since the actor uses gradient ascent to update its parameters. As stated in Chapter 2, policy gradients, and policy-based method in general, map directly states to actions with a parametrized policy function, $\pi_\theta(s, a)$. Then, the function's parameters are updated as follows:

$$\Delta\theta = \alpha \nabla_\theta J(\theta) \tag{5.1}$$

where:

$\Delta\theta$ = parameters update
α = step-size parameter
$J(\theta)$ = any policy objective function
$\nabla_\theta J(\theta)$ = the policy gradient

Each actor-critic algorithm differs on how the update is computed. Generally, the common denominator is the use of likelihood ratios in the policy gradient (i.e., the direction in which the gradient moves). This trick easies the computation of estimations needed in stochastic policies. The resulting term is known as the score function, $\nabla_\theta log \pi_\theta(s, a)$. This concept is widely used in other statistics and ML problems. In this context, the term comprises the actor. Hence, the critic is what separates actor-critic algorithms.

The most common methods are REINFORCE, A2C, and Asynchronous Advantage Actor-Critic (A3C). REINFORCE is the simplest of the group. It uses rewards directly instead of a value function. It translates to higher variance while

using a function would mean higher bias (although, it can balance the trade-off). Equation 5.2 shows how the update of parameters works in REINFORCE models for a particular timestep, t.

$$\Delta\theta_t = \alpha\nabla_\theta log\pi_\theta(s_t, a_t)R_t \tag{5.2}$$

Where the new term, R_t, is the return (i.e., the discounted or undiscounted reward up to that timestep). This part of the update is the critic. Note that some researchers might not call it critic in this particular algorithm as it is not a value function. The critic is improved in two ways to reduce the noise (i.e., the high variance). It uses a value function, instead of the returns, called the target, and a baseline. The combination of both is often referred to as the advantage function (Equation 5.3).

$$A(s, a) = Q_w(s, a) - V_v(s) \tag{5.3}$$

where:

$A(s, a)$ = the advantage function
$Q_w(s, a)$ = the critic's target, a parametrized action-value function
$V_v(s)$ = the critic's baseline, a parametrized value function

Roughly speaking, the advantage function indicates how good it is to take a specific action in a state, $Q_w(s, a)$, compared to the overall state value, $V_v(s)$. In practice, the two new sets of parameters v, w are reduced to just one by using as the target either the return again or Temporal Difference (TD) learning (a method to bootstrap the return). The final advantage function parameters are often updated by minimizing the mean squared error of its difference. As opposed to REINFORCE, A2C algorithms employ the advantage function as the critic to update the policy's parameters (Equation 5.4).

$$\Delta\theta_t = \alpha\nabla_\theta log\pi_\theta(s_t, a_t)A_w(s_t, a_t) \tag{5.4}$$

Even with the advantage function, high variance may slow down the learning. Parallel environments are utilized to alleviate this problem. If the policy is updated synchronously (all environment update simultaneously) is a parallelized A2C.

However, if the updates are independent (i.e., asynchronous), it is called A3C. For a long time, A3C seemed the preferred option, but a study by OpenAI [119], one of the most renowned DRL research teams, suggested otherwise. They found that the synchronous model has a more stable learning process since the amount of data in each update is more significant. Another characteristic was that A2C is better suited for GPUs, and A3C is for CPUs. Aside from these three models, other advanced techniques are found in the Literature since RL is gaining much attention lately. However, the most widely-used and the foundation of other actor-critic methods are these three.

For the reasons mentioned above, this project employs a synchronously parallelized A2C. The next Subsection covers the implementation of an A2C agent as an energy-aware resource allocation manager in Mercury.

5.1.2 Actor-Critic Implementation

So far, the explanation has been problem-agnostic. This time, an A2C model is devised as Mercury's resource allocation manager (i.e., SDN controller) for the implemented ADAS scenario. The Subsection is divided into several parts to facilitate comprehension.

Problem Definition

To better understand the RL model, its goal needs to be defined first. As stated in Chapter 3, the SDN controller manages how the network is configured between APs and EDCs. The APs also connect to the UE (i.e., the vehicles), which request sessions to process their data-intensive applications (i.e., the ADAS application). Hence, the SDN controller operates the resource allocation of these sessions in the EDCs' PUs through the AP-EDC mapping.

During the simulation, the SDN controller updates periodically and one-by-one the connections between APs to EDCs. To decide how to assign them, the SDN controller receives the status of the EDCs. This information features power consumption, utilization, and the distance to the AP to be assigned. Here the A2C model fulfills the SDN controller's role. Therefore, the RL agent's goal is to manage the AP-EDC mapping using the information from the EDCs to be as energy-efficient as possible.

Reinforcement Learning Formulation

After describing our goal, the SDN controller task, that the A2C model is in charge of, needs to be formulated as an RL problem. The different RL elements mentioned in the introduction in Chapter 2 are devised as follows:

The Environment: It is the ADAS application simulated using Mercury, not only the SDN controller where the DRL model is implemented, but the entire scenario.

State Space: The states are comprised of the information from the EDCs. In particular, IT (P_{IT}) and cooling (P_{cool}) power consumption, power demand (P_{dem}, the sum of the former), and the utilization (u) are used for each of the three EDCs. Hence, the total state space size amounts to twelve. The distance to the AP was also considered but ultimately discarded. Our focus remains mostly on energy-awareness.

Action Space: In each timestep, the action that the algorithm takes is the selection of an EDC for the requested AP. Thus, the action space is comprised of the three EDCs in the scenario.

Reward Function: Our idea is to minimize the total power consumption to maximize energy savings. The optimization problem is described as:

$$\min \sum_{i=1}^{3} P_{IT,i} + P_{cool,i} \tag{5.5}$$

Consequently, in a timestep, the reward function that minimizes the consumed energy is denoted as:

$$r_t = R(s_t, a_t, s_{t+1}) = -\sum_{i=1}^{3} P_{IT,i,t+1} + P_{cool,i,t+1} = -\sum_{i=1}^{3} P_{dem,i,t+1} \tag{5.6}$$

Note that for a state-action pair, the reward is based on the state that follows. This simple function proved the most successful. Other variations were tested, such as applying a penalty to higher utilization factors to avoid saturation (Equation 5.7) or using the difference between consecutive states (Equation 5.8). These strategies either hindered the goal (i.e., sheer energy optimization) or achieve similar results with more complex formulas.

$$r_t = \begin{cases} -(1+U)\sum_{i=1}^{3} P_{dem,i,t+1} & u_{a,t} = 100\% \\ -\sum_{i=1}^{3} P_{dem,i,t+1} & \text{otherwise} \end{cases} \tag{5.7}$$

where:

U = utilization penalty, ranges from 0 to ∞
$u_{a,t}$ = chosen EDC (i.e., action a) utilization

$$r_t = -(\sum_{i=1}^{3} P_{dem,i,t+1} - \sum_{i=1}^{3} P_{dem,i,t}) \tag{5.8}$$

The Agent: The agent refers to the policy most of the time. As mentioned before, the A2C policy maps directly states to actions. Hence at every timestep, the input is the state, and the output is a probability distribution of the actions. The chosen action is sampled from this distribution. This method enhances exploration. The policy's parametrized function that maps states and actions is a two-headed FNN (Figure 5.1), a DL-based architecture. This modified FNN has two outputs, the stated distribution, and the value function. As explained in the previous Subsection, two parametrized functions are needed for A2C models. First, the policy and second, the advantage function. In practice, both can be obtained from the same function. Note that the Figure employs the term $V(s)$ instead of $A(s, a)$ for the advantage value. This notation is because only the baseline function, $V(s)$ is parametrized. For the target $Q(s, a)$, the return, R, is employed.

In this architecture, there are shared and independent FNN layers between the two outputs. The gradient to fit the layer's parameters is propagated jointly. This method might seem a non-sense since these outputs differ widely. However, it has proved successful in renowned RL projects such as AlphaGo [120]. The DNN's hyperparameters were first fixed to values found in other tested works [118, 121, 122], and then modified using random search manually.

Table 5.1 shows the hyperparameters. The architecture is rather small to reduce the impact on the simulator's speed performance as much as possible. It is worth noting that using only shared layers, or independent layers (i.e., two completely different FNN) yielded similar results. Other important hyperparameters are covered in the training procedure, such as the loss function, optimizer, batch size, and feature scaling strategy.

Another aspect of the agent's implementation is the frequency with which it takes actions within Mercury's simulations. This simulator features a deep granularity, so the AP-EDC mapping update occurs several times within a second. In addition to this high rate, the process is independent for each AP. On the other hand, the frequency of new UE requesting services to the EC network is significantly lower. For this reason, the updates are reduced by selecting the same EDC for all APs within a timestep. This strategy decreases the necessary interactions with the agent by 90% and improves the model stability while maintaining Mercury's granularity.

Finally, if the agent selects an EDC whose utilization is at 100%, the chosen EDC changes to the closest to the AP since it is the baseline SDN strategy.

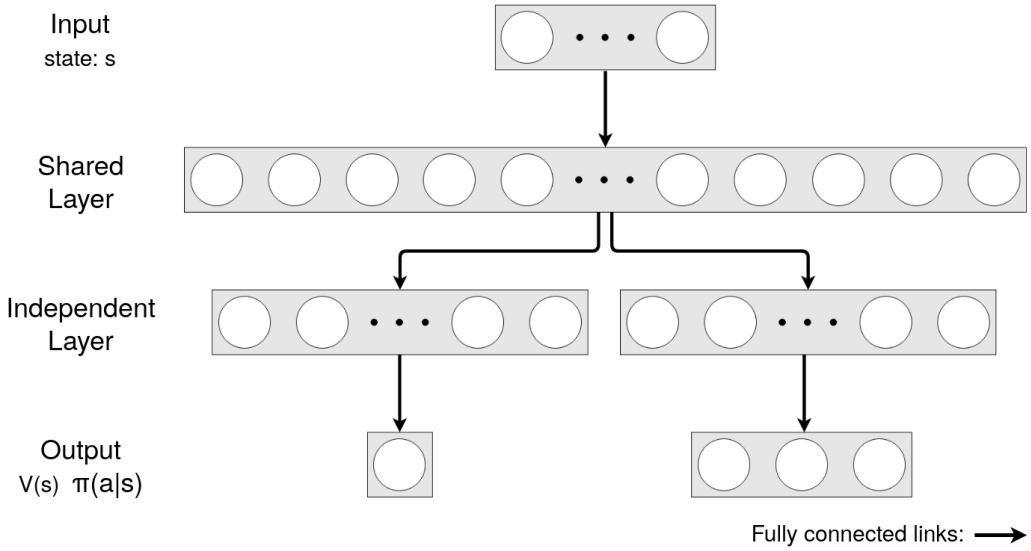


Figure 5.1: Two-headed FeedForward Neural Network.

Table 5.1: Two-headed FeedForward Neural Network's hyperparameters.

Hyperparameter	Value
Input size	12 (state space)
Output size	3+1 (action space + $V(s)$)
Hidden layers	x1 shared, x1 independent
Neurons	128, 64
Activation function	ReLU, Softmax

Training Procedure

After describing the problem and DRL model, the next part is devising how the model is trained to fit the DNN's parameters and achieve the best results. The procedure is again based on tested works that utilize A2C models [118, 121, 122] and tweaked to the specific requirements of this environment. Algorithm 5.1 summarized this process.

The first element to consider is the feature scaling strategy. This operation has a notable impact on the speed convergence of DL models. Since RL problems are inherently dynamic, feature scaling is not as simple as in supervised learning, in which datasets are often fixed. In a dynamic environment, there are three options: no scaling, neural network regularization, and fixed scaling parameters. This MSc thesis employs the latter. Once the EC computing scenario to be optimized has been configured, a first simulation is run to obtain the scaling parameters utilized for the training process. Hence, the mean and variance are computed from this first simulation to perform feature standardization:

$$X_s = \frac{X - \mu}{\sigma} \tag{5.9}$$

where:

X, X_s = original and scaled input features (i.e., the state)
μ, σ = mean and standard variation from preliminary simulation

The training starts after obtaining these scaling parameters. The first step is to initialize the DNN's parameters randomly. Then, several parallel Mercury instances are launched. As stated in the actor-critic theory, this method reduces the high variance of these models. The number of parallel simulations was set to three since several experiments showed no further improvements for higher numbers. Another factor was the computer's health that runs Mercury since parallelizing too many instances may be detrimental. Then, the DNN is broadcasted to all environments. In each simulation, the trajectory containing states, actions, and rewards is stored for later use.

Once all the simulations are over, the return of each timestep of each trajectory is computed (Equation 5.10). The discount factor, γ, is set to 0.99 to

give similar importance to immediate and future rewards. In theory, this way of computing the return is enough, but sometimes it is common practice to standardize the return to speed up the convergence. This procedure might also mess up the training procedure, but our experiments showed that it was beneficial. In our case, it is not only the return standardized but also the reward of each trajectory. This decision was made after analyzing the return. It was biased towards the rewards at the end of trajectory due to the rewards being inherently negative. The return should resemble the reward since, in this case, it is the discounted cumulative reward in a trajectory. In Figure 5.2, reward standardization (right) makes the return akin to the reward, eliminating the bias (left).

$$R_t = \sum_{t^{'}=t}^{T-1} \gamma^{t^{'}-t} r_{t^{'}} \tag{5.10}$$

where:

R_t = return in timestep t
$r_{t^{'}}$ = reward in timestep $t^{'}$
T = length of the trajectory

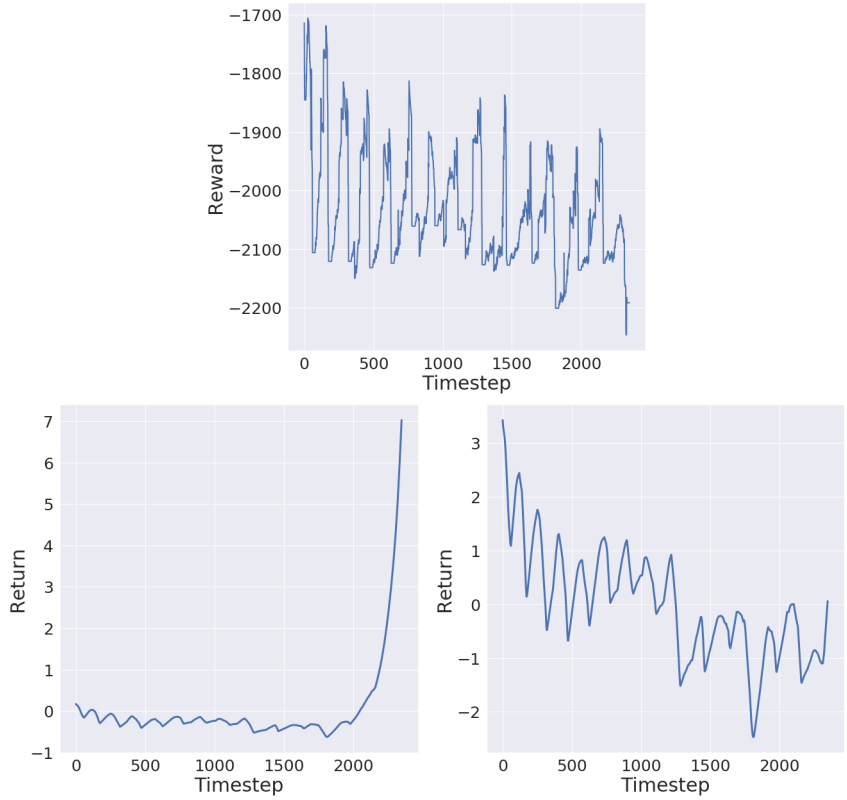


Figure 5.2: Return without (left) and with (right) prior reward (top) standardization.

After obtaining the return, the loss function is computed to update the parameters using the optimizer. As stated in the RL formulation, the gradient is propagated jointly through the two outputs. Hence, the loss function is comprised of both the policy loss and value (or advantage) loss. In practice, an additional entropy term that enhances exploration is also added.

