A Cognitive Serverless Framework for the Cloud-Edge Continuum

D4.1 COGNIT Serverless Platform - Scientific Report - a

Version 1.0

31 October 2023

Abstract

COGNIT is an AI-enabled Adaptive Serverless Framework for the Cognitive Cloud-Edge Continuum that enables the seamless, transparent, and trustworthy integration of data processing resources from providers and on-premises data centers in the cloud-edge continuum, and their automatic and intelligent adaptation to optimise where and how data is processed according to application requirements, changes in application demands and behaviour, and the operation of the infrastructure in terms of the main environmental sustainability metrics. This document describes the research and development carried out in WP4 “AI-enabled Distributed Serverless Platform and Workload Orchestration” during the First Research & Innovation Cycle (M4-M9), providing details on the status of a number of key components of the COGNIT Framework (i.e. Cloud-Edge Manager and AI-Enabled Orchestrator) as well as reporting the work related to supporting Energy Efficiency Optimization in the Multi-Provider Cloud-Edge Continuum.

This project is funded by the European Union’s Horizon Europe research and innovation programme under Grant Agreement 101092711 – SovereignEdge.Cognit

Deliverable Metadata

Project Title:	A Cognitive Serverless Framework for the Cloud-Edge Continuum
Project Acronym:	SovereignEdge.Cognit
Call:	HORIZON-CL4-2022-DATA-01-02
Grant Agreement:	101092711
WP number and Title:	WP4. AI-enabled Distributed Serverless Platform and Workload Orchestration
Nature:	R: Report
Dissemination Level:	PU: Public
Version:	1.0
Contractual Date of Delivery:	30/09/2023
Actual Date of Delivery:	31/10/2023
Lead Author:	Monowar Bhuyan (UMU) & Paul Townend (UMU)
Authors:	Malik Bouhou (CETIC), Aritz Brosa (Ikerlan), Idoia de la Iglesia (Ikerlan), Sébastien Dupont (CETIC), Joan Iglesias (ACISA), Tomasz Korniluk (Phoenix), Johan Kristiansson (RISE), Antonio Lalaguna (ACISA), Ignacio M. Llorente (OpenNebula), Marco Mancini (OpenNebula), Alberto P. Martí (OpenNebula), Philippe Massonet (CETIC), Nikolaos Matskanis (CETIC), Daniel Olsson (RISE), Michał Opala (OpenNebula), Per-Olov Östberg (UMU), Goiuri Peralta (Ikerlan), Samuel Pérez (Ikerlan), Bruno Rodríguez (OpenNebula), Juan José Ruiz (ACISA), Kaja Swat (Phoenix), Thomas Ohlson Timoudas (RISE), Iván Valdés (Ikerlan), Constantino Vázquez (OpenNebula).
Status:	Submitted

Document History

Version	Issue Date	Status[1]	Content and changes
0.1	20/10/2023	Draft	Initial Draft
0.2	27/10/2023	Peer-Reviewed	Reviewed Draft
1.0	31/10/2023	Submitted	Final Version

Peer Review History

Version	Peer Review Date	Reviewed By
0.1	27/10/2023	Erik Elmroth (UMU)
0.1	27/10/2023	Marco Mancini (OpenNebula)

Summary of Changes from Previous Versions

First Version of Deliverable D4.1

[1] A deliverable can be in one of these stages: Draft, Peer-Reviewed, Submitted, and Approved.

Executive Summary

This is the first version of Deliverable D4.1, the COGNIT Serverless Platform Scientific Report, produced in WP4 “AI-enabled Distributed Serverless Platform and Workload Orchestration”. It describes in detail the progress of the software requirements that have been active during the First Research & Innovation Cycle (M4-M9) in connection with these main components of the COGNIT Framework:

Cloud-Edge Manager

- **SR4.3** Serverless Runtime Deployment:

  Deploy Serverless Runtime as Virtualized Workloads (e.g. Containers or VMs/microVMs) on the cloud-edge infrastructure.

- **SR4.4** Metrics, Monitoring, Auditing:

  Edge-Clusters monitoring, Serverless Runtimes metrics collection and continuous security assessment.

- **SR4.5** Authentication & Authorization:

  Authentication and authorization mechanisms for accessing cloud-edge infrastructure resources by the devices for offloading workloads.

AI-Enabled Orchestrator

- **SR5.1** Building Learning Model:

  Implement AI/ML model based on collected metrics from Edge Cluster entities and serverless runtimes deployed across the distributed cloud-edge continuum.

- **SR5.2** Smart Deployment of Serverless Runtimes:

  Implement a Smart Workload Orchestrator (SWO) that exposes a REST API used by the Cloud-Edge Manager for requesting the deployment plans used for provisioning the Serverless Runtimes.

- **[NEW] SR5.3** Scheduling Mechanisms:

  Implement a scheduler that will place the Serverless Runtimes on the Edge-Clusters resources according to the deployment plan provided by the AI-Enabled Orchestrator.

This deliverable has been released at the end of the First Research & Innovation Cycle (M9), and will be updated with incremental releases at the end of each research and innovation cycle (i.e. M15, M21, M27, M33).

Table of Contents

Abbreviations and Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
DaaS	Data as a Service
DB	Database
FaaS	Function as a Service
GPU	Graphics Processing Unit
HTTP	Hypertext Transfer Protocol
IAM	Identity and Access Management system
IP	Internet Protocol
IoT	Internet of Things
JSON	Javascript Object Notation
ML	Machine Learning
OS	Operating System
QoS	Quality of Service
REST	Representational State Transfer
S3	Simple Storage Service
SDK	Software Development Kit
SLA	Service Level Agreement
SQL	Structured Query Language
VM	Virtual Machine
YAML	Yaml Ain’t a markup language

1. Cloud-Edge Manager

The Cloud-Edge Manager is responsible for managing the cloud-edge continuum infrastructure and performing actions to manage the lifecycle of the different Serverless Runtimes, collecting their metrics and monitoring the infrastructure resources they use.

The main responsibilities of the Cloud-Edge Manager are, thus:

- Exposing through an API the operations for managing the cloud-edge continuum infrastructure (i.e. physical computational hosts, networks and storages across multi-cloud providers and edge locations) and managing the Serverless Runtimes, that are used by the AI-Enabled orchestrator to optimise the execution of the applications based on the requirements sent by the device.
- Monitoring both the cloud-edge infrastructure and the Serverless Runtimes to provide the AI-Enabled Orchestrator with information to implement automatic and intelligent adaptation for the placement of the Serverless Runtimes .
- Providing authentication and authorization mechanisms for accessing and securing resources such as physical hosts, virtual machines, networks, services, etc.

[SR4.3] Serverless Runtime Deployment

Description

The Serverless Runtime is the main management unit of the COGNIT Framework. It is defined by a document (a JSON file) that conveys all the information for its automatic deployment on the distributed cloud-edge continuum. The document containing the requirements is sent by the Device Client to the Provisioning Engine that communicates with the Cloud-Edge Manager.

The Cloud-Edge Manager functionality able to manage the lifecycle of Serverless Runtimes has been implemented using OpenNebula's OneFlow component[2]. The Provisioning Engine communicates with the Cloud-Edge Manager through the Oneflow API to create, read and delete Serverless Runtimes as required by the devices.

Architecture & Components

OneFlow allows users and administrators to define, execute and manage multi-tiered applications, called Services, composed of interconnected Virtual Machines with deployment dependencies between them, and to deploy and manage them as a single entity. Each group of VMs is called a Role, and can depend on other roles. The OneFlow component is able to implement elasticity policies; the service can be scaled up or down depending on the needs, in order to add or remove virtual machines from it. This existing OpenNebula open source component is leveraged to deploy Serverless Runtimes as a single entity, even if they are composed of more than one Virtual Machine.

[2] https://docs.opennebula.io/stable/installation_and_configuration/opennebula_services/oneflow.html

To successfully deploy a Serverless Runtime upon request, a new appliance for OpenNebula clouds has been created. This appliance consists of a Virtual Machine, based on openSUSE containing the Serverless Runtime software properly configured to run at start time. This appliance has been created under the name of “FaaS Runtime 7643f57fca5424dcec427af3077a76a5ccc27637”, referring to the particular commit the Serverless Runtime has been installed from.

Data Model

The Serverless Runtime consists of two main components:

1. A **Function as a Service** (FaaS) Runtime component that allows the execution of functions on resources of the cloud-edge continuum resources
2. A **Data as a Service** (DaaS) component that allows uploading data to the Serverless Runtime exploiting data locality. The DaaS can implement different protocols and backend storages (e.g. S3, SQL DB, etc.).

Serverless Runtime metadata is stored in two separate locations. One resides in the Cloud-Edge Manager, defined as OpenNebula OneFlow Services Templates (i.e. JSON files) where the two roles corresponding to the FaaS and DaaS components and their dependencies are defined. These Service Templates act as different flavours of the Services Runtimes; currently we have four different flavours in the COGNIT infrastructure corresponding to the four different Use Cases. These flavours contain different libraries that conform to the runtime of the functions that are called by the Device Client when the Serverless Runtime is running.

The other piece of metadata that defines a Serverless Runtime is provided by the Device Client when it requests the creation of a Serverless Runtime to the Provisioning Engine. Both pieces of metadata are then combined by the OneFlow component upon request by the Provisioning Engine.

The next segment shows a JSON object corresponding to the Service Template for the Serverless Runtime with the two components, FaaS and DaaS (although the DaaS component is not going to be deployed, note the cardinality set to 0). This JSON object is created by the Provisioning Engine and provided to the OneFlow component, requesting it to merge its contents (vm_template_contents) with the Service Template definition corresponding to the flavour requested by the Device Client to the Provisioning Engine (see Deliverable D3.1 for more information on the Provisioning Engine). All device information, scheduling policies, etc. are provided at the Role level, since the OpenNebula scheduler (and, therefore, the AI-Enabled Orchestrator) works at the Virtual Machine level for placement:

```
{
  "name": "Serverless Runtime",
  "deployment": "straight",
```

```
  "description": "",
  "roles": [
    {
      "name": "FAAS",
      "cardinality": 1,
      "elasticity_policies": [],
      "scheduled_policies": [],
      "vm_template_contents": "CPU=0.44
HOT_RESIZE=[CPU_HOT_ADD_ENABLED=\"YES\",
MEMORY_HOT_ADD_ENABLED=\"YES\"]
MEMORY_RESIZE_MODE=\"BALLOONING\"
DISK=[IMAGE_ID=\"5\",
SIZE=\"420\"]
VCPU=3
MEMORY=111
VCPU_MAX= \"6\"
MEMORY_MAX=\"222\"
SCHEDULING: {POLICY: \"energy\", REQUIREMENTS: \"FREECPU > 10\"}
DEVICE_INFO: {LATENCY_TO_PE: 100, GEOGRAPHIC_LOCATION: \"Madrid\"}"
    },
    {
      "name": "DAAS",
      "cardinality": 0,
      "elasticity_policies": [],
      "scheduled_policies": [],
      "vm_template_contents": ""
    }
  ],
}
```

JSON object defining a Service Template representing a Serverless Runtime

API & Interfaces

The OpenNebula OneFlow RESTful API is used by the Provisioning Engine to create, read, and delete the Serverless Runtimes as required by the devices.

