SERRANO

TRANSPARENT APPLICATION DEPLOYMENT IN A SECURE, ACCELERATED AND COGNITIVE CLOUD CONTINUUM

Grant Agreement no. 101017168

Deliverable D6.8
Final version of business, end user and technical evaluation

Programme:	H2020-ICT-2020-2
Project number:	101017168
Project acronym:	SERRANO
Start/End date:	01/01/2021 – 31/12/2023

Deliverable type:	Report
Related WP:	WP6
Responsible Editor:	INB
Due date:	31/12/2023
Actual submission date:	14/01/2024

Dissemination level:	Public
Revision:	Final

Revision History

Date	Editor	Status	Version	Changes
09.06.2023	INB	Draft	0.1	Initial version with Table of Contents
28.11.2023	INB	Draft	0.2	Update in contents
06.12.2023	UVT	Draft	0.2	Contribution to Section 3.4
12.12.2023	ICCS	Draft	0.2	Contribution to Section 3 and Section 3.3
14.12.2023	NBFC	Draft	0.2	Contribution to Section 4.3
19.12.2023	INB	Draft	0.3	Integrate and edit provided contributions
20.12.2023	ICCS	Draft	0.4	Contribution to Section 3.1, 3.2 and 3.4
20.12.2023	AUTH	Draft	0.4	Contribution to deliverable
20.12.2023	CC	Draft	0.4	Contribution to Section 4
22.12.2023	INB	Draft	0.5	Integrate the contributions from the partners
22.12.2023	INTRA	Draft	0.5	Contribution to Section 7.7
22.12.2023	IDEKO	Draft	0.5	
27.12.2023	INB	Draft	0.6	Updated summary
28.12.2023	INB	Draft	0.7	Updated Section 5.2
02.01.2024	INB	Draft	0.8	Update input from UVT and AUTH.
03.01.2024	ICCS	Draft	0.9	Updated Section 3 and 7
07.01.2024	INB	Draft	1.0	Update Section 5. Fix internal refences to figures, tables and sections.
10.01.2024	INB	Draft	1.1	Fix formatting.
12.01.2024	INB	Final	1.2	Address review comments.

Author List

Organisation	Author
INB	Maria Oikonomidou, Ferad Zyulkyarov
UVT	Gabriel Iuhasz, Adrian Spataru
ICCS	Aristotelis Kretsis, Panagiotis Kokkinos, Emmanouel Varvarigos, P. Makris, K. Steriotis, V. Zagorakis
NBFC	Anastasios Nanos
INNOV	Efstathios Karanastasis, Efthymios Chondrogiannis, Andreas Litke, Filia Filippou
AUTH	Dimosthenis Masouros, Argyris Kokkinis, Kostas Siozios
CC	Marton Sipos, Marcell Fehér, Daniel E. Lucani
INTRA	Makis Karadimas, Paraskevas Bourgos
IDEKO	Julen Aperribay, Javier Martín, Aitor Fernández

Internal Reviewers

Aitor Fernández, Javier Martín (IDEKO)
Kassie Papasotiriou, Stelios Pantelopoulos (INNOV)

Abstract: This deliverable (D6.8) is the second of two reports that are scheduled to present the business, end user and technical evaluation outcomes of the tasks 6.3, 6.4 and 6.5. It presents the final integration, deployment and the evaluation of the three SERRANO use cases (i.e., Secure Storage, FinTech Analysis, and Anomaly Detection in Manufacturing Settings) into the SERRANO platform, which we refer to as platform and use cases demo. The activities related to demonstrating the use case applications started with their adaptation for the SERRANO platform. These adaptation and preliminary integration and deployment activities were reported in D6.4. This report continues with the final integration, deployment, evaluation and fine tuning of the use case applications running on the SERRANO platform. It includes the final results obtained from the production-ready demos deployed and executed on the cloud cluster running the SERRANO platform.

Keywords: SERRANO platform, integrated components, development environment, integration environment, software deployment, verification, validation

Table of Contents

List of Figures

List of Tables

Abbreviations

A4C	Alien4Cloud
AES	Advanced Encryption Standard
AI	Artificial Intelligence
AISO	AI-enhanced Service Orchestrator
API	Abstract Programming Interface
ARDIA	A Resource reference model for Data-Intensive Applications
AWS	Amazon Web Service
BOM	Bill of Materials
CI/CD	Continuous Integration / Continuous Development
CPU	Central Processing Unit
CTH	Central Telemetry Handler
D	Deliverable
DevSecOps	Development, Security, and Operations
DI	Deployment Intent
DL	Deep Learning
DPO	Dynamic Portfolio Optimisation
DPU	Data Processing Unit
DTW	Dynamic Time Warping
EDE	Event Detection Engine
EFT	Electronic Funds Transfer
EOL	End of Life
ETA	Enhanced Telemetry Agent
ETF	Exchange-Traded Fund
ETL	Extract, Transform, Load
FFT	Fast Fourier transform
FLOPS	Floating-point Operations per Second
FPGA	Field-Programmable Gate Array
FUSE	Filesystem in USErspace
GB	Gigabyte
GCM	Galois/Counter Mode
GDPR	General Data Protection Regulation
GF	Galois Fields
GPU	Graphics Processing Unit
HPC	High Performance Computing
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HW	Hardware
ID	Identifier
IO	Input Output
IoT	Internet of Things
JSON	JavaScript Object Notation
K8s	Kubernetes
KNN	K-Nearest Neighbours algorithm

KPI	Key Performance Indicator
kTLS	Kernel Transport Layer Security
LAN	Local Area Network
MB	Megabyte
ML	Machine Learning
MQTT	MQ (IBM MQ) Telemetry Transport
ms	Millisecond
NIC	Network Interface Controller
OS	Operating system
PCIe	Peripheral Component Interconnect express
PMDS	Persistent Monitoring Data Storage
RAM	Random Access Memory
REST	Representational State Transfer
RO	Resource Orchestrator
ROT	Resource Orchestration Toolkit
SaaS	Software as a Service
SAR	Service Assurance and Remediation
SDK	Service Development Kit
SOP	SERRANO Orchestrator Plugin
SSL	Secure Sockets Layer
TLS	Transport Layer Security
TOSCA	Topology and Orchestration Specification for Cloud Applications
UC	Use Case
UI	User Interface
URL	Uniform Resource Locator
UID	Universally Unique Identifier
VM	Virtual Machine
XRT	Xilinx Runtime Environment
YAML	YAML Ain't Markup Language

1 Executive Summary

Deliverable D6.8 “Final version of business, end user and technical evaluation” reports on the final integrations and the evaluation of the three SERRANO use cases. This deliverable is a follow up from D6.4, which reported about the integration of the platform that was in progress and contained preliminary results. D6.8 presents demos of the SERRANO platform as well as use cases demos. It evaluates 1) the Secure Storage Use Case that aims at demonstrating the envisioned capabilities of the SERRANO platform in the context of secure file sharing and storage, 2) the Fintech Use Case that leverage the cloud continuum capabilities of the SERRANO project within the context of investment portfolio management, representing automated management of investment portfolios, and 3) the Anomaly Detection in Manufacturing Settings Use Case that aims to diagnosis critical machine elements by proposing an approach where data analysis is performed continuously on the SERRANO platform.

The deliverable elaborates about activities related to fine tuning and optimisation of the use case applications to run at scale. It presents the results from the evaluation of the final version of the SERRANO platform in KPI tables. The production or final version of the SERRANO platform integrates all the components and subsystems from the second development iteration (M20-M36). Technical details about the final version of the SERRANO platform are reported on D6.7 (M36).

2 Introduction

2.1 Purpose of this document

The purpose of this deliverable is to report on the final integration and evaluation of the SERRANO use cases Secure Storage (Task 6.3), FinTech (Task 6.4), and Anomaly Detection (Task 6.5) into the SERRANO platform. Since the first report D6.4 (M20), the use case partners worked closely with the integration partners to optimise and fine tune the use case applications for the SERRANO platform. The document reports about their integration and optimisation experience as well as the detailed technical and business evaluation of the use case applications. It includes the description of SERRANO platforms demos which represent the final integration of all components in a production ready system. The evaluation results obtained from the demo are presented in dedicated KPI tables. The document elaborates the benefits and the challenges of developing applications for the SERRANO platform, and provides feedback about the business implications.

2.2 Document structure

The present deliverable is split into eight major sections:

- Section 1 is an executive summary.
- Section 2 serves as an introduction and describes the main goals of the deliverable.
- Section 3 describes the SERRANO platform demos.
- Section 4 is dedicated to Secure Storage Use Case (CC) and describes the Secure Storage demo and the evaluation results.
- Section 5 is dedicated to FinTech Use Case (INB) and describes the FinTech demo and the evaluation results.
- Section 6 is dedicated to Anomaly Detection in Manufacturing Settings Use Case (IDEKO) and describes the use case demo and evaluation results.
- Section 7 presents the evaluation results for the SERRANO platform.
- Section 8 briefly summarises the evaluation of all Use Cases.

2.3 Audience

The deliverable is public and available to anyone interested in the final release of the SERRANO integrated platform and the SERRANO use cases. Moreover, this document can also be useful to the general public for a better understanding of the framework and scope of the SERRANO project.

3 SERRANO Platform Demo

SERRANO adopts a lifecycle methodology to facilitate the seamless application deployment and cognitive resource orchestration across the distributed and heterogeneous edge, cloud, and HPC continuum. Initially, users provide their applications along with a high-level infrastructure agnostic (application intent) description of their requirements (*step a*). Next, SERRANO performs the application profiling and decompose the high-level requirements into specific service goals (*step b*). Then, the SERRANO cognitive orchestration mechanisms allocate and deploy the application's microservices into the available resources (*step c*). Finally, the service assurance mechanisms based on real-time telemetry and appropriate machine reasoning techniques ensure that applications perform as intended (*step d*) while triggering any required re-optimisation.

According to the SERRANO architecture [1], these steps are organised into three main control flow phases:

- Application description and high-level requirements translation (*step a and b*)
- Cognitive resource orchestration and transparent deployment (*step c*)
- Service assurance and dynamic adjustments (*step d*)

This section complements the comprehensive evaluation of the SERRANO platform conducted via project use cases (Sections 4, 5, 6). It includes a series of compelling demonstrators that showcase the project's technological developments, emphasizing key components and services of the SERRANO platform. The evaluation demonstrators, namely Demo-1, Demo-2, Demo-3, Demo-4, within this section focus both on the component-to-component interactions and application-to-system ones. Table 1 summarises the platform evaluation demonstrators and outlines the corresponding SERRANO architecture control phases and lifecycle steps that encompass.

Table 1: Platform evaluation demonstrators and SERRANO architecture control phases.

Phase	Demo-1	Demo-2	Demo-3	Demo-4
Application description and high-level requirements translation.	✓		✓	
Cognitive resource orchestration and transparent deployment.	✓	✓	✓	✓
Service assurance and dynamic adjustments.				✓

3.1 Demo-1: Intent-driven operation and transparent application deployment

Description: The demonstration evaluates the ability of the SERRANO platform to support the transparent deployment of cloud-native applications across the heterogeneous edge and cloud platforms that it unifies. We aim to showcase the provision of an abstraction layer that automates the operation and fully exploits the available diverse resources, supporting a develop once, deploy everywhere approach. The users describe their applications through a neat web-based user interface as well as provide an infrastructure-agnostic description (i.e., the deployment intent) for their deployment objectives. SERRANO orchestration mechanisms initially map the intent to infrastructure-specific parameters and cognitively assign the application workload to available edge and cloud resources. Then, through the developed deployment and telemetry mechanisms, the SERRANO platform coordinates the seamless application deployment and the automatic monitoring of the deployed microservices across the SERRANO platform. For all these operations, the demonstrator utilises the SERRANO SDK.

SERRANO services and components:

- Alien4Cloud UI (A4C) and SERRANO Orchestration Plugin (SOP)
- AI-enhanced Service Orchestrator (AISO) and ARDIA Framework Abstraction Models
- Resource Orchestrator (RO) and Orchestration Drivers
- Resource Optimisation Toolkit (ROT)
- Data Broker
- Telemetry Service:
  - Central Telemetry Handler (CTH)
  - Enhanced Telemetry Agent (ETA)
  - Edge Storage Probe and K8s Probe
  - Persistent Monitoring Data Storage (PMDS)
- SERRANO lightweight virtualisation mechanisms

KPIs measured/evaluated: GEN.1, GEN.2, GEN.3, TEL.1, TEL.4, TEL.6, RES.1, RES.2, RES.7, SRV.1, SRV.2, SRV.3, SRV.4, SRV.5, SRV.7, SRV.8, INT.1, INT.6

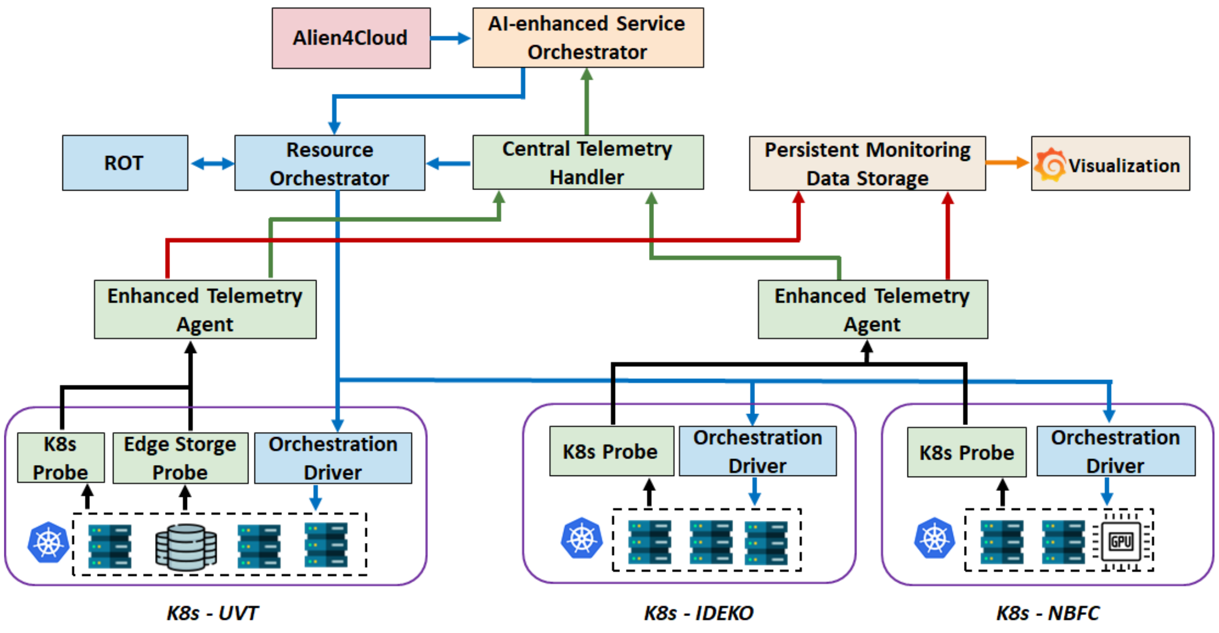


Figure 1: SERRANO testbed setup for platform evaluation Demo-1.

Scenario Description:

The demonstration scenario encompasses three Kubernetes (K8s) clusters, each characterized by distinct features, all seamlessly integrated through the SERRANO Resource Orchestrator (*GEN.1*). Table 2 summarises their main characteristics. Additionally, the setup includes four (4) edge storage locations, leveraging SERRANO edge storage devices deployed in the UVT K8s cluster. Moreover, the SERRANO telemetry mechanisms automatically capture performance metrics from both the available infrastructure resources and deployed applications (*TEL.1, TEL.6*). These metrics are subsequently processed and stored (*TEL.4*), facilitating orchestration decisions (SRV.3, SRV8). The telemetry data are used by many services within the SERRANO platform, including the AI-enhanced Service Orchestrator, Resource Orchestrator, Resource Optimisation Toolkit, and Visualisation component.

Table 2: K8s clusters within the SERRANO platform for Demo-1.

Name	Location	Characteristics
K8s - UVT	UVT premises (Timisoara, Romania)	3 x worker nodes, total CPU cores: 48, total RAM: 202 GB, total disk space: 624 GB
K8s - NBFC	NBFC premises (Athens, Greece)	5 x worker nodes, total CPU cores: 30, total RAM: 100 GB, total disk space: 1470 GB, availability of hardware acceleration (GPUs)
K8s – IDEKO	IDEKO premises (Elgoibar, Spain)	3 x worker nodes, total CPU cores: 12, total RAM: 14 GB, total disk space: 610 GB

In this demonstration scenario, we focus on Position Service from the Anomaly Detection in Manufacturing Settings use case (Section 6), which includes three microservices (*GEN.2, GEN.3, INT.1*). Two different Deployment Intents (DI) are defined (*SRV.1*) to showcase the intent-driven operation (*SRV.7*) and transparent application deployment within the SERRANO platform (Table 3). The deployment objectives, within these intents, provide high-level descriptions for the deployment scopes that guide their automatic mapping (*SRV.3, SRV.4, SRV.5*) to actual infrastructure-specific deployment scenarios/objectives (*SRV.2*) from the AISO (*SRV.8*). The provided deployment scenarios are then used for the actual orchestration and deployment of the application microservices (*RES.1, RES.2, RES.7*). For all these operations, we utilise the SERRANO SDK (*INT.6*).

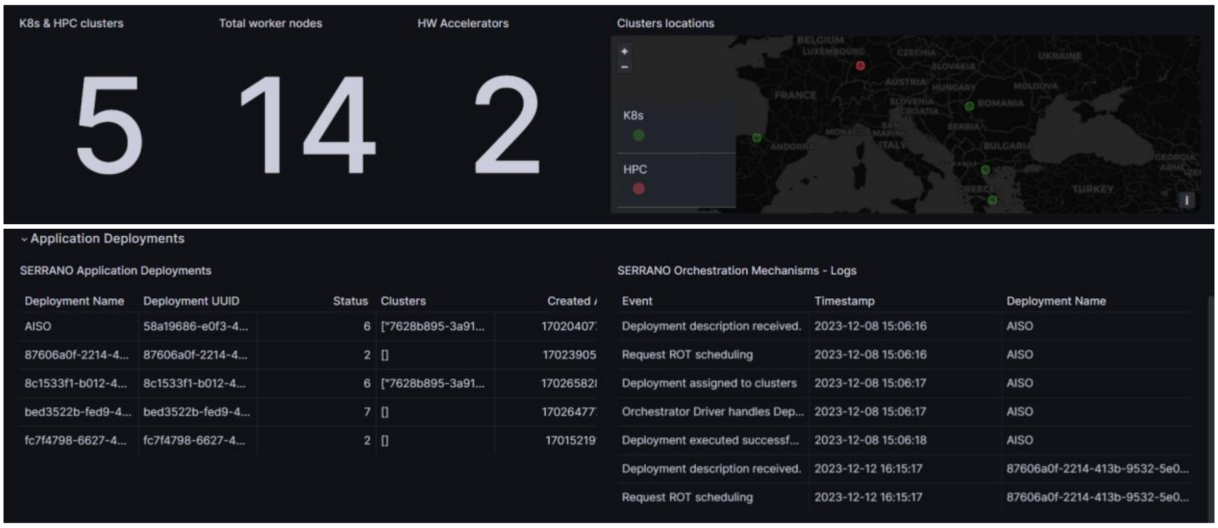
To this end, this demonstration scenario includes the following phases:

1. Application and Deployment Objectives Description: Users interact with the A4C's web-based interface to provide the application description and deployment objectives.
2. Translation of Deployment Objectives: The AI-enhanced Service Orchestrator translates the provided deployment objectives into infrastructure-specific deployment scenarios.
3. High-level Resource Orchestration: The Resource Orchestrator assigns application microservices to available platforms through the Resource Optimisation Toolkit (ROT).
4. Transparent Application Deployment: The Resource Orchestrator, supported by its Orchestration Drivers, ensures the transparent deployment of applications.

Table 3: Deployment intents for Demo-1.

ID	Description
DI1.1	The total execution time and the response latency of the "***position-service-classifier-training***" service should be as low as possible
DI1.2	The total energy consumption of the three micro-services should be as low as possible

In addition, a Grafana dashboard (Figure 2) dynamically displays real-time data sourced from the SERRANO telemetry services (*TEL.1, TEL.4*). It also presents relevant events and logs from the SERRANO orchestration mechanisms (*RES.1, RES.2, RES.7*). These events and logs are related to the orchestration decisions and actual deployment of user applications within the individual edge and cloud K8s clusters.



SERRANO Application Deployments

Deployment Name	Deployment UUID	Status	Clusters	Created /
AISO	58a19686-e0f3-4...	6	["7628b895-3a91...	17020407
87606a0f-2214-4...	87606a0f-2214-4...	2	[]	17023905
8c1533f1-b012-4...	8c1533f1-b012-4...	6	["7628b895-3a91...	17026582
bed3522b-fed9-4...	bed3522b-fed9-4...	7	[]	17026477
fc7f4798-6627-4...	fc7f4798-6627-4...	2	[]	17015219

SERRANO Orchestration Mechanisms - Logs

Event	Timestamp	Deployment Name
Deployment description received.	2023-12-08 15:06:16	AISO
Request ROT scheduling	2023-12-08 15:06:16	AISO
Deployment assigned to clusters	2023-12-08 15:06:17	AISO
Orchestrator Driver handles Dep...	2023-12-08 15:06:17	AISO
Deployment executed successf...	2023-12-08 15:06:18	AISO
Deployment description received.	2023-12-12 16:15:17	87606a0f-2214-413b-9532-5e0...
Request ROT scheduling	2023-12-12 16:15:17	87606a0f-2214-413b-9532-5e0...

Figure 2: Grafana dashboard for Demo-1.

<u>Application and deployment objectives description</u>

The Alien4Cloud (A4C) platform has been configured to use the SERRANO Orchestrator Plugin (SOP) and SERRANO-TOSCA extension. The TOSCA extension allows for the definition of applications using components that are packaged as container images (*INT.1*) to which an intent can be attached (*SRV.1*). The Orchestrator Plugin is used to deploy an application on the SERRANO platform and achieves this using the following steps:

1. It generates the Kubernetes descriptors for the components based on the parameters defined in the corresponding config map fields. This includes the generation and attachment of volumes defined using the TOSCA extension in Alien4Cloud.
2. It translates the intent from the TOSCA specification to the AISO model. It includes the Kubernetes descriptors generated at the previous step in the request for the AISO.
3. Contacts the AISO, which creates deployment objectives based on the intent, and contacts in turn the Resource Orchestrator to handle the execution. The deployment unique identifier (UUID) is passed back to the A4C Orchestrator plugin.
4. The deployment UUID is used to inspect the status of the application components and logs related to their deployment.

The A4C Topology Editor interface has been used to generate the application presented in Figure 3. When generating the Kubernetes descriptors, the Config Maps of the components are updated to use the correct IP and port of the SERRANO core components (i.e., Secure Storage and Data Broker).

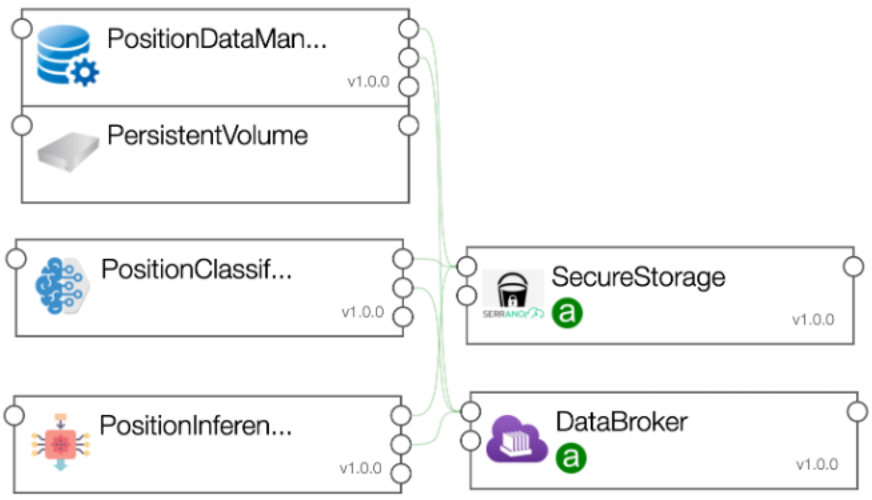
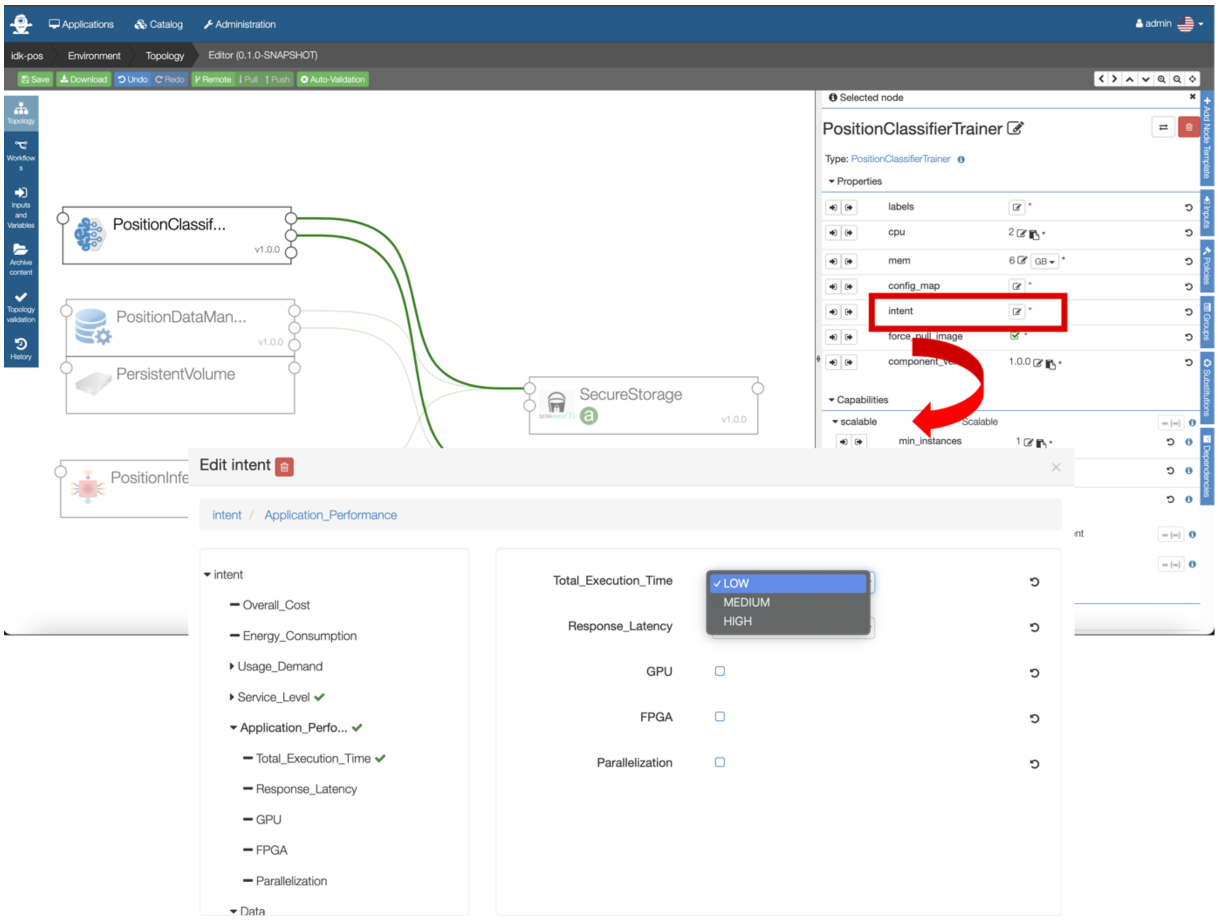


Figure 3: Position Service defined visualised in Alien4Cloud Topology Editor.



PositionClassifierTrainer

Type: PositionClassifierTrainer

Properties

labels	
cpu	2
mem	6 GB
config_map	
intent	
force_pull_image	✓
component_version	1.0.0

Capabilities

scalable — Scalable

min_instances 1

Edit intent

intent / Application_Performance

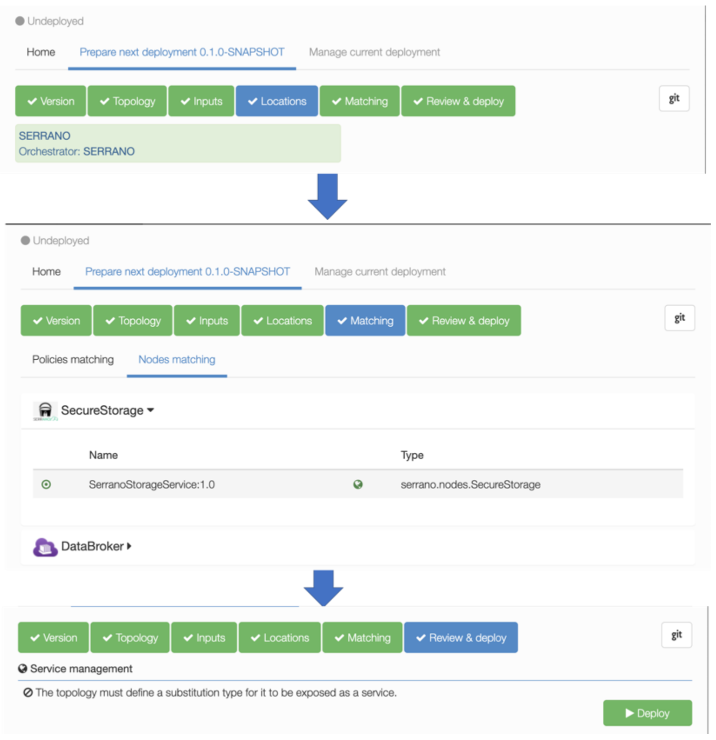
- intent
  - Overall_Cost
  - Energy_Consumption
  - Usage_Demand
  - Service_Level ✓
  - Application_Perfo... ✓
    - Total_Execution_Time ✓
    - Response_Latency
    - GPU
    - FPGA
    - Parallelization
  - Data

Total_Execution_Time	[✓ LOW] MEDIUM HIGH
Response_Latency	
GPU	☐
FPGA	☐
Parallelization	☐

Figure 4: Intent configuration in Alien4Cloud Topology Editor.

The user can click on one of the components and select the intent field to open the intent configuration dialogue. Figure 4 presents how to find and open this dialogue for the Classifier Training component of the application, which is the most computationally intensive. In this case, the user selects a ‘LOW’ value for the Total Execution Time parameter under the Application Performance category (*ID1.1*). The user can see the different categories for intent configuration on the left side of the dialogue.

There are several small steps that the user needs to get through before final deployment. These are presented in Figure 5. The first step is matching the dependencies, more specifically, the Secure Storage and Data Broker services. These are registered in the Alien4Cloud platform and are automatically selected when choosing the SERRANO location. The Config Maps of the three containers will be generated with the endpoints of the services registered in Alien4Cloud. The final action is to click the Deploy Button.



SERRANO
Orchestrator: SERRANO

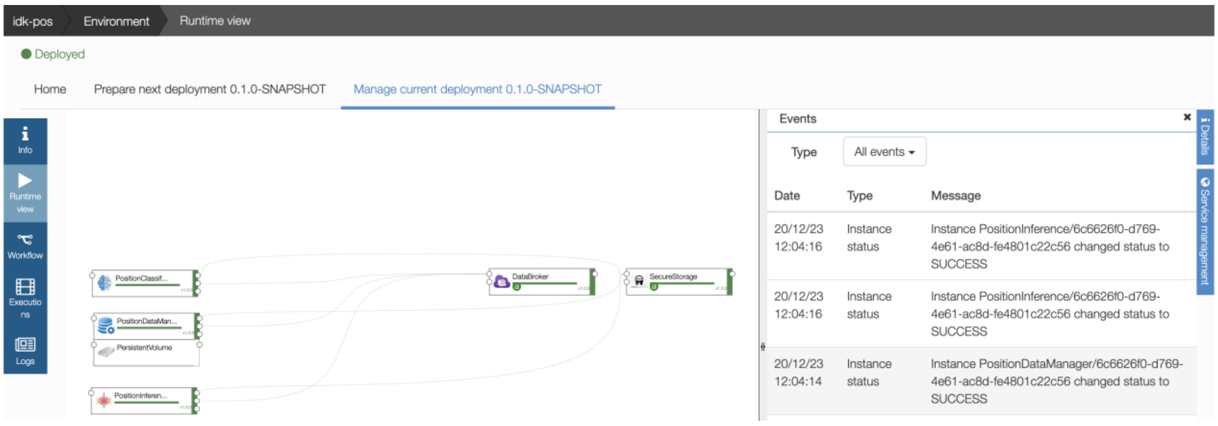
	Name		Type
⊙	SerranoStorageService:1.0	●	serrano.nodes.SecureStorage

The topology must define a substitution type for it to be exposed as a service.

Figure 5: Location selection, abstract service matching and deploy button.

Once the deploy button has been clicked the Orchestrator Plugin generates the Kubernetes deployment descriptors (YAML) for each component, updating all fields related to the Secure Storage and Data Broker components. After this, the plugin generates the intent request, adding to it the generated Kubernetes descriptors. Finally, this request is sent to the AISO (via a JSON message), which returns the deployment ID after successfully contacting the Resource Orchestrator.

The deployment ID is used to retrieve the status of the deployment. The user can inspect the status of the components in the Runtime View of the current deployment, as presented in Figure 6. Each component will be coloured based on its status: red if failed, orange if processing, and green if deployed. Moreover, the right sidebar presents the events related to these components from the status point of view.



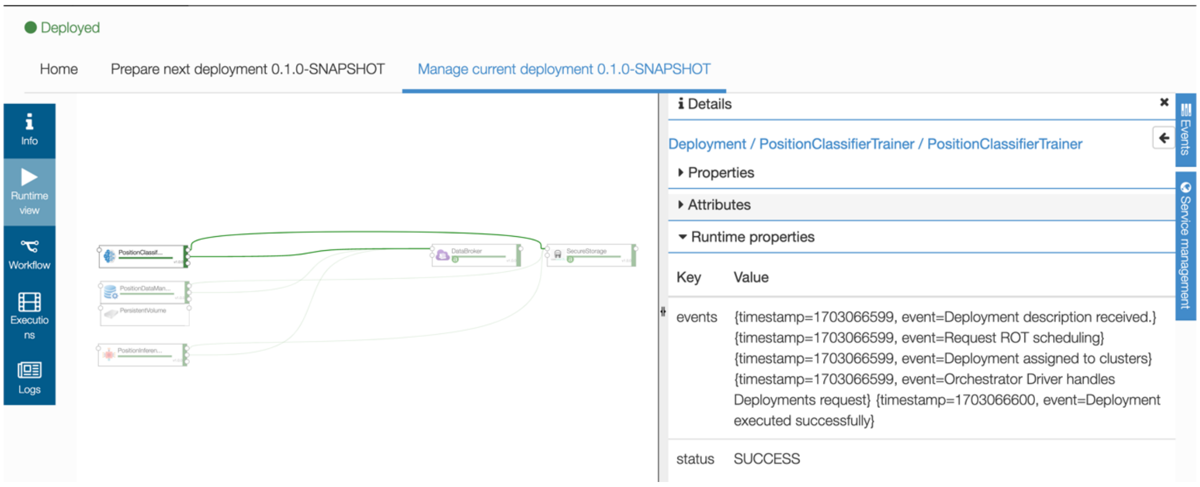
Events

Type: All events

Date	Type	Message
20/12/23 12:04:16	Instance status	Instance PositionInference/6c6626f0-d769-4e61-ac8d-fe4801c22c56 changed status to SUCCESS
20/12/23 12:04:16	Instance status	Instance PositionInference/6c6626f0-d769-4e61-ac8d-fe4801c22c56 changed status to SUCCESS
20/12/23 12:04:14	Instance status	Instance PositionDataManager/6c6626f0-d769-4e61-ac8d-fe4801c22c56 changed status to SUCCESS

Figure 6: Runtime View for the deployed application in Alien4Cloud.

Finally, the user can check the SERRANO-related logs by selecting a specific component and exploring its runtime properties. An example is presented in Figure 7, focusing on the classifier training component. The "events" field provides a comprehensive view of the request's journey within the SERRANO platform, tracking its path until it reaches the designated Orchestrator Driver determined by ROT. This Orchestrator Driver then executes the deployment using the Kubernetes descriptors generated by the A4C Orchestrator Plugin.



Details

Deployment / PositionClassifierTrainer / PositionClassifierTrainer

Properties

Attributes

Runtime properties

Key	Value
events	{timestamp=1703066599, event=Deployment description received.} {timestamp=1703066599, event=Request ROT scheduling} {timestamp=1703066599, event=Deployment assigned to clusters} {timestamp=1703066599, event=Orchestrator Driver handles Deployments request} {timestamp=1703066600, event=Deployment executed successfully}
status	SUCCESS

Figure 7: Runtime properties inspection for SERRANO deployed components.

High-level requirements translation

The AI-enhanced Service Orchestrator (AISO) receives as input the JSON description produced by the SERRANO Orchestrator Plugin (SOP) through the Alien4Cloud platform. The description follows the predefined format (as specified in the AISO Open API) that has been developed based on the elements of the Application Model (part of the ARDIA framework). It includes the deployment descriptor (i.e., application YAML), and the parameters expressing the particular application and/or user requirements (i.e., goals/intents) (*SRV.1*).

The AISO then processes the data provided to suggest suitable resources (from the available ones) for deploying the application micro-services so that the application requirements and user's goals or intents are satisfied. For this purpose, the AISO uses the relevant Mapping Rules from the pool of available ones (examples provided in the deliverables D5.4 and D6.7, M31 and M36 respectively), and the collected telemetry data from the available resources and deployed applications (*SRV.4, SRV.5, and SRV.8*).

Table 4: Deployment intent parameters and output objectives for Demo 1.

ID	Deployment Intent Parameters	Deployment Objectives
DI1.1	{ "deployment_descriptor_yaml":"*application-YAML-as-a-String*", "application_constraints":[{ "component_id": position-service-classifier-training", "Application_Performance_Response_Latency":"LOW", "Application_Performance_Total_Execution_Time":"LOW"}], "application_workflow" :[{ "component_id": "position-service-data-manager" }, { "component_id": "position-service-classifier-training" }, { "component_id": "position-service-model-inference" }] }	{ "deployment_description": "*application-YAML-as-a-String*", "deployment_objectives": [{ "component_id": "position-service-model-inference", "node_type": ["EDGE"], "accelerator_type": ["GPU"] }, { "component_id": " position-service-classifier-training", "node_type": ["CLOUD"], "accelerator_type": ["GPU"] }], "name": "AISO" }
DI1.2	{ "deployment_descriptor_yaml":"*application-YAML-as-a-String*", "application_constraints": [{ "Energy_Consumption":"LOW"}], "application_workflow": [{ "component_id": "position-service-data-manager" }, { "component_id": "position-service-classifier-training" }, { "component_id": "position-service-model-inference" }] }	{ "deployment_description": "*application-YAML-as-a-String*", "deployment_objectives": [{ "component_id": "position-service-classifier-training", "node_type": ["EDGE"], "accelerator_type": ["GPU", "FPGA"]} , { "component_id": "position-service-model-inference", "node_type": ["EDGE"], "accelerator_type": ["GPU", "FPGA"]}], "name": "AISO" }

The translation of high-level objectives into deployment objectives for the two deployment scenarios is summarised in Table 4. For example, nodes with high-profile acceleration devices are favoured in the first case since the training process should be completed in a limited amount of time (i.e., execution time should be low). Moreover, the two microservices that handle data are split among edge and cloud resources to achieve the low response latency requirement. The "*position-service-model-inference*" that constantly analyses data from sensors is mapped to edge resources while the most computationally-intensive "*position-service-classifier-training*" service to cloud resources with more advanced acceleration capabilities.

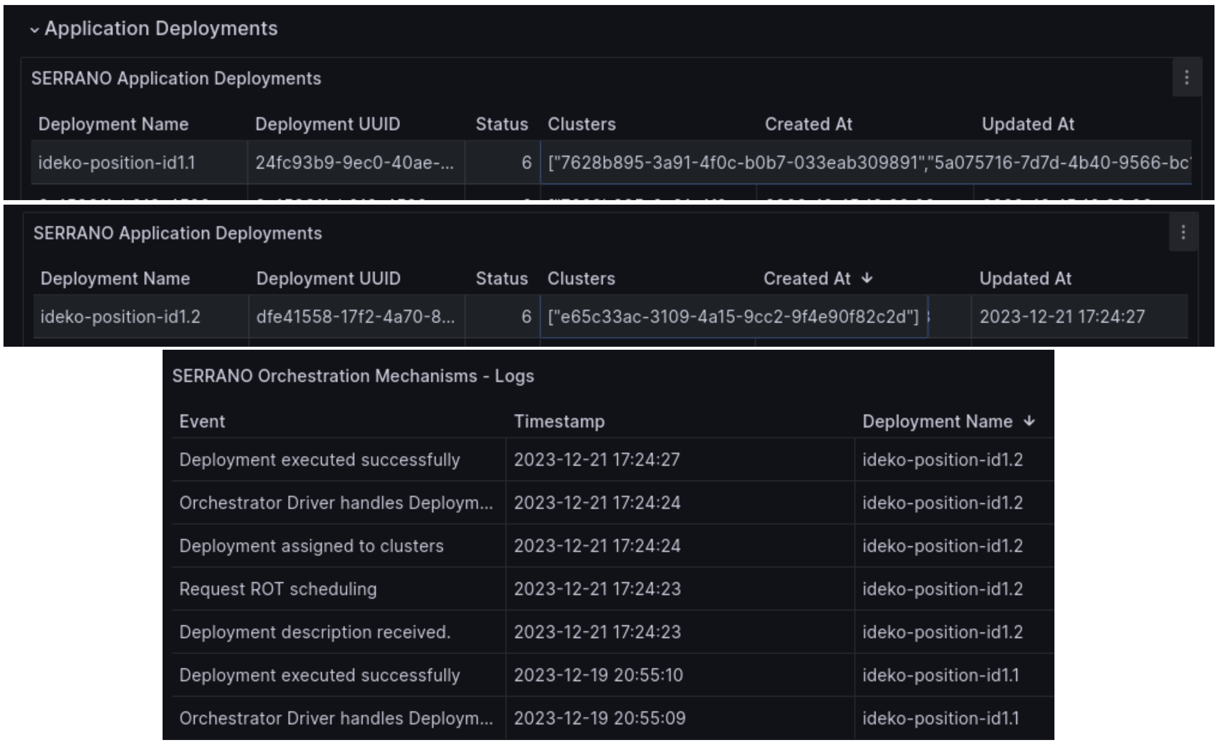
The AISO also interacts with the Central Telemetry Handler (CTH) to retrieve the available resources and their characteristics within the SERRANO platform (*SRV.3*) and accordingly selects the ones which satisfy the specific need (*SRV.7*). The previous decisions are driven by the telemetry data, which indicate that the average execution time of the training process is much lower when GPU accelerators on cloud nodes are used in comparison to the usage of GPUs in edge.