The policy loss is the A2C update term explained in the actor-critic theory, and the value loss is the MSD of the advantage function difference. The entropy loss is computed for the probability distribution of the actions, $\pi(A_t \mid s_t, \theta_\pi)$. This term penalizes unbalanced distributions to avoid always selecting the same actions. A constant β is applied to balance this term that should not dominate over the other losses.

Equation 5.11 shows the total loss term for a timestep. The loss is computed for each timestep and then accumulated. Note that the policy and entropy losses should be maximized, while the value loss should be minimized. The signs of the first two losses are adjusted since DL tools such as PyTorch and TensorFlow are designed to minimize the loss. Finally, the optimizer Adam is utilized to update the parameters using the accumulated mean loss (Equation 5.12).

$$\begin{aligned} \min \mathscr{L}_t = & -log(\pi(a_t \mid s_t, \theta_\pi)) \cdot (R_t - V(s_t)) \\ & - \beta \cdot \pi(A_t \mid s_t, \theta_\pi) \cdot log(\pi(A_t \mid s_t, \theta_\pi)) \\ & + 0.5 \cdot (R_t - V(s_t))^2 \end{aligned} \tag{5.11}$$

$$\Delta\theta_\pi = \alpha \nabla_{\theta_\pi} \bar{\mathscr{L}} \tag{5.12}$$

The process from the parallel simulations to this parameter update constitutes an episode. Episodes are repeated until the convergence criterion is met. This criterion is either achieving the maximum reward over several consecutive episodes or finishing predefined training times to avoid extremely long procedures. Table 5.2 shows a summary of the training hyperparameter. Algorithm 5.1 summarized the training process. Note that our devised model is model-free, on-policy, offline, and an actor-critic within the RL taxonomy presented in Chapter 2.

Evaluation Metrics

The model is evaluated during two stages. First during training, several elements of the procedure are monitored for each timestep and episode. Timestep-wise, the tracked elements are the chosen action's log-probability ($log(\pi(a_t \mid s_t, \theta_\pi))$), the reward ($r_t$), the return ($R_t$), the baseline ($V(s_t)$), the advantage function ($A(s_t, a_t)$), and the actions' probabilty distribution ($\pi(A_t \mid s_t, \theta_\pi)$). These values help to identify issues such as the biased return explained in the training procedure (Figure 5.2). All of them have been discussed already in this report.

Episode-wise, the recorded elements are the total loss, entropy loss, mean reward, and length. The mean reward is computed as the episode's reward divided by the episode's length. The mean is prefered since episodes do not have equal

length. Overall, these features help to assess the model's change of behavior over time and the maximum reward criterion better.

Figure 5.3 and 5.4 depicts all monitored elements (the former also shows the rolling averages). Most of these aspects are often tracked in RL problems.

Once the model training and the evaluation thereof are completed, the model is tested against the baseline SDN controller strategy, the location-aware strategy introduced in Chapter 3 that is already implemented in Mercury. The evaluation metrics are EDCs' power consumption, PUE, and the transmission delay. This analysis is covered in Chapter 6.

Table 5.2: Model's training hyperparameters.

Hyperparameter	Value
Feature scaling	Fixed standardization
Parallel simulations	3
Return γ	0.99
Return & Reward	Standardization
Loss $\mathscr{L}$	$\mathscr{L}_{policy} + \mathscr{L}_{value} + \mathscr{L}_{entropy}$
Entropy β	0.001
Optimizer	Adam
Learning rate	0.001
Batch size	Entire episode
Episodes	Meeting convergence criterion
Convergence criterion	Max reward or train time

Algorithm 5.1 Resource allocation manager's A2C training

Preliminary simulation to obtain feature scaling parameters μ, σ
Initializate DNN's parameters θ_π randomly
Note: DNN outputs are $\pi(A \mid s, \theta_\pi)$ and $V(s)$
while maximum reward or training time not reached **do**
  Launch N parallel simulations for preset time
  Broadcast DNN to parallel simulations
  while parallel simulations running **do**
    for parallel simulation **do**
      Collect states s, actions a, and rewards r trajectory
    end for
  end while
  Initializate loss: $\mathscr{L} \leftarrow 0$
  for trajectory, T **do**
    Standardize rewards, r_T
    for timestep, t **do**
      Compute return: $R_t = \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}$
    end for
    Standardize returns, R_T
    for timestep, t **do**
      Compute loss: $\mathscr{L}_t = \mathscr{L}_{p,t} + \mathscr{L}_{v,t} + \mathscr{L}_{H,t}$
      $\mathscr{L}_{p,t} = -log(\pi(a_t \mid s_t, \theta_\pi)) \cdot (R_t - V(s_t))$ ▷ $a_t \in A_t$
      $\mathscr{L}_{v,t} = +0.5 \cdot (R_t - V(s_t))^2$
      $\mathscr{L}_{H,t} = -\beta \cdot \pi(A_t \mid s_t, \theta_\pi) \cdot log(\pi(A_t \mid s_t, \theta_\pi))$
      $\mathscr{L} \leftarrow \mathscr{L} + \mathscr{L}_t$
    end for
  end for
  Update DNN's parameters: $\theta_\pi \leftarrow \theta_\pi + \alpha \nabla_{\theta_\pi} \bar{\mathscr{L}}$
end while

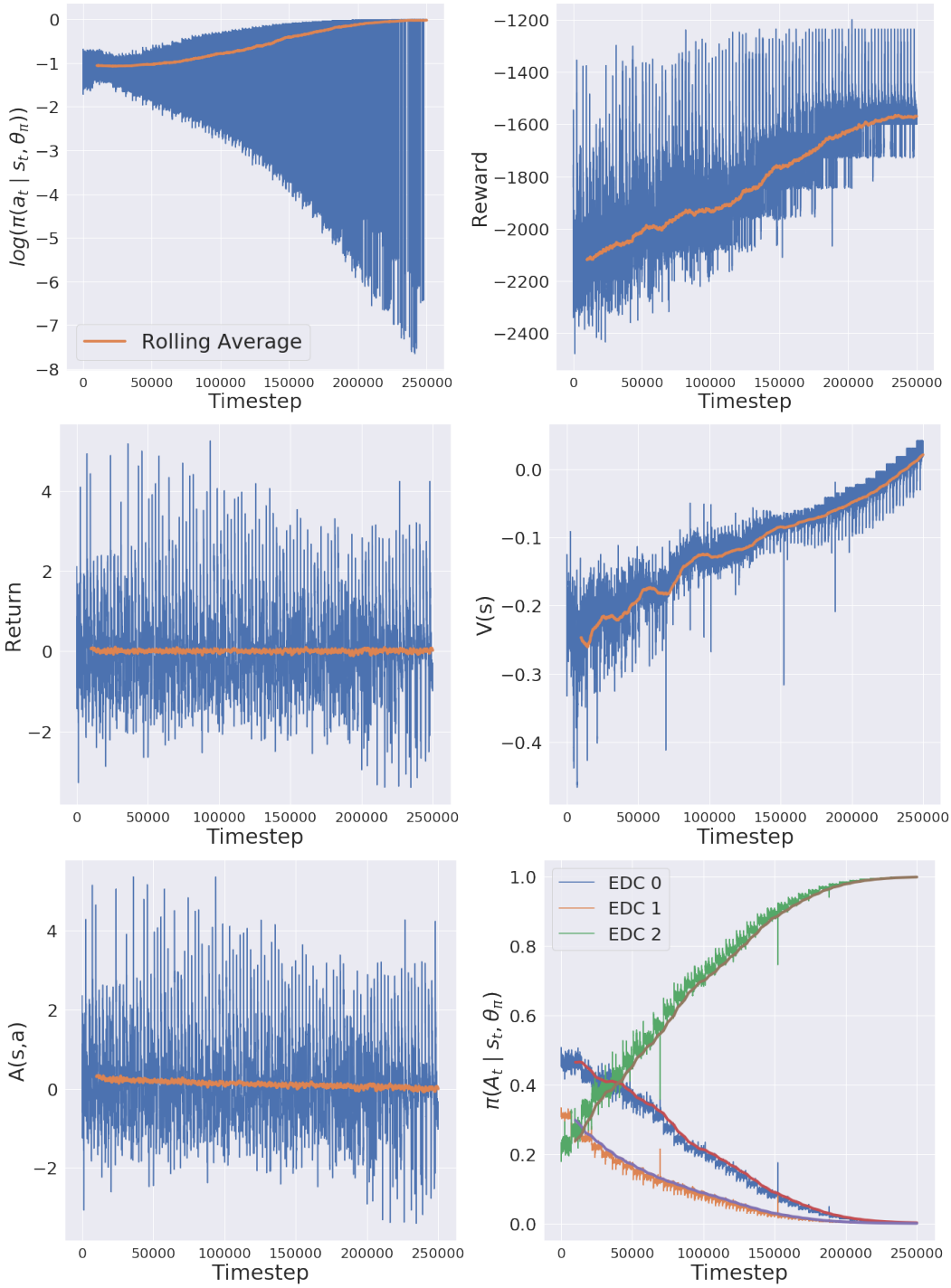


Figure 5.3: Evaluation metrics with rolling averages during model training.

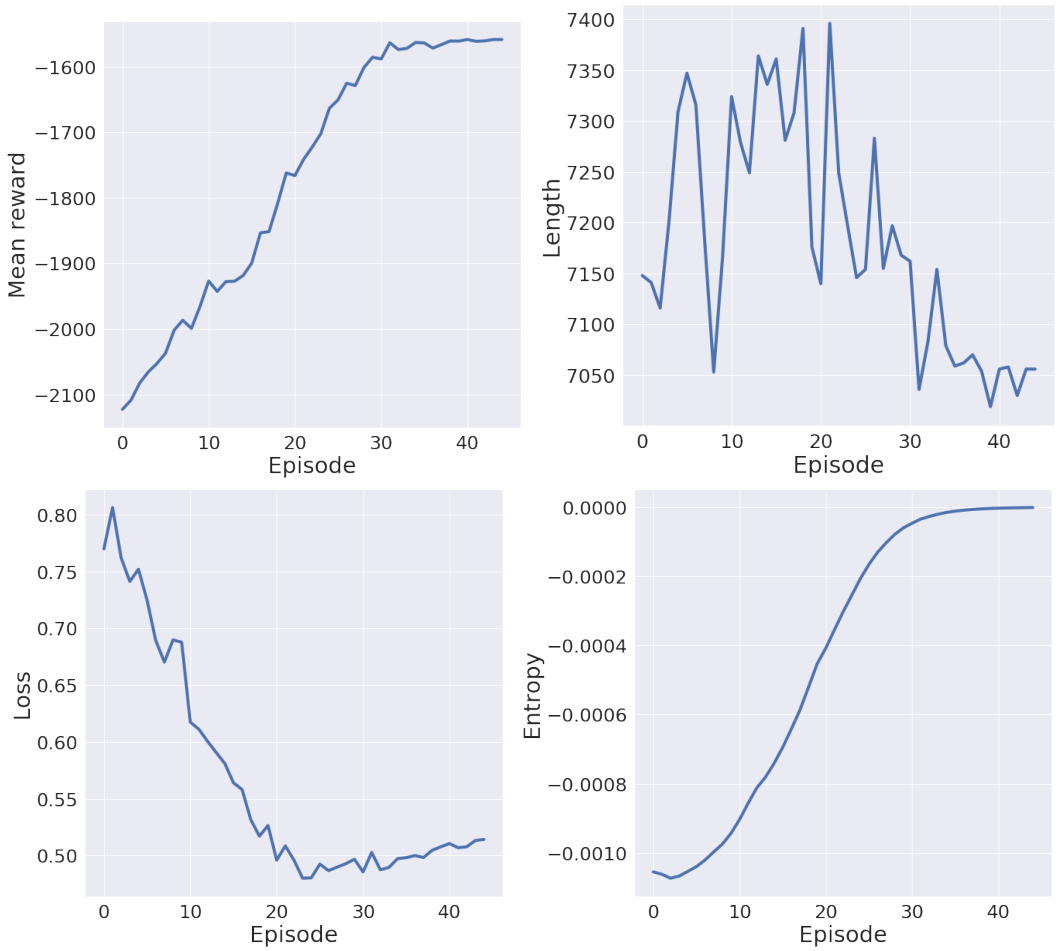


Figure 5.4: Evaluation metrics for trained models.

5.2 Edge Computing Scenario Configuration

So far, this MSc thesis has developed models for the EC elements that comprise the devised ADAS application. In this Section, the configuration of each model is presented. Three scenarios are proposed for their optimization. First, an air-based scenario using its corresponding EDC model. Second, a immersion-based scenario in the same fashion as the previous one. Finally, third, a heterogeneous scenario in which the two types of EDC share the stage. Before introducing them, some general considerations that apply to all three cases are given. The objective is to configure the simulations in the most realistic way possible.

5.2.1 General Considerations

In the same way, as in the actor-critic implementation, this Subsection is divided into several parts to facilitate understanding. The three devised scenarios share the aspects discussed below.

Mercury's Version, Elements, and Simulation Time

This work employs the *lite* version of Mercury. This variant ignores the transmission-delay-related functionalities during simulations. Since delays are only analyzed after training, these features can be turned off during the learning part. In our experiments, this version proved to be more than ten times faster than the *classic* version. It has been a vital factor as it contributed to a massive reduction in training time.

There are three types of elements arranged in the scenario, EDCs, APs, and UE. The chosen number for the former two has already been discussed. There are three EDCs and ten APs, as in Figure 3.4. Each EDC's IT equipment includes normally 10 PUs (i.e., GPUs), but the number varies depending on the scenario. On the other hand, the UE is set to 50 vehicles. Although Mercury allows up to 100, the training time overly increases. Hence, the UE was fixed to balance the trade-off between time performance and interestingly enough layouts.

For the same reasons, the simulation time was set to 180 seconds, even though Mercury features up to 600 seconds of simulation. This configuration enables the computation of 30 episodes in 45 minutes on average. Table 5.3 shows the mentioned parameters so far. These numbers have been utilized during most of this work's development, but they might change slightly in some experiments. In the event of these changes, they will be clearly stated.

Table 5.3: Configuration of Mercury's version, elements, and simulation time.

Parameter	Value
Mercury version	*Lite*
Edge Data Centers	3
Processing Units	10 GPUs*
Access Points	11
User Equipment	50 vehicles
Simulation time	180 s

*depending on the scenario.

Resource Allocation in GPUs

As explained in Chapter 4, GPUs can feature up to four (immersion-based) or five (air-based) sessions (i.e., CNNs) in parallel. Therefore, an analysis of how to allocate the sessions is applied. The analysis has already been validated by the research community in previous publications [32], and further tested in Mercury [34].

There are two options. The first option is to allocate new sessions in either the GPU with minimum or maximum utilization. The decision is between spreading the number of sessions evenly across all GPUs or piling them up in the least amount

of GPUs possible. The second option is to keep the unused GPUs switched off or in idle state. In this decision, there is an efficiency trade-off between energy consumption (switched off) and delay (idle state). The analysis of these strategies in both works [32, 34] concludes that the most energy-efficient configuration is to dispatch new sessions to the GPU with the highest utilization and keep the unused turned off. Finally, Mercury fixed the clock frequencies of the GPUs during the entire simulation. Both the principal clock and memory clock are set to their maximum frequency as these PU units offer better performance using these values. Table 5.4 shows the configuration.

Table 5.4: Configuration of GPU resource allocation.

Parameter	Value
Dispatch strategy	Maximum
Unused GPUs	Powered off
GPU main clock	Maximum*
GPU memory clock	Maximum*

*exact frequency depends on GPU model.

ADAS Session

The devised ADAS application needs some configuration to work as a session in Mercury. The most critical parameter is the utilization factor of each session. As previously mentioned, a session is equivalent to a CNN model. Mercury defines utilization using percentage. Thus this value is set to 25% (for scenarios with four parallel sessions in a GPU) and 20% (for scenarios with five parallel sessions).

Since GPUs' clocks and utilization are set to a fixed value, in this case the maximum, during all experiments, the session's processing time is the same for every scenario. The optimization of this parameter will be covered in future work. Table 5.5 lists the configuration of this parameter and others related to delays for reproducing the results in Chapter 6.