Action	Verb	Endpoint	Response
Create a Serverless Runtime	POST	/service_template /<id>/action	Status code 201 (Created) : request was successful and a new resource has been created.
Get information about the Serverless Runtime	GET	/service/<id>	Status code 200 (OK) with a JSON representation of the Serverless Runtime in the HTTP body.
Delete a Serverless Runtime	DELETE	/service_template /<id>	Status code 204 (No content): the request has been accepted for processing, but no info in the response.

Table 1.1. API used by the Provisioning Engine to create/read/delete Serverless Runtimes

[SR4.4] Metrics, Monitoring, Auditing

Description

Metrics collected by monitoring systems provide valuable information on the operational efficiency, resource utilisation and sustainability of data centres and severless environments. The OpenNebula monitoring system[3] includes several metrics related to each compute node involved in the operations managed by an OpenNebula cloud, including the monitoring of the OpenNebula instance itself.

OpenNebula offers a native integration with the open source solution Prometheus[4] where, through its own exporters, it is able to gather all the information and metrics from each node of the infrastructure. However, these metrics lack data related to the energy consumption of the system processes.

The monitoring of the COGNIT Framework needs to be able to rely on energy consumption metrica to detect anomalies and improve energy efficiency.

Analysis

Some tools have been analysed to extract the energy consumption metrics from the Hosts and virtual environments. The more relevant open source tools found are the following:

- **PowerAPI**[5]: middleware toolkit for building software-defined power metres, focused on Host energy consumption with some extensions for VM energy consumption monitoring.
- **Scaphandre**[6]: metrology agent dedicated to energy power consumption metrics, more focused on cloud environments, supporting bare metal servers, VMs and Containers monitoring.

The major distinctions between the two tools can be found in the following table:

PowerAPI	Scaphandre
Data storage needs to be managed by the user since the tool only generates the metrics.	Database to store metrics is included and managed by the tool to store metrics over time.
More focused on general energy consumption metrics.	More focused on cloud environments metrics (VMs, Containers, Pods, bare-metal servers).
Requires more configuration to obtain data from virtualized processes.	Out of the box solution for Pods and Containers monitoring.

Table 1.2. Comparison of PowerAPI and Scaphandre

[3] docs.opennebula.io/6.6/installation_and_configuration/opennebula_services/monitoring.html
[4] docs.opennebula.io/6.6/management_and_operations/monitor_alert/overview.html
[5] github.com/powerapi-ng/powerapi
[6] github.com/hubblo-org/scaphandre

The advantages that Scaphandre offers are clearly evident. From the solutions already incorporated for monitoring processes in the cloud, the different metrics output options (*stdout*, *json*, *prometheus exporters*, etc.), as well as its clear focus on cloud and virtualized environments. This is why Scaphandre was selected as the tool to be used for the extraction of energy consumption metrics in COGNIT.

Architecture & Components

The energy consumption monitor system is composed of three main components, depicted in the High Level Architecture contained in the following figure:

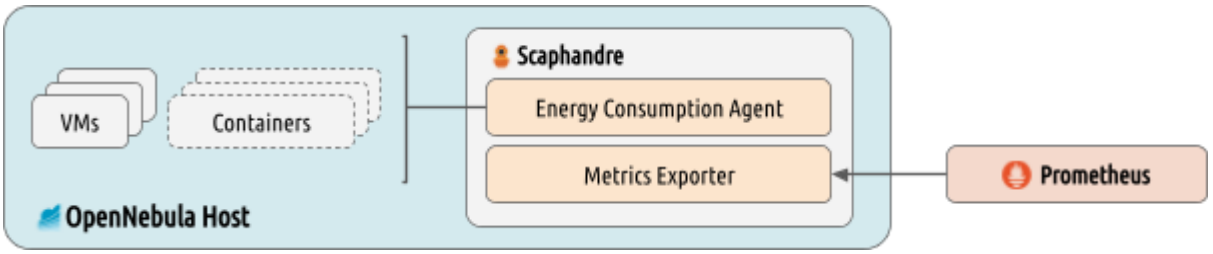

Figure 1.1. Scaphandre integration with OpenNebula's monitorisation workflow

This architecture is replicable in each OpenNebula Host and it is composed by the following modules:

- **OpenNebula Host**: includes all the technology needed in order to virtualize and orchestrate the cloud resources along all the monitorization probes and metrics for Host monitorization, including CPU and memory usage, network traffic, etc.
- **Scaphandre**: expands the OpenNebula native monitorization capabilities by also gathering energy consumption metrics from VMs, Containers, and Pods. Scaphandre uses their own agent to collect the energy metrics consumption, then uses a dedicated exporter in order to offer the metrics results to Prometheus.
- **Prometheus**: pulls the metrics from the Scaphandre exporter, integrating these metrics with the metrics collected from OpenNebula owns metrics generated by their exporters. This Prometheus instance is common for all the OpenNebula Host so only one Prometheus per deployment is needed.

Prometheus implements the HTTP pull model to gather metrics from the Scaphandre exporter. The endpoint used by Prometheus to gather metrics can be configured in its YAML file configuration (*prometheus.yml* by default).

Data Model

The most relevant metrics provided by the Scaphandre exporter[7] related to energy consumption are described in the following table:

Metric	Description
`scaph_host_power_microwatts`	Aggregation of several measurements to give a try on the power usage of the whole host, in microwatts.
`scaph_process_power_consumption_microwatts`	Power consumption due to the process, measured at the topology level, in microwatts. PROCESS_EXE being the name of the executable and PROCESS_PID being the pid of the process.

Table 1.3. Scaphandre energy consumption metrics

Scaphandre adds labels to these metrics to filter them by process, pods, VMs or containers. The available labels to filter by these objects are listed in the following table:

Metric	Description
`vmname`	Name of the VM in libvirt. Following the OpenNebula name rules, the VMs name will follow the form `one-<id>`.
`container_scheduler`	Identifies the container scheduler of the container. Possible values are docker or kubernetes.
`container_id`	The ID of the container got from `/proc/PID/cgroup`.

Table 1.4. Scaphandre labels

Additionally, the `host_id` label needs to be added in the Prometheus configuration file in order to filter by OpenNebula Host ID as shown below:

```
- job_name: 'scaphandre-onehost-X'
  static_configs:
    - targets: ['onehost-X:8080']
      labels:
        host_id: X
```

OpenNebula Host ID label in Prometheus configuration file

[7] hubblo-org.github.io/scaphandre-documentation/references/metrics.html

API & Interfaces

Metrics from the Scaphandre exporter may only be pulled from the OpenNebula Host when Prometheus scrapes them, the exporter should not perform scrapes based on their own timers, meaning all scrapes should be synchronous. The table below shows the endpoint exposed by the Scaphandre exporter:

Action	Verb	Endpoint	Request Body	Response
Gather metrics	GET	/metrics	-	Status code 200 (OK) with all the metrics collected by the exporter

Table 1.5. Scaphandre exporter endpoints

An example of the metrics returned by the exporter is shown below:

```
scaph_process_power_consumption_microwatts{vmname="one-3",host_id=”0"} 22807
scaph_process_power_consumption_microwatts{vmname="one-4",host_id=”0"} 38051
scaph_process_power_consumption_microwatts{vmname="one-2",host_id=”1"} 15910
```

Scaphandre exporter metrics output

[SR4.5] Authentication & Authorization

Description

Currently, authentication across the whole COGNIT Framework is delegated through the Cloud-Edge Manager. OpenNebula implements a native auth mechanism which can be based on different authentication backends[8]. The current implementation of the Device Client and the Provisioning Engine only supports basic HTTP authorization, so only the native OpenNebula mechanism based on username and password is used.

Upon receiving a request to create a Service Runtime from a Device Client, the Provisioning Engine delegates the credentials provided by the Device Client to the Cloud-Edge Manager. It then requests the creation or retrieval of a Serverless Runtime, to then request the execution of a FaaS function. In this first development cycle, this request to the Serverless Runtime is not authenticated. This flow is described in the figure below.

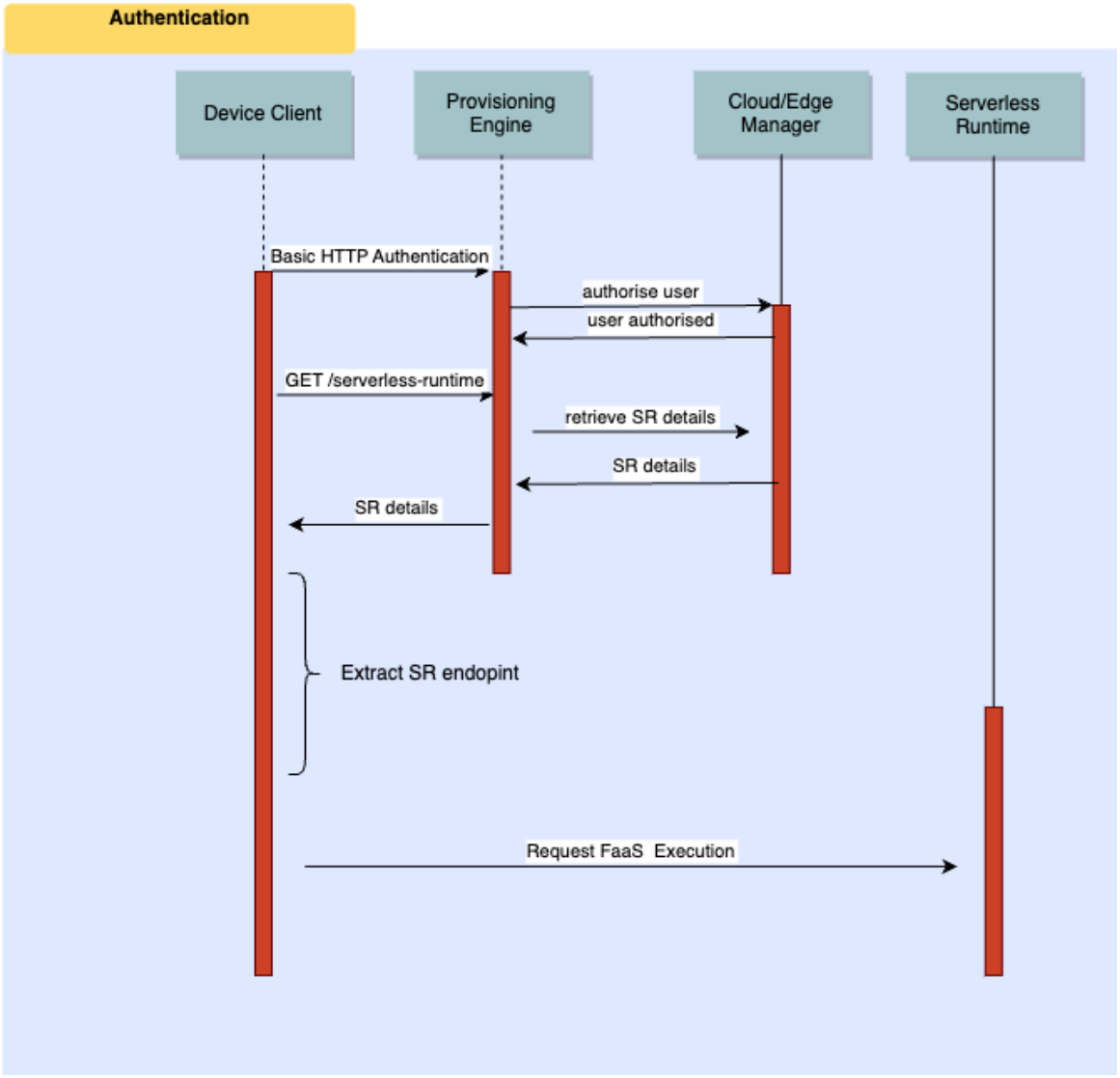

Figure 1.2. Authentication flow in the current COGNIT Framework

[8] https://docs.opennebula.io/6.6/installation_and_configuration/authentication/index.html