The output of the AISO is a JSON description with a predefined format (as also specified in the AISO Open API) created based on the elements of the Resource Model of the ARDIA Framework. It includes the given deployment descriptor (application YAML) and the particular objectives for the RO regarding the deployment of the microservices (*SRV.2*). The input and output of the AISO are presented in Table 4. The generated JSON description is provided to the Resource Orchestrator, which returns the deployment unique identifier (i.e., UUID) through which the deployed application microservices can be managed (i.e., check the current status of the deployment or un-deploy the micro-services of the given application, if necessary) using either the AISO or RO services, as already described in the deliverable D6.7 (M36).

<u>Resource orchestration and transparent application deployment</u>

The SERRANO Resource Orchestrator receives the request for the application deployment from the AISO and creates the appropriate SERRANO Deployment object that guides SERRANO orchestration mechanisms in making information decisions based on the provided application description and deployment objectives.

During the orchestration phase, the Resource Orchestrator requests the ROT (*RES.1*) to provide the necessary orchestration decision for the application deployment. The request description to the ROT Controller includes the deployment objectives, as provided by the AISO, and the application graph. The ROT aims to match the provided requirements with the most suitable cloud and edge platforms and resource configurations. Figure 8 shows details from the SERRANO orchestration mechanisms for the two deployment scenarios.



⌄ Application Deployments

SERRANO Application Deployments

Deployment Name	Deployment UUID	Status	Clusters	Created At	Updated At
ideko-position-id1.1	24fc93b9-9ec0-40ae-...	6	["7628b895-3a91-4f0c-b0b7-033eab309891","5a075716-7d7d-4b40-9566-bc		

SERRANO Application Deployments

Deployment Name	Deployment UUID	Status	Clusters	Created At ↓	Updated At
ideko-position-id1.2	dfe41558-17f2-4a70-8...	6	["e65c33ac-3109-4a15-9cc2-9f4e90f82c2d"]		2023-12-21 17:24:27

SERRANO Orchestration Mechanisms - Logs

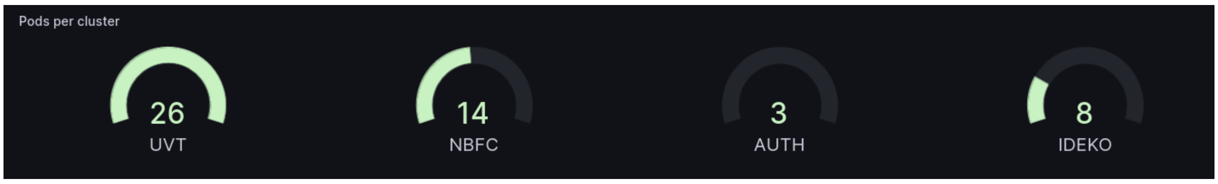
Event	Timestamp	Deployment Name ↓
Deployment executed successfully	2023-12-21 17:24:27	ideko-position-id1.2
Orchestrator Driver handles Deploym...	2023-12-21 17:24:24	ideko-position-id1.2
Deployment assigned to clusters	2023-12-21 17:24:24	ideko-position-id1.2
Request ROT scheduling	2023-12-21 17:24:23	ideko-position-id1.2
Deployment description received.	2023-12-21 17:24:23	ideko-position-id1.2
Deployment executed successfully	2023-12-19 20:55:10	ideko-position-id1.1
Orchestrator Driver handles Deploym...	2023-12-19 20:55:09	ideko-position-id1.1

Figure 8: SERRANO Deployment objects and orchestration mechanisms logs.

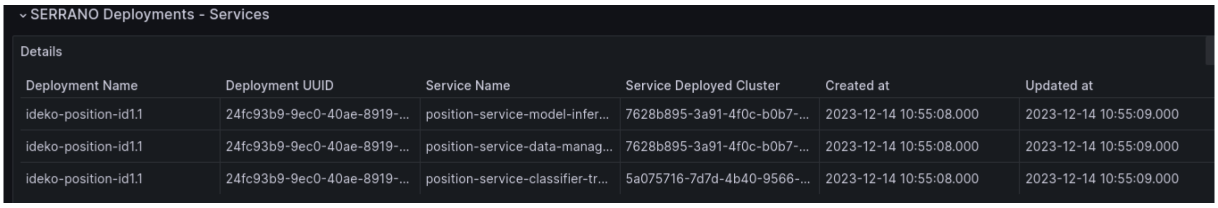
The Clusters column in the *"SERRANO Application Deployments"* table (Figure 8) presents the ROT's decision for the assignment of the application microservices to the available K8s clusters (*RES.2*). As can be noticed for the first deployment intent, the orchestration mechanisms selected to split the microservices into two K8s clusters (*"7628b895-3a91-4f0c-b0b7-033eab309891"*, "5a075716-7d7d-4b40-9566-bc1a33ee70c2"). In contrast, in the second scenario, all microservices were assigned to the same K8s cluster (*"e65c33ac-3109-4a15-9cc2-9f4e90f82c2d"*).

Next, the Resource Orchestrator initiates the final phase that handles the transparent application deployment across the SERRANO platform without any other user intervention. It provides to the Orchestration Drivers a declarative description of the workload requirements that will be used by the platform-level orchestration mechanisms for the final deployment decisions (*RES.7*). The Orchestration Drivers at the selected platforms receive the deployment instructions and coordinate the seamless workload deployment.

Next, we provide more details focused on in the first of the two deployment scenarios. Figure 9 shows details from the Grafana dashboard regarding the number of applications microservices each in K8s cluster before and after the deployment (Figure 9a) as well as details about the specific deployment microservices (Figure 9b).



(a)


SERRANO Deployments - Services
Details

Deployment Name	Deployment UUID	Service Name	Service Deployed Cluster	Created at	Updated at
ideko-position-id1.1	24fc93b9-9ec0-40ae-8919-...	position-service-model-infer...	7628b895-3a91-4f0c-b0b7-...	2023-12-14 10:55:08.000	2023-12-14 10:55:09.000
ideko-position-id1.1	24fc93b9-9ec0-40ae-8919-...	position-service-data-manag...	7628b895-3a91-4f0c-b0b7-...	2023-12-14 10:55:08.000	2023-12-14 10:55:09.000
ideko-position-id1.1	24fc93b9-9ec0-40ae-8919-...	position-service-classifier-tr...	5a075716-7d7d-4b40-9566-...	2023-12-14 10:55:08.000	2023-12-14 10:55:09.000

(b)

Figure 9: Deployment of position service microservices for the first deployment scenario (DI1.1).

In addition, the orchestration mechanisms automatically configure (*TEL.6*) the SERRANO telemetry mechanisms to start the automatic monitoring of the deployed application across the selected platforms (*TEL.1*) and also register the deployed application to the Service Assurance mechanisms (Section 3.4). The collected telemetry data are also stored in the PMDS service (*TEL.4*). Figure 10 shows the memory usage by the application microservices as reported by the custom Grafana dashboard.

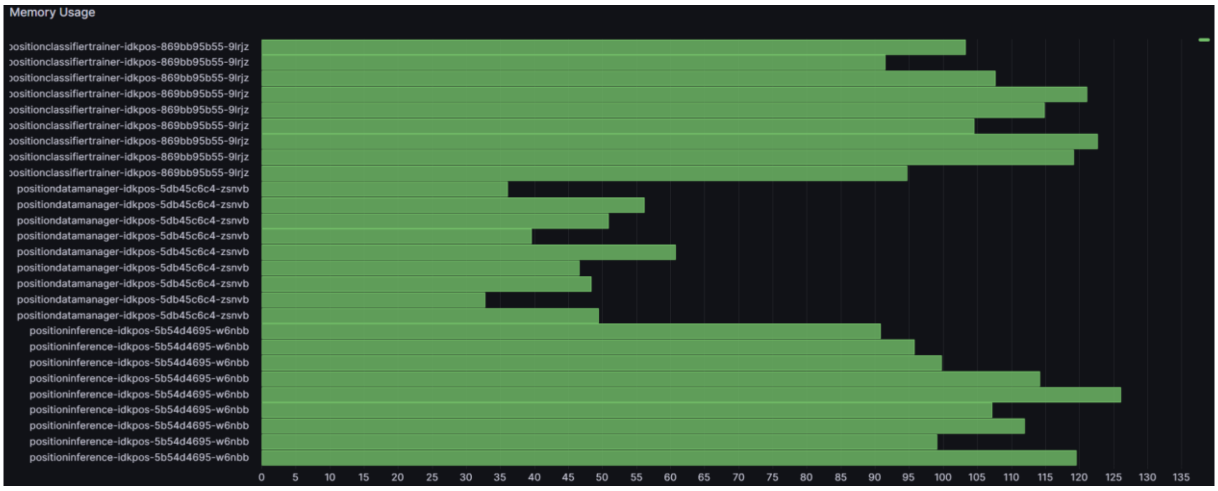
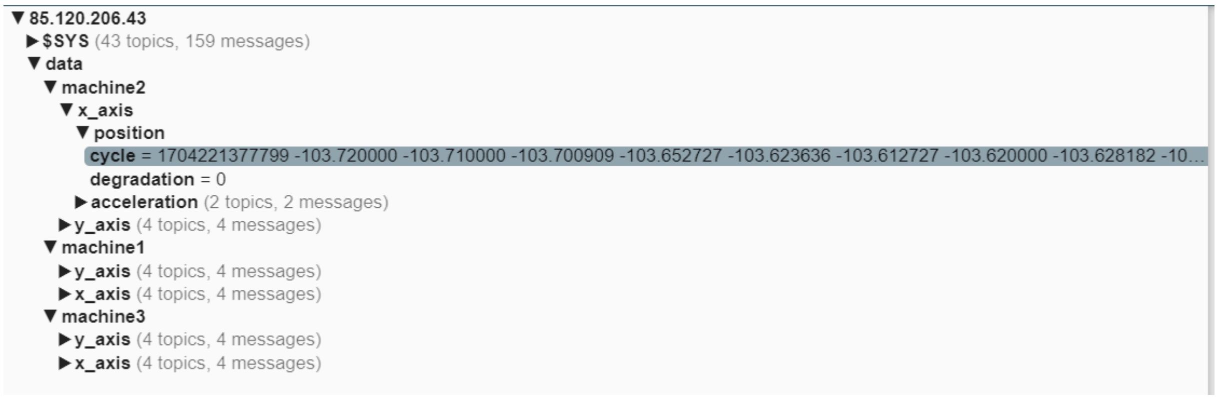


Figure 10: Memory usage of application microservices.

In order to verify the successful application deployment, we enable the corresponding machine ball screw simulator. The simulator service provides streaming data that triggers the execution of the SERRANO-accelerated kernels. More information for the simulator, the kernels, and the overall data workflow within the Position Service are available in deliverable D6.7 (M36). The following Figure 11 presents the data exchange through the MQTT interface between the three microservices of the service.



```
▼ 85.120.206.43
  ▶ $SYS (43 topics, 159 messages)
  ▼ data
    ▼ machine2
      ▼ x_axis
        ▼ position
          cycle = 1704221377799 -103.720000 -103.710000 -103.700909 -103.652727 -103.623636 -103.612727 -103.620000 -103.628182 -10…
          degradation = 0
        ▶ acceleration (2 topics, 2 messages)
      ▶ y_axis (4 topics, 4 messages)
    ▼ machine1
      ▶ y_axis (4 topics, 4 messages)
      ▶ x_axis (4 topics, 4 messages)
    ▼ machine3
      ▶ y_axis (4 topics, 4 messages)
      ▶ x_axis (4 topics, 4 messages)
```

Figure 11: Data exchange through the MQTT among the microservices of the Position Service application.

Finally, a video for this demonstration is available in the project's YouTube channel in the following link: https://www.youtube.com/@serranoproject7470

3.2 Demo-2: On-demand seamless execution of SERRANO-accelerated kernels

Description: This demonstration validates the on-demand seamless execution of SERRANO hardware- and software-accelerated kernels within heterogeneous hardware resources (i.e., FPGA, GPU, HPC) across the federated edge, cloud, and HPC SERRANO platform. We aim to showcase the overall integration to seamlessly deploy SERRANO-accelerated kernels across heterogeneous hardware resources. The users describe their execution requests through a web-based user interface, select the input data from predefined datasets, and provide deployment objectives. SERRANO orchestration mechanisms decide the execution platform, while the SERRANO deployment mechanisms coordinate the automatic data movement and kernel execution. The demonstrator shows that different input requirements led to different deployment configurations along with the performance evaluation for their end-to-end execution. For all these operations, we utilise the SERRANO SDK (*INT.6*).

SERRANO services and components:

- SERRANO accelerated kernels
- vAccel and SERRANO lightweight virtualisation mechanisms
- SERRANO HPC Gateway
- Data Broker & Secure storage service
- Resource Orchestrator (RO) & Orchestration Drivers
- Resource Optimisation Toolkit (ROT)
- Telemetry Service:
    - Central Telemetry Handler (CTH)
    - Enhanced Telemetry Agent (ETA)
    - K8S Probe and HPC Probe
    - Persistent Monitoring Data Storage (PMDS)

KPI measured/evaluated: GEN.1, GEN.2, GEN.4, ACC.4, ACC.5, TEL.1, TEL.4, RES.1, RES.2, RES.7, INT.1, INT.6

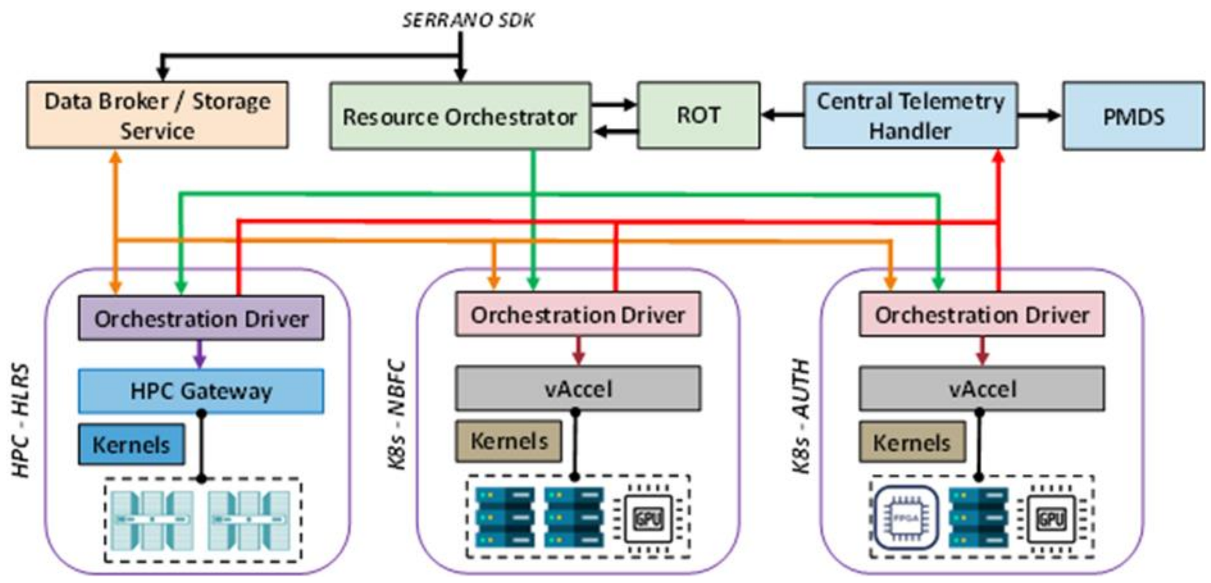


Figure 12: SERRANO testbed setup for platform evaluation demo-2.

Scenario Description:

The demonstration scenario includes the two Kubernetes clusters in the SERRANO testbed that provide hardware acceleration capabilities. SERRANO partners NBFC and AUTH offer these testbeds, including GPU and FPGA devices. The clusters also encompass SERRANO technological developments such as the vAccel and lightweight virtualisation mechanisms that enable flexible and interoperable hardware acceleration by abstracting the hardware-specific implementation and integration details (*GEN.4, INT.6*). In addition, there is available the HPC platform, provided in SERRANO by HLRS, that provides enormous capacity for computationally intensive and data analysis tasks. This exceptional unification of highly diverse resources allows the SERRANO platform to cater to application constraints and deployment objectives while calibrating the configuration of available resources (*GEN.1, GEN.2*).

The demonstration scenario uses the library of SERRANO-accelerated kernels (Table 5). They leverage both hardware and software acceleration techniques to enhance applications' performance and energy efficiency on cloud and edge devices, such as GPUs, FPGAs and HPC platforms (*ACC.4, ACC.5*).

Table 5: SERRANO-accelerated kernels and their supported execution platforms for Demo-2.

Kernel Name	GPU Acceleration	FPGA Acceleration	HPC Acceleration
KNN	✓		
KMEANS	✓		✓
FFT			✓
KALMAN		✓	✓
SAVGOL	✓		✓

In addition, a Grafana dashboard (Figure 13) dynamically displays real-time data sourced from the SERRANO telemetry services (*TEL.1, TEL.4*). It also presents relevant events and logs from the SERRANO orchestration mechanisms. These events and logs are related to orchestration and deployment actions by the SERRANO services for the on-demand execution of the SERRANO-accelerated kernels. Moreover, the dashboard shows the collected metric from the kernels' executions and various statistics.

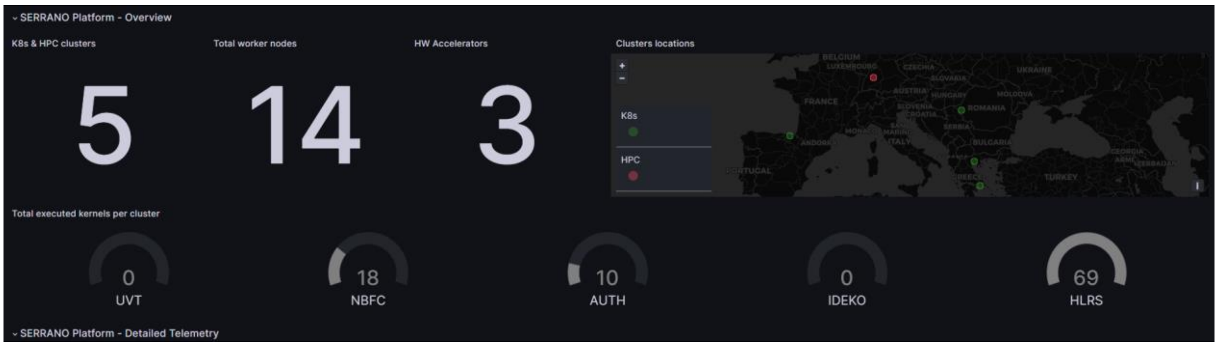
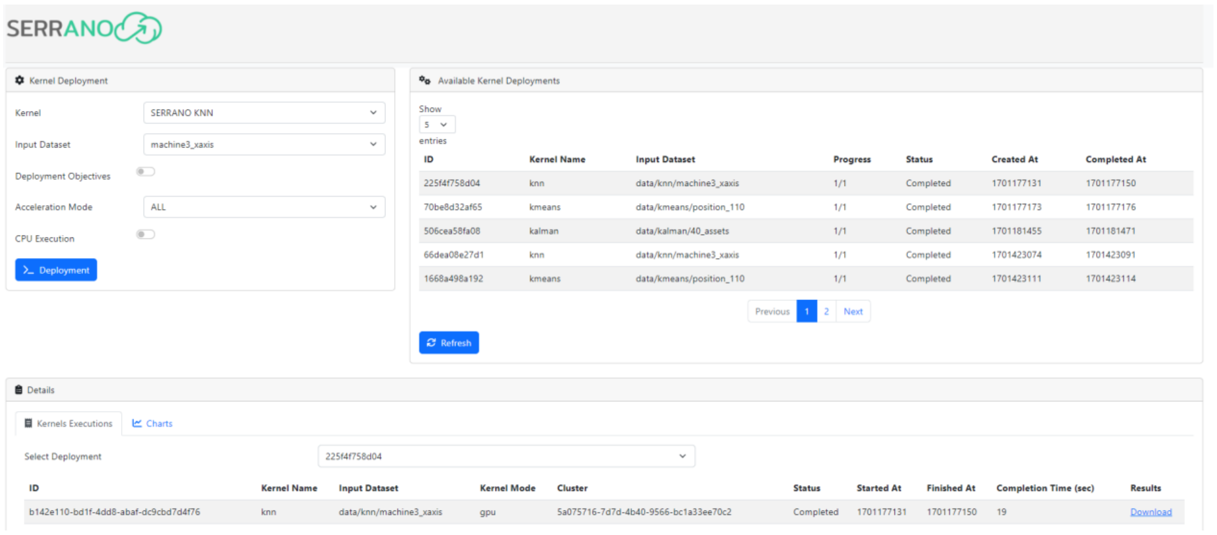


Figure 13: Grafana dashboard for Demo-2.

A web-based application written in Python that facilitates the demonstration is also available (Figure 14). The application is deployed within the UVT K8s cluster. It utilises the SERRANO SDK (*INT.6*) to request the on-demand execution of the SERRANO-accelerated kernels. The application also provides charts to visualize the detailed monitoring of the performance of the executed kernels that are automatically collected and stored by the SERRANO telemetry mechanisms (*TEL.1, TEL.4*).



Kernel Deployment

Kernel	SERRANO KNN
Input Dataset	machine3_xaxis
Deployment Objectives	
Acceleration Mode	ALL
CPU Execution	

Available Kernel Deployments

Show 5 entries

ID	Kernel Name	Input Dataset	Progress	Status	Created At	Completed At
225f4f758d04	knn	data/knn/machine3_xaxis	1/1	Completed	1701177131	1701177150
70be8d32af65	kmeans	data/kmeans/position_110	1/1	Completed	1701177173	1701177176
506cea58fa08	kalman	data/kalman/40_assets	1/1	Completed	1701181455	1701181471
66dea08e27d1	knn	data/knn/machine3_xaxis	1/1	Completed	1701423074	1701423091
1668a498a192	kmeans	data/kmeans/position_110	1/1	Completed	1701423111	1701423114

Details

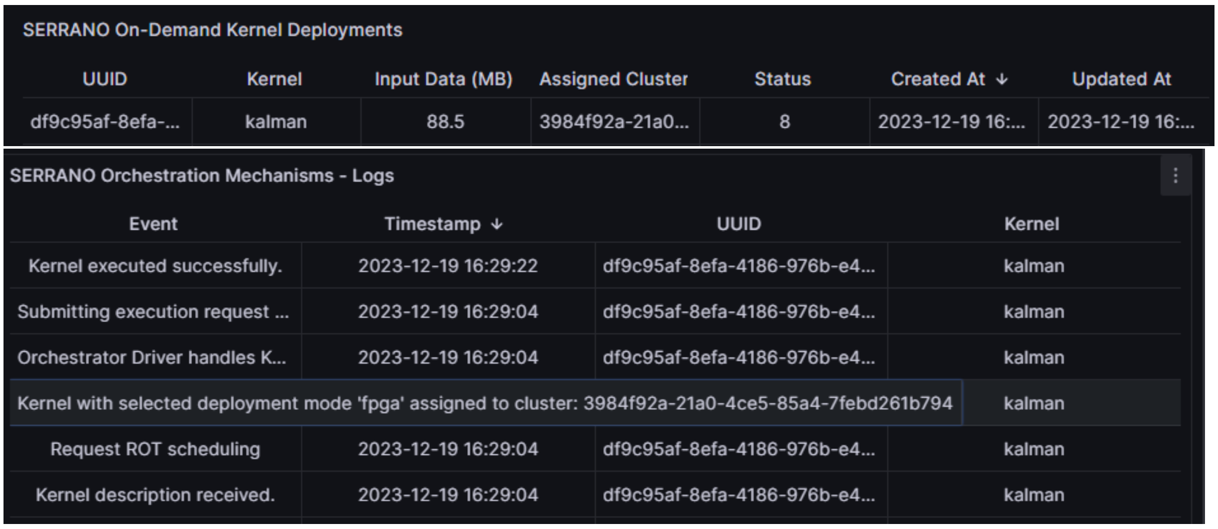
Select Deployment: 225f4f758d04

ID	Kernel Name	Input Dataset	Kernel Mode	Cluster	Status	Started At	Finished At	Completion Time (sec)	Results
b142e110-bd1f-4dd8-abaf-dc9cbd7d4f76	knn	data/knn/machine3_xaxis	gpu	5a075716-7d7d-4b40-9566-bc1a33ee70c2	Completed	1701177131	1701177150	19	Download

Figure 14: Demo-3 web-based application.

We requested the execution of several kernels with different input data and deployment objects using our web-based application. Next, we detail the end-to-end execution for two of them. The first request is about the execution of the SERRANO-accelerated implementation of the KALMAN kernel, providing a large input dataset from the second project use case (Section 5). The input data include 2000 entries with 200 values for each entry and is of total size of 88.5 MB. The execution objective was the minimization of the energy consumption along with the accelerated execution. The second corresponds to the execution of the SERRANO accelerated implementation of the KMEANS kernel, providing a large input dataset from the third project use case (Section 6). The input data has 16384 entries with 520 values per entry and a total size of 81 MB, whereas the execution objective was the minimization of the execution time.

The Resource Orchestrator (*RES.4*) allocates each kernel execution request to available platforms with accelerated resources based on decisions made by the ROT (*RES.1*). The ROT also defines the type of acceleration hardware, GPU or FPGA, for edge and cloud platforms. Then, the SERRANO orchestration mechanisms seamlessly coordinate the required data movement and request to low-level mechanisms, such as vAccel and lightweight virtualisation mechanisms or the HPC Gateway, to execute the kernel (*RES.7*). Figure 15 provides details from the operation of the SERRANO orchestration mechanisms.



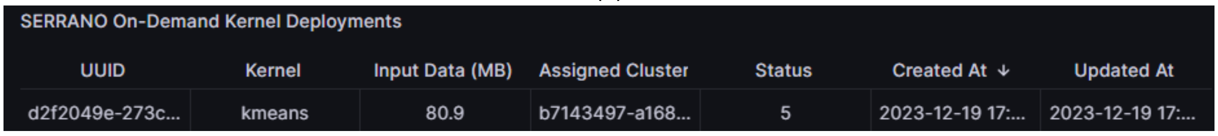
SERRANO On-Demand Kernel Deployments

UUID	Kernel	Input Data (MB)	Assigned Cluster	Status	Created At ↓	Updated At
df9c95af-8efa-...	kalman	88.5	3984f92a-21a0...	8	2023-12-19 16:...	2023-12-19 16:...

SERRANO Orchestration Mechanisms - Logs

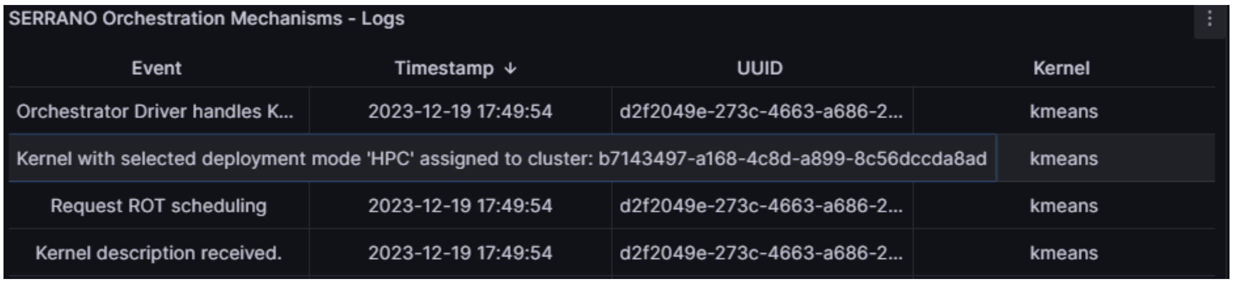
Event	Timestamp ↓	UUID	Kernel
Kernel executed successfully.	2023-12-19 16:29:22	df9c95af-8efa-4186-976b-e4...	kalman
Submitting execution request ...	2023-12-19 16:29:04	df9c95af-8efa-4186-976b-e4...	kalman
Orchestrator Driver handles K...	2023-12-19 16:29:04	df9c95af-8efa-4186-976b-e4...	kalman
Kernel with selected deployment mode 'fpga' assigned to cluster: 3984f92a-21a0-4ce5-85a4-7febd261b794			kalman
Request ROT scheduling	2023-12-19 16:29:04	df9c95af-8efa-4186-976b-e4...	kalman
Kernel description received.	2023-12-19 16:29:04	df9c95af-8efa-4186-976b-e4...	kalman

(a)



SERRANO On-Demand Kernel Deployments

UUID	Kernel	Input Data (MB)	Assigned Cluster	Status	Created At ↓	Updated At
d2f2049e-273c...	kmeans	80.9	b7143497-a168...	5	2023-12-19 17:...	2023-12-19 17:...



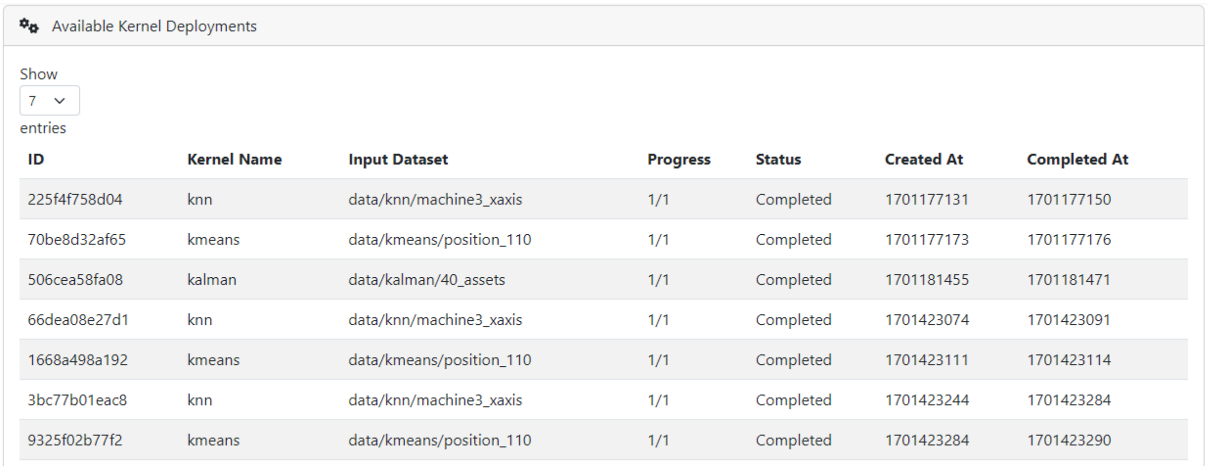
SERRANO Orchestration Mechanisms - Logs

Event	Timestamp ↓	UUID	Kernel
Orchestrator Driver handles K...	2023-12-19 17:49:54	d2f2049e-273c-4663-a686-2...	kmeans
Kernel with selected deployment mode 'HPC' assigned to cluster: b7143497-a168-4c8d-a899-8c56dccda8ad			kmeans
Request ROT scheduling	2023-12-19 17:49:54	d2f2049e-273c-4663-a686-2...	kmeans
Kernel description received.	2023-12-19 17:49:54	d2f2049e-273c-4663-a686-2...	kmeans

Figure 15: Kernel executions and orchestration mechanisms logs: (a) the first execution request, (b) the second request.

We can notice that for the first execution request the orchestration mechanisms selected the FPGA-based accelerated kernel and the execution assigned to the AUTH K8s cluster (UUID *"3984f92a-21a0-4ce5-85a4-7febd261b794"*) that provides that acceleration platform. The second policy assigned to the HPC platform (UUID *"b7143497-a168-4c8d-a899-8c56dccda8ad"*) since the HPC-based accelerated kernel was selected.

The web-based application shows the available kernel execution requests (Figure 16) and presents the real-time progress for each kernel execution request (Figure 17). Moreover, the SERRANO telemetry mechanisms monitor the infrastructure resources and kernel executions (*TEL.1*). They store the collected telemetry data in the PMDS service (*TEL.4*).

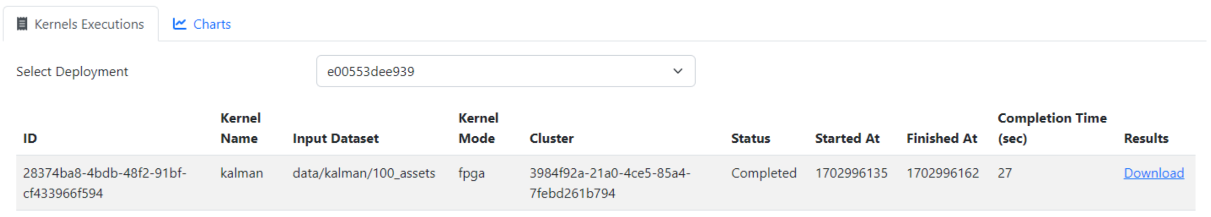


Available Kernel Deployments

Show 7 entries

ID	Kernel Name	Input Dataset	Progress	Status	Created At	Completed At
225f4f758d04	knn	data/knn/machine3_xaxis	1/1	Completed	1701177131	1701177150
70be8d32af65	kmeans	data/kmeans/position_110	1/1	Completed	1701177173	1701177176
506cea58fa08	kalman	data/kalman/40_assets	1/1	Completed	1701181455	1701181471
66dea08e27d1	knn	data/knn/machine3_xaxis	1/1	Completed	1701423074	1701423091
1668a498a192	kmeans	data/kmeans/position_110	1/1	Completed	1701423111	1701423114
3bc77b01eac8	knn	data/knn/machine3_xaxis	1/1	Completed	1701423244	1701423284
9325f02b77f2	kmeans	data/kmeans/position_110	1/1	Completed	1701423284	1701423290

Figure 16: Available kernel execution requests.



Kernels Executions | Charts

Select Deployment: e00553dee939

ID	Kernel Name	Input Dataset	Kernel Mode	Cluster	Status	Started At	Finished At	Completion Time (sec)	Results
28374ba8-4bdb-48f2-91bf-cf433966f594	kalman	data/kalman/100_assets	fpga	3984f92a-21a0-4ce5-85a4-7febd261b794	Completed	1702996135	1702996162	27	Download

Figure 17: Progress of kernel execution requests.

For each successfully executed kernel, the users can download the results. In addition, the application automatically provides a graphical representation of the end-to-end execution time as provided by the SERRANO telemetry mechanisms. Figure 18 shows the relevant information for the first request, i.e. the execution of the KALMAN kernel. The y-axis is the time in milliseconds (ms). The left chart illustrates the required time for each of the three main phases that are involved in the execution of an accelerated kernel in SERRANO. The other chart on the right provides a more detailed breakdown of the total execution time. A video for this demonstration is available on the project's YouTube channel at the following link: https://www.youtube.com/@serranoproject7470

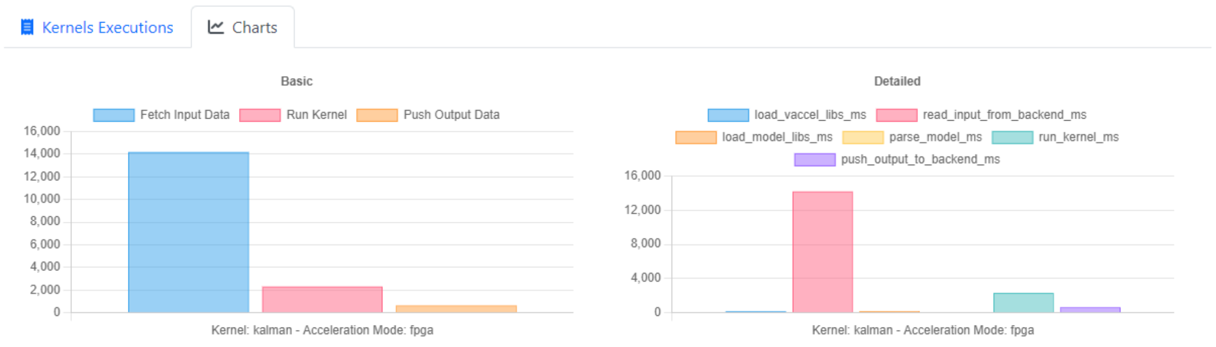


Figure 18: End-to-end kernel execution time as reported by the telemetry services.

3.3 Demo-3: Intent-driven operation and automatic storage policy creation

Description: This demonstration evaluates the capability of SERRANO's orchestration mechanisms to interact with the SERRANO-enhanced Secure Storage Service to create secure storage policies cognitively. We aim to showcase that storage is a core, well-integrated, and easy to use component of the SERRANO platform. The users provide a high-level description of the requested storage policy, the storage policy intent. SERRANO, facilitated by the Resource Optimisation Toolkit's developed algorithms, then translates this intent into resource-specific parameters. Subsequently, the Resource Orchestrator creates the specified secure storage policy through the On-premise Storage Gateway component of the SERRANO Secure Storage service.

To evaluate the performance implications associated with diverse combinations of edge and cloud storage locations, two different storage policies are created. These policies guide the creation of storage buckets that are used for uploading and downloading data. For all these operations, the demonstrator utilises the SERRANO SDK. Section 4 details the storage policies and presents an in-depth performance evaluation of the related SERRANO-provided technological advancements.

SERRANO services and components:

- Secure Storage Service:
  - On-premise storage gateway
  - SERRANO edge storage devices
- Resource orchestrator
- Resource Optimisation Toolkit (ROT)
- Telemetry Service:
  - Central Telemetry Handler (CTH)
  - Enhanced Telemetry Agent (ETA)
  - Edge Storage Probe
  - Persistent Monitoring Data Storage (PMDS)

KPIs measured/evaluated: UC1.1, UC1.7, SIR.1, SIR.2, SIR.3, TEL.1, TEL.4, RES.1, RES.4, INT.6

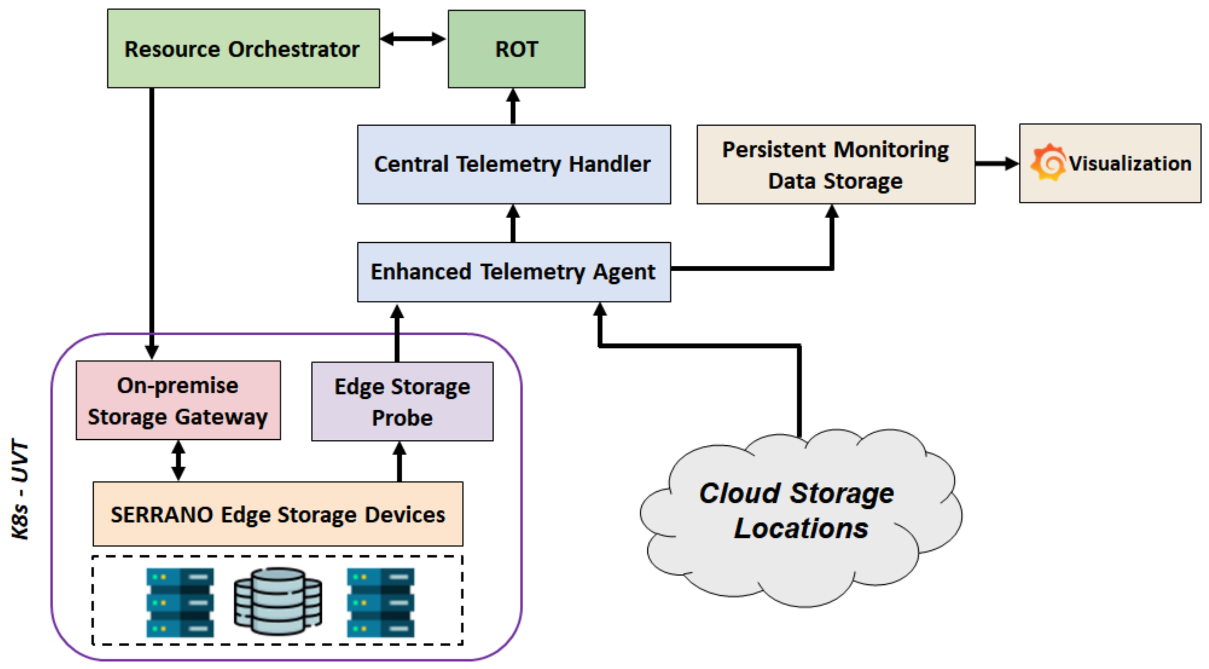
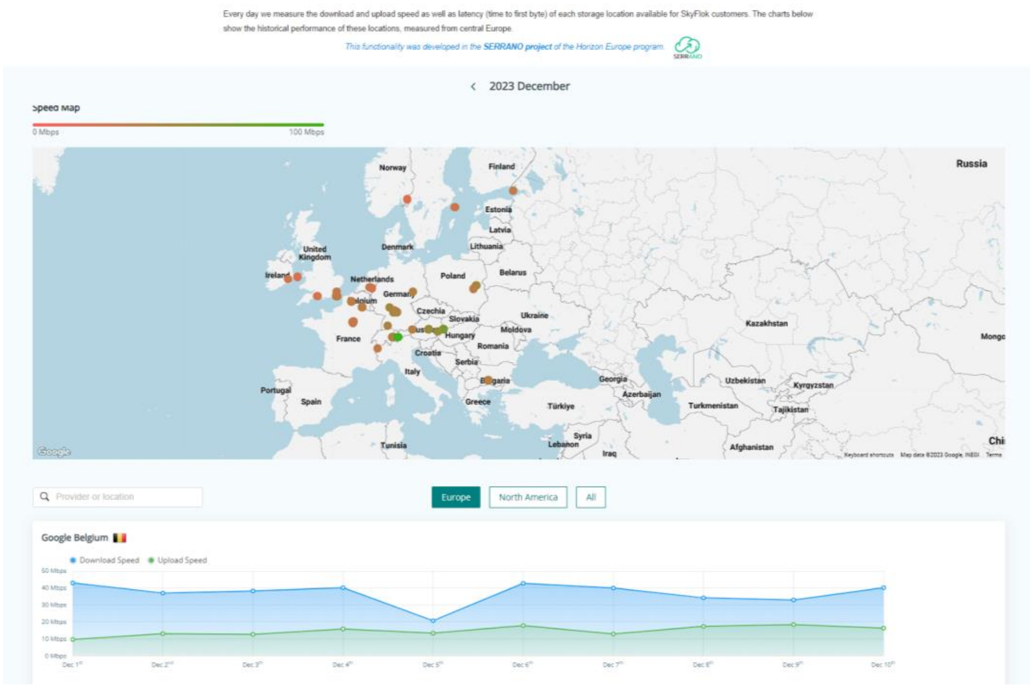

Resource Orchestrator
ROT
Central Telemetry Handler
Persistent Monitoring Data Storage
Visualization
Enhanced Telemetry Agent
On-premise Storage Gateway
Edge Storage Probe
SERRANO Edge Storage Devices
K8s - UVT
Cloud Storage Locations

Figure 19: SERRANO testbed setup for platform evaluation demo-3.