Table 5.5: Configuration of the ADAS session.

Parameter	Value
Utilization	20% or 25%
Min time open/closed	10 s/0.5 s
Timeout	0.2 s
Window Size	1 s
Header	0
Generation rate	1,000,000 bps
Packaging time	0.5 s

Processing Units

Both PU (i.e., the GPU) models share a collection of parameters. This set is related to processing time-related variables. They are set to values that have been tested in Mercury [4]. There is no further analysis of them as they are fixed in all scenarios. They are mentioned for the sake of reproducing the results. Table 5.6 shows the final values.

Table 5.6: Configuration of Processing Units.

Parameter	Value
t_{on}	1 s
t_{off}	10 s/0.5 s
t_{start}	0.2 s
t_{stop}	1 s
t_{op}	0 s
Max start-stop	1

5.2.2 Air-Based Scenario

As defined in the GPU resource allocation, both GPU's clocks are set to their maximum frequency. In addition to this fact, the utilization factor is set to 20% for the air-based model. Figure 5.5 shows the GPU's estimated power consumption with this configuration. When hosting sessions, the model ranges from 95 W at 20% of utilization (i.e., one session) to 110 W at 100% utilization (i.e., five parallel sessions). This range is narrow, hence it might lead to rather modest energy savings.

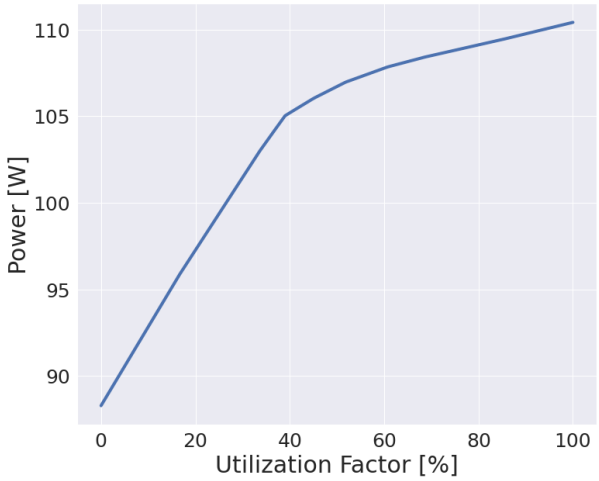


Figure 5.5: Air-based GPU power model with maximum clock frequencies.

The air-based cooling system model only includes one variable, the IT inlet temperature. Typical inlet temperatures may vary between 16°C and 27°C [123].

An evaluation of the PUE as a function of the inlet temperature is developed to set the latter to realistic energy consumption values. Figure 5.6 depicts their relation.

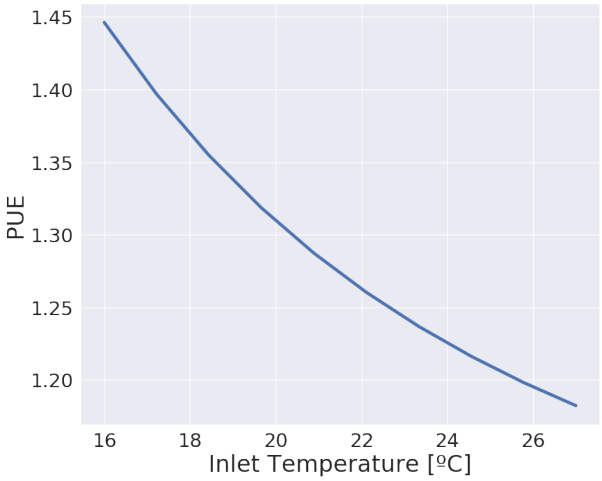


Figure 5.6: PUE subject to IT inlet temperature.

The average PUE in most of today's data centers fluctuates between 1.6 and 2.5, with powerhouses such as Google and Facebook achieving values lesser than 1.1 [124, 125]. Therefore, this air-cooled model is reasonably forgiving. Since setting the model to PUE values of more than 1.45 is unrealistic temperature-wise, the inlet temperature is set to 20°C, a standard value in data centers [123].

5.2.3 Immersion-Based Scenario

The methodology is the same as in the air system. The GPU's clocks are set to the highest frequency, and the utilization factor to 25% since the model can host four parallel sessions. Figure 5.7 depicts the power estimations with the devised PU.

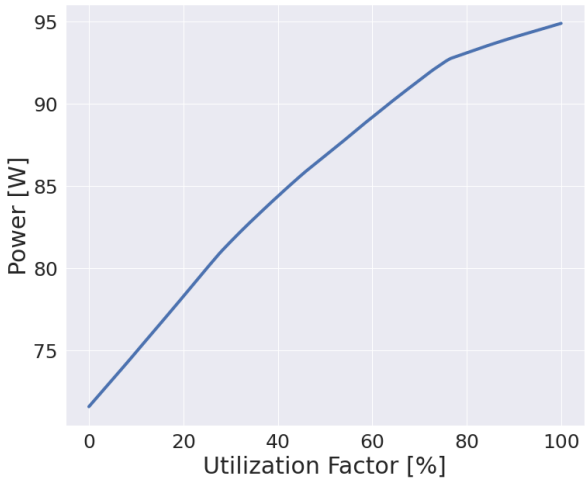


Figure 5.7: Immersion-based GPU power model with maximum clock frequencies.

When hosting sessions, the power estimations range from 80 W to 95 W. Hence, it has a similar behavior as the previous model, albeit more linear. It is worth noting that the difference here is that the immersion model consumes less power. This fact might be related to two design options. First, the GPU's fans are removed to avoid immersion-related problems, hence reducing also the consumed power. Second, the DL backend in this model (TensorFlow [98]) is a far superior software tool than the one used in the previous model (PlaidML [97]).

The two-phase immersion cooling model is solely based on the pump installed in our real-sized prototype. Figure 5.8 shows the two-phase immersion tank. This container has a capacity of 50 kW. Hence, the pump is designed to recirculate water for a system of this magnitude. The designed immersion-based EDC includes around 10 GPU that consumes 95 W at most. A plausible reduction of the cooling power is applied to fit better in our ADAS scenario.

Figure 5.8: Real-sized two-phase immersion cooling tank.

Right now, the considered hardware units to take part in the real-sized prototype's experiments are the two modeled GPUs and another one, the Sapphire Pulse RX 5700 8G GDDR6 [126]. Combining the three GPU models, the maximum average power is roughly 100 W for the tested applications. The following reduction factor is applied to balance the immersion cooling model:

$$\alpha = \frac{GPUs\ per\ EDC \cdot 100\,W}{50,000\,W} \tag{5.13}$$

Aside from the power dimensioning, the immersion cooling model includes one variable, the temperature difference between the input and output temperature (ΔT). As in the previous case, a realistic temperature difference is more suitable

for a 50 kW container instead of our EDC. To work in an optimal operating range (Figure 4.6), this difference needs to be less than 1°C (the exact number depends on the amount of GPUs). Figure 5.9 depicts the results of applying the reduction factor (for 10 GPUs) and a proper temperature difference.

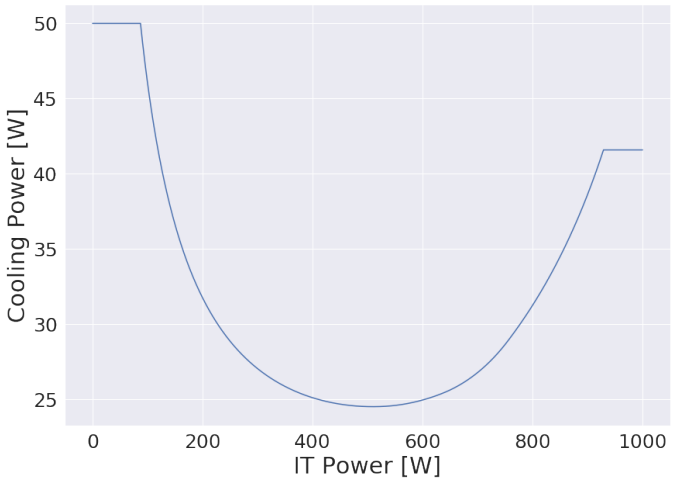


Figure 5.9: Immersion power consumption subject to IT demand

5.2.4 Heterogeneous Scenario

Finally, this MSc thesis proposes a heterogeneous scenario that leverages both types of EDC. Since there is a mismatch between the parallel sessions each EDC's PU can host, the air-based GPU model is reduced to four sessions, the same as in the immersion-based model. In any case, this number was proposed in one of our previous publications [32] as the most efficient number of parallel sessions for the air-based model.

The layout features either two air-based EDCs and one immersion-based EDC or the opposite. It is expected from the previous study on both power consumption models that the DRL agent would prefer to allocate resources int immersion-based EDCs as they consume less energy. The next Chapter delves into the results of each of the three configured scenarios.

Chapter 6

Results

This Chapter covers the results obtained by optimizing the proposed EC scenarios. Section 6.1 presents the air-based scenario results. Section 6.2 and Section 6.3 do the same for the immersion-based and heterogeneous scenarios, respectively. To obtain these results, this MSc thesis has defined the following elements:

- An EC application based on ADAS. It uses CNNs. The application is employed by vehicles in the configured EC scenarios **(Section 3.2)**.
- An air-cooled EDC that includes IT and cooling energy costs. The IT model employs a DNN trained with real traces from the abovementioned application. The air cooling system is borrowed from the Literature [109]. It is devised as a Hot Aisle/Cold Aisle layout **(Section 4.1)**.
- An immersion-cooled EDC that again includes both equipment and cooling. The immersion-based architecture harnesses two-phase immersion cooling systems. A small prototype was developed to extract realistic data traces (Figure 4.4). The IT model employs a DNN. The cooling model utilizes the pump installed in our real-sized prototype **(Section 4.2)**.
- An energy-conscious resource allocation manager using DRL, specifically an A2C model. The manager maps APs to EDCs to strategically allocate the resources needed for hosting ADAS sessions. The model is trained to optimize the power consumption in the EDCs **(Section 5.1)**.
- Different configurations of EC scenarios that feature a single RAN with UE, APs, EDCs, the core network (the resource allocation manager), and the ADAS application using Mercury [4, 34], an EC simulator **(Section 5.2)**.

All experiments were executed using an ASUS TUF Gaming FX505GD laptop with an Intel-Core i7-8750H CPU 2,2 GHz processor and a 16 GB 2667 MHz DDR4 memory. The DRL agent (PyTorch) and the GPU models (TensorFlow) were implemented in the CPU. Most models' training time ranged from 30 minutes to two hours.

To assess the results, the principal evaluation metric is the EDCs' total power consumption, which, in this case, is the sum of IT and cooling consumption (Equation 6.1). Moreover, the PUE is also employed. This metric's purpose is to evaluate the cooling system's efficiency (Equation 6.2). Bear in mind that the PUE does not reflect overall energy optimization, which is what this project seeks primarily. So, for instance, if the IT power consumption is optimally reduced while maintaning the cooling consumption, the PUE would worsen (better values are close to one). In this report, this metric's goal is to assess whether our cooling consumption models behave like the Literature suggests.

Both metrics are often utilized in today's data centers, though with more contributions such as light consumption. These additional contributions represent much less energy than those of IT and cooling, hence they are neglected.

$$P\,[W] = P_{IT} + P_{cool} \tag{6.1}$$

$$PUE = \frac{P_{IT} + P_{cool}}{P_{IT}} \tag{6.2}$$

6.1 Air-Based Scenario

For the air-based scenario, three different configurations of GPUs per EDC were tested to assess how this parameter influences the energy consumed. Table 6.1 shows the mean and peak power consumption, $\bar{P}$ and P_p, and the mean PUE, $P\bar{U}E$, obtained using the location-aware baseline and our energy-aware model. The reduction percentage of each pair is also listed.

Table 6.1: Air-based scenario energy results.

Strategy	$GPUs$[1]	$\bar{P}\,[kW]$[2]	[%]	$P_p\,[kW]$[2]	[%]	$P\bar{U}E$[3]	[%]
Baseline	5	1.60	-14.63	1.71	-15.33	1.31	0.00
Energy		1.37		1.45		1.31	
Baseline	10	1.59	-12.66	1.71	-15.23	1.31	0.00
Energy		1.39		1.44		1.31	
Baseline	15	1.60	-12.66	1.71	-15.23	1.31	0.00
Energy		1.40		1.45		1.31	

The results are somewhat similar for all configurations in both mean and peak consumed power. The mean energy reduction oscillates from 12.66% to 14.63% using our energy-approach, which is a noticeable decrease. On the other hand, the PUE does not change since the air cooling model is linearly proportional to the IT

[1] GPUs per EDC
[2] Mean ($\bar{P}$) and peak (P_p) power consumption
[3] Mean PUE ($P\bar{U}E$)

demand (see Equation 4.4). Figure 6.1 illustrates the power consumption using the best scenario, ten GPUs per EDC, over the simulation's duration. During almost the entire simulation, our energy-aware model consumed less energy than the location-aware baseline. Closer values are found during the simulation's start.

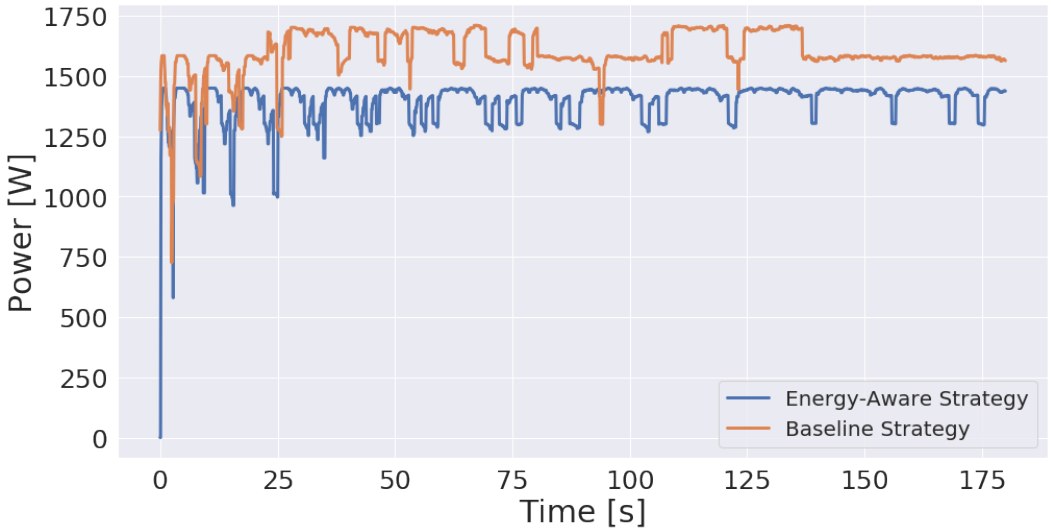


Figure 6.1: Air-based best scenario energy performance comparison.

The transmission delay is also analyzed. Mercury's version *classic* is employed to gather data about delays. Table 6.2 compares the consumed power and transmission delay of both strategies. The mean transmission delays is expressed as $\bar{\tau}$. There is a clear increment in time utilizing our model, roughly one-tenth of a second. Figure 6.2 displays the delay over time. It reconfirms that, for most of the simulation, our model produces worse delay values.

Table 6.2: Air-based scenario delay results.

Strategy	$GPUs$	$\bar{P}\,[kW]$	[%]	$P_p\,[kW]$	[%]	$\bar{\tau}[s]$[4]	[%]
Baseline	10	1.59	-12.66	1.71	-15.23	0.23	+39.14
Energy		1.39		1.44		0.32	

An option to alleviate this problem is to turn on some of the available GPUs. This mode increases power consumption as GPUs are consuming while waiting for new sessions. However, these units are ready to process data, hence a quicker response. Table 6.3 shows the energy-related results for the same three GPUs per EDC configuration using two GPUs in this mode called hot standby.

These configurations yielded more noticeable differences than the previous ones. The mean consumed power reduction varies from 6.19% to 15.47%. The largest EDC suffers the biggest drop in performance while the others remain similar to the previous values.

