EXTRACT

A distributed data-mining software platform for extreme data across the compute continuum

D1.4 EXTRACT Use-Cases Evaluation

Version 1.1

Documentation Information

Contract Number	**101093110**
Project Website	www.extract-project.eu
Contractual Deadline	M39, 31st March 2026
Dissemination Level	PU
Nature	R
Author	LRI
Contributors	BSC, OBSPARIS, IBM, IKERLAN, MATH
Reviewer	SIXSQ
DOI	10.5281/zenodo.19207989
Keywords	Personalized Evacuation Route (PER), EXTRACT, Compute Continuum, Reinforcement Learning (RL), Scientific Data Processing (TASKA), Urban Digital Twin (UDT), 5G MEC, Workflow Orchestration, Radio Astronomy, Branching Dueling Q-Network (BDQ), Key Performance Indicators (KPIs), MurMuRe, SpikeNet

The EXTRACT Project has received funding from the European Union's Horizon Europe programme under grant agreement number 101093110.

Change Log

Version	Description Change
V0.1	LRI-OBS: Definition of Table of Contents and overall document structure
V0.2	LRI-OBS: Drafting of Executive Summary and Introduction
V0.3	LRI - MATH - OBS - BSC: Definition of evaluation framework and methodology, including KPI alignment
V0.4	IBM: Integration contribution on “closing the loop” and architecture updates
V0.5	OBS: Integration and consolidation of contribution for TASKA use case
V0.6	BSC: Integration of Multi-Agent Reinforcement Learning contribution
V0.7	MATHEMA: Integration oftechnical contribution and model-related components
V0.8	LRI-OBS: Completion of PER use case integration and validation results
V0.9	LRI-OBS-BSC: Cross-use-case analysis, lessons learned, and sustainability considerations
V1.0	LRI - OBS - BSC: Consolidation and final revision
V1.1	Updated version, including EC recommendations: 1. expanded the PER evaluation with additional scalability experiments and large-scale validation; 2. clarified the heuristic baseline and experimental methodology; 3. strengthened the discussion of RL limitations and future work; 4. improved the description of extreme-data characteristics for both PER and TASKA; 5. updated architectural descriptions and figures; 6. clarified evaluation results and conclusions in response to the reviewers' recommendations.

Table of Contents

1. Executive Summary

Deliverable D1.4 – *EXTRACT Use-Cases Evaluation* presents the final validation and assessment of the two EXTRACT use cases: **PER (Personalised Evacuation Routing)** and **TASKA (Scientific Data Processing and Analysis)**, as defined in the Grant Agreement and developed throughout Work Package 1. In line with Task 1.4, the deliverable provides a structured, KPI-driven evaluation of how the EXTRACT architecture, tools, and workflows meet the project's objectives and expected outcomes.

The evaluation represents the culmination of the iterative design and implementation process initiated in D1.2 and consolidated in D1.3. It assesses the performance, robustness, and maturity of the EXTRACT platform under realistic operational conditions, including simulation-based validation and pilot-level deployments. Particular emphasis is placed on the system's ability to operate across the compute continuum, integrating edge (MEC), cloud, and HPC resources.

The adopted evaluation framework aligns with Objective O4.1 and combines **baseline comparisons**, **KPI-driven performance analysis**, and **infrastructure and application-level monitoring**. This approach enables a comprehensive assessment of both use-case-specific performance and cross-cutting system properties such as scalability, efficiency, responsiveness, and interoperability.

For the **PER use case**, the evaluation focuses on crisis management performance, including evacuation efficiency, system responsiveness, and real-time adaptability enabled by reinforcement learning and edge computing. Validation activities, including simulation scenarios and pilot testing in Venice, demonstrate the system's capability to support dynamic and personalised evacuation strategies.

For the **TASKA use case**, the evaluation addresses large-scale scientific data processing workflows, assessing improvements in processing efficiency, data reduction, scheduling, and detection accuracy across distributed computing environments. Results highlight the effectiveness of EXTRACT in enabling scalable and efficient analysis of high-volume scientific data.

Overall, the evaluation confirms the feasibility and added value of the EXTRACT approach, while also identifying remaining limitations and areas for refinement. These findings inform recommendations for further optimisation, as well as future exploitation and long-term sustainability of the EXTRACT platform.

Version 1.1 – Revision Note

This revised version of **Deliverable D1.4 – EXTRACT Use Cases Evaluation** has been prepared in response to the recommendations provided in the **Final Project Review Report**.

To facilitate the review process:

- **All additions and modifications introduced in Version 1.1 are highlighted in blue throughout the document.**
- **The revised deliverable incorporates the clarifications and expanded presentation of the validation activities described in the consortium's post-review response.**

The main revisions are summarized below.

Reviewer's Recommendation / Concern	Section Added or Modified in v1.1	Page(s) in v1.1	Action Taken & Content Overview
Clarify the operation of the PER workflow using the heuristic model	Section 3.4.1 & Section 4.3.1	Page 10, Pages 30–31	Expanded description of the heuristic routing workflow, deterministic path baseline, and overall validation methodology.
Clarify closed-loop edge operations visually	Figure 4.2 (Section 4.1.1)	Page 14	Modified the main PER architecture design diagram to explicitly illustrate the "5G MEC Edge" logical block and show how the real-time feedback loop is closed.
Clarify extreme-data characteristics of the PER use case	Section 4.1.3.1 UDT Scalability (NEW)	Page 18	Defines high velocity, variety, variety from distributed sources, and graph density in the Venice pilot.
Demonstrate PER infrastructure scalability independently of the RL policy	Section 4.3.1.2 UDT Ingestion Scalability Tests (NEW)	Page 30	Validates data ingestion scaling up to 150,000 simulated devices (and stress tests of up to 1.1M) on Kubernetes.
Demonstrate simulator engine scalability under high workloads	Section 4.4.1 Scalability and Performance (NEW)	Page 37	Proves simulator performance limits under high-density loads independently of reinforcement learning.
Explicitly acknowledge Reinforcement Learning limitations	Section 4.6.1 Scope of the scalability validation (NEW)	Page 46	Acknowledges BDQ convergence degradation on graphs exceeding 150 nodes, decoupling algorithm constraints from PER infrastructure scalability.
Resolve overlapping/mislabeled evaluation plots	Figures 4.5 & 4.6 (Section 4.3)	Pages 32–33	Cleanly separated and correctly labeled the evacuation routing outputs into Violin plots (Figure 4.5) and ECDF curves (Figure 4.6).

