Grant Agreement No.: 101070473

Call: HORIZON-CL4-2021-DATA-01

Topic: HORIZON-CL4-2021-DATA-01-05

Type of action: HORIZON-RIA

D6.1 Cost-Effective and Energy-Aware Infrastructure

Revision: v.1.0

Work package	WP 6
Task	Task T6.1, T6.2, T6.3
Due date	30/11/2023

Submission date	21/12/2023
Deliverable lead	TUB
Version	1.0
Authors	Nasir Asadov, Christian Pinto, Lorenzo Moro, Marcello Coppola, Matteo Franzil, Vlad Coroama
Reviewers	Simon Hinterholzer (BOR), José Manuel Bernabé (UMU)

Abstract	This document defines the general architectural framework defined in FLUIDOS for the creation of a carbon-aware and cost-effective infrastructure, spanning across multiple technological domains (e.g., cloud, edge, single devices, embedded devices) and multiple administrative boundaries (e.g., enterprise campus, telco operator, cloud provider).
Keywords	

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	15/09/2023	1st version of the draft	Nasir Asadov (TUB), Matteo Franzil (FBK)
V1.0	21/12/2024	Final version of the draft	Nasir Asadov (TUB), Matteo Franzil (FBK), Vlad Coroama (TUB), Christian Pinto (IBM), Lorenzo Moro (TOPIX), Marcello Coppola (STM)

Disclaimer

Funded by Horizon Europe Framework Programme of the European Union

COPYRIGHT NOTICE

Project co-funded by the European Commission in the Horizon Europe Programme		
Nature of the deliverable:	**R**	
Dissemination Level		
PU	Public, fully open, e.g. web	**X**
CL	Classified, information as referred to in Commission Decision 2015/444/EC	
CO	Confidential to FLUIDOS project and Commission Services	

** R: Document, report (excluding the periodic and final reports)*

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

OTHER: Software, technical diagram, etc.

Funded by Horizon Europe Framework Programme of the European Union

EXECUTIVE SUMMARY

Decentralized computing offers significant benefits such as improved scalability, capacity utilization, fault tolerance, and privacy. Additionally, it can also bring about two energetic and environmental benefits: By its very decentralized nature, and as long as some of the loads can be shifted, the system can take advantage of lower carbon electricity, thus lowering the overall computing-related carbon emissions. Secondly, through a better utilization of available computing resources, it can minimize hardware production, which can save energy, materials, emissions, and costs.

This FLUIDOS deliverable thus focuses on developing an energy- and carbon-aware computing model. This model aims to enhance energy efficiency and reduce greenhouse gas emissions via two key strategies: 1) shifting computing loads in geographical location and execution time to utilize low-carbon electricity, and 2) reducing the need for new device production through an architecture that maximizes the use of existing devices, thereby being cost-efficient and minimizing the production footprint. To allow for the shifting in time discussed under point 1) above, both future computing loads and future carbon intensities of electricity need to be forecasted; the document addresses these predictions as well.

The document also touches upon the methodologies for assessing the environmental impact of Information and Communication Technology (ICT). Two approaches are highlighted: the first focuses on the operational electricity consumption of ICT equipment, common in computing and electrical engineering studies. The second is the life-cycle assessment (LCA), widely used in environmental sciences, offering a more comprehensive view by including the embodied footprint of devices. LCA is particularly relevant in FLUIDOS for its holistic perspective on environmental impact assessment.

Funded by Horizon Europe
Framework Programme of the European Union

TABLE OF CONTENTS

Funded by Horizon Europe Framework Programme of the European Union

Funded by Horizon Europe
Framework Programme of the European Union

LIST OF FIGURES

Funded by Horizon Europe Framework Programme of the European Union

LIST OF TABLES

Funded by Horizon Europe Framework Programme of the European Union

ABBREVIATIONS

LCA	Life Cycle Assessment
ICT	Information and Communication Technology
WP	Work Package
DC	Data Center
AI	Artificial Intelligence
ML	Machine Learning
CI	Continuous Integration
CD	Continuous Development
FaaS	Function as a Service
API	Application Programming Interface
CPU	Central Processing Unit
GPU	Graphics Processing Unit
SLO	Service Layer Objective
SLA	Service Layer Agreement
VCC	Virtual Capacity Curve
MOER	Marginal Operational Emission Rate
IPCC	International Panel on Climate Change
CRD	Custom Resource Definition
LCI	Life Cycle Inventory
LCIA	Life Cycle Impact Assessment
FU	Functional Unit
GWP	Global Warming Potential
GHG	GreenHouse Gas
CEMS	Continuous Emissions Monitoring System

Funded by Horizon Europe
Framework Programme of the European Union

MEF	Marginal Emission Factors
SDK	Software Development Kit
RAPL	Running Average Power Limit
SoC	System on a Chip
DRAM	Dynamic Random Access Memory
ACPI	Advanced Configuration and Power Interface
PnP	Plug and Play
APM	Advanced Power Management
IPMI	Intelligent Platform Management Interface
BIOS	Basic Input/Output System
UEFI	Unified Extensible Firmware Interface
OS	Operating System
BMC	Baseboard Management Controller
KEPLER	Kubernetes-based Efficient Power Level Exporter
eBPF	extended Berkeley Packet Filter
SSH	Secure Shell
BM	Bare-Metal
VM	Virtual Machine
QoS	Quality of Service
HTTP	HyperText Transfer Protocol
IoT	Internet of Things
I/O	Input/Output
CAPEX	Capital Expense
OPEX	Operational Expense
PCIe	Peripheral Component Interconnect express
CXL	Compute Express Link

Funded by Horizon Europe Framework Programme of the European Union

PV	Photovoltaics
TLC	Telecommunications
UPS	Uninterrupted Power Supply
CO_2	Carbon dioxide
H_2	Hydrogen
CNN	Convolutional Neural Network

Funded by Horizon Europe Framework Programme of the European Union

1 Intro & Motivation

Decentralized computing presents a range of advantages, including heightened scalability, better capacity utilization, enhanced fault tolerance, and bolstered privacy. Nonetheless, decentralized computing can also be associated with adverse environmental consequences due to either underutilized computing resources and thus wasted idle operational energy and embodied production energy, or due to the usage of high-carbon electricity when elsewhere idle resources could be employed with similar energy consumption but a reduced carbon footprint.

To address these issues, the aim of work package 6 (WP6) is to develop an energy- and carbon-aware computing model as part of FLUIDOS. The innovative framework implemented in FLUIDOS is anticipated to drive energy efficiency and reductions in greenhouse gas emissions through two primary mechanisms:

- The capability to shift loads in space and time to take advantage of low-carbon electricity, but also,
- The avoidance of device production through an architecture that allows for a better utilization rate of available devices and is thus both cost-efficient and minimizes the production footprint embodied into devices.

To achieve these goals, WP6 comprises three main tasks:

1. A *carbon-aware computing model* capable of shifting loads in space and time to take advantage of low-carbon electricity (T6.1),
2. A flexible and *cost-effective architecture* that allows the minimization of deployed devices, leading thus to the reduction of both environmental impact and costs (T6.2), and
3. An *AI model for performance prediction* needed for the time shift of loads (T6.3). As temporal shifts can obviously only be performed into the future, it is imperative to estimate future computing needs and upcoming carbon intensities of electricity.

To reduce the environmental impact of any product or system, it first needs to be accurately measured. Over the past approximately 20 years, two somewhat distinct approaches have emerged in the field of environmental assessment for Information and Communication Technology (ICT). Assessments found in computing and electrical engineering publications tend to center around the electricity consumed during the operation of ICT equipment. On the other hand, the well-established life-cycle assessment (LCA) methodology, commonly employed in environmental sciences, has also been applied to examine the environmental impact of ICT. While the former studies often benefit from access to high-quality primary data and the domain expertise of the authors, LCA offers a more comprehensive

Funded by Horizon Europe Framework Programme of the European Union

and, therefore, a more semantically accurate modeling method. In particular, LCA allows to account for the embodied footprint of devices, which can in some cases dominate the overall impact of [1], [2]. LCA within FLUIDOS is introduced further below.

Funded by Horizon Europe
Framework Programme of the European Union

2 T6.1: CARBON-AWARE COMPUTING (TUB)

This section presents the fundamental principles of the environmental optimization strategies employed within FLUIDOS, as well as the principles governing the environmental impact assessment that serves as a prerequisite. It is organized as follows: Section 2.1 presents the related work, i.e., existing methods of carbon-aware computing, discussing their individual benefits and drawbacks. Section 2.2 discusses how LCA principles will be applied to FLUIDOS, including challenges. Section 2.3 presents the data sources used for estimating the carbon intensities of electricity. Finally, and building on all these considerations, Section 2.4 presents the principles and proposed architecture of the FLUIDOS carbon-aware scheduler.

The Cost Manager in Figure 1 designates and aggregation and integration of WP6 results into the overall FLUIDOS architecture. Since it is determined to consume and advertise metrics related to local hardware and grid condition the closest cooperation partner for it is the Resource Acquisition Manager, defined in WP2.

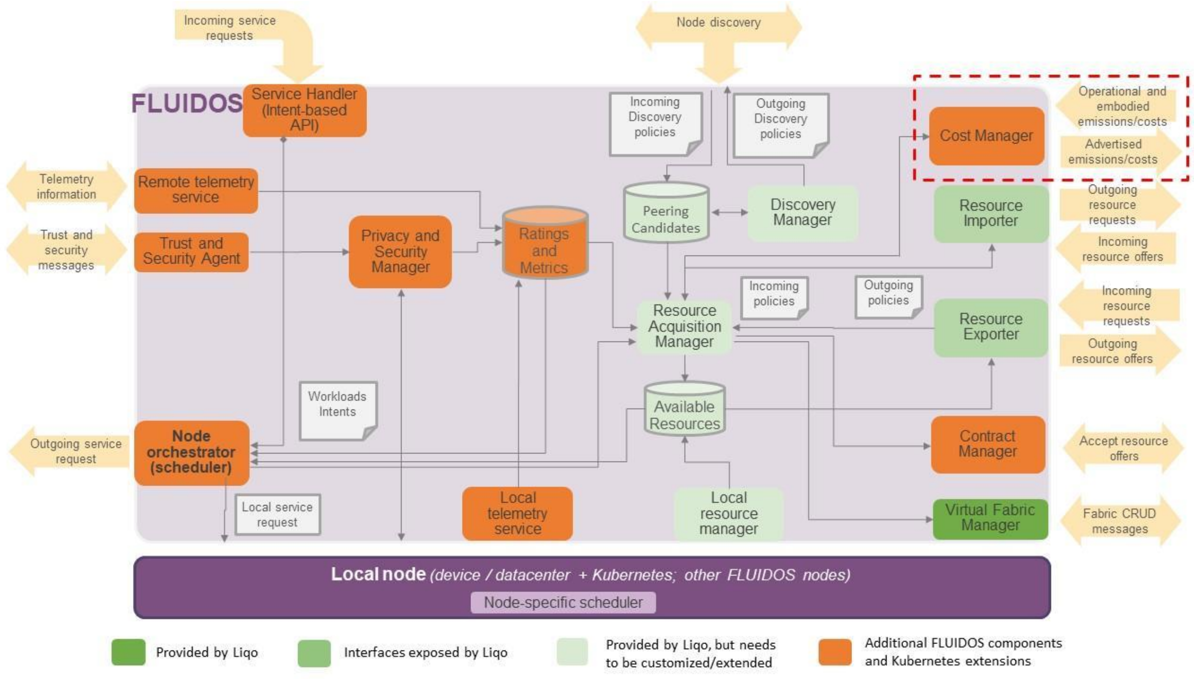
Incoming service requests
Node discovery
FLUIDOS
Service Handler (Intent-based API)
Incoming Discovery policies
Outgoing Discovery policies
Cost Manager
Operational and embodied emissions/costs
Advertised emissions/costs
Telemetry information
Remote telemetry service
Trust and security messages
Trust and Security Agent
Privacy and Security Manager
Ratings and Metrics
Peering Candidates
Discovery Manager
Resource Importer
Outgoing resource requests
Incoming resource offers
Incoming policies
Outgoing policies
Resource Acquisition Manager
Resource Exporter
Incoming resource requests
Outgoing resource offers
Workloads Intents
Available Resources
Outgoing service request
Node orchestrator (scheduler)
Contract Manager
Accept resource offers
Local service request
Local telemetry service
Local resource manager
Virtual Fabric Manager
Fabric CRUD messages
Local node (device / datacenter + Kubernetes; other FLUIDOS nodes)
Node-specific scheduler
Provided by Liqo
Interfaces exposed by Liqo
Provided by Liqo, but needs to be customized/extended
Additional FLUIDOS components and Kubernetes extensions

Figure 1. The Cost Manager (in the red dashed box) is a characteristic unit of WP6 as seen in overall FLUIDOS architecture.

An overview of the task interactions among subtasks within the work package is given in Figure 2. Reduction of both operational and embodied carbon emissions are seen as two focus points of this work package. The details on how to achieve this reduction are given in boxes with dashed borders and are discussed in respective subchapters further in this report.

Funded by Horizon Europe Framework Programme of the European Union

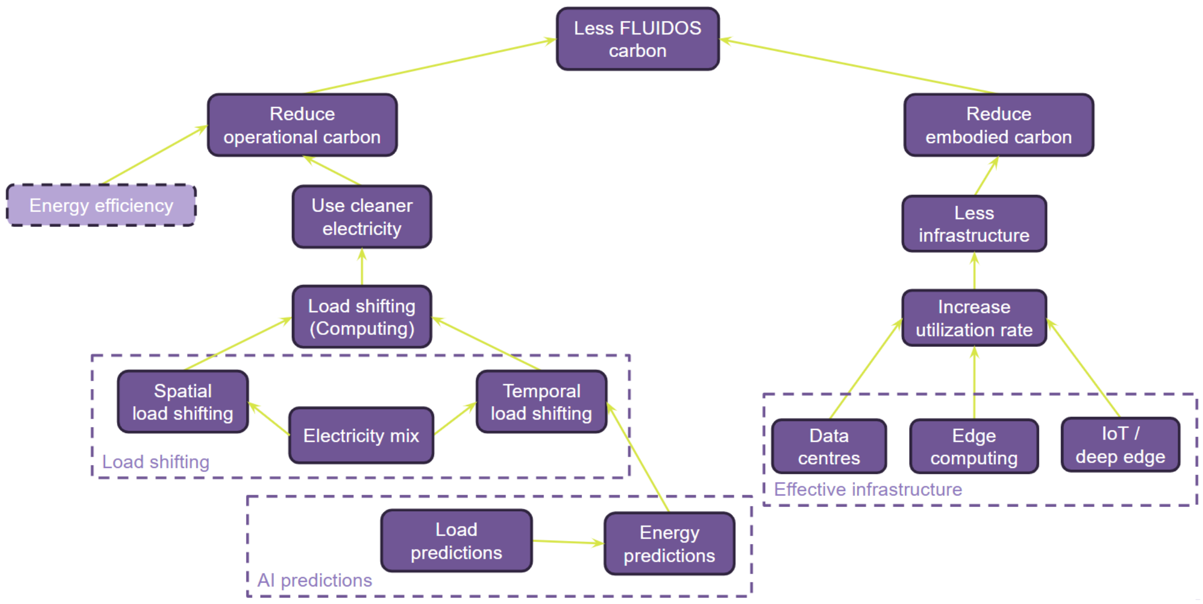

Figure 2. Overview of WP6 scope.

2.1 EXISTING METHODS AND RELATED WORK

To begin the breakdown of the individual studies it is crucial to elaborate on the classification of typical workloads (see Section 2.1.1). This gives a basis for the understanding of which type of workloads have the highest potential for carbon savings and therefore must be prioritized. Further, in sections 2.1.2-2.1.10, most relevant studies on carbon-aware scheduling have been listed and their summary with possible implications of developing a carbon-aware scheduler have been presented. Finally, in Section 2.1.11 an overarching assessment of the limits for carbon savings is given.

2.1.1 Workload classification

According to a classification provided in [3], workloads shiftable by duration can be further broken down into short-running (majority of jobs executed in DCs; e.g. FaaS or CI/CD), long-running (e.g. ML training, scientific simulations or big data analysis) and continuously running workloads (e.g. continuous services such as user-facing API or computationally intensive workloads like blockchain mining, protein folding, etc). Out of those three types, long-running workloads represent the highest potential for carbon savings because of their high energy-intensity and an observable horizon for their execution. Execution time of the workloads has proven to be the cornerstone for understanding the purpose of using demand forecasting for job scheduling, since scheduled workloads (e.g., batch jobs such as nightly tests/builds, periodic backups, etc), depending on their time constraint, be shifted in both directions in time. This is to contrast the so-called “Ad Hoc Workloads” (e.g., FaaS, CI/CD, ML training) the shifting potential of which is limited to the future solely. Lastly, the dichotomy of workload interruptibility is elaborated on. To this point, interruptible workloads (e.g., one big: ML training or

Funded by Horizon Europe Framework Programme of the European Union

multiple small tasks: generation of monthly reports) are naturally preferred over non-interruptible workloads (e.g CI/CD jobs, database migration and backups), since the former can be broken down in a way that maximizes carbon savings.

2.1.2 Google's Carbon-Aware Computing

Developed by Google [4] for its data centers across the globe, this concept envisions shifting computation loads in time and space to take advantage of low-carbon electricity.

The core components are as follows:

1. Fetching pipeline for next day's carbon intensity forecast
2. Power models pipeline for statistical model training which maps CPU usage to power consumption for various power domains characteristic of Google's heterogeneous infrastructure landscape.
3. Load forecasting pipeline responsible for generating the next day's forecasts for workload demand (both flexible and inflexible) on the cluster level. An important characteristic of this pipeline is the evaluation of the forecasting error, which proves to be crucial for the following two components.
4. Optimization pipeline co-optimizes the next day's expected carbon footprint and power peaks. The latter are subject to infrastructure and application service level objective (SLO) constraints as well as contractual and resource capacity limits.
5. SLO violation detection raises a flag when the daily flexible demand of a cluster is not met. This triggers the cluster being excluded from the shaping pool for a week for the load forecasts (component #3) to adapt to the rise in demand.

The algorithm yields the so-called Virtual Capacity Curves (VCCs), which are further used to reshape the peaks of flexible load and shift them towards more favorable times of the day. As demonstrated in Fig. 3, the blue hatched areas represent delayed workloads of the daily flexible capacity that are shifted across the calendar days. Whereas orange hatching stands for the workloads that are being shifted within one calendar day. The VCCs yielded by the proposed multiobjective optimization algorithm are depicted in red and demonstrate peak shaving of the total load (flexible + inflexible) that happens typically around mid-day. Meanwhile, the execution of daily inflexible load (purple) is left intact and is a benchmark for a successful load-shifting strategy yielded by the algorithm on a given day. In addition to cutting down on carbon emissions the above-mentioned peak shaving facilitates a potential reduction of default cluster capacity for future installations. This, in turn, minimizes infrastructure costs and/or hardware wear.