[4]Mean transmission delay ($\bar{\tau}$)

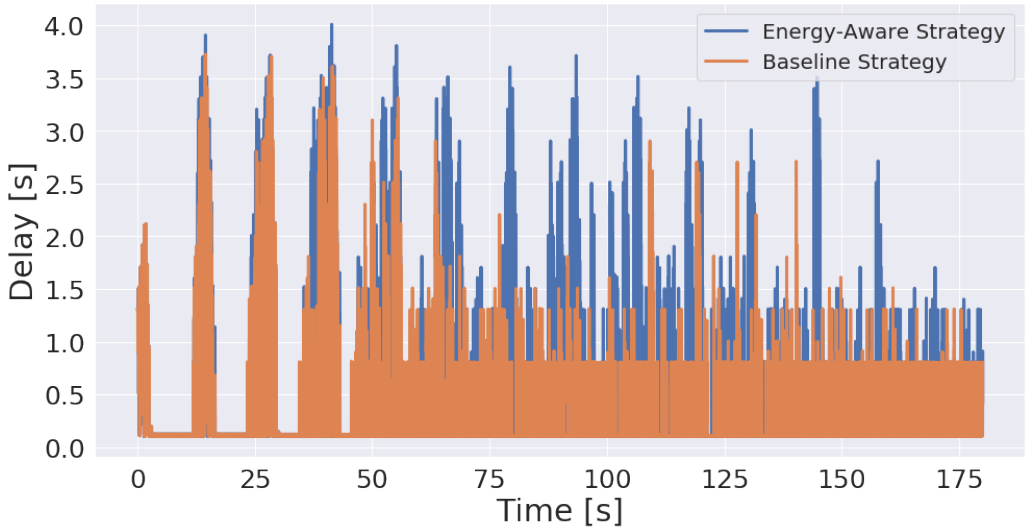


Figure 6.2: Air-based best scenario delay performance comparison.

Table 6.3: Air-based scenario energy results (hot standby).

Strategy	$GPUs$	$\bar{P}\,[kW]$	[%]	$P_p\,[kW]$	[%]	$P\bar{U}E$	[%]
Baseline	5	1.86	-13.12	1.94	-13.34	1.31	0.00
Energy		1.61		1.68		1.31	
Baseline	10	2.23	-15.47	2.40	-20.41	1.31	0.00
Energy		1.88		1.91		1.31	
Baseline	15	2.22	-6.19	2.40	-10.77	1.31	0.00
Energy		2.09		2.14		1.31	

This fact is related to how the A2C model often improves the allocation of resources. Our algorithm tries to allocate the maximum number of sessions in the same EDC. Thus, the largest EDC hardly leverages the GPUs in hot standby of underutilized EDCs. Figure 6.3 illustrates the scenario with the best results (i.e., ten GPUs per EDC).

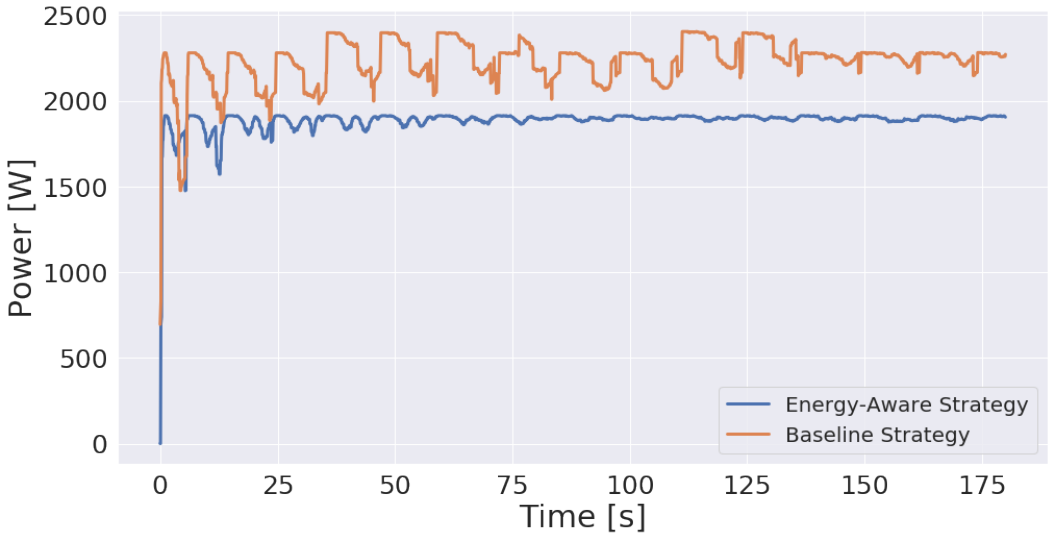


Figure 6.3: Air-based best scenario energy performance comparison (hot standby).

Here, the difference in consumed energy is even more notable than in Figure 6.1. Table 6.4 shows the transmission delay results for the best power consumption scenario with two GPUs in hot standby. The delay reduction compared to the previous case is remarkable. In this scenario, both solutions achieve almost the same perceived delay. However, there are still higher delay peaks in our energy-aware strategy.

Table 6.4: Air-based scenario delay results (hot standby).

Strategy	$GPUs$	$\bar{P}\,[kW]$	[%]	$P_p\,[kW]$	[%]	$\bar{\tau}[s]$	[%]
Baseline	10	2.22	-15.47	2.40	-20.41	0.21	+2.06
Energy		1.88		1.91		0.21	

Figure 6.4 depicts this issue. For almost the whole simulation, both strategies present a similar delay. However, there are still two moments of higher network saturation using our model.

While results are acceptable, QoS criteria should be analyzed in the future along with energy-aware ones. This combination might be achieved using delay-aware reward functions, for instance.

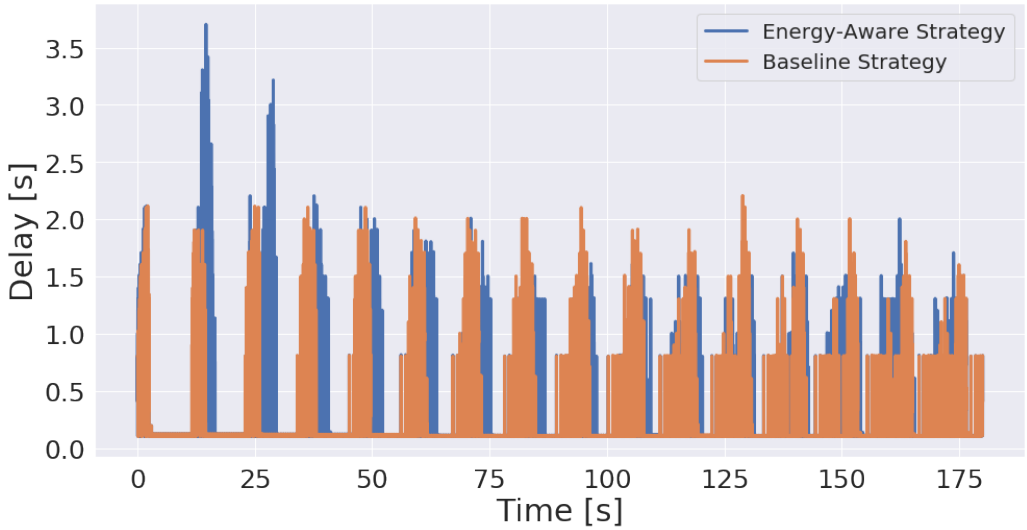


Figure 6.4: Air-based best scenario delay performance comparison (hot standby).

6.2 Immersion-Based Scenario

As in the previous scenario, our model is tested against the baseline using different configurations of GPUs per EDC. The performance metrics are the same as before. Table 6.5 lists the results for the immersion-based configuration without GPUs in hot standby.

Table 6.5: Immersion-based scenario energy results.

Strategy	$GPUs$	$\bar{P}\,[kW]$	[%]	$P_p\,[kW]$	[%]	$P\bar{U}E$	[%]
Baseline	5	1.22	-8.16	1.26	-0.27	1.04	+0.89
Energy		1.12		1.26		1.05	
Baseline	10	1.34	-5.11	1.43	-1.00	1.13	-0.42
Energy		1.27		1.42		1.12	
Baseline	15	1.41	-1.85	1.50	+0.90	1.18	-1.52
Energy		1.38		1.52		1.16	

The improvement obtained with respect to the baseline is slightly less than in the air-cooled scenarios, showing energy savings between 1.85% and 8.16%. It was expected for several reasons. As explained earlier, the immersion-cooled GPU model (Figure 5.7) behaves more linearly than the air-cooled one (Figure 5.5). Another aspect previously discussed is that this GPU model can process one less session (four compared to five in the air-based model). Thus, the EDCs has less capacity, and hence combinations to allocate the sessions.

These two facts decrease potential optimization savings. On the other hand, energy reduction decreases as the GPU per EDC increases. This behavior can be attributed to the increased cooling energy consumption illustrated in the mean PUE.

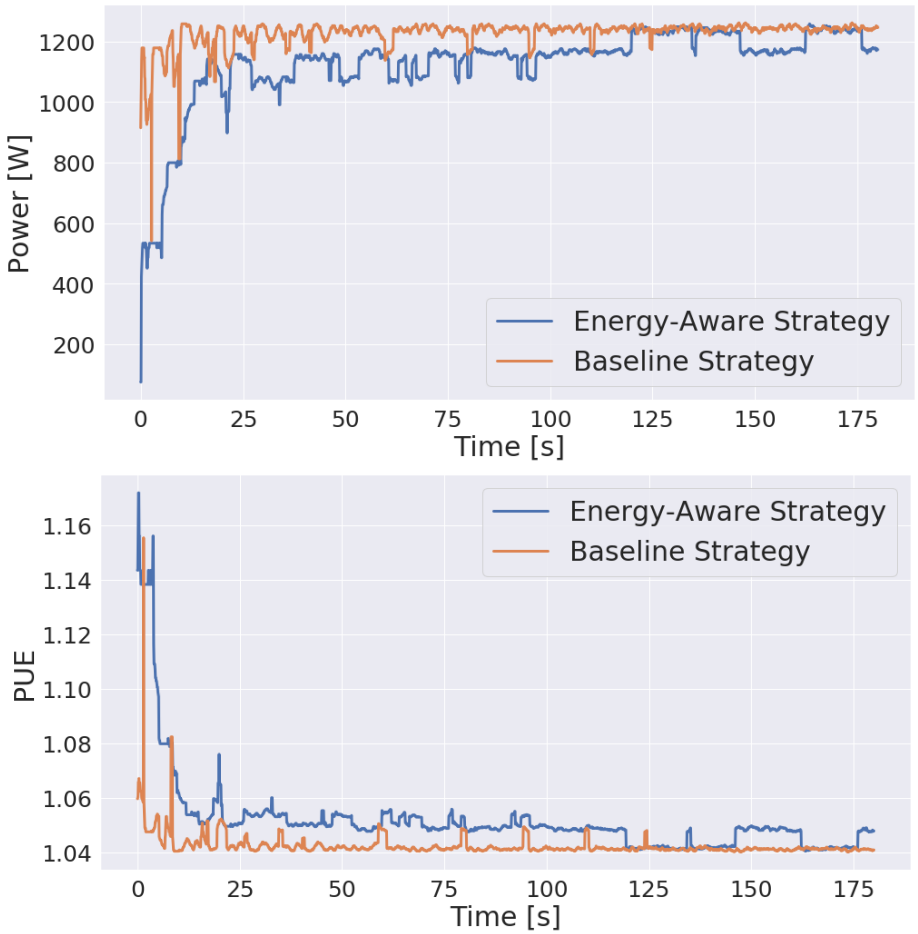


Figure 6.5: Immersion-based best scenario performance comparison.

Figure 6.5 depicts the performance comparison, using the power consumption and PUE, of the model with the highest savings from the last Table. Looking at the power consumption, it is clear that during most of the simulation, the difference is narrower than in the air-based scenario. Our resource allocation manager obtains better results at the beginning when the utilization is lower and therefore has more space to manage the sessions. Then, the PUE values are rather similar, implying that the cooling power plays a lesser role in the optimization.

As in the previous Section, the mode hot standby is applied as it shows better delay properties. Table 6.6 lists the results. The trend is akin to the air-based scenario. The scenario with the least amount of GPUs sees its savings drop a bit since having GPUs in hot standby narrows the potential energy savings in small scenarios. The scenario with ten GPUs obtains the best results compared to the baseline as in the air-based scenario.

Finally, the largest scenario shows almost no improvement. The reasons for this minor decrement are the immersion-cooled EDCs (less potential savings) and the hot standby GPUs (large EDCs are likely never to reach their limit, hence they usually have these GPUs available).

Table 6.6: Immersion-based scenario energy results with (hot standby).

Strategy	$GPUs$	$\bar{P}\,[kW]$	[%]	$P_p\,[kW]$	[%]	$P\bar{U}E$	[%]
Baseline	5	1.33	-4.87	1.39	-0.25	1.02	+0.24
Energy		1.26		1.39		1.03	
Baseline	10	1.64	-6.60	1.79	-7.54	1.05	+1.61
Energy		1.53		1.65		1.06	
Baseline	15	1.72	-0.56	1.88	-2.13	1.10	-1.36
Energy		1.72		1.84		1.09	

On the other hand, PUE has decreased overall. This fact is related to an increase in IT consumption (i.e., the GPUs in hot standby) while the cooling consumption remains the same. Figure 6.6 illustrates the results with the highest savings of this group. Here, the performance of the two strategies looks more similar than in any other analyzed scenarios. Nevertheless, our model still shows better behavior.

Besides strategy comparisons, the most notable finding is that immersion-cooled scenarios consume significantly less energy than the air-cooled ones for the same service thanks to both IT and cooling systems being less power-greedy. So far, in the evaluated configurations and strategies, the mean energy reduction between the two types of homogeneous scenarios is roughly 18.33%, with a maximum of 28.5%.

Finally, the cooling model's configuration proved successful as the PUE fluctuates from 1.02 to 1.18, close enough to what the Literature suggests for these systems, and much more efficient than air-cooled ones nowadays (typically between 1.6 and 2.5).

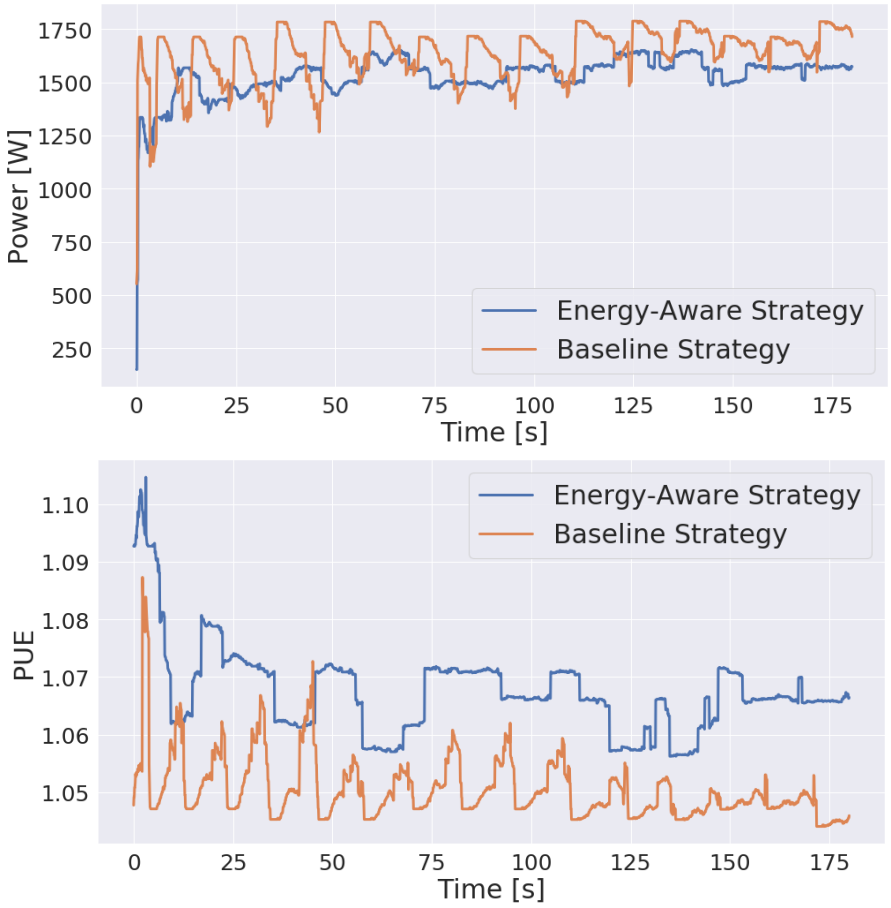


Figure 6.6: Immersion-based best scenario performance comparison (hot standby).