Clarify extreme-data characteristics & evaluation scope of TASKA	Section 5.3.2.1 Scope of the TASKA use cases evaluation (NEW)	Page 62	Explains data volumes (8 TB/hr), speed, sparsity, and parallel computing configurations across multi-cloud sites.
Justify Dataplug vs. Static Partitioning baseline comparison	Section 5.3.5.1 Comments on Dataplug in the context of scientific data repositories in the Cloud (NEW)	Page 67	Explains S3 HTTP byte-range limits and peer-reviewed CCGrid 2024 validation justifying the baseline choice.

The overall structure of the deliverable has intentionally been preserved. The revisions introduced in Version 1.1 have been incorporated to address the recommendations of the **Final Project Review** while maintaining consistency with the original deliverable.

2. Introduction

2.1. Relationship with other WPs and Previous Deliverables

Deliverable D1.4 builds on D1.2 and D1.3, which defined and consolidated the PER and TASKA use cases and their integration within the EXTRACT platform. While those deliverables focused on design and implementation, D1.4 provides their final evaluation.

The evaluation relies on inputs from other technical work packages. In particular, WP3 (D3.4) provides the compute continuum infrastructure and monitoring mechanisms enabling performance measurement across edge, cloud, and HPC environments. WP4 (D4.4) defines the monitoring and validation framework used to collect and analyse system and application-level data.

In addition, WP5 (D5.3) provides a complementary perspective by assessing project-level KPIs related to dissemination, exploitation, and sustainability, supporting the interpretation of the use-case evaluation results in terms of impact, uptake, and long-term viability.

D1.4 leverages these components to perform a KPI-driven evaluation of the use cases, ensuring alignment with the objectives and metrics defined in the Grant Agreement.

2.2. Structure of the Document

This document is structured into eight main sections, followed by annexes.

Section 1 provides the executive summary.

Section 2 introduces the document and its objectives.

Section 3 presents the evaluation framework and methodology, including validation objectives, KPI alignment, baseline definitions, and data collection approaches.

Sections 4 and 5 detail the evaluation of the PER and TASKA use cases, respectively, covering baseline comparisons and KPI-driven performance assessment.

Section 6 presents the overall evaluation results and lessons learned across use cases.

Section 7 addresses sustainability, transferability, and exploitation roadmap considerations.

Section 8 concludes the document.

The document is complemented by annexes providing supporting material, followed by acronyms, abbreviations, and references.

3. Evaluation Framework and Methodology

3.1. Validation Objectives per Use Case

The validation of the EXTRACT use cases is conducted following a goal-driven, qualitative and quantitative assessment framework, in line with the objectives and Key Performance Indicators (KPIs) defined in the Grant Agreement. This framework aims to systematically evaluate the performance and effectiveness of both the PER and TASKA use cases under realistic operational conditions.

The evaluation is structured across multiple levels. At the **use-case level**, it assesses the achievement of domain-specific objectives, such as evacuation efficiency and responsiveness in PER, and data processing performance and accuracy in TASKA. At the **technical level**, it evaluates the contribution of core components developed in WP2, WP3, and WP4, including data infrastructure, orchestration, and monitoring mechanisms. Finally, at the **system-wide level**, the evaluation considers cross-cutting properties of the EXTRACT platform, such as latency, scalability, energy efficiency, and security.

This multi-level approach ensures a comprehensive assessment of both the functional performance of the use cases and the overall behaviour of the EXTRACT system.

3.1.1. PER Validation Objectives

The validation objectives of the RL component of the PER use case are:

- **Model convergence and robustness**: Verify that the RL agent exhibits stable policy convergence during training and produces valid routing decisions within defined latency and safety constraints during inference.
- **Inference latency**: Ensuring the RL inference engineer executes within the strict latency constraints required by the UDT-SIM-RL loop, confirming its viability for edge deployment.
- **Scalability**: Evaluate whether the agent can process routing requests efficiently for larger maps and higher numbers of individuals while maintaining route optimality and system responsiveness.

3.1.2. TASKA Validation Objectives

The validation objectives of the components of the framework of the TASKA use case are:

- **Scientific integrity and traceability:** The automation provided by the workflow framework (real-time or offline processing) should ensure the data integrity (e.g. data partitioning, data compression) as well as the quality and traceability of the scientific products.
- **Data processing latency:** Ensuring that the data processing workflow solution is faster than classical data processing (in a single processing center with manual triggering)
- **Scalability:** Evaluate whether the workflow framework enables scalability to larger datasets without loss of (relative) performance.

3.2. Alignment with KPIs and Expected Outcomes

The evaluation is aligned with the KPIs and expected outcomes defined in the GA, ensuring a direct mapping between system performance and project objectives. For each use case, observed results are assessed against the relevant KPIs, covering both use-case-specific aspects and cross-cutting system properties (e.g. latency, scalability, resource efficiency).

This alignment enables a consistent comparison between targeted and achieved performance, supporting a transparent validation of the EXTRACT platform.

3.3. Use-Case Specific KPIs (O4.1)

The evaluation of the EXTRACT use cases is based on a set of use-case-specific KPIs defined under Objective O4.1 in the GA. These KPIs provide measurable indicators to assess the performance and impact of the PER and TASKA use cases, reflecting their distinct operational goals.

For the **PER use case**, the KPIs focus on crisis management performance and user-centred aspects, including **Timeliness**, **Effectiveness**, **Rapidity**, **Safety**, **Fairness**, and **Technology Acceptance**. These indicators capture the system's ability to provide timely and reliable evacuation support, ensure equitable routing strategies, and achieve user trust and usability in emergency scenarios.