2.1.3 Microsoft's Carbon-Aware Kubernetes

With a similar purpose as Google, Microsoft [5] put forward the idea of expanding a vanilla Kubernetes scheduler with a weighting algorithm to optimize the node-selection process for the purposes of sustainability. Apart from the fundamental components of Kubernetes and functionality developed by [6] this concept from Microsoft incorporates carbon intensity data and a customizable weighting

Funded by Horizon Europe Framework Programme of the European Union

algorithm. The most basic version of this algorithm deals with normalized "Marginal Operating Emission Rate" or MOER for each cluster node. The idea is to take the MOER value of an individual node and divide it by the total MOER values across all nodes to get a normalized percentage weighting of each node. Optionally, a latency constraint, as well as predicted MOER values, can be built into the weighting algorithm for a more sophisticated decision process.

2.1.4 Let's Wait Awhile

Some authors, such as [3], made open accessibility of their work a priority which rendered the use of commercially available data for grid carbon intensity from providers such as electricityMap and WattTime not possible. Thus, the data from a literature review by the International Panel on Climate Change (IPCC) [7] served as the basis for carbon intensity calculations in this study. The estimations encompass the complete life cycle of energy sources used and are based on median values for carbon intensity from the literature review. Even though marginal carbon intensity manages to capture the cause-effect relationship of load shifting better than its counterpart, this metric has been deemed impractical for demand management because of its high uncertainties.

The findings from this study have shown benefits in carbon savings both for short-term (shifting to after sunrise in countries with abundant solar and shifting to later in the night in most countries) and longer-term shifting (to weekends for more than 20% savings). Two of these delay-tolerant workloads have been evaluated experimentally to investigate the sensitivity of parameters such as execution time constraints, scheduling strategies and the accuracy of carbon intensity forecast. The experiments have validated the analytical projections and proven that relaxing time constraints and exploiting the interruptibility of workloads yield in systematic carbon-efficiency gains.

2.1.5 CarbonScaler

In the paper by [8], CarbonScaler, a cloud-based autoscaler that optimizes compute-intensive workloads, such as ML training and scientific computations, for reducing carbon emissions is discussed. CarbonScaler focuses on accounting for energy consumption at the tenant level, utilizing CPU and GPU resources. It employs temporal shifting by delaying job execution during high carbon periods, and spatial shifting by selecting regions with lower carbon footprint for job execution. The paper demonstrates the efficacy of CarbonScaler in reducing carbon emissions for different workloads, configurations, and cloud regions. The authors plan to extend CarbonScaler to cluster-wide scheduling in the future, which presents challenges such as heterogeneity, resource pressure, priorities, and power management.

2.1.6 CarbonExplorer

The paper by [9] discusses the topic of renewable energy in data centers and how it can contribute to carbon reduction. Previous research has focused on on-site renewable energy generation using local solar power and microgrids, but the paper argues that hyperscale data centers can invest in renewable energy on the grid at a larger scale. The use of batteries for energy storage is also explored, as they can help mitigate intermittent renewable energy supply. The declining cost of lithium-ion batteries and

Funded by Horizon Europe
Framework Programme of the European Union

their potential impact on data center design and management are discussed. The paper also highlights the environmental and health risks associated with lithium extraction and battery disposal. Carbon-aware scheduling is proposed to optimize job scheduling based on time-series analysis of renewable energy supplies and data center energy demands. The paper acknowledges that broader sustainability considerations such as electronic waste and water usage for cooling are beyond the scope of the study. Carbon Explorer, a design space exploration tool, is introduced to determine carbon-optimal investment strategies for renewable energy, energy storage, and computation shifting in data centers. The paper concludes by emphasizing the variability of carbon-optimal strategies based on geographic locations and the need for accurate data and transparent reporting in determining carbon footprint optimality.

2.1.7 GreenCourier

The thesis by [10] proposes GreenCourier, an intelligent scheduling approach for serverless computing, i.e. for executions of Function-as-a-Service (FaaS). Real-time data from external sources on carbon intensity are used to make scheduling decisions. The approach aims to minimize carbon emissions during function execution by selecting the most carbon-efficient location. The particularity of this study consists of the use of software infrastructure choice, namely, on top of Knative and Kubernetes, Liqo was used to establish a multi-cluster topology of Kubernetes clusters distributed in various geographical regions. Experimental results show that GreenCourier effectively reduces the carbon footprint by 8.7% and 17.8% per function execution compared to default and geo-aware schedulers. It also outperforms other schemes in correctly identifying ecologically viable regions and deploying pods.

2.1.8 Carbon-Aware Kubernetes Scheduler by Piontek et al.

The primary aim in this work [11] is to reschedule the non-critical tasks to minimize overall CO2 emissions without violating the service level agreement (SLA) deadlines for each job. A constraint added is that if a shiftable job has waited for 24 hours, it must be executed immediately. The SLA for a job is established such that critical jobs, or non-critical jobs that have been pending for over 24 hours, must be run as soon as feasible. Otherwise, non-critical jobs can be delayed up to 24 hours.

To minimize a cluster's total carbon emission, the overall load of the cluster should adapt to decrease power usage during periods of high CO2 intensity and increase during periods of low CO2 intensity. An optimization problem is formulated to minimize the CO2 emissions of a cluster during a specified time interval. The objectives include:

- Ensuring each server's utilization stays below a predefined threshold.
- Meeting the defined SLA for all service and batch jobs.
- Ensuring that the sum of the utilization of all nodes during the time interval remains consistent with a default scheduler.

2.1.9 KEDA

Funded by Horizon Europe Framework Programme of the European Union

One of the latest additions to the workload shifting field is the Carbon Aware KEDA (Kubernetes Event Driven Autoscaling [12]) Operator, which was announced by Microsoft in June of 2023 [13]. Judging from its configuration file, KEDA is enabling a carbon-aware scaling mechanism that adjusts the number of running replicas of a service (“maxReplicaCount”) based on carbon intensity data. If carbon intensity is low, it might allow more replicas, and if it's high, it might reduce them. However, there are conditions under which it may override the “eco mode” and run a specified number of replicas regardless of carbon intensity, especially during specific scheduled times.

2.1.10 Karmada

The system's design involves creating resource templates in the Karmada API server, which can be standard objects like configmaps or custom resources and Custom Resource Definitions (CRDs). These resources aren't scheduled as pods in the control plane cluster. Users specify a Propagation Policy, detailing which resources to pick and their placement. These clusters are selected by naming them in the cluster affinities [14].

The carbon-aware-karmada-operator, a prototype developed as a proof of concept, extends the Kubernetes API by introducing a CRD named CarbonAwareKarmadaPolicy. This CRD facilitates spatial shifting based on carbon intensity, detailing the member clusters and their locations. The operator ranks each member cluster by carbon intensity, selecting those with the lowest values, and reevaluates this ranking every five minutes with updated carbon data. The operator fetches carbon intensity data using the grid-intensity-go library from the Green Web Foundation. It supports various data providers, such as Electricity Maps and WattTime, and maintains an in-memory cache to reduce unnecessary API calls and associated energy use. A significant aspect of this operator is transparency; it updates the status of the custom resource, indicating selected clusters and current carbon intensity data. Furthermore, it emits Prometheus metrics for each cluster, denoting carbon intensity and activity status.

2.1.11 Quantifying the limits of spatiotemporal load shifting

One of the most recent studies [15] has attempted to quantify the upper limits of workload shifting. The primary finding states that the upper bound on carbon saving from such kind of spatiotemporal workload shifting is often largely limited by savings from simple policies, whereas sophisticated techniques provide diminishing returns. According to the authors, the potential for carbon savings coming from temporal shiftings is capped at 25%. Achieving this maximum value is challenging in real life, mostly due to the highest savings coming from scenarios, in which long jobs have high flexibility as well as no resource constraints and accurate knowledge of future grid carbon intensity is available. Fundamentally, regions with more renewables and higher carbon intensity amplitudes benefit from temporal shifting the most.

Deferring the start time of a task results in diminishing carbon savings as the task's duration increases. Short tasks reap the most advantages. The more flexibility in the start time, the higher the potential savings. However, the average carbon reduction across all tasks and regions is under 12%, even with

Funded by Horizon Europe
Framework Programme of the European Union

significant flexibility. Some tasks in specific regions might achieve savings of up to 45%. Nevertheless, the overall effectiveness of adjusting start times will be minimal in most areas, except for very short tasks.

Capacity constraints have been shown to diminish carbon savings by a large margin. This might explain higher reported savings in studies based primarily on simulations. In this study, the integration of capacity constraints has reduced the savings by more than 70%.

For the case of spatial workload migration the study highlighted interactive requests and inference-serving AI/ML systems to benefit the most, since those workloads typically do not have any data dependencies and therefore reduce networking overhead. In regions with diverse carbon intensity profiles savings can reach up to 60% at the expense of a slight increase in latency.

It is worth mentioning that the dynamic of the current situation will likely change. With the increased adoption of renewables, carbon levels will vary more, making temporal shifting more beneficial. A greener grid will necessitate better spatial shifting strategies due to overlapping carbon intensities in different regions.

2.2 LCA WITHIN FLUIDOS

The unique selling proposition of the FLUIDOS carbon-aware scheduler is its accurate sustainability model that is to be enriched with “cradle-to-grave” assessment of the material and energy flows to and from the system both in the manufacturing and its operational phase. To adapt the generic LCA methodology to the FLUIDOS context, the following subsections explore the landscape of LCA in the context of distributed computing systems and lay the foundation to the development of a holistic approach to account for environmental impacts.

2.2.1 Overview of Life Cycle Assessment basics

Life Cycle Assessment (LCA) is the internationally accepted and standardized approach to determine the environmental impact of products and organizations as fact-based as possible. The LCA method deals with both the emissions and resources along all five life stages of a product, which are: raw material extraction, raw material manufacturing, product manufacturing, use stage and end of life. According to ISO 14044 [16], there are four phases of the LCA. First, the goal and scope of the given LCA study along with the product system, system boundary, functional unit (FU), reference flow and motivation are to be defined. In the next phase, data is to be collected and assigned to input/output flows. This phase is also known as Life Cycle Inventory (LCI). To bring the various indicators to a common denominator, resource consumption and emissions are to be characterized to environmental impact categories. After this phase, also called Life Cycle Impact Assessment (LCIA), an evaluation of the results and a sensitivity analysis are typically carried out. Among different available LCIA methods, ReCiPe [17] is the most commonly applied in a European context and provides both midpoint and

Funded by Horizon Europe Framework Programme of the European Union

endpoint results, where endpoint results are an aggregation of the midpoint ones for the purpose of easier interpretation and comparison amongst different studies.

In the context of FLUIDOS, the LCA perspective complements the focus on the energy consumption during operation by an inclusion of the environmental burden of the provision of the hardware infrastructure. Avoidance of device production through a better utilization rate of available devices is a promising optimization mechanism, less explored in comparison to carbon intensity of operational electricity, that contributes to the full picture of the environmental footprint per unit of computation in carbon-aware computing systems.

2.2.2 Definition of a Function Unit

As given in [9] the FU is the quantified utility of a product system for use as a comparison unit and is to be used as a normalization for computation tasks. The FU should relate rather to the function a product fulfills rather than the physical product itself. This means, for example, “seating support for one person working at a computer for one year” rather than “one computer workstation chair”. As initially addressed by [18], a “typical product” in the field of ICT is dynamic and, therefore, contrasts with the classical understanding of functionality. For instance, despite a dramatic reduction of energy needed per transistor, the total energy requirement of the ICT sector did not follow this pattern, primarily due to ever-increasing demand for more powerful chips that contain many more transistors.

The reference flow is based on this concept and is defined as the quantity of resources, products and emissions needed to fulfill the function of the chosen FU. These respective quantities per FU are, however, a particular challenge to calculate for services relevant for FLUIDOS, since the percentage of the allocated resource/product/emission per task is unknown due to the uncertainty of the hardware lifespan.

2.2.3 Distributing the hardware production

In the field of IT and of carbon-aware computing due to a particularly limited amount of manufacturer data a partial LCA is the most viable option. Here, the focus is placed mostly on the allocation for the production and the use phase. Given the heterogeneity of edge devices a sensible aggregation strategy needs to be developed to initiate an LCA study. This is because the emissions from their production vary strongly depending on the given device and its production facility [28].

When deciding on the most energy-efficient and least Global Warming Potential (GWP) - intensive time and place for computation, FLUIDOS considers per unit workload:

1. Current energy consumption of available devices.
2. Environmental performance of energy sources (measured by GWP).
3. Energy consumption during the production phase.
4. GWP of the production phase.

Funded by Horizon Europe
Framework Programme of the European Union

FLUIDOS's primary environmental objective is carbon-aware load distribution. The aim is to distribute computing loads in a manner that minimizes GHG, given that all other conditions are met. Challenges arise when accounting for the production phase of devices:

- The unpredictability of ICT devices' lifespan could lead to a biased preference for older devices.
- Consequently, newer devices could be underutilized, hindering efficiency.

To mitigate this, production phase flows should be divided by a device's expected computational tasks, requiring assessments of expected computation cycles per device and knowledge of device characteristics and lifespans.

Lifespan and Use Phase Distinctions:

- Devices like smartphones and tablets have shorter lifespans (few years) compared to data centers, which last over a decade [19], [20].
- Lifespans can vary between brands, especially in consumer electronics [21]. This requires custom resource and emission allocation per device.

GWP Distribution across Device Lifecycle:

- GWP distribution varies among devices. For instance, tablets and laptops have a major GWP share in the production phase. In contrast, servers and data storage units have a smaller relative share (per unit workload) of production phase GWP [22], [23].

ICT Sector Evolution:

- ICT sector advancements have led to increased energy efficiency, especially for servers and data storage units, as evidenced by Koomey's Law [16].
- About 80-95% of GWP for these devices arises from energy consumption during use [22]. Hence, replacing existing functional hardware might be more GWP-efficient than continued usage. However, if the operational energy demand is provided by renewable energy as the most straightforward optimization measure, the production of the hardware becomes the hotspot of the whole service.

2.3 DATA SOURCES

The scheduling algorithm demands a significant amount of variables to be fed in as time series and therefore acquisition of reliable data sources is one of the major goals in the first stage of algorithm development. The two major data sources, namely for quantifying the “greenness” of the grid and energy consumption of the hardware are presented in the following two subchapters.

2.3.1 Carbon Intensity

Funded by Horizon Europe
Framework Programme of the European Union

According to [24], operational energy and carbon performance represent the most promising paths to low-carbon data centers, even from an embodied (hardware manufacturing) emission perspective. Therefore, in striving to reduce the carbon footprint of computations particular attention is to be placed on the quality of operational electricity, namely at its carbon intensity. It refers to how many grams of carbon dioxide (CO2) are released to produce a kilowatt hour (kWh) of electricity [25].

Fundamentally, two types of grid carbon intensities are used for carbon-aware scheduling: average and marginal [26]. Average grid carbon intensity is a viable option for scheduling purposes; however, it fails to capture the causalities in the electricity grid. Marginal carbon intensity data on the other hand reflects the increased carbon intensity after scheduling new jobs in case a new power plant with typically higher environmental burden must go online.

There are multiple sources for gathering this data, some of which are open source, while others belong to commercial institutions. The most prominent examples of the latter are WattTime [27] and electricityMap [28], both of which provide not only the current estimation but also a day-ahead forecast of marginal grid carbon intensities for various locations across the globe. WattTime uses empirical modeling techniques to analyze marginal emissions caused by electricity generation in real-time [29]. They use Continuous Emissions Monitoring System (CEMS) data to study how power plants respond to changes in electricity demand. WattTime places focus on data-driven causal models rather than assumption-driven models and they have extended the techniques in the published literature to leverage real-time power grid data and additional datasets on renewable energy curtailment.

A number of grid operators [30]–[32] have open-sourced their data, however, load shifting across several of those regions proves to be challenging due to the heterogeneity of the metrics used.

One of the most detailed models producing reliable carbon intensity data, namely on marginal emission factors (MEFs) has been developed for Germany by [33]. All available data including both generation and transmission has been aggregated for a predefined number of nodes. This, in turn, enables a dynamic assessment of the load shifting strategies among those nodes, since taking into consideration the heterogeneity of carbon intensity data for different nodes within one country has a high potential of providing additional carbon savings. The model is in the process of being expanded to EU-countries and its data proves to be of high value for load shifting across the continent.

A recent addition to the carbon intensity tracking solution is the Kubernetes Carbon Intensity Exporter by Microsoft [34]. An advantage of this solution is its integration with the Carbon Aware SDK [35], as it runs in the same pod and allows the exporter to integrate any carbon intensity data source available through itself. Working together with Azure, the SDK maps the region that it is running to a geolocation to pass it to the WattTime API. The next step for the exporter is to fetch the carbon intensity forecast for the next 24 hours every 12 hours and save the data to a configmap on the cluster to be later passed to the KEDA operator. The configmap design has the advantage of a custom exporter with a different source of carbon intensity data being developed [36].

2.3.2 Energy measurement tools

Funded by Horizon Europe Framework Programme of the European Union

The following sections will analyze the available tools to collect power consumption metrics from Linux-based systems.

Intel RAPL

RAPL (Running Average Power Limit) is a reporting interface for cumulative energy usage of multiple system-on-chip (SoC) power domains. The RAPL energy reporting capability has been present on Intel SoC devices for several generations, and energy reporting is industry standard practice. This energy information is used by Intel processors for internal SoC management functions, such as controlling SoC power limits in conjunction with Intel Turbo Boost Technology.

At runtime, certain privileged software may execute platform power and temperature management. Energy information from RAPL may be used by such applications to adjust system performance or track power use. Energy data is occasionally utilized in server systems to perform rack-level power management and efficiency loading across units.

Platforms in RAPL are separated into domains for fine-grained reporting and control. A RAPL domain is a physically significant power management domain. The exact RAPL domains offered in a platform differ depending on the product category. RAPL implements four power domains:

1) Package Domain: This power domain represents the power consumption of the CPU's complete package, including cores and additional components (i.e., integrated graphic card).
2) DRAM Domain [37]: This power domain accounts for the power consumption of the DRAM. It is only accessible on servers.
3) PP0/Core Domain (PowerPlane 0): is used to monitor the power of the CPU cores only.
4) PP1/Graphic Domain (PowerPlane 1): is used to monitor the power of the CPU's graphic component alone. Because of this, it is only available for non-server architectures. Graphic components are not included in Intel server designs.

NOTE: The reported power consumption at the Package domain is: PPKG=PP0+ PP1.

Each RAPL domain supports:

- ENERGY_STATUS for power monitoring.
- POWER_LIMIT and TIME_WINDOW for controlling power.
- PERF_STATUS for monitoring the performance impact of the power limit.
- RAPL_INFO for information on measurement units, the minimum and maximum power supported by the domain.

Funded by Horizon Europe
Framework Programme of the European Union

In addition, RAPL has 32-bit performance counters for each power domain to monitor energy use and overall throttled time.

ACPI interface

The Advanced Configuration and Power Interface (ACPI) specification is an open standard that was created by a group of hardware and software companies including HP, Intel, Microsoft, Phoenix and Toshiba. It defines standard interfaces that allow operating systems to configure motherboard devices and control power. ACPI may be used for specific hardware components as well as the full system. Furthermore, it monitors the state of the system and utilizes power management techniques by altering the CPU running frequency and putting unnecessary components to sleep.