6.3 Heterogeneous Scenario

As stated in Chapter 5, the last optimized scenarios combine the two types of EDCs. Since the layout includes three EDCs, there exist two options to configure it. First, a setup with two air-cooled and one immersion-cooled EDCs. This scenario is referred to as *heterogeneous I* henceforth. Second, a configuration with one air-cooled and two immersion-cooled EDCs. In the same fashion as before, this scenario is called *heterogeneous II*. Table 6.7 illustrates the results for the first case, the heterogeneous I scenario without hot standby. These scenarios are representative of the progressive adoption of two-phase immersion cooling in EC solutions.

Table 6.7: Heterogeneous I scenario energy results.

Strategy	$GPUs$	$\bar{P}\,[kW]$	[%]	$P_p\,[kW]$	[%]	$P\bar{U}E$	[%]
Baseline	5	1.69	-19.22	1.72	-5.28	1.26	-4.86
Energy		1.36		1.63		1.20	
Baseline	10	1.78	-27.42	1.92	-26.78	1.20	-15.01
Energy		1.30		1.40		1.10	
Baseline	15	1.81	-33.63	1.94	-28.97	1.31	-20.51
Energy		1.20		1.38		1.05	

Here, our model demonstrates that it is a far superior strategy energy-wise than the baseline. The best model achieves a remarkable energy reduction of 33.63%. The decisive factor is that the algorithm tries to allocate everything in the immersion-cooled EDC since, as explained earlier, it consumes way less energy than the air-cooled one. It is worth noting that the savings increase with the number of GPUs per EDC. The underlying reason is simple, the immersion-cooled EDC (like the others) has more capacity to host sessions, hence more sessions are stored there, resulting in more savings. This fact is also noted in the PUE, since the lowest value is found with the largest EDC, implying that the immersion-cooled EDC, which has lower PUE values, hosts most of the requests.

Figure 6.7 depicts the simulation of the best scenario, the one with 15 GPUs. Looking at the power consumption (top image), the difference between the two approaches is tremendous. From start to finish, our model beat the baseline by roughly 600 W. The utilization factor (bottom image) reaffirms that these savings are due to the allocation in the immersion-cooled EDC. As in the other scenarios, the hot standby mode is applied since it is helpful delay-wise.

Table 6.8 illustrates the results with the mode hot standby. As in the other scenarios, this mode reduces savings, particularly for the small and large scenarios. The reasons are the same as before. First, the five-GPUs scenario does not have much freedom to optimize since two GPUs are switched on no matter what. Second, the ten-GPUs scenario is again the least affected, and the one with the best results. Third, the fifteen-GPUs scenario cannot fill any EDC completely, hence the hot standby GPUs are always consuming in all three. Nevertheless, the results are again great. The reduction in power consumption goes from 7.67% to 23.75%.

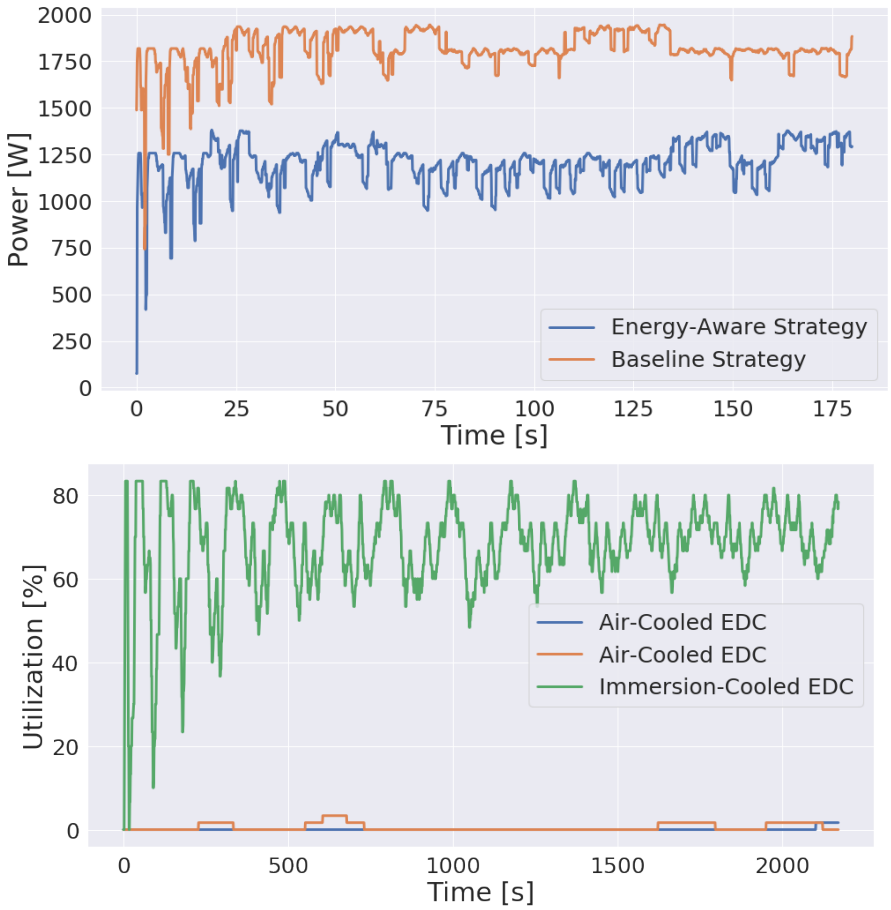


Figure 6.7: Heterogeneous I best scenario performance comparison.

Table 6.8: Heterogeneous I scenario energy results (hot standby).

Strategy	$GPUs$	$\bar{P}\,[kW]$	[%]	$P_p\,[kW]$	[%]	$P\bar{U}E$	[%]
Baseline	5	1.80	-7.67	1.86	-0.69	1.24	-1.84
Energy		1.67		1.85		1.22	
Baseline	10	2.31	-23.75	2.51	-26.30	1.27	-10.26
Energy		1.77		1.85		1.14	
Baseline	15	2.34	-22.61	2.53	-21.89	1.29	-14.93
Energy		1.81		1.98		1.10	

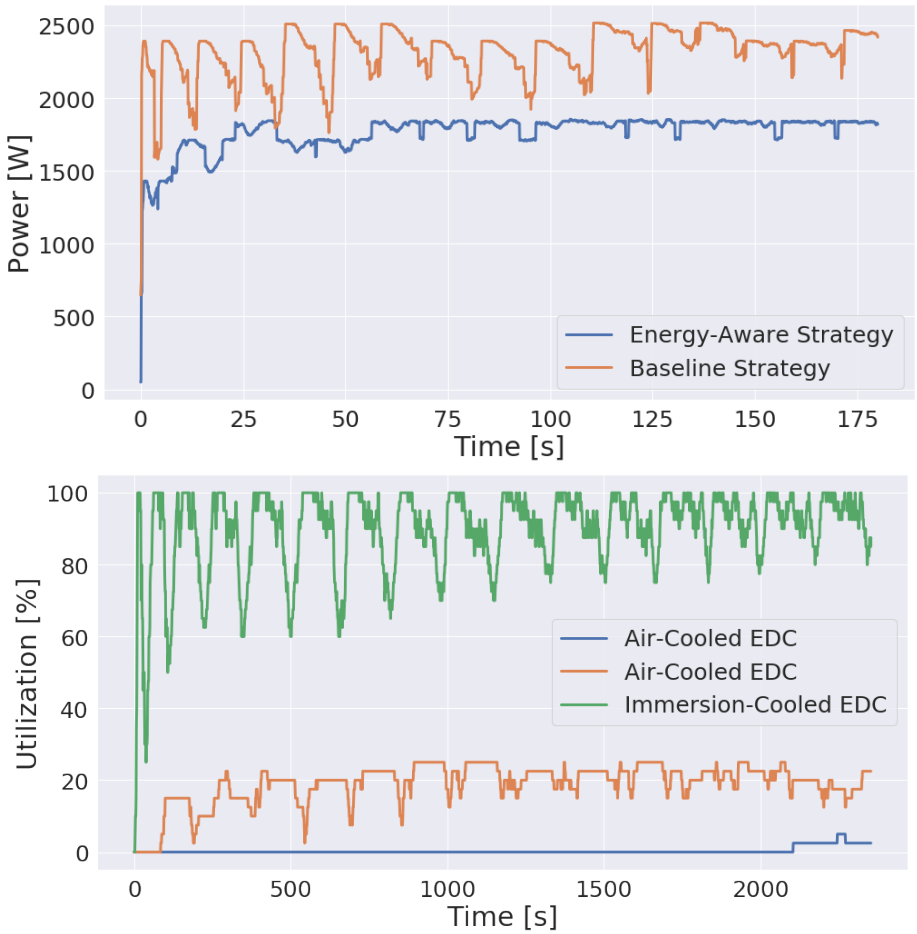


Figure 6.8: Heterogeneous I best scenario performance comparison (hot standby).

Figure 6.8 shows the results for the best scenario. As for the power consumption (top image), the difference is evident, with more than 500 W, on average, between both strategies. Since this scenario includes ten GPUs per EDC, the utilization (bottom image) illustrates how the immersion-cooled EDC fills, and one of the other two take the rest of the sessions. As stated before, the heterogeneous II scenario is also carried out. Table 6.9 lists the results for the scenarios without hot standby GPUs.

Table 6.9: Heterogeneous II scenario energy results.

Strategy	$GPUs$	$\bar{P}\,[kW]$	[%]	$P_p\,[kW]$	[%]	$P\bar{U}E$	[%]
Baseline	5	1.45	-20.26	1.49	-6.78	1.16	-6.39
Energy		1.16		1.39		1.09	
Baseline	10	1.54	-20.49	1.64	-14.35	1.20	-12.84
Energy		1.23		1.41		1.04	
Baseline	15	1.59	-21.00	1.68	-20.88	1.23	-10.26
Energy		1.25		1.33		1.11	

The energy reduction is again remarkable, but do not reach the highest limits of the last one. Instead, the savings are practically the same for all configurations, roughly 20.50%. These somewhat similar results can be explained by comparing them to those in Table 6.7. In that scenario, the options with five and ten GPUs consume more than the one with fifteen GPUs since the latter can allocate every session in the immersion-cooled EDC.

This time, there are two immersion-cooled EDCs, hence the first two configurations can reduce their power consumption. On the other hand, the fifteen-GPUs scenario cannot improve further and only increases its consumption a bit due to the other immersion-cooled EDC's cooling system.

Even though the most affected scenario is the last one, it still is the best of the group. Figure 6.9 depicts the corresponding simulation. As for power consumption (top figure), the drop in performance in contrast to the heterogeneous I scenario (Figure 6.7) is evident. The mean difference drops from 600 W to roughly 350 W, which still is a great outcome. The utilization (bottom image) shows the same behavior as before, further proving that the optimization strategy is not worse, but this configuration offers less potential savings.

As in the rest of scenarios, two GPUs in hot standby are employed. Table 6.10. The effect on the results is much the same. The most benefit configuration is again the one with ten GPUs per EDC, which sees its savings practically unchanged. The other two configurations suffer a decrease in performance proportional to the one in the heterogeneous I scenario (Table 6.8) for the same reasons as before. Figure 6.10 illustrates the performance comparison. The power consumption (top image) and the utilization (bottom image) behave just like its counterpart in Figure 6.8. Note that the air-cooled EDC is hardly used to process sessions. Insofar as these experiments show, all tested hot standby configurations behave similarly in terms of energy.

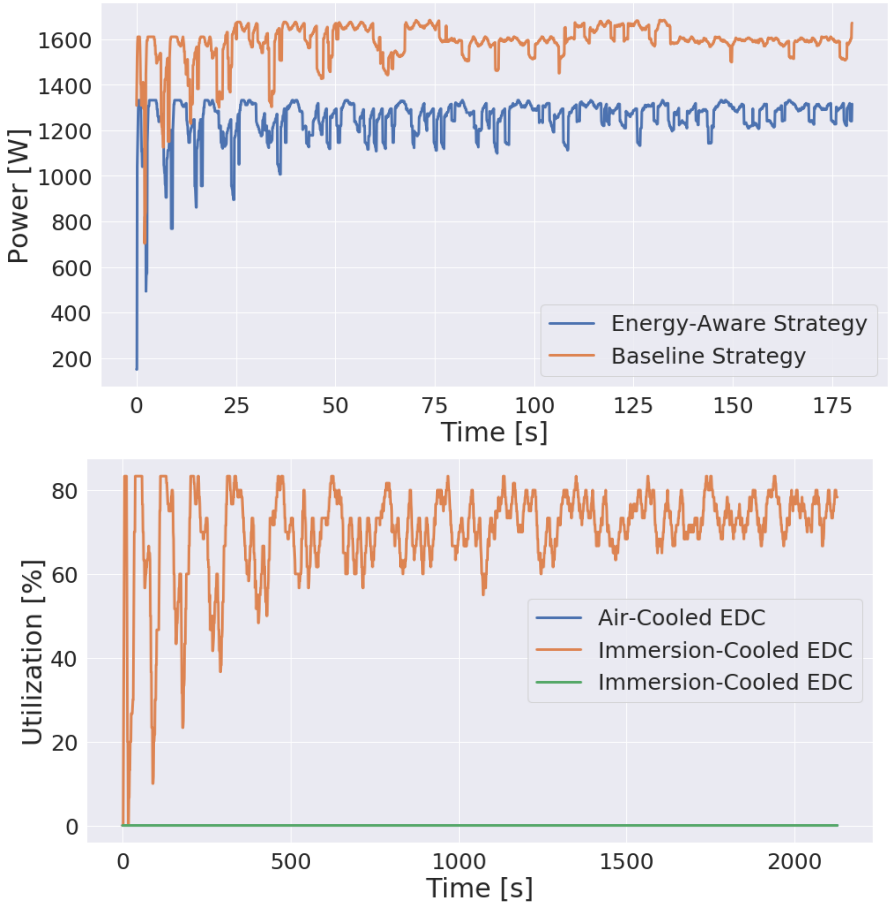


Figure 6.9: Heterogeneous II best scenario performance comparison.

Table 6.10: Heterogeneous II scenario energy results (hot standby).