For the **TASKA use case**, the KPIs address large-scale scientific data processing and analysis, including **Processing/Observation ratio**, **Selective Storage Reduction**, **Detection Fidelity**, **Restoration and Calibration**, and **Data Delivery**. These indicators evaluate the efficiency and accuracy of the data processing pipeline, as well as the system's ability to optimise storage and deliver high-quality scientific outputs.

Together, these KPIs form the basis for the quantitative and qualitative evaluation presented in the following sections, enabling a structured assessment of the EXTRACT platform across both application domains.

3.3.1. PER KPIs

The evaluation of the PER use case is aligned with the defined KPIs:

- **Timeliness / Rapidity:** Measured as the reduction in evacuation time, based on the average time (or steps) required for evacuees to reach safe zones. Performance is assessed against a Dijkstra-based baseline.
- **Safety:** Evaluated by the system's ability to avoid routing evacuees through hazard zones.
- **Effectiveness:** Assessed through the successful completion of evacuation under realistic, high-density conditions.
- **Fairness:** Measured by the proportion of evacuees reaching safety, ensuring balanced routing without disadvantaging specific groups.

3.3.2. TASKA KPIs

The evaluation of the TASKA use case is aligned with the defined KPIs:

- **Efficiency**: Measured by the reduction in total data reduction time (Processing/observation ratio), the amount of resources employed (selective Storage reduction of storage space with efficient raw data cadence selection, depending on the strategy) and the data delivery time to the end-user.
- **Composability and Dynamic scheduling**: To evaluate the capability of the platform to be flexible to various workflow needs and recipes.
- **Scientific integrity & robustness**: The workflow framework and resource allocation and job distribution should not alter the scientific quality of the data products along the workflow (trueness of the detected transient sources)
- **Scalability**: To evaluate if the proposed method scales with larger data amounts compared to a manual human-driven data processing.

3.4. Baseline Measurements

Baseline measurements are derived from earlier stages of the project to enable a consistent comparison of achieved results against initial performance.

For the **PER use case**, the baseline is defined in **Task 1.2** and corresponds to the initial routing approach used prior to the integration of reinforcement learning, including traditional shortest-path methods (e.g. Dijkstra-based routing).

For the **TASKA use case**, the baseline is established in **Task 1.3** and reflects the initial data processing workflows and configurations before optimisation through the EXTRACT framework.

These baselines are consistent with the MVP results reported in **D1.2** and the final use-case configurations described in **D1.3**, and serve as reference points for assessing performance improvements in the final evaluation.

3.4.1. PER baseline

For the PER use case, the baseline is defined as a static shortest-path routing approach, specifically utilizing Dijkstra's algorithm. While effective for single-agent distance minimization, this baseline calculates routes without considering dynamic environmental variables such as real-time crowd density or the physical bandwidth (capacity) of the streets. This static baseline serves as the benchmark to evaluate the RL agent's ability to dynamically optimise routing by incorporating these previously ignored complex, multi-agent constraints.

3.4.2. TASKA baseline

For the TASKA baseline, it is defined from the average time of a manual reduction of typical data sets volume on a specific multi-node computing facility. The scientific products are also checked for integrity and correctness by comparing the outputs from the workflow with the output of the manual data processing. A dedicated evaluation tool has been developed to monitor and decide the best resource allocation strategy to minimise the workflow processing time.

3.5. Monitoring infrastructure

The monitoring architecture developed in the context of WP3 and WP4 has been deployed in the infrastructures of both use cases. This deployment enables continuous monitoring of the underlying infrastructures to ensure that both use cases operate as intended.Due to the specific nature of the TASKA use case (i.e. use case A), some components of the monitoring infrastructure had to be deployed manually (More information can be found in the software repository).

The PER use case presents several infrastructure-specific characteristics that are relevant from the monitoring architecture perspective:

- **Distributed deployment across the compute continuum:** Monitoring components are deployed across edge (MEC), cloud, and HPC environments, enabling end-to-end observability of the system. This distribution supports the collection of metrics from heterogeneous resources and ensures visibility of system behaviour across all execution layers.
- **Federated monitoring setup:** Each Compute Continuum Site maintains local monitoring components, while selected metrics are aggregated to provide a global view of system performance. This approach ensures scalability and reduces data transfer overhead while preserving cross-site observability.
- **Real-time metrics collection:** Given the time-sensitive nature of the use case, monitoring focuses on capturing metrics such as latency, system responsiveness, and resource utilisation in near real-time. This enables continuous assessment of system behaviour during dynamic evacuation scenarios.
- **Integration with orchestration mechanisms:** Monitoring data is exposed through standard interfaces and consumed by orchestration components to support data-driven execution. This allows the system to adapt dynamically to changing conditions during runtime.

- **Application-level metrics export:** In addition to infrastructure-level monitoring, PER-specific components expose custom metrics related to routing performance and system behaviour. These metrics complement the infrastructure data and support a more comprehensive evaluation of the use case.

The TASKA use case presents several infrastructure-specific differences that are relevant from the monitoring architecture perspective:

- **Restricted networking**: Networking among the nodes participating in the TASKA infrastructure is more constrained. As a result, the set of dashboards used to monitor different aspects of the Kubernetes cluster—typically downloaded from external endpoints—has been pre-downloaded and included within the Ansible project. These dashboards are then provided directly to the corresponding containers at pod creation time, rather than allowing the pods to retrieve them autonomously.
- **Selective exporter deployment**: Some of the exporters described in the aforementioned WPs were deemed not relevant for this use case and have therefore been excluded by disabling the corresponding roles in the Ansible playbook. In some cases, this is due to the absence of the monitored resources; in others, the exclusion aims to reduce infrastructure resource consumption. The disabled exporters include GPU, network metrics, and energy consumption exporters.
- **Manual deployment of the monitoring API**: The Ansible role responsible for deploying the monitoring API, which enables programmatic access to monitored metrics, was deployed manually rather than via Ansible. This approach was necessary because the TASKA infrastructure does not provide the required dependencies for the Ansible Kubernetes plugin used by this role.
- **Application-level metrics export**: In addition to infrastructure-level monitoring, use-case-specific applications expose custom metrics to be collected by the monitoring platform. This allows application-level performance and behaviour to be observed and integrated into the overall monitoring framework, complementing the metrics gathered from the underlying infrastructure.

```python
import prometheus_client as prom

prom.start_http_server(8000)

latency_histogram = prom.Histogram(
    name: "example_latencies_ms",
    documentation: "Latencies",
)
success_counter = prom.Counter(
    name: "example_success_messages",
    documentation: "Successful message exchanges",
)
fail_counter = prom.Counter(
    name: "example_fail_messages",
    documentation: "Failed message exchanges",
)
total_counter = prom.Counter(
    name: "example_total_messages",
    documentation: "Total message exchanges",
)


def log_success(latency_ms: float) -> None:
    latency_histogram.observe(latency_ms)
    success_counter.inc()
    total_counter.inc()


def log_failure() -> None:
    fail_counter.inc()
    total_counter.inc()
```

Figure 3.1: *Example of metrics exporting metrics using Python*

```
        additionalScrapeConfigs:
          - job_name: "nvidia_jetson"
            static_configs:
              - targets:
                  - jetson-node-exporter.monitoring.svc.cluster.local:8001
```

Figure 3.2: *Example of exporter discovery configurations inside the install-prometheus Ansible role*

This approach enables consistent and reproducible evaluation, as well as comparability of results across different validation scenarios.

➤ Verification of the monitoring architecture and its correct operation has been carried out through pilot tests, controlled simulations, and log-based analysis. These activities ensure that monitoring components operate reliably across the different execution environments and that the collected data is consistent and complete.

KPI measurement is based on standardised data collection and analysis procedures applied across both use cases. Metrics are derived from monitoring data and application-level outputs, ensuring traceability between observed system behaviour and the KPIs defined in the Grant Agreement.

4. Evaluation of the PER Use Case

This section presents the evaluation results of the PER use case under representative operational conditions. The analysis focuses on the system's performance with respect to the KPIs defined in Section 3, using data collected during simulations and pilot activities.

The results assess the effectiveness of the RL-based routing approach compared to the baseline configuration described in Section 3.4, with particular attention to its ability to support adaptive, multi-agent-aware evacuation across the compute continuum.

In addition, the section reflects the evolution of the PER architecture throughout the project, highlighting updates introduced as a result of validation activities and KPI-driven optimisation.

4.1. PER Objectives and Final Architecture Reference

4.1.1 Architectural Overview and Design Evolution

During the second half of the project, a significant change in PER architecture was realized, with several improvement goals in mind:

1. Properly align external device handling in a single edge CCS (Compute Continuum Site) under a PER-specific logic (the "MEC")
2. Match real-world implementations that can make good use of modern resources such as 5G networks (i.e. 5G MEC - Multi-Access Edge Computing cluster)

Figure 4.1 below shows the original design for PER, as presented during the mid-project review. This design shows several flows that together comprise the PER operation, in accordance with the Data Mining Framework (see D2.3). One flow handles infinite internal high-speed RL training of the PER model in the HPC CCS. Another flow handles capturing "good" snapshots of the model based on the current city state and scoring, and delivering them to "production use" at the inference service which is supposed to deliver information to devices (people's smartphones). Finally, the main flow is supposed to both collect data from devices and external services, pass it to the UDT in the Cloud CCS to construct and publish the updated city state and consult with the current model in the inference service and derive instructions for devices how to proceed to safety. While this design is logically sound, it lacked in both aspects listed in the improvement goals above, namely which PER logic should handle the execution of the main flow at the edge and what is a good candidate infrastructure for hosting such logic and additional local services it might require.

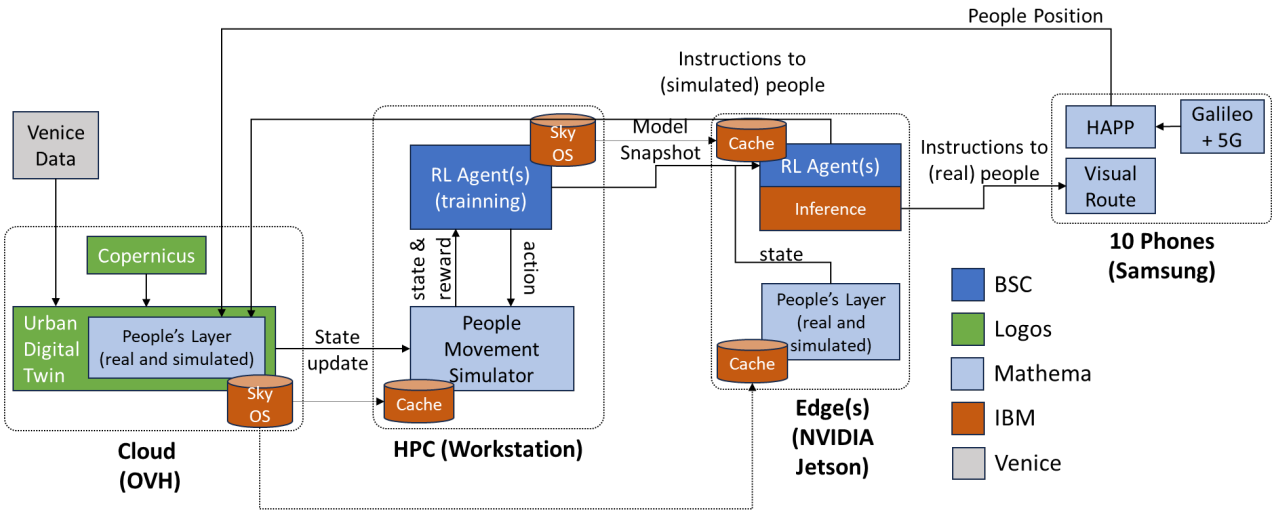


Figure 4.1: *Original PER Design*

Consequently, a modified design was accepted, as shown in Figure 4.2. In particular, a new "MEC" component is now added. Despite its name, this edge component is actually logical, as it plays a critical role in properly closing the loop of the main flow of PER. However, it is aptly named so, because it is intended to leverage modern 5G-MEC infrastructure (if such exists at the designated edge CCS location) to efficiently interact with devices at high speed.