ACPI is intended to allow the operating system to control each hardware component. Prior to the development of ACPI, power management was handled via Plug and Play (PnP) and Advanced Power Management (APM) subsystems, which were implemented in hardware and hence were less versatile in terms of management capabilities. ACPI, on the other hand, is implemented in the operating system (OS) layer and thus gives greater freedom for component management, as well as being hardware independent (if the hardware meets the ACPI standard).

The ACPI allows the operating system to push certain hardware devices to a low power consumption state when they are not in use. Similarly, if the operating system determines that the applications do not demand a huge number of resources, ACPI can move the entire environment to a low power consumption mode.

The ACPI specification details different device states to represent the execution of the system. Specifically, they can be classified into:

- **G states**, representing the system's global execution state (e.g., working, sleeping, …).
- **D states**, representing the device-dependent states (e.g., fully ON, …).
- **C states**, representing the CPU power states. They detail the different CPU execution states (e.g., operating state, halt, sleep, …).
- **P states**, representing the operating frequency of the CPU cores (e.g., maximum frequency, frequency scaled, …).

Using ACPI it is thus possible to interact with the different states of the device and collect relevant metrics on the device execution. Furthermore, it is possible to modify the various states according to energy-aware management algorithms.

IPMI interface

IPMI is a collection of computer interface requirements for a self-contained computer subsystem that offers administration and monitoring capabilities independently of the host system's CPU, firmware (BIOS or UEFI), and operating system. IPMI is a set of interfaces used by system administrators for out-

Funded by Horizon Europe Framework Programme of the European Union

of-band administration and monitoring of computer systems. For example, IPMI allows the administration of a machine that is switched off or otherwise unresponsive to connect to the hardware over a network rather than an operating system or login shell.

The use of a standardized interface and protocol allows IPMI-based systems management software to control many different servers. IPMI, as a message-based, hardware-level interface specification, functions independently of the operating system (OS) to enable administrators to remotely control a system in the absence of an operating system or system management software. As a result, IPMI functionalities can be used in one of three scenarios:

- Before an OS has booted (allowing, for example, the remote monitoring or changing of BIOS settings).
- When the system is powered down.
- After OS or system failure – the key characteristic of IPMI compared with in-band system management is that it enables remote login to the operating system using Secure Shell (SSH).

IPMI messaging may be used by system administrators to monitor platform status (such as system temperatures, voltages, fans, power supplies, and chassis intrusion).

The intelligence in the IPMI architecture is provided by the baseboard management controller (BMC). It is a customized microcontroller that is integrated into the motherboard of a computer, most commonly a server. The BMC is in charge of overseeing the interaction between system management software and platform hardware. BMC has its own firmware and RAM.

Temperature, cooling fan speeds, power status, OS status, and other characteristics are reported to the BMC by various sensors embedded into the computer system. The BMC monitors the sensors and can transmit network warnings to a system administrator if any of the metrics exceed pre-set thresholds, signaling a potential system failure.

KEPLER

Kepler (Kubernetes-based Efficient Power Level Exporter) [38] is a Kubernetes-based solution for energy and power consumption monitoring on a cluster of servers. It uses eBPF to probe CPU performance counters and Linux kernel tracepoints and collect the exact CPU consumption of the different processes running on the system. Kepler then correlates such information with the power consumption metrics of the device to compute the contribution of the different processes to the overall power consumption of the device.

Specifically, power consumption metrics are collected using the tools detailed in the previous section (INTEL RAPL, ACPI, IPMI) if available. If not available or incomplete, Kepler relies on a pre-trained neural network to generate an estimate of the power consumption, the Kepler Model Server. The main feature of the Kepler Model Server is to return a power estimation model corresponding to the request containing target granularity (node in total, node per each processor component, pod in total, pod per

Funded by Horizon Europe Framework Programme of the European Union

each processor component), available input metrics, and model filters such as accuracy. In addition, the online trainer can be deployed as a sidecar container to the server (main container) to execute training pipelines and update the model on the fly when power metrics are available.

The default Kepler installation consists of a pod, the Kepler Exporter, deployed on each node of the Kubernetes cluster. The pod is then in charge of collecting all the metrics previously mentioned and interacting with the model server to get estimates if the actual metrics are not available.

Metrics can then be collected and stored using Prometheus and the OpenTelemetry standard for data visualization and post-processing. Examples of exported metrics are:

- **kepler_container_core_joules_total** (Counter) This measures the total energy consumption on CPU cores that a certain container has used.
- **kepler_container_gpu_joules_total** (Counter) This measures the total energy consumption on the GPUs that a certain container has used.
- **kepler_node_core_joules_total** (Counter) Similar to container metrics, it represents the aggregation of all containers running on the node and operating system (i.e., “system_process”).
- **kepler_node_gpu_joules_total** (Counter) Similar to container metrics, it represents the aggregation of all containers running on the node and operating system (i.e., “system_process”).

Depending on the environment, Kepler's power consumption metrics collection varies [39]. It can be deployed in both BM and VM settings.

Bare-Metal (BM) Environment:

- Direct collection of real-time system power metrics is possible.
- Power metrics comprise both dynamic (related to resource utilization) and idle power (constant irrespective of system load).
- The Ratio Power model divides dynamic power across all processes. For example, if a program uses 10% of the CPU, it takes up 10% of the total CPU power.
- Idle power estimation adheres to the GreenHouse Gas (GHG) protocol guideline, splitting it among processes/containers based on their size.
- Different resource utilizations in Kepler are gauged differently, e.g., CPU instructions for CPU utilization and cache misses for memory utilization.

Virtual machine (VM) Environment:

- Direct power measurement for a VM isn't possible in public clouds.
- Kepler's proposed solution involves initially deploying it in the bare-metal node, where it measures real-time power metrics. This data is then exposed to the VM, either through "hypervisor Hypercalls" or specific files that the VM can access, like the cGroup file.

Funded by Horizon Europe
Framework Programme of the European Union

- Another instance of Kepler within the VM then employs the Ratio Power Model to evaluate power usage by processes in the VM.
- In absence of the passthrough approach, Kepler estimates VM's dynamic power consumption using trained power models.
- Limitation: VMs, especially in public clouds, can't split idle power since one can't discern how many other VMs are operating on the host.

2.4 FLUIDOS CARBON-AWARE SCHEDULER

Having discussed the state of the art and generic approaches to carbon-aware scheduling, the foundations for the definition of the FLUIDOS carbon-aware scheduler has been laid. In the first subsection the principles of the scheduling logic are described on a high level, whereas in the following subchapter the actual prototyping challenges of the designed algorithm are shown, culminating in pseudocode demonstration of the proposed algorithm.

2.4.1 Principles

Spatial Scoring Principles

1. Ease of Implementation: Spatial scoring is relatively straightforward to implement, primarily requiring live data on carbon intensity.

2. Data Retrieval and Processing: It involves obtaining data from the Kubernetes API server about pods and nodes, either through direct HTTP calls or client libraries.

3. Geographical Data Utilization: The geographic region of each node is identified and used in the scoring process.

4. Carbon Intensity Integration: Current carbon intensity data is incorporated, obtained either directly from providers or via a Metrics Server aggregating various sources.

5. Node Scoring and Selection: Nodes are scored based on key-value pairs, normalized, and the node with the highest score is selected for the pod deployment.

6. Database Interaction: The chosen node's information is updated in the podSpec field and communicated to the K8s cluster's ETCD database via the API server.

Temporal Scoring Principles

1. Complexity and Sophistication: This approach is more complex than spatial scoring, requiring a nuanced understanding of workload characteristics and timings.

Funded by Horizon Europe Framework Programme of the European Union

2. Pod Characterization: Each pod is characterized by a tuple, including its name, shiftability, and CPU utilization.

3. Time Interval Definitions: Two different time intervals are defined - one for the overall inspection period and the other for a selected subset within it.

4. Predictive Analysis: The algorithm searches for the most appropriate time frame for each pod based on predicted carbon intensity and workload.

5. Forecast Sensitivity and Threshold Setting: The algorithm must balance forecasts of carbon intensity and workload, setting threshold values to avoid scheduling conflicts and ensure system resilience.

6. Dynamic Workload Limitation: Workload limits are dynamically set based on hardware and service level agreements (SLAs), with contingencies for critical tasks and peak utilizations.

7. Capacity and Stability Trade-off: A portion of CPU capacity is reserved to improve system stability and Quality of Service (QoS), albeit at the cost of reduced cluster capacity for workload shifting.

8. Conditional Scheduling Logic: The scheduling decision is made through a multi-level conditional structure, prioritizing critical tasks, considering waiting times, and evaluating current CPU utilization and optimal carbon intensity windows.

2.4.2 Architecture and prototyping

Choosing the best custom scheduler development method largely hinges on the specific needs and constraints of the project. Writing a custom scheduler provides the highest level of control but also brings along the most complexity (see Table 1). The Scheduler Extender and Scheduler Framework Plugins are middle-ground options, allowing for customization while still leveraging the existing ”kube-scheduler” base. Lastly, existing solutions offer the advantage of community-backed tools but may not entirely fit unique use-cases.

Table 1. Pros and Cons of different Kubernetes Scheduler customization methods.

Method	Pros	Cons
Own custom scheduler	1. Complete control over scheduling logic. 2. Flexibility to design for specific use cases. 3. Can coexist with the default scheduler.	1. High complexity. 2. Needs to handle updates and compatibility issues with newer Kubernetes versions.
Scheduler extender	1. Leverages existing	1. Potentially

Funded by Horizon Europe Framework Programme of the European Union

	features of the vanilla scheduler. 2. Sophisticated enough for most scheduling goals.	increased latency due to extra HTTP calls. 2. Maintenance with Kubernetes updates may be required.
Scheduler Framework Plugins	1. Easy to manage due to its modularity. 2. Overrides or extends only the specified phases of scheduling.	1. Requires knowledge on specific plugin points. 2. Maintenance is still not fully automatic.

A scheduler extender is a service that can help to make scheduling decisions. It is an HTTP server that the main scheduler communicates with via HTTP APIs. The extender can filter nodes and prioritize them, and the main scheduler combines these results with its results. This approach allows you to add custom scheduling behavior without changing the main scheduler:

- Ease of Prototyping
- Language Agnostic: You can write extenders in any language, not limited to the language the main scheduler is written in (Go).
- Loose Coupling: It interacts with the main scheduler over HTTP APIs, allowing it to be a standalone service, which promotes separation of concerns and easy maintainability.

During the process of assigning a pod, the extender facilitates the involvement of an external entity to both sieve and rank nodes. Two distinct HTTP/HTTPS requests are dispatched to the extender, individually representing "filter" and "prioritize" functions. If scheduling the pod isn't feasible, the scheduler intends to displace pods with inferior priority from nodes, directing them to the extender's "preempt" function, provided it's set up. The extender has the ability to provide a narrowed down selection of nodes along with fresh victims to the scheduler. Furthermore, by executing the "bind" procedure, the extender has the discretion to link the pod to the apiserver [40].

Spatial scoring

Arguably, the implementation of spatial scoring is the easiest way of load shifting since live data on carbon intensity typically suffices for the scheduler to take a definitive action. However, spatial scoring unlike temporal scoring discussed further below requires an analysis of embodied emissions of node hardware to get a full picture of the carbon footprint of the executed jobs in the form of the “Total Node Score” or TNS (see Figure 3). To calculate TNS embodied emissions of the node hardware and the operational emissions from executing the specific job are added up to get a value of kgCO2e per hour of execution at nominal load of the node. In future iterations, the algorithm should be refined to include the characterization of pods based on their anticipated runtime.

Funded by Horizon Europe Framework Programme of the European Union

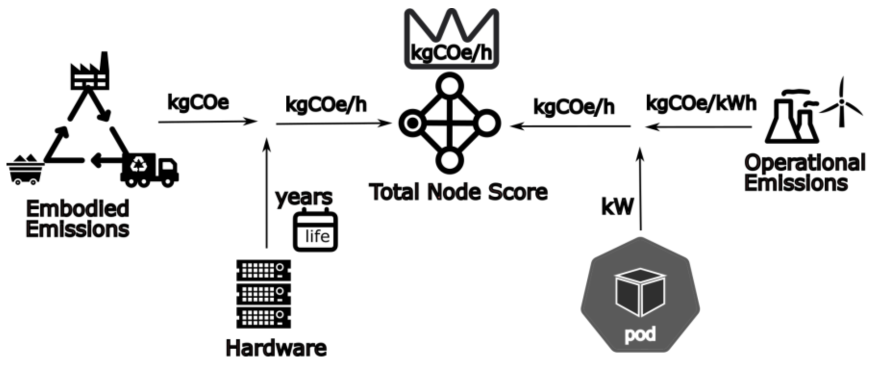

Figure 3. Calculation of the Total Node Score.

Similar to [6] and [10] the spatial scoring logic for the FLUIDOS carbon-aware scheduler extender will define a function that takes in the pod object along with a list of available node objects from the Kubernetes API server. This can be done either by spelling out HTTP calls manually, such as “GET /api/v1/namespaces/{namespace}/pods/{name}” for details on a specific pod or “GET /api/v1/nodes'' to get the list of nodes, or by using client libraries which exist for programming languages like Go [41] and Python [42]. The algorithm flowchart and its pseudocode are provided in Figure 4 and Figure 5 respectively.

Funded by Horizon Europe
Framework Programme of the European Union

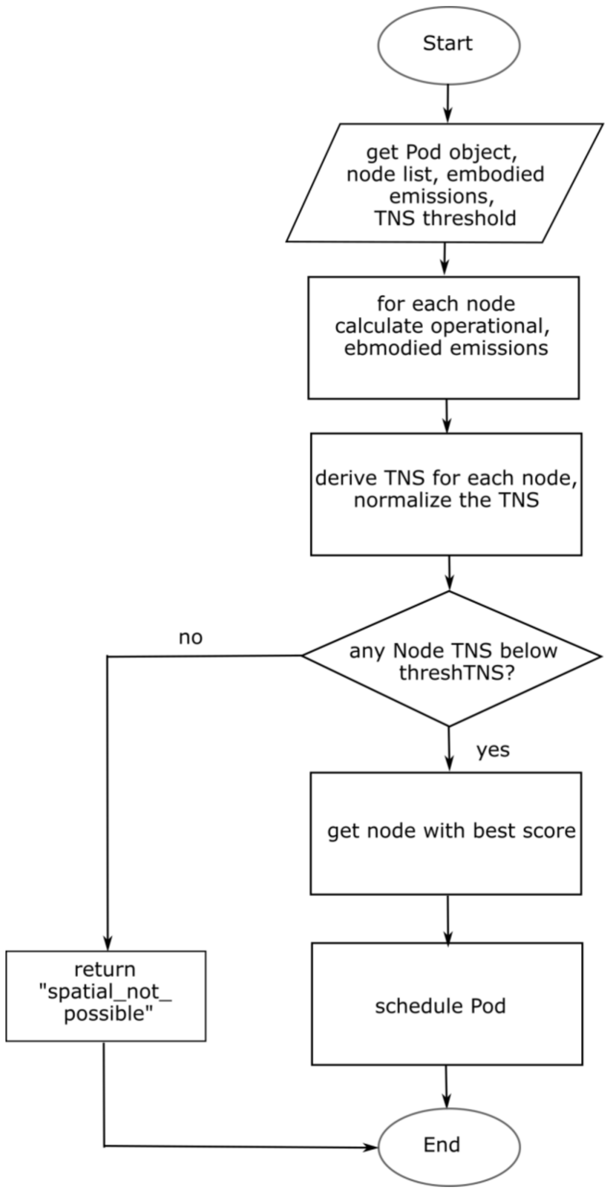
Start
get Pod object, node list, embodied emissions, TNS threshold
for each node calculate operational, ebmodied emissions
derive TNS for each node, normalize the TNS
any Node TNS below threshTNS?
no
yes
get node with best score
schedule Pod
return "spatial_not_ possible"
End

Figure 4. Flowchart of the spatial scoring algorithm.

Funded by Horizon Europe Framework Programme of the European Union

```
Algorithm 1 Spatial Scoring
 1: function SPATIAL_SCORING(pod, nodes, emb_emis_data, thresh_tns)
 2:     for each node in nodes do
 3:         $region \leftarrow node.get\_annotation('region')$
 4:         $operational\_score \leftarrow retrieve\_operational\_data(region) \times pod.power\_consumption$
 5:         $emb\_emis\_score \leftarrow calculate\_emb\_emis\_score(node, emb\_emis\_data)$
 6:         $tns \leftarrow operational\_score + emb\_emis\_score$
 7:         $update\_and\_store\_total\_node\_score(region, tns)$
 8:         $normalize\_total\_node\_scores()$
 9:     end for
10:     if all(tns > thresh_tns for tns in nodes' tns_list) then
11:         return 'spatial_not_possible'
12:     else
13:         $optimal\_node \leftarrow get\_node\_with\_best\_score()$
14:         $assign\_pod\_to\_node(pod, optimal\_node)$
15:         return $pod$
16:     end if
17: end function
```

Figure 5. Pseudocode of the spatial scoring function.

1. **Function Definition:**
    - **function Spatial_Scoring(pod, nodes, emb_emi_data, thresh_tns)**: This line defines the **Spatial_Scoring** function. It takes four parameters: **pod** (representing a pod object), **nodes** (a list of node objects), **emb_emis_data** (data about embedded emissions) and **thresh_tns** (a threshold for the Total Node Score).
2. **Iterating Over Nodes:**
    - **for each node in nodes:** The function iterates over each node in the provided list of nodes.
3. **Extracting and Calculating Scores:**
    - **region = node.get_annotation('region')**: Retrieves the region information from the current node.
    - **operational_score = retrieve_operational_data(region) * pod.power_consumption**: Calculates the operational score for the node based on data retrieved for the region and the power consumption of the pod.
    - **emb_emis_score = calculate_emb_emis_score(node, emb_emis_data)**: Calculates the embedded emissions score for the node based on the provided embodied emissions data.
    - **tns = operational_score + emb_emis_score**: Summarizes the total node score (TNS) by adding the operational and embedded emissions scores.
4. **Updating and Storing Scores:**

Funded by Horizon Europe Framework Programme of the European Union

- **update_and_store_total_node_score(region, tns)**: Updates and stores the total node score for each region.
- **normalize_total_node_scores()**: Normalizes the total node scores.

5. **Evaluating Scores Against Threshold:**
   - **if all(tns > thresh_tns for tns in nodes' tns_list):**: Checks if all TNS values are below the provided threshold **thres**hTNS.
     - If true, **return 'spatial_not_possible'**: Indicates that spatial scoring is not possible (i.e., no node meets the required criteria).
6. **Selecting Optimal Node:**
   - If not all TNS values are below the threshold:
     - **optimal_node = get_node_with_best_score()**: Finds the node with the highest score.
     - **assign_pod_to_node(pod, optimal_node)**: Assigns the pod to this optimal node.
7. **Returning the Pod:**
   - **return pod**: Returns the **pod** object, now assigned to the optimal node.