Scenario Description:

The demonstration scenario includes a large number (59) of cloud storage locations worldwide accessible through the Chocolate Cloud (CC) SkyFlok service [2]. There are 24 GDPR-compliant cloud storage locations (*SIR.3*). It also includes four (4) edge storage locations based on the SERRANO edge storage devices, which are deployed in the UVT K8s cluster of the SERRANO testbed. Chocolate Cloud (CC) has also developed a solution that measures the download and upload speed as well as latency (time to first byte) of each storage location available for

SkyFlok customers (Figure 20) once every 24 hours. This information is also made available to ROT through the SERRANO telemetry mechanisms.

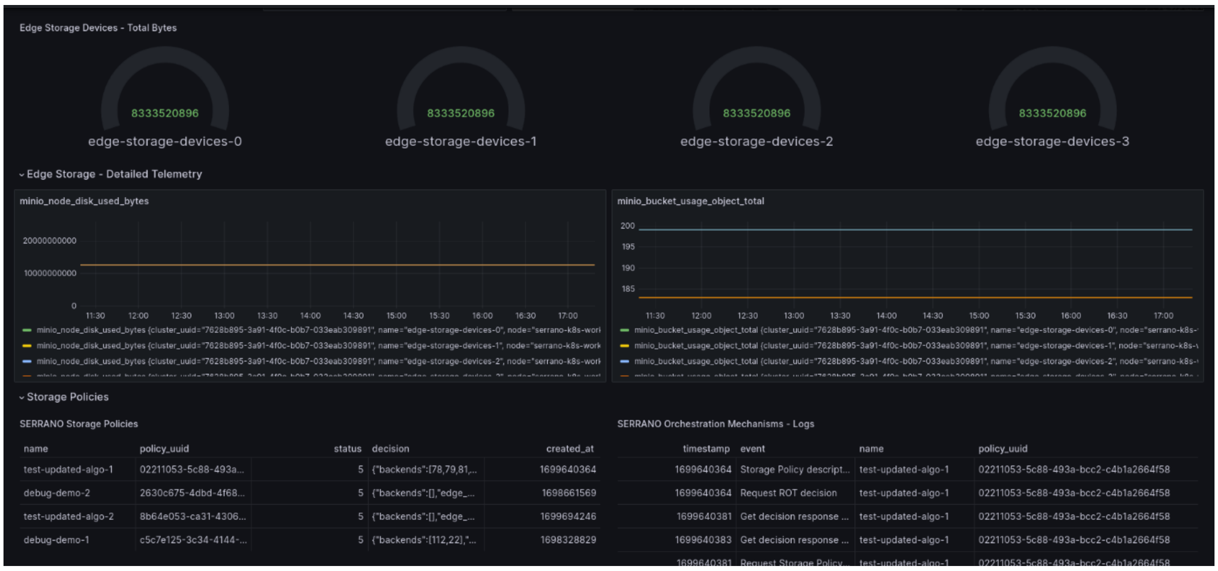

Every day we measure the download and upload speed as well as latency (time to first byte) of each storage location available for SkyFlok customers. The charts below show the historical performance of these locations, measured from central Europe.

This functionality was developed in the ***SERRANO project*** *of the Horizon Europe program.*

Figure 20: Cloud storage locations benchmarking[1].

In addition, a Grafana dashboard (Figure 21) dynamically displays real-time data sourced from the SERRANO telemetry services (*TEL.1, TEL.4*). It also presents relevant events and logs from the SERRANO orchestration mechanisms. These events and logs are related to the translation of the user intent to specific infrastructure parameters. Furthermore, the dashboard captures the automatic requests made by the orchestration mechanisms the On-premise Storage Gateway within the Secure Storage service, marking the initiation of the actual setup for the specified storage policy.

[1] https://www.skyflok.com/backend-performance/



```
minio_node_disk_used_bytes {cluster_uuid="7628b895-3a91-4f0c-b0b7-033eab309891", name="edge-storage-devices-0", node="serrano-k8s-worl
minio_node_disk_used_bytes {cluster_uuid="7628b895-3a91-4f0c-b0b7-033eab309891", name="edge-storage-devices-1", node="serrano-k8s-work
minio_node_disk_used_bytes {cluster_uuid="7628b895-3a91-4f0c-b0b7-033eab309891", name="edge-storage-devices-2", node="serrano-k8s-worl
```

```
minio_bucket_usage_object_total {cluster_uuid="7628b895-3a91-4f0c-b0b7-033eab309891", name="edge-storage-devices-0", node="serrano-k8s-
minio_bucket_usage_object_total {cluster_uuid="7628b895-3a91-4f0c-b0b7-033eab309891", name="edge-storage-devices-1", node="serrano-k8s-
minio_bucket_usage_object_total {cluster_uuid="7628b895-3a91-4f0c-b0b7-033eab309891", name="edge-storage-devices-2", node="serrano-k8s-
```

SERRANO Storage Policies

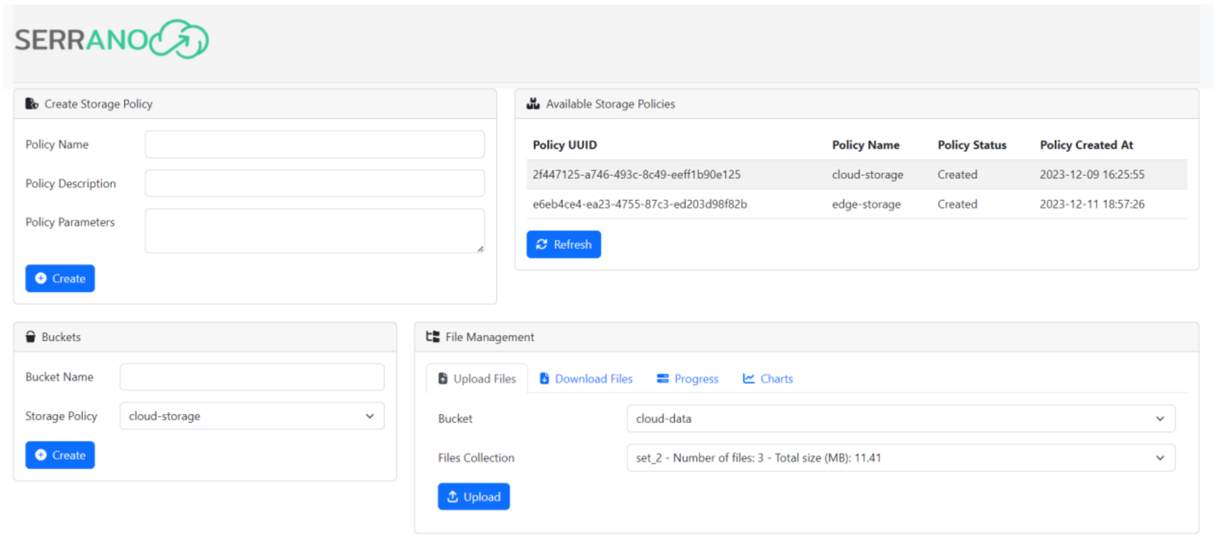
name	policy_uuid	status	decision	created_at
test-updated-algo-1	02211053-5c88-493a...	5	{"backends":[78,79,81,...	1699640364
debug-demo-2	2630c675-4dbd-4f68...	5	{"backends":[],"edge_...	1698661569
test-updated-algo-2	8b64e053-ca31-4306...	5	{"backends":[],"edge_...	1699694246
debug-demo-1	c5c7e125-3c34-4144-...	5	{"backends":[112,22],"...	1698328829

SERRANO Orchestration Mechanisms - Logs

timestamp	event	name	policy_uuid
1699640364	Storage Policy descript...	test-updated-algo-1	02211053-5c88-493a-bcc2-c4b1a2664f58
1699640364	Request ROT decision	test-updated-algo-1	02211053-5c88-493a-bcc2-c4b1a2664f58
1699640381	Get decision response ...	test-updated-algo-1	02211053-5c88-493a-bcc2-c4b1a2664f58
1699640383	Get decision response ...	test-updated-algo-1	02211053-5c88-493a-bcc2-c4b1a2664f58
1699640381	Request Storage Policy...	test-updated-algo-1	02211053-5c88-493a-bcc2-c4b1a2664f58

Figure 21: Grafana dashboard for Demo-3.

Moreover, we developed a web-based application written in Python that facilitates the demonstration (Figure 22). The application is deployed within the UVT K8s cluster. It utilises the SERRANO SDK (*INT.6*) to request the creation of storage policies, create buckets associated with a storage policy, upload data to the buckets, and download from them. The application also provides charts to visualize the results from the various upload and download operations.



Policy Name	
Policy Description	
Policy Parameters	

Policy UUID	Policy Name	Policy Status	Policy Created At
2f447125-a746-493c-8c49-eeff1b90e125	cloud-storage	Created	2023-12-09 16:25:55
e6eb4ce4-ea23-4755-87c3-ed203d98f82b	edge-storage	Created	2023-12-11 18:57:26

Bucket Name	
Storage Policy	cloud-storage

Bucket	cloud-data
Files Collection	set_2 - Number of files: 3 - Total size (MB): 11.41

Figure 22: Demo-3 web-based application.

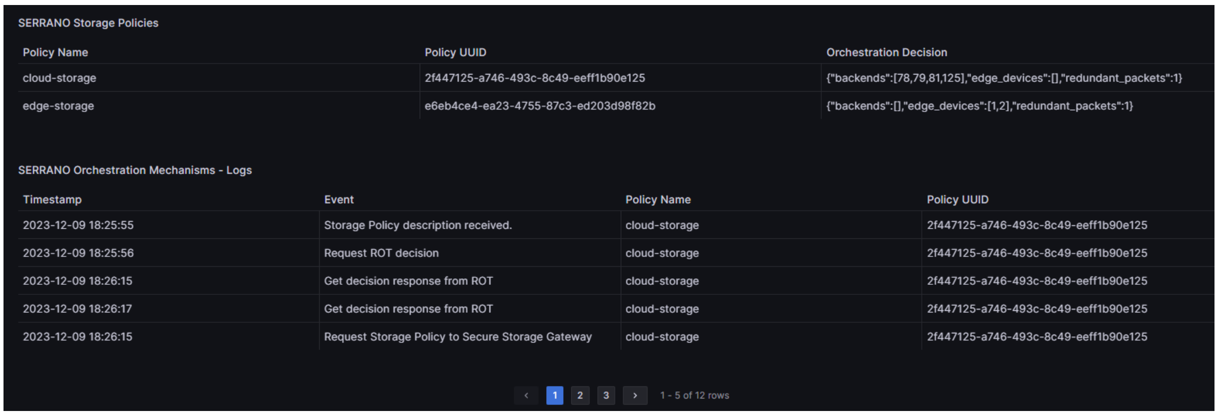
We created two distinct secure storage policies using our web-based application. The first policy is tailored for the exclusive utilisation of cloud storage locations, whereas the second favours the exclusive use of edge storage locations that are based on the SERRANO edge storage devices. Each storage policy is characterised by a set of parameters that describe the user intent of the requested policy. These parameters include: (i) *size*, integer values within the range [0,10], where larger values indicate a preference for storing substantial data, favouring cloud resources over edge resources, (ii) *cost,* integer values within range [0,10], a higher value favours cloud resources, as they are generally considered to have lower costs compared to edge resources, (iii) *volatility,* integer values [0,10], higher values indicate a preference to resources with lower download costs, (iv) *latency,* integer values [0,10], higher values indicate a greater tolerance for latency, and (v) *availability, integer* values [0,4], the value determines the corresponding number of replica resources, with options including [1,2,4,8,12].

Figure 23 shows the two secure storage policy intents and their associated parameters for the SERRANO orchestration mechanisms. This representation provides a clear overview of how each policy aligns with specific storage objectives, guiding SERRANO orchestration mechanisms in making informed decisions based on their unique requirements.

Only cloud storage locations	***Only edge storage locations***
`{ "name": “cloud-storage", "description": "", "policy_parameters": { "data_size": 0, "cost": 5, "volatility": 0, "availability": 2, "latency": 0, "lat": 45.7472357, "lng": 21.2316107 } }`	`{ "name": “edge-storage", "description": "", "policy_parameters": { "data_size":0, "cost":0, "volatility":0, "availability": 1, "latency": 1, "lat": 45.7472357, "lng": 21.2316107 } }`

Figure 23: Storage policies description for Demo-3.

The provision is managed by the Resource Orchestrator (*RES.4*) according to the ROT's decisions (*RES.1*). The SERRANO orchestration mechanisms aim to match the requirements of these storage policies with the most suitable cloud and edge storage locations, erasure coding configuration, and encryption scheme. They also automatically create the new secure storage policies without any other intervention from the users (*UC1.7*). Figure 24 shows details from the SERRANO orchestration mechanisms for the two defined secure storage policies.



SERRANO Storage Policies

Policy Name	Policy UUID	Orchestration Decision
cloud-storage	2f447125-a746-493c-8c49-eeff1b90e125	{"backends":[78,79,81,125],"edge_devices":[],"redundant_packets":1}
edge-storage	e6eb4ce4-ea23-4755-87c3-ed203d98f82b	{"backends":[],"edge_devices":[1,2],"redundant_packets":1}

SERRANO Orchestration Mechanisms - Logs

Timestamp	Event	Policy Name	Policy UUID
2023-12-09 18:25:55	Storage Policy description received.	cloud-storage	2f447125-a746-493c-8c49-eeff1b90e125
2023-12-09 18:25:56	Request ROT decision	cloud-storage	2f447125-a746-493c-8c49-eeff1b90e125
2023-12-09 18:26:15	Get decision response from ROT	cloud-storage	2f447125-a746-493c-8c49-eeff1b90e125
2023-12-09 18:26:17	Get decision response from ROT	cloud-storage	2f447125-a746-493c-8c49-eeff1b90e125
2023-12-09 18:26:15	Request Storage Policy to Secure Storage Gateway	cloud-storage	2f447125-a746-493c-8c49-eeff1b90e125

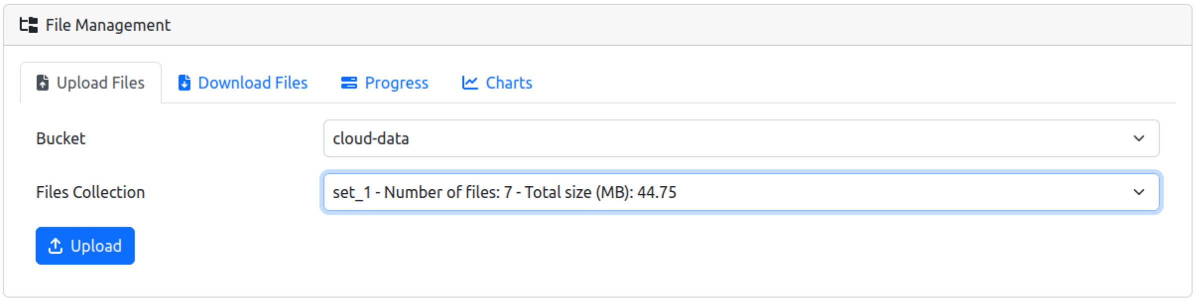
Figure 24: Created secure storage policies and orchestration mechanisms logs.

The Orchestration Decision column in the first table presents the ROT's decision for each requested secure storage policy. We can notice that for the first policy the orchestration mechanisms selected four cloud storage locations (*{**"backends":[78,79,81,125]**, "edge_devices":[],"redundant_packets":1}*), whereas for the second polity two edge storage locations ({"backends":[], ***"edge_devices":[1,2]***, "redundant_packets":1}).

The web-based application displays the policy UUID and name for each successfully created storage policy. This information is then used to create buckets linked to a specific storage policy. For this demonstration, we created the *"cloud-data"* bucket, based on the *"cloud-storage"* policy, and the *"edge-data"* bucket, corresponding to the *"edge-storage"* policy. These actions are executed through the SERRANO SDK, capitalizing on S3-compatible storage interfaces provided by the SERRANO Secure Storage service (*SIR.2*). The storage policy of a bucket is specified through the "*LocationConstraint*" setting, a feature of the S3 API.

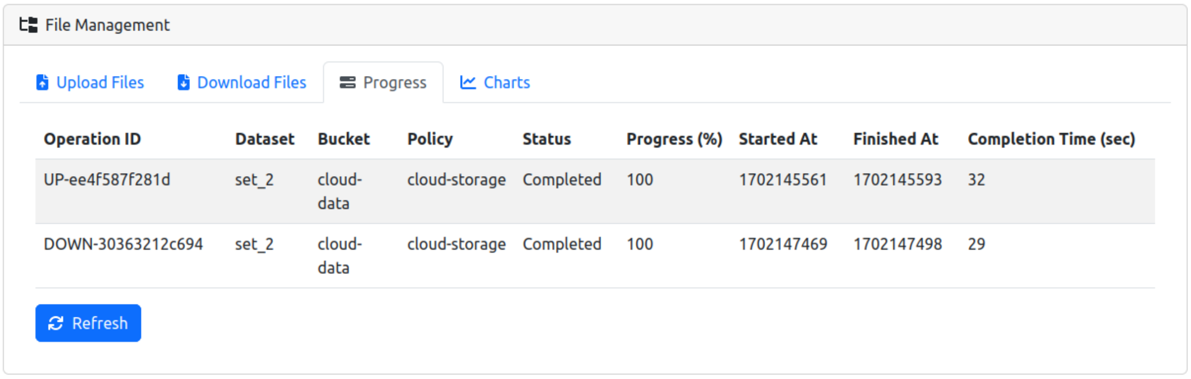
Subsequently, these buckets are used to upload data (Figure 25). To this end, the application offers three predefined datasets summarised in Table 6.

The application presents the real-time progress for both upload and download operations (Figure 26). These operations are seamlessly executed through the SERRANO SDK, employing its S3-compatible methods (*SIR.2)*. The SERRANO telemetry mechanisms monitor the performance of the SERRANO edge storage devices (*TEL.1*) and store the collected telemetry data in the PMDS service (*TEL.4*). Figure 27 shows the increase in the number of stored objects in the selected edge devices during the upload operations.



Bucket	cloud-data
Files Collection	set_1 - Number of files: 7 - Total size (MB): 44.75

Figure 25: Upload data to bucket in SERRANO Secure Storage service.



Operation ID	Dataset	Bucket	Policy	Status	Progress (%)	Started At	Finished At	Completion Time (sec)
UP-ee4f587f281d	set_2	cloud-data	cloud-storage	Completed	100	1702145561	1702145593	32
DOWN-30363212c694	set_2	cloud-data	cloud-storage	Completed	100	1702147469	1702147498	29

Figure 26: Progress of download and upload operations.

Table 6: Demo-3 predefined datasets.

Dataset Name	Number of Files	Total Size (MB)
set_1	7	44.75
set_2	3	11.41
set_3	3	25.68

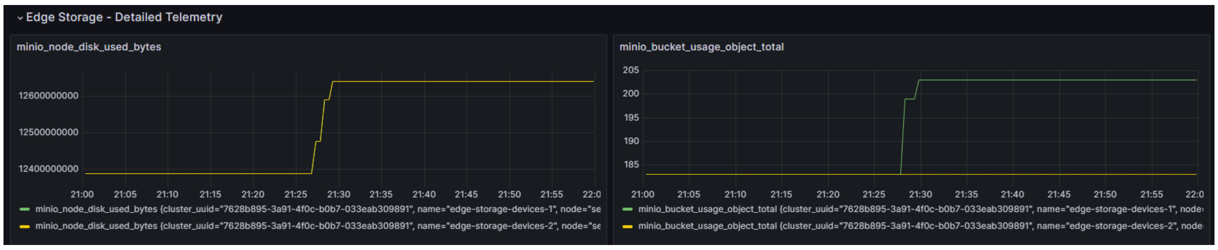
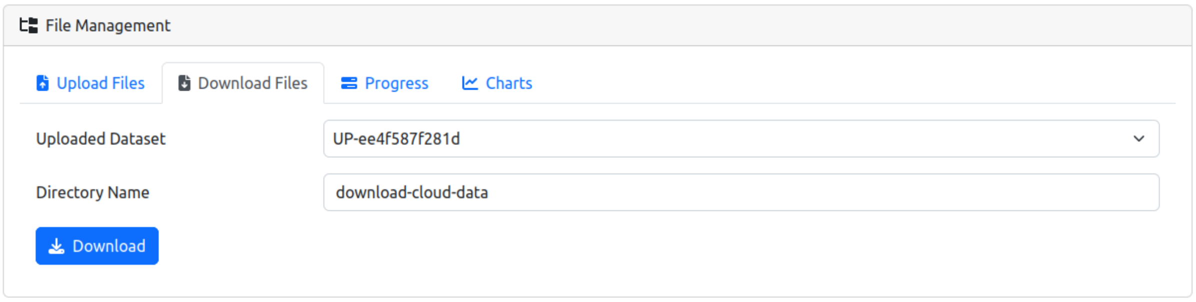


Figure 27: Grafana dashboard - Real-time telemetry data from SERRANO edge storage devices.

In order to evaluate the performance of the two storage policies, we requested the upload of all available datasets in the two buckets. Moreover, we downloaded (Figure 28) all the uploaded data from both buckets. Table 7 provides the total completion time for uploading and downloading the available datasets in the created buckets associated with the two defined secure storage policies. Moreover, Figure 29 provides a graphical representation of the same performance evaluation results as provided automatically by the web-based application.



Uploaded Dataset: UP-ee4f587f281d

Directory Name: download-cloud-data

Figure 28: Download data from bucket in SERRANO Secure Storage service.

Table 7: Total time for uploading and downloading the available datasets using two different storage policies.

Dataset Name	Total Upload (sec)		Total Download (sec)	
	cloud-storage	*edge-storage*	*cloud-storage*	*edge-storage*
set_1	30	10	36	5
set_2	10	5	7	6
set_3	12	5	7	2

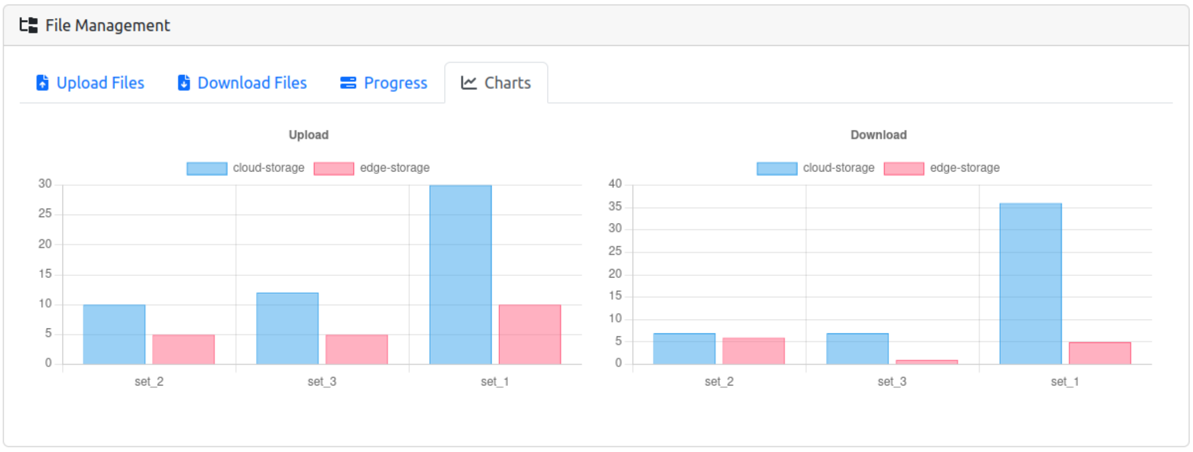


Figure 29: Performance evaluation of the two secure storage policies.

As expected, the storage policy that uses edge storage locations reduces data access latency for upload and download operations for all different datasets. Cloud storage performs noticeably slower than edge storage. Moreover, the demonstration shows the successful integration of edge devices into the SkyFlok and SERRANO ecosystem as a primary means to reduce latency for data storage and retrieval (*UC1.1, SIR.1*). Finally, a video for this demonstration is available in the project's YouTube channel in the following link: https://www.youtube.com/@serranoproject7470

3.4 Demo-4: Service Assurance and Remediation

Description: This demonstration evaluates the capability of Service Assurance and Remediation (SAR) component to ensure that applications are executed within normal boundaries as defined by the users and of the SERRANO orchestration mechanisms to react accordingly. This definition of "normal behaviour" is given by the datasets used for training and validating Machine Learning (ML) predictive models.

Once normal operation is defined and detection begins, any anomalous event is used to trigger proactive or reactive dynamic adjustment of the application deployment via the Resource Orchestrator. In essence, the SAR component can be viewed as a passive component as it requires access to other SERRANO services to make the necessary adjustments. For example, SAR is integrated with the Telemetry System, from which it fetches monitoring data, and with the Resource Orchestrator, which defines the current deployment to be monitored (via query construction and namespaces). The Resource Orchestrator also processes the analysis results sent by SAR via the SERRANO Message Broker through a dedicated Kafka topic.

In order to showcase a typical use case of the SAR, we devised a demonstration scenario that includes the following phases:

1. Querying the SERRANO telemetry system and using the data to train a predictive model on historical data.

2. Instantiating the predictive model and analysing near real-time monitoring data from a running application.
3. Showcasing anomaly detection and reporting on Grafana, including the creation of a custom dashboard.
4. Publishing analysis results via the SERRANO Message Broker to inform Resource Orchestrator remediation actions.
5. Performing remediation actions through the dynamic redeployment of affected microservices by leveraging the SERRANO orchestration and deployment mechanisms.

SERRANO services and components:

- Event Detection Engine
- Resource Orchestrator and Resource Optimisation Toolkit (ROT)
- Message Broker
- Telemetry Service:
    - Central Telemetry Handler (CTH)
    - Enhanced Telemetry Agent (ETA)
    - K8s Probe
    - Persistent Monitoring Data Storage (PMDS)

KPIs measured/evaluated: GEN.1, GEN.2, GEN.4, TEL.1, TEL.2, TEL.4, TEL.5, TEL.6, RES.1, RES.2, RES.3, RES.5, RES.6, RES.7, INT.1, INT.6

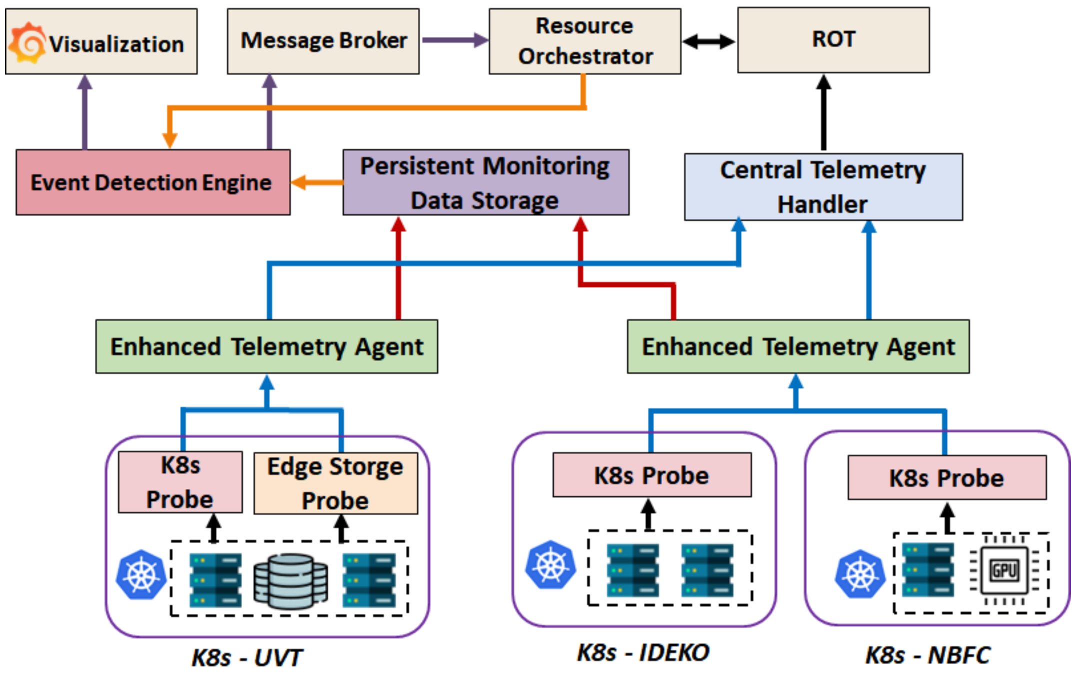


Figure 30: SERRANO testbed setup for platform evaluation Demo-4.

<u>Scenario:</u>

<u>Query and Model Training</u>

Traditionally ML model training is a batch job that requires historical data. This, in most cases, leads to relatively long training times which can be even more time consuming if performance optimisation is also used. Thus, training, optimisation and validation of ML models are done offline. First, the user has to define several parameters in the configuration file used by the Event Detection Engine (EDE) component from SAR:

- **Connector** - In this section, users can define the input and output data sinks. Because EDE is designed to be a distributed system, users have also to define a Dask scheduler. If an existing Dask cluster exists, it must be set here; if not, a local cluster can be defined. Finally, users define the query used for the historical data.
- **Filter/Augmentation** - In these sections, users define any pre-processing operations which are required, including but not limited to: scaling, filtering, deleting, feature engineering, etc.
- **Analysis** - In this section, users can run several data analytics tasks such as; Pearson Correlation, data visualisation etc.
- **Training/Detection** - In these sections, users define ML methods for model creation and validation. Here, users can also define optimisation methods.

Figure 31 shows an example configuration file used to generate training data (last 2 weeks of monitoring) (*TEL.1, TEL.4*) and then train an unsupervised predictive model, IsolationForest (*RES.3*). For this demo we have chosen the IDEKO use-case as the application to be monitored (*INT.1, INT.6*).

It should be noted that pre-processing, ML models, evaluation functions, as well as analysis methods are user definable and can be customised and extended based on user preference. The demonstration presented here is not to be considered as a complete overview of the EDE tool. For more examples and information, please refer to the documentation from the official repository [3].

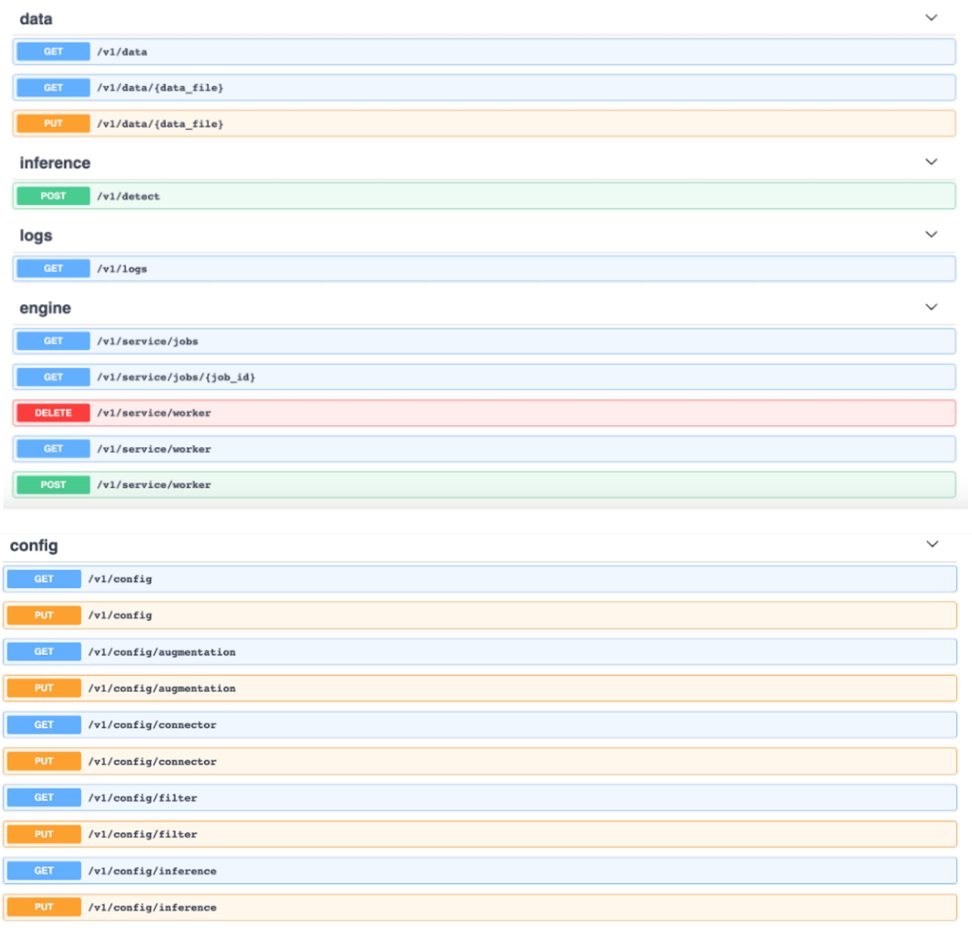
```
Connector:
  PMDS:
    Endpoint: 'http://pmds.services.cloud.ict-serrano.eu'
    Cluster_id: 7628b895-3a91-4f0c-b0b7-033eab309891
    Start: '-2w'
    End: ''
    Groups:
      - general
      - cpu
      - memory
      - network
      - storage
    Namespace: uvt-aspataru
  Dask:
    SchedulerEndpoint: local
    Scale: 3
    SchedulerPort: 8787
    EnforceCheck: false
  KafkaEndpoint: <Serrano Message Broker>
  KafkaPort: 9092
  KafkaTopic: edetopic
  GrafanaUrl: 'http://85.120.206.26:32000'
  GrafanaToken: <token>
  GrafanaTag: ede_test
  MetricsInterval: 1m
  QSize: 0
  Index: time
  QDelay: 10s
Augmentation:
  Scaler:
    StandardScaler:
      copy: true
Mode:
  Training: true
  Validate: false
  Detect: true
Training:
  Type: clustering
  Method: isoforest
  Export: sr_isolationforest_1
  MethodSettings:
    n_estimators: 100
    max_samples: 10
    contamination: 0.07
    verbose: true
    bootstrap: true
Detect:
  Method: isoforest
  Type: clustering
  Load: sr_isolationforest_1
  Scaler: StandardScaler
  Analysis:
    Plot: true
```

Figure 31: Example EDE configuration file.

Inference

For the real-time analytics, the model trained in the previous step is instantiated and exposed to the end user via a RESTful service. The resources exposed to the end-user are, in fact, just mirroring the settings from the YAML configuration file. The main difference is that in the current version, training is not supported only inference is. This was done because training is a long running process; thus, adding it is not a pertinent requirement.

Figure 32 shows the OpenAPI based interface (*GEN.4*) where users can set all the integration endpoints as well as queries and predictive models to be used for inference (*RES.3, TEL.5*).



data
- GET /v1/data
- GET /v1/data/{data_file}
- PUT /v1/data/{data_file}

inference
- POST /v1/detect

logs
- GET /v1/logs

engine
- GET /v1/service/jobs
- GET /v1/service/jobs/{job_id}
- DELETE /v1/service/worker
- GET /v1/service/worker
- POST /v1/service/worker

config
- GET /v1/config
- PUT /v1/config
- GET /v1/config/augmentation
- PUT /v1/config/augmentation
- GET /v1/config/connector
- PUT /v1/config/connector
- GET /v1/config/filter
- PUT /v1/config/filter
- GET /v1/config/inference
- PUT /v1/config/inference

Figure 32: Event Detection Engine REST API Interface.

It should be also noted that this web service uses asynchronous background workers for handling the initialization of detection tasks. Keep in mind that EDE itself is built on a distributed processing backend, Dask, thus these background service workers, after initialising detection, only monitor the resulting Dask tasks. It is not guaranteed that the service worker and detection task should be on the same physical compute node.

The current status of the detection task can be fetched using the job resources as seen in Figure 33 where an example status message can also be seen.

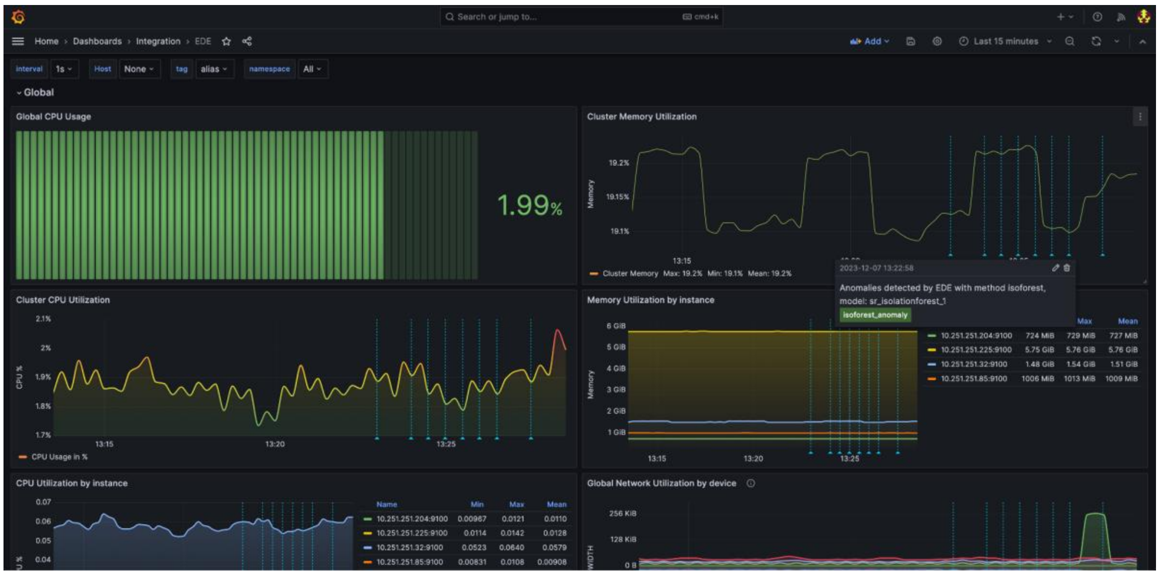
```
{
    "finished": false,
    "meta": {
        "progress": "[2023-12-07 11:21:22] : [INFO] EDE PMDS Executing parallel query with 5 jobs"
    },
    "status": "started"
}
```

Figure 33: Example job status response.

<u>Reporting</u>

Once anomaly detection methods have been executed (*RES.5*), reporting these detections is very important. Furthermore, the root cause analysis results also have to be reported. In order to do this, EDE is capable of pushing metadata about the anomalous event to both the Message Broker as well as a Grafana instance (*RES.6, TEL.4*). If no dashboard is set in Grafana where annotations can be sent, EDE will generate a default dashboard with all available metrics.

Figure 34 shows this Generic EDE Grafana Dashboard including the annotations with the anomalies identified. We should mention here that not all visualisation types support annotations in Grafana. For example, in Figure 34 the Global CPU Usage has no annotations.


Search or jump to...
Home › Dashboards › Integration › EDE
Add
Last 15 minutes
interval 1s Host None tag alias namespace All
Global
Global CPU Usage
1.99%
Cluster Memory Utilization
Cluster Memory Max: 19.2% Min: 19.1% Mean: 19.2%
Cluster CPU Utilization
CPU Usage in %
Memory Utilization by instance
CPU Utilization by instance
Global Network Utilization by device

2023-12-07 13:22:58

Anomalies detected by EDE with method isoforest, model: sr_isolationforest_1

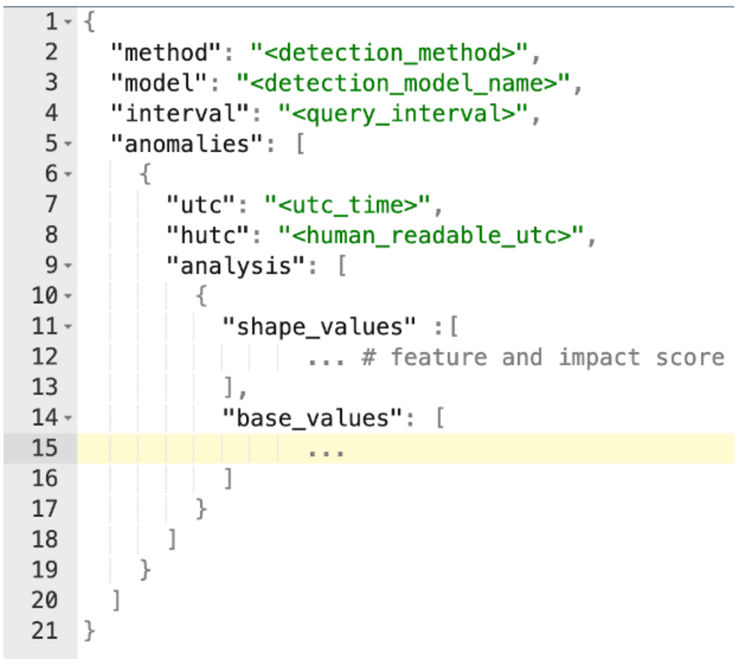
isoforest_anomaly

		Max	Mean
10.251.251.204:9100	724 MiB	729 MiB	727 MiB
10.251.251.225:9100	5.75 GiB	5.76 GiB	5.76 GiB
10.251.251.32:9100	1.48 GiB	1.54 GiB	1.51 GiB
10.251.251.85:9100	1006 MiB	1013 MiB	1009 MiB

Name	Min	Max	Mean
10.251.251.204:9100	0.00967	0.0121	0.0110
10.251.251.225:9100	0.0114	0.0142	0.0128
10.251.251.32:9100	0.0523	0.0640	0.0579
10.251.251.85:9100	0.00831	0.0108	0.00908

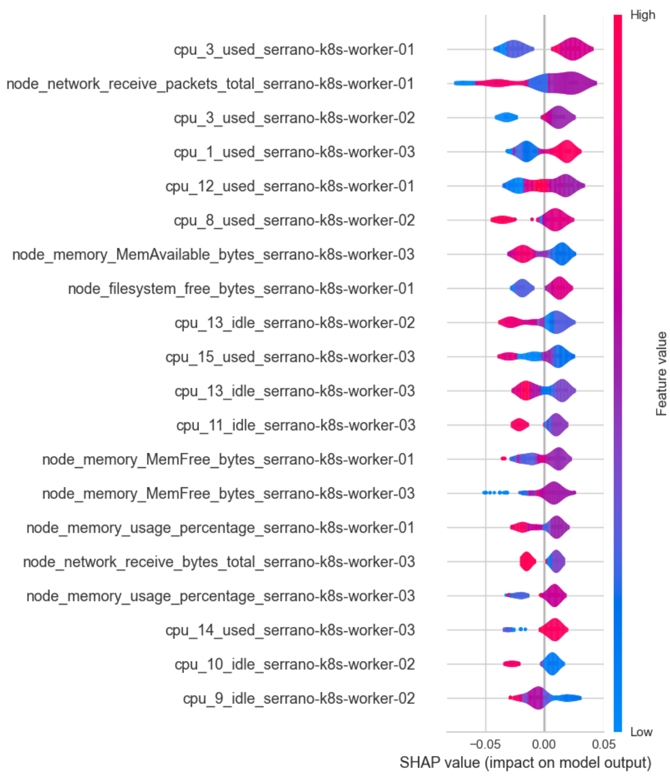
Figure 34: EDE Grafana Dashboard and annotations.