Analysis

In this first cycle we have undertaken a comprehensive analysis to implement a fine-grained Identity and Access Management (IAM) mechanism, which will play a pivotal role in the authorization of requests across the entire COGNIT stack. In this first development cycle, the project has recognized the need for a robust access control mechanism capable of ensuring secure and encrypted authentication and authorization. The ultimate goal is to establish a comprehensive IAM system that can cater to the diverse needs of the COGNIT project. As such, the project aims to achieve a higher level of security and control by implementing this fine-grained IAM mechanism. This is aligned and complemented by SR6.1 Advanced Access Control, where a policy based access control (PBAC) will be implemented.

To illustrate its implementation, we have identified three critical communication points within the COGNIT stack, each of which will employ a combination of techniques and technologies, such as JWT (JSON Web Token), SSL (Secure Sockets Layer), and OpenNebula's authentication capabilities. These communication points and their corresponding mechanisms are detailed below:

Device Client to Provisioning Engine

- Communication will be secured using both JWT and SSL.
- JWT (JSON Web Tokens) will be employed to ensure secure identity verification and authorization of the Device Client.
- SSL (Secure Sockets Layer) encryption will be utilized to protect the integrity and confidentiality of the data transferred between the Device Client and the Provisioning Engine.

Provisioning Engine to Cloud-Edge Manager

- Communication between the Provisioning Engine and the Cloud-Edge Manager will be secured through SSL encryption.
- Cloud Servers OpenNebula Authentication[9] will be utilised for this connection, using the oneadmin account, which serves as the root account in an OpenNebula cloud.
- However, a mechanism similar to the OpenNebula serveradmin account will be used, a special authentication mechanism that enables servers to execute operations on behalf of another user, thereby ensuring a correct level of security and isolation that are key requirements of a multi-tenancy environment.

Device Client to Serverless Runtime

- For communication between the Device Client and the Serverless Runtime, a combination of JWT and SSL will be used.
- JWT can be employed to encapsulate other needed auth mechanisms, like Keycloak tokens, enhancing the security and management of identity and access for the Device Client.

[9] https://docs.opennebula.io/6.6/integration_and_development/references/cloud_auth.html#cloud-servers-authentication

- The use of this encapsulation also enables the possibility of the Serverless Runtime reaching out to external services on behalf of the Device Client.
- SSL encryption will guarantee the confidentiality and integrity of data transferred during this interaction.

The fine-grained IAM mechanism within the COGNIT Framework is a critical component that will enhance the security, authentication, and authorization of requests across the project. It offers a comprehensive approach to ensure that every communication point is both secure and efficient. By implementing these technologies and mechanisms, the COGNIT Project is taking a significant step towards achieving its overarching goal of robust Identity and Access Management.

2. AI-Enabled Orchestrator

The AI-Enabled Orchestrator offers recommendations to enable optimal placement of Serverless Runtimes that aligns with application requirements while also promoting efficient resource utilisation. It is designed to be able to eventually adapt dynamically to shifting application requirements and resource availability in the cloud-edge continuum. To achieve this, the AI-Enabled Orchestrator will use multi-objective optimization to concurrently evaluate several objectives simultaneously. This will ensure that trade-offs between conflicting goals, such as maximising performance while minimising cost, are well-balanced. The main focus of this initial version of the implementation has been on preliminary research and on the integration of its different subcomponents.

[SR5.1] Building Learning Models

Description

Algorithms must be developed and refined to provide the capability for automated placement of serverless runtimes based on application and resource requirements and state. These placement decisions are not static; they must be actively monitored and dynamically optimised based on changing demand and environmental state. However, existing service providers are still unclear on how to automate the orchestration process for complex heterogeneous workloads across the cloud-edge continuum. Machine learning (ML) has been applied in workload orchestration for different downstream tasks, including characterising workloads, optimal placement, load balancing, scaling, scheduling, and resource allocation. However, there is no single solution that provides a complete picture of any task.

Analysis

Smart deployment of serverless runtimes in a Continuum system is a complex and challenging task; in the context of the COGNIT project as well as in general across cloud-edge continuum. Machine learning plays a key role to induce intelligence across the downstream tasks (e.g. dynamic placement, characterise workloads) in the cloud-edge continuum. Before discussing these tasks, here it includes foundational details of machine learning algorithms and how they are employed. ML algorithms can be classified as classical and advanced or deep learning. They are further classified as supervised, semi-supervised, unsupervised and reinforcement learning, how data are used to train the models for a particular task. The classical ML algorithms (e.g., K-nearest neighbour, decision tree, random forest, support vector machine, regression) perform well even if some cases have a lower amount of data and less in black-box. Advanced ML or deep learning algorithms (e.g., Long Short Term Memory, Transformer, Autoencoder) are data-hungry and need a huge amount of resources to train a model, but they perform well in complex scenarios [16, 17]. The following figure shows a taxonomy of the classical and advanced ML methods applied for cloud orchestrators:

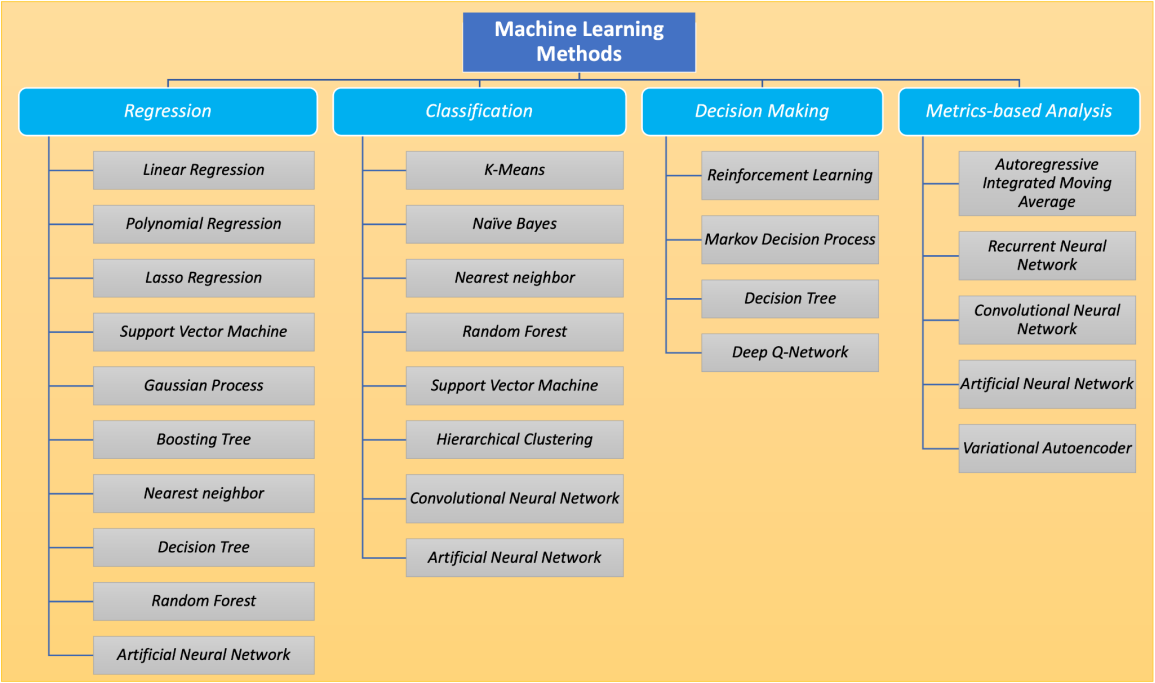

Figure 2.1. Taxonomy of ML methods applied in cloud orchestrators

Based on the tasks, ML models are classified as regression, classification, decision-making, and metric-based analysis. Below are some examples of ML methods.

Random Forest

Random Forest is a supervised learning algorithm, which is developed based on the decision tree and ensemble learning. One of the recent for workload prediction in serverless framework [18] employed this algorithm because it is computationally less intensive and needs a low amount of data for training. An example structure for the Random Forest algorithm is given in the following figure:

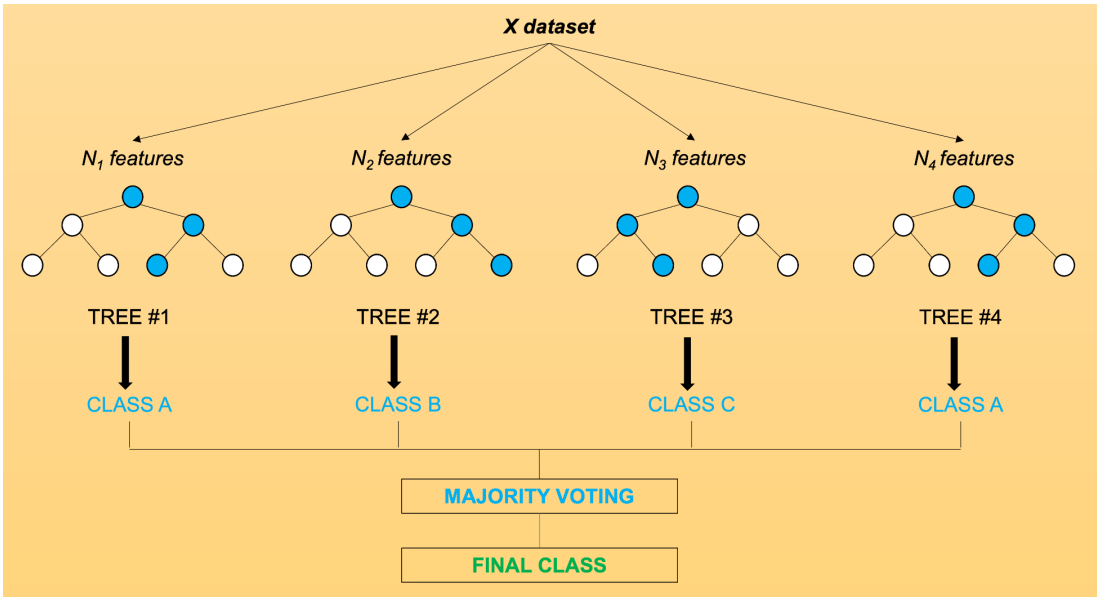

Figure 2.2. Random Forest

Attention-based RNN

Building a model for multiple time steps prediction has often been a challenging task for diverse applications, including FaaS workloads in serverless. One of the recent models developed based on Recurrent Neural Networks (RNN). The input sequence encodes into new representation with encoder and decodes the representation into an target sequence.