Temporal scoring

Scoring procedure for temporal workload shifting requires a more sophisticated approach. The algorithm flowchart and its pseudocode are provided in Figure 7 and Figure 8 respectively. Here, both workload and carbon intensity forecasts play a crucial role, while embodied emissions are not relevant anymore since the workload stays on the given node under all circumstances.

An important task which has been encapsulated in its own function is the calculation of the so-called "optimal sliding window" (see Figure 6). This is derived by comparing a subset of hours to be devoted for task execution within a specified time horizon (dictated by the available forecast data) by their average carbon intensity and workload amounts. Here, higher resolutions are potentially capable of yielding higher carbon savings but are more sensitive to forecasting errors.

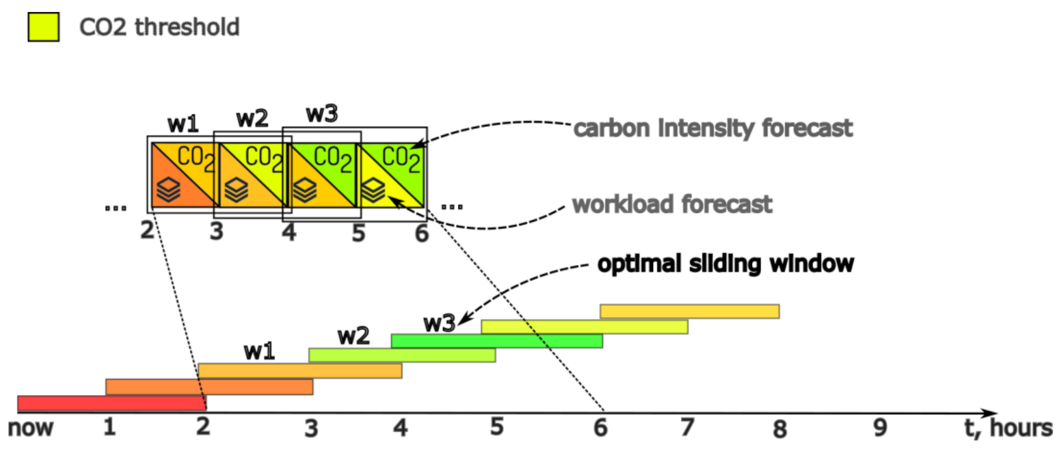

Figure 6. Calculation of the optimal sliding window.

Funded by Horizon Europe Framework Programme of the European Union

Given the double uncertainty arising from the combination of carbon intensity and workload forecast a priority needs to be set, in case the carbon intensity forecast gives green light to the scheduler, but the load forecast implies that the total load will be critically high. To avoid this clash, a threshold value for carbon intensity (potentially dynamically, based on local grid conditions) needs to be selected. This value would allow a bigger pool of nodes to be available for scheduling, in case the node with the most beneficial carbon intensity conditions is awaiting an overflow. Another key variable in the temporal scoring algorithm is the workload limit, which is set dynamically for every hour by the workload prediction pipeline. At this point two factors are critical for system resilience, the general threshold (upper_threshold) for CPU overflow and in cases of peak utilization an additional amount of CPU power is reserved for system-critical tasks (prediction_threshold). The values to be picked are dependent on the hardware and SLAs. To give an example, if 10% of the CPU is assumed to be blocked, the maximum possible utilization is 90% and in case non-critical tasks populate this designated capacity additional 10% of the maximum CPU utilization will be blocked for critical tasks. These measures reduce the capacity of the cluster for workload shifting by 20% in total but are expected to significantly improve the overall stability and therefore the quality of service (QoS). Additionally, for most servers from the energy efficiency perspective it is not attractive to exceed 80% utilization [43], which provides another reason for artificially limiting maximum available capacity on a given machine.

Funded by Horizon Europe
Framework Programme of the European Union

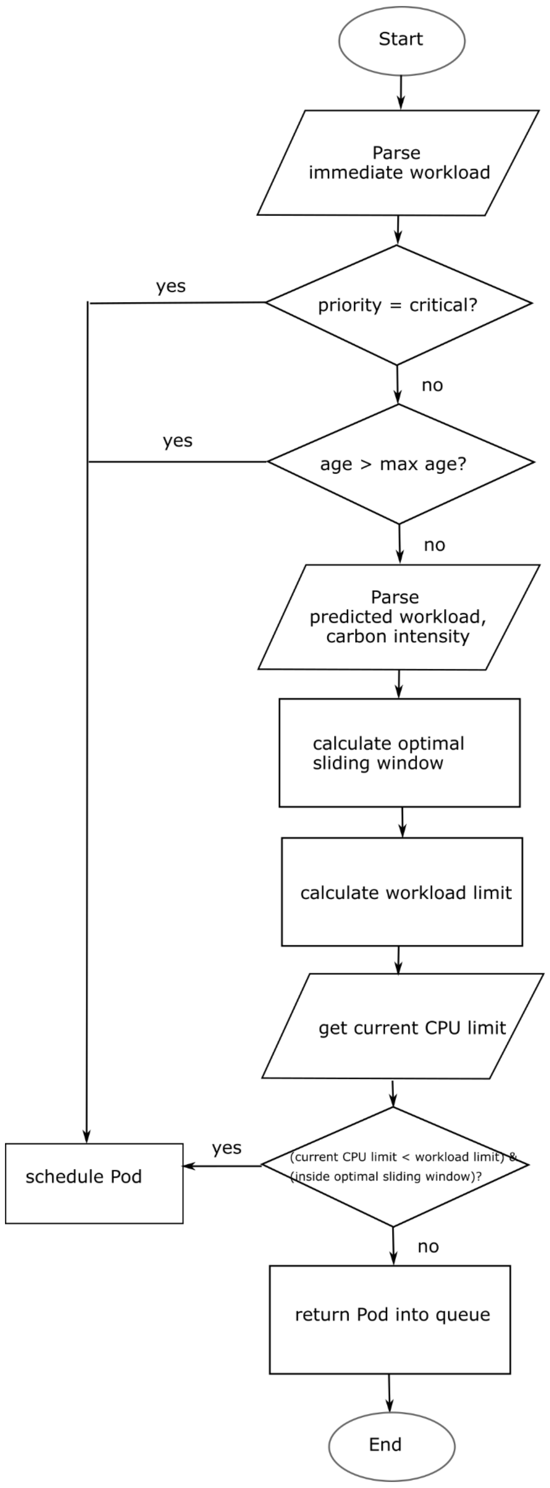

Figure 7. Flowchart temporal scoring algorithm.

Funded by Horizon Europe Framework Programme of the European Union

Algorithm 2 Temporal Scoring with Optimal Window Finding

```
1:  $ci \leftarrow$ carbon intensity forecast
2:  $iw \leftarrow$ immediate workload
3:  $pw \leftarrow$ predicted workload
4:  function FIND_OPTIMAL_WINDOW(ci, pw)
5:      $optimal\_index \leftarrow -1$
6:      $filtered\_frames \leftarrow$ FILTER(sliding_window_frames, CO2 < thresh_CO2)
7:      $sorted\_frames \leftarrow$ SORT(filtered_frames, by predicted_workload)
8:      if not ISEMPTY(sorted_frames) then
9:          $optimal\_frame \leftarrow sorted\_frames[0]$
10:         $optimal\_index \leftarrow$ INDEXOF(sliding_window_frames, optimal_frame)
11:     end if
12:     return $optimal\_index$
13: end function
14: function TEMPORAL_SCORING(iw, ci, pw)
15:     $pod\_list \leftarrow$ list of (iw.pod.name, iw.pod.priority, iw.pod.resources)
16:     for each pod in pod_list do
17:         if priority_class(pod) = 'critical' then
18:             schedule(pod)
19:         else if age_of(pod) > max_pod_age then
20:             schedule(pod)
21:         else
22:             $optimal\_window\_index \leftarrow$ FIND_OPTIMAL_WINDOW(ci, pw)
23:             if optimal_window_index $\neq$ -1 then
24:                 $workload\_limit \leftarrow upper\_threshold - workload\_prediction\_data[hour] \times prediction\_threshold$
25:                 $cpu\_limit \leftarrow$ get_cpu_limit_from_kubectl(pod)
26:                 if $cpu\_limit < workload\_limit$ and is_within_optimal_CO2_window(optimal_window_index, ci) then
27:                     schedule(pod)
28:                 else
29:                     requeue(pod)
30:                 end if
31:             else
32:                 requeue(pod)
33:             end if
34:         end if
35:     end for
36: end function
```

Figure 8. Pseudocode of the temporal scoring function.

Funded by Horizon Europe Framework Programme of the European Union

1. **Temporal_Scoring Function:**
   - The **Temporal_Scoring** function takes in parameters: immediate workload, carbon intensity, and workload prediction data. Its purpose is to determine the best time to schedule a pod, taking into account the current and predicted state of the system and environmental factors.
2. **Initialization of Pod List:**
   - A list of pod characteristics (pod list) is initialized, containing information like pod name, priority, and resources.
3. **Iterating Over Pods:**
   - The algorithm iterates over each pod in the pod list to determine the optimal scheduling time.
4. **Priority and Age Checks:**
   - For each pod, if its priority class is 'critical' or if the pod's age exceeds a maximum threshold (indicating it has been waiting too long), the pod is scheduled immediately.
5. **Finding the Optimal Window:**
   - If the pod is not critical or old, the algorithm proceeds to find the optimal scheduling window using the **find_optimal_window** function described below.
6. **Scheduling Decision Based on Window:**
   - If an optimal window is found and the pod's CPU limit is within the calculated workload limit for that window, the pod is scheduled.
   - If no suitable window is found or the CPU limit exceeds the workload limit, the pod is returned to the queue to be considered later.
7. **find_optimal_window Function:**
   - This helper function is called by **Temporal_Scoring** and is responsible for selecting the optimal window from available time frames, based on carbon intensity and workload predictions.
8. **Filtering and Sorting Time Frames:**
   - The function first filters out time frames where the CO2 intensity is higher than the acceptable threshold.
   - The remaining frames are then sorted based on their predicted workload, prioritizing those with the lowest workload.
9. **Selecting the Optimal Frame:**
   - If the sorted list of time frames is not empty, the first frame (with the lowest workload) is considered optimal.
   - The index of this optimal frame in the context of the original list of time frames is identified and stored in **optimal_index**.
10. **Return Optimal Index:**
    - The **find_optimal_window** function concludes by returning the index of the optimal frame. If no frame meets the criteria, it returns **-1**.

Funded by Horizon Europe Framework Programme of the European Union

Spatiotemporal scoring

The spatiotemporal algorithm, in its present iteration, functions as a proxy to schedule pods using either spatial or temporal scoring methods, supplemented by two conditional clauses. Spatial scoring is prioritized because it avoids the complexities of double forecasting uncertainty. This algorithm represents an initial exploration of how to integrate both spatial and temporal shifting. Even in its current form, it represents a novelty, as the literature does not simultaneously explore spatial and temporal shifting.

In its next iteration, we plan on expanding this algorithm to no longer favor spatial over temporal shifting, but to look into all nodes that come into question for a (current or future) optimal window. The current algorithm flowchart and its pseudocode are provided in Figure 9 and Figure 10 respectively.

Funded by Horizon Europe Framework Programme of the European Union

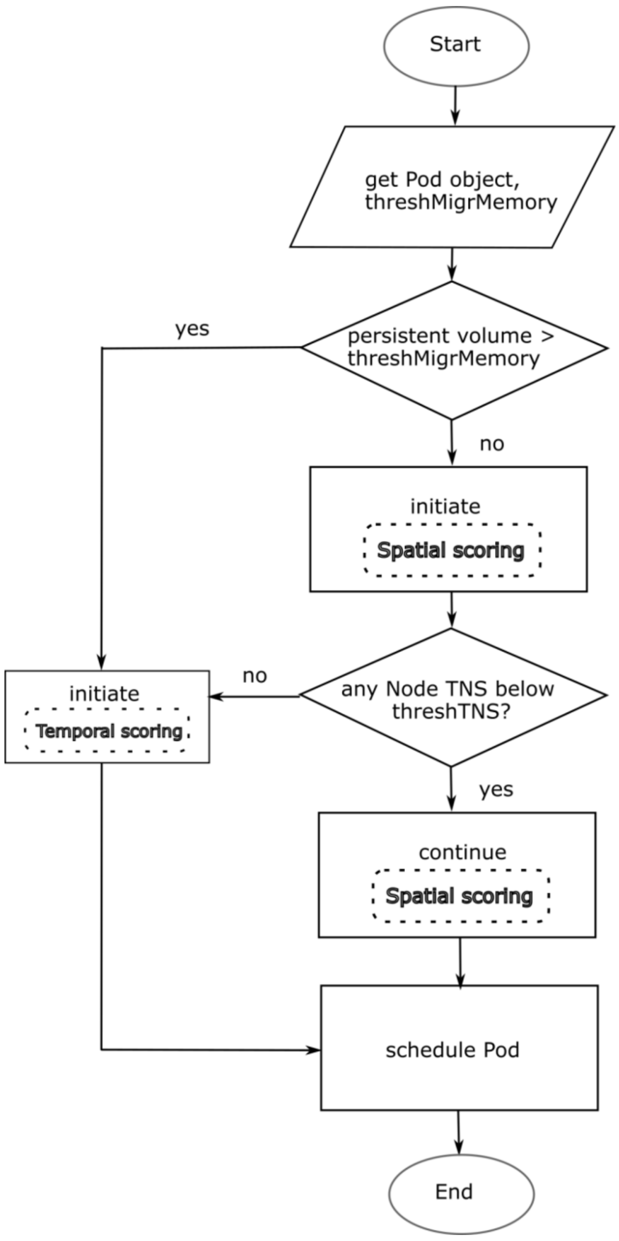

Figure 9. Flowchart spatiotemporal scoring algorithm.

```
Algorithm 3 Spatiotemporal Scoring
 1: $ci \leftarrow$ carbon intensity forecast
 2: $iw \leftarrow$ immediate workload
 3: $pw \leftarrow$ predicted workload
 4: function SPATIOTEMP_SCORING(pod, thresh_migr_memory, nodes, emb_emis_data, thresh_tns, iw, ci, pw)
 5:     if GETPERSISTENTVOLUME(pod) $>$ thresh_migr_memory then
 6:         return TEMPORAL_SCORING(iw, ci, pw)
 7:     else
 8:         $spatial\_result \leftarrow$ SPATIAL_SCORING(pod, nodes, emb_emis_data, thresh_tns)
 9:         if $spatial\_result =$ 'spatial_not_possible' then
10:             return TEMPORAL_SCORING(iw, ci, pw)
11:         else
12:             return $spatial\_result$
13:         end if
14:     end if
15: end function
```

Figure 10. Pseudocode of the spatiotemporal scoring function.

1. **Function Definition:**
    - **function Spatiotemp_Scoring(...)**: The function **Spatiotemp_Scoring** is defined with several parameters: **pod** (the pod object to be scheduled), **thresh_migr_memory** (threshold for the pod's persistent volume to decide between temporal and spatial scoring), **nodes** (list of node objects for possible pod placement), **emb_emis_data** (data on embedded emissions for spatial scoring), **thresh_tns** (threshold for total node scoring), **iw**, **ci**, and **pwd** (parameters for temporal scoring).
2. **Persistent Volume Check:**
    - **if get_persistent_volume(pod) > thresh_migr_memory**: The function first checks if the persistent volume size of the pod exceeds the migration memory threshold. If it does, this suggests that migrating the pod may be too resource-intensive, and temporal scoring is preferred.
3. **Temporal Scoring Call:**
    - **return Temporal_Scoring(...)**: If the persistent volume is larger than the threshold, the function proceeds to call **Temporal_Scoring** with the immediate workload, carbon intensity, and workload prediction data, and then returns its result.
4. **Spatial Scoring Attempt:**
    - **else**: If the persistent volume is not larger than the threshold, the function attempts spatial scoring.

Funded by Horizon Europe Framework Programme of the European Union

- **spatial_result = Spatial_Scoring(...)**: The function calls **Spatial_Scoring** with the pod, list of nodes, embedded emissions data, and the TNS threshold to attempt to find a suitable node for the pod based on spatial factors.

5. **Spatial Result Check:**
   - **if spatial_result == 'spatial_not_possible'**: After attempting spatial scoring, the function checks if the result indicates that spatial placement is not possible.
6. **Fallback to Temporal Scoring:**
   - **return Temporal_Scoring(...)**: If spatial scoring is not possible, the function falls back to temporal scoring, calling the **Temporal_Scoring** function again with the necessary parameters and returning its result.
7. **Successful Spatial Scoring:**
   - **else**: If the spatial scoring is successful (i.e., it doesn't return 'spatial_not_possible'),
   - **return spatial_result**: The function returns the result of the spatial scoring, which presumably contains the selected node for the pod placement or some indication that spatial placement was successful.

Funded by Horizon Europe Framework Programme of the European Union

3 T6.2 COST-EFFECTIVE INFRASTRUCTURE (IBM)

Computing infrastructure represents an important investment that businesses and research institutions make to support the execution of their workloads of interest. However, the advent of cloud resources and the adoption of the cloud model down to edge devices moves the burden away from final users and towards the infrastructure providers. On the contrary IoT deployments remain mostly directly linked with the final applications and therefore a responsibility of the user. Regardless of the ownership, cost-effectiveness and efficient usage of computational resources remains a matter of utmost importance.

Cost-effectiveness can be represented in various ways: increased energy efficiency (ops/W), reduced hardware acquisition, increased utilization of existing hardware and use of cheaper sources of energy. However, in the context of FLUIDOS, no single approach can be applied to all layers of the continuum, mostly because of the unique features and requirements that need to be considered at each level. In FLUIDOS we are exploring cost-effectiveness at three different layers of the continuum: data center, edge infrastructure and IoT devices. These three separated explorations aim at increasing the cost-effectiveness at a different level, but their combined utilization will provide an even stronger benefit to the overall continuum.

At the data-center level, IBM is exploring the novel concept of Composable Disaggregated Systems. Multiple cloud providers have publicly claimed their infrastructure is not fully utilized, with resource usage peaking at around 60% for CPU and memory [44]. At the same time, highly parallel accelerators are widely used in a range of applications (e.g., scientific computing and machine learning). These accelerators are often highly expensive devices in the order of tens of thousands of euros per device and are also specialized for specific types of computation. Equipping an entire datacenter with the right mix of accelerators is therefore a daunting task. A disaggregated infrastructure instead breaks the boundaries of the single server and connects devices in pools over a high-speed interconnect fabric. Servers can be dynamically composed out of pooled resources, enabling higher resource utilization and the composition of resources tailored to incoming workloads.

At the Edge infrastructure level, TOPIX is considering cost-effectiveness of the networking infrastructure. Computing efficiency can't be achieved without considering network efficiency, so telco operators still play an important role in this process. A cost-effective network infrastructure requires efficient devices, possibly powered by renewable energy sources, and a network topology which keeps traffic as local as possible, avoiding sending bits to large-scale data centers if not needed. This paradigm becomes pivotal when considering remote regions, where geography and orography raise traditional infrastructure costs thus decreasing efficiency (e.g mountain valleys): these are the scenarios where edge solutions based on solar energy, hydrogen and natural cooling may represent a key factor in achieving high-performance and cost-effective IT services, thus empowering local facilities and reducing digital divide.