The root cause analysis takes the form of feature rankings computed using Shapely values (*RES.6*). This will push for each detection instance a JSON descriptor into the Message Broker of the form:



```
{
  "method": "<detection_method>",
  "model": "<detection_model_name>",
  "interval": "<query_interval>",
  "anomalies": [
    {
      "utc": "<utc_time>",
      "hutc": "<human_readable_utc>",
      "analysis": [
        {
          "shape_values" :[
              ... # feature and impact score
          ],
          "base_values": [
              ...
          ]
        }
      ]
    }
  ]
}
```

In the above Figure 36, "shape_values" consists of key-value pairs with the key being the feature name and the value being the score obtained. Another visualisation of this analysis can be seen in Figure 35. In this figure, we can also see that each feature encodes not only the feature name but also the originator of each measurement. This will allow exact pinpointing of the issue being detected. This information is then retrieved by the Resource Orchestrator that executes automatically all the necessary actions.



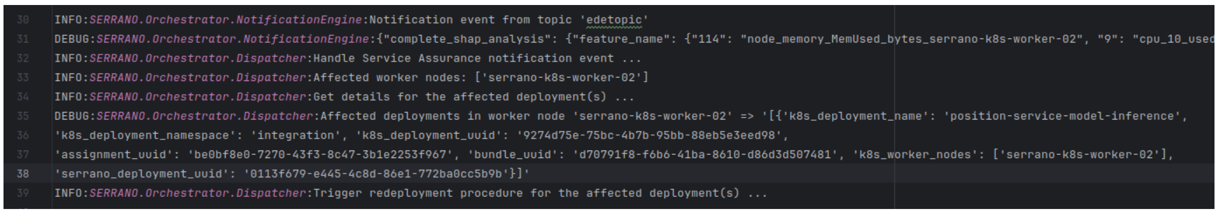
.

Figure 35: Root cause analysis - Feature Ranking

Automatic application redeployment

The SERRANO Resource Orchestrator is subscribed to the Kafka topic that the Event Detection Engine (EDE) forwards the information related to the detected events within the SERRANO platform. Upon receiving the event description, it coordinates the necessary redeployment actions to mitigate the performance degradation for the affected deployed applications (*RES.2, RES.3*). To this end, the Resource Orchestrator analyses the details in the ***"analysis"*** and ***"shape_value"*** fields, determines the affected worker nodes and applications' microservices that are assigned to them (*TEL.4, INT.1*). These operations involve the interaction among various components such as the Event Detection Engine, Resource Orchestrator, Message Broker, and Central Telemetry Handler (GEN.4, *TEL.1, INT.6*).

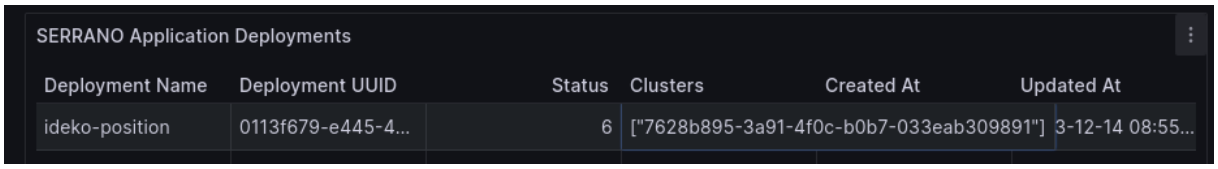
Figure 36 shows logs from the operation of the Resource Orchestrator components after receiving the anomalous event notification.



```
INFO:SERRANO.Orchestrator.NotificationEngine:Notification event from topic 'edetopic'
DEBUG:SERRANO.Orchestrator.NotificationEngine:{"complete_shap_analysis": {"feature_name": {"114": "node_memory_MemUsed_bytes_serrano-k8s-worker-02", "9": "cpu_10_used
INFO:SERRANO.Orchestrator.Dispatcher:Handle Service Assurance notification event ...
INFO:SERRANO.Orchestrator.Dispatcher:Affected worker nodes: ['serrano-k8s-worker-02']
INFO:SERRANO.Orchestrator.Dispatcher:Get details for the affected deployment(s) ...
DEBUG:SERRANO.Orchestrator.Dispatcher:Affected deployments in worker node 'serrano-k8s-worker-02' => '[{'k8s_deployment_name': 'position-service-model-inference',
'k8s_deployment_namespace': 'integration', 'k8s_deployment_uuid': '9274d75e-75bc-4b7b-95bb-88eb5e3eed98',
'assignment_uuid': 'be0bf8e0-7270-43f3-8c47-3b1e2253f967', 'bundle_uuid': 'd70791f8-f6b6-41ba-8610-d86d3d507481', 'k8s_worker_nodes': ['serrano-k8s-worker-02'],
'serrano_deployment_uuid': '0113f679-e445-4c8d-86e1-772ba0cc5b9b'}]'
INFO:SERRANO.Orchestrator.Dispatcher:Trigger redeployment procedure for the affected deployment(s) ...
```

Figure 36: Orchestration mechanisms logs and analysis of detected anomalous event detection.

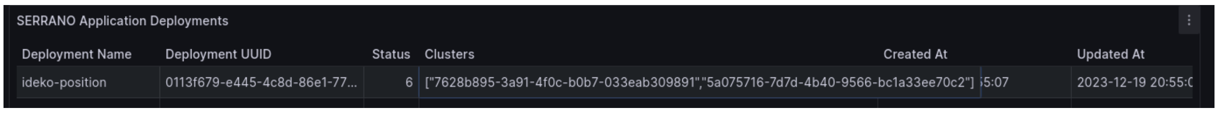
At the same time, the SERRANO telemetry framework is notified for anomalous performance in the affected cluster and automatically collects measurements based on the streaming telemetry (*TEL2, TEL.6*), where continuous measurements are sent at a rate much shorter than the typical monitoring approach. Next, the Resource Orchestrator requests the ROT (*RES.1*) to assign the affected microservices across the available edge and cloud resources (*GEN.1*) while excluding the affected worker nodes from the orchestration decision. Figure 37 shows details from the SERRANO orchestration mechanisms for the redeployment of the affected microservices.



SERRANO Application Deployments

Deployment Name	Deployment UUID	Status	Clusters	Created At	Updated At
ideko-position	0113f679-e445-4...	6	["7628b895-3a91-4f0c-b0b7-033eab309891"]		3-12-14 08:55...

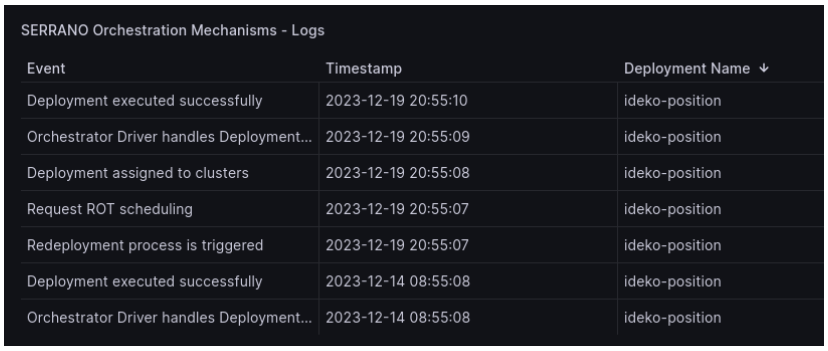
(a)



SERRANO Application Deployments

Deployment Name	Deployment UUID	Status	Clusters	Created At	Updated At
ideko-position	0113f679-e445-4c8d-86e1-77...	6	["7628b895-3a91-4f0c-b0b7-033eab309891","5a075716-7d7d-4b40-9566-bc1a33ee70c2"]	5:07	2023-12-19 20:55:

(b)



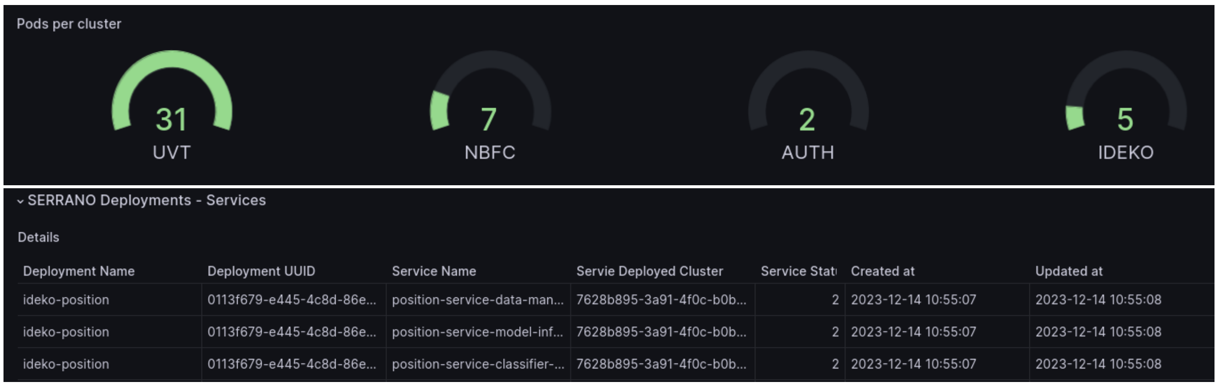
SERRANO Orchestration Mechanisms - Logs

Event	Timestamp	Deployment Name
Deployment executed successfully	2023-12-19 20:55:10	ideko-position
Orchestrator Driver handles Deployment...	2023-12-19 20:55:09	ideko-position
Deployment assigned to clusters	2023-12-19 20:55:08	ideko-position
Request ROT scheduling	2023-12-19 20:55:07	ideko-position
Redeployment process is triggered	2023-12-19 20:55:07	ideko-position
Deployment executed successfully	2023-12-14 08:55:08	ideko-position
Orchestrator Driver handles Deployment...	2023-12-14 08:55:08	ideko-position

Figure 37: SERRANO Deployment objects and orchestration mechanisms logs.

The Clusters column in the *"SERRANO Application Deployments"* table (Figure 37) presents the ROT's decision for the assignment of the application microservices to the available K8s clusters (*GEN.2*, *RES.2*). We can notice that, initially (Figure 37a), all microservices were deployed in the same cluster (UVT K8s cluster – UUID *"7628b895-3a91-4f0c-b0b7-033eab309891"*). In contrast, after the redeployment (Figure 37b), the affected microservice (i.e. *"position-service-model-inference"*) was moved to a second cluster (NBFC K8s cluster - UUID *"5a075716-7d7d-4b40-9566-bc1a33ee70c2"*), and the SERRANO Deployment object has two assignment entries.

Finally, the Resource Orchestrator interacts with the Orchestration Drivers (*RES.2, RES.7*) to perform the transparent deployment across the SERRANO platform without any other user intervention. Figure 38 shows details from the custom Grafana dashboard regarding the number of applications microservices each in K8s cluster before and after the redeployment as well as details about the position service microservices.

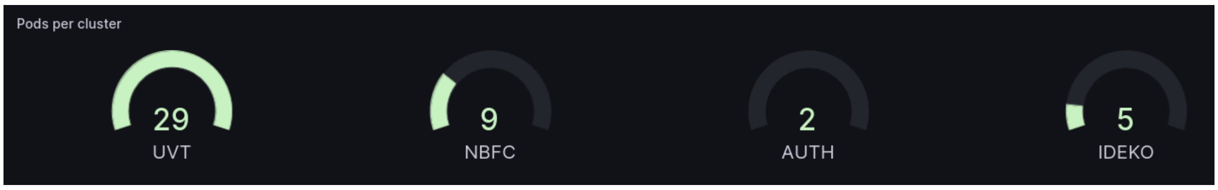


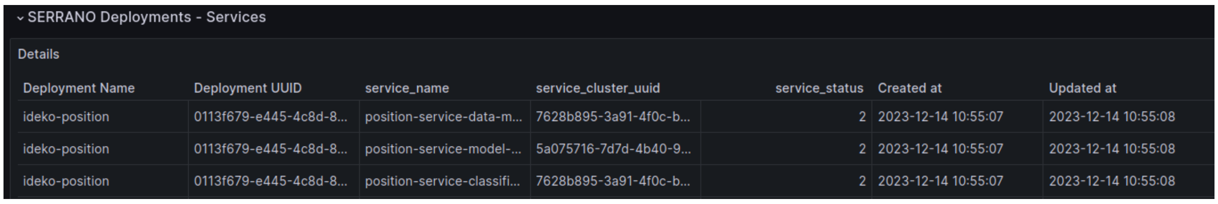
SERRANO Deployments - Services

Details

Deployment Name	Deployment UUID	Service Name	Servie Deployed Cluster	Service Stat	Created at	Updated at
ideko-position	0113f679-e445-4c8d-86e...	position-service-data-man...	7628b895-3a91-4f0c-b0b...	2	2023-12-14 10:55:07	2023-12-14 10:55:08
ideko-position	0113f679-e445-4c8d-86e...	position-service-model-inf...	7628b895-3a91-4f0c-b0b...	2	2023-12-14 10:55:07	2023-12-14 10:55:08
ideko-position	0113f679-e445-4c8d-86e...	position-service-classifier-...	7628b895-3a91-4f0c-b0b...	2	2023-12-14 10:55:07	2023-12-14 10:55:08

(a)





⌄ SERRANO Deployments - Services

Details

Deployment Name	Deployment UUID	service_name	service_cluster_uuid	service_status	Created at	Updated at
ideko-position	0113f679-e445-4c8d-8...	position-service-data-m...	7628b895-3a91-4f0c-b...	2	2023-12-14 10:55:07	2023-12-14 10:55:08
ideko-position	0113f679-e445-4c8d-8...	position-service-model-...	5a075716-7d7d-4b40-9...	2	2023-12-14 10:55:07	2023-12-14 10:55:08
ideko-position	0113f679-e445-4c8d-8...	position-service-classifi...	7628b895-3a91-4f0c-b...	2	2023-12-14 10:55:07	2023-12-14 10:55:08

Figure 38: Grafana dashboard – Position service microservices before and after the automatic redeployment.

Moreover, a video for this demonstration is available in the project's YouTube channel in the following link https://www.youtube.com/@serranoproject7470.

4 Secure Storage Use Case

The Secure Storage Use Case highlights the SERRANO integrated platform's storage capabilities. Beyond the core storage functionality, it uses the acceleration capabilities developed as part of the project to improve performance and the intelligent orchestration features to meet user requirements closely.

4.1 Use case description

This UC focuses on providing secure, high-performance storage of files with lower latency than a purely cloud-based approach. It achieves this goal by extending Chocolate Cloud's commercial SkyFlok multi-cloud distributed storage service with on-premises edge devices that act as storage locations. Medium and large businesses (250+ employees) that are SkyFlok customers would like to extend their use of Chocolate Cloud's SkyFlok service.

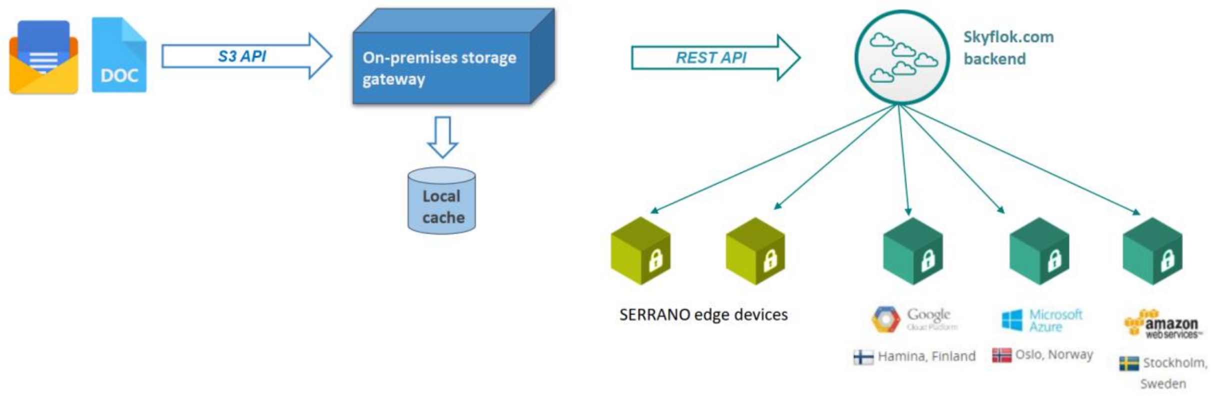


Figure 39: Secure Storage UC components.

SkyFlok works great for file-based collaboration and file sharing workflows as well as for data archival purposes. However, given its fully cloud-based architecture, it lags behind in terms of latency compared to an on-premises storage solution. Moving data closer to the edge can significantly improve download and upload latencies. While many customers choose SkyFlok over competing solutions thanks to the privacy guarantees it offers, privacy concerns remain a major impediment to the more wide-spread adoption of cloud storage in general. Due to legal requirements or internal policies, enterprises want strong guarantees that their data cannot be accessed by third parties, including the storage provider. Conventional cloud storage can only achieve this to a limited degree. Moving the file encryption/decryption process on-premises, under full control of the enterprise, is key to providing these guarantees. To make it even more appealing to enterprise customers, files are accessed through an S3-compatible API. This makes it very easy to interact with the storage service, removing one of the obstacles usually faced by companies when they want to migrate from an existing storage solution to a new one.

The technical details of the use case are presented in Deliverable 2.4 and Deliverable 6.7. The Secure Storage Service is described in Deliverable 3.4.

4.2 Demo

4.2.1 Mid-project Secure Storage Demonstrator

An early version of the use case was presented in the mid-project review in M20 through the "Secure Storage Demonstrator". It has been included in this deliverable for completeness but will only be presented in the final project review again if specifically requested. A detailed description of this demonstrator can be found in Deliverable D6.6 [4].

The M20 Demo focused on showing the S3-compatible API for managing buckets and objects and using it for file storage operations (**SIR.2**). To highlight the widespread applicability of this solution, the demonstrator featured s3fs [5], a well-known S3 client accessing the API. S3fs enables Linux, MacOS, and FreeBSD users to seamlessly mount an S3-backed FUSE [6] file system on a device's system. This makes it possible to very easily create a cloud-based storage and sharing solution that appears to the user as a conventional folder. This is shown in Figure 40. Using fstab [7], it can be mounted automatically during startup and behave similarly to the desktop clients available for services such as Dropbox or Google Drive.

The demo included the following steps:

- Map showing cloud storage locations.
- Using docker to run the Gateway and 3 SERRANO edge locations.
- Creating a storage policy through the SkyFlok web admin interface, detailed explanation of the 3+3 hybrid policy, where 3 locations are edge devices and 3 are cloud locations in the EU.
- Create S3 bucket using AWS CLI using the newly created storage policy.
- Create a folder on the local machine.
- Mount S3 bucket into the newly created folder using s3fs.
- Copy a .pptx file into folder.
- Check using SkyFlok web admin interface that the file has been uploaded and uses the locations we have previously selected.
- Open the .pptx file using the file system explorer.
- Unmount folder using the file system explorer.

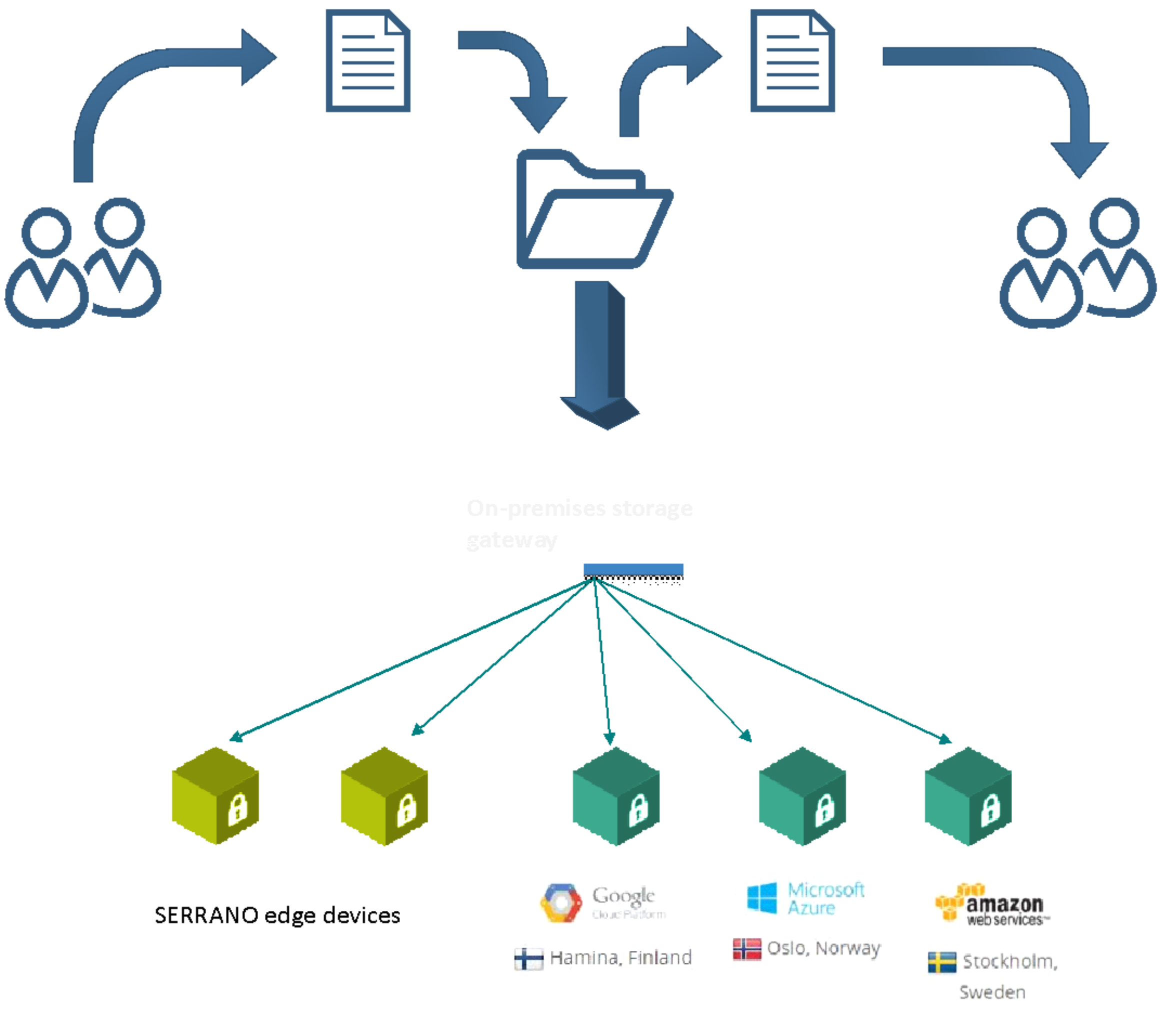


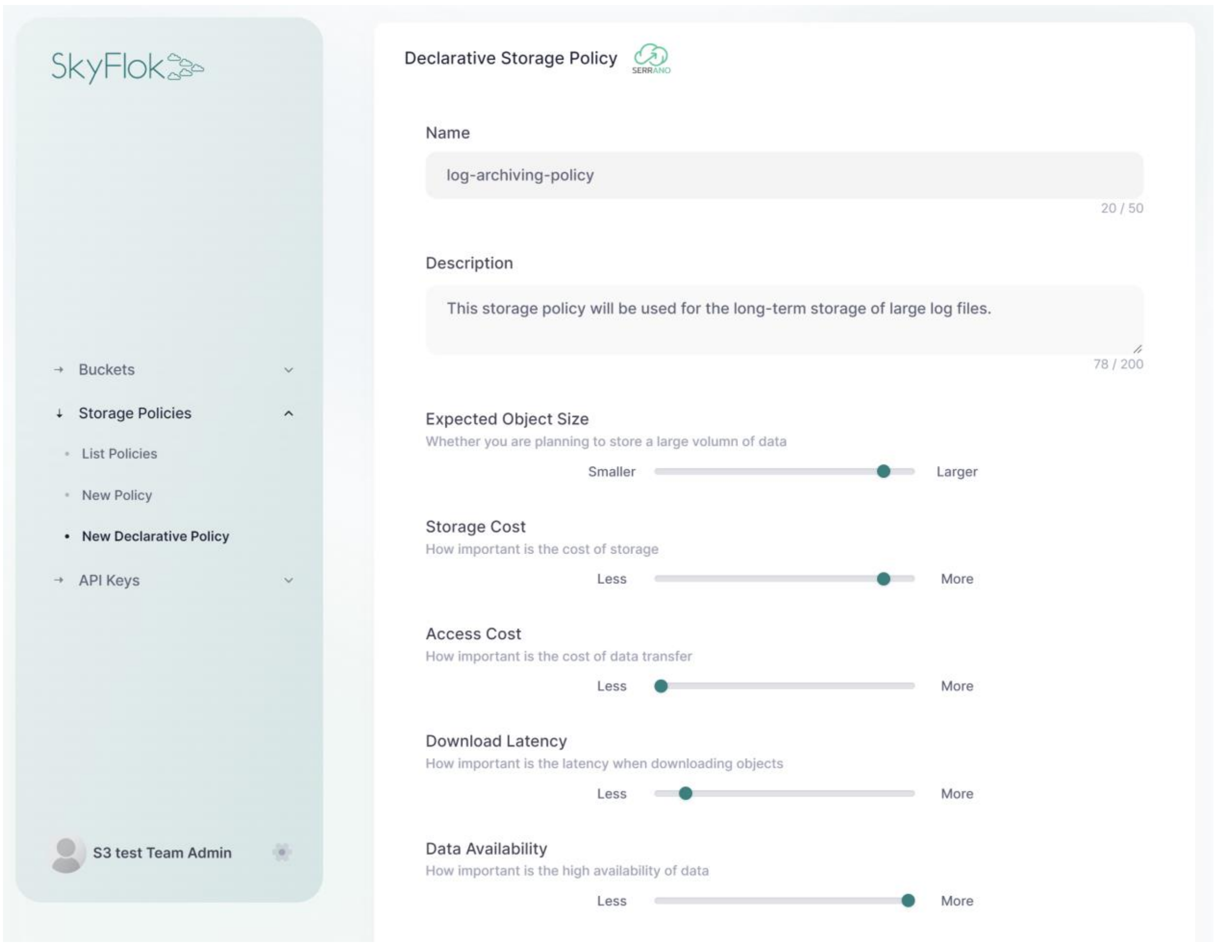
Figure 40: M20 Secure storage demonstrator: overview of sharing files using s3fs and FUSE and the SERRANO-enhanced Storage Service.

4.2.2 Developer web portal and cloud performance

This demonstrator showcases the user-friendly interfaces of the SERRANO-enhanced Secure Storage Service. While there are no KPIs directly connected to this aspect apart from **GEN.3**, the usability of the project's outcomes is an important requirement. It is also a key factor in Chocolate Cloud's exploitation plans.

To enhance the usability of the SERRANO-enhanced Storage Service, we have created a web portal to cater to the needs of the developers who will use the service. The features have been selected by studying the online interfaces of object storage providers and based on the project's requirements with special emphasis on the use cases. The developer portal will also play an important role in the exploitation of the project's outcomes.

Its features revolve around letting developers manage three core entity types: S3 buckets, storage policies, and API keys. Compared to the REST APIs, it presents a friendly, graphical environment suitable for non-technical users as well. Figure 41 shows the interface where users can create a new storage policy, by declaring the application's intent. This information is then submitted to the SERRANO orchestration services.



Declarative Storage Policy

Name: log-archiving-policy (20 / 50)

Description: This storage policy will be used for the long-term storage of large log files. (78 / 200)

Expected Object Size
Whether you are planning to store a large volumn of data
Smaller — Larger

Storage Cost
How important is the cost of storage
Less — More

Access Cost
How important is the cost of data transfer
Less — More

Download Latency
How important is the latency when downloading objects
Less — More

Data Availability
How important is the high availability of data
Less — More

Figure 41: Developer portal – second step of the new storage policy creation wizard.

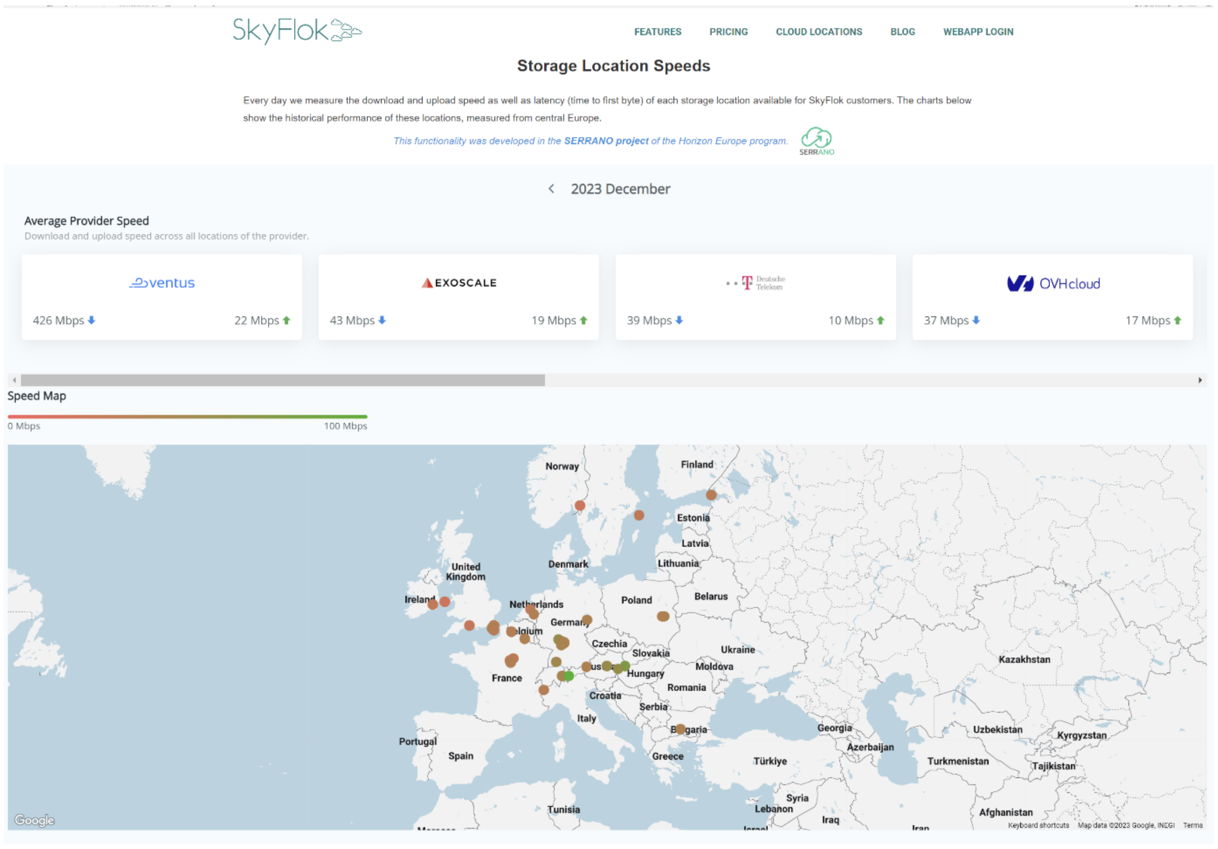
As part of our efforts to disseminate the results of SERRANO, we have created a website [8] that publishes cloud monitoring information for all supported cloud storage locations. The data is collected using the schema described in Deliverable D3.4 [9] and includes locations across the globe from the three major global and most EU-based cloud providers. Data is aggregated from daily measurements and included starting with June 2023. Measurements are currently being performed from central Europe with plans to extend this to locations in Western Europe and the US. By disseminating this information to the general public, we aim to raise awareness of both SERRANO and SkyFlok. We also plan to use it as part of the marketing campaign for SkyFlok S3, a software product that we are developing based on the technological advances made by SERRANO.

The demonstrator will first present the entities managed through the Developer web portal:

- Buckets: listing of all buckets of a team and metadata associated with individual buckets.
- Storage Policies: list existing storage policies, new policy creation wizard, declarative storage policy creation.
- API keys: list existing API keys, create a new API key.

The demo will then show the information provided by the website:

- First, the possibility to choose the month for which performance data is presented, as shown in Figure 42. This refreshes the map view, granting a quick overview of how the measured performance evolved through a simple red-green colour scale.



Storage Location Speeds

Every day we measure the download and upload speed as well as latency (time to first byte) of each storage location available for SkyFlok customers. The charts below show the historical performance of these locations, measured from central Europe.

This functionality was developed in the ***SERRANO project*** *of the Horizon Europe program.*

2023 December

Average Provider Speed

Download and upload speed across all locations of the provider.

ventus	426 Mbps ⬇	22 Mbps ⬆
EXOSCALE	43 Mbps ⬇	19 Mbps ⬆
Deutsche Telekom	39 Mbps ⬇	10 Mbps ⬆
OVHcloud	37 Mbps ⬇	17 Mbps ⬆

Speed Map

0 Mbps 100 Mbps

Figure 42: Cloud performance website showing map of EU cloud storage locations and their relative performances.

- Second, we scroll down to see a way to filter the locations based on their geographic locations and we use the search function to find a cloud provider we are interested in. We look at the presented data as shown in Figure 43, noticing how download and upload speeds compare to each other. We can also see that different locations have different variances in their measured performance, especially in the case of approximated latency.

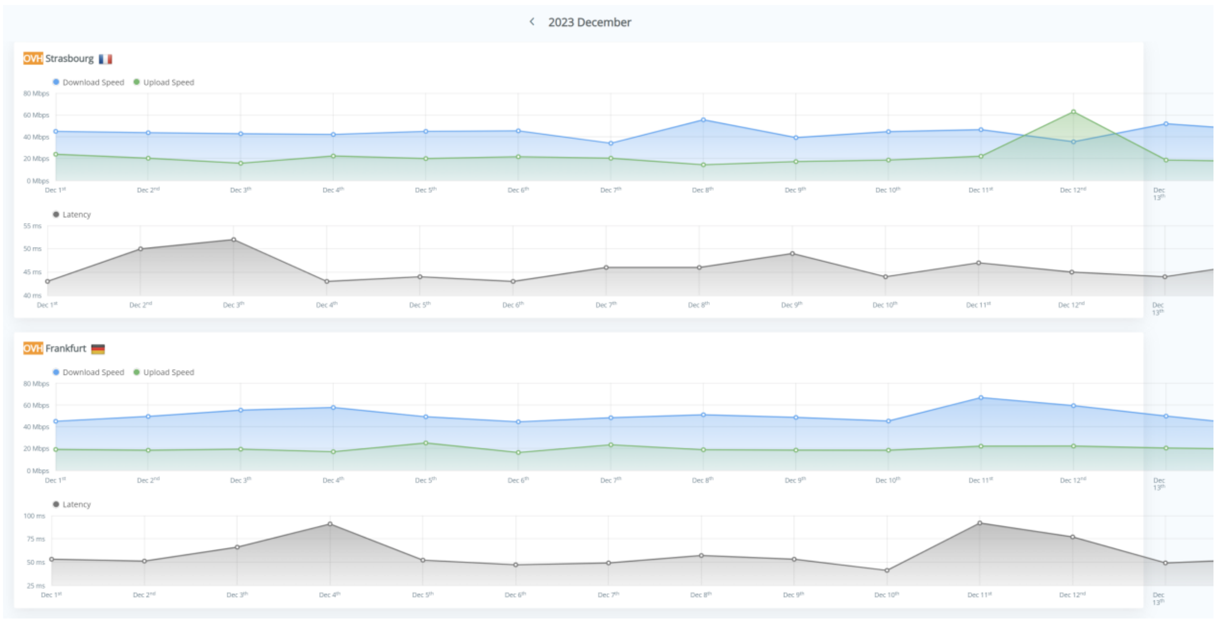


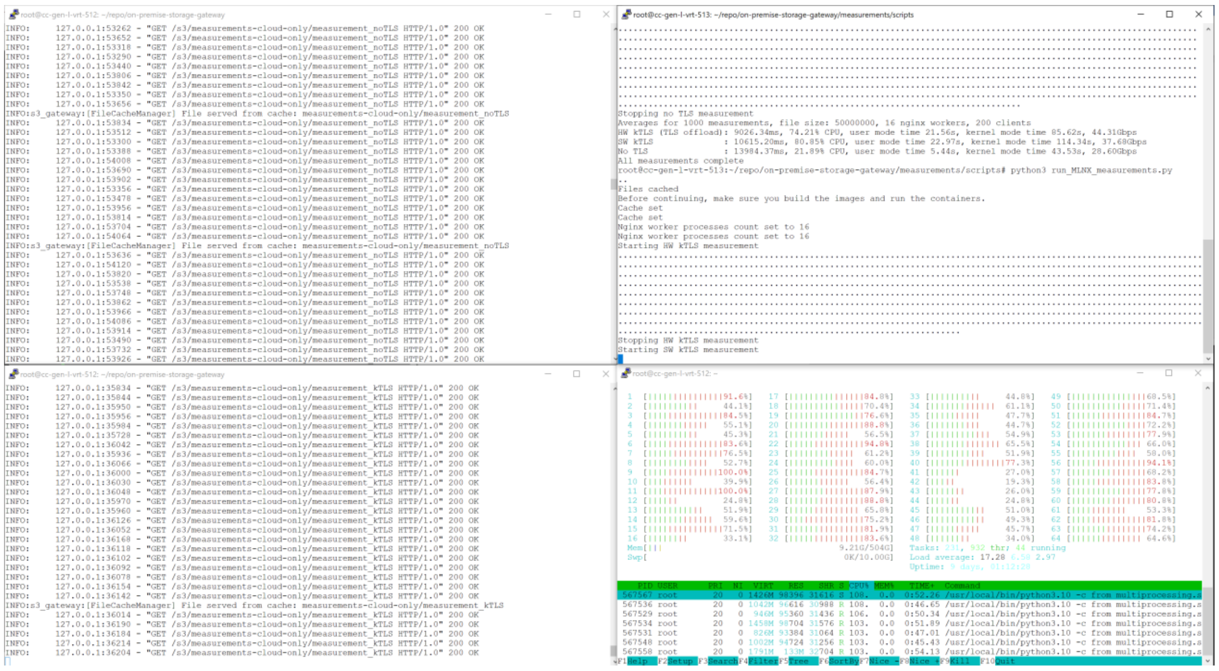
Figure 43: Cloud performance website showing detailed information of OVH Strasbourg and Frankfurt locations for the month of December 2023.

4.2.3 TLS offloading

Whenever an Nvidia DPU is available, the On-premises Storage Gateway can perform the TLS encryption directly on this resource for outgoing connections. TLS-offloading decreases the overall CPU cycles needed to serve each request, which provides performance benefits in scenarios with a CPU bottleneck. The overall time to retrieve an object is reduced and the overall outgoing throughput of the system increases. This demonstrator has been created to provide a deep dive into how TLS offloading has been integrated with the SERRANO-enhanced Storage Service and illustrate how the associated KPIs (**UC1.6**, **SIR.4**, **SIR.5**) have been evaluated.