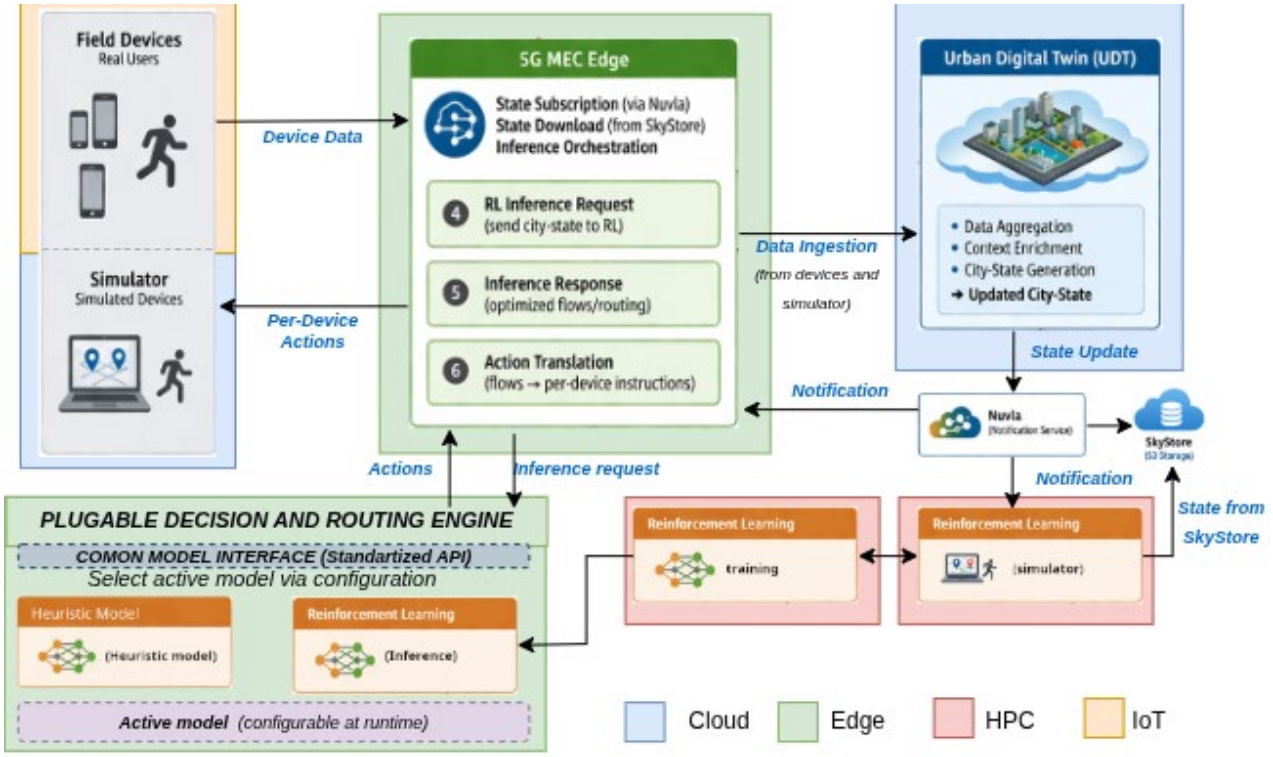


Figure 4.2: *New PER Design with "MEC" component*

With this new design, the main flow loop becomes clearer, as the MEC initiates and executes much of the data exchange related to edge operations.

1. The MEC collects data from devices (possibly using 5G operations) and publishes raw data updates for UDT to collect. This can be done periodically, based on bandwidth capacity, etc.
2. The UDT picks up raw data updates and constructs an updated "city state," which serves as a real-time state representation of the environment. This includes static graph topology (street connections, physical capacity bandwidths, and edge delays representing traversal times), the fixed locations of safe and danger nodes, and dynamic variables such as the real-time location of people across the network. The updated city state is then published.
3. The updated city state is picked up both by the simulator in HPC CCS and by the MEC in edge CCS.
4. The simulator (SIM) executes RL agent actions based on this state, handling the physical movement mechanics of evacuees through the street network. The outcome of these movements is evaluated using a reward mechanism shaped to prevent bottlenecks. This loop continuously trains the RL model to act as a dynamic load balancer across the urban grid.
5. The simulator also uses the updated city state to update training of the RL model, which may lead to eventual publishing of a new "good" model snapshot, more suitable for the updated city state.
6. The MEC uses the updated city state as input for calling the inference service (which hosts the current RL model). The RL agent computes routing instructions tailored to the current crowd distribution; for each node in the urban graph, the action space defines the proportion of evacuees to send to each connected downstream node. The inference service replies with a quantitative state change (i.e., changes in amounts of people at different locations).
7. The MEC uses a device instruction generation layer to convert the quantitative state change into device-specific instructions.
8. The MEC sends instructions back to devices, using 5G operations if possible. During the entire crisis management, instructions are periodically and repeatedly sent to devices.

➤ The PER architecture has evolved significantly from its initial version described in D1.2 and D1.3 towards a more distributed and adaptive system aligned with the compute continuum paradigm. Key updates include the integration of edge (MEC) capabilities to support low-latency decision-making, the incorporation of data-driven orchestration mechanisms developed in WP3, and the adoption of a federated monitoring framework as specified in WP4. These changes were driven by validation activities and the need to meet KPI requirements, particularly in terms of timeliness, scalability, and responsiveness under dynamic, high-density evacuation scenarios. The resulting architecture enables more efficient coordination of resources and supports real-time, multi-agent routing decisions.