An attention layer is integrated to the encoder-decoder RNN model that allows the network to focus on parts of the input sequence that are relevant to predicting target sequence. The attention model is jointly trained with other components of the model that are able to predict multiple time steps. An attention-based RNN architecture is given in the following figure:

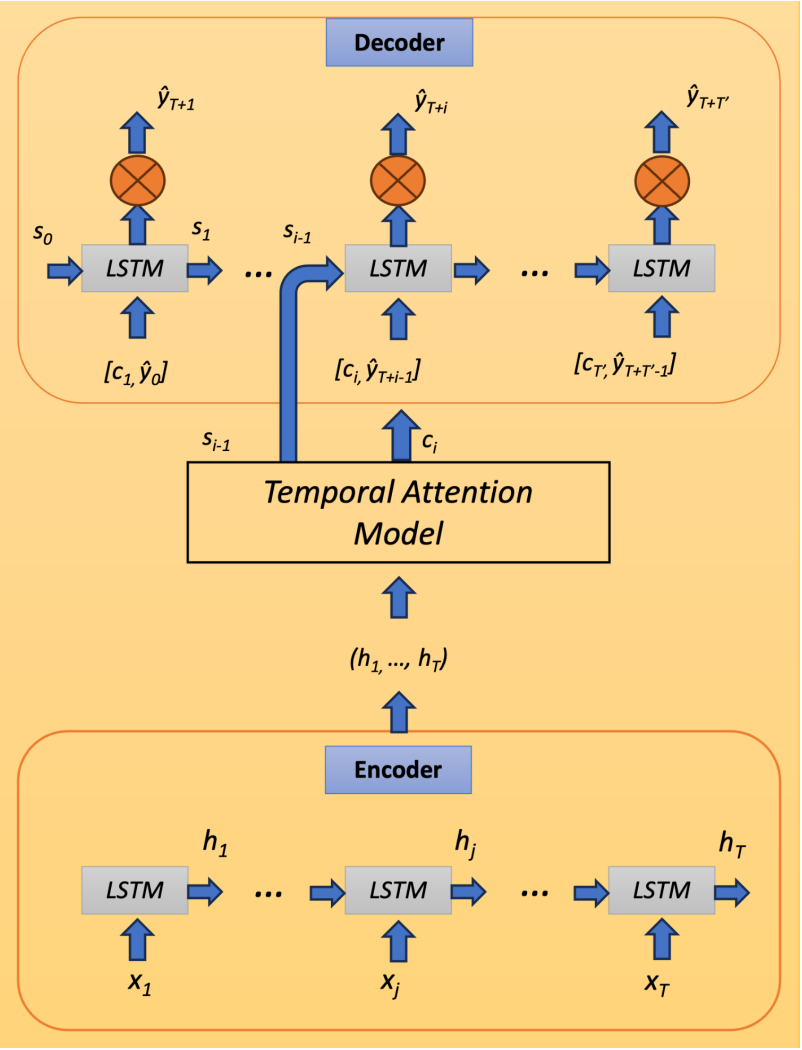

Figure 2.3. Attention-based RNN

ML-based orchestrators are classified based on the application architecture, optimization objectives, infrastructures, behaviour modelling and prediction, and resource provisioning. The following figure illustrates the taxonomy of ML-based orchestrators.

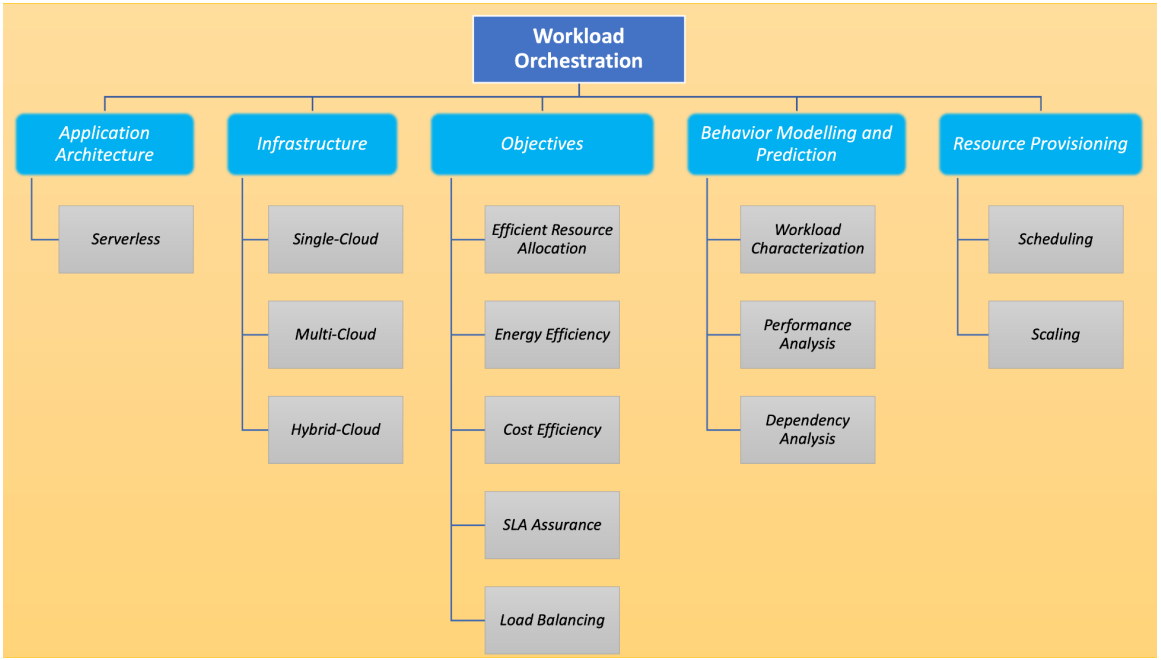

Figure 2.4. Taxonomy of ML-based orchestrators

One of the recent works shows data-centric function orchestration in serverless, where the platform provides a data-bucket abstraction to hold the intermediate data generated by function orchestration [15]. Another work [14] is focused on a multi-objective scheduling policy for a serverless-based cloud-edge continuum. However, there are several opportunities to induce AI for orchestration that enable it to place serverless runtimes optimally across the cloud-edge continuum.

Optimization objectives

Given the variety of applications and cloud infrastructures, optimising serverless runtimes depends on the metrics collected from applications and resources. Orchestration solutions are often designed to satisfy multiple objectives that either satisfy resource requirements or satisfy the Quality of Service (QoS). Achieving a balance between the multiple optimization objectives remains a central challenge in ML-based serverless runtime placement and resource management. A few important and relevant optimization objectives are listed below.

- Define and processing of data: Defining where data is being processed according to Serverless Runtime deployment attributes for performance, cost, security, and energy requirements. While resources must be close to the final use for latency-sensitive applications, the resources must be close to the data sources for data-intensive applications. If edge resources are limited or insufficient, functions can be executed on the cloud for compute-intensive applications.
- Energy Efficiency. The continuous growth of cloud data centres consumes tremendous power. One of the key objectives for geo-distributed placement of serverless runtimes is to prioritise green energy hosts. Optimising energy efficiency and reducing power grid congestion, through optimized geographic distribution of workloads in distributed cloud-edge infrastructures that adapt in

real-time, to available energy and reduce environmental impact - exploiting the opportunities offered by the edge-cloud continuum while incentivizing the local generation and use of green energy.

- Resource Efficiency: The key challenge here is to utilise the resource metrics and system states to optimise resource utilisation using ML-based methods. Dynamic application task placement and orchestration for cost-efficient, secure and optimal performance in the continuum layers through advanced deep learning methods is another key challenge.
- SLA Assurance: To fill the dynamic requirements of applications, diverse service level agreements (SLAs) are often identified, including response time, throughput, initialization time, etc. These SLAs become prime optimization criteria to meet the requirements of dynamic workloads for serverless runtimes. Improving the reliability of the infrastructure by anticipating failures and local power grid congestion.
- Cost Efficiency: Market-based cloud solutions prioritise cost efficiency, as they follow the pay-as-you-go model. The goal is to minimise overall financial costs while meeting user-defined QoS requirements.
- Load Balancing: Distribute loads across the networks among runtimes evenly based on a specific policy like RoundRobin. Its results will improve system scalability, availability, and network performance. Implementing dynamic load balancing to respond to changing requirements from the application execution flow or to changes in capacity demanded by the Serverless Runtime due to data size or processing needs variability.

Before we can design, implement, and validate effective algorithms to perform all of these functions, it is first necessary to adequately develop formal models that accurately identifies the many components of a continuum system (applications, compute resources, storage, energy providers, business process, etc.), together with their interactions. This is particularly relevant when considering the energy efficiency of such a system - an area currently under-developed in the existing literature.

We have explored recent modelling literature and have categorised it into seven brackets: research focus, Continuum coverage, formal model, energy model, optimization objectives, type of applied technique, the evaluation method, and prospective application areas. This is illustrated in Tables 2.1 and 2.2 below.

After careful investigation of the current research on Continuum modelling, we observe that in the literature several formal models for traditional cloud systems have been proposed {Benzadri2013, Khosravi2017} but these do not capture the dynamic nature of cloud-edge systems or integrate stochastic properties, energy providers, pricing, and renewable energy sources.

Most work assumes a single datacenter, precluding intrinsic challenges faced with the management of federated systems, such as how to monitor and schedule multiple complex resources across multiple networks in a scalable and decentralised manner with SLO awareness {Nastic2021}, and how to balance accuracy with decision making latency

(many recent approaches, such as {Gao2020, Zhang2019}, use machine-learning methods that are too slow to provide the ultra-low latency scheduling required by edge applications).

A basic model that integrates nodes with energy providers is presented in {Xu2021} but does not consider federated edge systems or cross-site monitoring issues, while {Li2018} only considers micro-grid integration with edge nodes, with no centralised cloud integration. Of work that does consider energy-aware federated systems, little has been achieved; {Minh2022} propose an integration between smart grids and cloud-edge systems but the proposed architectural model is extremely high-level and does not consider monitoring overhead, task properties, or decentralised control.