The demonstrator is composed of the following steps:

- Present the components involved in the demonstrator: the two instances of the Gateway running inside Docker containers and the S3 client application.
- While measurements are running, a separate terminal window is used to show CPU usage related to TLS encryption. One terminal window shows the output of the client application, and each instance of the Gateway has a separate terminal window. This is shown in Figure 44.
- Set HW kernel TLS to off and run measurement.
- Set HW kernel TLS to on and run measurement.
- Run measurement using the instance with no TLS.
- Compare results.



```
root@cc-gen-l-vrt-512: ~/repo/on-premise-storage-gateway
INFO:     127.0.0.1:53262 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53652 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53318 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53290 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53440 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53806 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53842 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53350 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53656 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:s3_gateway:[FileCacheManager] File served from cache: measurements-cloud-only/measurement_noTLS
INFO:     127.0.0.1:53834 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53512 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53300 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53388 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:54008 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53690 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53902 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53356 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53478 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53956 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53814 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53704 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:54064 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:s3_gateway:[FileCacheManager] File served from cache: measurements-cloud-only/measurement_noTLS
INFO:     127.0.0.1:53636 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:54120 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53820 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53538 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53748 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53862 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53966 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:54086 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53914 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53490 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53732 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:53926 - "GET /s3/measurements-cloud-only/measurement_noTLS HTTP/1.0" 200 OK
```

```
root@cc-gen-l-vrt-513: ~/repo/on-premise-storage-gateway/measurements/scripts
......................................................................................................
......................................................................................................
......................................................................................................
......................................................................................................
......................................................................................................
......................................................................................................
......................................................................................................
......................................................................................................
....................................................................
Stopping no TLS measurement
Averages for 1000 measurements, file size: 50000000, 16 nginx workers, 200 clients
HW kTLS (TLS offload): 9026.34ms, 74.21% CPU, user mode time 21.56s, kernel mode time 85.62s, 44.31Gbps
SW kTLS               : 10615.20ms, 80.85% CPU, user mode time 22.97s, kernel mode time 114.34s, 37.68Gbps
No TLS                : 13984.37ms, 21.89% CPU, user mode time 5.44s, kernel mode time 43.53s, 28.60Gbps
All measurements complete
root@cc-gen-l-vrt-513:~/repo/on-premise-storage-gateway/measurements/scripts# python3 run_MLNX_measurements.py
..
Files cached
Before continuing, make sure you build the images and run the containers.
Cache set
Cache set
Nginx worker processes count set to 16
Nginx worker processes count set to 16
Starting HW kTLS measurement
.......................................................................................................
.......................................................................................................
.......................................................................................................
.......................................................................................................
.......................................................................................................
.......................................................................................................
.......................................................................................................
.......................................................................................................
.............................................................
Stopping HW kTLS measurement
Starting SW kTLS measurement
```

```
root@cc-gen-l-vrt-512: ~/repo/on-premise-storage-gateway
INFO:     127.0.0.1:35834 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:35844 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:35950 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:35956 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:35984 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:35728 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36042 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:35936 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36066 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36000 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36030 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36048 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:35970 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:35960 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36126 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36052 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36168 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36118 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36102 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36092 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36078 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36154 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36142 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:s3_gateway:[FileCacheManager] File served from cache: measurements-cloud-only/measurement_kTLS
INFO:     127.0.0.1:36014 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36190 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36184 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36214 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
INFO:     127.0.0.1:36204 - "GET /s3/measurements-cloud-only/measurement_kTLS HTTP/1.0" 200 OK
```

```
root@cc-gen-l-vrt-512: ~
  1  [|||||||||||||||91.6%]   17 [|||||||||||||||84.8%]   33 [||||||||||      44.8%]   49 [|||||||||||||||68.5%]
  2  [||||||||||      44.1%]   18 [|||||||||||||||70.4%]   34 [|||||||||||||   61.1%]   50 [|||||||||||||||71.4%]
  3  [|||||||||||||||84.5%]   19 [|||||||||||||||76.6%]   35 [||||||||||      47.7%]   51 [|||||||||||||||84.7%]
  4  [||||||||||||    55.1%]   20 [|||||||||||||||88.8%]   36 [||||||||||      44.7%]   52 [|||||||||||||||72.2%]
  5  [||||||||||      45.3%]   21 [||||||||||||    56.5%]   37 [||||||||||||    54.9%]   53 [|||||||||||||||77.9%]
  6  [|||||||||||||||83.6%]   22 [|||||||||||||||94.8%]   38 [||||||||||||||  65.5%]   54 [||||||||||||||  66.0%]
  7  [|||||||||||||||76.5%]   23 [|||||||||||||   61.2%]   39 [|||||||||||     51.9%]   55 [|||||||||||||   58.0%]
  8  [||||||||||||    52.7%]   24 [|||||||||||||   60.0%]   40 [|||||||||||||||77.3%]   56 [|||||||||||||||94.1%]
  9  [||||||||||||||100.0%]   25 [|||||||||||||||84.7%]   41 [||||||          27.0%]   57 [|||||||||||||||68.2%]
  10 [|||||||||       39.9%]   26 [||||||||||||    56.4%]   42 [|||||           19.3%]   58 [|||||||||||||||83.8%]
  11 [||||||||||||||100.0%]   27 [|||||||||||||||87.9%]   43 [|||||||         26.0%]   59 [|||||||||||||||77.8%]
  12 [||||||          24.8%]   28 [|||||||||||||||88.8%]   44 [||||||          24.8%]   60 [|||||||||||||||80.8%]
  13 [|||||||||||     51.9%]   29 [||||||||||||||  65.8%]   45 [|||||||||||     51.0%]   61 [||||||||||||    53.3%]
  14 [|||||||||||||   59.6%]   30 [|||||||||||||||75.2%]   46 [|||||||||||     49.3%]   62 [|||||||||||||||81.8%]
  15 [|||||||||||||||71.5%]   31 [|||||||||||||||81.9%]   47 [||||||||||      45.7%]   63 [|||||||||||||||74.2%]
  16 [||||||||        33.1%]   32 [|||||||||||||||83.6%]   48 [|||||||||       34.0%]   64 [||||||||||||||  64.6%]
  Mem[|||                                   9.21G/504G]   Tasks: 231, 932 thr; 44 running
  Swp[                                       0K/10.00G]   Load average: 17.28 6.58 2.97
                                                          Uptime: 9 days, 01:12:28

    PID USER       PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 567567 root        20   0 1426M 98396 31616 S 108.  0.0  0:52.26 /usr/local/bin/python3.10 -c from multiprocessing.s
 567536 root        20   0 1042M 96616 30988 R 108.  0.0  0:46.65 /usr/local/bin/python3.10 -c from multiprocessing.s
 567529 root        20   0  946M 95360 31436 R 106.  0.0  0:50.34 /usr/local/bin/python3.10 -c from multiprocessing.s
 567534 root        20   0 1458M 98704 31576 R 103.  0.0  0:51.89 /usr/local/bin/python3.10 -c from multiprocessing.s
 567531 root        20   0  826M 93384 31064 R 103.  0.0  0:47.01 /usr/local/bin/python3.10 -c from multiprocessing.s
 567548 root        20   0 1002M 94724 31256 R 103.  0.0  0:45.43 /usr/local/bin/python3.10 -c from multiprocessing.s
 567558 root        20   0 1791M  133M 32704 R 103.  0.0  0:54.13 /usr/local/bin/python3.10 -c from multiprocessing.s
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Figure 44: TLS offloading demo showing the outputs of the two instances of the Gateway (top left: no TLS, bottom left: kernel TLS) client application (top right) and the CPU utilisation on the machine hosting the instances (bottom right).

The setup is the same as that used to evaluate **UC1.2, UC1.6**, **SIR.4** and **SIR.5** and is presented in Section 4.3.

4.3 Evaluation

The KPIs for the Secure Storage Use Case were selected so that they can be used to evaluate whether the main goals and objectives of the UC have been achieved. They also reflect the integration points with various platform components and features and are not limited to the core storage features.

Table 8: UC1 technical success criteria

ID	KPI	Success criterion	Estimated target value	Result
UC1.1	Read and write latency reduction with respect to existing cloud locations	Successful integration of edge devices into the SkyFlok and SERRANO ecosystem with the goal of reducing latency.	Reduction of 10 - 50%	For file sizes between 1MB and 15MB Read: 62-70% Write: 39-45%
UC1.2	Number of applications using the service simultaneously	Demonstration of client applications storing data in the edge/cloud infrastructure using S3 REST API.	20 instances	200

UC1.3	Reduction in time taken to encode and decode data with respect to a CPU-based solution	Demonstration of GPU- and FPGA- accelerated RLNC encoding and decoding algorithms running on the on-premises storage gateway.	Reduction of 20-30%	Up to 2.1x speedup on the cloud FPGA cards. Up to 6.6x speedup on the edge FPGA devices
UC1.4	Reduction in time taken to encrypt and decrypt data with respect to a CPU-based solution	Demonstration of GPU- or FPGA-accelerated AES-GCM encryption and decryption algorithms running on the on-premises storage gateway.	Reduction of 20-30%	Up to 220x speedup on the cloud T4 GPU devices. Up to 147x speedup on the edge Jetson GPU
UC1.6	Reduction in CPU load associated with encryption for TLS connections with respect to no hardware acceleration	Using DPU-based hardware acceleration for encryption of TLS connections on the On-premises storage gateway.	Reduction of 10-20%	23-37%
UC1.7	Storage task execution that involves the creation of a new storage policy without intervention from the user	Transparent operation with regard to the choice of storage locations. Each user application that issues a storage task should state its requirements. The SERRANO resource orchestrator should create/assign a storage policy automatically.	Demonstration successful	Demonstrated through Intent-driven operation and automatic storage policy creation demo
UC1.8	Storage task execution in a sandboxed environment.	Transparent operation deployed using the SERRANO orchestrator. Based on the Security Tiers defined in WP3.	Demonstration successful	Deployment described in this section

UC1.1 has been evaluated using a measurement script [10] specifically designed for this purpose. The script uses Boto3 to make S3 PUT_OBJECT and GET_OBJECT requests, measuring uploads and downloads separately. Each scenario has been measured 100 times with average values being shown in Figure 46 and Figure 47. To get a precise idea of how much time each part of each workflow takes, the code of the Gateway has been instrumented. An API designed to control measurements and retrieve results has been implemented specifically to evaluate these KPIs. It has been documented in Deliverable 6.7 [11].

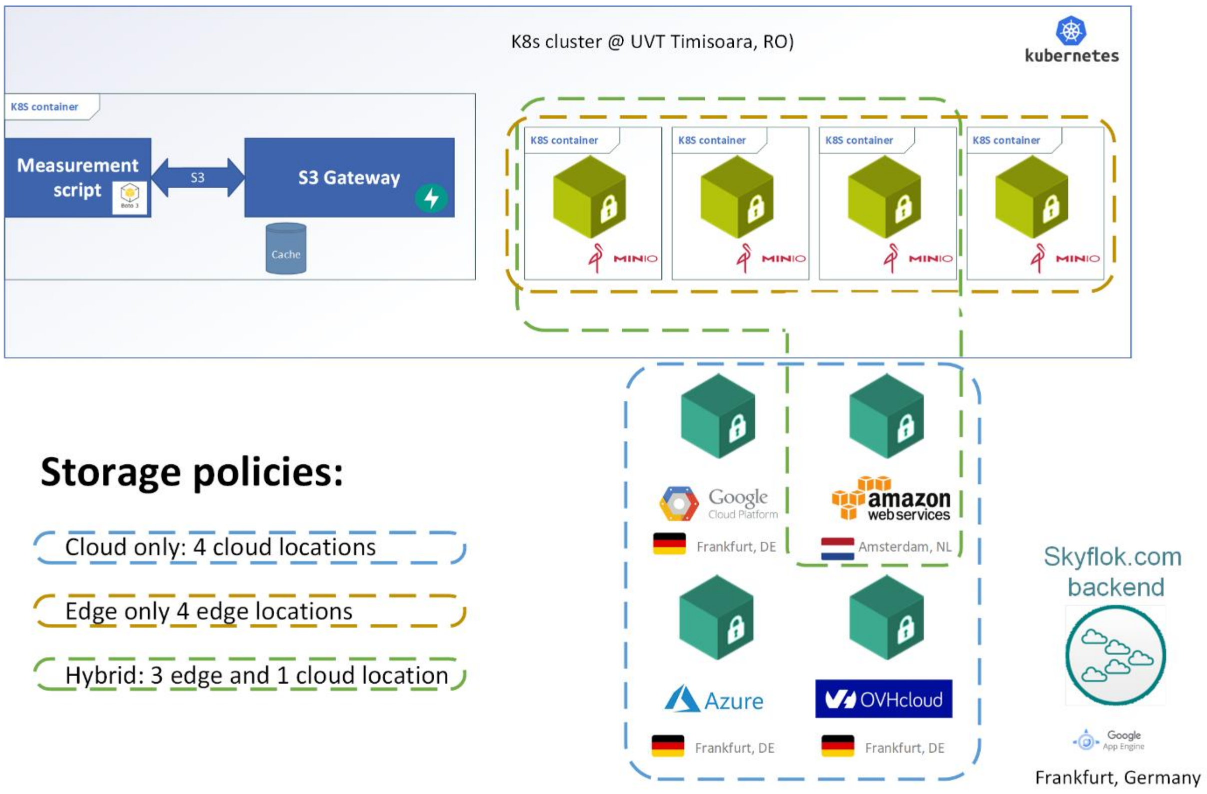


Figure 45: Measurement setup for evaluating read and write performance of different storage policies.

Figure 45 shows the measurement setup in terms of deployment. The Gateway is collocated with the measurement script in the same container, hosted on the K8s cluster of UVT in Timisoara. Regarding storage locations, 4 SERRANO edge devices have been deployed to the same cluster and 4 cloud locations have been selected from Central and Western Europe. These cover the three major global cloud providers (Amazon, Google and Microsoft) as well as one of the largest EU-based providers (OVH). To be able to compare how different storage strategies affect upload and download performance, we have created three separate storage policies and assigned each to a different S3 bucket. Every policy distributes data to 4 storage locations in a 3+1 erasure coded (Random Linear Network Coding over GF(2^8)) configuration. Thus, any 3 out of 4 locations are sufficient to retrieve the data. They all employ AES GCM with a key size of 256 bits for encryption and DEFLATE level 7 for compression. Caching is turned off and randomly generated file data is used. The cloud-only policy uses only cloud locations, the edge-only policy utilises only edge locations while the hybrid policy uses three SERRANO edge devices and one cloud location. The precise setup is shown on Figure 45.

Results for uploading files of sizes 1MB, 5MB, 10MB and 15MB are shown in Figure 46. We can see a major reduction in the time taken to upload the erasure coded fragments when comparing a cloud-only to and edge-only policy, with a roughly order of a magnitude difference. This translates into a reduction of the overall time taken (**UC1.1**) of between 39.4%-44.7%. The hybrid policy also results in faster file uploads, improvements vary between 19.5%-27.9%. We can also observe that the different parts of the workflow don't change based on file size or storage policy. A single exception is the retrieval of the upload links (explained in Deliverable 3.4 [9]), as the computational load required to create the signature is slightly lower for edge locations, compared to the cloud locations in question.

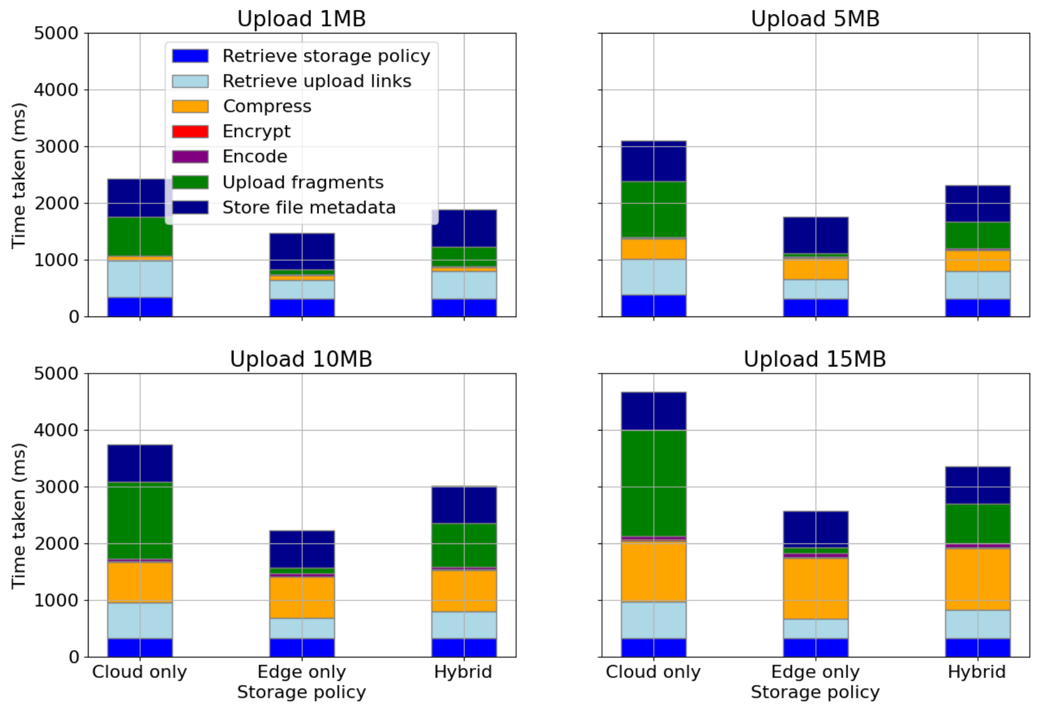


Figure 46: Time taken to upload a file, comparing a cloud-only an edge-only and a hybrid storage policy.

The improvement to download performance (**UC1.1**) shown in Figure 47 is even greater at between 62.4%-70.8% for the edge-only and 28.6%-41.5% for the hybrid policy.

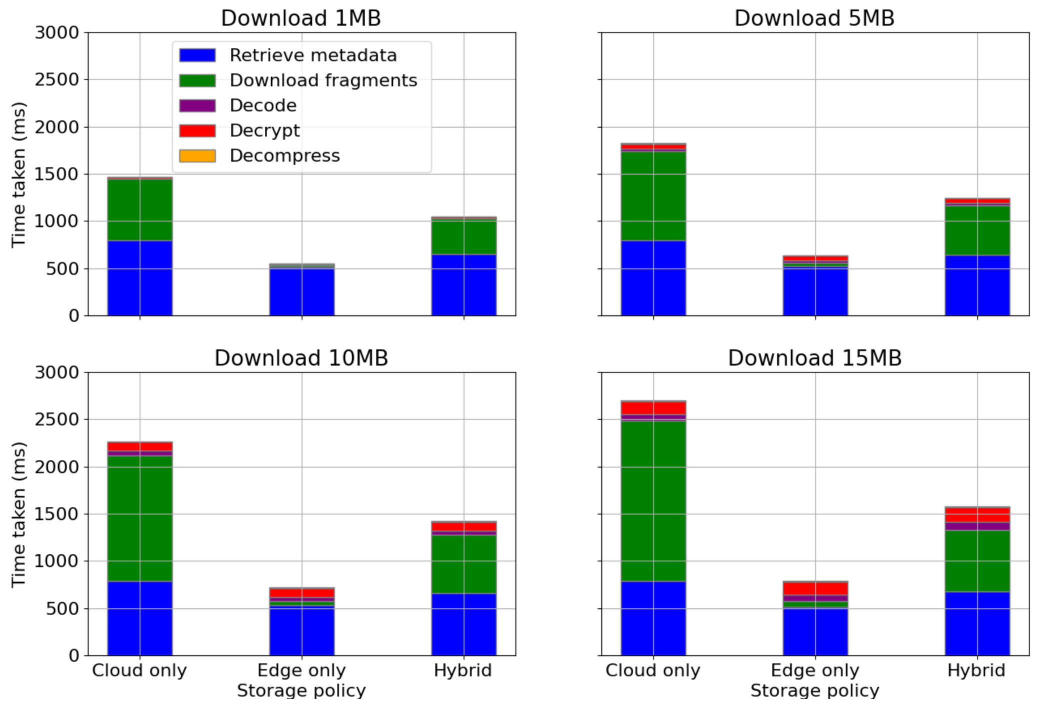


Figure 47: Time taken to download a file, comparing a cloud-only an edge-only and a hybrid storage policy

Given that this workflow entails both less data processing and fewer round trips to the Skyflok.com backend, performance is more heavily dictated by the time needed to download the erasure coded fragments. In fact, in this particular scenario where enough data is stored at the edge, the hybrid policy would match the edge-only policy with a download algorithm that favours fragments stored at the edge. The current naive algorithm always uses the cloud location.

We have also briefly evaluated the impact of caching using a separate measurement script [12], with results shown in Figure 48. Using caching with cloud-only storage policies improves performance by roughly an order of magnitude. The gains grow with file size. In comparison, edge-only policies show only a small benefit in absolute terms and a significant but more modest one in relative terms.

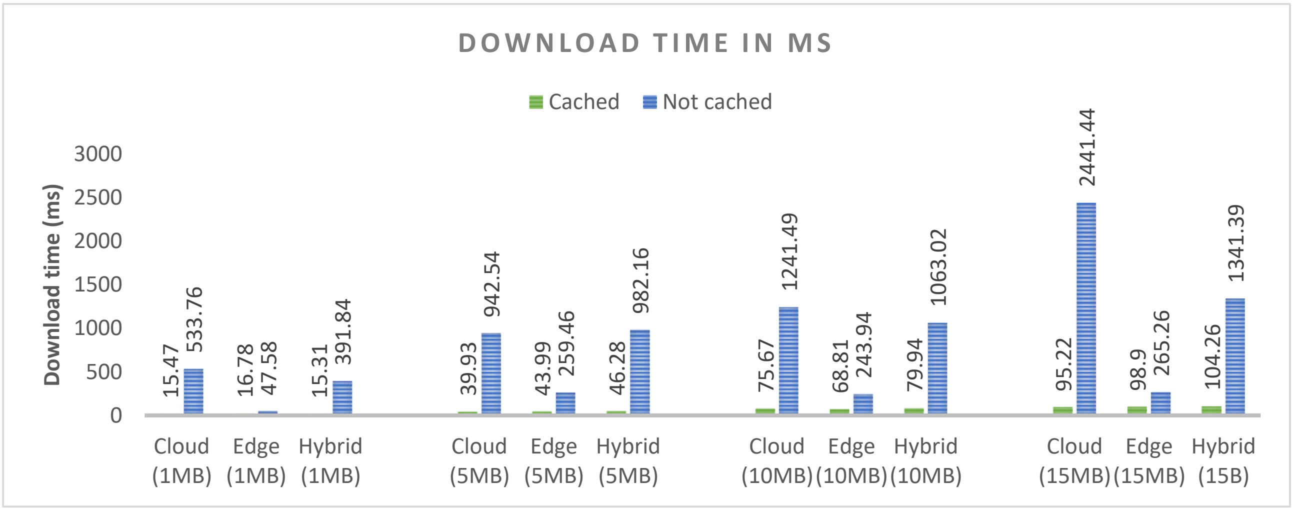


Figure 48: Impact of caching on the time needed to download a file.

We have also briefly examined the performance gains of multipart uploads for large files, compared to regular uploads. Multipart uploads allow the client to fragment a file into smaller parts, facilitating separate uploads for each fragment. This approach also makes it possible to parallelise the requests. We have conducted a single measurement for files of sizes between 10MB and 200MB, with the results shown on Figure 49. The experiment [13] involved evaluating segment sizes of 5MB and 10MB and allowed for a maximum concurrency of 5. This means that at most 5 HTTP requests were made by the client during each multipart upload. The Gateway was configured to use 8 worker processes, and caching was turned off. The cloud-only storage policy from the previous experiments was used.

For smaller files, we could not see any gain from using multipart uploads. In some cases, performance was actually worse compared to regular uploads. This can be explained by the additional HTTP requests that need to be made: first, a multipart upload must be created, then each part must be uploaded separately, and finally it must be completed. In these cases, the gains from parallelisation were overshadowed by the overhead of the additional requests. Starting from 50MB and up, the gains became more substantial and increased with file size. Uploading a 200MB file was almost 2x faster through the multipart upload workflow. We can also observe that, at least for larger files, larger part size is beneficial as it reduces the number of HTTP requests that must be made. We expect that users of the service would need to fine-tune the part size and the level of parallelism based on file size, outgoing connection bandwidth towards the storage locations and the CPU resources available at the Gateway. This can be done on a per-file basis as the Storage Service allows this level of flexibility.

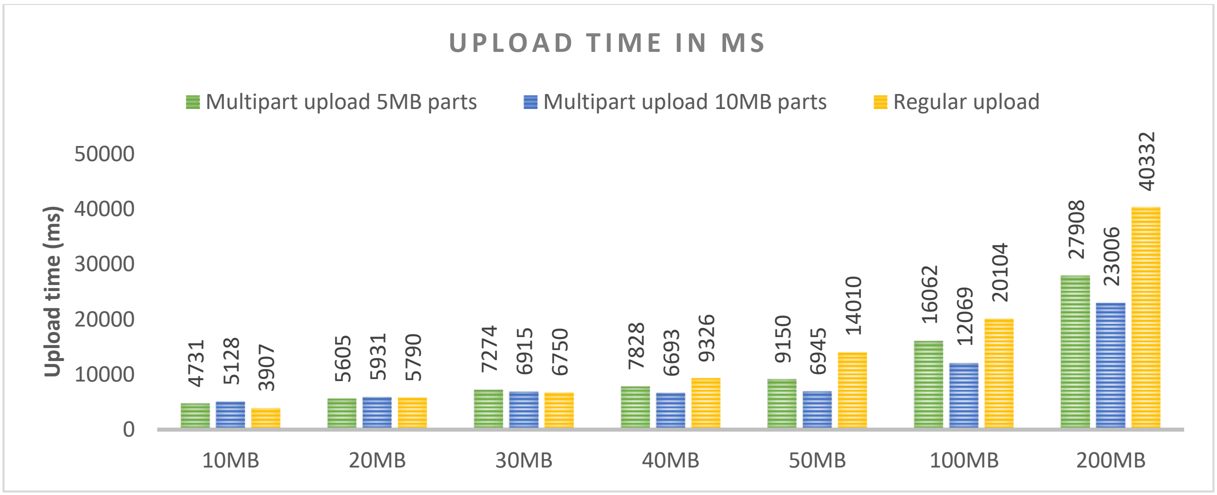


Figure 49: Comparing multipart uploads to regular uploads.

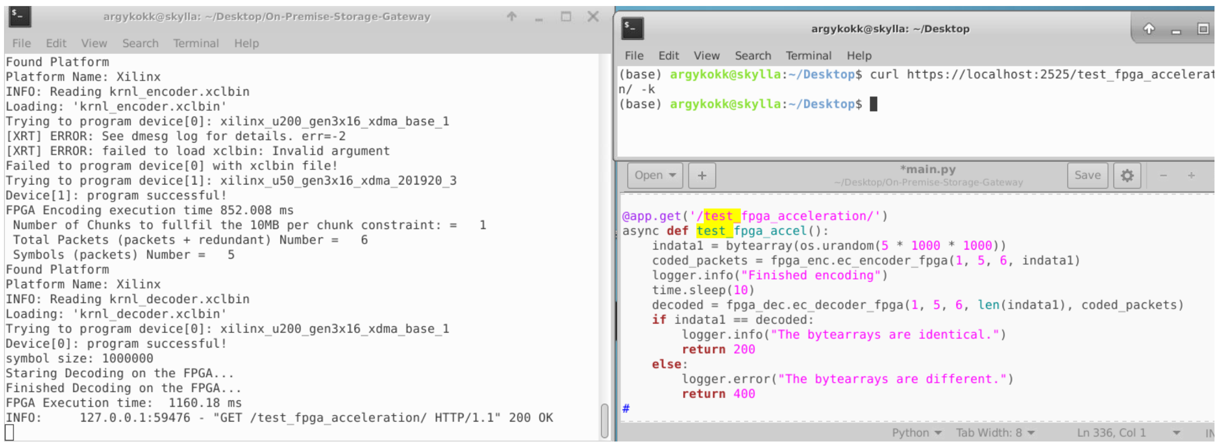
To integrate the FPGA-developed application into the UC1 flow and evaluate **UC1.3**, we decoupled the host executable from the FPGA accelerator by creating a shared library. This library encapsulates the application required to initialise the FPGA platform, perform memory transactions to and from the acceleration card, and return the outputs calculated by the FPGA. We developed a wrapper function in two versions: one for communication with the erasure-coding encoder shared library application and another for the decoder. The encoder/decoder accelerator parameters, as well as the input/output data, are provided through a Python function to the wrapper, which then initiates the application's shared object. Communication between the 'libified' application and the FPGA acceleration card is facilitated by invoking the Xilinx Runtime Environment (XRT) libraries.

The docker image for this execution setup was developed based on the Xilinx image libraries. These libraries are necessary to expose the Xilinx runtime libraries, which are executed inside the container, to the PCIe space of the host machine, including the installed FPGA hardware platforms.

REST endpoints on the Gateway are defined to allow the execution of the UC1 flow with or without the FPGA cards.

- The ***enable_fpga_acceleration*** endpoint is used to enable the use of the FPGA applications: curl https://localhost:2525/enable_fpga_acceleration/ -k
- The ***disable_fpga_acceleration*** endpoint is used to disable the use of the FPGA applications: curl https://localhost:2525/disable_fpga_acceleration/ -k

The following test was initially performed to evaluate the execution of the encode/decode FPGA applications inside this environment, by calling the ***test_fpga_acceleration*** endpoint.


argykokk@skylla: ~/Desktop/On-Premise-Storage-Gateway
File Edit View Search Terminal Help

```
Found Platform
Platform Name: Xilinx
INFO: Reading krnl_encoder.xclbin
Loading: 'krnl_encoder.xclbin'
Trying to program device[0]: xilinx_u200_gen3x16_xdma_base_1
[XRT] ERROR: See dmesg log for details. err=-2
[XRT] ERROR: failed to load xclbin: Invalid argument
Failed to program device[0] with xclbin file!
Trying to program device[1]: xilinx_u50_gen3x16_xdma_201920_3
Device[1]: program successful!
FPGA Encoding execution time 852.008 ms
 Number of Chunks to fullfil the 10MB per chunk constraint: =   1
 Total Packets (packets + redundant) Number =   6
 Symbols (packets) Number =   5
Found Platform
Platform Name: Xilinx
INFO: Reading krnl_decoder.xclbin
Loading: 'krnl_decoder.xclbin'
Trying to program device[0]: xilinx_u200_gen3x16_xdma_base_1
Device[0]: program successful!
symbol size: 1000000
Staring Decoding on the FPGA...
Finished Decoding on the FPGA...
FPGA Execution time:  1160.18 ms
INFO:     127.0.0.1:59476 - "GET /test_fpga_acceleration/ HTTP/1.1" 200 OK
```

argykokk@skylla: ~/Desktop
File Edit View Search Terminal Help

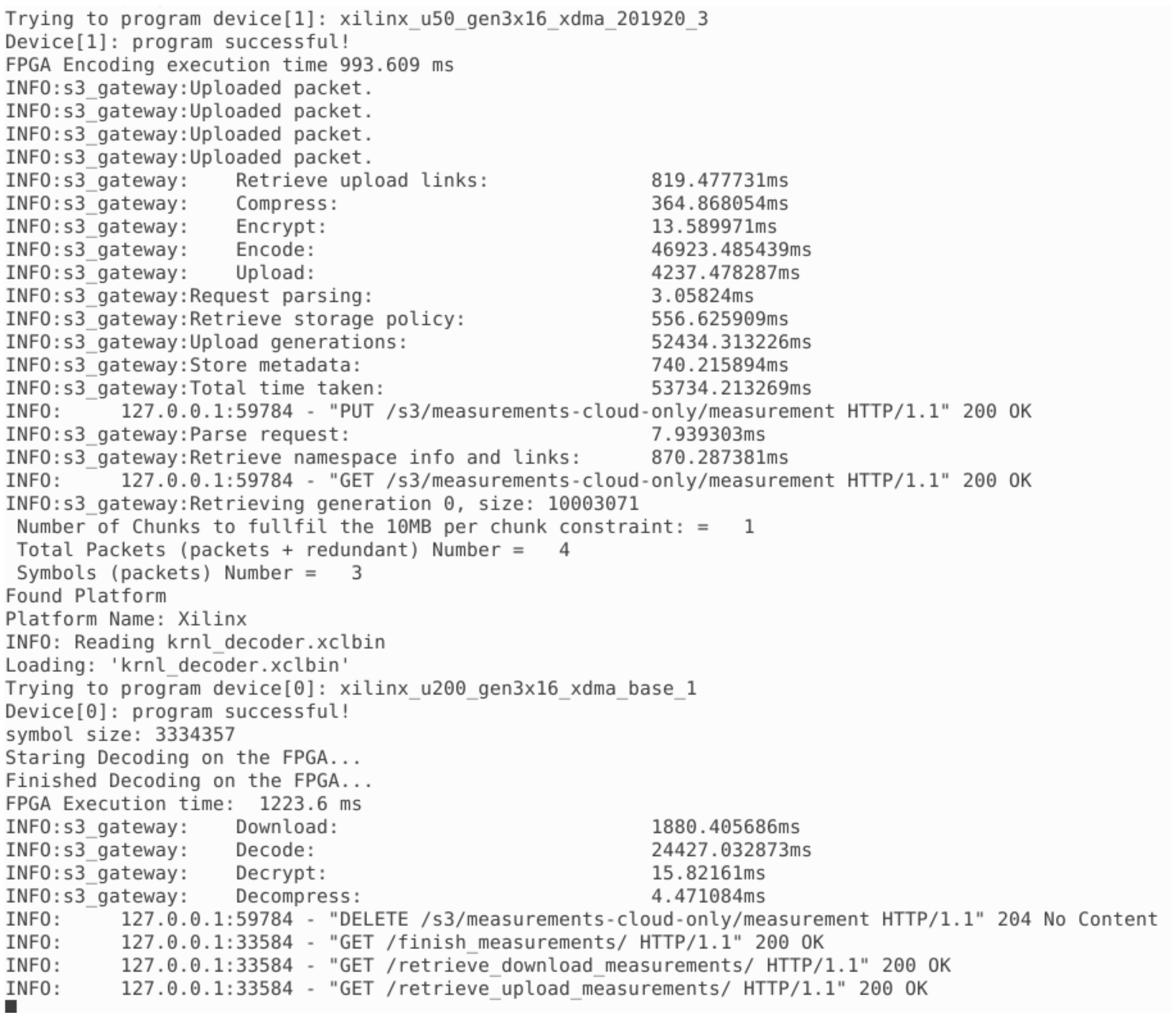
```
(base) argykokk@skylla:~/Desktop$ curl https://localhost:2525/test_fpga_accelerat
n/ -k
(base) argykokk@skylla:~/Desktop$ 
```

```
@app.get('/test_fpga_acceleration/')
async def test_fpga_accel():
    indata1 = bytearray(os.urandom(5 * 1000 * 1000))
    coded_packets = fpga_enc.ec_encoder_fpga(1, 5, 6, indata1)
    logger.info("Finished encoding")
    time.sleep(10)
    decoded = fpga_dec.ec_decoder_fpga(1, 5, 6, len(indata1), coded_packets)
    if indata1 == decoded:
        logger.info("The bytearrays are identical.")
        return 200
    else:
        logger.error("The bytearrays are different.")
        return 400
#
```

Figure 50: FPGA acceleration test UC1.

In the above test, the U50 FPGA card is used for the erasure coding encoder application and the U200 card for the decoding task. An input of approximately 5MB is provided to the encoder high-level function alongside the encoding parameters (i.e number of encoding chunks, number of encoding symbols and overall number of encoding packets). This is done similarly for the decoding task.

For the end-to-end FPGA-based execution a 10MB test file was selected. In this flow, the UC1 stages are executed in the system's CPU and the encoding/decoding tasks on the FPGA platforms, as shown in Figure 51. The call to the endpoints, delays invoking the shared object that calls the FPGA accelerator. Additionally, there is an execution time overhead induced by the Xilinx runtime libraries that are called through the containerised shared memory space. However, the execution time of the encoder and decoder kernels show an improvement (up to 2.1x on the same platforms) compared to the execution of the CPU-based application, as those measurements. The corresponding evaluation is described in Deliverable D4.4 [14].



```
Trying to program device[1]: xilinx_u50_gen3x16_xdma_201920_3
Device[1]: program successful!
FPGA Encoding execution time 993.609 ms
INFO:s3_gateway:Uploaded packet.
INFO:s3_gateway:Uploaded packet.
INFO:s3_gateway:Uploaded packet.
INFO:s3_gateway:Uploaded packet.
INFO:s3_gateway:    Retrieve upload links:              819.477731ms
INFO:s3_gateway:    Compress:                           364.868054ms
INFO:s3_gateway:    Encrypt:                            13.589971ms
INFO:s3_gateway:    Encode:                             46923.485439ms
INFO:s3_gateway:    Upload:                             4237.478287ms
INFO:s3_gateway:Request parsing:                        3.05824ms
INFO:s3_gateway:Retrieve storage policy:                556.625909ms
INFO:s3_gateway:Upload generations:                     52434.313226ms
INFO:s3_gateway:Store metadata:                         740.215894ms
INFO:s3_gateway:Total time taken:                       53734.213269ms
INFO:     127.0.0.1:59784 - "PUT /s3/measurements-cloud-only/measurement HTTP/1.1" 200 OK
INFO:s3_gateway:Parse request:                          7.939303ms
INFO:s3_gateway:Retrieve namespace info and links:      870.287381ms
INFO:     127.0.0.1:59784 - "GET /s3/measurements-cloud-only/measurement HTTP/1.1" 200 OK
INFO:s3_gateway:Retrieving generation 0, size: 10003071
 Number of Chunks to fullfil the 10MB per chunk constraint: =   1
 Total Packets (packets + redundant) Number =    4
 Symbols (packets) Number =    3
Found Platform
Platform Name: Xilinx
INFO: Reading krnl_decoder.xclbin
Loading: 'krnl_decoder.xclbin'
Trying to program device[0]: xilinx_u200_gen3x16_xdma_base_1
Device[0]: program successful!
symbol size: 3334357
Staring Decoding on the FPGA...
Finished Decoding on the FPGA...
FPGA Execution time:  1223.6 ms
INFO:s3_gateway:    Download:                           1880.405686ms
INFO:s3_gateway:    Decode:                             24427.032873ms
INFO:s3_gateway:    Decrypt:                            15.82161ms
INFO:s3_gateway:    Decompress:                         4.471084ms
INFO:     127.0.0.1:59784 - "DELETE /s3/measurements-cloud-only/measurement HTTP/1.1" 204 No Content
INFO:     127.0.0.1:33584 - "GET /finish_measurements/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:33584 - "GET /retrieve_download_measurements/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:33584 - "GET /retrieve_upload_measurements/ HTTP/1.1" 200 OK
```

Figure 51: end-to-end execution of UC1 with FPGA-accelerated erasure coding.

To evaluate **UC1.2**, **UC1.6**, **SIR.4,** and **SIR.5**, we have implemented a different measurement setup, deployed to an Nvidia's lab located in Israel. The overview on Figure 52 shows the two host machines (named 512 and 513), each equipped with an Nvidia DPU (on a Mellanox Technologies MT2892 Family [ConnectX-6 Dx]) and directly linked to each other through a 100 Gbps LAN connection. They both have 512GB of RAM and two AMD EPYC 73F3 CPUs with 16 cores each, installed into separate sockets. Host machine 512 has the Gateway in two flavours: one with kernel TLS and a baseline one with no TLS, both running inside Docker containers. Integration has been achieved by building a custom version of the OpenSSL 3.0.0 library. This library is loaded into the kernel TLS container and automatically detects what HW resources are available. If an appropriate DPU is detected and HW TLS offloading is enabled, computations related to TLS encryption are performed on the DPU. To be able to more accurately measure and better separate CPU load associated with TLS encryption, the OpenSSL library is loaded by Nginx, rather than the Gateway's Python application. Nginx acts as a TLS termination proxy, forwarding incoming HTTPS requests to the Python application as HTTP requests through the host's loopback interface. Conversely, Nginx acts as a simple HTTP proxy in the container without TLS, forwarding HTTP requests without changes to the Gateway.

Host machine 513 represents the clients and can simulate a large number of parallel requests. It is running a custom measurement script [15] that automates all the steps and makes results consistent and repeatable.

The experiment is focused on measuring:

- CPU usage of Nginx worker processes – user time and kernel CPU time. This is done on machine 512 by looking at /proc/{PID}/stat. We found this gave the most accurate measurement of how many CPU cycles are used by Nginx.
- Read throughput, measured at the client to provide an accurate representation of the capabilities of the Gateway.

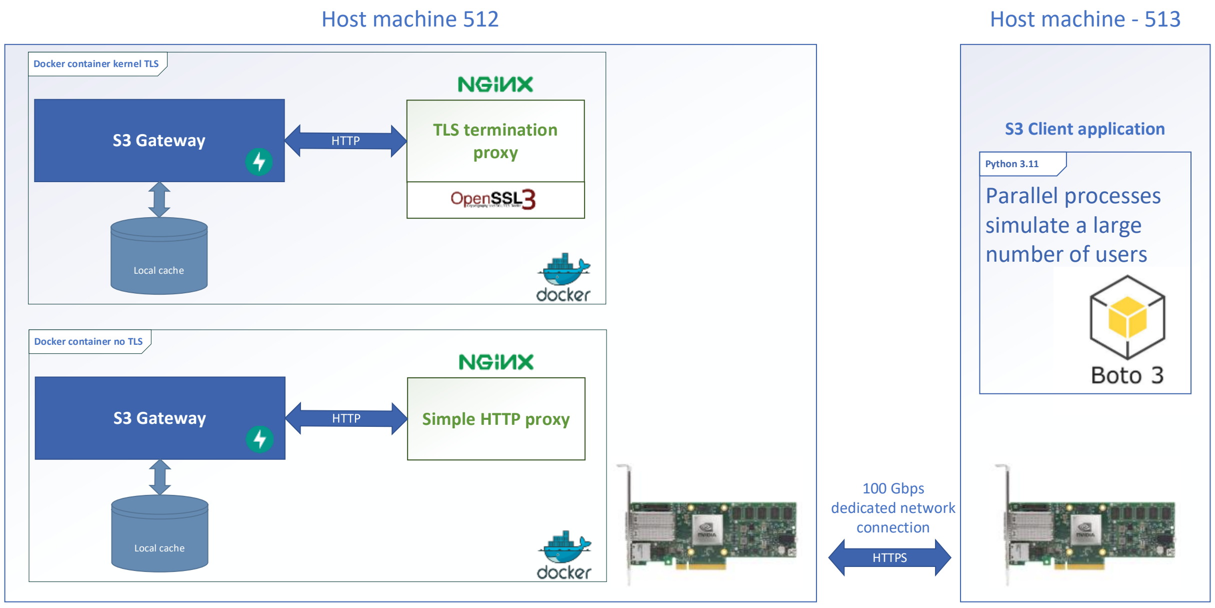


Figure 52: Overview of measurement setup for evaluating TLS-offloading in the context of the Secure Storage use case.

We have fixed the number of workers processes the Python Gateway uses to 32. Given the host's hardware, this should be sufficient not to become a performance bottleneck and thus affect results. File caching is set to on with the same intent. Each measurement is repeated 1000 times, and a 50MB file is uploaded, then downloads are measured. We compare HW kernel TLS (TLS offloading) with SW kernel TLS and the baseline approach with no TLS. We look at different concurrent numbers of clients of 20, 50, and 100. This is proper parallelism as the client application uses a different process for each HTTP request (avoiding the limitations imposed by the Python GIL), taken from a pool of processes. This allows us to measure the system at various loads. We also look at different numbers of Nginx worker processes: 1, 2, 3 and 4. This allows us to evaluate how the Storage Service performs given scenarios with different CPU bottlenecks.

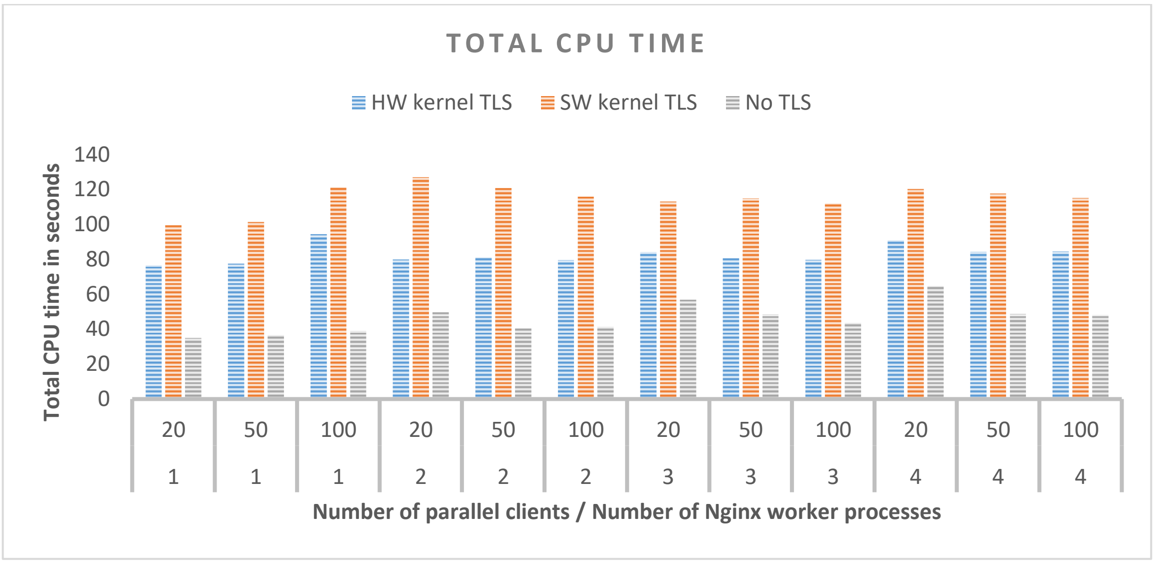


Figure 53: Total CPU time used by Nginx worker processes while serving storage requests.

Looking at the plots of Figure 53, we can clearly see the TLS offloading working, reducing the total number of CPU cycles (**UC1.6** and **SIR.5**) used by Nginx worker processes by 23%-37%, when compared against SW kernel TLS. However, this is a pessimistic approximation (lower bound) as Nginx worker processes must also perform tasks not related to TLS encryption. as well. An optimistic approximation (upper bound) can be inferred by subtracting the CPU usage measured on the no TLS container, considering that the product of the operation is close to the actual CPU time used for TLS encryption. This results in a decrease between 35%-60%. We can observe that these gains are achieved even with 4 Nginx worker processes, a scenario that is less bottlenecked by the CPU. Figure 54 shows very similar gains when observing the kernel CPU time separately. This is somewhat expected given the use of kernel TLS.

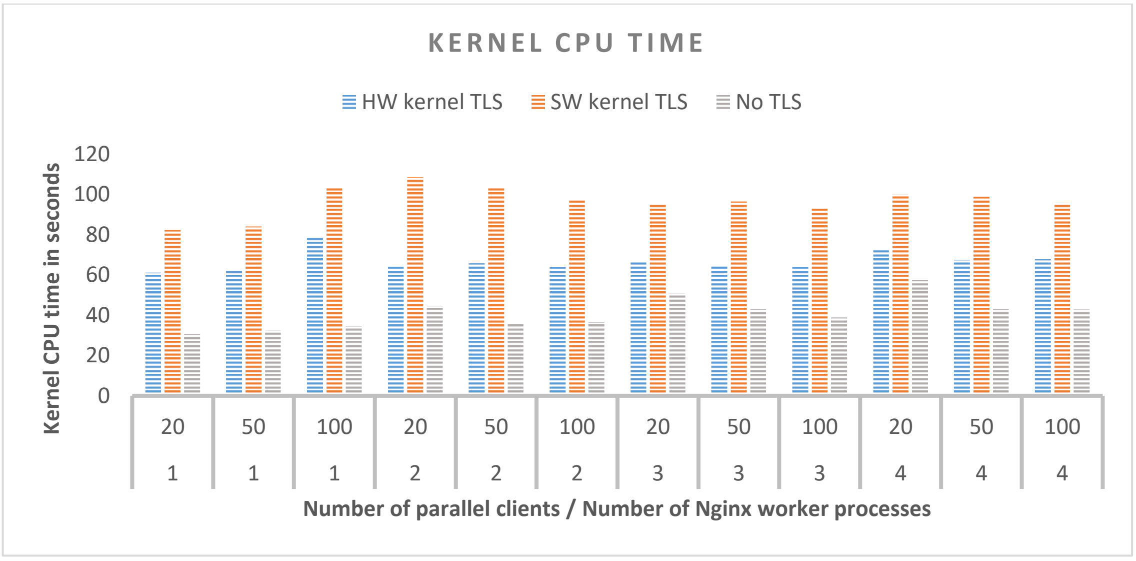


Figure 54: Kernel CPU time used by Nginx worker processes while serving storage requests.

We also evaluated the total read throughput of the SERRANO-enhanced Storage service when using TLS offloading and have seen an improvement between 6% and 60% compared to SW kernel TLS (**SIR.4**). The improvement varies greatly with the number of Nginx worker processes. With smaller numbers, the TLS encryption becomes more of a bottleneck, and thus, the gains are more pronounced.

We have tested this setup with up to 200 clients accessing the service in parallel (**UC1.2**) and 16 Nginx worker processes. This resulted in the throughput figures included in Table 9.

Table 9: Throughput measured technical success criteria

HW kernel TLS	SW kernel TLS	No TLS
35.19 Gbps	33.51 Gbps	39.14 Gbps

It would be possible to serve even more parallel clients. However, the number of open sockets of the measurement application was so high that the OS-level limit on the number of files open by any single process was reached. By relaxing this constraint, higher numbers can be achieved.

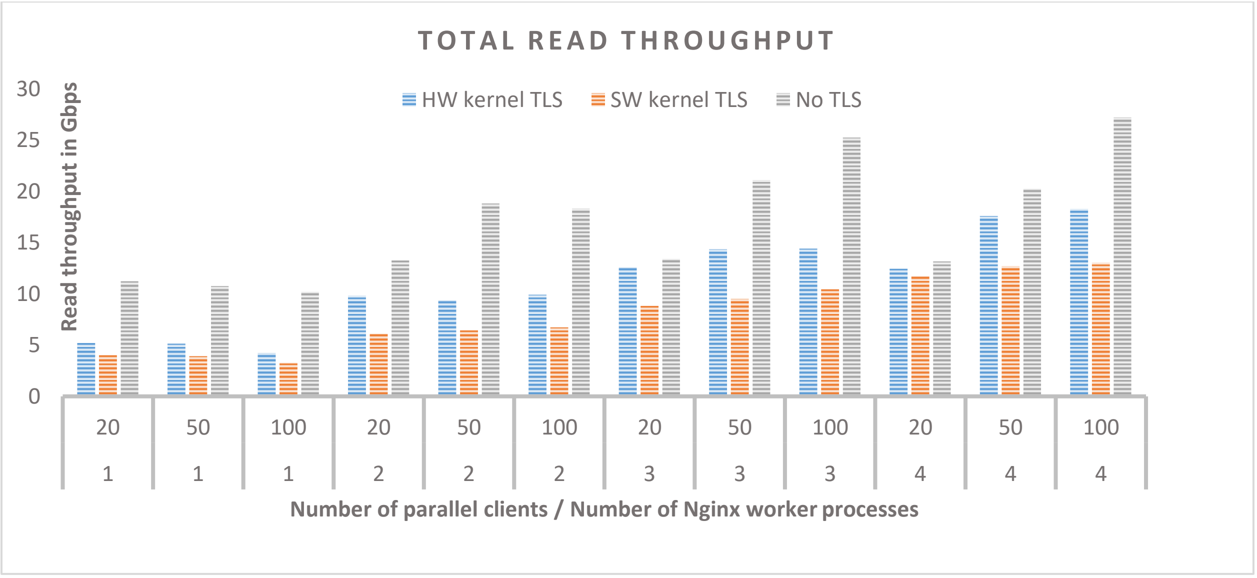


Figure 55: Read throughput of the SERRANO-enhanced Storage Service using TLS offloading.

The successful achievement of **UC1.7** has been shown using the intent-driven operation and automatic storage policy creation demonstrator described in Section 4.2.

The achievement of **UC1.8** is described in the following. The Gateway is spawned in a sandboxed environment as a container deployment on K8s. This is transparent to the administrator and the user of the service, as the sandboxed environment is embedded in the systems software running on the worker and control-plane nodes. Specifically, the installation of the storage gateway at the local cluster is done according to the instructions from the relevant partner (CC), using the provided helm chart. The changes needed to enable sandboxing rely on the runtimeClassName parameter used for the specific deployment. In this case, the runtime class refers to the underlying low-level container runtime used to host the container: `kata-fc`. This container runtime boots an AWS Firecracker microVM and spawns the Gateway's container in this microVM, isolating it from the rest of the workloads running on the infrastructure. In the snippet below we show that the service is running as advertised:

```
# kubectl describe pod -n serrano-cc cc-gw-on-premise-storage-gateway-56f74b8f6-m2n8b
Name:                 cc-gw-on-premise-storage-gateway-56f74b8f6-m2n8b
Namespace:            serrano-cc
Priority:             0
Runtime Class Name:   kata-fc
Service Account:      default
Node:                 bf/192.168.4.117
Start Time:           Mon, 18 Dec 2023 21:35:22 +0000
Labels:               app.kubernetes.io/instance=cc-gw
                      app.kubernetes.io/name=on-premise-storage-gateway
                      pod-template-hash=56f74b8f6
Annotations:                                                cni.projectcalico.org/containerID:
6ec104d93cc675fcd4bf21a8c59832a05b9a88788f5e8a5f5c629027a0944e5c
[snipped]
Status:               Running
IP:                   192.168.231.58
IPs:
  IP:           192.168.231.58
Controlled By:  ReplicaSet/cc-gw-on-premise-storage-gateway-56f74b8f6
Containers:
  on-premise-storage-gateway:
    Container ID:   containerd://8559a7ceb761f0635cfbae6560eb97470dddaba41c7f0acc9e3ac048e65bda82
    Image:                                                   harbor.nbfc.io/nubificus/serrano/cc-gw-
signed@sha256:0c3a96549c792a19dcd17f3e5f60db198bbe1d31d954e80bc39ca2a0d252a893
    Image      ID:                                           harbor.nbfc.io/nubificus/serrano/cc-gw-
signed@sha256:0c3a96549c792a19dcd17f3e5f60db198bbe1d31d954e80bc39ca2a0d252a893
    Port:           2525/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Mon, 18 Dec 2023 21:35:34 +0000
    Ready:          True
    Restart Count:  0
    Liveness:       http-get http://:http/ delay=10s timeout=1s period=10s #success=1 #failure=3
    Readiness:      http-get http://:http/ delay=10s timeout=1s period=10s #success=1 #failure=3
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-m4fgl (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
[snipped]
Events:
  Type    Reason          Age   From               Message
  ----    ------          ----  ----               -------
  Normal  Scheduled       13m   default-scheduler  Successfully assigned serrano-cc/cc-gw-on-premise-
storage-gateway-56f74b8f6-m2n8b to bf
  Normal  AddedInterface  13m   multus             Add eth0 [192.168.231.58/32] from k8s-pod-network
  Normal  Pulling         13m   kubelet            Pulling image "harbor.nbfc.io/nubificus/serrano/cc-
gw-signed@sha256:0c3a96549c792a19dcd17f3e5f60db198bbe1d31d954e80bc39ca2a0d252a893"
  Normal    Pulled                  13m      kubelet                      Successfully  pulled  image
"harbor.nbfc.io/nubificus/serrano/cc-gw-
signed@sha256:0c3a96549c792a19dcd17f3e5f60db198bbe1d31d954e80bc39ca2a0d252a893"    in    563.786005ms
(563.806985ms including waiting)
  Normal  Created         13m   kubelet            Created container on-premise-storage-gateway
  Normal  Started         13m   kubelet            Started container on-premise-storage-gateway
```

If we examine the node where this container is running (`bf`), we can see the AWS Firecracker instance running jailed, with the specific container id:

```
root@bf:~# ps -ef |grep 6ec104d93cc675fcd4bf21a8c59832a0
root     4115085       1  0 21:35 ?        00:00:00 /opt/kata/bin/containerd-shim-kata-v2 -namespace
k8s.io    -address    /run/containerd/containerd.sock    -publish-binary    /usr/bin/containerd    -id
6ec104d93cc675fcd4bf21a8c59832a05b9a88788f5e8a5f5c629027a0944e5c
root     4115141 4115085  0 21:35 ?        00:00:05 /firecracker -id 6ec104d93cc675fcd4bf21a8c59832a0
-start-time-us   1142763162158   -start-time-cpu-us   76147   -parent-cpu-time-us   0   -config-file
/fcConfig.json
```

Additionally, this sandboxed container is running in a separate namespace, with a specific security policy enabled:

```
# kubectl describe ns serrano-cc
Name:         serrano-cc
Labels:       ubernetes.io/metadata.name=serrano-cc
              policy.sigstore.dev/include=true
Annotations:  <none>
Status:       Active
```

This feature, detailed in D3.4 [9] and D6.7 [11], prevents non-signed container images from running on this specific namespace (attestation). This signature is verified by GitHub's OpenID service, along with the original signature, issued by GitHub itself, when the container image is being built. This process ensures that the container image is legitimate and verified by an authenticated third-party. More details for the deployed policy can be found below:

```
# kubectl describe clusterimagepolicies.policy.sigstore.dev
Name:         nbfc-policy
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  policy.sigstore.dev/v1beta1
Kind:         ClusterImagePolicy
Metadata:
  Creation Timestamp:  2023-12-18T21:32:07Z
  Finalizers:
    clusterimagepolicies.policy.sigstore.dev
  Generation:        1
  Resource Version:  585379312
  UID:               72694581-b673-4b2f-9e69-c0ed2a1f7d36
Spec:
  Authorities:
    Keyless:
      Identities:
        Issuer:           https://token.actions.githubusercontent.com
        Subject Reg Exp:  https://github.com/nubificus/.*/.github/workflows/*@*
      URL:                https://fulcio.sigstore.dev
    Name:                 authority-0
  Images:
    Glob:  **
  Mode:    enforce
Status:
  Conditions:
    Last Transition Time:  2023-12-18T21:32:07Z
    Status:                True
    Type:                  ConfigMapUpdated
    Last Transition Time:  2023-12-18T21:32:07Z
    Status:                True
    Type:                  KeysInlined
    Last Transition Time:  2023-12-18T21:32:07Z
    Status:                True
    Type:                  PoliciesInlined
    Last Transition Time:  2023-12-18T21:32:07Z
    Status:                True
    Type:                  Ready
  Observed Generation:     1
Events:
  Type    Reason          Age   From                         Message
  ----    ------          ----  ----                         -------
  Normal  FinalizerUpdate  29m  clusterimagepolicy-controller  Updated "nbfc-policy" finalizers
```

5 FinTech Use Case

The Fintech UC demonstrates the automatic optimisation of InbestMe's dynamic investment management. The SERRANO project contributes to the investment management use case by accelerating critical parts of the overall Dynamic Portfolio Optimisation (DPO) application and providing a framework that simplifies the deployment, management, operation, and monitoring. Additionally, for the provision of investment management as a service (SaaS), the UC benefits from the Secure Storage service that keeps the data of third parties secure. The UC also demonstrates the advantages of cloud-based acceleration of various computationally intensive operations. The SERRANO platform is also beneficial for InbestMe because it reduces cloud costs and improves the quality of the services. The use case provider is able to easily deploy multiple instances of the investment management platform on local as well as external cloud resources.

5.1 Use case description

This UC aims to demonstrate the cloud continuum capabilities of the SERRANO platform within the context of dynamic investment portfolio optimisation. The UC is based on microservices architecture while leveraging the project developments for transparent deployment on the cloud as well as the seamless execution of the SERRANO accelerated kernels in cloud and edge resources that include FPGA or GPU accelerators and HPC platforms.

Portfolio optimisation (Figure 56) is a very representative and demanding investment management process that can benefit from SERRANO. It starts by getting the market data required for the analysis, which are analysed by applying a set of technical calculations. Subsequently, forecasting algorithms and various investment strategies are applied, in parallel, in the investment instruments. The output from the forecasting and the investment strategies is used for creating new investment profiles. The investment profiles are again analysed by applying forecasting methods and back testing. Finally, the investment profiles are rebalanced to match the expected distribution of the investment profiles, but this component is not part of the SERRANO project.

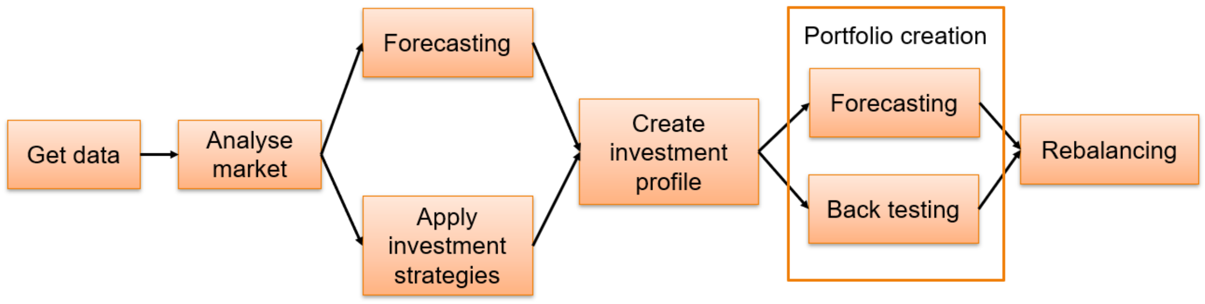


Figure 56: Portfolio optimisation workflow.

The Dynamic Portfolio Optimisation (DPO) application, is a container-based application deployed on the SERRANO platform using the SERRANO SDK. This application primarily focuses on portfolio construction and optimisation.

It begins by gathering necessary data like market data, investment profiles, asset classes, and strategy rules. These data elements are securely stored using the Secure Storage Service of the SERRANO platform. Following data acquisition, the application conducts market analysis, a process that employs the kernels Black Scholes, Kalman, Savgol, and wavelet that require substantial computational resources. To enhance this step's efficiency, the DPO application utilises the seamless access to the SERRANO-accelerated kernels provided by the SERRANO SDK.

These accelerated kernels are instrumental in calculating technical indicators for various investment instruments, leveraging historical price data secured through the platform's Secure Storage service. The overall process and interactions between the DPO application's services and the SERRANO platform components are depicted in Figure 57. The figure highlights how these components are integrated using the SERRANO SDK and the interfaces exposed by the SERRANO platform services.

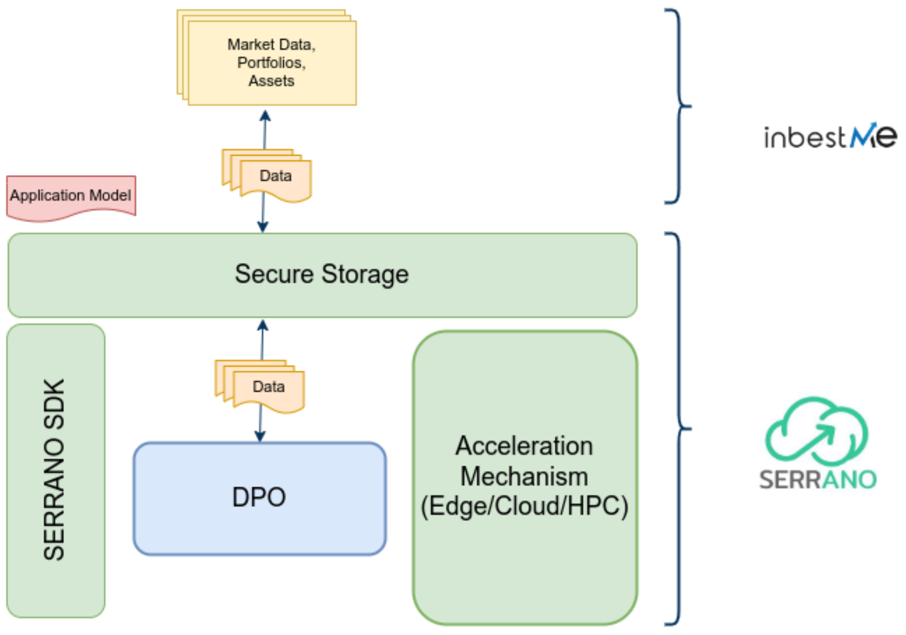

Market Data, Portfolios, Assets
Data
Application Model
Secure Storage
Data
SERRANO SDK
DPO
Acceleration Mechanism (Edge/Cloud/HPC)
inbestMe
SERRANO

Figure 57: Interactions between the fintech use case DPO service and SERRANO platform core components.

5.2 Demo

The following sections present a series of demonstrations highlighting the application of Dynamic Portfolio Optimisation (DPO) within the SERRANO framework, focusing on financial portfolio analysis and optimisation. These demonstrations encompass a range of experiments, from simulating financial portfolio analysis on the SERRANO platform to optimizing investment portfolios for balancing risk and return, and a comparative performance analysis of DPO using the SERRANO acceleration mechanisms across varied datasets. Each evaluation scenario leverages key components of the SERRANO framework and evaluates crucial KPIs like cloud adaptation, instance deployment, and rates of market and portfolio analysis. These demonstrations collectively showcase the effectiveness, efficiency, and scalability of the SERRANO framework in real-world financial applications.

5.2.1 Simulating Financial Portfolio Analysis: Exploring DPO Application on SERRANO platform.

This evaluation scenario aims to assess the capabilities of the SERRANO framework in the context of financial portfolio analysis. The focus is on accessing the DPO application through a browser, utilizing the UVT Kubernetes infrastructure, and evaluating various SERRANO functionalities, including INB components and SERRANO platform components, such as the library of SERRANO-accelerated kernels, secure storage service, and secure and trusted execution through SERRANO's lightweight virtualization mechanisms. The study also considers key performance indicators (KPIs) outlined in D6.2 (M27), specifically the conversion/adaption to cloud-based containers, independent instances deployment, and the rate of market analysis.

The experiment involves connecting to the DPO application via https://dpoapp.services.cloud.ict-serrano.eu/api/v1/dpoapp/ on the UVT Kubernetes platform. The input requirements are stored securely in CC's secure storage, and necessary inputs are provided for the DPO to run. Upon completion of execution, the resulting required metrics are downloaded locally.

The KPIs measured align with Deliverable D6.6 Section 5.2.2 which are also listed and further elaborated in this Deliverable in Section 5.3 and include the following:

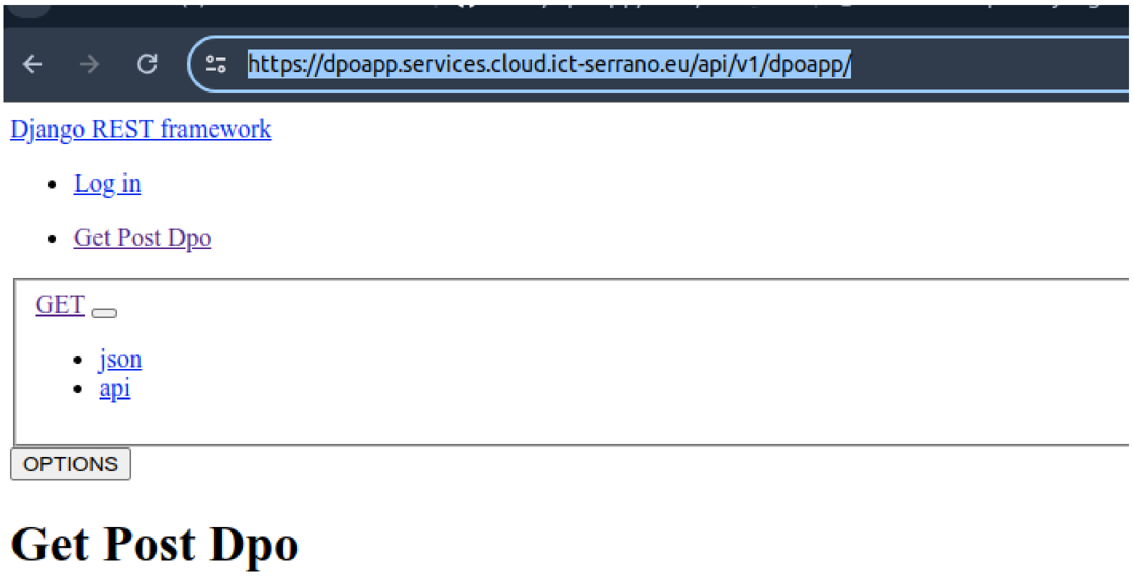
1. Conversion/Adaption to Cloud-Based Containers: Assessing the system's ability to adapt to cloud-based container environments, emphasizing flexibility and scalability.
2. Independent Instances Deployment: Evaluating the deployment of independent instances within the SERRANO framework to gauge its efficiency and resource utilisation.
3. Rate of Market Analysis: Measuring the speed and efficiency of market analysis processes within the financial portfolio context.

This experiment is going to be demonstrated in real-time so as to show the monitoring and recording mechanisms employed to capture the performance and outcomes of the experiment.

Following, we describe a comprehensive step-by-step overview of the DPO application's functionality, accompanied by illustrative images.

Step 1: Accessing the DPO Application

- Begin by visiting the publicly available URL of the application at https://dpoapp.services.cloud.ict-serrano.eu/api/v1/dpoapp/


https://dpoapp.services.cloud.ict-serrano.eu/api/v1/dpoapp/

Django REST framework

- Log in
- Get Post Dpo

GET

- json
- api

OPTIONS

Get Post Dpo

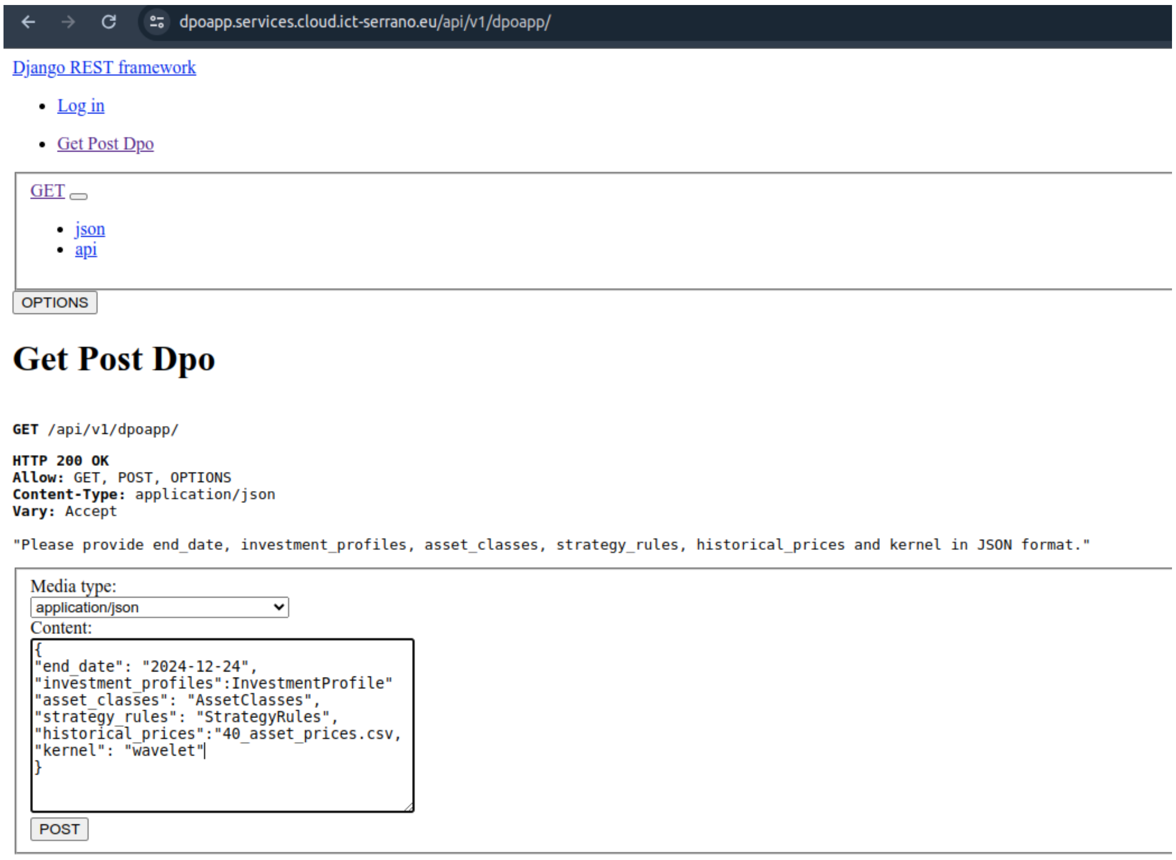
Figure 58: DPO publicly available API.

Step 2: Configuring Parameters and Initiating DPO Execution

- Specify the parameters of the required data in JSON format and submit a POST request. An example input format is provided below:

```
{
    "end_date": "2024-12-24",

    "investment_profiles": "InvestmentProfiles",

    "asset_classes": "AssetClasses",

    "strategy_rules": "StrategyRules",

     "asset_prices.csv",

    "kernel": "wavelet"
}
```

It is required to ensure that all files mentioned above are pre-stored in the SERRANO secure storage service for successful application access and execution. The 'kernel' parameter can be one of the following: wavelet, Savgol, Kalman, Black Scholes.


dpoapp.services.cloud.ict-serrano.eu/api/v1/dpoapp/

Django REST framework

- Log in
- Get Post Dpo

GET

- json
- api

OPTIONS

Get Post Dpo

```
GET /api/v1/dpoapp/

HTTP 200 OK
Allow: GET, POST, OPTIONS
Content-Type: application/json
Vary: Accept

"Please provide end_date, investment_profiles, asset_classes, strategy_rules, historical_prices and kernel in JSON format."
```

Media type: application/json

Content:

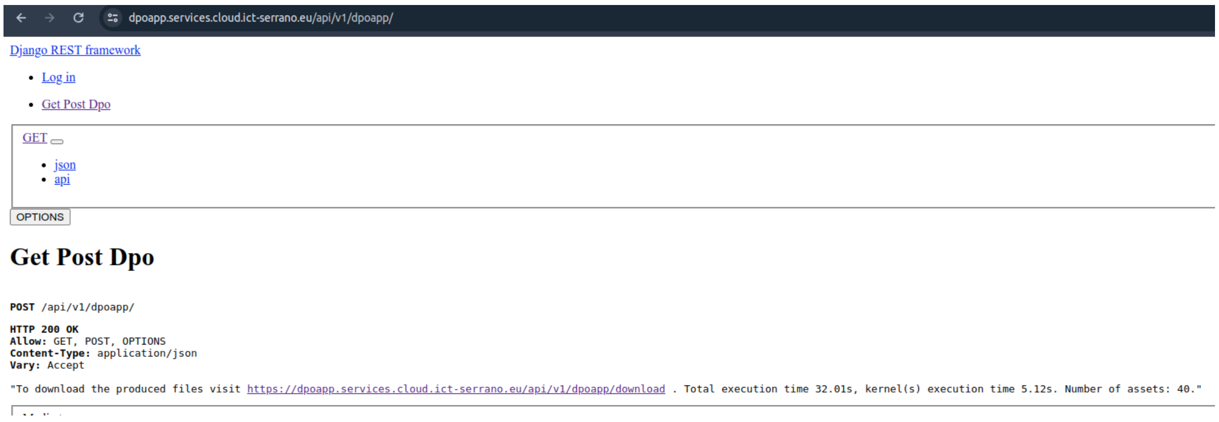
```
{
"end_date": "2024-12-24",
"investment_profiles":InvestmentProfile"
"asset_classes": "AssetClasses",
"strategy_rules": "StrategyRules",
"historical_prices":"40_asset_prices.csv,
"kernel": "wavelet"
}
```

POST

Figure 59: DPO API and input.

Step 3: Post-Execution Information

- Upon completion of Step 2, the system will display a screen providing details on where the generated files can be downloaded, the execution time of the DPO, and the selected kernel's run.


dpoapp.services.cloud.ict-serrano.eu/api/v1/dpoapp/

Django REST framework

- Log in
- Get Post Dpo

GET

- json
- api

OPTIONS

Get Post Dpo

```
POST /api/v1/dpoapp/

HTTP 200 OK
Allow: GET, POST, OPTIONS
Content-Type: application/json
Vary: Accept