4.1.2 Edge Coordination Layer: 5G MEC

The 5G MEC component represents the core operational layer of the PER system, enabling real-time interaction between field devices, the Urban Digital Twin, and the RL inference service. Its

primary role is to orchestrate the decision loop at the edge, ensuring low-latency processing and timely delivery of evacuation instructions.

The MEC is implemented as a horizontally scalable service deployed on Kubernetes. Multiple pods operate concurrently, using a **leader–worker architecture** to balance efficiency and consistency. At any given time, one pod is elected as the leader and is responsible for requesting routing decisions from the RL inference service. The remaining pods act as workers, handling communication with connected devices.

Once inference results are received, the leader publishes routing actions to a shared Redis datastore. Worker pods retrieve these actions and deliver corresponding instructions to devices through WebSocket connections. This separation of responsibilities ensures that inference is performed only once per state update while enabling scalable communication with a large number of devices.

This design provides several key advantages:

- **Scalability**, through distributed device handling
- **Fault tolerance**, via automatic leader re-election
- **Low latency**, enabled by edge deployment
- **Consistency**, ensured by centralized decision sharing

By closing the loop between sensing, decision-making, and actuation, the MEC component enables the PER system to operate effectively in real-time emergency scenarios.

4.1.3 Urban Digital Twin (UDT) Architecture and Scalability

The Urban Digital Twin (UDT) serves as the central data integration and processing platform of the PER system. It is responsible for aggregating heterogeneous data sources, enriching them, and generating a consistent and up-to-date representation of the urban environment.

The UDT architecture is structured into two complementary layers: an **infrastructure layer** and a **microservices layer** (see Figure 4.3).

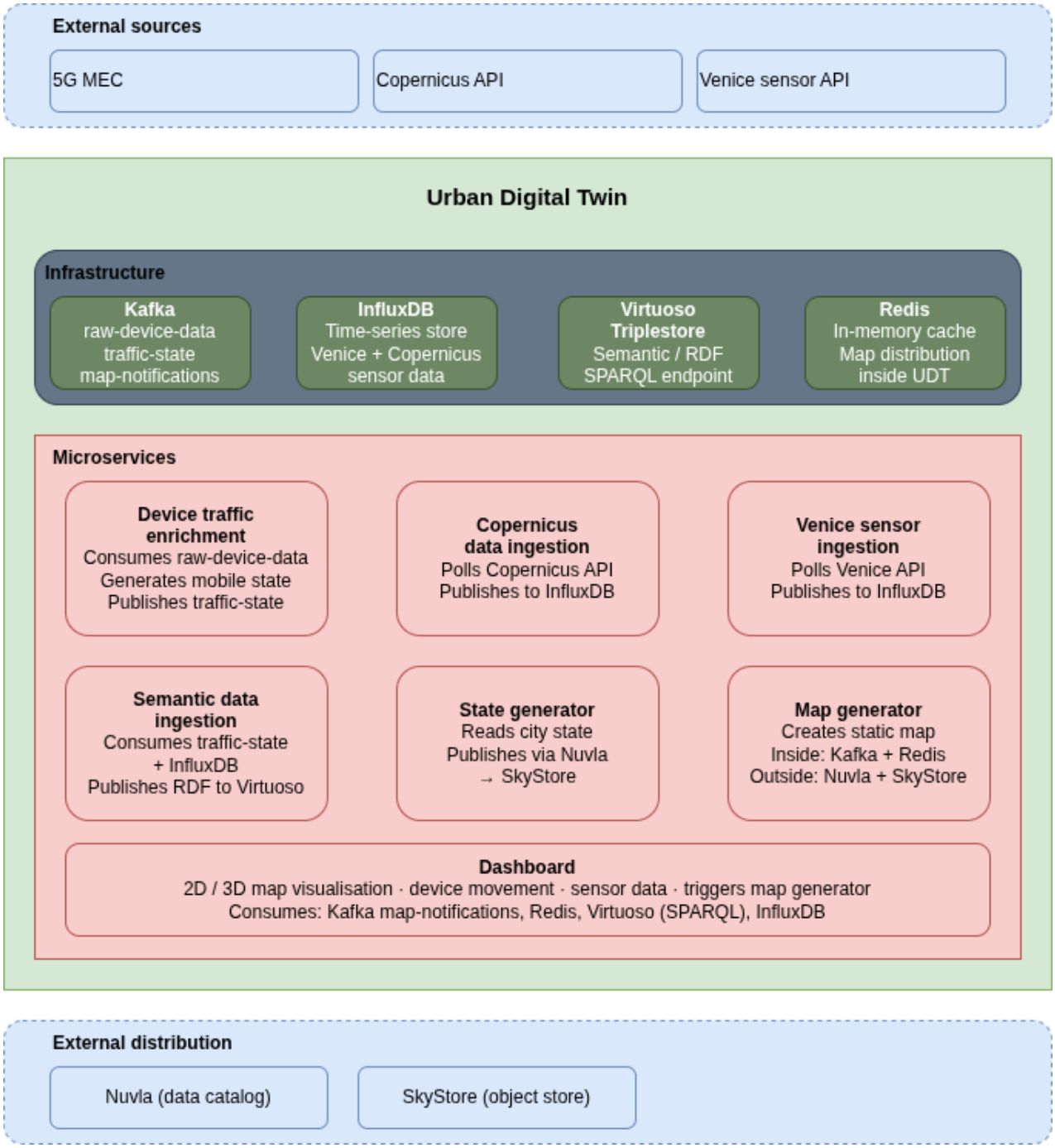


Figure 4.3: *UDT architecture*

Infrastructure Layer

The infrastructure layer provides the technological backbone required for data storage, streaming, and semantic integration:

- **Kafka** acts as the event-streaming backbone, enabling asynchronous communication between services and supporting real-time data propagation through dedicated topics.

- **InfluxDB** provides time-series storage for sensor data, ingestion events, and temporal metadata, enabling efficient time-window queries and monitoring.
- **Redis** serves as a high-speed in-memory datastore for real-time state access, supporting fast retrieval of the latest device and traffic information.
- **Virtuoso** hosts the semantic layer, storing RDF graphs that represent enriched knowledge about the urban environment and enabling SPARQL-based querying and interoperability.