Strategy	$GPUs$	$\bar{P}\,[kW]$	[%]	$P_p\,[kW]$	[%]	$P\bar{U}E$	[%]
Baseline	5	1.57	-8.93	1.63	-0.71	1.14	-2.58
Energy		1.43		1.62		1.11	
Baseline	10	1.97	-20.22	2.15	-21.84	1.16	-7.70
Energy		1.57		1.68		1.07	
Baseline	15	2.00	-12.34	2.19	-13.40	1.19	-8.31
Energy		1.76		1.89		1.09	

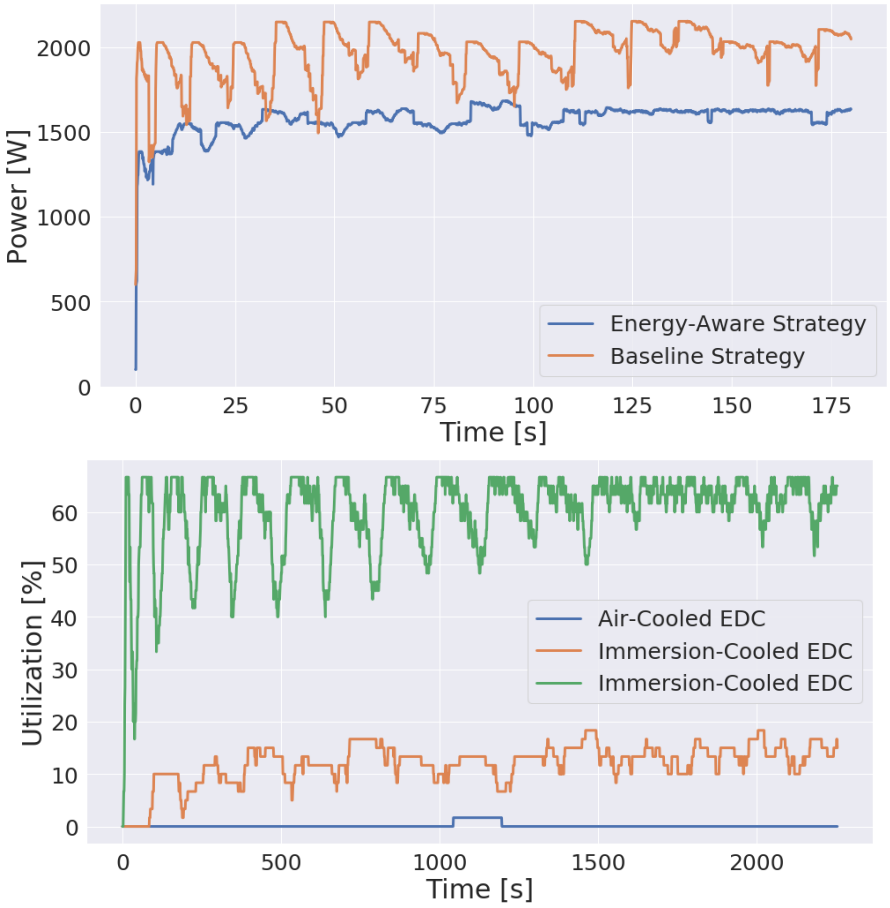


Figure 6.10: Heterogeneous II best scenario performance comparison (hot standby).

From the analysis of the heterogeneous scenarios, some conclusions are drawn. The immersion-cooled EDC stood out as the most efficient energy-wise facility, which yielded the most substantial energy savings. Perhaps, the most signficant finding is that heterogeneous scenarios, which are much more diverse and have higher complexity, bring out the best of sophisticated algorithms as DRL models. The results have been notably better in contrast to more simple scenarios. They pave the way to the formulation of even more complex layouts where DRL-based solutions might make the difference and help obtain considerable energy savings in real EC setups.

Aside from the three types of scenarios, the dimensioning of EDCs through the variation of their resources, in this case, GPUs, proved vital to optimize future deployments. As seen in these results, the medium-sized configurations (ten GPUs per EDC) harness much better the additional hot standby GPUs than the small-sized and large-sized ones (five and fifteen GPUs each).

Chapter 7

Reflection on Research

This Chapter covers reflections on the project and future work that originates from it. This MSc thesis just scratches the surface of the possibilities for optimization in EC scenarios using DRL and Mercury. Section 7.1 dwells on the development and achievements of the project. Section 7.2 presents ideas for future work that stem from this MSc thesis.

7.1 Conclusions

Chapter 1 introduced the motivation for developing this project. The rise of a broad range of new applications based on emerging technologies with truly different requirements makes the current one-dimensional computing approach of CC obsolete, hence a new paradigm that complements it arises, EC. The optimization of EC is crucial to accelerate its deployment and foster a more scalable, sustainable, and greener future. The combination of SOTA cooling systems based on two-phase immersion cooling and predictive resource allocation leveraging the latest advances in AI pave the way for these much-needed improvements to happen. The objectives of the MSc thesis are also outlined.

Chapter 2 delved into our two methods for optimizing EC scenarios. It presented a comprehensive study on current cooling systems in today's data center market. Three principal types of strategies were identified, air-based, water-based, and immersion-based. The former is the most common approach in the field. However, it has several drawbacks. It is the most expensive energy and space-wise, two vital factors in EC. Moreover, the most efficient air-based strategy is location-dependent, limiting its adoption. Water-based systems solve some problems of air-based ones such as corrosion and vibration, but present new critical problems such as leakages. Furthermore, they have lower heat transfer capacity than dielectric-liquid-based strategies. Dielectric-liquid-based solutions have the highest potential, but they have not been adopted so far. In particular, two-phase immersion cooling could eliminate almost all cooling-related energy consumption, which accounts, on average, for 40% of the total.

On the other hand, this Chapter also presented many works found in the Literature that have successfully shown how dynamic resource allocation do improve and optimize the operation of computing resource. The analysis attempts to give a holistic view by analyzing: thermal-aware vs. power-aware, cloud vs. edge, DRL vs. non-DRL, and IT vs. cooling. The Chapter concluded with a brief introduction to a broad topic like DRL, the algorithm employed to optimize the EC scenarios.

Chapter 3 introduced the EC simulator, Mercury, and the devised ADAS application for our EC scenario. Mercury proved to be a great tool to implement all the elements that were designed to make the most realistic simulations possible. The first element devised was the EC application. It is based on ADAS and DL. Our main goal was to found a realistic dataset. The chosen dataset features video footage of drivers to assess whether they are paying attention to the road. This application is a perfect example of a data-intensive service with critical latency and bandwidth constraints that would benefit vastly from EC facilities. The application utilizes CNNs which are SOTA solutions in CV. This Chapter also presented the baseline for our energy-aware optimizations, a location-aware strategy that is entirely naive to anything energy-related. Note that until this project, it was the only strategy available in Mercury.

Chapter 4 described the design and implementation of the most fundamental part of the EC scenario, the EDCs. Two types were proposed, one air-cooled and another immersion-cooled. Real hardware and prototypes were utilized, and developed if needed, to gather data and make the most realistic power consumption models possible. Both EDCs model a GPU, which are the most suited units to run the CNN-based application. It utilizes an FNN to estimate the power consumed by the GPU subject to the ADASapplication, GPU's clocks, and utilization. The former is measured as the number of parallel sessions (i.e., CNNs) in the GPU. Some problems arose at this point. The GPUs were from AMD, which does not support DL applications such as CNNs particularly well. This project ran into many memory-related problems before achieving a feasible GPU model.

Once all those issues were solved, the final results were great. The air-cooled GPU model achieved an NRMSD of 2.45%, and an R^2 of 99.01% in the test set. A paper presenting this model, along with a preliminary study on resource allocation, was validated by the research community [32]. The immersion-cooled also achieved excellent results, an NRMSD of 3.15%, and an R^2 of 97.97% in the test set. Besides the models, the two-phase immersion cooling prototype devised for the experiments was a big success and improvement compared to our previous system (also validated by the research community [29]), thanks to a real heat exchange mechanism that prevented leakages.

Apart from the GPU models that comprise the IT equipment of EDCs, the cooling system was also developed to make these data centers even more realistic. The air-based system was borrowed from another project [109] and tweaked to meet what the Literature showed. After analyzing it, this power consumption model was rather indulgent for how much a conventional air-based data center consumes. It was expected a PUE between 1.6 and 2.5, but it could not reach more than 1.45. As for the immersion-cooled EDC, the cooling model is based on the pump installed in

our real-sized two-phase immersion tank. The cooling consumption was computed as a function of the required flow rate (linearly correlated with the IT power) for a fixed temperature difference inside the tank. Since this pump is devised for a much larger EDC, a reduction factor based on the real-sized tank and our EDC was applied to produce a more realistic two-phase immersion cooling system (i.e., extremely low PUE). The chosen temperature difference made the pump work at its optimal operating range.

Chapter 5 presented the DRL-based and energy-aware resource allocation manager in charge of optimizing the EC scenarios. The chosen DRL algorithm was a synchronously parallelized A2C. The development of a functional model was harder than implementing standard DL solution such as FNNs, CNNs, or RNNs. RL is a vast field with many little details that affect the performance of the model.

The best reward function was the simplest one, which minimized the power demand of the EDC. The standardization of the rewards proved to be essential for the success of the model. The agent's DL architecture, a two-headed FNN, yielded great results from the start and avoided going for more complex networks. Many parameters concerning the training of the model were evaluated to assure that the results are reproducible. Even though DRL is not the simplest and most straightforward solution available, it showed great potential for solving complex environments that will be explored in much detail in the future. After adjusting all aspects of the algorithm and using a CPU-based architecture for running the experiments, models took between 30 minutes and two hours to converge.

Aside from this model, this Chapter also described the configuration of several EC scenarios to be optimized using the elements mentioned above and other parameters that Mercury features in their options. To set them to realistic values, this project utilized parameters already tested and validated [4, 34]. In particular, three types of scenarios were defined: air-based, immersion-based, and heterogeneous (the combination of the other two).

Chapter 6 summarized the most relevant results of the energy-aware optimization of the EC scenarios. As for the air-based scenario, EDCs with different numbers of GPUs were evaluated. On average, the energy reduction oscillated from 6.19% to 15.46% compared to the baseline strategy, being the latter value more common. An important finding was that the baseline model performed better time-wise. The mean transmission delay was 39.14% higher with our solutions. By switching on a couple of GPUs ready to host new sessions, these difference shrank to just 2.06%. However, this procedure did not solve a few high peaks. A future study combining energy and delay aware optimizations to meet QoS criteria should be carried out in the future.

As for the immersion-based scenario, the results compated to the baseline were slightly inferior to the previous scenario. The energy improvement ranged from 0.56% to 8.16%. The reason for these reduced savings is two-fold. First, the immersion-cooled GPU model behaves more linearly than the air-cooled one, hence less room for improvement. Second, the number of parallel sessions in this GPU model is four compared to five in the air-based model. Thus, the EDCs have

lesser resources and combinations to allocate the sessions. Aside from a reduced optimization potential, the immersion-cooled EDCs consumed way less energy since their two parts, IT and cooling, are less power-greedy. On average, this energy reduction was 18.33%, with a maximum of 28.5%. Finally, the cooling model's configuration based on the real-sized pump proved successful since it yielded PUE values close enough, albeit a bit higher, to what the Literature shows (1.02-1.18 compared to that of 1.02-1.03). Furthermore, these values are much better than those of air-cooled data centers nowadays (between 1.6 and 2.5).

As for the heterogeneous scenario, the energy savings were by far the best. Almost all scenarios surpassed the 20% mark, being 33.63% the highest power consumption reduction of this Section, and the entire Chapter. This outcome can be attributed to the increased scenario's complexity that provides much higher optimization potential that a sophisticated DRL-based algorithm can exploit much better.

This fact in addition to the energy efficiency showed by the immersion-cooled EDC is vital for the future of EC. It paves the way to the deployment and optimization of even more intricate scenarios with energy-cost-delay trade-offs and many different applications. The heterogeneous scenarios are the most suitable for a practical implementation since service providers will gradually introduce immersion-cooled solutions that will have to coexist with air-cooled ones. The DRL model's ability to optimize these layouts could enable, or at least ease, the development of Smart Cities with truly different services based on IoT, AI, 5G, to name a few.

This MSc thesis has successfully achieved the goals set at the beginning of the project in Section 1.5:

- **The simulation of realistic EC scenarios**: This part have been achieved with the successful modeling of an EC application, the two types of EDC, the DRL-based resource allocation manager, and the use of Mercury.

- **The modeling of realistic EDCs**: A particular emphasis was put on the EDC modeling. For doing so, real hardware and the development of a two-phase immersion cooling prototype help accomplish this goal.

- **The study of energy optimizations in EC scenarios**: Thanks to DRL, the energy-aware resource allocation manager could beat the baseline strategy energy-wise. Nonetheless, many improvements could be developed in the future for conceiving a more holistic strategy.

Perhaps, the most important conclusion of this MSc thesis is that the combination of DRL optimization and Mercury enables innumerable possibilities to optimize highly complex EC scenarios going forward. This fact would potentially accelerate the deployment of EC solutions and, thus, a more scalable, sustainable, and greener future.

7.2 Future Work

In the last Section of this paper, some possible new steps are discussed for the future of this research. As mentioned earlier, this MSc thesis only scratches the surface of what could be devised utilizing DRL and Mercury. The potential future lines can be divided as follows:

Improving the Current Model

Even though the results were positive, the training time might be reduced by exploring new procedures. For instance, the EDC-AP mapping occurs each timestep. These timesteps are small, so updating every two, five, or ten steps might not change the outcome and reduce the training time since the DRL model interfere less.

On the other hand, the hyperparameters were set mostly using values from other pieces of research. Those values proved prosperous, but hyperparameter tuning techniques such as hyperband or gradient-based could potentially improve the model. As for the RL parameters, the discount factor, γ, could also be tested further. It could even be removed and changed for a moving average. To test the model's robustness, the UE could be randomized for achieving more unique scenarios.

New Reinforcement Learning Models

In this MSc thesis, actor-critic algorithms were tested, in particular, A2C models. In the future, other popular choices from the same family, such as A3C, or a different one, such as DQNs, could potentially improve the results. In the Literature, many projects that involve DRL and EC prefer value-based solutions (mostly DQNs).

New Deep Learning Architectures

The two-headed FNN proved to be sufficient for achieving great results. However, in more complex scenarios, the use of more advanced DL architectures that unlock the true potential of this technology could be tested. For instance, to model time series problems like this one, RNN architectures based on Long Short-Term Memory (LSTM) or one-dimensional CNNs could vastly improve the results.

Multi-Objective Optimization

The optimization of energy consumption also revealed other underlying and unavoidable problems, such as transmission delays. In the future, one of the top priorities should be to optimize several variables to offer the most holistic solution possible.

These variables could be time-related such as the abovementioned transmission delays, utilization-related to avoid saturation in edge facilities, and cost-related to optimize energy prices for the service providers and end-users.

Smart-Grid and Economic Viability

Concerning the previous aspect, the EC scenarios could feature economic costs to make them more realistic. Another potential improvement is the optimization of energy sources to introduce a power-grid variable to the mix. It could be done by modeling renewable energy sources, for instance. As mentioned in Chapter 1, efficient management of the power grid is essential for the deployment of EC solutions.

Another aspect to be improved related to the scenario definition is the physical footprint of the different cooling solutions. As stated earlier, two-phase immersion cooling reduces by a factor of ten this footprint compared to conventional air-based solutions. If this fact is well reflected in the scenario, two-phase immersion solutions' true potential could be addressed better.

New Edge Computing Applications

So far, in our research, there is only one type of EC application. As the results showed, the more heterogeneous the scenario is, the more could be optimized. Adding new services based on entirely different applications would make the scenarios more realistic and legitimate the use of sophisticated technologies such as DRL.

An essential aspect of the GPUs, their time performance, was not modeled in this MSc thesis. Although it has been tested in one of our previous publications [32], the introduction of the time spent computing a session certainly improves the power consumption modeling of the EDCs.

Thermal-Aware Edge Data Center Modeling

As described in Chapter 2, the temperature in two-phase immersion cooling is vital for efficient heat removal. The temperature is closely related to the power consumed in the PUs and hence in the user's demand. By successfully developing models that capture this behavior, the potential energy savings and efficiency would be unmatched in contrast to any other commercial solution.

New IT equipment and Cooling Systems

The EDC models are mostly based on data extracted from small prototypes or theory. Once our real-sized two-phase immersion tank is fully operational and ready

to use, the models trained with data from the tank will behave as truly realistic edge facilities.

EC does not necessarily need GPUs to be useful. Other types of PUs, such as cloud servers that fit better in many other applications than GPUs, could be modeled. These models would make the scenario even more diverse.

New Baselines

The current baseline is a location-aware strategy that is focused on improving the transmission delay. Other baselines based on new families of algorithms, such as dynamic programming or evolutionary methods, could prove DRL as the best possible solution for this research.

Bibliography

[1] L. Castellazzi, A. Maria, and P. Bertoldi, "Trends in data centre energy consumption under the European Code of Conduct for data centre energy efficiency," tech. rep., European Commission, 12 2017.