Funded by Horizon Europe
Framework Programme of the European Union

STM is considering the introduction of a cost/power effective IoT edge computing infrastructure. The IoT Edge computing infrastructure compared to the traditional centralized Cloud approaches, will implement distributed computations across a variety of heterogeneous tiny IoT devices connected to several sensors, cameras, etc. In order to implement a cost-effective solution, IoT edge infrastructure brings computing capability closer to data sources. In addition, the paradigm introduces hierarchical layers of computing infrastructure enabling to move/share computation and services toward the far edge. Using this proposed approach the IoT infrastructure will reduce the global power footprint while lowering the implementation cost.

3.1 COMPOSABLE DISAGGREGATED INFRASTRUCTURE

3.1.1 State of the art and overview

The advantages offered by composability have been explored over the past decade both in software-based solutions as well as at a lower hardware level [45][46]. The level of support for composability however has varied, mostly concentrating on providing flexibility for connecting I/O subsystems (the lower-hanging fruit) all the way to sharing memory (the most difficult target). Solutions have varied from the composability of resources in external shared enclosures through sharing of resources in the participating servers themselves all the way to an all-or-nothing approach, where participating systems share everything.

One of the key advantages of a composable systems architecture is the ability to reduce stranded resources and improve the utilization of resources overall. Stranded resources occur within a server when the ratio of deployed resources does not match the ratio allocated. For example, if customers are primarily requesting larger memory instances, then the deployed nodes will have the memory fully allocated while the CPUs remain underutilized.

To make things worse, actual utilization often is substantially lower than the initial resource allocation because users must provision for peak utilization and/or overestimate their resource consumption to avoid crashes due to process eviction. On average, CPU, and memory utilization in a published Google cluster usage dataset [44] never exceeds 60% and 50% respectively, even though more than 100% of the CPUs (over committed) and 80% of the memory are allocated. A random sampling of servers in different data centers shows an average CPU and memory utilization of 17.76% and 77.93%, respectively [47]. In a more recent paper based on data from Alibaba [48], the authors note – in line with the earlier findings – that their average resource utilization is only 38% for CPU and 88% for memory. While utilization can be improved with advancements in workload schedulers, the fixed server configurations at the heart of current data center architectures will still limit the overall resource utilization possible. Unallocated resources present both a CAPEX (capital expense) and OPEX (operating expense, e.g, space and power), which do not generate income for the data center operator when they are mostly idle.

Today, the most advanced approach to hardware composability employs PCIe switching – the industry pacesetter was Dolphin which introduced external PCIe (Gen1) switches all the way back in 2006.

Funded by Horizon Europe Framework Programme of the European Union

Complete solutions are offered by Liqid [49] and GigaIO [50], followed by more specialized solutions from Dolphin [51] and H3 Platform [52]. Composability of memory has been to date mostly an academic exercise with some relevant prototypes from industrial partners [53]–[58]. However, the advent of the Compute Express Link (CXL) standard [59], supported starting from PCIe gen 5, will enable load/store memory semantics on a PCIe fabric and support combined I/O and memory composition, leading the way towards full composable systems.

3.1.2 Activities in FLUIDOS

In the context of FLUIDOS, IBM is working on performing a cost-effectiveness analysis of a composable GPU-based system, where GPUs can be dynamically attached to hosts depending on workload needs. For a proper evaluation of cost-effectiveness to be carried out, we have identified two important milestones that have to be reached first: 1) the availability of real composable hardware that can be used for performing realistic performance comparisons on the selected applications, and 2) a software framework that enables automating the composition of resources and therefore the testing activities.

IBM will be using pre-owned composable hardware from third party vendors such as Liqid [49] and GigaIO [50] to perform all the experiments required for this task. This infrastructure enables the dynamic composition of PCIe (gen4) devices and in the context of the FLUIDOS exploration, it will be used to compose GPUs. Future results will be reported as necessary in the upcoming deliverables. However, no access to the infrastructure will be possible to any partner in the FLUIDOS consortium. On the software side, IBM is co-chairing an effort of the OpenFabrics alliance called Sunfish [60], aimed at defining an open and universal framework for the management of disaggregated composable infrastructure (see Figure 11).

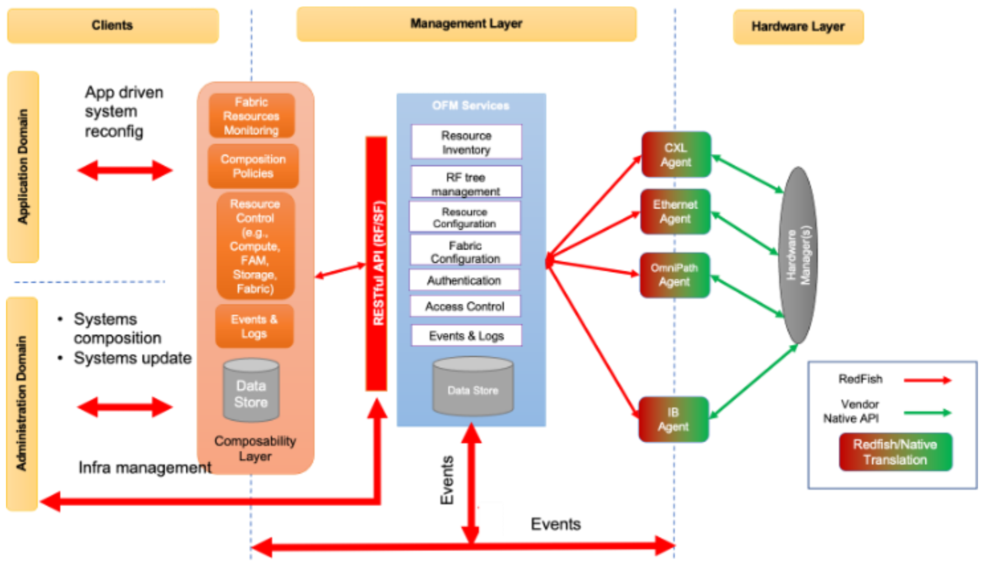

Figure 11. Sunfish framework architectural diagram.

Funded by Horizon Europe Framework Programme of the European Union

Sunfish adopts the DMTF Redfish specification [61] as the standard schema for its APIs. Redfish is a well adopted object schema used in the management of hardware and complex systems. The main idea behind Sunfish is that of being hardware vendor agnostic. Users, also known as clients (see Fig. 15), interact with Sunfish using a fully Redfish compliant RESTful API schema. On the hardware side, Sunfish hardware agents convert vendor specific APIs to Redfish and connect to the Sunfish core components. Client requests are processed and forwarded to the relevant hardware via the said agents. Clients rely on the same API regardless of the hardware connected to their system.

A composition layer is defined and used for hosting resources control and composition policies. This allows clients to specify systems via a set of requirements (e.g., CPU model, GPU model and amount of memory) and constraints (e.g., memory latency lower than a specific value). The composition policies are in charge of processing the client's requirement and identifying the right hardware that fulfills the client's request. The Sunfish framework is an open-source project hosted in the OpenFabrics Management Framework GitHub Organization[1].

The integration of Sunfish with Kubernetes will enable extensive automated testing of cloud-based applications on composable infrastructure, and therefore the collection of the data necessary for performing a cost-effectiveness study based on composable GPUs. IBM is leading the development effort of the Sunfish framework and is working on finalizing the required support for enabling the FLUIDOS exploration activities.

3.1.3 Applications of interest

Artificial Intelligence is undoubtedly the most important class of workloads, against which all cloud and infrastructure providers are (at the time of writing) trying to optimize their infrastructure for, with the goal of increasing the competitiveness of the services they offer.

AI applications are usually divided into two families: 1) training and fine-tuning of models; 2) model inference service. Training and fine-tuning refer to the set of steps that are performed for "building" a model (represented by a set of so-called weights) using large input datasets used for learning a specific task. This activity, especially with the latest Large Language Models (LLM), requires a specially built infrastructure with thousands of GPUs and fast networks. When training an LLM, one single task can also occupy an entire cluster and use 4000+ GPUs in unison. Inferencing instead, is a much lighter task with only a few GPUs (1 to 4) required for serving a model. Inference is the class of applications that we have selected for this activity.

[1] https://github.com/OFMFWG

Funded by Horizon Europe Framework Programme of the European Union

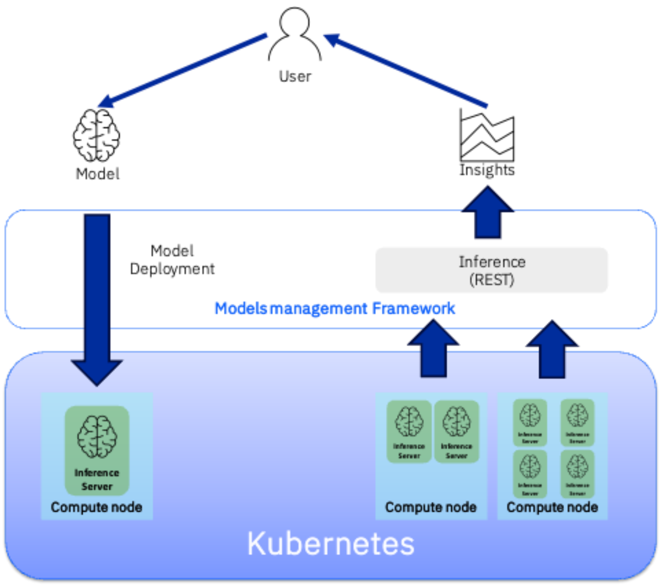

Figure 12. AI model inferencing system.

In a cloud-based inferencing system (see Figure 12), users deploy their models for inference through a models management framework such as Caikit [62]. Model weights are downloaded either from private repositories or open ones such as HuggingFace [63], and deployed on Kubernetes compute nodes ready for serving inference requests. Usually an inference server, such as the Text Generation Inference server [2], exposes a RESTful interface through which users can send inference requests.

The characteristics to be considered when deploying models in such a system are the number and model of GPUs required for each model, the amount of memory and CPU cores the inference server requires for performing requests queuing, batching, etc. In general, it is possible to serve multiple models from the same worker node depending on the model requirements and available resources. In a production environment the mix of different models, relying on different GPU models and numbers, and different amounts of CPU resources create fragmentation and lead to jailed resources and inefficient utilization of the infrastructure. The goal of this exploration is that of using a composable infrastructure for matching the resources required by each model maximizing the utilization of hardware resources, and therefore making the system more cost-effective. Exploratory results will be provided in the next release of this deliverable.

Table 2 summarizes a set of AI models of interest from the IBM Watsonx [64] platform that will be considered for the evaluation. All the models listed are openly available on HuggingFace.

[2] Text Generation Inference Server: https://github.com/IBM/text-generation-inference

Funded by Horizon Europe Framework Programme of the European Union

Table 2. Selected AI models of interest.

Model name	Tasks
google/flan-t5-xxl	message classification, questions answering, text summarization
google/flan-ul2	message classification, questions answering, text summarization
EleutherAI/gpt-neox-20b	message classification, text generation, text summarization
bigcode/starcoder	code generation

3.2 Infrastructure at the Edge (TOPIX)

3.2.1 Introduction

In the scope of defining an energy- and carbon-aware computation model for FLUIDOS that can shift loads both in time and geography, a cost-effective infrastructure is needed as a key resource to enable this behavior.

Creating affordable infrastructure is currently a fundamental challenge for IT managers. Achieving computing efficiency necessitates attention to network efficiency, making the role of telecommunications operators crucial. An economical network setup demands the use of effective devices that ideally utilize renewable energy and a network design that minimizes long-distance data transmission, keeping traffic localized and bypassing large data centers when unnecessary. This approach is especially critical in remote areas where the terrain and topography, such as in mountainous regions, can inflate the cost of conventional infrastructure and reduce its efficiency. In

Funded by Horizon Europe Framework Programme of the European Union

such contexts, edge computing solutions that leverage solar power, hydrogen fuel, and natural cooling techniques become essential. They are key to delivering high-performance and cost-efficient IT services, enhancing local resources, and narrowing the digital divide.

3.2.2 Scope

The escalating global demand for clean and sustainable energy sources has triggered a profound exploration of alternatives. Renewable energy has garnered significant attention for its potential to combat climate change and reduce our dependence on fossil fuels. Among the array of renewable options, hydrogen has emerged as an exceptionally versatile and promising solution with applications spanning multiple sectors. One sector poised to benefit from hydrogen integration is the realm of Edge data centers. These facilities serve as critical hubs for data processing and storage, often in remote or off-grid locations.

Telecommunications networks, a cornerstone of our modern connected world, depend on these Edge Data Centers for seamless connectivity, especially in areas where traditional power infrastructure is scarce. Yet, ensuring a continuous and reliable power supply to these facilities presents a formidable challenge. In this context, this document delves into the application of hydrogen in renewable energy scenarios specifically tailored to the needs of Edge Data Centers. The primary objective is to showcase the substantial potential of hydrogen in addressing the energy supply challenges faced by Data Centers and promoting sustainable, efficient telecommunication infrastructure.

The document's core purpose is to provide a brief analysis of hydrogen's role in renewable energy applications within the context of Edge Data Centers. It examines three pivotal aspects of hydrogen utilization: production, storage, and application. Through a thorough exploration of these dimensions, the document seeks to illustrate how hydrogen-based systems can offer a dependable and resilient power supply to Edge Data Centers, even in the most remote or off-grid locations. Furthermore, it elucidates the advantages and challenges associated with hydrogen as an energy source, offering valuable insights into its prospects within the dynamic and ever-evolving landscape of Edge Data Centers.

Technical survey

Having an energy-efficient network and IT infrastructure at the edge, which involves decentralized computing and data processing closer to end-users or devices, presents a set of unique challenges and considerations. Edge computing, while advantageous for reduced latency and improved performance, comes with the following challenges related to energy efficiency:

- Limited Resources: Edge locations, such as remote industrial sites or cell towers, may have limited access to resources like electricity and cooling infrastructure. This can make it challenging to design and operate energy-efficient IT systems.
- Harsh Environmental Conditions: Edge locations are often exposed to harsh environmental conditions, including extreme temperatures, humidity, and dust. These conditions can impact the efficiency and reliability of IT equipment.

Funded by Horizon Europe Framework Programme of the European Union

- Energy Sources: Access to clean and reliable energy sources, especially renewable energy, may be limited in certain edge locations. Relying on conventional grid power, which may be fossil-fuel-based, can hinder energy efficiency efforts.
- Cooling and Heat Management: Managing the temperature and cooling of IT equipment in challenging environments is a significant concern. Traditional cooling methods can be energy-intensive, affecting overall efficiency.
- Scalability: Edge facilities need to be scalable to accommodate increasing workloads and data processing demands. Ensuring scalability while maintaining energy efficiency is a complex task.
- Maintenance and Remote Management: Edge locations are often remote and may not have on-site IT staff. This makes the management, maintenance, and monitoring of IT infrastructure more challenging, potentially affecting efficiency.
- Connectivity and Latency: Low-latency connectivity is a fundamental requirement for edge computing, which can sometimes conflict with the goal of energy efficiency. Ensuring both low-latency and energy-efficient operations can be a balancing act.
- Security and Compliance: Edge data centers and IT systems may need to comply with security and regulatory requirements, which can add complexity and affect the choice of energy-efficient solutions.
- Integration with Centralized Infrastructure: Edge IT infrastructure needs to seamlessly integrate with centralized data centers and cloud services. Ensuring compatibility while maintaining efficiency is crucial.
- Resource Optimization: Making the most of available resources, such as optimizing workloads, data transfer protocols, and content delivery, is essential for energy efficiency at the edge.

Addressing these challenges requires a combination of innovative solutions, best practices, and careful planning. Energy-efficient server hardware, power management, and cooling solutions designed for harsh environments must be employed. Renewable energy sources, such as solar, wind, and/or hydrogen (H2), should be integrated where feasible to reduce reliance on fossil fuels. Customized IT infrastructure and data center designs that account for space constraints and environmental conditions are necessary. Energy monitoring and management systems should be implemented to optimize resource usage and reduce waste. Strategies that effectively integrate edge facilities with centralized data centers and cloud services should be developed to enable efficient data processing and management. Additionally, enhancing the resilience of edge infrastructure through redundancy, backup power, and disaster recovery planning is crucial.

By addressing these challenges and adopting energy-efficient practices, we can effectively leverage edge computing while minimizing its environmental impact and operational costs.

Funded by Horizon Europe Framework Programme of the European Union

3.2.3 Proposed solution

Exploring renewable energy solutions for the specific needs of Edge Data Centers involves careful consideration of various factors and strategies.

In the technical evaluation, several critical aspects are considered to ensure the resilience and future readiness of the solutions:

- Energy Demand Assessment: The starting point for assessing renewable energy solutions is to understand the energy requirements of the telecommunication infrastructure, including base stations, data centers, and communication equipment. Analyzing power needs, load profiles, and energy consumption patterns is essential to determine suitable renewable energy solutions.
- Renewable Energy Sources: Several energy sources are commonly used in telecommunication sites. Solar energy, often harnessed through solar photovoltaic (PV) systems, is widely adopted. It's crucial to evaluate factors like available space, solar irradiation levels, system sizing, and integration with existing infrastructure. Wind energy is considered in regions with favorable wind resources, while hydrogen serves as an energy carrier for transport.

The proposed solution for this project focuses on reducing the carbon footprint by combining multiple renewable energy sources, such as solar and hydrogen, in hybrid systems. These systems offer a reliable and consistent energy supply, supported by energy storage solutions like batteries or hydrogen storage. Efficient energy management and monitoring systems play a crucial role in optimizing renewable energy utilization. These systems may include smart grid technologies, remote monitoring, predictive analytics, and real-time energy performance monitoring.

Scalability and modularity are essential considerations, given the diverse nature of the sites where the Data Center will be deployed. Evaluating solutions that can accommodate future growth and adapt to changing energy needs is important. Resilience and backup power capabilities are assessed to ensure continuous operation during adverse conditions or grid outages. Backup power options like fuel cells may be considered to enhance reliability.

Staying updated on technological advancements in the renewable energy sector is crucial during the implementation phase. This involves researching innovative technologies, such as solar tracking systems, high-efficiency PV panels, energy-efficient cooling solutions, and energy storage advancements. Lastly, compliance with relevant standards and regulations is essential to ensure that the selected renewable energy solutions align with industry best practices. Certifications such as ISO 9001 for quality management and ISO 14001 for environmental management may be considered to verify compliance.

In summary, the evaluation process for implementing renewable energy solutions for Edge Data Centers encompasses various factors, including energy demand assessment, the choice of renewable energy sources, hybrid systems, efficient management, scalability, resilience, technological

Funded by Horizon Europe Framework Programme of the European Union

advancements, and compliance with industry standards. These considerations aim to create sustainable and reliable energy solutions for Edge Data Centers.

3.2.4 Implementation

The project idea, aimed at implementing a green TLC tower, stems from the observation, as mentioned earlier, that the installation of telecommunications towers in locations often situated in areas with unreliable electrical supply has, over the years, faced numerous challenges related to energy provision. Often, these challenges have led to the design of solutions based on renewable sources, primarily solar and wind power, and the integration of traditional generators of varying capacities, albeit always powered by fossil fuels. The ongoing shift towards ecological practices, amplified by the current historical context, increasingly calls for the adoption of entirely 'green' systems.