Modelling and Prediction

Modelling behaviour of application workloads in serverless is the study of complex system behaviour and patterns by analysing application-level metrics. It helps in near-accurate prediction for ML techniques, it further benefits orchestration's optimization objectives and resource provisioning decisions.

Workload characterization is key for assessing dynamic applications and understanding workload behaviours, which is essential for optimal serverless runtime placement. The following table compares ML-based workload characterization methods:

Method	Infrastructure	Application Architecture	Model	Objective	Advantages	Limitations
ANN, 2023 [13]	Single cloud	Monolithic	CNN with ReLu	Classify data points into relevant workload classes	Performance improvement	Scalability
Metrics-based Analysis, 2021 [1]	Hybrid cloud	Microservice	Bi-LSTM	Task arrival rate prediction	Performance improvement	Inaccuracy in long-term forecasts
Metrics-based analysis, 2020 [2]	Single cloud	Serverless	LSTM	Request arrival rate prediction	High prediction accuracy	Simplicity of application models
Classification, 2020 [3]	Single cloud	Monolithic	K-means ++	Resource demand prediction	High scalability	Limited accuracy under high load variance
Metrics-based analysis, 2020 [4]	Single cloud	Monolithic	ARIMA	Request arrival rate prediction	Capability of large data scales	Inaccuracy under trend turning
Metrics-based analysis, 2020 [5]	Hybrid Cloud	Microservice	IGRU-SD	Resource demand prediction	Low error rates	Unclear demonstration of the relationship

						between resource allocation and energy efficiency

Table 2.1. Summary of workload characterization methods

Resource Provisioning

Resource provisioning for applications is challenging due to the diversity of cloud workloads, resource heterogeneity and complex applications. To address this complexity, modern resource provisioning strategies focus on ML-based approaches to optimise resource allocation accurately and efficiently.

Scheduling determines the initial placement of serverless runtimes, considering various application and task structures. It significantly impacts application performance and resource efficiency. One recent work provided on greedy scheduling policies for serverless framework [14]. The following table summarises the existing scheduling methods:

Method	Infrastructure	Application and Arch.	Task Structure	Objective	Advantages	Limitations
Greedy, 2023 [14]	Cloud-edge continuum	Serverless	Multiple	Cost and latency	Multi-objective task placement	Highly dependent
Heuristic, 2021 [6]	Hybrid cloud	Serverless	Single	Cost and latency minimization	Multi-objective task placement	Limited accuracy under high load variance
Heuristic, 2020 [7]	Hybrid cloud	Serverless	Graph-based	Cost minimization	SLA- assurance	Simplicity of application workloads
Heuristic, 2020 [3]	Single cloud	Serverless	Multiple independent	Resource utilisation improvement and energy saving	Reduction of cold start and response latency	Poor efficiency for tasks with long lifetimes

Table 2.2. Summary of scheduling methods

Even though there are solutions for orchestration of workloads, there are several open challenges for cloud-edge continuum workload orchestration based on the dynamic changes of serverless runtimes. However, these workloads can be optimised with multi-objective optimization for placement of runtimes. The following table lists some challenges and potential solutions:

Challenge in workload orchestration	Potential AI-based solutions
How to characterise workloads by modelling and predicting requests behaviour and resource usage patterns?	LSTM, K-means++, Bi-LSTM, ATeen-LSTM, GRU, Attention-based RNN
How to analyse inter-dependency between applications and classify them?	Random Forest, BO, GP, CNN, and LSTM
How to analyse dependency of tasks in applications?	SVM, Graph neural networks, Probabilistic neural networks
How to achieve energy efficient resource scheduling for serverless runtimes?	MDP, Q-learning,
How to balance the trade-offs between different metrics during workload orchestration?	Actor-critic, ANN, and RF
How to alleviate function cold starts in serverless computing?	Q-learning, LSTM, Linear regression
How to make scaling/migration decisions for runtimes?	Model-based RL, MDP

Table 2.3. Potential AI-enabled orchestration solutions for optimal serverless runtime placements

The overall analysis of the state-of-the-art on Cloud-Edge Continuum highlights the lack of unified systems, formal models, advanced machine learning models, and methods to seamlessly integrate various energy factors including temporal pricing, renewable energy sources, energy provider requirements, resource restrictions, and balance consumption over large-areas with other non-Cloud consumers.

Research in this field typically results in either reference architectures or simulated system environments, with computing, networking, and storage resource management serving as the primary focus. These observations demonstrate the absence of a unified resource orchestration technique capable of integrating the pricing models, types of workloads, multi-objective optimization, monitoring, and controlling strategies, QoS and SLO requirements of end-users, heterogeneous systems and networking technologies, energy policies, energy providers, energy sources, and administration of compute and network resources in the energy-aware federated Cloud-Edge Continuum.

There is, therefore, a clear need to bridge this gap and exploit the modelling of cloud-edge continuum key components, their relevant stochastic properties and interactions, and their integration with key energy factors as well as advanced machine learning models to induce intelligence for across downstream tasks.

References

1. Chan KY, Abu-Salih B, Qaddoura R, Ala'M AZ, Palade V, Pham DS, Del Ser J, Muhammad K. Deep Neural Networks in the Cloud: Review, Applications, Challenges and Research Directions. *Neurocomputing* 2023 May 13:126327.

2. Zhiheng Zhong, Minxian Xu, Maria Alejandra Rodriguez, Chengzhong Xu, and Rajkumar Buyya. 2022. Machine Learning-based Orchestration of Containers: A Taxonomy and Future Directions. *ACM Comput. Surv.* 54, 10s, Article 217 (January 2022).

3. Mohapatra AD, Oh K. Smartpick: Workload Prediction for Serverless-enabled Scalable Data Analytics Systems. *Middleware* 2023.

4. Majid Moradi Aliabadi, Hajar Emami, Ming Dong, Yinlun Huang, Attention-based recurrent neural network for multistep-ahead prediction of process performance, *Computers & Chemical Engineering*, Vol. 140, 2020.

5. Yu M, Cao T, Wang W, Chen R. Following the data, not the function: Rethinking function orchestration in serverless computing. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI'23)* 2023 (pp. 1489-1504).

6. L. Angelelli, A. A. da Silva, Y. Georgiou, M. Mercier, G. Mounié and D. Trystram, "Towards a Multi-objective Scheduling Policy for Serverless-based Edge-Cloud Continuum," *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, Bangalore, India, 2023, pp. 485-497

7. Kumar, Gaurav, Kshira Sagar Sahoo, and M Bhuyan. "Towards a Workload Mapping Model for Tuning Backing Services in Cloud Systems." *International Conference on Database and Expert Systems Applications*. Cham: Springer Nature Switzerland, 2023.

8. Hossain M, Mebratu D, Hasabnis N, Jin J, Chaudhary G, Shen N. CWD: A Machine Learning based Approach to Detect Unknown Cloud Workloads. arXiv preprint arXiv:2211.15739. 2022 Nov 28.

9. Ming Yan, XiaoMeng Liang, ZhiHui Lu, Jie Wu, and Wei Zhang. 2021. HANSEL: Adaptive horizontal scaling of microservices using Bi-LSTM. Applied Soft Computing 105, 1 (2021), 107216–107230.

10. Zhiheng Zhong, Jiabo He, Maria A. Rodriguez, Sarah Erfani, Ramamohanarao Kotagiri, and Rajkumar Buyya. 2020. Heterogeneous task co-location in containerized cloud computing environments. In Proceedings of the 2020 IEEE International Symposium on Real-Time Distributed Computing. 79–88.

11. Jashwant Raj Gunasekaran, Prashanth Thinakaran, Nachiappan C. Nachiappan, Mahmut Taylan Kandemir, and Chita R. Das. 2020. Fifer: Tackling resource underutilization in the serverless era. In Proceedings of the 2020 International Middleware Conference. 280–295.

12. Shubo Zhang, Tianyang Wu, Maolin Pan, Chaomeng Zhang, and Yang Yu. 2020. A-SARSA: A predictive container auto-scaling algorithm based on reinforcement learning. In Proceedings of the 2020 IEEE International Conference on Web Services. 489–497

13. Yao Lu, Lu Liu, John Panneerselvam, Bo Yuan, Jiayan Gu, and Nick Antonopoulos. 2020. A GRU-based prediction framework for intelligent resource management at cloud data centres in the age of 5G. IEEE Transactions on Cognitive Communications and Networking 6, 2 (2020), 486–498.

14. L. Angelelli, A. A. da Silva, Y. Georgiou, M. Mercier, G. Mounié and D. Trystram, "Towards a Multi-objective Scheduling Policy for Serverless-based Edge-Cloud Continuum," *IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, Bangalore, India, 2023, pp. 485-497

15. Anirban Das, Shigeru Imai, Stacy Patterson, and Mike P Wittie. 2020. Performance optimization for edge-cloud serverless platforms via dynamic task placement. In Proceedings of the 2021 IEEE/ACM International Symposium on Cluster, Cloud, and Internet Computing. 41–50.

16. Nabeel Akhtar, Ali Raza, Vatche Ishakian, and Ibrahim Matta. 2020. COSE: Configuring serverless functions using statistical learning. In Proceedings of the IEEE INFOCOM 2020 - IEEE Conference on Computer Communications. 129–138.

17. Jashwant Raj Gunasekaran, Prashanth Thinakaran, Nachiappan C. Nachiappan, Mahmut Taylan Kandemir, and Chita R. Das. 2020. Fifer: Tackling resource underutilization in the serverless era. In Proceedings of the 2020 International Middleware Conference. 280–295.

[SR5.2] Smart Deployment of Serverless Runtimes

Architecture & Components

The architecture for AI-Enabled Orchestrator is straightforward with primary focus on integration in this cycle of implementation. The Cloud-Edge Manager sends a request to the AI-Enabled Orchestrator, which contains a list of potential hosts for placement in turn, the AI-Enabled Orchestrator interacts with the Cloud-Edge Manager to gather additional details about the shortlisted hosts, such as their present workload, CPU, and remaining memory. The figure below shows the high-level architecture of the AI-Enabled Orchestrator:

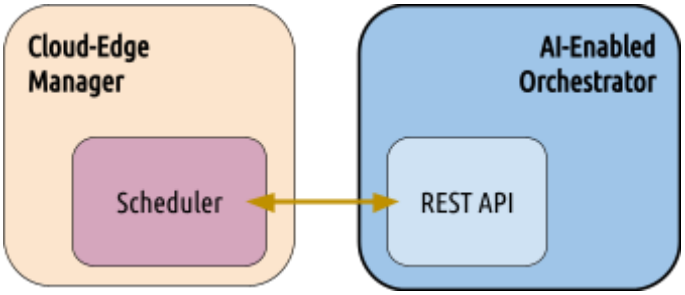

Figure 2.5. High-level architecture of the AI-Enabled Orchestrator

The AI-Enabled Orchestrator is implemented in Python and deployed as a microservice either as a standalone Docker container. In the first version of the AI-Enabled Orchestrator, an improved FFD (First-Fit Decreasing) algorithm (see below) has been implemented and deployed to prioritise placement recommendation of VMs with associated hosts. There are several AI algorithms under implementation based on the downstream tasks (e.g. workload characterization and prediction, dynamic and optimal placement). These algorithms will be provided as plug-ins, allowing the AI-Enabled Orchestrator to efficiently adapt to a wide variety of scenarios and requirements.