"To download the produced files visit https://dpoapp.services.cloud.ict-serrano.eu/api/v1/dpoapp/download . Total execution time 32.01s, kernel(s) execution time 5.12s. Number of assets: 40."
```

Figure 60: Completion of DPO execution, useful information and link to access produced files.

Step 4: Downloading Generated Files

- Download the generated files from the provided URL https://dpoapp.services.cloud.ict-serrano.eu/api/v1/dpoapp/download. The application is now ready to accept new requests.

In the live demo, we will showcase the execution of each of the four SERRANO-accelerated kernels (Wavelet, Savgol, Kalman, Black Scholes) on datasets of different sizes, illustrating the diverse outcomes produced by each kernel.

5.2.2 Optimizing Investment Portfolios: Balancing Risk and Reward

In this evaluation, we aim to showcase the graphical representation of KPIs relevant to business success. The primary tool employed for this demonstration is the Efficient Frontier. This approach not only involves explaining the significance of the Efficient Frontier but also highlights the advantages gained by inbestMe through the integration of SERRANO and DPO technologies.

The Efficient Frontier is a fundamental concept in portfolio theory, illustrating the optimal balance between risk and return. Utilizing this graphical representation allows for a comprehensive analysis of various portfolios, particularly in the context of business success. In the case of inbestMe, the Dynamic Portfolio Optimisation (DPO) technique becomes crucial. DPO facilitates automated portfolio creation and enables the analysis of a vast number of portfolios with multiple assets, a process that was previously handled manually and limited to 240 assets internally at inbestMe.

The fintech use case gains substantial advantages by integrating DPO, the portfolio management processes to the SERRANO:

1. Automated Portfolios Creation: Significantly reduces the need for manual portfolio creation.
    - Enhances efficiency and accuracy in the creation of diverse portfolios.
2. Automated Market & Asset Analysis:
    - Employs DPO to automate market and asset analysis processes.
    - Handles a vast number of assets, potentially exceeding 9500, in contrast to the 825 assets managed manually at inbestMe.
3. Acceleration of Market & Asset Analysis:
    - Outpaces human portfolio managers in the speed of market and asset analysis.
    - Leverages SERRANO to accelerate the overall analysis process.
4. Creation of Multiple Portfolios:

- Facilitates the creation of potentially over 3000 portfolios, surpassing the current manual creation of 240 portfolios at inbestMe.

5. Creation of Portfolios with Reduced Risk and High Returns:

   - DPO, in conjunction with SERRANO, enables the creation of portfolios that strike a balance between risk and return.

The success of this evaluation is gauged through the assessment of several KPIs, as outlined in Section 5.2.2 of Deliverable D6.6 but also in the following Section 5.3. These KPIs include:

- Conversion/Adaption to cloud-based containers.
- Independent instances deployment.
- Rate of market analysis.
- Rate of portfolio analysis.

The accompanying plots visually capture the enhanced portfolio management capabilities achieved by inbestMe, thanks to the integration of SERRANO and Dynamic Portfolio Optimisation (DPO).

Figure 61 depicts the notable advancements in portfolio management achieved by inbestMe through the adoption of SERRANO and Dynamic Portfolio Optimisation (DPO).

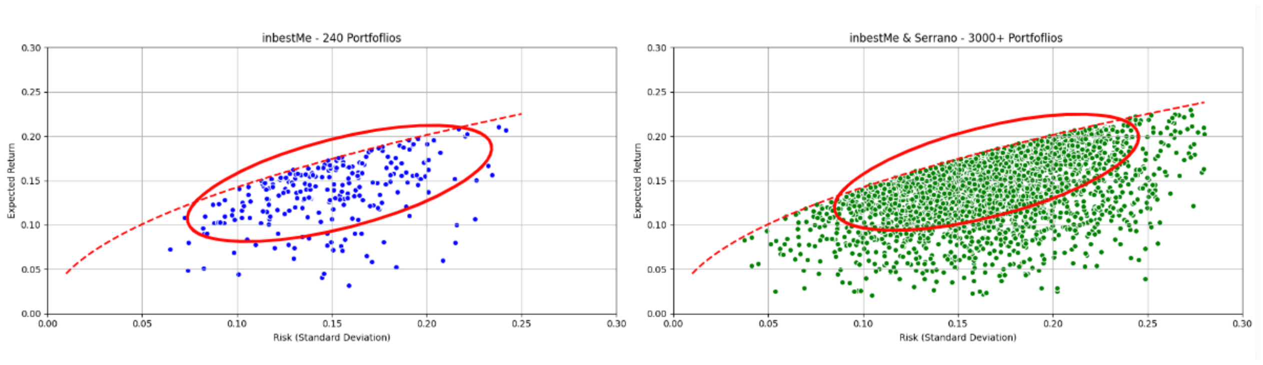


Figure 61: Efficient Frontier: InbestMe Portfolios vs. SERRANO DPO Portfolios

On the left, the plot with blue dots represents InbestMe's current efficient frontier. Each dot is a portfolio, positioned by its risk and expected return.

Moving to the right plot, the green dots illustrate the results of implementing SERRANO's DPO. This comparison clearly shows a significant increase in the number of generated portfolios. More importantly, it highlights DPO's ability to create portfolios that not only have a higher expected return but also maintain the lowest possible risk.

The advantage of using DPO is clear. It allows for the automated creation of multiple portfolios that achieve optimal balance, surpassing the limitations of manual portfolio construction. This efficiency and effectiveness in generating high-performing portfolios advocate for DPO as a transformative tool in portfolio management.

5.2.3 Local vs Accelerated DPO Performance Analysis Across Diverse Datasets

In this evaluation, we have conducted a comprehensive analysis of the DPO performance on local containers and SERRANO's accelerated system. This evaluation focuses on running the fintech application across various datasets of differing sizes, utilizing the components of the SERRANO platform. The primary aim is to evaluate the performance capabilities of the DPO components within SERRANO. Furthermore, this study aims to demonstrate the advantages of utilizing the SERRANO platform for the fintech use cases.

Our evaluation is centred on critical KPIs such as the efficiency of Conversion/Adaptation to Cloud-Based Containers and the efficacy of Independent Instances Deployment. These factors are of significant importance, considering that our experiments are conducted within the SERRANO cloud environment. In addition, we have examined the Rate of Market and Portfolio Analysis to determine the practical effectiveness of these components in real-world fintech applications. Table 10 presents a detailed analysis of the performance outcomes observed when executing DPO applications on 9 datasets, each utilizing the accelerated setup. Each table is structured to include the following columns:

- Size: Dataset Size in MB.
- Kernels execution - Accelerated: Average duration (in seconds) for five runs of accelerated the kernels.
- Kernel Execution - Local: Average duration (in seconds) for five runs of the kernel on local deployment (not accelerated).
- DPO Functionalities: Average duration (in seconds) for five runs of DPO functionalities, excluding the kernels.
- Total Execution - Accelerated: Average total duration (in seconds) for completing five DPO runs when kernels accelerated.
- Total Execution - Local: Average total duration (seconds) for completing five DPO runs on local set up.

These tables offer a comprehensive view of the time efficiency of each system in various aspects of the DPO application execution.

Table 10: Execution Information for DPO execution on Accelerated and Local set up.

Size (MB)	Kernel Execution - Accelerated	Kernel Execution - Local	DPO Functionalities	Total Execution - Accelerated	Total Execution - Local
93	93	395.71	16.29	109	412
102	96	432.19	16.76	112	449
127	112	542.19	16.87	128	559
169	140	712.81	17.48	157	730
212	186	907.53	19.18	205	926
254	197	1072.34	18.98	216	1091
296	243	1267.97	20.52	263	1288
338	289	1430.38	21.11	310	1451

The results from the analysis of DPO performance, based on the data provided, reveal significant insights into the effectiveness of the SERRANO platform's accelerated systems compared to local container execution. The evaluation, conducted across various dataset sizes, demonstrates a consistent trend in performance metrics. There is a pronounced reduction in kernel execution time when utilizing the accelerated systems. The overhead introduced by DPO functionalities remains relatively stable across different dataset sizes, indicating a scalable and efficient integration of these functionalities within the SERRANO framework. The total execution time, encompassing both kernel execution and DPO functionalities, also shows a marked improvement with acceleration.

Overall, the data underscores the performance of the SERRANO accelerated systems over local containers, especially in terms of kernel execution and total processing times. This implies that for fintech applications requiring high computational efficiency, the utilisation of SERRANO's accelerated systems can offer substantial benefits in terms of speed and performance.

5.3 Evaluation

The KPIs of this category were selected based on factors that affect the satisfaction of business and technical requirements. For the business success criteria, first, we want to reduce cloud costs by at least 50% by deploying a hybrid cloud infrastructure. Furthermore, we care to increase the portfolio performance by at least 10%, which will maximise the returns and minimise the risks correlated to improving accuracy in forecasting and market predictions. Regarding the technical success criteria, we want to migrate our applications and services to cloud-native successfully. Additionally, we target to deploy independent cloud-based instances of our system for third parties. It is also essential to rate markets and portfolios through continuous analysis of them. Moreover, technical success will be considered to create real-time orders using live market prices.

Table 11: UC2 business success criteria

ID	KPI	Success criterion	Estimated target value	Result
UC2.1	Percentage of cloud costs reduction	Reduce cloud costs by deploying a hybrid cloud infrastructure	Reduced by 50% or more	We have deployed more than 70% of services and cut more than 50% costs in cloud infrastructure bills.
UC2.2	Percentage of portfolio performance increase	Increase portfolio performance (return/risk)	Increased by 10% or more	10%-15% improvement, the new portfolios are closer to the efficient frontier.
UC2.3	Percentage of improvement	Improve accuracy of forecasting and prediction	Improved by 10% or more	15%-25% improvement, we can run forecasting for more assets and different periods.

Table 12: UC2 technical success criteria

ID	KPI	Success criterion	Estimated target value	Result
UC2.4	Conversion/adaptation to cloud-based containers	Convert/adapt the INB applications and system to cloud-based containers	Conversion/adaptation successful	Porting is successful.
UC2.5	Independent instances deployment	Deploy independent cloud-based instances of the INB system for third parties	Deployment successful	Deployment is successful.
UC2.6. REM	Real-time orders creation	Create real-time orders using live prices	Real-time orders creation successful	We excluded real-time order creating in the demo due to its complexity to run in sandbox environment.
UC2.7	Rate of market analysis	Continuous market analysis	100 financial assets per hour or more	We can analyze more than 1000 assets per hour.
UC2.8	Rate of portfolio analysis	Continuous portfolio analysis	100 portfolios per hour or more	We can analyze more than 1000 assets per hour.

Managing heterogeneous cloud infrastructure is complex. As a result, companies like INB opt to use a single cloud provider to facilitate the management and administration of cloud resources. With SERRANO and **UC2.1,** INB deployed a hybrid cloud infrastructure consisting of local and cloud computing units. UC2.1 measures the number of resources deployed in the cloud and locally as well as the utilisation of the cloud vs. local resources. It aims at having at least 50% of the utilisation be in local compute resources or at least 50% of the resources are local deployments. As a result, this achieves 50% direct cloud cost reduction.

Investment portfolio performance is the actual return of an investment portfolio in a specific period. In **UC2.2**, we measured the return of the portfolios after integrating the DPO application with the SERRANO developments. The evaluation was successful since by executing more often and accurate portfolio management activities we achieved a 10% performance increase.

Similarly, **UC2.3** measures the accuracy of portfolio analysis, predictions, and forecasting. Unlike UC2.2, U2.3 is related to estimating the expected return. Thanks to being able to perform more computation-intensive operations, we improved the accuracy of the predictions.

UC2.4 aims to demonstrate the SERRANO enhancements that facilitate the transparent deployment of INB's applications and services across federated edge/cloud resources. To this end, INB has ported its application to leverage the SERRANO-provided lightweight virtualisation mechanisms. UC2.4 was evaluated by completing the porting of the applications. Similarly, **UC2.5** was evaluated by successfully demonstrating the deployment of multiple instances of INB applications and services across the heterogeneous edge, cloud, and HPC resources that are unified by the SERRANO platform.

UC2.6 has been removed due to infrastructure limitations, as it is practically unfeasible to set up a demo and demonstrate the creation of real-time orders based on live prices. Specifically, obtaining real-time live prices requires a specific connection to market-data providers, and executing real-time orders also requires a specific connection to exchanges. Therefore, setting up such infrastructure in a test environment would be unfeasible.

UC2.7 and **UC2.8** are measured similarly. Specifically, in UC2.7, we measured how many investment instruments were analysed, and in UC2.8, measured how many portfolios were analysed and managed.

6 Anomaly Detection in Manufacturing Settings Use Case

Downtime of failed devices in an industrial plant must be kept to a minimum to achieve high system availability. In the very competitive manufacturing world, getting the most out of the machine may be the difference between being competitive or not. Thus, and mainly after the irruption of the Industry 4.0 paradigm, many techniques and methods are being widely applied to meet a simple yet complex goal: keep the machine working most of the time.

Companies that manufacture expensive high added-value parts are very demanding regarding machine availability and quality assurance. As a result, predictive maintenance, remaining lifetime assessment, and diagnosis of critical machine elements are state-of-the-art practices. However, some techniques that are used require the machine to stop before performing the analysis.

The use case leverages the SERRANO platform to change the state-of-the-art approach and to perform machine component status assessment without stopping the machine. This UC aims to develop a system that is able to detect anomalies by processing the amount of data generated in near real-time by high-frequency sensors.

6.1 Use case description

This UC proposes a deployment approach where data analysis is performed continuously while the hardware equipment keeps running most of the time, and the state of the various independent components, along with the overall status, is continuously reported. Moreover, the UC is focused on a single critical component, ball screws. A ball screw is a mechanical linear actuator that transfers rotational motion to linear motion, moving the machine on the x and y-axis. Ball screws are expensive and critical machine components whose breakage implies stopping the machine for a significant time. Each machine has two ball screws, x and y axis.

The UC has developed a Data Processing Application to analyse real-time signals from the ball-screw sensors and check for anomalies, detecting anomalous behaviours that may affect the part quality, and predict imminent failures. This application has been divided into two different services that analyse the data coming from the position sensors and the data from the acceleration sensors of the ball screw.

- *Position Processor Service*: Classifies the difference between the expected and the actual position during a time interval as normal or anomalous. The system adapts to the expected degradation of the component during its useful life. Data is gathered from position sensors (linear and angular).
- *Acceleration Processor service*: Classifies the vibration signal as normal or anomalous. The system adapts to the expected degradation of the component during its useful life. Data is gathered from acceleration sensors (vibration data).

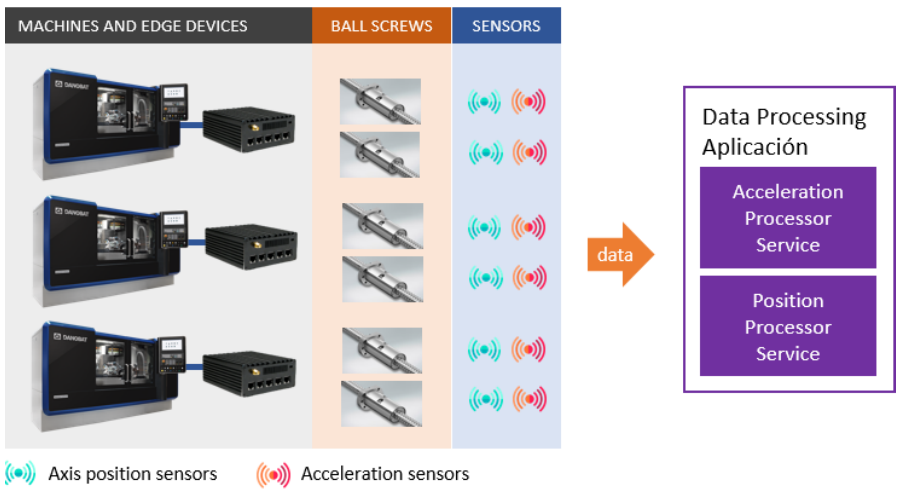

MACHINES AND EDGE DEVICES
BALL SCREWS
SENSORS
data
Data Processing Aplicación
Acceleration Processor Service
Position Processor Service
Axis position sensors
Acceleration sensors

Figure 62: Data Processing Application to analyses real-time signals from ball-screw sensors.

In addition, to obtain data from real machines at IDEKO's facilities, a test bench has been built with two sensorised ball screws (X and Y axis), simulating data from machines in a real-production scenario. The generated data is sent to SERRANO to be analysed in order to detect anomalies through the applications/services (Data Processing Application).

The anomaly detection services created by IDEKO have been implemented in container-based applications on the SERRANO platform via the Alien4Cloud platform. Specifically, this deployment utilises the *SERRANO Orchestrator* plugin, which is designed to interpret the TOSCA language and seamlessly interact with the SERRANO framework.

The streaming data integration with the SERRANO platform is done through the Data Broker component. *Data Broker* provides an interface based on the MQTT protocol to facilitate the publication and consumption of the data generated from the simulated machines' ball screws to use case applications/services and other SERRANO components.

Specifically, the developed anomaly detection services leverage the *SERRANO SDK* to facilitate their seamless access to the SERRANO accelerated kernels. The SDK abstracts the integration with the SERRANO hardware *acceleration mechanisms* in edge/cloud and HPC. These SERRANO developments provide better performance and optimisation of the highly computationally intensive kernels used (e.g., DTW, KMeans, KNN, or FTT) by the Model Inference and Classifier Training services. Moreover, the *S3-compatible Secure Storage* interface is used to store the last N streaming data received through the Data Broker. This way, the required data is stored and accessible by all SERRANO components and the use case services.

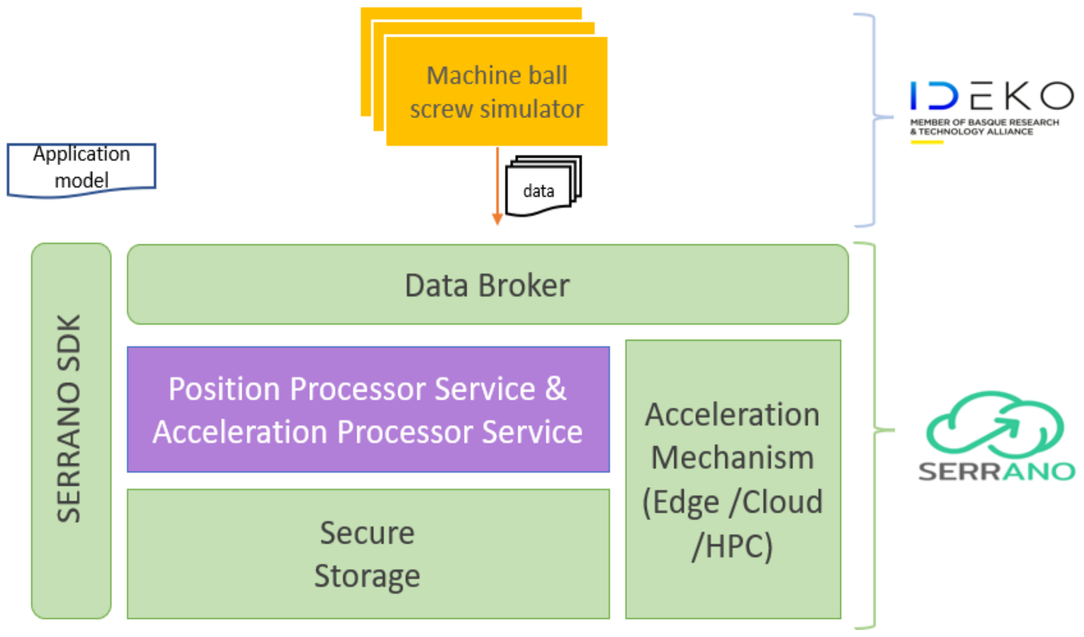

Machine ball screw simulator
data
Application model
IDEKO
MEMBER OF BASQUE RESEARCH & TECHNOLOGY ALLIANCE
Data Broker
SERRANO SDK
Position Processor Service & Acceleration Processor Service
Secure Storage
Acceleration Mechanism (Edge /Cloud /HPC)
SERRANO

Figure 63: Interactions between the use case developed services and core components of the SERRANO platform.

The idea is to reduce the classifier training time and the needed time to make a new prediction through the streaming data. This enables the early detection of possible imminent failures of the ball screw, eliminating also their occurrence and providing greater control of the health status of the ball screw in real time. The SERRANO platform is needed since the current techniques and resources available at the edge cannot support the above operations.

6.2 Demo

The demonstrations associated with this use case has been categorised into two different sections:

- D1. Application deployment into SERRANO platform using Alien4Cloud
- D2. Acceleration Mechanisms (Service Orchestration)

6.2.1 D1 - Application deployment into SERRANO platform using Alien4Cloud

Description: This demo demonstrates how a data processing application for anomaly detection in machine tool equipment can be deployed into the SERRANO platform. This demonstration evaluates the capability of SERRANO's mechanisms to deploy the services in different locations and resources. The services developed by IDEKO for anomaly detection have been deployed in container-based applications on the SERRANO platform through the Alien4Cloud platform, more specifically using the SERRANO Orchestrator plugin developed to interpret the TOSCA language and interact with the SERRANO framework.

Main SERRANO services and components involved in the demo:

- ARDIA Framework, AI-enhanced Service Orchestrator
- Resource Orchestrator, Orchestration Drivers
- Data Broker

KPIs measured/evaluated:

- *GEN.1 / GEN.2:* The UC demonstrates the capability of the SERRANO framework to provide transparent deployment of applications across federated and heterogeneous infrastructures, leveraging the high-level requirements description of the application.
- *GEN.3:* This demonstrator covers one of the three use cases.
- *SRV.1:* The UC is described in the terms the ARDIA model defines.

Scenario Description: The demonstration first introduces the UC and the SERRANO infrastructure. Then, the demonstration compiles through the Alien4Cloud the application requirements into the Application Descriptor File and the application deployment YAML that defines the components to be deployed, among other important deployment-related configuration parameters (as presented in section 3.1 Demo-1).

The actual deployment of the UC services takes places the Alien4Cloud platform (Figure 64), while the application status and performance (e.g., inference execution times) are illustrated in a custom Grafana dashboard (Figure 65).

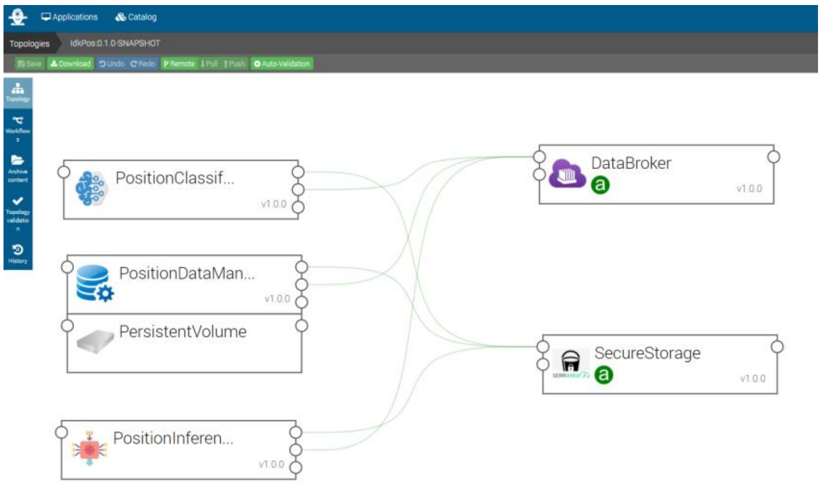
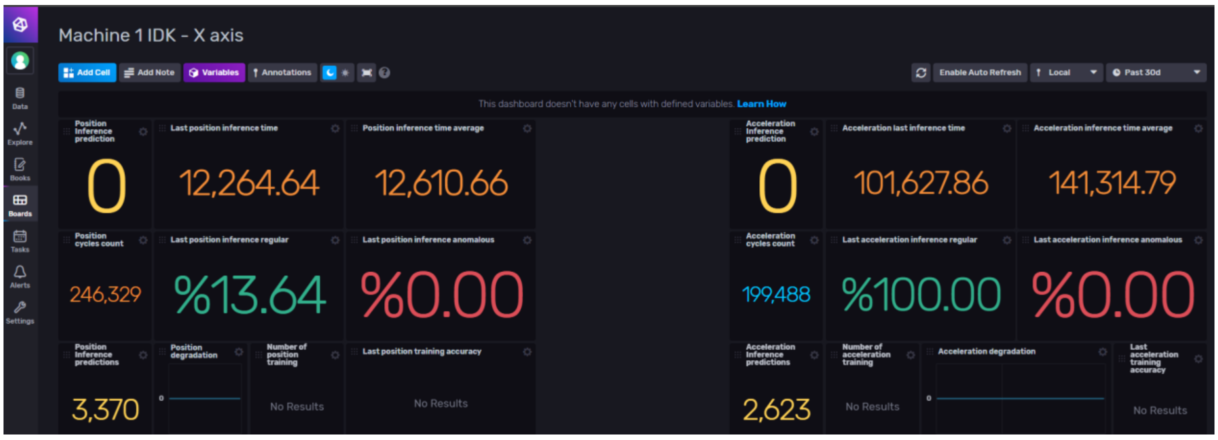


Figure 64: Alien4Cloud user interface.



Machine 1 IDK - X axis

This dashboard doesn't have any cells with defined variables. Learn How

Position Inference prediction: 0
Last position inference time: 12,264.64
Position inference time average: 12,610.66
Acceleration Inference prediction: 0
Acceleration last inference time: 101,627.86
Acceleration inference time average: 141,314.79
Position cycles count: 246,329
Last position inference regular: %13.64
Last position inference anomalous: %0.00
Acceleration cycles count: 199,488
Last acceleration inference regular: %100.00
Last acceleration inference anomalous: %0.00
Position Inference predictions: 3,370
Position degradation: 0
Number of position training: No Results
Last position training accuracy: No Results
Acceleration Inference predictions: 2,623
Number of acceleration training: No Results
Acceleration degradation: 0
Last acceleration training accuracy: No Results

Figure 65: Grafana dashboard monitoring the inference execution times.

6.2.2 D2. Acceleration Mechanisms

Description: This demonstrator uses the application deployed in the previous demo (D1) to demonstrate the use of the SERRANO SDK to transparently request the on-demand deployment of SERRANO-accelerated kernels (e.g., KNN). By leveraging these SERRANO developments, the use case applications increase their performance and throughput by keep up with the increasing demand for fast and efficient processing of the collected data.

Main SERRANO services and components involved in the demo:

- Resource Orchestrator and Orchestration Drivers
- SERRANO Telemetry Framework
- Resource Optimization Toolkit
- vAccel and lightweight virtualization mechanisms
- Library of SERRANO-accelerated kernels

KPIs measured/evaluated:

- *GEN.4 / INT.6:* The use case will make use of the SDK for any kind of interaction with the platform.
- *ACC.4:* The use case integrates accelerated kernels.
- *RES.3:* The demonstrator will show how the Orchestrator places containers based on the telemetry measured on every instant.
- *SRV.7:* The demonstrator will cover partial or complete execution of components in both edge and cloud.

Scenario Description: The focus is on highlighting the capabilities of the use case applications to efficiently handle a large number of inference requests within a specific time window by overcoming the constraints posed by the resource-limited edge devices. To this end, the microservices rely on transparently executing the hardware-accelerated kernels across the SERRANO platform that help address the scalability challenges. The scenario assumes a ball screw nearing its end of life (EOL), demanding enhanced precision on the health status assessment compared to a brand new one. The available resources include computational, storage, and acceleration devices across edge/cloud and HPC platforms (Figure 12) that are unified by the SERRANO platform services.

The demonstration starts displaying the current status of the application, through:

- the Grafana dashboard that illustrate how inferences are being executed
- the SERRANO logs that illustrate where are the kernels being executed

After this, a modification of the initial scenario is deliberately introduced to simulate the case when the ball screw is reaching its end of life. This will cause the application requirements to no longer be met. Hence, the SERRANO orchestration and deployment mechanisms will automatically offload the corresponding workloads to the most appropriate hardware acceleration resources to use the benefits of the SERRANO-accelerated kernels.

During the demonstration, there are available details from the operation of the SERRANO mechanisms that show the framework's decisions to keep requirements in place. Moreover orchestration, the use case Grafana dashboard illustrates how the inference time increases after the scenario modification and how returns to a regular state after SERRANO framework decisions.

6.3 Evaluation

The current state-of-the-art techniques in the detection of anomalies in critical components such as the ball screw are based on stopping the machine and executing controlled and measured movements, comparing their performance with previous measurements. This approach however, leads to obtaining low knowledge of the machine's component health and decreasing machine availability.

The success of this use case is based on leveraging the SERRANO platform to go beyond the state-of-the-art approach and to perform the ball-screws' status assessment without stopping the machine, moving to real time health assessment. As a result, a deeper knowledge of the components health is achieved, while and machine availability is increased.

The following is a summary and review of KPIs for the Business Success Criteria.

Table 13: UC3 business success criteria.

ID	KPI	Success criterion	Estimated target value	Result
UC3.1	Transition to real-time anomaly detection	Transition from on-demand to real time data analysis for anomaly detection to reduce machine stoppages	Transition successful	SERRANO's mechanisms allow our UC to: • Timely process streaming data leveraging SERRANO's kernel acceleration mechanisms. • Continuously infer the status of balls crews every 12 seconds using data coming from accelerometers. • Continuously infer the status of balls screws every 10 seconds using data coming from position sensors for ML model trained with 110 observations.
UC3.2	Anticipation of failures	Anticipate failures comparing to current state-of-the-art techniques	Anticipation successful	SERRANO's mechanisms allow our UC to: • Perform continuous components health assessment. • Predict imminent failures with enough time to avoid severe machine failures.
UC3.3	Anomaly detection accuracy increase	Increase of anomaly detection accuracy (avoiding nuisance alerts and false positives/negatives)	Increase by 35% or more	SERRANO's mechanisms allow our UC to: • Reduce nuisance alerts or false positives. • Improve classifier accuracy by 10%.

UC3.1 Transition to real-time anomaly detection

The transition from on-demand component diagnosis to online diagnosis involves the installation of new hardware elements, the development of new software systems that dynamically identify optimal execution conditions, processing systems, monitoring dashboards, and, above all, it implies the need to process a volume of data several orders of magnitude higher. The latter is the cornerstone on which to build the rest of the solution, and it is what SERRANO has managed to develop.

Our use case has successfully simulated a real-time component diagnosis scenario by leveraging SERRANO's mechanisms. Our simulated scenario has been able to:

- Timely process streaming data leveraging SERRANO’s kernel acceleration mechanisms.
- Continuously infer the status of balls crews every 12 seconds using data coming from accelerometers.
- Continuously infer the status of balls crews every 10 seconds using data coming from position sensors.

Figure 66 shows results obtained from on-demand component diagnosis performed in the ball screw test bench at IDEKO facilities using a cycle frequency of a week, which is usually the manufacturing client’s most used periodicity for running components’ diagnosis. The chart displays 52 analyses carried out during 2021 year and the anomaly score obtained for each of them.

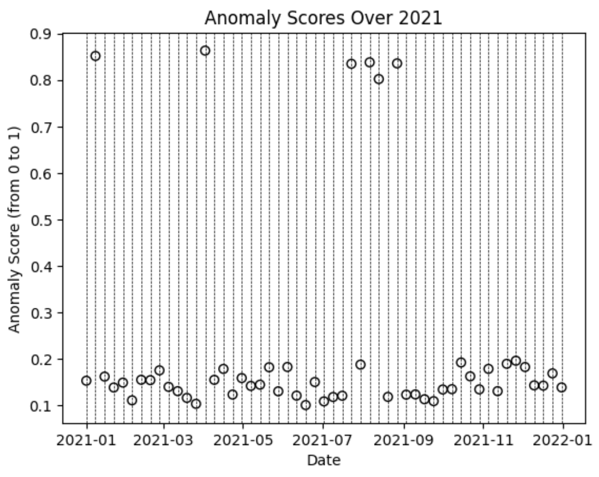


Figure 66: Summary of 2021 results.

Figure 67 shows the results obtained for a single month of the same year. Five single analyses were performed during January 2021.

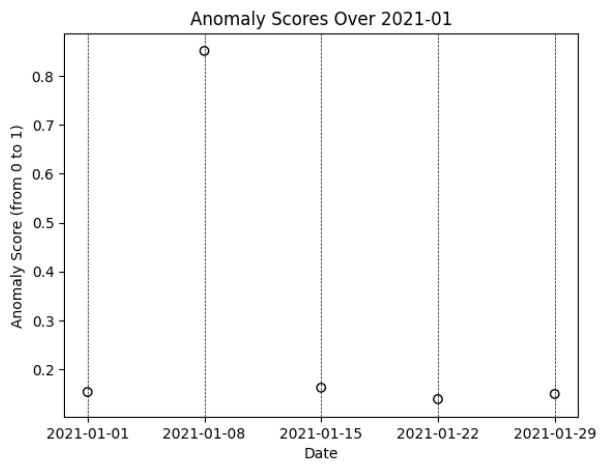


Figure 67: Results of January 2021.

For comparison, the next chart displays some results of the continuous diagnosis strategy running over the SERRANO framework. Using this strategy, the component health is computed every 10 seconds. Figure 68 shows some results from a single day, 2023-04-21, obtained by our Position Service, which was developed during the project. As the chart displays, in a matter of 20 minutes the number of component diagnosis inferences is greater than the inferences for a whole year using the classical approach.

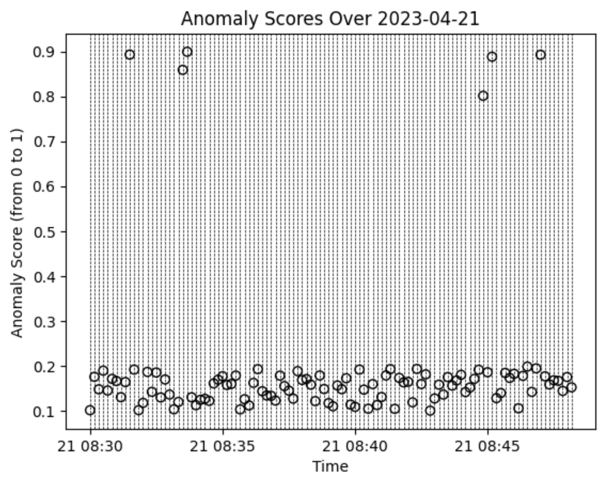


Figure 68: Some results from 2023-04-21.

Taking as a basis for this evaluation the Position Service of this use case, the next charts illustrate the results obtained for the streaming data requirements along with the ML models' complexity. In particular, Figure 69 compares the time needed to process streaming data by our Position Service. The x-axis indicates the number of observations used to train the ML model, while the y-axis indicates the time needed to infer a new observation using each ML model. The colour series indicates the time using different acceleration mechanisms in different infrastructures within the SERRANO platform. As depicted in Figure 12, the final release of the SERRANO platform incorporates two types of hardware acceleration resources (i.e., GPU and FPGA) in two of the Kubernetes clusters. The NBFC K8s cluster includes 3 GPUs

(Jetson AGX Xavier, Jetson Nano, Nvidia RTX 2060) and the AUTH K8s cluster includes 2 GPUs (Nvidia T4) and 2 FPGAs (Xilinx Alveo, Xilinx MPSoC).

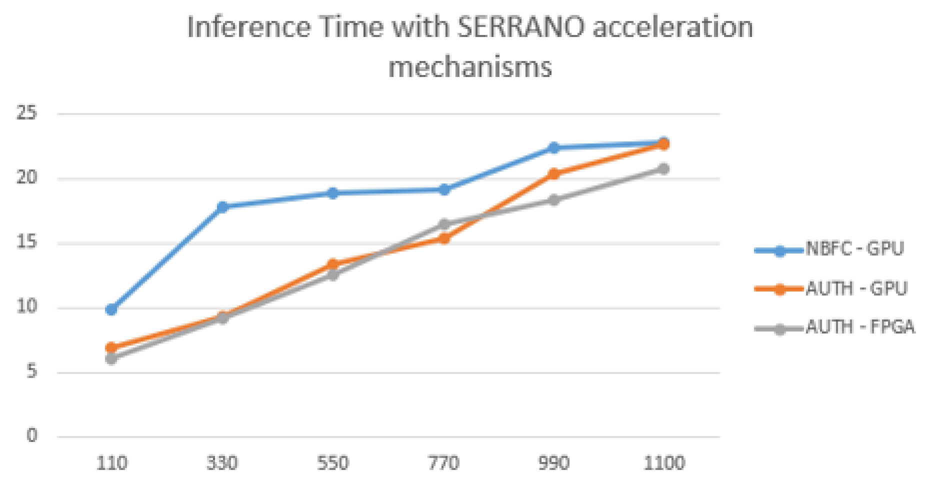


Figure 69: Inference time with SERRANO acceleration mechanisms for different range of ML models (Y axis - prediction time in seconds and X axis - observations of the generated model).

Figure 69 perfectly illustrates the challenges of the use case. While the systems target is making inferences every 10-12 seconds, this is achieved using an ML model trained using 110 observations. The inference time increases as the number of observations used to train the model increases.

Figure 70 shows the inference time results when the available infrastructure also utilizes the IDEKO's edge devices. This illustrates the extreme differences in processing time between the execution of the ML model on a commodity, regular, hardware at the edge, and the SERRANO powered acceleration mechanisms and infrastructure.

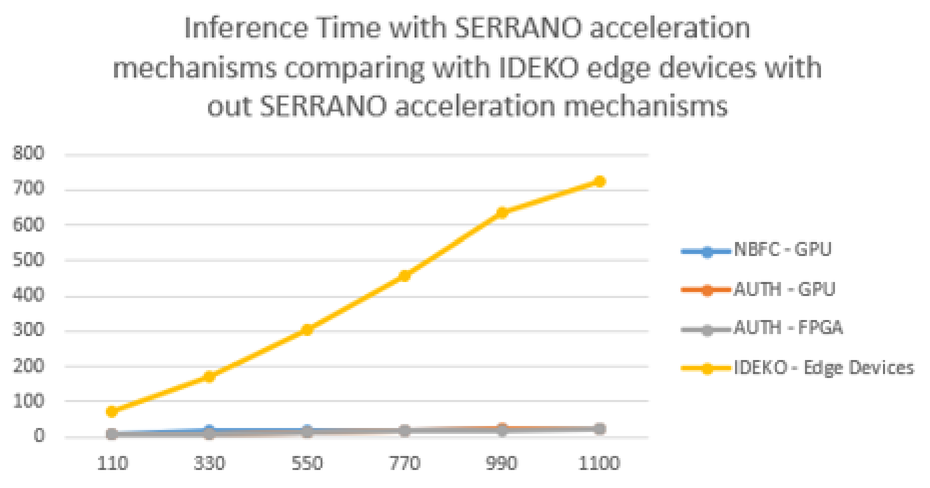


Figure 70: Inference time with SERRANO acceleration mechanisms and IDEKO's edge device (Y axis - prediction time in seconds and X axis - observations of the generated model).

Additionally, the capacity of the SERRANO platform's to facilitate classifier retraining for the acceleration service for batches of machine big data has been evaluated. Throughout the lifespan of the ball screw, the initial conditions undergo changes. Consequently, the initial classifier model must also be adjusted, adapting to the new conditions of the ball screw. Hence, the importance of retraining the model, allowing it to adapt to the evolving situations of the ball screw. To this end, intentionally the classifier retraining process using various sets of machine data (simulated data) has been initiated, mimicking a scenario where 80% of recent inferences were deemed anomalous. Consequently, the acceleration service detected changes in the ball screw's conditions, signalling the necessity for a model update. Subsequently, the SERRANO orchestration mechanisms dynamically offloaded the execution of the required computations to the HPC platform due to the big data. In this case, the SERRANO-accelerated version of the KMEANS kernel for the HPC platform was employed. The following table summarizes the end-to-end execution times for the on-demand execution of the KMEANS kernel. This strategic adjustment is crucial since the local resources within the use case infrastructure are insufficient to provide the updated model version for these dataset sizes.

Table 14: Execution times for the SERRANO-accelerated KMEANS kernel (acceleration services) for classifier retraining model on HPC with different batches of machine data

Dataset	Total datapoints	Move Data to HPC (sec)	Kernel Execution (sec)	Move Data from HPC (sec)	Total Execution Time (sec)
cycle_26	852020	11	483	10	504
cycle_104	3408080	12	396	17	425
cycle_156	5538130	23	330	10	363
cycle_208	6816160	12	462	11	485
cycle_234	7668180	8	407	10	425
cycle_260	8520200	13	406	8	427

The evaluation results indicate that the SERRANO platform, equipped with acceleration, orchestration, and security mechanisms developed in the project context can address the challenge of processing the data volume from a common setup of self-diagnostic cycles for critical components in the machine tool sector.

UC3.2 Anticipation of failures

Anticipation of failures is a side effect of the transition mentioned in the previous success criteria. Current state-of-the-art techniques perform periodical (e.g. weekly) analysis for predicting and detecting anomalies due to the need to stop the machine to execute diagnosis tasks. This approach can oversee failures happening between running two diagnoses due to extremely fast component degradations. The usage of SERRANO's developments allows our use case to:

- Perform continuous components health assessment
- Predict imminent failures with enough time to avoid severe machine failures

At the end of January 2022 (bold vertical line in the bellow chart), the bearing package supporting the Y-axis balls crew of IDEKO's test bench broke. The anomaly score for the previous weeks, computed using the classic weekly analysis strategy, was insufficient to foresee the incidence (see Figure 71). The classic strategy was not able to predict it because the origin of the failure was a wrong ball screw preload after an erroneous maintenance task, days after the execution of the last diagnosis test.

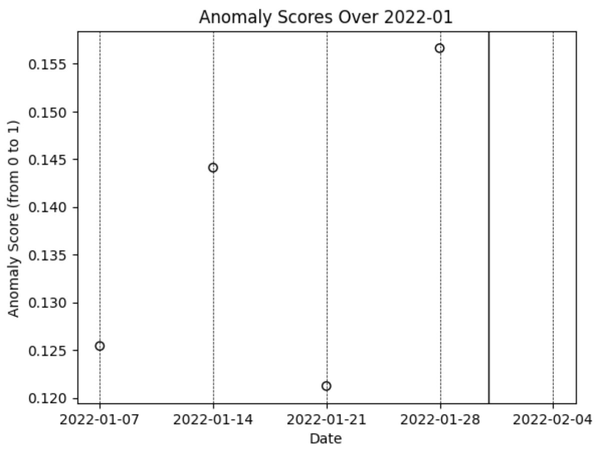


Figure 71: Results from 2022-01, where a vertical line locates the breakage.

Our team used those days' data to feed the continuous anomaly detection system developed during the project. Experimental tests suggest that the continuous strategy would have successfully predicted the failure with enough time to, at least, avoid the damage that a sudden breakage can cause. Results from these tests are illustrated in Figure 72, where, after some filtering, an increasing pattern in the anomaly detection score is clearly identified.

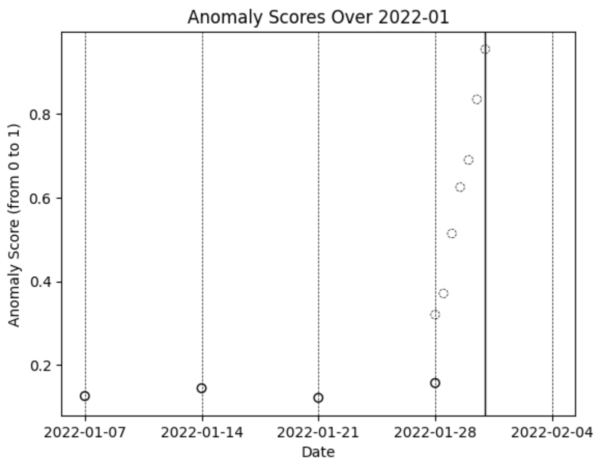


Figure 72: Hypothetical results utilizing SERRANO platform.

Experiments using real breakage data suggest, as also experience do, that a continuous monitoring system allows one to anticipate the failures, especially for components that fail suddenly, without a prior clear pattern of degradation. These situations are often caused by a human error during a maintenance task, among other root causes. The data volume involved in this task is, however, the main difficulty to tackle. SERRANO framework, as described before, has demonstrated the ability to process these data effectively, enabling the implementation of this anomaly detection system on a manufacturing plant.

UC3.3 Anomaly detection accuracy increase

In order to evaluate these success criteria IDEKO created a scenario that compares the two component health assessment strategies: (i) current state-of-the-art periodical machine self-diagnosis cycles and (ii) continuous self-diagnosis using SERRANO powered framework. The following sections give important details of the scenario setup to better understand the context and the results.

Machine

The scenario was set up in IDEKO's ball screw test bed. As described in previous deliverables, the test bed comprises two ball screws working 24/7 to speed up degradation and enables the execution of both on-demand and continuous tests.

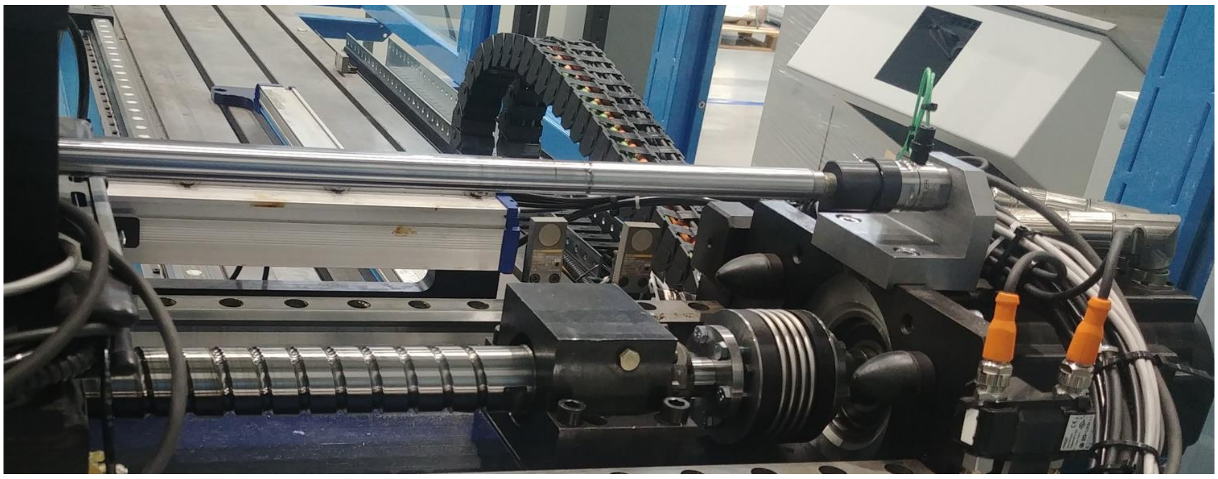


Figure 73: Test bed ball screw.

Hardware setup

The picture below gives an overview of both hardware and software systems involved in the scenario setup.

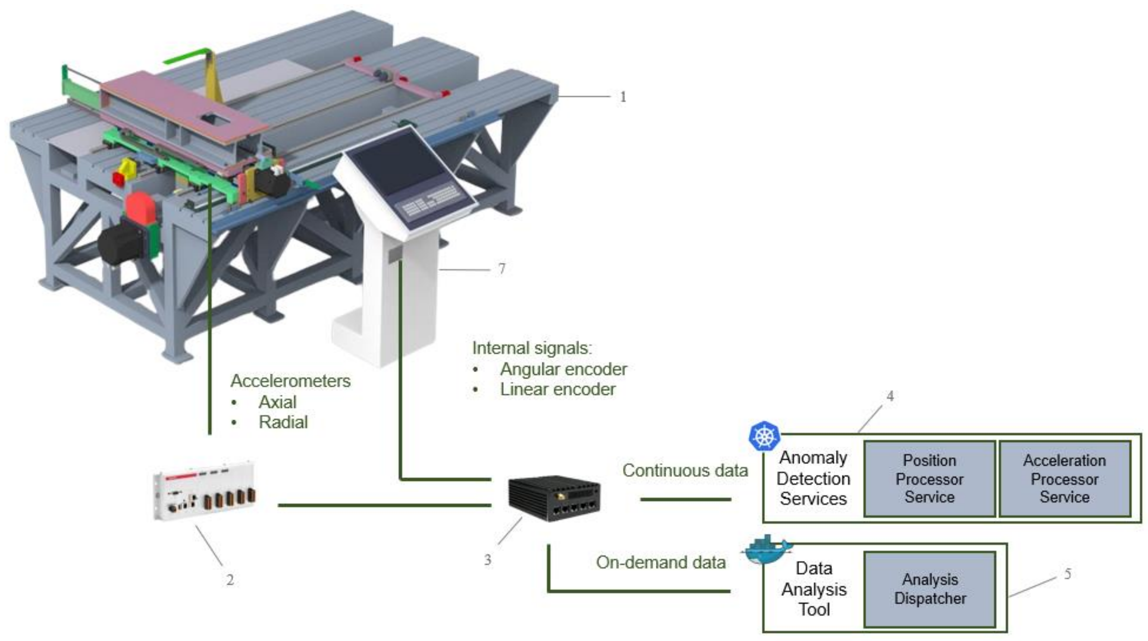


Figure 74: Different anomaly detection set up on demand vs continuous data.

The test bench (1) accelerometers are connected to a high frequency data acquisition system (2). Ball screws' position signals are gathered directly from the Siemens Sinumerik CNC (7). Both acceleration and position signals are fed into our IoT Gateway (3), where the tests run. The IoT Gateway runs the necessary software components for running both classical and periodical tests (5) as well as continuous tests (4).

<u>Scenario execution conditions</u>

The test bench has been working for 10 days, during which the two health assessment strategies were executed repeatedly:

1. Periodical machine self-diagnosis cycles have been executed every 24 hours.
2. Acceleration and Position services have been running and executing component health assessment every few seconds.

During the test period, 7 cycles were executed in total using the first strategy and 77761 cycles were executed using the second one. After the test period executions data were compiled and analysed and the results are detailed below.

<u>Results</u>

The first strategy incurs on a data-resolution problem. Distinguishing execution with anomalies from those that are normal, can became a problem. This is mainly because of the lack of resolution in the data offered by the analysis using this strategy, which incurs to a potential loss of information due to the lack of data for the health of the component in the time period between one to another analysis.

Figure 75 shows a scenario (one analysis per day) where it is difficult to distinguish anomalies. The dispersion of the data makes the system, which is forced to group executions into only two groups (anomalous and normal values), tend to fail, often offering false positives.

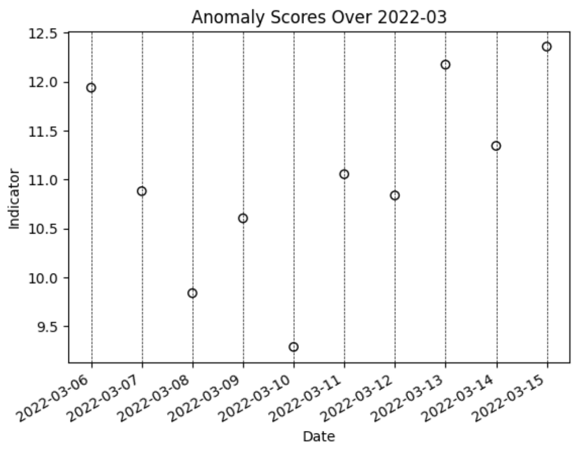


Figure 75: No anomalies are captured.

In contrast, Figure 76 shows a scenario where the second strategy is applied, using the SERRANO technological developments. In this case, the health of the component is computed every 10 seconds. Results clearly indicate that the anomaly detection system produces more robust results, avoiding a large number of nuisance alerts.

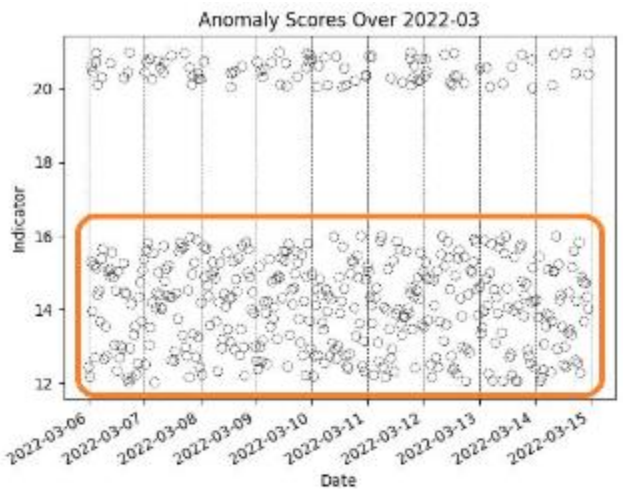
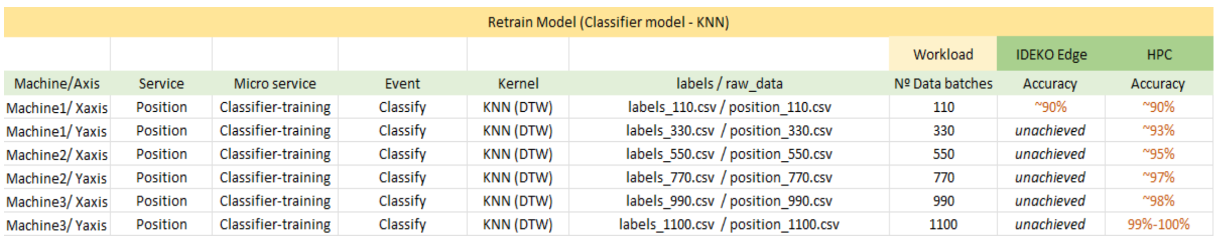


Figure 76: Anomalies are captured using SERRANO platform.

NOTE: the number of points within the orange stripe has been reduced for the sake of clarity. That stripe is comprised of about 77700 points.

Although it is not possible to estimate the accuracy of the detection system at full accuracy, the above graph shows a significant change in data resolution, which allows the clustering algorithm to perform much more accurate division of anomalous observations than the traditional system.

Furthermore, during the project, several benchmarking tasks were executed to compare the accuracy of the model training tasks using different acceleration strategies. The following image (Figure 76) gives an overview of the results of executing a model training task for the KNN with different input data sizes and the accuracy obtained for each of them. The comparison is done by executing the training task on IDEKO's edge devices and comparing it with the HPC approach. Using IDEKO's edge device, the training was only successful with the minimum data size used for the benchmarking. Comparing the accuracy between this and the biggest trained model in the HPC infrastructure, the accuracy has improved by, approximately, 10%.



Retrain Model (Classifier model - KNN)								
						Workload	IDEKO Edge	HPC
Machine/Axis	Service	Micro service	Event	Kernel	labels / raw_data	Nº Data batches	Accuracy	Accuracy
Machine1/ Xaxis	Position	Classifier-training	Classify	KNN (DTW)	labels_110.csv / position_110.csv	110	~90%	~90%
Machine1/ Yaxis	Position	Classifier-training	Classify	KNN (DTW)	labels_330.csv / position_330.csv	330	*unachieved*	~93%
Machine2/ Xaxis	Position	Classifier-training	Classify	KNN (DTW)	labels_550.csv / position_550.csv	550	*unachieved*	~95%
Machine2/ Yaxis	Position	Classifier-training	Classify	KNN (DTW)	labels_770.csv / position_770.csv	770	*unachieved*	~97%
Machine3/ Xaxis	Position	Classifier-training	Classify	KNN (DTW)	labels_990.csv / position_990.csv	990	*unachieved*	~98%
Machine3/ Yaxis	Position	Classifier-training	Classify	KNN (DTW)	labels_1100.csv / position_1100.csv	1100	*unachieved*	99%-100%

Figure 77: Benchmarking in position service comparing the accuracy of the model training.

The accuracy values presented in the preceding table were derived through an evaluation process. First, the data used to build the classifier model underwent labelling, distinguishing them as anomalous or non-anomalous using the KMEANS clustering method. Subsequently, a robust validation technique, K-fold cross-validation, was employed, involving the division of the dataset into five distinct sets. The model's predictions were then compared against these labelled datasets. This approach ensured a rigorous assessment by iteratively training and validating the model on different subsets of the data.

The following table provides a summary and review of KPIs that indicate the technical success of UC3.

Table 15: UC3 technical success criteria.

ID	KPI	Success criterion	Estimated target value	Result
UC3.4	Rate of streaming data processing	Being able to quickly process large amounts of streaming data	20MB/s or more	4MB/s processed with both acceleration as position services.
UC3.5	Increased availability of machine	Increase of machine availability	2% or more, measured in a monthly basis	2.57% improvement on the test bench.

UC3.4 Rate of streaming data processing

The rate of the streaming data processing has been measured with the current use case scenario setup, which, in summary is comprised of:

- 3 machines with 2 ball screws each = 6 ball screws
- 2 accelerometers for per ball screw
- 2 position sensors (linear encoders) per ball screw

The data volume generated by the above setup is summarized as follows:

Sensor	Volume (MB/s)	Total volume 6 ball screws (MB/s)
2x Accelerometer	1.86 * 2 = 3.72	22.32
2x Position	0.124 * 2 = 0.248	1.488
TOTAL	3,968 MB/s	23.808 MB/s

The image (Figure 78) below describes the setup of the scenario.

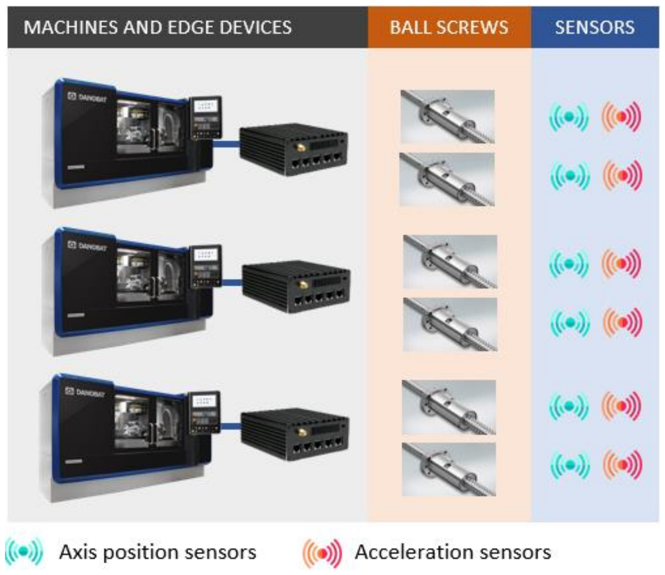


Figure 78: Architecture of the scenario setup.

In summary, the current UC scenario setup produces about 24MB/s when both developed service applications (i.e., Position Service and Acceleration Service) run in parallel. In this situation, the applications run smoothly leveraging SERRANO's platform internal mechanisms.

UC3.5 Increased availability of machine

This KPI is a side effect of the KPIs above. KPI 3.2 clearly describes a real use case where machine availability can be dramatically increased, however, having real historical data for a larger period of time and access to actual customer maintenance incidents registries would make our case even stronger.

Taking this into account, the measurement of this KPI has been based on testbed data for the entire period for which historical data is available. For this purpose, the data from the historical component has been crossed with the recorded incidents. For evaluating machine availability improvements, incidents close to the moment of detection of an anomaly are considered that would have been avoided.

Taken this into account, this yields an improvement of 2.57% in machine availability. Detailed results are displayed in the table below.

Table 16: Effective availability of the test bench compared to a simulated scenario with continuous monitoring.

Source	Working hours	Breakdown hours
Effective	1663	216
Simulated	1707	172
Improvement	2.57%	

Our simulated scenario yields a positive conclusion on the effect on machine availability of a scenario where the health of machine components is measured continuously.

7 SERRANO Project KPIs

7.1 KPIs related to general project requirements

The following set of KPIs are related to Objective 1 “Define an intent-driven paradigm of federated infrastructures consisting of edge, cloud, and HPC resources” and Objective 6 “Demonstrate the capabilities of the secure, disaggregated, and accelerated SERRANO platform in supporting highly demanding, dynamic and safety-critical applications.” of the SERRANO project. They correspond to the general vision of the project to enable the cognitive and transparent application deployment over the federated edge, cloud, and HPC infrastructures.