[2] 3M, "Data driven, advancing a sustainable future." https://multimedia.3m.com/mws/media/1525287O/data-driven-immersion-cooling-infographic-high-res-pdf.pdf. [Online; accessed 21-May-2020].

[3] R. Evans and J. Gao, "Deepmind ai reduces energy used for cooling google data centers by 40%." https://blog.google/topics/environment/deepmind-ai-reduces-energy-used-for/. [Online; accessed 20-May-2020].

[4] R. Cárdenas, P. Arroba, R. Blanco, P. Malagón, J. L. Risco-Martín, and J. Moya, "Mercury: a modeling, simulation, and optimization framework for data stream-oriented iot applications," Simulation Modelling Practice and Theory, vol. 101, p. 102037, 12 2019.

[5] R. Sánchez-Corcuera, A. Nuñez-Marcos, J. Sesma-Solance, A. Bilbao-Jayo, R. Mulero, U. Zulaika, G. Azkune, and A. Almeida, "Smart cities survey: Technologies, application domains and challenges for the cities of the future," International Journal of Distributed Sensor Networks, vol. 15, no. 6, p. 1550147719853984, 2019.

[6] ARM, "The IoT Business Index 2020," tech. rep., ARM Inc., 2020.

[7] "Gartner Says 5.8 Billion Enterprise and Automotive IoT Endpoints Will Be in Use in 2020." https://www.gartner.com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io. [Online; accessed 18-May-2020].

[8] F. Dahlqvist, M. Patel, A. Rajko, and J. Shulman, "Growing opportunities in the Internet of Things," tech. rep., McKinsey, 2019.

[9] M. Patel, J. Shangkuan, and C. Thomas, "What's new with the Internet of Things?," tech. rep., McKinsey, 2017.

[10] Insider-Intelligence, "The security and privacy issues that come with the Internet of Things," tech. rep., Business Insider, 2020.

[11] "Surgeon performs world's first remote operation using '5g surgery' on animal in china." https://www.independent.co.uk/life-style/gadgets-and-tech/news/5g-surgery-china-robotic-operation-a8732861.html. [Online; accessed 19-May-2020].

[12] "5G Healthcare." https://www.ericsson.com/en/networks/trending/insights-and-reports/5g-healthcare. [Online; accessed 19-May-2020].

[13] NVIDIA, "Self-driving safety report," tech. rep., NVIDIA Corp, 2018.

[14] W. Jonshon, K. Sparks, B. Daly, R. Gyurek, K. Balachandran, J. Barnhill, L. Merrill, B. Markwalter, A. Drobot, J. Foerster, and D. Hatfield, "5G Edge Computing Whitepaper," tech. rep., FCC, 2018.

[15] "What Edge Computing Means for Infrastructure and Operations Leaders." https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders/. [Online; accessed 19-May-2020].

[16] Alibaba, "What is edge computing?." https://www.alibabacloud.com/knowledge/what-is-edge-computing. [Online; accessed 18-May-2020].

[17] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog Computing: Platform and Applications," in 2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb), pp. 73–78, 2015.

[18] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," MobiSys 2014 - Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services, 06 2014.

[19] R. Buyya and S. N. Srirama, Internet of Things (IoT) and New Computing Paradigms, pp. 1–23. 2019.

[20] N. Jones, "How to stop data centres from gobbling up the world's electricity," Nature, vol. 561, no. 7722, pp. 163–167, 2018.

[21] Storm-Report, "Electricity prices in europe – who pays the most?." https://strom-report.de/electricity-prices-europe/. [Online; accessed 18-May-2020].

[22] P. Jones, "Overheating brings down Microsoft data center." https://www.datacenterdynamics.com/en/news/overheating-brings-down-microsoft-data-center/. [Online; accessed 20-May-2020].

[23] P. A. García, Proactive Power and Thermal Aware Optimizations for Energy-Efficient Cloud Computing. PhD thesis, Universidad Politécnica de Madrid, 2017.

[24] D. E. Kirk, Optimal control theory: an introduction. Courier Corporation, 2004.

[25] L. Tomás and J. Tordsson, "An Autonomic Approach to Risk-Aware Data Center Overbooking," IEEE Transactions on Cloud Computing, vol. 2, no. 3, pp. 292–305, 2014.

[26] N. Lazic, C. Boutilier, T. Lu, E. Wong, B. Roy, M. Ryu, and G. Imwalle, "Data center cooling using model-predictive control," in Advances in Neural Information Processing Systems, pp. 3814–3823, 2018.

[27] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," Information Fusion, vol. 42, pp. 146 – 157, 2018.

[28] GIGABYTE, "Two-Phase Liquid Immersion Cooling." `https://www.gigabyte.com/Solutions/Cooling/immersion-cooling`. [Online; accessed 20-May-2020].

[29] J. Pérez, S. Pérez, J. Moya, and P. Arroba, Thermal Prediction for Immersion Cooling Data Centers Based on Recurrent Neural Networks: 19th International Conference, Madrid, Spain, November 21–23, 2018, Proceedings, Part I, pp. 491–498. 11 2018.

[30] Rasberry, "Raspberry Pi Official Webpage." `https://www.raspberrypi.org/`. [Online; accessed 20-May-2020].

[31] 3M, "3M Novec 7100 Engineered Fluid." `https://www.3m.com/3M/en_US/company-us/all-3m-products/~/3M-Novec-7100-Engineered-Fluid/?N=5002385+3290667247&rt=rud`. [Online; accessed 20-May-2020].

[32] S. Pérez, J. Pérez, P. Arroba, R. Blanco, J. Ayala, and J. Moya, "Predictive gpu-based adas management in energy-conscious smart cities," 10 2019.

[33] Shappire Technology, "PULSE RX 580 8G G5." `https://www.sapphiretech.com/en/consumer/pulse-rx-580-8g-g5`. [Online; accessed 20-May-2020].

[34] R. Cárdenas, P. Arroba, J. Moya, and J. L. Risco-Martín, "Edge federation simulator for data stream analytics," 11 2019.

[35] Huawei, "FusionModule1000A All-in-One Data Center." `https://e.huawei.com/en/products/network-energy/dc-facilities/ids1000-a`. [Online; accessed 21-May-2020].

[36] Rittal, "Modular Data Centers in Containers." `https://www.rittal.com/it-solutions/en/solution/data-center-container/`. [Online; accessed 21-May-2020].

[37] Energy-Star, "Hot Aisle/Cold Aisle Layout." `https://www.energystar.gov/products/low_carbon_it_campaign/12_ways_save_energy_data_center/hot_aisle_cold_aisle_layout`. [Online; accessed 21-May-2020].

[38] B. Muralidharan, S. K. Shrivastava, M. Ibrahim, S. A. Alkharabsheh, and B. G. Sammakia, "Impact of cold aisle containment on thermal performance of data center," in ASME 2013 International Technical Conference and Exhibition on Packaging and Integration of Electronic and Photonic Microsystems, American Society of Mechanical Engineers Digital Collection, 2013.

[39] Facebook, "Prineville, or data center dashboard: Pue and wue." `https://sustainability.fb.com/efficiency-dashboard/prineville/`. [Online; accessed 21-May-2020].

[40] K. Heinemeier, "Free cooling: At what cost?," ACEEE Summer Study on Energy Efficiency in Buildings., 2014.

[41] Liquid Cool Solutions, "Prineville, or data center dashboard: Pue and wue." `https://www.liquidcoolsolutions.com/is-a-liquid-cooled-data-center-in-your-future/`. [Online; accessed 21-May-2020].

[42] M. Ellsworth, Jr and M. Iyengar, "Energy efficiency analyses and comparison of air and water cooled high performance servers," 01 2009.

[43] S. Alkharabsheh, B. Sammakia, S. Shrivastava, and R. Schmidt, "Dynamic models for server rack and crah in a room level cfd model of a data center," in Fourteenth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), pp. 1338–1345, IEEE, 2014.

[44] A. J. Lachapelle and M. T. Phillips, "Datacenter in-row cooling units," Apr. 18 2017. US Patent 9,629,285.

[45] U.S. Department of Energy, "Passive rear door for controlled hot air exhaust." `https://datacenters.lbl.gov/sites/all/files/rdhx-doe-femp.pdf`. [Online; accessed 21-May-2020].

[46] P. Karki, Kailash, S. Novotny, P. Radmehr, Amir, and P. Patankar, Suhas, "Use of passive, rear-door heat exchangers to cool low to moderate heat loads," ASHRAE Transactions, vol. 117, pp. 26–33, 2011.

[47] Data Center Knowledge, "It doesn't take a supercomputer to justify liquid cooling." `https://www.datacenterknowledge.com/archives/2017/05/22/rdhx-systems-help-with-high-density-data-center-cooling`. [Online; accessed 21-May-2020].

[48] GYGABYTE, "Direct-to-chip liquid cooling." `https://www.gigabyte.com/Solutions/Cooling/asetek-liquid-cooling`. [Online; accessed 21-May-2020].

[49] J. Rinke, "Targeted liquid cooling for a system," May 13 2014. US Patent 8,724,322.

[50] Engineered Fluids, "Single-phase, liquid immersion cooling." `https://www.gigabyte.com/Solutions/Cooling/asetek-liquid-cooling`. [Online; accessed 21-May-2020].

[51] D. W. Sundin, "Mineral Oil, White Oil and Synthetic Dielectric Coolants," tech. rep., Engineered Fluids, LLC., 2017.

[52] Electronics Materials Solutions Division, "The next generation of data centers is here," tech. rep., 3M, 2019.

[53] Electronics Materials Solutions Division, "3M Two-Phase Immersion Cooling - High Level Best Practices for System Fabrication," tech. rep., 3M, 2019.

[54] Allied Control, "Datatank container unit 1.4mw." `https://www.allied-control.com/publications/DataTank_Product_Info.pdf`. [Online; accessed 21-May-2020].

[55] Serve the Home, "Wiwynn two-phase immersion cooling system for ocp nodes." `https://www.servethehome.com/wiwynn-two-phase-immersion-cooling-system-for-ocp-nodes/`. [Online; accessed 21-May-2020].

[56] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong, "Enacloud: An energy-saving application live placement approach for cloud computing environments," in 2009 IEEE International Conference on Cloud Computing, pp. 17–24, IEEE, 2009.

[57] L. Liu, H. Wang, X. Liu, X. Jin, W. B. He, Q. B. Wang, and Y. Chen, "Greencloud: a new architecture for green data center," in Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session, pp. 29–38, 2009.

[58] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," Future generation computer systems, vol. 28, no. 5, pp. 755–768, 2012.

[59] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 826–831, IEEE, 2010.

[60] Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach," IEEE Transactions on Parallel and Distributed Systems, vol. 19, no. 11, pp. 1458–1472, 2008.

[61] T. Mukherjee, A. Banerjee, G. Varsamopoulos, S. K. Gupta, and S. Rungta, "Spatio-temporal thermal-aware job scheduling to minimize energy consumption in virtualized heterogeneous data centers," Computer Networks, vol. 53, no. 17, pp. 2888–2904, 2009.

[62] L. Wang, G. Von Laszewski, J. Dayal, X. He, A. J. Younge, and T. R. Furlani, "Towards thermal aware workload scheduling in a data center," in 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks, pp. 116–122, IEEE, 2009.

[63] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah, and C. Hyser, "Renewable and cooling aware workload management for sustainable data centers," in Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems, pp. 175–186, 2012.

[64] F. Ahmad and T. Vijaykumar, "Joint optimization of idle and cooling power in data centers while maintaining response time," ACM Sigplan Notices, vol. 45, no. 3, pp. 243–256, 2010.

[65] T. Chen, X. Wang, and G. B. Giannakis, "Cooling-aware energy and workload management in data centers via stochastic optimization," IEEE Journal of Selected Topics in Signal Processing, vol. 10, no. 2, pp. 402–415, 2015.

[66] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," ETSI white paper, vol. 11, no. 11, pp. 1–16, 2015.

[67] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," IEEE Journal on Selected Areas in Communications, vol. 34, no. 12, pp. 3590–3605, 2016.

[68] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," IEEE Transactions on Wireless Communications, vol. 17, no. 3, pp. 1784–1797, 2018.

[69] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks," IEEE Access, vol. 4, pp. 5896–5907, 2016.

[70] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," IEEE Transactions on Communications, vol. 64, no. 10, pp. 4268–4282, 2016.

[71] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty, and A. Y. Zomaya, "Secure and sustainable load balancing of edge data centers in fog computing," IEEE Communications Magazine, vol. 56, no. 5, pp. 60–65, 2018.

[72] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," computer, vol. 50, no. 10, pp. 58–67, 2017.

[73] J. L. Greathouse and G. H. Loh, "Machine learning for performance and power modeling of heterogeneous systems," in 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 1–6, IEEE, 2018.

[74] R. Ge, R. Vogt, J. Majumder, A. Alam, M. Burtscher, and Z. Zong, "Effects of dynamic voltage and frequency scaling on a k20 gpu," in 2013 42nd International Conference on Parallel Processing, pp. 826–833, IEEE, 2013.

[75] J. Lim, N. B. Lakshminarayana, H. Kim, W. Song, S. Yalamanchili, and W. Sung, "Power modeling for gpu architectures using mcpat," ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 19, no. 3, pp. 1–24, 2014.

[76] H. Nagasaka, N. Maruyama, A. Nukada, T. Endo, and S. Matsuoka, "Statistical power modeling of gpu kernels using performance counters," in International conference on green computing, pp. 115–122, IEEE, 2010.

[77] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., "Mastering the game of go without human knowledge," Nature, vol. 550, no. 7676, pp. 354–359, 2017.

[78] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al., "Grandmaster level in starcraft ii using multi-agent reinforcement learning," Nature, vol. 575, no. 7782, pp. 350–354, 2019.

[79] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al., "Solving rubik's cube with a robot hand," arXiv preprint arXiv:1910.07113, 2019.

[80] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press, 2018.

[81] UC Berkeley, "Deep reinforcement learning." `http://rail.eecs.berkeley.edu/deeprlcourse/`. [Online; accessed 25-May-2020].

[82] O. AI, "Welcome to spinning up in deep rl!." `https://spinningup.openai.com/en/latest/`. [Online; accessed 25-May-2020].

[83] I. Galatzer-Levy, K. Ruggles, and Z. Chen, "Data science in the research domain criteria era: Relevance of machine learning to the study of stress pathology, recovery, and resilience," Chronic Stress, vol. 2, p. 247054701774755, 01 2018.

[84] Z. Salloum, "Top down view at reinforcement learning." `https://towardsdatascience.com/top-down-view-at-reinforcement-learning-f4a8b35ebf9a`. [Online; accessed 26-May-2020].

[85] S. Green, C. M. Vineyard, and a C. K. Koc, "Impacts of mathematical optimizations on reinforcement learning policy performance," in 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8, 2018.

[86] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," IEEE Transactions on Cognitive Communications and Networking, vol. 3, no. 3, pp. 361–373, 2017.

[87] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4005–4018, 2019.

[88] D. Zeng, L. Gu, S. Pan, J. Cai, and S. Guo, "Resource management at the network edge: A deep reinforcement learning approach," IEEE Network, vol. 33, no. 3, pp. 26–33, 2019.

[89] Z. Ning, P. Dong, X. Wang, J. J. P. C. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," ACM Trans. Intell. Syst. Technol., vol. 10, Oct. 2019.

[90] Amazon, "Adas and autonomous driving." `https://aws.amazon.com/automotive/autonomous-driving/`. [Online; accessed 27-May-2020].

[91] K. Diaz-Chito, A. Hernández-Sabaté, and A. M. López", ""a reduced feature set for driver head pose estimation"," "Applied Soft Computing", vol. "45", pp. "98 – 107", "2016".

[92] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "A parsimonious model of mobile partitioned networks with clustering," in 2009 First International Communication Systems and Networks and Workshops, pp. 1–10, 2009.

[93] Python, "Official webpage." `https://www.python.org/`. [Online; accessed 29-May-2020].

[94] Jupyter, "Official webpage." `https://jupyter.org/`. [Online; accessed 4-Jun-2020].