Algorithm #1: FFD placement

Input: VM configuration and available hosts

Output: Assignment results

```
Sort vms by decreased order of Cpu
Assignment_results = []
for vm in vms:
get available hosts resources
  for host in renewable_hosts:
        if vm['CPU'] < host['CPU'] and vm['Memory'] < host['Memory']:
      result = [vm, host]
    if vm unassigned:
            for host in unrenewable_hosts:
```

```
                    If vm[‘CPU’] < host[‘CPU’] and vm[‘Memory’] <
host[‘Memory’]:
                    result = [vm, host]
  if vm unassigned:
         Result = [vm, none]
  Assignment_results.append(result)
return Assignment_results
```

Initial implementation of a First-Fit Decreasing (FFD) algorithm

Data Model

Information about the Data Model is reported in presented the next section SR5.3

API & Interfaces

Communication between the Cloud-Edge Manager and the AI-Enabled Orchestrator is done using a HTTP protocol. The placement request is directed to the AI-Enabled Orchestrator, which runs a HTTP server implemented using FastAPI. The request carries JSON data, as illustrated below. The AI-Enabled Orchestrator needs to rank all hosts specified in the HOST_IDS array, taking into account both the system state and the provided capacity requirements.

Action	Verb	Endpoint	Request Body	Response
Request a placement plan for PENDING Serverless Runtimes	POST	/	JSON representation of the VMs associated with pending Serverless Runtime services and the list of available HOSTs.	Status code 200 (Success) if the execution was successful. A JSON with information about the placement is returned. 400 (Bad request) if there is any error.

Table 2.4. API that defines the endpoint of the AI-Enabled Orchestrator

[SR5.3] Scheduling Mechanisms

Description

The default OpenNebula scheduler implements a matchmaking algorithm that assigns Virtual Machines in pending state to suitable hypervisor hosts, or nodes for short. This algorithm is implemented in two different phases. The first phase involves filtering the available nodes to remove those that are not suitable for a particular VM (this can happen for multiple reasons, for instance not enough capacity in terms of CPU and/or memory). In the second stage of the algorithm, these shortlisted nodes are ordered by priority, and the one with higher priority is chosen to deploy the VM on it.

The scheduler has been extended so the second phase (prioritization) can be delegated to an external scheduler that can then implement additional algorithms tailored to your specific use case or business logic. Communication is established through a straightforward REST API, which will be encrypted in subsequent development cycles.

Architecture & Components

In the context of the High-Level Architecture of the Cloud-Edge Manager's Scheduler extension, Figure 2.6 below provides a visual representation of the system's structure. You can refer to this figure to gain a deeper understanding of how the components interact and the overall layout of the COGNIT Architecture.

When the Cloud-Edge Manager receives a request from the Provisioning Engine to create a new Serverless Runtime, it initiates an internal process to add a new entry in the Virtual Machine Pool. This entry corresponds to a Virtual Machine in a 'Pending' state.

The matchmaking Scheduler, a core component, plays a pivotal role in the system's operation. It runs once in every scheduling cycle, typically set to a 30-second interval but configurable. During its execution, it gathers comprehensive information about all available hosts and Virtual Machines in the 'Pending' state.

Following this data retrieval phase, the matchmaking Scheduler engages in the Filtering process, as elaborated upon in earlier sections. Instead of proceeding directly to the prioritisation phase, a critical step involves the scheduler making a synchronous call to the AI-Enabled Orchestrator REST API. This call is made through a POST request, as documented in the Data Model section. The information concerning hosts and Virtual Machines is transmitted during this request.

It is important to note that this API call operates with a timeout mechanism in place. If the AI-Enabled Orchestrator does not respond in a timely fashion, the matchmaking Scheduler assumes the responsibility of making the scheduling decision. This ensures that the system maintains a degree of resilience and timely operation even in the face of potential delays.

Moreover, it's worth mentioning that the information passed to the AI-Enabled Orchestrator may not be exhaustive for making all the scheduling decisions. In such cases, the XMLRPC API of the Cloud-Edge Manager can be leveraged to request additional information about specific hosts or Virtual Machines. Furthermore, the system can tap into Prometheus integration with the Cloud-Edge Manager's monitoring system. This

integration enables the querying of valuable insights, such as energy consumption, to enrich the decision-making process. This multifaceted approach ensures that the scheduling system is well-informed and adaptable, making it capable of handling diverse scenarios effectively.

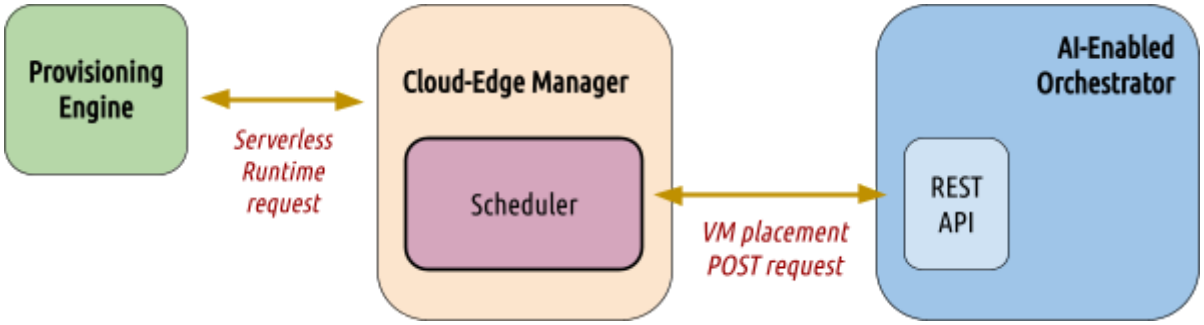

Figure 2.6. High level architecture of the Cloud-Edge Manager's Scheduler extension

Data Model

This section delves into the underlying structure of JSON documents exchanged between components in the Cloud Manager scheduler extension. These JSON documents are central to the effective communication and coordination of resources within the system. The following table provides comprehensive insight into the attributes and their meanings within these documents, elucidating how information is formatted and transmitted.

Fields	Type
CAPACITY	An array indicating the VM's capacity requirements
CAPACITY/CPU	Requested relative CPU shares (e.g. 0.5)
CAPACITY/MEMORY	Memory in kilobytes
CAPACITY/DISK	Additional disk space in the system datastore (in megabytes)
HOST_IDS	A list of IDs for hosts meeting the VM requirements
ID	VM's ID
STATE	Current state in string form
ATTRIBUTES	Custom attributes as specified in scheduler configuration in EXTERNAL_VM_ATTR.

Table 2.5. Attributes between OpenNebula and AI-Enabled Scheduler communication

The following examples describe a JSON schema of the data sent to the AI-Enabled Orchestrator API and the expected return, respectively:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "VMS": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "VM_ATTRIBUTES": {
            "type": "object",
            "properties": {
              "GNAME": { "type": "string" },
              "UNAME": { "type": "string" }
            },
            "required": ["GNAME", "UNAME"]
          },
          "CAPACITY": {
            "type": "object",
            "properties": {
              "CPU": { "type": "number" },
              "DISK_SIZE": { "type": "number" },
              "MEMORY": { "type": "number" }
            },
            "required": ["CPU", "DISK_SIZE", "MEMORY"]
          },
          "HOST_IDS": {
            "type": "array",
            "items": { "type": "integer" }
          },
          "ID": { "type": "integer" },
          "STATE": { "type": "string" }
        },
        "required": ["VM_ATTRIBUTES", "CAPACITY", "HOST_IDS", "ID", "STATE"]
      }
    }
  },
  "required": ["VMS"]
}
```

JSON Schema for Scheduler to AI-Enabled Orchestrator communication

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "VMS": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "ID": { "type": "integer" },
          "HOST_ID": { "type": "integer" }
        },
        "required": ["ID", "HOST_ID"]
      }
    }
  },
  "required": ["VMS"]
}
```

JSON Schema for AI-Enabled Orchestrator to Scheduler communication

To illustrate this point, let's consider another example (below) where the following JSON document serves as a prime representation. Within this JSON structure, a request for provisioning three Virtual Machines (VMs) is articulated, along with their specified matching hosts:

```
{
  "VMS": [
    {
      "VM_ATTRIBUTES": {
        "GNAME": "oneadmin",
        "UNAME": "oneadmin"
      },
      "CAPACITY": {
        "CPU": 1.5,
        "DISK_SIZE": 1024,
        "MEMORY": 131072
      },
      "HOST_IDS": [
        3,
        4,
        5
      ],
      "ID": 32,
```

```
      "STATE": "PENDING"
    },
    {
      "VM_ATTRIBUTES": {
        "GNAME": "users",
        "UNAME": "userA"
      },
      "CAPACITY": {
        "CPU": 1.5,
        "DISK_SIZE": 1024,
        "MEMORY": 131072
      },
      "HOST_IDS": [
        3,
        4,
        5
      ],
      "ID": 33,
      "STATE": "PENDING"
    },
    {
      "VM_ATTRIBUTES": {
        "GNAME": "users",
        "UNAME": "userA"
      },
      "CAPACITY": {
        "CPU": 1.5,
        "DISK_SIZE": 1024,
        "MEMORY": 131072
      },
      "HOST_IDS": [
        3,
        4,
        5
      ],
      "ID": 34,
      "STATE": "PENDING"
    }
  ]
}
```


HTTP Payload example for POST request from Scheduler to AI-Enabled Orchestrator

The external scheduler should respond with a compliant structure, incorporating the chosen HOST_ID for each VM. For example, the response to the aforementioned request might appear as follows:

```
{
  "VMS": [
    {
      "ID": 32,
      "HOST_ID": 3
    },
    {
      "ID": 33,
      "HOST_ID": 3
    },
    {
      "ID": 34,
      "HOST_ID": 5
    }
  ]
}
```

HTTP Payload example for POST answer from AI-Enabled Orchestrator to Scheduler

API & Interfaces

The Scheduler employs a straightforward REST API for its operation. When the external scheduler is configured, the OpenNebula scheduler initiates a POST operation directed to the specified URL. The sole mode of interaction between the Cloud-Edge Manager's Scheduler and the AI-Enabled Orchestrator occurs through this POST request and its corresponding response, facilitated by HTTP payloads.