Table 17: KPIs related to general project requirements

ID	KPI	Description/Innovation	Estimated target value	Result
GEN.1	Unification of edge, cloud and HPC platforms.	SERRANO should unify federated infrastructures, with edge, cloud and HPC resources, through the provision of novel automation and orchestration mechanisms.	Transparent deployment of workloads in all unified platforms.	The final release of the SERRANO platform (D6.7) successfully unifies edge, cloud, and HPC platforms by offering among the others the transparent deployment of workloads, on-demand execution of SERRANO-accelerated kernels, and intent-driven creation of secure storage policies. These functionalities were successfully tested and evaluated through all the platform and use cases demonstrations that described in Sections 3, 4, 5 and 6.

GEN.2	Abstractions for interoperable and infrastructure agnostic deployments.	Applications have to express their high-level requirements in an infrastructure agnostic manner.	SERRANO mechanisms should translate the high-level requirements to infrastructure specific parameters.	The SERRANO orchestration mechanisms (i.e., AI-enhanced Service Orchestrator, Resource Orchestrator, and Resource Optimisation Toolkit) successfully provided the required functionalities. These operations were extensively tested and evaluated through the four platform demonstrations and by the implementation and evaluation of the three project use cases.
GEN.3	Applications successfully demonstrating SERRANO capabilities.	Successful demonstration of SERRANO platform capabilities covering the individual UC requirements and the specific metrics for security, performance, interoperability, and usability.	3 use cases.	The project uses cases and the platform demonstrations were carefully selected and designed to cover and evaluate all the project use cases. Their description provides references to the project KPIs with special emphasis on security, performance, interoperability, and usability.

GEN.4	Availability of open and well-defined interfaces	SERRANO should provide a complete Service Development Kit (SDK) to support effectively the creation, orchestration, deployment, monitoring and adaptation of novel applications.	The three project UCs should utilise the provided interfaces to interact with the SERRANO platform.	The three project use cases and all the platform demos utilise the SERRANO SDK to interact with the SERRANO platform services. More details are available in the respective sections in this deliverable and deliverable D6.7 (M36)

These high-level KPIs were successfully tested, featured, and verified as part of the platform and use case demonstrators that were performed during the project's final phase (M30-M36). GEN.1 and GEN.2 are related to evaluating the SERRANO orchestration, telemetry, and deployment mechanisms. GEN.1 was assessed through the evaluation scenarios for the platform demonstrators and the three use cases, where the developed mechanisms transparently from the end users selected the most appropriate resources from the edge, cloud, and HPC platforms. GEN.1 was also successfully evaluated for the HPC platforms as the respective SERRANO-accelerated kernels were seamlessly executed and managed in the available HPC infrastructure through the SERRANO orchestration mechanisms and SERRANO HPC Gateway. GEN.2 was verified based on the ability of the AI-Enhanced Service Orchestrator and Resource Orchestrator to translate the high-level requirements to infrastructure-specific deployment objectives for deploying applications and defining secure storage policies. GEN.3 was assessed by the number of use cases and platform demonstrators that SERRANO designed, implemented, and evaluated. GEN.4 was achieved since all platform demonstrators and the project use cases utilised the provided SDK to interact with the SERRANO platform services.

7.2 KPIs related to edge, cloud and HPC acceleration requirements

The KPIs related to edge, cloud, and HPC acceleration concern the accelerated UC applications created for the SERRANO project as well as the context-aware run-time orchestration system for the edge and cloud accelerators. These mainly focus on accelerated applications' performance and energy efficiency and the benefits of performing approximate computing techniques. The KPIs are related to Objective 4 "Provide acceleration and energy efficiency at the edge and cloud" of the SERRANO project, and are also described in the description of the work.

Table 18: KPIs related to edge, cloud and HPC acceleration requirements

ID	KPI	Description/Innovation	Estimated target value	Result
ACC.1	Context-aware run-time configuration	Enable context-aware run-time configuration of the approximate kernels at edge and cloud sides to meet application's/service's latency requirements	use-case dependent	Developed several different approximate versions of accelerated kernels, which trade off accuracy for performance/energy. These kernels can be exposed as different endpoints through NBFC's FaaS service and consumed by the SEERANO orchestrator to enable context-aware run-time configuration both at Edge and Cloud.
ACC.2	Improve energy efficiency	Improve energy efficiency of cloud and edge nodes over existing general-purpose architectures by utilizing FPGA and GPU accelerators	10x-100x	Final Energy gains range from ~1.3x up to ~5000x reduction for Cloud FPGAs (Xilinx Alveo) and from ~1.15x up to ~2875x for Edge FPGAs (Xilinx MPSoC) compared to the respective board's CPU performance (i.e., Intel and ARM processors).

ACC.3	Energy savings from approximate computing	Obtain further energy savings (depending on the target domain) through approximate computing techniques, without significant performance and quality degradation	20-35%	Additional energy savings range between ~20% up to ~950% for Cloud FPGAs and ~20% up to ~980% for Edge FPGAs. 4 kernels accelerated for the Anomaly Detection in Manufacturing Settings use case (IDK), 4 kernels accelerated for the FinTech use case (INB) and 3 kernels acerated for the Secure Storage use case (CC). The kernels have been accelerated for various target devices (including both heterogeneous Cloud & Edge FPGAs and GPUs), resulting in more than 100 different variations in total.

ACC.4	Number of hardware accelerators	The number of hardware accelerators (IP blocks) to be developed per use case study, mapped to SERRANO platform and used in the use case scenarios	>= 3	Additional energy savings range between ~20% up to ~950% for Cloud FPGAs and ~20% up to ~980% for Edge FPGAs. 4 kernels accelerated for the Anomaly Detection in Manufacturing Settings use case (IDK), 4 kernels accelerated for the FinTech use case (INB) and 3 kernels acerated for the Secure Storage use case (CC). The kernels have been accelerated for various target devices (including both heterogeneous Cloud & Edge FPGAs and GPUs), resulting in more than 100 different variations in total.
ACC.5	Number of interoperable functions using hardware accelerators	The number of UC hardware acceleration functions available as a serverless function, regardless of their hardware-specific implementation. Each function maps to the respective hardware-specific implementations from ACC.4.	>= 2	4 functions/kernels are accelerated in hardware.

The **ACC.2** KPI is measured as follows: Each UC algorithm has its corresponding hardware accelerator. Those accelerators are executed standalone for a pre-defined input in the SERRANO's infrastructure and the platform's energy consumption is extracted during the accelerator's execution. The extracted energy consumption is compared to the energy that is consumed by a general-purpose platform when the UC algorithm is executed standalone for the same input.

The **ACC.3** KPI is measured as follows: For the UC algorithms that allow an error margin at least one approximate accelerator is developed. The approximate GPU/FPGA designs are executed standalone in the SERRANO's infrastructure for a pre-defined input and the platform's energy consumption is extracted. The extracted energy consumption is compared to the energy consumption of the accurate version when it is executed standalone on the same platform for the same input.

With respect to the measurements of **ACC.3** for HPC, the benchmarking of the HPC services is performed with different data sizes and approximation parameters, e.g. data precision and loop perforation. Therefore, the parameters, such as the execution time, performance (FLOPS), and energy consumption, are measured and compared to the precise execution of the HPC services in order to determine the KPI value.

7.3 KPIs related to secure infrastructure requirements

The KPIs related to secure infrastructure requirements cover the main aspects of the SERRANO secure data storage mechanisms. These focus on usability, performance, reliability, and security, enabling increased observability within the SERRANO platform by collecting and correlating monitoring and telemetry data.

Table 19: KPIs related to secure infrastructure requirements.

ID	KPI	Description/Innovation	Estimated target value	Result
SIR.1	Reduction in data access latency	Reduction of read and write time for files, when system is augmented with edge storage locations and the On-premises storage gateway, compared to a purely cloud-based scenario. Reduction should be measurable for all file sizes. Estimated target value is for files with size 1 kB.	50%	For file sizes between 1MB and 15MB Read: 62-70% Write: 39-45%
SIR.2	S3 endpoint coverage of bucket and object CRUD	The Secure storage service should support the basic functionality of the S3 API concerned with buckets and objects. In particular, creating, deleting and listing buckets, creating, retrieving, deleting, listing and retrieving objects.	100%	100%, with additional endpoints to support multipart uploads and HTTP Range queries

SIR.3	Support for GDPR-compliant cloud storage locations	The Secure storage service should make it possible to use EU-based, GDPR-compatible cloud locations. This is vital in being able to accommodate for legal requirements some SERRANO users might have.	>=10	24
SIR.4	Performance improvement for TCP connections.	Speed up of TCP data transfer throughput, compared to a set-up with SW kTLS (without TLS offload).	10%-150%	9%
SIR.5	Performance improvement for TCP connections in line rate	Reduction in CPU utilisation in comparison to a set-up with SW kTLS (without TLS offload).	Reduction of 10-20%	40%

SIR.1 Reduction in data access latency: A measurement application (described in Section 4.3) has been created that uses the On-premises Storage Gateway's REST interface to first create a set of storage policies that use purely cloud-based locations, purely edge locations and a combination of the two. Following this, a measurement campaign has been conducted covering a wide range of file sizes as part of UC1.

SIR.2 S3 endpoint coverage of bucket and object CRUD: Coverage can be checked by comparing the S3-compatible storage interface's list of endpoints (Swagger UI [16] or OpenAPIv3 [17]) to Amazon's documentation [18]. This feature has also been showcased in the demo created for the mid-project review, described in Section 4.2.1.

SIR.3 Support for GDPR-compliant cloud storage locations: This information is published on the SkyFlok website [19] and is also available through an endpoint of the Cloud Telemetry API [20].

SIR.4 Performance improvement for TCP connections: We compared the improvement of TCP data transfer throughput, to a set-up with SW kTLS (without TLS offload). Measuring the throughput is done using mlnx_perf tool, which prints the throughput. The TLS acceleration demonstrator covers this KPI. This KPI has also been evaluated in the context of UC1, as described in Section 4.3.

SIR.5 Performance improvement for TCP connections in line rate: We compared the CPU utilisation of the setup in line rate, to a set-up with SW kTLS (without TLS offload). Measuring the CPU utilisation is done with the mpstat command, and checking the "idle" column, then reducing the output from 100 and getting the CPU utilisation. The TLS acceleration demonstrator covers this KPI. UC1.6, a derivative of this KPI has been evaluated in the context of UC1, as described in Section 4.3.

7.4 KPIs related to network and cloud telemetry framework requirements

The KPIs related to network and cloud telemetry framework requirements cover the main aspects of the SERRANO telemetry framework mechanisms, which enable increased observability within the SERRANO platform through the collection and correlation of monitoring and telemetry data.

Table 20: KPIs related to network and cloud telemetry framework requirements

ID	KPI	Description/Innovation	Estimated target value	Result
TEL.1	Availability of appropriate telemetry probes.	The lowest part of the SERRANO telemetry framework are the resource-specific probes that collect and forward the necessary inventory and monitoring data.	Probes to collect inventory and monitoring information from edge/cloud platforms based on Kubernetes, HPC platforms, SERRANO edge devices and on-premise gateway components.	The final release of the SERRANO platform includes the following monitoring probes: (i) K8s probe, HPC probe, and edge storage devices probe. Moreover, it is integrated with the Cloud Telemetry API from the Secure Storage service to retrieve telemetry data for the available cloud storage locations.
TEL.2	Support of streaming telemetry.	The telemetry framework should be able through the designed data analytics mechanisms to enable the data-driven provision of streaming telemetry information.	On-demand provision of concurrent streaming telemetry sessions with granularity up to 5 seconds.	The final release of the SERRANO platform supports the dynamic and on-demand provision of streaming telemetry data. A number of integration tests were performed during the evaluation of the final SERRANO platform and an example is reported in D6.7 (M36).

TEL.3	Provide estimations for network-related parameters.	The SERRANO telemetry framework mechanisms and the designed AI/ML methods should intelligently monitor the interconnection links across the distributed edge, cloud and HPC infrastructures.	Network-related metrics (e.g., available bandwidth, delay) should be available and considered by the SERRANO cognitive orchestration algorithms.	This requirement is partially implemented in the SERRANO platform through the provision of telemetry data from the available cloud storage locations by the Secure Storage service. In addition, we performed several theoretical and appropriate algorithmic solutions were developed related to the ML-based provision of estimations for network-related parameters. These works are reported in deliverables D5.3 (M15) and D5.4 (M31), also leading to two publications in internal peer-reviewed conferences and journals.

TEL.4	Storage and aggregation of telemetry information.	Other core services of the SERRANO platform along with the platform administrators should have access to the overall collected telemetry information and relevant events.	The telemetry API and a web-based user interface should provide an aggregated view of the available resources along with their current status across the overall SERRANO platform	The final release includes the Persistent Monitoring Data Storage (PMDS) that provides long-term storage for the collected telemetry data. This functionality was successfully evaluated through several tests and its usage in Demo-4 "Service Assurance and Remediation" platform evaluation. Moreover, the availability of the aggregated collected telemetry data is evaluated through several Grafana dashboards that support the platform evaluation scenarios and project use cases.
TEL.5	Correlation of telemetry data to infer metrics and localise failures.	A set of AI/ML algorithms based on collected data will provide feedback at the telemetry framework and trigger the orchestration and service assurance mechanisms.	ML algorithms will correlate information from a number of different data sources. Failure detection rate will be measured.	The final release of the SERRANO platform provides the corresponding functionality through the ML methods by the Service Assurance and Remediation service. This component leverages the collected telemetry data from the diverse monitoring probes.
TEL.6	Reprogrammable monitoring probes.	The SERRANO probes should support the configuration of their operation through a predefined REST interface.	Support at least: a) enable/disable data collection, b) enable/ disable reporting, c) change reporting interval, d) set alarm level for specific parameters.	The final version of all SERRANO monitoring probes supports the targeted functionalities. Their implementation was evaluated through several integration tests.

TEL.1 was assessed against the number of different telemetry probes that were deployed in the final version of the SERRANO evaluation testbed. More specifically, the SERRANO evaluation testbed included at least one probe for the following infrastructure resources: (i) edge/cloud platforms based on K8s, (ii) HPC platform, (iii) SERRANO edge storage devices, and (iv) cloud storage locations. Moreover, the monitoring probes for the computational resources in edge, cloud, HPC platforms collected telemetry data for the deployed applications and SERRANO-accelerated kernels. The SERRANO Enhanced Telemetry Agents successfully collected metrics from all these probe types and there was a successful integration with the HPC Gateway for collection HPC telemetry data.

TEL.2 was verified using code instrumentation in the Enhanced Telemetry Agent and monitoring probes along with specifically designed evaluation scenarios that triggered the on-demand provision of streaming telemetry. One of these scenarios is reported in deliverable D6.7 (M36), where streaming telemetry is trigger upon receiving a notification by the Service Assurance and Remediation service for some anomalous event within the SERRANO platform. Next, the ETA initiated a streaming session by requesting detailed telemetry data every 5 seconds regarding the CPU and memory performance at the affected worker nodes. The performed integration and evaluation tests assessed that the SERRANO telemetry framework mechanisms are able to automatically start, configure, and terminate the provision of streaming telemetry data.

TEL.6 was extensively tested and verified using a number of specific tests. In this case, the performed tests assessed the ability of the ETA components to enable and disable data collection, and change the reporting period for each probe type.

TEL.3 and **TEL.5** were evaluated based on the availability of estimations for at least two network-related metrics to SERRANO orchestration algorithms and the provision of feedback to the telemetry framework for adapting the monitoring probes.

TEL.4 was featured during the final demonstrators with the provision of several Grafana dashboards that, by utilizing the telemetry API, display the appropriate collected telemetry information and relevant events. The evaluation was successful since the end users and other SERRANO services (i.e., RES.1, RES.5, RES.6, SRV.8) were able to retrieve the stored telemetry data through the provided API.

7.5 KPIs related to resource orchestration and service assurance

The KPIs in this section cover the most critical aspects of the SERRANO resource orchestrator and the service assurance mechanisms. These functionalities are related to Objective 5 “Cognitive resource orchestration and transparent application deployment over edge/fog-cloud/HPC infrastructures” of the SERRANO project.

Table 21: KPIs related to resource orchestration and service assurance

ID	KPI	Description/Innovation	Estimated target value	Result
RES.1	Availability and scalable execution of cognitive and multi-object orchestration algorithms	The Resource Optimisation Toolkit should expose well-defined interfaces to facilitate the inclusion of different orchestration algorithms. Moreover, it should automatically scale the number of available instances to cope with the orchestration requests.	Integration of at least 3 designed algorithms and ability to support cloud-native execution.	Several tests were performed to verify the ROT's ability to scale with the number of requests. Moreover, the final version integrates three of the SERRANO designed algorithms (reported in D5.4) that were also used in all platform (Section 3) and use cases evaluations.
RES.2	Coordinate the workload placement across multiple orchestration platforms.	The SERRANO platform through the Resource Orchestrator should orchestrate multiple local orchestrators across the federated and heterogeneous infrastructure.	Support of Kubernetes (k8s) for edge/cloud platforms and Slurm and PBS-based batch job schedulers for HPC platforms.	The SERRANO Orchestration Drivers successfully manage platforms with local orchestrators based on K8s and HPC schedulers. This functionality was evaluated in platform demonstrations Demo-1, Demo-2, Demo-4, and in all
RES.3	Dynamic and data-driven adjustments in workload and data placement.	The SERRANO orchestration mechanisms should coordinate data and workload migration operations within the platform according to the feedback by the telemetry and service assurance mechanisms.	Automatic workload migration both across different edge/cloud resources and cloud/HPC.	This KPI is successfully covered and evaluated through the platform demonstration Demo-3 and the on-demand execution of the SERRANO accelerated kernels for project use cases in Sections 5 and 6.

RES.4	Cognitive distributed secure data storage.	The SERRANO platform should select the appropriate storage locations and coordinate the data movement between applications and the distributed storage resources according to applications' requirement and the available resources.	Data-driven definition and selection of storage policies for the optimal distribution of data across multiple edge and cloud storage locations.	The successful assessment of this KPI is described in Section 4.3. Two different storage policies were automatically created by the SERRANO orchestration mechanisms based on the user high-level requirements.
RES.5	Service Assurance using Event Detection	The SERRANO platform must be capable to detect performance related anomalies from the monitoring data. The anomalies targeted by SERRANO are contextual and temporal which are not easily identified using single attribute analysis.	Support for various pre-processing and ML based anomaly detection methods. Including both supervised and unsupervised methods. The reduction of both false positive and false negative detection of anomalous instances is of paramount importance. Supervised methods under 5% while for unsupervised methods under 15% false positive detection.	Several experiments have been carried out with the main focus on showcasing the training and inference pipelines for the EDE component. These experiments include Hyper-parameter optimisation of several supervised methods as well as unsupervised methods. Based on these experiments we have a baseline of what ML methods (including their parameters) yield good predictive performance, well under the 5 and 15% thresholds.

RES.6	Root Cause Analysis	The SERRANO platform should be capable of indicating why a particular anomalous event has occurred. This capability enables the identification of the root cause for such events.	Development and integration of Explainable AI type methods which provide additional data regarding the causes for the detected anomalous instances. This data information is to be used by the SERRANO for autonomous or semi-autonomous remediation action. Analysis execution time under 1 minute for each inference window.	Using Shapely values computed on a per predictive model we are capable of indicating what are the most impactful feature when detecting anomalous events. In conjunction with the Telemetry system naming convention of these features we can indicate for each prediction what the most likely probable cause of an anomalous event is.
RES.7	Declarative approach for describing the workload requirements.	The Resource Orchestrator should cognitively decide for the overall assignment of the applications' workloads along with the desired performance state.	The local orchestrators should take the actual deployment decisions based on the provided desired state.	Using the SERRANO Resource Orchestrator's exposed interface, the Orchestration Drivers generically described the workload deployment requirements to K8s orchestration mechanisms for the edge/cloud platforms and batch scheduler in HPC. This functionality was successfully covered and evaluated in platform demonstrations (Demo-1, Demo-2, Demo-3) and use case evaluations for the second and third project use cases.

RES.7	Declarative approach for describing the workload requirements.	The Resource Orchestrator should cognitively decide for the overall assignment of the applications' workloads along with the desired performance state.	The local orchestrators should take the actual deployment decisions based on the provided desired state.	Using the SERRANO Resource Orchestrator's exposed interface, the Orchestration Drivers generically described the workload deployment requirements to K8s orchestration mechanisms for the edge/cloud platforms and batch scheduler in HPC. This functionality was successfully covered and evaluated in platform demonstrations (Demo-1, Demo-2, Demo-3) and use case evaluations for the second and third project use cases.

RES.1 is related to the Resource Optimisation Toolkit (ROT) operation and was evaluated through ROT usage in almost all final demonstrators of the SERRANO platform and all project use cases. **RES.2** was successfully assessed as the final version of the SERRANO platform includes two different Orchestration Drivers that are able to manage the different low-level orchestration platforms that the SERRANO Resource Orchestrator unifies. The evaluation was performed through the final platform and use case demonstrators. The Resource Orchestrator, based on the decisions of the ROT, was described the applications that then were successfully deployed over edge/cloud and HPC platforms. For the edge/cloud platforms, the application services were deployed correctly through the respective Orchestration Driver and their state was retrieved and manage by the Orchestration Driver. With respect to **RES.2** for HPC, the evaluation was successful since the corresponding Orchestration Driver was able to submit and manage HPC jobs through the SERRANO HPC Gateway. **RES.3** and **RES.7** were evaluated using code instrumentation in the Resource Orchestrator and Orchestration Driver, as well as through the platform demonstrations Demo-1, Demo-2 and Demo-4 that showcased the automatic workload placement and migration between different edge/cloud and HPC resources. The evaluation was successful since the workload migrations triggered by the telemetry (**TEL.5**) and service assurance (**RES.5**) services, and the provided workload was transparently executed in edge/cloud resources through the SERRANO lightweight virtual mechanisms and in HPC through the HPC Gateway. Regarding **RES.7**, the evaluation was successful since the SERRANO Orchestration Drivers were able to

transparently describe the platform-specific workload requirements to K8s scheduling mechanisms and HPC Gateway.

RES.4 was successfully evaluated through the SERRANO platform demonstration "Intent-driven operation and automatic storage policy creation" in Section 4.3. This section provides specific details regarding the platform services involved in this process, along with a performance evaluation for uploading and downloading files using two different storage policies, one with cloud-only storage resources and one with SERRANO edge-only storage locations.

Regarding the Service Assurance and Remediation component, we have two pertinent KPIs. **RES.5** describes the performance measures used for ML-based methods. Two values are given, one for supervised and one for unsupervised methods detailing the upper threshold of a false positive detection. Although performance metrics such as F1 or Jaccard Index are more suitable when dealing with unbalanced datasets, false positive detection is of more significant impact as it can lead to unnecessary adaptation and distributions in the platform. **RES.6** aims to reduce the execution time necessary for root cause analysis. In the case of large inference windows, the execution time can significantly impact the capability of providing actionable feedback to the orchestrator. To this end, we have several accelerated and approximate variants of the analysis algorithms.

7.6 KPIs related to the ARDIA Framework and service orchestration requirements

The KPIs related to the ARDIA framework and service orchestration requirements were initially specified in deliverable D2.4 to measure the achievement level of the relevant project innovations in this topic. The KPIs were updated and reworked in deliverable D6.6 to cover the main aspects of the service orchestration and the ARDIA models, which enable information communication among the various SERRANO components for application- and service-orchestration purposes. The KPIs specified enable users to examine the extent to which the Models, developed as part of the ARDIA framework, cover the UC application requirements, including data-intensive security-critical applications. The specified KPIs also provide more information about the level of achievement of service-orchestration-related topics, such as the extent to which user-defined high-level, infrastructure-agnostic application requirements can be translated to more specific resource constraints and the role of the AI/ML techniques in this process.

Table 22: KPIs related to service orchestration requirements

ID	KPI	Description/Innovation	Estimated target value	Result
SRV.1	Application Model Expressiveness	The ARDIA Application Model should be capable of expressing all important parameters (regarding intent, profile, requirements) of an application in an infrastructure independent way.	100% coverage of the elements used for describing the UC applications.	success
SRV.2	Resource Model Expressiveness	The ARDIA Resource Model should contain all the elements being necessary for the deployment of an application from the Resource Orchestrator's point of view	100% coverage of the elements used by the Resource Orchestrator.	success
SRV.3	Telemetry Data Model Expressiveness	The ARDIA Telemetry Data Model should provide the elements required for capturing all the data collected from SERRANO infrastructure components.	100% coverage of the concepts relevant to the data shared between SERRANO components for the project purposes	success
SRV.4	ARDIA Mappings	Sufficient mappings between the high-level and medium-/low-level ARDIA models should be provided for aiding the service orchestration process.	100% of well-defined correspondences among the elements of the ARDIA models are covered.	success
SRV.5	Application Constraints Translatability	The functionality provided by the Service Orchestrator should be driven by the description of each application taking into account the translatable constraints specified along with their relative importance.	80% of application high-level constraints specified for UCs are translated to intermediate or low level constraints on average.	success
SRV.6	Security- and Privacy-aware Service Orchestration	Service Orchestration should not violate UC security and privacy constraints.	100% of relevant security and privacy parameters are taken into consideration by the Service Orchestrator.	success

SRV.7	Infrastructure-agnostic service orchestration	The ARDIA framework and the Abstraction Models developed should cover the crucial parameters of the respective components enabling service orchestration to support the execution of an application in several different infrastructure types.	Three different types of infrastructure covered including but not limited to cloud, edge and HPC.	success
SRV.8	AI-enabled Service Orchestration	Intelligent Service Orchestration with the usage of ML techniques taking into account telemetry data from the whole application lifecycle. More precisely, revise existing constraints or introduce new ones through the usage of such techniques.	At least 20% of initial constraints are affected through the usage of these mechanisms.	success

The evaluation of the KPIs mentioned in Table 22 above (**SRV.1-SRV.8**) was done via several tests, mainly driven by the data provided by each UC provider, the collected telemetry data, and the log files produced by the ARDIA Framework and especially by the AI-enhanced Service Orchestrator. The above data were used to evaluate the extent to which the abstraction models can cover user needs, the percentage of the user-defined constraints mapped to more specific resource constraints, and the contribution of AI/ML techniques in this process. Finally, the collected data were used to ensure that the particular UC requirements are satisfied. In addition, these findings are successfully demonstrated by means Demo-1 and UC3.D1 presented in this deliverable. In summary, all of the KPIs were covered successfully.

More specifically, the evaluation of the first three KPIs regarding the expressiveness of the three Abstraction Models of the ARDIA Framework was based on tests performed in collaboration with the respective project partners. In particular, the UC providers were able to express all their application requirements using the elements of the Application Model (**SRV.1**). Also, the technical experts involved with the development of the relevant SERRANO components validated that all important parameters required by the respective components (AI-enhanced Service Orchestrator, Resource Orchestrator) for the deployment and execution of an application and its micro-services were included in the Resource Model (**SRV.2**). Moreover, all the telemetry data collected could be expressed using the elements of the Telemetry Data Model (**SRV.3**).

The following two KPIs focus on the mapping rules specified and their usage by the AI-enhanced Service Orchestrator (AISO) and hence their evaluation was based on log files collected. Analysis of the log files indicated that several Mapping Rules were specified which covered well the mapping needs and corresponded correctly to the intended transformations among the elements of the ARDIA model (**SRV.4**). Accordingly, these mapping rules were used by the AISO, on the basis of the constraints and priorities specified each time by the end-user, to successfully translate the high-level constraints to deployment objectives, i.e. to appropriate intermediate or low-level constraints, also taking into account their relevant importance (**SRV.5**).

The evaluation of the last three KPIs was based on the analysis of both the elements of the three Abstraction Models and the Mapping Rules specified and validated while using the SERRANO platform. More precisely, the Application Model parameters regarding security and privacy were successfully mapped to the appropriate elements of the Resource Model and are taken into account with priority (in case of conflict) for application deployment (**SRV.6**). Also, the developed Abstraction Models allow for infrastructure-agnostic description of high-level requirements and intent, and in combination with the Mapping Rules specified enable the deployment of applications and their microservices to different resource types, including edge devices, HPC and Cloud (**SRV.7**). Finally, ML techniques were used for the specification of several Mapping Rules on the basis of telemetry data collected from the respective resources. These mapping rules affect the translation of initial constraints into deployment objectives based on previous experience from the application lifecycle (**SRV.8**).

7.7 KPIs related to integration and platform development requirements

The KPIs of this category were selected based on factors that affect the satisfaction of requirements related to integration and platform development. In the table below (Table 23).

Table 23: KPIs related to integration and platform development requirements.

ID	KPI	Description/Innovation	Estimated target value	Result
INT.1	Deployment using containers	Components that can be containerised will provide images that are stored in Docker registry and can be used to facilitate the deployment of these components in other environments	Available Docker images for all containerised components	The deployment of components using containers was successfully achieved. Docker images for all containerised components were created and stored in the Docker registry. These images can now be readily used to deploy the components in different environments, significantly simplifying the process and ensuring consistency across deployments.
INT.2	Integration point documentation	Integration points will provide documentation either in the API Spec document or through documents to describe the input, output, and intended functionality of all functions that are exposed by each component.	Documentation for all integration points	Comprehensive documentation for all integration points was successfully created and compiled. This documentation includes detailed descriptions of the input, output, and intended functionality of each function exposed by every component
INT.3	Code of components and CI/CD configuration in the same repo	All components that provide source code will store the code in the GitHub repository of the Project. Also, CI/CD will be stored inside the corresponding folders of the repository.	One GitHub repository for all components and CI/CD configuration files	The integration of the code of components and their corresponding CI/CD configurations into a single GitHub repository was successfully completed.
INT.4	Critical security vulnerabilities in component source code	DevSecOps approach enables code vulnerability scanning during each build of the components resulting in a report of all serious bugs and security vulnerabilities that have to be resolved.	0 (No security vulnerabilities should be present.)	The implementation of the DevSecOps generated detailed reports identifying all serious bugs and security vulnerabilities. As a result, all identified critical security vulnerabilities in the component source code were addressed and resolved.

INT.5	Unit and Integration tests	All components will provide unit and, when applicable, integration tests as part of the CI/CD pipeline. These will need to run successfully to create a new build of the components.	Available unit and integration tests for all components, covering all platform functionalities.	The implementation of unit and integration tests for all components as part of the CI/CD pipeline was successfully completed. Each component now includes comprehensive unit tests, and where applicable, integration tests.
INT.6	Availability of SERRANO SDK	The SERRANO SDK will expose the developed APIs from the individual SERRANO services to support the development of deployment of applications that fully leverage the provided innovations.	The three project UCs should utilise the SERRANO SDK to interact with the SERRANO platform.	The SERRANO Software Development Kit (SDK) was successfully developed and made available, effectively exposing the APIs from the individual SERRANO services. This SDK facilitates the development and deployment of applications that leverage the innovations provided by the SERRANO platform.

Regarding **INT.2**, the documentation for all integration points has been checked manually, ensuring it conforms to the OpenAPI Spec in GitHub or is among other supporting files within the relevant GitHub repositories. Regarding **INT.3**, vulnerabilities have been checked using SonarQube, as part of the CI/CD pipeline all partners use. Vulnerabilities reported by Dependency-Track and Trivy have been minimised by updating dependencies and system software. For **INT.5**, unit and integration tests are run by the aforementioned CI/CD pipeline. It was the responsibility of each component developer to write and check the status of their tests. Concerning the other, more self-explanatory KPIs defined in this subsection, these were assessed based on their description with the help of the use case applications.

8 Summary

D6.8 builds upon D6.4 (M20) and presents the final results from the comprehensive evaluation of the SERRANO platform. While D6.4 focused on results obtained during ongoing integration work based on the initial release of the SERRANO platform (M1-M18), D6.8 evaluates the final release of the SERRANO platform. The evaluation was performed through the three project use cases across diverse domains with demanding and heterogeneous requirements. Additionally, several platform-level demonstrations were conducted. Section 3 describes how the SERRANO platform accomplishes its objectives through four successful demos. Each use case is separately addressed in dedicated sections providing insights into how the SERRANO platform was evaluated through the project's use case applications. Finally, Section 7 further enriches the KPIs introduced in D6.2 (M18) and finalized in D6.6 (M27) by incorporating the results from the numerous evaluations.

Based on the evaluation results and the presented demos, the SERRANO project has successfully achieved its goal of attaining the specified KPI results. The SERRANO project successfully and significantly expanded the boundaries of cloud computing by demonstrating the transparent and secure deployment of complex business applications. This deployment spans a computing continuum that integrates federated and highly diverse computational and storage resources, including HPC nodes, GPUs, FPGAs, edge storage devices, accelerated and configurable networks, on-side edge-computing, while hiding all this complexity through a cognitive layer of abstraction.

9 References

[1] “Deliverable 2.5 - Final Version of SERRANO Architecture,” SERRANO consortium, 2022.

[2] “SkyFlok,” Chocolate Cloud, [Online]. Available: https://www.Skyflok.com.

[3] “SERRANO Service Assurance Edge repository.,” SERRANO consortium, [Online]. Available: https://github.com/ict-serrano/service-assurance-ede.

[4] “Deliverable D6.6 - Final version of KPIs and evaluation methodology,” SERRANO Consortium, 2023.

[5] “s3f repository,” [Online]. Available: https://github.com/s3fs-fuse/s3fs-fuse.

[6] “FUSE (Filesystem in USErspace),” [Online]. Available: https://www.kernel.org/doc/html/latest/filesystems/fuse.html.

[7] “fstab documentation,” [Online]. Available: https://wiki.archlinux.org/title/fstab.

[8] “SkyFlok cloud storage performance.,” Chocolate Cloud, [Online]. Available: https://www.skyflok.com/backend-performance/.

[9] “Deliverable D3.4 - Final release of SERRANO Secure Infrastructure Layer,” SERRANO Consortium, 2023.

[10 “SERRANO Measurement script for UC1.1 and UC1.2 repository.,” SERRANO Consortium, [Online]. Available: https://github.com/ict-serrano/On-Premise-Storage-Gateway/blob/master/measurements/scripts/run_performance_measurements.py .

[11 “Deliverable 6.7 - Final version of SERRANO integrated platform,” SERRANO Consortium, 2023.

[12 “SERRANO measurement script for assessing caching performance repository.,” SERRANO Consortium, [Online]. Available: https://github.com/ict-serrano/On-Premise-Storage-Gateway/blob/master/measurements/scripts/run_cache_measurements.py.

[13 “SERRANO measurement script for measuring multipart upload performance repository.,” SERRANO Consortium, [Online]. Available: https://github.com/ict-serrano/On-Premise-Storage-Gateway/blob/master/measurements/scripts/run_large_file_measurements.py .

[14 “Deliverable D4.4 - Final Release of the SERRANO Cloud and Edge Acceleration Platforms and Tools,” SERRANO Consortium, 2023.

[15 "SERRANO automated measurement script for UC1.6 repository.," SERRANO Consortium, [Online]. Available: https://github.com/ict-serrano/On-Premise-Storage-Gateway/blob/master/measurements/scripts/run_MLNX_measurements.py.

[16 "SERRANO Swagger UI documentation.," SERRANO Consortium, [Online]. Available: https://on-premise-storage-gateway.services.cloud.ict-serrano.eu/docs .

[17 "SERRANO OpenAPI v3 gateway repostory.," SERRANO Consortium, [Online]. Available: https://github.com/ict-serrano/On-Premise-Storage-Gateway/blob/master/openapi.json.

[18 "Amazon S3 documentation.," Amazon, [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/API/API_Operations_Amazon_Simple_Storage_Service.html .

[19 "SkyFlok cloud locations.," Chcoloate Cloud, [Online]. Available: https://www.skyflok.com/cloud-locations-3/.

[20 "SERRANO Cloud Telemetry API.," SERRANO Consortium, [Online]. Available: https://on-premise-storage-gateway.services.cloud.ict-serrano.eu/cloud_locations.

[21 "Python unittest framework:," [Online]. Available: https://docs.python.org/3/library/unittest.html.