Together, these components ensure that data can be ingested, processed, stored, and retrieved efficiently across the system.

Microservices Layer

On top of this infrastructure, a set of microservices implements the core data-processing pipeline:

- **Data ingestion services** (Copernicus and Venice sensor ingestion) collect external environmental and sensor data and store them in the platform.
- **Device-traffic-enrichment** processes raw device observations to derive aggregated mobility indicators such as crowd density and road occupancy.
- **Semantic-data-ingestion** transforms operational data into RDF representations aligned with the system ontology and stores them in the semantic layer.
- **State-generator-v2** combines all available data to produce an updated city-state representation, which is published through Nuvla and stored in SkyStore.
- **Map_generator_v4** creates spatial representations and map outputs to support visualization and operational awareness.
- **UDT dashboard** provides a user interface for monitoring the city state and visualizing system outputs.

This layered architecture ensures a clear separation between data infrastructure and application logic, while enabling modularity, extensibility, and scalability.

4.1.3.1 UDT Scalability (NEW)

The critical dimension for UDT scalability is the ingestion of device updates, whose volume can grow substantially as the number of connected devices increases. This scalability challenge is addressed at each stage of the ingestion and processing pipeline.

At the source, the simulator generates device updates and runs as a set of pods under Kubernetes; the volume of simulated devices that can be handled is scaled by increasing the number of simulator pods. The 5G-MEC component acts as the entry point for device data into the UDT and scales horizontally as incoming traffic increases, allowing it to absorb a growing number of concurrent device connections. The 5G-MEC forwards the received device data into Apache Kafka, where it is published to a dedicated topic. The scalability of this ingestion stage is governed by the number of partitions configured for that topic, since partitions define the level of parallelism

available for both writing and consuming messages. Scaling beyond the currently configured number of partitions requires a redeployment with an updated topic configuration, as the partition count cannot be changed dynamically at runtime.

Once device data is stored in Kafka, a Kafka Streams-based processing pipeline consumes from the topic and scales its processing in line with incoming traffic, up to the parallelism allowed by the number of partitions. If traffic approaches the limit anticipated at deployment time, a new deployment with an increased partition count is required to raise this ceiling further. During this stage, the incoming device-level data is aggregated and reduced down to the granularity of city nodes, rather than being retained at the individual-device level. The State-generator service then consumes this reduced representation and produces an updated city state from it, relying on a Kafka Streams KTable to maintain a continuously updated snapshot of the state of each road and node in the network.

This staged design — pod-level scaling for simulated device generation, horizontal scaling of the 5G-MEC ingress, and partition-bounded parallelism through Kafka and Kafka Streams — allows the UDT to scale its device-ingestion capacity by adjusting the relevant component at whichever stage becomes the bottleneck, rather than requiring uniform scaling of the entire pipeline. Quantitative results from scaling this pipeline are presented in Sections 4.3.1.2 (UDT scalability test), 4.4.1 (Simulator scalability), 4.5 (Real-World Pilot Validation).

4.1.4 Reinforcement Learning Architecture

Branching Dueling Q-Network

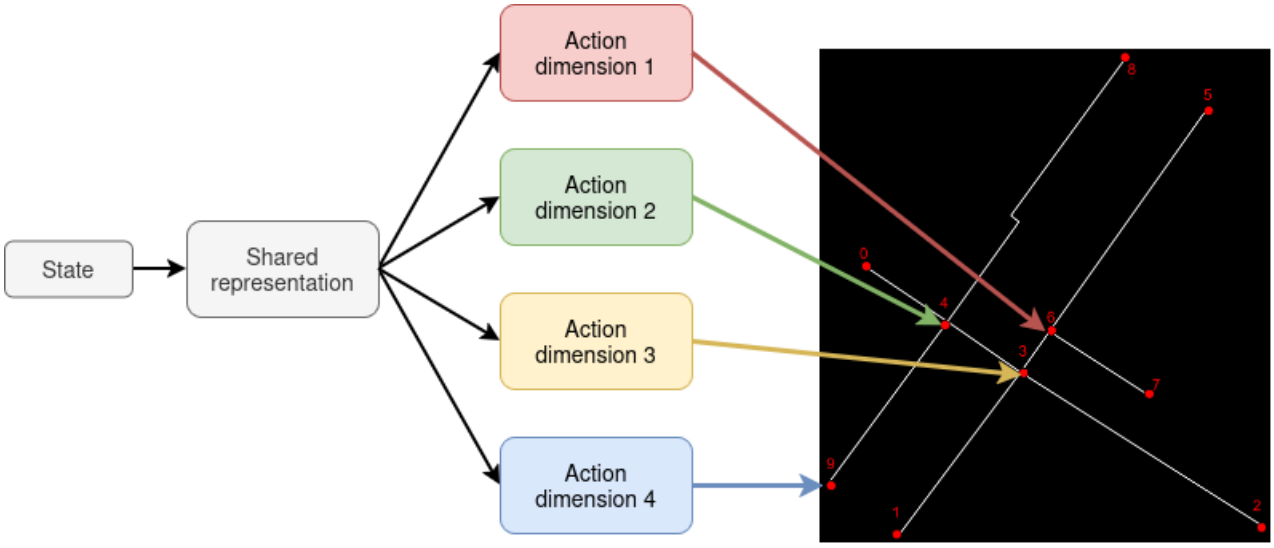


Figure 4.4: *Branching Dueling Q-Network*

The architecture used for the Reinforcement Learning model is a Branching Dueling Q-Network (BDQ, Figure 4.4), an extension of the classical Deep Q-Network (DQN) framework designed to handle high-dimensional and structured action spaces more efficiently. Unlike standard DQN,

which assumes a discrete and flat action space, BDQ factorizes the action space into multiple independent branches, each corresponding to a distinct action dimension.