To offer a visual understanding of the REST API's structure, the following example provides a simplified template designed to assist in the creation of the AI-Enabled Orchestrator API. This template, implemented in Ruby and utilising the Sinatra web framework, serves as a valuable resource. The central objective of this scheduler is to take the initial list of hosts for each virtual machine and employ a host allocation randomization strategy based on the Virtual Machine ID:

```
{
  "VMS": [
    {
      "ID": 32,
      "HOST_ID": 3
    },
    {
      "ID": 33,
```

```
      "HOST_ID": 3
    },
    {
      "ID": 34,
      "HOST_ID": 5
    }
  ]
}
```

Ruby Template codifying the AI-Enabled Orchestrator API

3. Energy Efficiency Optimization in the Multi-Provider Cloud-Edge Continuum

Preliminary research on energy efficiency optimization

Recent years have seen a rapid expansion of renewable energy supply, both on the grid-level (regionally) and through (highly distributed) on-site power generation. The local availability of renewable energy supply varies between different locations and also throughout the day, depending on many factors including the local energy mix (e.g., hydro, PV, wind), weather conditions, and the time of day. These fluctuations and unpredictability of the energy supply pose many challenges to modern energy systems. To further complicate things, power capacity constraints of the power transmission and distribution networks are another challenge in many urban areas, which necessitates implementation of demand response mechanisms (dynamically varying the power demand in response to power supply and grid capacity).

The geographically distributed nature of the cloud-edge continuum, as well as the temporal flexibility of some computing tasks, makes it possible to adapt the scheduling and placement of workloads based on the local availability of renewable energy. Improving energy efficiency and sustainability are key considerations for the AI-Enabled Orchestrator in COGNIT when scheduling and determining placement of workloads within the cloud-edge continuum. In terms of energy efficiency, the aims of COGNIT are twofold:

- To increase the amount of renewable energy used—or more precisely, reduce the amount of non-renewable energy used.
- To reduce the amount of energy used to process the workloads managed by COGNIT, i.e. manage the infrastructure and scheduling/placement of workloads in such a way that the energy used is minimised.

To achieve these objectives, the AI-Enabled Orchestrator will take into account:

- The availability of renewable energy, and more specifically the carbon intensity of the energy supply, at different cloud-edge locations.
- Energy profiles of different applications and energy efficiency of different hosts, to improve the pairing between Serverless Runtimes and hosts.
- Migration overhead when deciding whether or not to migrate a Serverless Runtime.
- Power grid congestion and available power capacity.

The AI-Enabled Orchestrator will have access to up-to-date day-ahead forecasts of renewable energy supply at the different locations/edge clusters, as well as detailed monitoring data of the energy use of the Hosts and Serverless Runtimes.

The COGNIT Project will address a number of challenges that arise when considering the highly distributed multi-provider context of the cloud-edge continuum:

- Creating forecasts of available “renewable compute capacity” across the cloud-edge continuum.
- Predicting the energy profiles of different Serverless Runtimes, and the suitability of running different Serverless Runtimes on different hosts.

- Use a data-driven AI approach, rather than considering pre-determined and static energy models, to estimate energy use.
- Introducing energy performance metrics suitable for a highly heterogeneous, multi-provider cloud-edge context.
- Developing a carbon- and grid-aware AI-Enabled Orchestrator.

The current state of the art and challenges related to these aspects are further explored in the following subsections.

Metrics and KPIs for measuring energy efficiency and sustainability

Several different metrics have been proposed for quantifying the energy performance of datacenters, such as [Lykou2018,Shao2022]:

- **PUE** (Power Utilisation Efficiency), which is the ratio *(Total energy used by the datacenter) / (Total energy used by the IT equipment)*, and reflects the overhead of the datacenter facilities (including cooling).
- **GEC** (Green Energy Coefficient), which is the ratio *(Total green energy used by the datacenter) / (Total energy used by the datacenter)*.
- **CUE** (Carbon Usage Effectiveness), which is the ratio *(Total GHG emissions) / (Total energy used by the IT equipment))*.
- **DCeP** (Data Center Energy Productivity), which is the ratio *(Useful work produced) / (Total energy used by the data center)*.
- **CPE** (Compute Power Efficiency), which is the ratio *(IT Equipment Utilisation * IT Equipment Power) / (Total facility power)*.
- **ERE** (Energy Reuse Effectiveness), which is the ratio *[(Total energy used by the datacenter) - (Reused energy)] / (Total energy used by the IT equipment)*, and takes into account energy recirculated into the energy system, e.g., through waste heat utilisation.

These metrics do not comprehensively evaluate the energy performance of datacenters, and new metrics are needed [Lykou2018,Shao2022]. They are also not directly comparable between datacenters, for example if there are large differences in hardware configurations between them, and they do not take into account energy used by network infrastructure (which is critical in a highly distributed cloud-edge continuum). It is also very challenging to attribute/estimate GHG emissions. Moreover, the energy used by IT equipment does not translate into *actual work done*, not the least since IT equipment has significant power draw even in idle state. The metrics DCeP and CPE attempt to bridge this gap, but it is not straight-forward to define what *useful work produced* or *IT equipment utilisation* actually mean, due to the complexity in understanding and quantifying utility and behaviour of different applications and workloads.

One attempt to provide a more detailed definition of the term *(Useful work produced)*, denoted by *W*, is proposed in [Sego2012], and is essentially a weighted sum of application tasks finished over an assessment period:

$$W = \sum_{j=1}^{N_a} \sum_{i=1}^{M_j} V_j \cdot U_j(t_{ij}, T_{ij}) \cdot C_{ij},$$

where $j = 1, ..., N_a$ indexes the applications, N_a is the number of applications, and $i = 1, ..., M_j$ indexes *Useful Computational Units* (UCU) initiated by application j during the assessment period, V_j is the (given) relative value of a UCU of application j, U_j is a prescribed utility function for timely completion of the UCU, and C_{ij} is a binary variable that is 1 if the UCU has completed during the assessment window, and 0 otherwise.

These are challenges that need to be addressed when defining and monitoring energy and sustainability performance metrics in a highly distributed multi-provider context, and will be key to defining good objective functions to guide efficient placement and scheduling of Serverless Runtimes.

Renewable and carbon-aware computing

The percentage of renewable energy, or more precisely the overall energy mix and carbon intensity of the power supply, varies over time and across locations. The precise energy mix depends on many factors, such as the weather, time of day, season, and end-user behaviours. Additionally, some cloud-edge locations may have on-site generation and energy storage. Using day-ahead forecasts of the energy supply mix, it is possible to proactively adapt the amount of computing resources to be made available at the different cloud-edge locations.

To effectively adapt the compute resources, it is necessary to forecast both 1) renewable availability and carbon intensity of the local energy mix at each location, considering both the grid and any available on-site supply, and 2) the compute demand, both system-wide and the demand at each location or cluster of locations (since some workloads may be constrained to certain locations). This idea is explored in a study by Google, which combines day-ahead forecasts of renewable energy supply with day-ahead forecasts of compute demand to generate carbon-aware Virtual Capacity Curves, to adapt the compute capacity throughout the day at different datacenter locations to increase the utilisation of renewable energy [Rad2023]. They consider the temporal flexibility of some of the loads to shape the load curves to fit the availability of renewable energy.

Reinforcement learning, with states given by current availability of CPU, memory, and I/O, current weather, and current electricity price, has been investigated for improving the utilisation of renewable energy when scheduling big data workloads [Xu2020].

Energy use and energy efficiency

Predicting the resource utilisation of an application over time, and estimating how much energy it will use, is a highly complex task. Furthermore, different applications have different utilisation patterns, i.e., in terms of network communication, I/O, memory access, CPU load, etc., that need to be taken into account when they share resources on a single server. This is further complicated by the non-linear relationship between utilisation rate (e.g., CPU load) and the performance (e.g. operations performed), which further depends on the specific hardware platform and configuration. This non-linear relationship is

illustrated, for example, in the results from the SPECpower Benchmark published by the Standard Performance Evaluation Corporation.[10]

This idea of reducing energy use by scheduling VMs with anti-correlated CPU utilisation patterns on the same machine is explored in [Shaw2019].

While it is not straight-forward to define what is meant by application performance, and to model the specific relationship between it and power or energy use, it is possible to formulate a few guiding principles:

- From an energy efficiency perspective, it is not necessarily optimal to maximise CPU/resource utilisation of active hosts.
- Idle/unused hosts use significant energy, but shutting down/starting new hosts takes time.
- The dynamic resource utilisation patterns of different workloads may interfere with one another, but proper prediction of such patterns and scheduling of Serverless Runtimes can reduce such interference and improve performance.
- Different hardware platforms and configurations provide different energy performance to different workloads.

Grid-aware computing, and the role of datacenters for grid stability

While datacenters are one of the fastest growing loads on the power grid, they also offer unique opportunities to help maintain grid balance and enable the integration of more renewables in the energy system, for example by engaging in frequency regulation markets through dynamic demand response—such grid services can be monetized and provide further sources of income [Paananen2021]. In terms of demand side flexibility, even without considering the temporal aspects of scheduling loads, there is significant economic potential in just migrating loads between datacenter locations to participate in different energy markets [Fridgen2017]. For example, the problem of curtailment, i.e., when the supply of renewable energy exceeds the demand, can be potentially mitigated by migrating workloads between different locations [Zheng2020].

The relevant markets and regulations for smart grids are still not fully developed, but it is important to investigate the role of the cloud-edge continuum in such future scenarios. In particular, as datacenters move closer to densely populated and urban areas, they will need to engage with local energy communities and schedule their loads accordingly, in accordance with local smart grid mechanisms and regulations. For example, the Universal Smart Energy Framework[11] (USEF) provides a basis for engaging high power intensity consumers, such as datacenters, to participate in local energy markets and provide various grid services to enhance grid stability.

The current energy aggregator services have been developed and deployed in different national and international demand response pilot projects to evaluate the potential of both behind-the-metre and front-of-the-metre network optimizations. However, such applications have not been fully explored in datacenters, and further investigations are

[10] https://www.spec.org/power_ssj2008/results/power_ssj2008.html.
[11] USEF Energy – Universal Smart Energy Framework

needed to understand which grid services are more financially viable for datacenter operators.

Conclusions and future work

In conclusion, to reduce the energy use of, and the amount of non-renewable energy used by, the cloud-edge continuum, it is important to predict compute demand and how it can be scheduled *over time* and *across locations*, and proactively adapt the compute capacity accordingly. Measures need to be implemented to:

- Identify the most energy efficient of the participating edge clusters (possibly depending on the specific application/workload).
- Use forecasts of local availability (and carbon intensity) of renewable energy (both from the grid and on-site), to produce forecasts of available compute capacity running on renewable energy, across locations. Both grid and on-site energy supply and energy storage solutions need to be considered and optimised.
- Forecast the compute demands and dependencies of applications, across time and space.
- Proactively schedule the workloads and infrastructure to pair workloads with the right hosts, to improve system-level energy performance.
- More research is needed to understand how datacenters can participate in the local energy markets to reduce grid congestion and improve overall grid resiliency and reduce system-wide carbon emissions, including monitoring of, and participation in, smart grid initiatives and standardisation.