In the conducted experimental project, the decision was made to employ hydrogen (H2) as the most promising solution for energy requirements. This choice is bolstered by the versatility of global applications across various scenarios, particularly in the field of mobility. This shift towards H2 adoption has been facilitated by innovations in storage and dispensing methods, which can be attributed to:

- Gaseous storage (in contrast to previous liquid methods) at high pressures, ensuring increased density without introducing new management risks.
- Local production using renewable sources, in contrast to the current supply chain, which primarily involves H2 consumption and relies on traditional chains powered by petroleum and coal.

Furthermore, Fuel Cell technologies offer considerable modularity and can serve as an option in specific low-power scenarios, including compatibility with methanol.

To bring the ongoing design phase to completion, several key steps have been necessary:

1. Developing a reference scenario and aligning the solution with the reference macro-model.
2. Identifying and validating market products and comprehending associated costs.
3. Clarifying all aspects related to adoption within the service context, which includes compliance with regulations, assessing space requirements, verifying management and maintenance costs, integrating automation systems for interventions (such as uninterrupted power supply "UPS"), implementing "run-time" monitoring, and establishing attainable autonomy.

The chosen solution for the project involves a continuity system capable of powering the TLC site through a hybrid system that incorporates renewable energy sources and the traditional electrical grid. The established TLC site maintains a continuous average power consumption of up to 3000W.

As a result, the selected design scenario encompasses:

Funded by Horizon Europe Framework Programme of the European Union

- Certification of a transmitting node powered by the electrical grid and an autonomous battery pack.
- Activation of an H2 UPS group, providing continuity in the event of electrical grid faults, recharging batteries when the minimum threshold is reached, ignition based on charging needs and complete automation.
- Remote monitoring with the potential for system remote control.
- Analysis of monthly H2 consumption and the number of interventions required over a six-month period.

Through these activities, particularly the latter, the aim is to validate the H2 solution and all the necessary processes to provide precise answers and address any doubts related to each phase of system utilization and management.

A subsequent project phase would entail expanding the solar park's capacity to enable autonomous and local H2 refueling. The integration of an electrolyzer into the production and storage process would necessitate an analysis of managing potential overproduction for recharging functions. This approach could lead to the recovery of excess components from the previous site configuration, such as downsizing the number of batteries, and an assessment of the viability of cogeneration for reusing the produced heat.

In conclusion, the validation of the hybrid energy supply system, which extensively utilizes renewable sources with an emphasis on energy savings, will enable the following:

1. Gaining familiarity with the technology.
2. Understanding the risks and limitations of the proposed innovative system.
3. Expanding its adoption, not limited to telecom services by addressing both the technological aspect (on-site production) and extending the model to other entities operating in the area. This expansion can potentially lead to the creation of energy communities that share H2 production resources and spaces, thereby broadening the scope of "green" power supply.
4. Determining how to replicate these results in other applications such as Edge Data Centers.

3.3 IoT COST/POWER-EFFECTIVE INFRASTRUCTURE (STM)

3.3.1 Scope

The Internet of Things (IoT) refers to the network of physical objects or "things" (referred as IoT Devices) embedded with sensors, software, and other technologies, which enables them to connect,

Funded by Horizon Europe Framework Programme of the European Union

trigger actions and exchange physical data with other IoT devices and systems over the internet. These connected devices can range from everyday household items such as refrigerators and thermostats to industrial machines in factories, creating a vast network where objects can communicate, collect, and exchange physical data. IoT technology enables various IoT applications and IoT services across different sectors, including smart homes, healthcare, transportation, agriculture, industrial automation, and more. Today IoT together with AI are transforming industries and society since they make it possible to automate routine activities while unlocking previously unattainable insights and functionality. IoT applications and cloud applications serve different purposes but can complement each other in various scenarios. In many practical implementations, IoT applications and cloud applications work together harmoniously. IoT devices collect data, which is then sent to the cloud for processing, analysis, and long-term storage. Users interact with the IoT devices through cloud-based interfaces, enabling remote access and management. This collaborative approach allows for efficient, scalable, and user-friendly IoT solutions. For such a reason we can refer to them as IoT cloud infrastructures that enable the management and processing of data in the context of the IoT.

Traditionally IoT and AI applications have traditionally been deployed in the centralized IoT cloud infrastructure because they require the computing resources of data centers to turn data into insights and action. In this centralized infrastructure, all data generated by IoT devices is sent to a central cloud server or data center for processing, storage, and analysis. Using these centralized infrastructures allows an easier way to manage and maintain the IoT solution since all data is stored in one location. In addition, it offers very simple scalability since it is easier to scale resources vertically (adding more storage and processing power) in a centralized cloud model. Furthermore, they provide a comprehensive data analysis as all data is in one location, making it easier to derive insights from large datasets. Two of the major drawbacks of the centralized approach is the latency introduced by the data transmission to a central server which might be critical in real-time applications and the Bandwidth Usage since all the sensor data is transmitted to a central location which consumes significant network bandwidth and power.

IoT Sensors have undergone extraordinary proliferation since the beginning of the 21st Century. Thanks to IoT, connected, smart sensors are all around us. In 2030 it is estimated that the billions of sensors on IoT devices could be responsible for 30% of Internet data traffic, thanks to the widespread deployment of 5G. This would significantly increase the carbon impact of IoT applications. As the number of applications grows, we need to reduce their dependence on power-hungry cloud computing. Addressing energy and cost efficiency in IoT cloud infrastructures is a way to improve environmental sustainability in terms of CO2 emissions while reducing the underline costs. IoT Edge computing provides a solution.

3.3.2 Survey

IoT Edge computing is a paradigm based on the communication, edge distributed computation and data exchange between physical IoT devices and applications [65], linking real life entities with the virtual world [66], [67]. This paradigm involves moving some portion of the storage and compute resources out of the data center, closer to where the data is being generated (such as sensors etc). In

Funded by Horizon Europe Framework Programme of the European Union

this way, applications and devices collect and analyze data on their own. This approach is gaining momentum as it offers meaningful innovation for applications as varied as consumer products, building management, predictive maintenance in the industry, autonomy for vehicles, and much more. It lowers energy consumption, better protects personal data, reduces latency, and allows decision autonomy at the point of use for increased control, learning, and intelligence. IoT Edge computing is also a distributed computing model that aims to reduce latency, improve power efficiency, and enhance the overall performance of applications and services by bringing computation and data handling closer to where they are needed. Some key advantages of the IoT edge computing paradigm are the Proximity to Data Sources, Low Latency, Bandwidth Optimization, flexible computing power, Improved Privacy and Security, decentralized architecture, low power computation.

The underlying architecture of the IoT Edge computing paradigm as described in D2.1 (see Figure 13) is a set of distributed IoT device nodes communicating with each other according to a hierarchical layer structure as shown in the figure below. According to Fremantle [68] IoT Device nodes located at the micro Edge are sensors, actuators or tags endowed with a computational unit and one or more network connections. They generally embed power efficient MCU, tiny GPU and TPU able to perform intelligent power efficient computations.

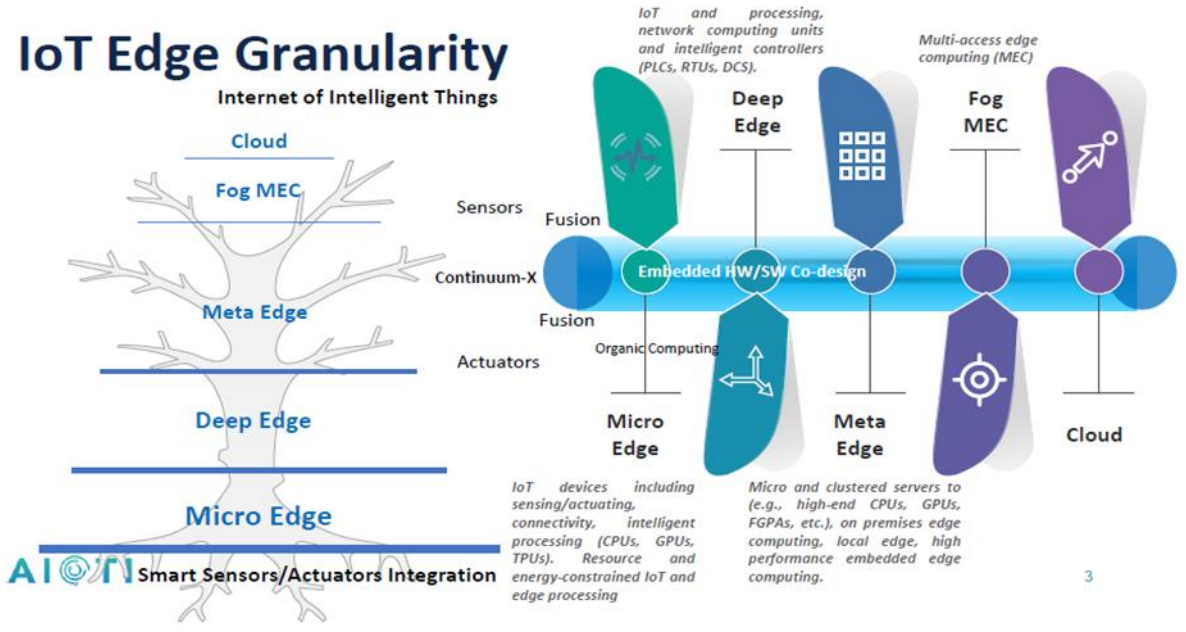
IoT Edge Granularity
Internet of Intelligent Things
Cloud
Fog MEC
Meta Edge
Deep Edge
Micro Edge
AIOTI
Smart Sensors/Actuators Integration
Sensors
Continuum-X
Actuators
Fusion
Fusion
Organic Computing
Embedded HW/SW Co-design
IoT and processing, network computing units and intelligent controllers (PLCs, RTUs, DCS).
Deep Edge
Multi-access edge computing (MEC)
Fog MEC
Micro Edge
Meta Edge
Cloud
IoT devices including sensing/actuating, connectivity, intelligent processing (CPUs, GPUs, TPUs). Resource and energy-constrained IoT and edge processing
Micro and clustered servers to (e.g., high-end CPUs, GPUs, FGPAs, etc.), on premises edge computing, local edge, high performance embedded edge computing.
3

Figure 13. IoT Edge granularity.

Moving up to the layering architecture we found IoT devices located at the Deep Edge layer that collect data and perform computation or data handling. They are intelligent controllers or network computing units such as gateways etc. The next layer is the Meta Edge. At this layer, we have devices that are micro and/or clustered servers installed on-premises. This layer provides the necessary performance to execute small scale cloud applications on-premises. The last layer is the cloud which handles vast

Funded by Horizon Europe Framework Programme of the European Union

amounts of data and performs complex processing and analysis. They can identify patterns, generate insights, and provide data visualizations. There are many ways in which edge-based computing is helping to make different industry sectors smarter and greener. Here are a few examples:

- Condition monitoring and predictive maintenance in factories can make operations smarter and more energy efficient. Sensors provide regular updates on the operating condition of machines to determine when they need servicing or if certain component parts need to be replaced. This reduces downtime and ensures machines run at optimal energy efficiency.
- Edge AI will also be critical for the next generation of Collaborative robots (cobots) designed to operate in real-time in the same workspace as humans to improve efficiency while ensuring their safety.
- Smart cities can use networks of millions of intelligent sensors and IoT nodes to improve monitoring, manage resources, assist citizens, and improve logistics with self-driven drones and vehicles.
- Bringing automation into the agricultural sector can help increase productivity and lower the environmental impact. Smart farm vehicles and machines will contribute to sustainability strategies by making it possible to use less water, fertilizer, and pesticides. Sensors coupled with edge AI allow the distribution of appropriate amounts of water or chemical substances to individual plants.

All these examples generate vast volumes of sensor data that would be highly energy and bandwidth-inefficient to send to the cloud for processing – as well as generating data protection and latency issues. Edge computing with AI provides a way to make them possible in a sustainable manner.

3.3.3 Proposed infrastructure solution

The proposed solution for this project focuses on reducing the carbon footprint of IoT systems by deploying a multilayered power efficient IoT Edge computing with Far Edge (see Figure 14) AI computation capabilities.

Funded by Horizon Europe
Framework Programme of the European Union

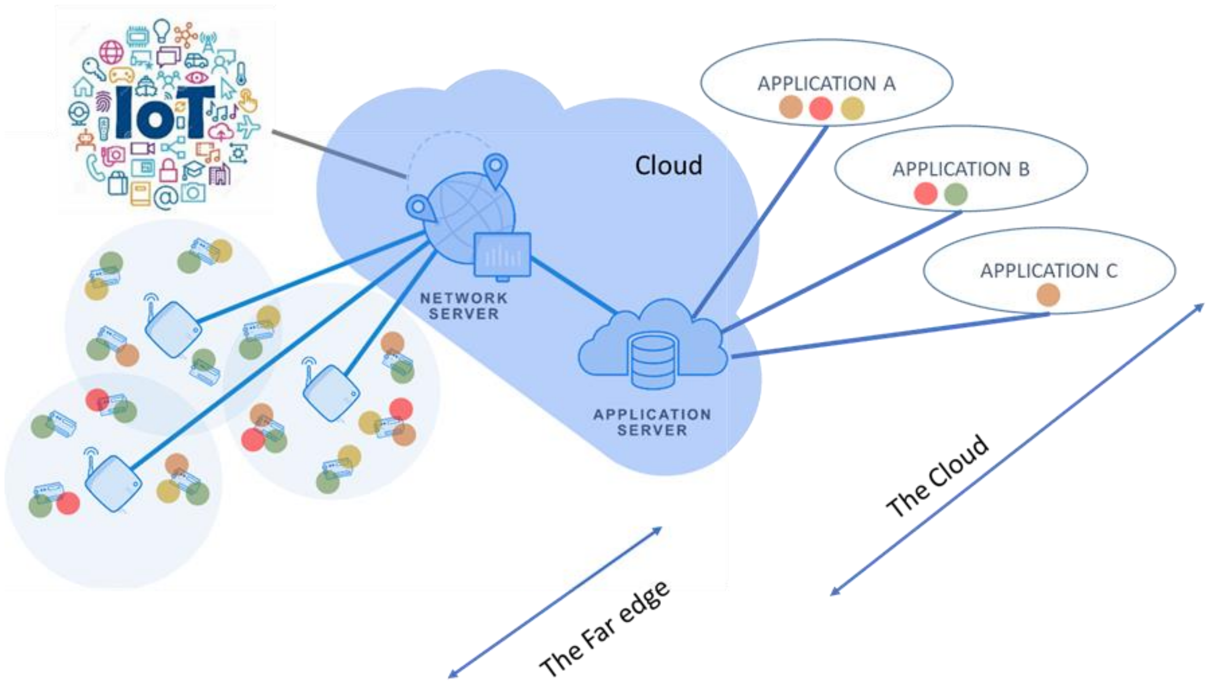

Figure 14. The Far Edge vs the Cloud.

One of the major advantages when implementing use cases following the IoT Edge Computing Paradigm with far-edge AI capabilities is to reduce power consumption by moving cloud services and resources toward the far edge of the network. The approach will reuse the FLUIDOS variant of KubeEdge framework together with the latest generation of IoT devices based on STM32 (see Figure 15). Using this variant of KubeEdge it is possible to further reduce the power consumption of the IoT infrastructure since the IoT devices will be seen as IoT service providers. In other words, several applications running in the cloud can have shared access to the services of data exposed by the IoT devices as illustrated in the figure above. This approach will create a continuum computing environment that spans from the cloud to the edge. As a matter of fact, KubeEdge leverages Kubernetes' container orchestration capabilities and extends them to the edge. It consists of a master node (like a Kubernetes master) running in the cloud and edge nodes at the edge of the network where we also have IoT device nodes based on low-power and low-cost micro-controllers. The master node called CloudCore manages edge nodes and ensures the synchronization of applications and data between the cloud and the micro edge where we will have plenty of sophisticated IoT device nodes.

Based on the latest technology advances, IoT device nodes are becoming very sophisticated offering many possibilities. These devices are becoming smarter and more powerful implementing many microservices that contribute to implementing power efficient distributed Applications compared to the more traditional cloud-centric approaches. The figure below illustrates one of these IoT device

Funded by Horizon Europe Framework Programme of the European Union

nodes implemented using the STM32 supporting many sensors. Using the physical data produced by the different sensors (LPS22HH, STTS751, etc.) the STM32 can perform some local computations which outcomes may be seen as microservices to be used by the application running in the cloud. The figure below also provides several examples of microservices such as step counters, baby crying, human activity recognition etc. It is important to point out that some of these services may be implemented using AI-based and ML-based models that are directly executed by the STM32 MCU or directly by the capable sensors instead of to the cloud.

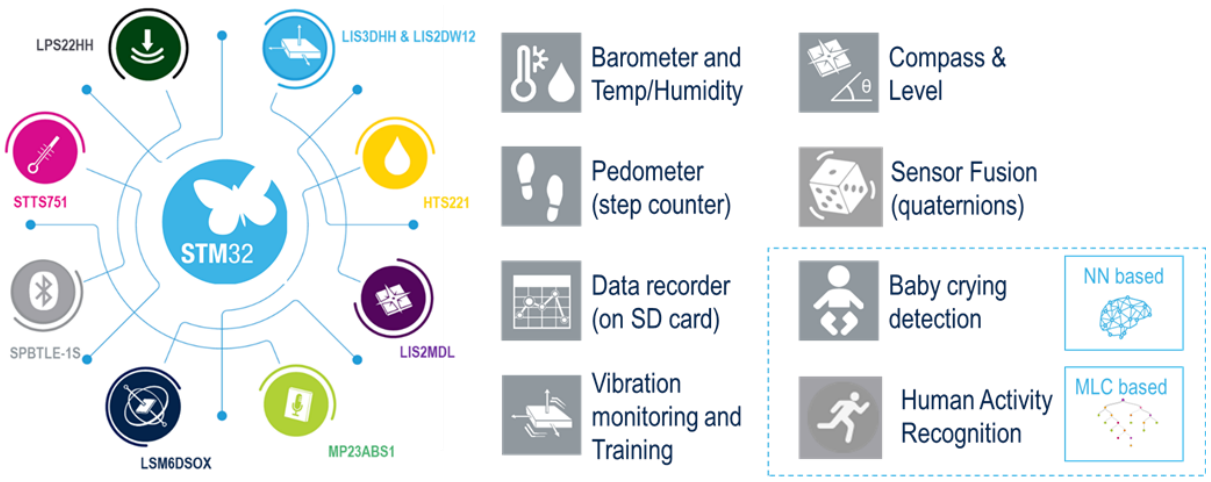

Figure 15. Local processing with STM32 and possible ST sensors.