[95] Visual Studio Code, "Official webpage." `https://code.visualstudio.com/`. [Online; accessed 4-Jun-2020].

[96] ROCm, "Official webpage." `https://rocmdocs.amd.com/en/latest/`. [Online; accessed 29-May-2020].

[97] PlaidML, "Official webpage." `https://www.intel.com/content/www/us/en/artificial-intelligence/plaidml.html`. [Online; accessed 29-May-2020].

[98] TensorFlow, "Official webpage." `https://www.tensorflow.org/`. [Online; accessed 29-May-2020].

[99] Keras, "Official webpage." `https://www.python.org/`. [Online; accessed 29-May-2020].

[100] Collectd, "Official webpage." `https://collectd.org/`. [Online; accessed 29-May-2020].

[101] Apache Kafka, "Official webpage." `https://kafka.apache.org/`. [Online; accessed 29-May-2020].

[102] Apache Cassandra, "Official webpage." `https://cassandra.apache.org/`. [Online; accessed 29-May-2020].

[103] NumPy, "Official webpage." `https://numpy.org/`. [Online; accessed 29-May-2020].

[104] SciPy, "Official webpage." `https://www.scipy.org/`. [Online; accessed 29-May-2020].

[105] Pandas, "Official webpage." `https://pandas.pydata.org/`. [Online; accessed 29-May-2020].

[106] Matplotlib, "Official webpage." `https://matplotlib.org/`. [Online; accessed 29-May-2020].

[107] Seaborn, "Official webpage." `https://seaborn.pydata.org/`. [Online; accessed 29-May-2020].

[108] Plotly, "Official webpage." `https://plotly.com/`. [Online; accessed 4-Jun-2020].

[109] J. D. Moore, J. S. Chase, P. Ranganathan, and R. K. Sharma, "Making scheduling" cool": Temperature-aware workload placement in data centers.," in USENIX annual technical conference, General Track, pp. 61–75, 2005.

[110] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[111] Keras Team, "Cifar10 cnn keras example." `https://github.com/keras-team/keras/blob/master/examples/cifar10_cnn.py`. [Online; accessed 30-May-2020].

[112] Shappire Technology, "PULSE RX 570 8G G5." `https://www.sapphiretech.com/en/consumer/pulse-rx-570-8g-g5`. [Online; accessed 31-May-2020].

[113] Neutrium, "Pump power calculation." `https://neutrium.net/equipment/pump-power-calculation/`. [Online; accessed 1-Jun-2020].

[114] Castle Pumps, "What does a pump curve show?." `https://www.castlepumps.com/info-hub/how-to-read-a-pump-curve/`. [Online; accessed 1-Jun-2020].

[115] Adam-Green, "Heat transfer." `https://adgefficiency.com/energy-basics-q-m-cp-dt/`. [Online; accessed 1-Jun-2020].

[116] The-Engineering-ToolBox,
"Calculate heating systems flow rates." `https://www.engineeringtoolbox.com/water-flow-rates-heating-systems-d_659.html`. [Online; accessed 1-Jun-2020].

[117] PyTorch, "Official webpage." https://pytorch.org/. [Online; accessed 2-Jun-2020].

[118] M. Lapan, Deep Reinforcement Learning Hands-On - Second Edition. Packt Publishing, 2020.

[119] OpenAI, "Openai baselines: Acktr and a2c." https://openai.com/blog/baselines-acktr-a2c/. [Online; accessed 3-Jun-2020].

[120] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., "Mastering the game of go without human knowledge," Nature, vol. 550, no. 7676, pp. 354–359, 2017.

[121] Pytorch Team, "Pytorch examples: A2c for cartpole." https://github.com/pytorch/examples/blob/master/reinforcement_learning/actor_critic.py. [Online; accessed 5-Jun-2020].

[122] C. Yoon, "Understanding actor critic methods and a2c." https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f#a557. [Online; accessed 5-Jun-2020].

[123] W. Torell and K. B. andVictor Avelar, "The Unexpected Impact of Raising Data Center Temperatures," tech. rep., Schneider Electric, 2016.

[124] Andy Lawrence, "Is pue actually going up?." https://journal.uptimeinstitute.com/is-pue-actually-going-up/. [Online; accessed 8-Jun-2020].

[125] Sam Chester, "What is power usage effectiveness?." https://www.colocationamerica.com/blog/what-is-pue. [Online; accessed 8-Jun-2020].

[126] Sapphire Tech, "Pulse rx 5700 8g gddr6." https://www.sapphiretech.com/en/consumer/pulse-radeon-rx-5700-8g-gddr6. [Online; accessed 8-Jun-2020].

Appendices

Appendix A

Ethic, Economic, Social and Environmental Aspects

A.1 Introduction

Emerging technologies such as the Internet of Things (IoT), Artificial Intelligence (AI), and Big Data are reshaping today's world. Smart cities hold the potential to improve society's way of life for the better, but it comes at a cost. The ever-growing demand for these services produces two significant side-effects.

First, the data centers that support this vast network experience sharp increases in energy consumption. As of today, the data center's electricity bill represents 1.3% of the global demand [1]. Moreover, they also constitute 2% of global CO_2 emissions.

Second, the current computing paradigm, Cloud Computing (CC), does not suffice for the full range of applications that are being developed at the moment. Services like autonomous driving or remote surgeries have critical latency and bandwidth constraints that cannot be ignored. CC concentrates its facilities in large and distant areas. Novel approaches that bring resources closer to end-users are needed. Edge Computing (EC) emerges out of this necessity. However, it causes new problems such as space efficiency (resources will be deployed in dense urban areas), and do not avoid older ones such as energy efficiency.

This MSc thesis seeks the optimization of Edge Computing (EC) scenarios that feature Edge Data Centers (EDCs), a reduced version of conventional data centers. Our goal is two-fold. On the one hand, the optimization of cooling systems that, on average, contributes to 40% of the energy consumed in data centers [1]. Our research employs two-phase immersion cooling that potentially reduce cooling costs to a mere 1% and the physical footprint by 10x in contrast to current cooling strategies [2]. On the other hand, the optimization of predictive resource allocation utilizing AI solutions, in particular Deep Reinforcement Learning (DRL). For instance, Google, a main driver of this field, has demonstrated that these methods may reduce by 15% of the global energy demand in their data centers [3].

A.2 Description of Relevant Impacts Related to the Project

The major impacts that involve this MSc thesis are divided as follows:

- **Economic Impact**: The optimization of EDCs employing novel cooling systems and predictive resource allocation represents a vast reduction in energy and space costs. These benefits could easily translate to other conventional data centers. This energy and space decrease has significant impacts on economic expenses. Thus, it could transform the current data center's market forever.

- **Social Impact**: The EC paradigm would enable the development of novel applications that, at this time, are unthinkable due to latency and bandwidth limitations under the current paradigm. For instance, Smart City's applications would impact agriculture, logistics, education, healthcare, waste management, and government [5]. It would change the way society thinks of cities.

- **Environmental Impact**: Perhaps the most direct impact of EC optimization are issues related to the environment. Data centers and, in general, the Information and Communications Technology (ICT) sector will continue to grow with no sign of stopping in the foreseeable future. Today's CO_2 emissions in data centers represent 2% of the global bill. Efficient use of resources is more critical than ever to build a greener future. In our project, the dielectric liquid, Novec 7100, used in our two-phase immersion solution has a quite low Global Warming Potential (GWP) in contrast to other Chlorofluorocarbons (CFC)-based options [31].

- **Legal Ethics and Professional Responsibility**: The EC paradigm is already backed by international standards institutions such as the European Telecommunications Standards Institute (ETSI) that regulate and give recommendations for ethical and responsible use [66]. As for our project, some parts of it have already been validated by the research community [29, 32].

A.3 Detailed Analysis of Some of the Relevant Impacts

There is a short introduction to the economic and social impact of our project involving Smart Cities' growth in Chapter 1 (Section 1.1). There is also a comprehensive study on the environmental impacts, analyzing energy and space costs, of current cooling solutions on the data center's market in Chapter 2 (Section 2.1). This analysis is summarized in Table 2.1. It also explains in detail why two-phase immersion cooling is the most viable system for reducing energy consumption, and hence CO_2 emissions.

A.4 Conclusions

This work proposes the optimization of EC scenarios by employing novel cooling systems and predictive resource allocation. The ICT sector needs to adapt to the growing demand for novel applications that have the potential to change people's way of life for the better, but also present a broad range of requirements. The optimization of EC scenarios accelerates its deployment and the sector's transformation, hence the changes needed for a more scalable, sustainable, and greener future.

Appendix B

Budget

B.1 Budget of Material Execution

Table B.1: Budget of material execution.

PROTOTYPES				
Concept	Cost	Usage (months)	Depreciation	Real Cost
Two-phase immersion container	100.00	4	100%	33.33
Computer valid for container	650.00	4	20%	43.33
Sapphire Pulse Radeon RX 580 (x1)	209.90	4	20%	13.99
Sapphire Pulse Radeon RX 570 (x1)	199,90	4	20%	13.33
3M Novec 7100 (5L)	255.00	4	100%	85.00
Total cost:				188.99
HARDWARE RESOURCES				
Concept	Cost	Usage (months)	Depreciation	Real Cost
ASUS TUF Gaming FX505GD	819.99	4	20%	54.67
Total cost:				54.67
LABOUR COSTS				
Position		Yearly Cost	Usage (months)	Real Cost
Graduate Engineer		20,800	4	6,933.33
Total Cost:				6,933.33
RAW MATERIALS AND CONSUMABLES				
Total Cost:				50.00
TOTAL COST:				**7,226.89**

B.2 General Expenses and Industrial Benefits

Table B.2: General expenses and industrial benefits.

Concept	Cost
Budget of Material Execution (BME)	7,226.89
General expenses (16% of BME)	1156.30
Industrial benefits (6% of BME)	433.61
Subtotal Budget:	**8,816.80**

B.3 Total Budget

Table B.3: Total budget.

Subtotal Budget	8,816.80
VAT (21% of Subtotal Budget)	1,851.53
Total Budget:	**10,668.33€**

Appendix C

WILO IPL Pump Datasheet

The datasheet of the Wilo IPL 50/115-0,75/2 pump installed in the recirculation system of our real-sized two-phase immersion cooling prototype. This documetation is utilized to model the immesion-based EDC's cooling system for the devised EC scenario.

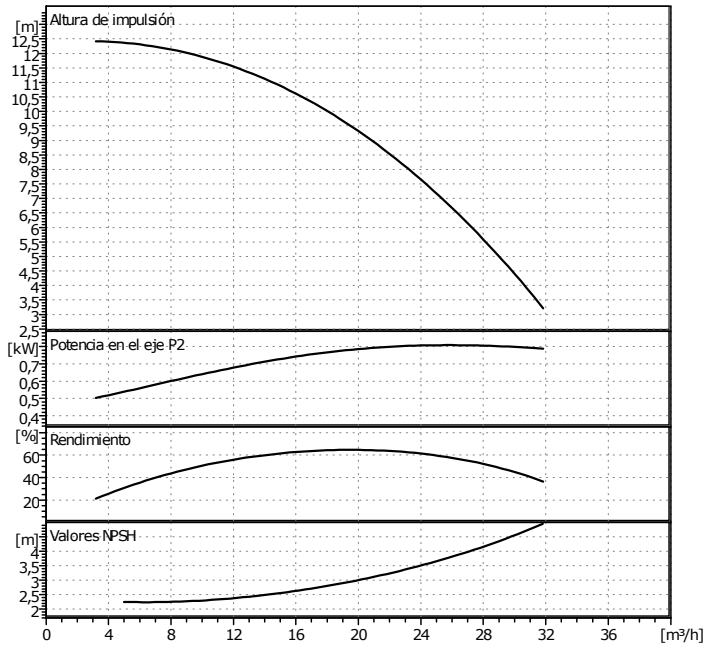
Teléfono
Telefax

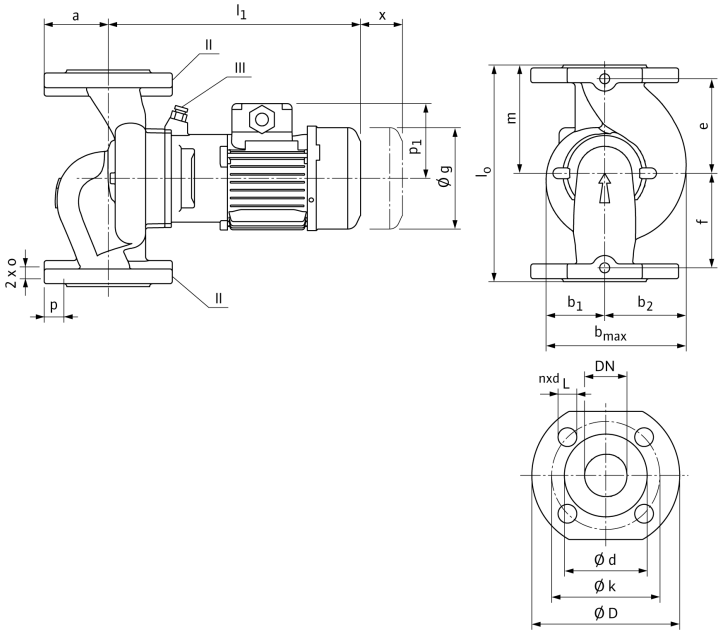
IPL 50/115-0,75/2

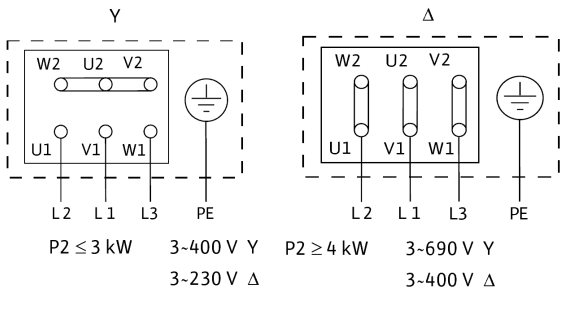
Instalación: Bomba Inline

wilo

Cliente	Projecto
Nº Cliente	Nº proyecto
Contacto	Nº pos.
Elaborado por	Location
	Fecha 18.10.2018







Datos de trabajo teóricos

Caudal	0	m³/h
Altura de impulsión	0	m
Fluido	Agua limpia	
Temperatura fluido	20	°C
Densidad	0,9983	kg/dm³
Viscosidad cinemática	1,005	mm²/s
Presión de vapor	0,02337	bar

Datos bomba

Marca	WILO	
Tipo	IPL 50/115-0,75/2	
Tipo inst.	Bomba simple	
Presión nominal máx.	PN10	
Temp. mín. fluido	-20	°C
Temp. máx. fluido	120	°C
Indice de eficiencia mínima (MEI≥)	=0,10	

Datos hidraúlicos (punto de trabajo)

Caudal		m³/h
Altura de impulsión		m
Potencia en el eje P2		kW
Velocidad	2900	1/min
NPSH		m
Diámetro rodete	103	mm

Materiales

Carcasa	EN-GJL-250
Eje	X 20 Cr 13 (1.4021)
Rodete	Sintético
Cierre mecánico	AQ1EGG (Estándar)
Linterna	EN-GJL-250
Eje partido	X 20 Cr 13 (1.4021)
Eje partido (versión N)	X 2 Cr NiMo 1810

Medidas

mm

a	75	m	140	dL	19		
b1	91	o	M10	n	4		
b2	101	Ø g	146	k	125		
bmax	192	p	20				
e	125	P1	128				
f	125	x	150				
l0	280	d	99				
l1	346	D	165				

Lado aspiración	DN 50	/ PN10
Lado impulsión	DN 50	/ PN10
Peso	27,2	kg

Datos del motor

Pot. nominal P2	0,75	kW
Velocidad nominal	2900	1/min
Tensión nominal	3~400 V, 50 Hz	
Intensidad máx. absorbida	1,7	A
Tipo de protección	IP 55	
Tolerancia tensión		

Referencia de la versión estándar 2089593