In summary, to enable grid-aware computing, and promote datacenters as active participants in future grid services, a holistic approach is needed to better adapt the current datacenter energy systems to developing standard energy market frameworks. It is necessary to investigate which grid services that datacenters should target, and develop interoperability frameworks for the multi-provider cloud-edge continuum based on local regulations and energy mix, as well as integrating their current Energy Management Systems (EMS) with local flexibility market operators and service platforms. Last, but not least, a sustainability assessment framework and methodology is needed to critically evaluate the above, given the local dynamics of carbon intensity and energy mix.

The COGNIT Project aims to contribute to all these points in the coming research and innovation cycles, with a special focus on the challenges associated with energy efficiency optimization across the cloud-edge continuum.

References

[Fridgen2017] Fridgen, G., Keller, R., Thimmel, M., & Wederhake, L. (2017). Shifting load through space–the economics of spatial demand side management using distributed data centers. *Energy Policy*, *109*, 400–413. https://doi.org/10.1016/j.enpol.2017.07.018

[Lykou2018] Lykou, G., Mentzelioti, D., & Gritzalis, D. (2018). A new methodology toward effectively assessing data center sustainability. *Computers & Security*, *76*, 327–340. https://doi.org/10.1016/j.cose.2017.12.008

[Paananen2021] Paananen, J., & Nasr, E. (2021). Grid-interactive data centers: enabling decarbonization and system stability. https://www.eaton.com/content/dam/eaton/markets/data-center/eaton-microsoft-grid-interactive-whitepaper-wp153031en.pdf

[Rad2023] A. Radovanović *et al.*, "Carbon-Aware Computing for Datacenters," in *IEEE Transactions on Power Systems*, vol. 38, no. 2, pp. 1270-1280, March 2023, doi: 10.1109/TPWRS.2022.3173250.

[Sego2012] Sego, L. H., Márquez, A., Rawson, A., Cader, T., Fox, K., Gustafson, W. I., & Mundy, C. J. (2012). Implementing the data center energy productivity metric. *ACM Journal on Emerging Technologies in Computing Systems*, *8*(4), 1–22. https://doi.org/10.1145/2367736.2367741

[Shao2022] Shao, X., Zhang, Z., Song, P., Feng, Y., & Wang, X. (2022). A review of Energy Efficiency Evaluation Metrics for data centers. *Energy and Buildings*, *271*, 112308. https://doi.org/10.1016/j.enbuild.2022.112308

[Shaw2019] Shaw, R., Howley, E., & Barrett, E. (2019). An energy efficient anti-correlated virtual machine placement algorithm using resource usage predictions. *Simulation Modelling Practice and Theory*, *93*, 322–342. https://doi.org/10.1016/j.simpat.2018.09.019

[Xu2020] Xu, C., Wang, K., Li, P., Xia, R., Guo, S., & Guo, M. (2020). Renewable energy-aware big data analytics in geo-distributed data centers with Reinforcement Learning. *IEEE Transactions on Network Science and Engineering*, *7*(1), 205–215. https://doi.org/10.1109/tnse.2018.2813333

[Zheng2020] Zheng, J., Chien, A. A., & Suh, S. (2020). Mitigating curtailment and carbon emissions through load migration between data centers. *Joule*, *4*(10), 2208–2222. https://doi.org/10.1016/j.joule.2020.08.001

Prototype of energy efficiency optimization for cloud-edge orchestration

A first prototype of energy efficiency optimization for cloud-edge orchestration has been developed during the First Research & Innovation Cycle (M4-M9) in order to explore a number of integration aspects of the COGNIT Framework.

The main purpose of the demonstrator was not to investigate advanced energy-aware placement algorithms, but rather to gain sufficient understanding on how to integrate and interact with the various COGNIT components under development. This understanding is crucial for building a working COGNIT platform and addressing in future cycles the development of more sophisticated AI/ML algorithms for workload orchestration.

The figure below illustrates the components of the current prototype:

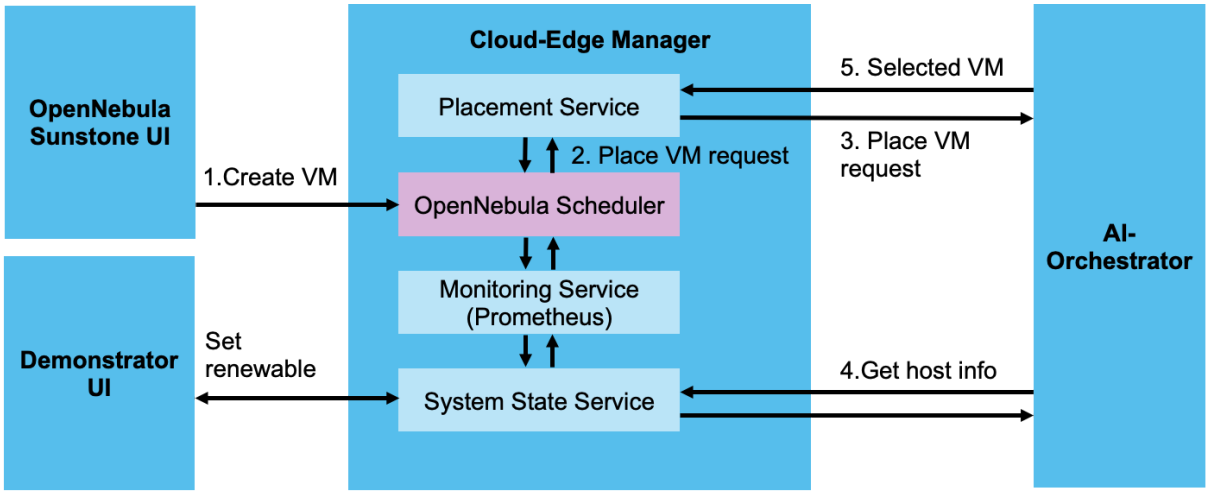

Figure 3.1. Overview of system components in the demonstrator.

The Cloud-Edge Manager serves multiple roles, including interacting with the OpenNebula Scheduler, monitoring and keeping track of the current system state. It is also responsible for interacting with the AI-Enabled Orchestrator.

The AI-Enabled Orchestrator's primary role in this first demonstrator is only to rank and suggest a suitable host for VM placement. When a VM is initiated through the OpenNebula Sunstone GUI service or by an HTTP request (Step 1), the Cloud-Edge Manager sends a request to the AI-Enabled Orchestrator, which contains a list of potential hosts for placement (Steps 2 & 3). In turn, the AI-Enabled Orchestrator interacts (Step 4) with the Cloud-Edge Manager to gather additional details about the shortlisted hosts, such as their present workload and remaining memory. After that, the AI-Enabled Orchestrator selects a host and communicates the recommendation back to the Cloud-Edge Manager, specifying the most optimal host for the new VM.

The selection algorithm is implemented by dividing the shortlisted hosts list into two groups: (1) those operating on renewable energy and (2) those not powered by renewable energy. Subsequently, hosts within these two lists are sorted based on the availability of memory and CPU cores. The final ranking is achieved by merging the two sorted lists,

placing the renewable host list before the non-renewable host list. The top selection is then simply made by selecting the first element in the combined list.

System State Service

A Prometheus database is used to routinely scrape host, VM, and system information from the OpenNebula instance. Running in a separate Docker container, the System State Service[12] extracts data from the Prometheus database every second, creating an in-memory database that mirrors the current system state, such as available memory and CPU on the hosts. Additionally, the System State Service provides an extra renewable state that indicates whether a host is powered by renewable energy.

The System State Service also provides a REST API. This API is used by the AI-Enabled Orchestrator to access host information and is used by the Demonstrator GUI[13] to visualise the current system state.

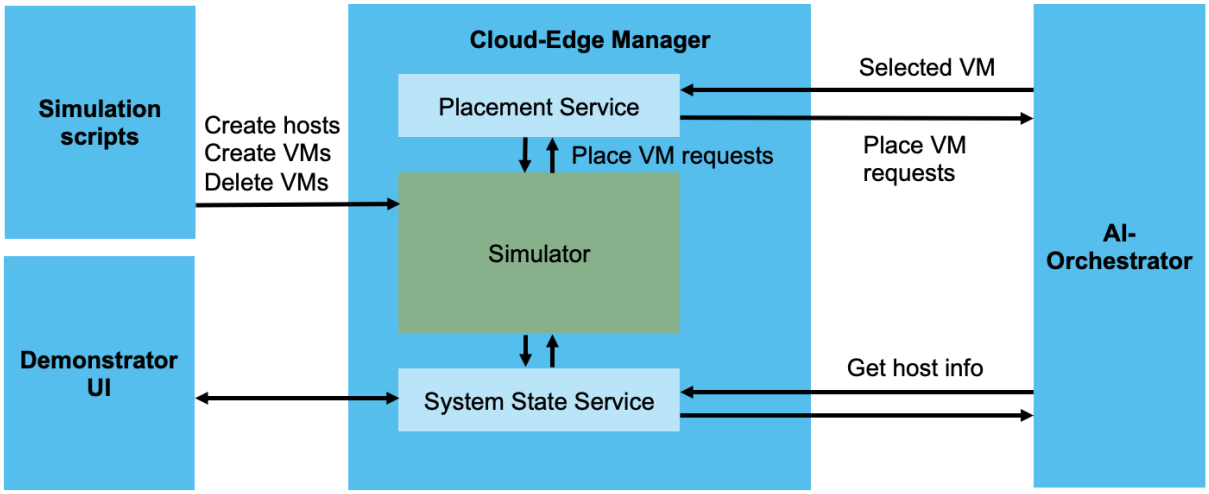

Figure 3.2. Overview of system components in the demonstrator when using a simulator.

Simulation

The Placement Service part of Cloud-Edge Manager introduces an abstraction layer, allowing for integration with schedulers other than OpenNebula's one. Specifically, this design makes it possible to substitute the entire OpenNebula system with a simulator, as shown in Figure 3.2. Using the simulator, a comprehensive testing of the AI-Enabled Orchestrator can be carried out. This enables examinations of various emulated scenarios and evaluation of system performance under diverse workloads. In the next iteration, we aim to use the developed simulator to investigate scalability aspects and observe the behaviour of the AI-Enabled Orchestrator under various workloads and energy availability conditions.

[12] https://github.com/SovereignEdgeEU-COGNIT/ai-orchestrator/tree/main/src/system-state-recorder
[13] https://github.com/SovereignEdgeEU-COGNIT/ai-orchestrator/tree/main/src/system-state-recorder-ui

Watch a video showing the behaviour of this initial prototype of energy efficiency optimization for cloud-edge orchestration: https://vimeo.com/879130827