Some of the key that features contribute toward reducing the carbon footprint that may be used to implement the proposed IoT Edge computing with Far-Edge computation capabilities are hereafter listed:

- Local Processing: By processing data locally on IoT device nodes, unnecessary data transmission to centralized cloud servers is minimized. Transmitting data over networks consumes power, especially in wireless communication. Local processing in IoT device nodes reduces the need for data transmission, saving power.
- Data Filtering and Aggregation: IoT device nodes can be programmed to filter and aggregate data before sending it to the cloud. This reduces the amount of data transmitted and, consequently, the power required for data transmission.
- Energy-Efficient Hardware: IoT device nodes can be designed with energy-efficient hardware components and processors such as the STM32. The STM32 MCU (Microcontrollers Units) family is well known for providing a range of low-power, low-cost MCUs suitable for IoT devices. These MCUs offer a combination of features and design considerations that help minimize energy consumption in IoT applications.
- Dynamic Power Management: IoT device nodes can implement dynamic power management strategies to adjust their power usage based on workload and activity levels. This can involve

Funded by Horizon Europe Framework Programme of the European Union

putting certain components to sleep when not in use or running at lower clock speeds during periods of low activity.

- Local Decision-Making: IoT device nodes can make real-time decisions locally, reducing the need to wake up a central server or cloud-based resource for every decision. This local decision-making capability can lead to lower power consumption for time-sensitive tasks.
- Optimized Communication Protocols: IoT device nodes can use optimized communication protocols that minimize the overhead associated with data transmission. These protocols are designed to be energy-efficient and can help reduce power consumption during communication with other devices or the cloud. One well-known example of these protocols is the LoRAWAN.
- Energy-Harvesting Solutions: Some IoT device nodes are equipped with energy-harvesting technologies such as solar panels or kinetic energy generators. These technologies allow devices to generate their own power, reducing their reliance on batteries or external power sources.
- Sleep Modes: IoT device nodes can be programmed to enter low-power sleep modes when not actively processing data or communicating. This helps extend the battery life of battery-powered devices.
- Device Management: Implementing effective device management and remote monitoring systems can help ensure that IoT device nodes are operating efficiently. This includes the ability to update firmware over the air, remote management of power states, and the possibility to detect and respond to anomalies that might lead to excessive power consumption.

In summary, all these considerations and analysis will offer a toolset for industry, automotive, health, agriculture, and many other sectors to continue to innovate while contributing to their decarbonization efforts thanks to a cost-power-effective solution for Cloud IoT infrastructure.

Funded by Horizon Europe
Framework Programme of the European Union

4 T6.3: AI FOR PERFORMANCE PREDICTION (FBK)

This chapter explores the capabilities, strengths, and weaknesses of AI-based and ML-based models for workload and energy prediction in FLUIDOS.

4.1 RELATED WORK

Carbon-aware and energy-aware computing are relatively new topics, with most of their development happening in the past decade. While prediction models empowering such philosophies have been successfully applied to various types of architectures recently, they have a long and extensive history in literature. In this section, we distinguish between two different types of prediction models: workload prediction and energy prediction.

4.1.1 Workload prediction

Workload prediction of computer systems has been an active area of research for decades. Early work focused on analyzing workload patterns to build predictive models. In 1996, Ntuen and Watson [69] proposed a general workload model based on system complexity and task difficulty. However, this work mostly focused on *modeling*, rather than predicting workload. Statistical models were indeed already being extensively used, owing to their flexibility, efficiency, and ease of use. Trend analysis and time series fitting had already been employed both in literature and practice, statically analyzing past data and making short-term predictions. However, only with the advent of machine learning did the literature on actual workload prediction begin to flourish.

We can categorize works that perform workload prediction depending on various aspects: for example, the type of approach, the environment, or the prediction timeframe. Speaking of the environment, one of the main settings found in literature is prediction for individual servers' workloads. For instance, [70] uses machine learning on historical data from a database service to predict server workload and detect overload. They found Random Forest most accurate, achieving 88.8% accuracy on synthetic data. [71] performed predictions on website data from the NASA and Saskatchewan web servers, achieving extremely high accuracy. However, with the advent of cloud computing, microservices, and virtualized workloads, performing workload prediction at the machine level started showing its limits.

Indeed, in recent years, most of the work in the literature has focused on workload prediction at the virtual machine level. Several works have been produced [72], most of them using ML techniques such as Long-Short Term Memory [73]–[75] and Multi-Layer Perceptrons [76].

Funded by Horizon Europe Framework Programme of the European Union

Finally, some other works still focus on more resource-constrained devices such as the ones at the Edge and for the Internet of Things, such as [77], [78], and [79]. Unfortunately, an underlying issue persists when operating in such a setting: the lack of datasets [80].

4.1.2 Energy prediction

A slightly different yet connected topic is the one concerned with energy prediction. While workload prediction focuses on predicting metrics such as CPU, memory, I/O usage and more, energy prediction is more concerned with the actual energy footprint of the device (or virtual machine, container, and more).

Back in the 1990s, the focus on energy efficiency was mostly on mobile devices and pocket computers. They often packed relatively energy-hungry processors but lacked both in battery capacity and energy management. Some studies such as [81] conducted mathematical studies on algorithms employed in mobile devices, such as pseudo-random number generators and sorting algorithms, verifying their energy efficiency.

While work on energy-constrained devices and, later, Internet-of-Things devices continued and prospered [82], [83], it soon became apparent that energy management was necessary even on power-hungrier machines. Some work started to be conducted on larger-scale architectures. For instance, [84], [85] tried to model the power consumption on a higher-scale, such as in data centers, and attempted to integrate it with virtual machine scheduling. [86] is a survey that summarizes the outcomes of the research on data center power modeling.

Finally, in the past few years, research has focused on carbon intensity prediction, which is strongly related to energy prediction. Indeed, as mentioned in section 6.1, several data sources provide real-time information about carbon intensity in countries and regions over the world. Thus, data centers are able to obtain precise estimations of their carbon footprint. Works such as [87] show how this information can be shown and manipulated at the application level, allowing software developers to manage metrics such as power usage, power discharging, and carbon intensity.

4.2 FORECASTING MODEL PROTOTYPE

A proof–of-concept Forecasting Model that feeds data back to the temporal scoring algorithm was developed in the scope of Task 6.3. The main purpose of the model is to predict the energy demand of a certain FLUIDOS node over a certain amount of time. Given a certain node, the model employs ML techniques and previously collected workload data and can provide a rough estimate of the expected energy consumption over the following day.

4.2.1 Datasets used and specifications

Funded by Horizon Europe Framework Programme of the European Union

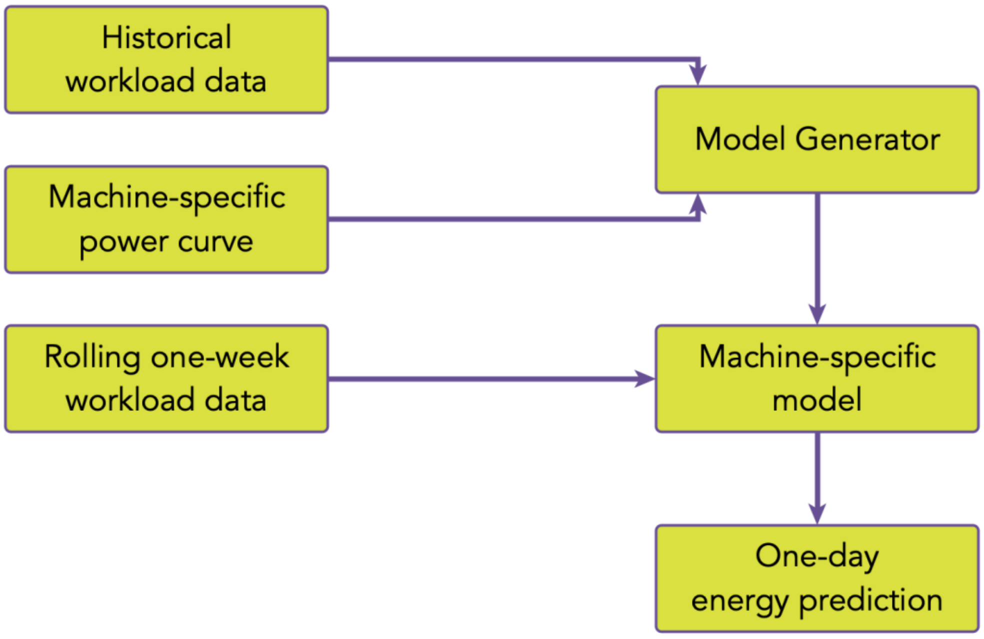
Historical workload data
Model Generator
Machine-specific power curve
Rolling one-week workload data
Machine-specific model
One-day energy prediction

Figure 16. Load prediction model workflow.

The proof-of-concept is written in Python version 3.10.4 and is platform-agnostic. The ML part of the model employs the popular Keras library. The way the model is implemented, predictions are hard to generalize and are specific to the very machine the model is being run on. Thus, no pre-trained files will be distributed; we expect each FLUIDOS node to run its copy of the model and run an initial training on its data. After a congruous amount of data collected, roughly at least two weeks from our experiments, the model will start providing accurate predictions.

For this proof-of-concept, we employed two different datasets to feed the model. The first is the Google Cluster Dataset [88], [89], a series of workload information collected from Google's Borg cluster deployments. This data includes a wide variety of data, ranging from simple and coarse CPU statistics to granular, per-container job information. From this dataset, we obtained rough CPU and memory usage averaged over a period of 15 minutes.

The second dataset is the Standard Performance Evaluation Corporation 2008 (SPECpower_ssj2008) benchmark. This dataset comprises thousands of results of the SPEC2008 benchmark, run on several different combinations of platform, CPU, memory, and more. We filtered and aggregated this data, obtaining some samples of possible power curves - functions that map resource consumption to energy consumption - to be used in the model.

The model is distributed in the form of a Docker image [90]. All information required for a successful deployment is available in the README of the aforelinked repository.

Funded by Horizon Europe
Framework Programme of the European Union

4.2.2 Model architecture

The model is composed of several sub-components, in particular:

- the data manipulation component,
- the learning component,
- the prediction component.

Additional components are provided in this proof-of-concept version, which are concerned with automating the parsing of the aforementioned datasets and the drawing of result graphs but are not fundamental to the inner workings of the application. Figure 16 shows the architecture of the model.

First, the data manipulation component is tasked with properly preparing the incoming workload data and power curves, transforming them into a form suitable for consumption from the learning and prediction components. In particular, the workload data is aggregated by machine ID, reordered, and a moving average is applied to reduce the amount of data points to one every 15 minutes. This is done both for CPU and memory data; the rest of the information is scrapped.

Then, sequences are prepared by further aggregating the data points over eight-day periods, of which the first seven are kept; using the power curves, the last day is instead transformed into a single number. This figure represents the amount of energy (in KWh) that the node consumed over that single day, and it will be used as a ground truth for the model. The seven days of data plus the energy consumption of the eighth day are then packed and saved together, to be used as a sequence by the other components. Further sequences are generated in the same manner, but by shifting the time window by a customizable amount, which defaults to an hour. With this approach, for example, nine days yield twenty-four different sequences to be used by the model.

The learning and prediction components are the key part of the application. The former takes in input a certain amount of data sequences (along with some user-provided parameters) and is tasked with training a deep learning model. In its current form, the ML model is a fully connected Convolutional Neural Network (CNN), composed of three convolutional layers and a final dense layer. As an input, the first layer takes 672 data points over two dimensions, i.e., seven days' worth of data every 15 minutes, both for CPU and memory usage. As an output, a dense layer condenses the model's information into a single value.

Funded by Horizon Europe
Framework Programme of the European Union

The hyperparameters for the model have been obtained using Keras' Tuner library, which automates the calculation and evaluation of hyperparameters of ML models. A summary of these layers is shown in *Figure 17*.

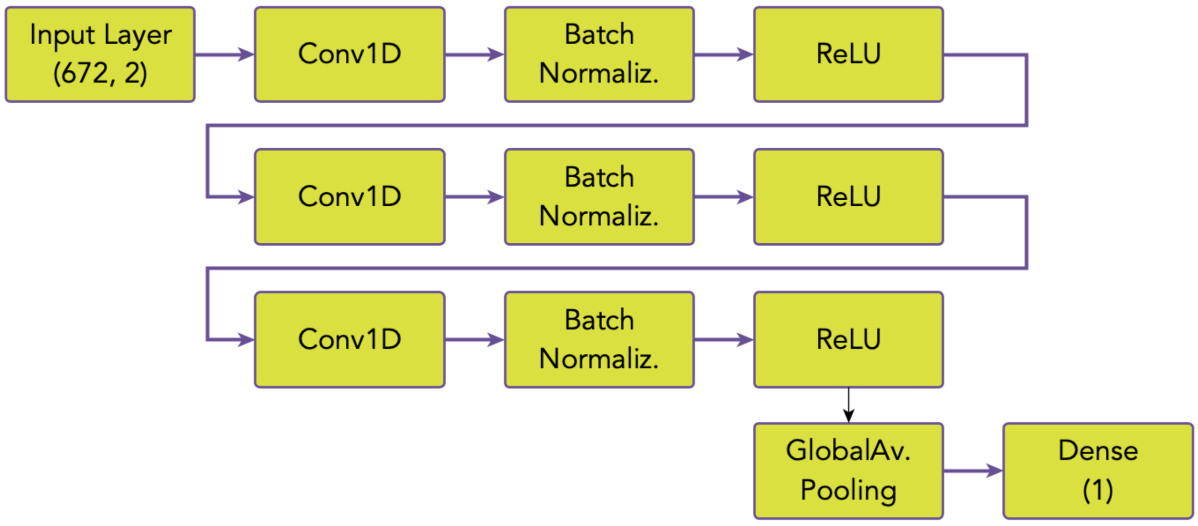

Figure 17. Layers of CNN used in the load prediction model.

Once created, the model is then trained using the Adam optimizer and Mean Squared Error as a loss function. Furthermore, several Keras callback functions aid in the gradient descent process, such as Early Stopping, Model Checkpointing, and more. In particular, the former halts the training phase when the loss function starts to diverge from the optimum, while the model checkpointing periodically saves the best model obtained so far, mitigating overfitting scenarios. Once a satisfactory result is obtained, the model is saved to disk and submitted to the prediction component. The system allows users to create a potentially unlimited number of models now, and additionally to retrain them with new data.

Finally, the prediction component takes into input a different set of workload data, a trained model, and queries the model. While it is important not to re-feed data used for training to the model — else the prediction would be a trivial task — the same power curve data must be maintained in order to obtain consistent results. Indeed, during our research, we found out that the current architecture does not well support training with data from a certain machine and predicting others. Indeed, each machine in the dataset probably had different specifications, usage, and therefore yielded different trends and statistics. After training the model on that particular machine, attempting to predict other machines' workload resulted in inconsistent if not wrong predictions at all.

With that in mind, the component outputs a single number, representing the energy (in KWh) the model estimates the machine will consume in the following day. It is then printed to the user, along with a collection of metrics that evaluate the performance of the model over that particular set of data. In particular, the users are presented with the R2 score (**coefficient of determination**), the mean squared error, and the mean absolute error.

Funded by Horizon Europe Framework Programme of the European Union

In a future production deployment, the prediction model will be running continuously, and a separate module will be tasked with fetching real-time performance metrics from FLUIDOS nodes and converting them to a suitable form for both the model generator and the predictor. Such data will be fed to a model that will output metrics, but then the results along with the previous metrics will be used to evaluate the model, eventually performing retraining to obtain better performance from it.

Funded by Horizon Europe
Framework Programme of the European Union

5 CONCLUSIONS

In this report, we have explored the challenging aspects of creating a sustainable computing infrastructure within the FLUIDOS project. Our focus began with the environmental impact of decentralized computing, where we recognized the importance of novel strategies for effective energy and carbon management. Among these, the implementation of spatial and temporal load shifting was highlighted as a key method for leveraging low-carbon electricity sources and optimizing energy use.

The report detailed aspects of carbon-aware computing, emphasizing the need to balance operational electricity consumption with the embodied footprint of computing devices. This balance was approached through Life Cycle Assessment (LCA), aiming to provide a more comprehensive understanding of environmental impacts.

Moreover, the economic feasibility of sustainable computing infrastructures was examined, pointing out the potential cost benefits of energy and carbon-aware strategies. This aspect underlined the possibility of wider adoption due to economic incentives.

Incorporating AI for performance prediction emerged as an intriguing solution, aligning technological advancements with environmental sustainability goals. This integration shows promise in enhancing resource efficiency while also addressing the concern of carbon emissions in computing infrastructures.

Looking ahead, the report sets specific goals to advance the field of sustainable computing. One such goal is the development and working demonstration of a carbon-aware load shifting algorithm. This demonstration aims to showcase the practical effectiveness of the algorithm in a lab setting and if possible also in real-world scenarios, providing tangible evidence of its benefits, in reducing the carbon footprint of computing infrastructures.

Another focal point for future work is the experimental assessment of the viability of disaggregated composable infrastructure. This research aims to explore the potential advantages and challenges of this approach, contributing to a deeper understanding of its impact on energy efficiency and overall sustainability in computing environments.

The report further identified the need for a detailed cost assessment of an off-grid edge data center. This assessment is crucial to determine the feasibility and economic implications of such infrastructures, which could significantly reduce costs while maintaining, or even enhancing, the sustainability of computing operations.

Funded by Horizon Europe Framework Programme of the European Union

REFERENCES

[1] R. Hischier, V. C. Coroama, D. Schien, and M. Ahmadi Achachlouei, “Grey Energy and Environmental Impacts of ICT Hardware,” in *ICT Innovations for Sustainability*, vol. 310, L. M. Hilty and B. Aebischer, Eds., in Advances in Intelligent Systems and Computing, vol. 310. , Cham: Springer International Publishing, 2015, pp. 171–189. doi: 10.1007/978-3-319-09228-7_10.

[2] C.-J. Wu *et al.*, “Sustainable AI: Environmental Implications, Challenges and Opportunities.” arXiv, Jan. 09, 2022. Accessed: Sep. 13, 2023. [Online]. Available: http://arxiv.org/abs/2111.00364

[3] P. Wiesner, I. Behnke, D. Scheinert, K. Gontarska, and L. Thamsen, “Let’s Wait Awhile: How Temporal Workload Shifting Can Reduce Carbon Emissions in the Cloud,” in *Proceedings of the 22nd International Middleware Conference*, Dec. 2021, pp. 260–272. doi: 10.1145/3464298.3493399.

[4] A. Radovanovic *et al.*, “Carbon-Aware Computing for Datacenters.” arXiv, Jun. 11, 2021. Accessed: Sep. 13, 2023. [Online]. Available: http://arxiv.org/abs/2106.11750

[5] B. Johnson, “Carbon-Aware Kubernetes,” Sustainable Software. Accessed: Sep. 15, 2023. [Online]. Available: https://devblogs.microsoft.com/sustainable-software/carbon-aware-kubernetes/

[6] A. James and D. Schien, “A Low Carbon Kubernetes Scheduler”.

[7] P. Burgherr, G. Heath, M. Lenzen, J. Nyboer, and A. Verbruggen, “William Moomaw (USA), Peter Burgherr (Switzerland), Garvin Heath (USA), Manfred Lenzen (Australia, Germany), John Nyboer (Canada), Aviel Verbruggen (Belgium)”.

[8] W. A. Hanafy, Q. Liang, N. Bashir, D. Irwin, and P. Shenoy, “CarbonScaler: Leveraging Cloud Workload Elasticity for Optimizing Carbon-Efficiency.” arXiv, Feb. 16, 2023. Accessed: Mar. 23, 2023. [Online]. Available: http://arxiv.org/abs/2302.08681

[9] B. Acun *et al.*, “Carbon Explorer: A Holistic Approach for Designing Carbon Aware Datacenters,” in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, Jan. 2023, pp. 118–132. doi: 10.1145/3575693.3575754.

[10] “Subramanian - Carbon-Aware Scheduling for Serverless Computing.pdf.”

[11] T. Piontek, K. Haghshenas, and M. Aiello, “Carbon emission-aware job scheduling for Kubernetes deployments,” *J. Supercomput.*, Jun. 2023, doi: 10.1007/s11227-023-05506-7.

[12] “KEDA,” KEDA. Accessed: Oct. 05, 2023. [Online]. Available: https://keda.sh/

[13] tomkerkhove, “Kubernetes Event-driven Autoscaling (KEDA) (Preview) - Azure Kubernetes Service.” Accessed: Oct. 05, 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/aks/keda-about

[14] R. Fairbanks, “Carbon aware spatial shifting of Kubernetes workloads using Karmada.” Accessed: Oct. 05, 2023. [Online]. Available: https://rossfairbanks.com/2023/07/12/carbon-aware-spatial-shifting-with-karmada/

[15] T. Sukprasert, A. Souza, N. Bashir, D. Irwin, and P. Shenoy, “Quantifying the Benefits of Carbon-Aware Temporal and Spatial Workload Shifting in the Cloud.” arXiv, Jun. 10, 2023. Accessed: Sep. 26, 2023. [Online]. Available: http://arxiv.org/abs/2306.06502

Funded by Horizon Europe Framework Programme of the European Union

[16] J. Koomey, S. Berard, M. Sanchez, and H. Wong, "Implications of Historical Trends in the Electrical Efficiency of Computing," *IEEE Ann. Hist. Comput.*, vol. 33, no. 3, pp. 46–54, Mar. 2011, doi: 10.1109/MAHC.2010.28.

[17] K. Lio, H. Alissa, and J. Sinistore, "Life Cycle Assessment for Cloud Hardware:".

[18] L. Deng and E. D. Williams, "Functionality Versus 'Typical Product' Measures of Technological Progress: A Case Study of Semiconductor Manufacturing," *J. Ind. Ecol.*, vol. 15, no. 1, pp. 108–121, Feb. 2011, doi: 10.1111/j.1530-9290.2010.00306.x.

[19] C. Mars, C. Nafe, and J. Linnell, "The Electronics Recycling Landscape Report".

[20] B. Whitehead, D. Andrews, A. Shah, and G. Maidment, "Assessing the environmental impact of data centres part 2: Building environmental assessment methods and life cycle assessment," *Build. Environ.*, vol. 93, pp. 395–405, Nov. 2015, doi: 10.1016/j.buildenv.2014.08.015.

[21] T. Makov, T. Fishman, M. R. Chertow, and V. Blass, "What Affects the Secondhand Value of Smartphones: Evidence from eBay," *J. Ind. Ecol.*, vol. 23, no. 3, pp. 549–559, Jun. 2019, doi: 10.1111/jiec.12806.

[22] "[EnEffLimICT]Grobe2022.pdf."

[23] L. M. Hilty and B. Aebischer, Eds., *ICT Innovations for Sustainability*, vol. 310. in Advances in Intelligent Systems and Computing, vol. 310. Cham: Springer International Publishing, 2015. doi: 10.1007/978-3-319-09228-7.

[24] E. Masanet, A. Shehabi, and J. Koomey, "Characteristics of low-carbon data centres," *Nat. Clim. Change*, vol. 3, no. 7, pp. 627–630, Jul. 2013, doi: 10.1038/nclimate1786.

[25] "What is carbon intensity? | National Grid Group." Accessed: Oct. 09, 2023. [Online]. Available: https://www.nationalgrid.com/stories/energy-explained/what-is-carbon-intensity

[26] "When to use Marginal Emissions (and when not to) | A New Shade of Green | Sherry Listgarten | Palo Alto Online |." Accessed: Oct. 09, 2023. [Online]. Available: https://www.paloaltoonline.com/blogs/p/2019/09/29/marginal-emissions-what-they-are-and-when-to-use-them

[27] "Watttime – The Power to Choose Clean Energy." Accessed: Mar. 06, 2023. [Online]. Available: https://www.watttime.org/

[28] "Electricity Maps | Live 24/7 CO_2 emissions of electricity consumption." Accessed: Mar. 29, 2023. [Online]. Available: https://app.electricitymaps.com/map

[29] "Marginal Emissions Methodology – Watttime." Accessed: Oct. 09, 2023. [Online]. Available: https://www.watttime.org/marginal-emissions-methodology/

[30] "ENTSO-E Transparency Platform." Accessed: Oct. 09, 2023. [Online]. Available: https://transparency.entsoe.eu/

[31] "Carbon Intensity." Accessed: Oct. 09, 2023. [Online]. Available: https://carbonintensity.org.uk/

[32] "Real-time Operating Grid - U.S. Energy Information Administration (EIA)." Accessed: Oct. 09, 2023. [Online]. Available: https://www.eia.gov/electricity/gridmonitor/index.php

[33] "Reinert et al. - 2023 - This is SpArta Rigorous Optimization of Regionall.pdf."

[34] "Kubernetes Carbon Intensity Exporter." Microsoft Azure, Sep. 27, 2023. Accessed: Oct. 05, 2023. [Online]. Available: https://github.com/Azure/kubernetes-carbon-intensity-exporter

[35] "Carbon Aware SDK." Green Software Foundation, Sep. 29, 2023. Accessed: Oct. 05, 2023. [Online]. Available: https://github.com/Green-Software-Foundation/carbon-aware-sdk

Funded by Horizon Europe
Framework Programme of the European Union

[36] R. Fairbanks, “Carbon aware temporal shifting of Kubernetes workloads using KEDA.” Accessed: Oct. 05, 2023. [Online]. Available: https://rossfairbanks.com/2023/06/05/carbon-aware-temporal-shifting-with-keda/

[37] S. Desrochers, C. Paradis, and V. M. Weaver, “A Validation of DRAM RAPL Power Measurements,” in *Proceedings of the Second International Symposium on Memory Systems*, in MEMSYS ’16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pp. 455–470. doi: 10.1145/2989081.2989088.

[38] “Kepler.” Sustainable Computing, Sep. 24, 2023. Accessed: Sep. 26, 2023. [Online]. Available: https://github.com/sustainable-computing-io/kepler

[39] “Exploring Kepler’s potentials: unveiling cloud application power consumption,” Cloud Native Computing Foundation. Accessed: Oct. 16, 2023. [Online]. Available: https://www.cncf.io/blog/2023/10/11/exploring-keplers-potentials-unveiling-cloud-application-power-consumption/

[40] “enhancements/keps/sig-scheduling/1819-scheduler-extender at 5320deb4834c05ad9fb491dcd361f952727ece3e · kubernetes/enhancements,” GitHub. Accessed: Sep. 20, 2023. [Online]. Available: https://github.com/kubernetes/enhancements/tree/5320deb4834c05ad9fb491dcd361f952727ece3e/keps/sig-scheduling/1819-scheduler-extender

[41] “client-go.” Kubernetes, Oct. 18, 2023. Accessed: Oct. 18, 2023. [Online]. Available: https://github.com/kubernetes/client-go

[42] “kubernetes-client/python: Official Python client library for kubernetes.” Accessed: Oct. 18, 2023. [Online]. Available: https://github.com/kubernetes-client/python

[43] “SPECpower_ssj2008 Results.” Accessed: Jul. 18, 2023. [Online]. Available: https://www.spec.org/power_ssj2008/results/power_ssj2008.html

[44] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, “Heterogeneity and dynamicity of clouds at scale: Google trace analysis,” in *Proceedings of the Third ACM Symposium on Cloud Computing*, San Jose California: ACM, Oct. 2012, pp. 1–13. doi: 10.1145/2391229.2391236.

[45] I.-H. Chung, B. Abali, and P. Crumley, “Towards a Composable Computer System,” in *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, Chiyoda Tokyo Japan: ACM, Jan. 2018, pp. 137–147. doi: 10.1145/3149457.3149466.

[46] B. Abali, R. J. Eickemeyer, H. Franke, C.-S. Li, and M. A. Taubenblatt, “Disaggregated and optically interconnected memory: when will it be cost effective?” arXiv, Mar. 03, 2015. Accessed: Oct. 16, 2023. [Online]. Available: http://arxiv.org/abs/1503.01416

[47] “IBM Research | Technical Paper Search | Data Centers in the Wild: A Large Performance Study(Search Reports).” Accessed: Oct. 16, 2023. [Online]. Available: https://dominoweb.draco.res.ibm.com/0c306b31cf0d3861852579e40045f17f.html

[48] J. Guo *et al.*, “Who limits the resource efficiency of my datacenter: an analysis of Alibaba datacenter traces,” in *Proceedings of the International Symposium on Quality of Service*, Phoenix Arizona: ACM, Jun. 2019, pp. 1–10. doi: 10.1145/3326285.3329074.

[49] “Liqid | Composable Infrastructure Software Platform.” Accessed: Oct. 16, 2023. [Online]. Available: https://www.liqid.com/

[50] “GigaIO | Rack-Scale Composable Infrastructure,” GigaIO. Accessed: Oct. 16, 2023. [Online].

Funded by Horizon Europe Framework Programme of the European Union

Available: https://gigaio.com/

[51] "PCI Express | High Speed Networks | Dolphin ICS." Accessed: Oct. 16, 2023. [Online]. Available: https://www.dolphinics.com/index.html

[52] "Memory Pooling CXL Expansion Chassis," H3 Platform. Accessed: Oct. 16, 2023. [Online]. Available: https://www.h3platform.com/

[53] C. Pinto *et al.*, "ThymesisFlow: A Software-Defined, HW/SW co-Designed Interconnect Stack for Rack-Scale Memory Disaggregation," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Athens, Greece: IEEE, Oct. 2020, pp. 868–880. doi: 10.1109/MICRO50266.2020.00075.

[54] H. Li *et al.*, "Pond: CXL-Based Memory Pooling Systems for Cloud Platforms," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, Vancouver BC Canada: ACM, Jan. 2023, pp. 574–587. doi: 10.1145/3575693.3578835.

[55] I. Calciu *et al.*, "Rethinking software runtimes for disaggregated memory," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, Virtual USA: ACM, Apr. 2021, pp. 79–92. doi: 10.1145/3445814.3446713.

[56] D. Masouros, C. Pinto, M. Gazzetti, S. Xydis, and D. Soudris, "Adrias: Interference-Aware Memory Orchestration for Disaggregated Cloud Infrastructures," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Montreal, QC, Canada: IEEE, Feb. 2023, pp. 855–869. doi: 10.1109/HPCA56546.2023.10070939.

[57] A. Patke *et al.*, "Evaluating Hardware Memory Disaggregation under Delay and Contention," in *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, Lyon, France: IEEE, May 2022, pp. 1221–1227. doi: 10.1109/IPDPSW55747.2022.00210.

[58] P. Koutsovasilis, M. Gazzetti, and C. Pinto, "A Holistic System Software Integration of Disaggregated Memory for Next-Generation Cloud Infrastructures," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, Melbourne, Australia: IEEE, May 2021, pp. 576–585. doi: 10.1109/CCGrid51090.2021.00067.

[59] "HOME," Compute Express Link. Accessed: Oct. 16, 2023. [Online]. Available: https://www.computeexpresslink.org

[60] "Sunfish - OpenFabrics Management Framework." Accessed: Oct. 16, 2023. [Online]. Available: https://www.openfabrics.org/openfabrics-management-framework/

[61] "REDFISH | DMTF." Accessed: Oct. 16, 2023. [Online]. Available: https://www.dmtf.org/standards/redfish

[62] "Caikit." Caikit, Sep. 28, 2023. Accessed: Oct. 17, 2023. [Online]. Available: https://github.com/caikit/caikit

[63] "Hugging Face – The AI community building the future." Accessed: Oct. 18, 2023. [Online]. Available: https://huggingface.co/

[64] "IBM watsonx - An AI and data platform built for business." Accessed: Oct. 19, 2023. [Online]. Available: https://www.ibm.com/watsonx

[65] T. Fan and Y. Chen, "A scheme of data management in the Internet of Things," in *2010 2nd IEEE InternationalConference on Network Infrastructure and Digital Content*, Beijing, China: IEEE, Sep. 2010, pp. 110–114. doi: 10.1109/ICNIDC.2010.5657908.

Funded by Horizon Europe Framework Programme of the European Union

[66] Y. Huang and G. Li, “A Semantic Analysis for Internet of Things,” in *2010 International Conference on Intelligent Computation Technology and Automation*, Changsha, China: IEEE, May 2010, pp. 336–339. doi: 10.1109/ICICTA.2010.73.

[67] D. Mazzei, G. Fantoni, G. Montelisciani, and G. Baldi, “Internet of Things for designing smart objects,” in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, Seoul, Korea (South): IEEE, Mar. 2014, pp. 293–297. doi: 10.1109/WF-IoT.2014.6803175.

[68] P. Fremantle, “A Reference Architecture for the Internet of Things,” 2015, doi: 10.13140/RG.2.2.20158.89922.

[69] C. A. Ntuen and A. R. Watson, “Workload prediction as a function of system complexity,” in *Proceedings Third Annual Symposium on Human Interaction with Complex Systems. HICS’96*, Aug. 1996, pp. 96–100. doi: 10.1109/HUICS.1996.549498.

[70] R. Cao, Z. Yu, T. Marbach, J. Li, G. Wang, and X. Liu, “Load Prediction for Data Centers Based on Database Service,” in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Jul. 2018, pp. 728–737. doi: 10.1109/COMPSAC.2018.00109.

[71] J. Kumar, R. Goomer, and A. K. Singh, “Long Short Term Memory Recurrent Neural Network (LSTM-RNN) Based Workload Forecasting Model For Cloud Datacenters,” *Procedia Comput. Sci.*, vol. 125, pp. 676–682, 2018, doi: 10.1016/j.procs.2017.12.087.

[72] M. Dayarathna, Y. Wen, and R. Fan, “Data Center Energy Consumption Modeling: A Survey,” *IEEE Commun. Surv. Tutor.*, vol. 18, no. 1, pp. 732–794, 2016, doi: 10.1109/COMST.2015.2481183.

[73] Y. S. Patel, R. Jaiswal, and R. Misra, “Deep learning-based multivariate resource utilization prediction for hotspots and coldspots mitigation in green cloud data centers,” *J. Supercomput.*, vol. 78, no. 4, pp. 5806–5855, Mar. 2022, doi: 10.1007/s11227-021-04107-6.

[74] E. Mouine, Y. Liu, J. Sun, M. Nayrolles, and M. Kalantari, “The Analysis of Time Series Forecasting on Resource Provision of Cloud-based Game Servers,” in *2021 IEEE International Conference on Big Data (Big Data)*, Dec. 2021, pp. 2381–2389. doi: 10.1109/BigData52589.2021.9671521.

[75] T. Wang, S. Ferlin, and M. Chiesa, “Predicting CPU usage for proactive autoscaling,” in *Proceedings of the 1st Workshop on Machine Learning and Systems*, Online United Kingdom: ACM, Apr. 2021, pp. 31–38. doi: 10.1145/3437984.3458831.

[76] J. J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, “Prediction of cloud data center networks loads using stochastic and neural models,” in *2011 6th International Conference on System of Systems Engineering*, Albuquerque, NM, USA: IEEE, Jun. 2011, pp. 276–281. doi: 10.1109/SYSOSE.2011.5966610.

[77] W. Miao *et al.*, “Workload Prediction in Edge Computing based on Graph Neural Network,” in *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, New York City, NY, USA: IEEE, Sep. 2021, pp. 1663–1666. doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00223.

[78] S. El Kafhali and K. Salah, “Efficient and dynamic scaling of fog nodes for IoT devices,” *J. Supercomput.*, vol. 73, no. 12, pp. 5261–5284, Dec. 2017, doi: 10.1007/s11227-017-2083-x.

[79] M. Aazam, M. St-Hilaire, C.-H. Lung, and I. Lambadaris, “PRE-Fog: IoT trace based probabilistic resource estimation at Fog,” in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Jan. 2016, pp. 12–17. doi: 10.1109/CCNC.2016.7444724.

Funded by Horizon Europe Framework Programme of the European Union

[80] O. Kolosov, G. Yadgar, S. Maheshwari, and E. Soljanin, “Benchmarking in The Dark: On the Absence of Comprehensive Edge Datasets”.

[81] R. Jain, D. Molnar, and Z. Ramzan, “Towards Understanding Algorithmic Factors Affecting Energy Consumption: Switching Complexity, Randomness, and Preliminary Experiments,” 2005.

[82] C. Bunse, H. Höpfner, E. Mansour, and S. Roychoudhury, “Exploring the Energy Consumption of Data Sorting Algorithms in Embedded and Mobile Environments,” in *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, May 2009, pp. 600–607. doi: 10.1109/MDM.2009.103.

[83] R. Jain, D. Molnar, and Z. Ramzan, “Towards a model of energy complexity for algorithms [mobile wireless applications],” in *IEEE Wireless Communications and Networking Conference, 2005*, Mar. 2005, pp. 1884-1890 Vol. 3. doi: 10.1109/WCNC.2005.1424799.

[84] S. Klingert, X. Hesselbach-Serra, M. P. Ortega, and G. Giuliani, Eds., *Energy-Efficient Data Centers*, vol. 8343. in Lecture Notes in Computer Science, vol. 8343. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. doi: 10.1007/978-3-642-55149-9.

[85] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing,” *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, May 2012, doi: 10.1016/j.future.2011.04.017.

[86] E.-V. Depasquale, F. Davoli, and H. Rajput, “Dynamics of Research into Modeling the Power Consumption of Virtual Entities Used in the Telco Cloud,” *Sensors*, vol. 23, no. 1, p. 255, Dec. 2022, doi: 10.3390/s23010255.

[87] A. Souza *et al.*, “Ecovisor: A Virtual Energy System for Carbon-Efficient Applications.” arXiv, Oct. 10, 2022. Accessed: Sep. 13, 2023. [Online]. Available: http://arxiv.org/abs/2210.04951

[88] Google, “google/cluster-data.” Google, Mar. 10, 2023. Accessed: Mar. 13, 2023. [Online]. Available: https://github.com/google/cluster-data

[89] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, “Large-scale cluster management at Google with Borg,” in *Proceedings of the Tenth European Conference on Computer Systems*, Bordeaux France: ACM, Apr. 2015, pp. 1–17. doi: 10.1145/2741948.2741964.

[90] M. Franzil, “risingfbk/fluidos-energy-predictor.” Accessed: Sep. 08, 2023. [Online]. Available: https://github.com/risingfbk/fluidos-energy-predictor

Funded by Horizon Europe Framework Programme of the European Union