In this application, the environment is a graph-based map, where nodes represent intersections and edges represent possible paths between them. At each decision step, the agent must determine how to distribute people's flow across the outgoing edges of a node.

Although this routing problem is naturally continuous, the BDQ framework operates on discrete actions. To make the problem compatible with BDQ, we use an action-encoding layer that discretizes the feasible flow allocations into a finite set of equally spaced values and retains only valid tuples whose components sum to 1. Each action therefore represents a flow distribution over the available outgoing edges. For example, for a node with three outgoing edges, the action (0.2, 0.2, 0.6) indicates that 20% of the flow is routed through the first edge, 20% through the second, and 60% through the third. This formulation allows the agent to express split-routing decisions, which is essential in routing and flow optimization problems.

Architecturally, BDQ combines a shared feature extraction module with multiple decision branches. The shared part of the network learns a global representation of the current evacuation state, including factors such as congestion, connectivity, and demand distribution. On top of this shared representation, the branching structure evaluates the routing alternatives associated with the active decision point, allowing the model to score different flow-splitting options efficiently.

The dueling component further separates the estimation of the overall quality of a given state from the relative benefit of each possible action in that state. In practice, this improves learning stability and helps the agent generalize more effectively across similar traffic situations. As a result, the model can learn not only which routing decisions are preferable, but also in which situations those decisions matter most.

RL Simulation Environment

The simulation environment is responsible for transforming real-world geographic data into a structured representation suitable for reinforcement learning. It takes map data from **OpenStreetMap** and converts it into a graph-based abstraction where nodes correspond to intersections or decision points, and edges represent traversable paths between them.

To ensure computational efficiency and tractability, the environment encodes this graph into a **multi-channel one-dimensional adjacency structure**. Rather than using a full adjacency matrix, which would scale poorly with large maps, the representation is compressed into a linearized format where each entry corresponds to a node or edge and is augmented with multiple feature channels. These channels encode dynamic and static properties of the system, including:

- The current number of people located at each node
- The flow of people along each edge
- The capacity constraints of edges at a given timestep

This design minimizes both the state dimensionality and memory footprint while preserving the essential topological and dynamical information required by the agent.

At each timestep, the environment constructs a **state vector** from this representation and feeds it into the Branching Dueling Q-Network. The model then outputs a set of branching actions, where each branch corresponds to a node and defines a distribution of flow across its outgoing edges. The environment interprets these actions as routing decisions and applies them to update the system dynamics.

Concretely, given an action tuple per node (e.g. (0.2, 0.2, 0.6)), the environment redistributes the population at that node across its outgoing edges according to the specified proportions. It then simulates the movement of people along edges, taking into account edge capacities and potential congestion delays.

After applying the action, the environment transitions to a new state by updating all node and edge values accordingly. A reward signal is then computed based on the objective of the system (e.g., minimizing congestion, travel time, or imbalance in distribution), and this reward is fed back to the agent.

This interaction loop: **state → action → environment transition → reward → next state**, forms the core training cycle of the reinforcement learning process.

A key design priority of the environment is **performance optimization**. Since reinforcement learning requires a large number of simulation steps, the environment is intentionally simplified to focus only on the essential dynamics: the position and movement of people through the network. Non-critical details (e.g., fine-grained physical simulation, visual fidelity, or extraneous geographic metadata) are omitted to reduce computational overhead.

Additionally, the use of a compact adjacency representation and vectorized updates allows the environment to scale to larger maps while maintaining fast iteration speeds. This is critical for enabling stable training of the BDQ model, as it ensures that sufficient experience can be generated within a reasonable time frame.

In summary, the environment acts as a high-performance simulator that bridges real-world map data and reinforcement learning, providing a minimal yet expressive representation of population flow dynamics and enabling efficient end-to-end training of the routing policy.

RL Reward mechanism

The outcome of these movements is evaluated using a reward mechanism explicitly shaped to prevent bottlenecks while remaining compatible with the Branching Dueling Q-Network (BDQ) and the graph-based environment. Since the agent outputs flow distributions across multiple edges rather than a single action, the reward reflects the *global impact* of many simultaneous routing decisions.

The function balances speed and safety through three components. A constant time penalty (-1 per step) encourages rapid evacuation and discourages inefficient or overly diffused flow allocations. A severe danger penalty (-10 per endangered person) enforces strict avoidance of hazardous zones—any non-zero flow assigned to unsafe edges is penalized proportionally, even after normalization of the BDQ outputs. Finally, a terminal bonus (+10) is awarded only when the entire population is evacuated with zero individuals at risk, reinforcing optimal end states.

Rewards are computed after the environment applies the agent's branching actions and simulates flow under capacity constraints. Because the state encodes node populations, edge flows, and limits, the reward naturally captures congestion and imbalance effects.

Although actions are generated per node (per branch), the reward is global, forcing implicit coordination across the network. Over time, this drives the agent to learn efficient, safe, and balanced routing strategies, effectively acting as a dynamic load balancer across the urban grid.

RL training and inference

The training and inference phases of the Branching Dueling Q-Network (BDQ) serve fundamentally different purposes and operate under distinct computational environments.

Training involves teaching the model to make optimal routing decisions by interacting with the simulation environment. During this phase, the model observes states, takes branching actions, receives rewards, and updates its network parameters to minimize temporal-difference errors. Training is, therefore, highly computationally intensive.

To train the agent efficiently, the learning process follows a progressive exploration strategy. At the beginning of training, the agent relies heavily on a heuristic routing policy that provides safe and meaningful guidance. As training advances, this guidance is gradually reduced, and the agent shifts to a combination of random exploration and learned decision-making. In the final stages, the policy becomes predominantly greedy, exploiting the knowledge accumulated during training.

This design is particularly important because the routing problem becomes significantly more complex as the size of the map increases. Larger topologies contain more intersections, more possible paths, and more congestion interactions, which increase both the number of decisions the agent must make and the time required to complete each training episode. In addition, when a junction contains more outgoing streets, the number of possible flow distributions grows rapidly, which makes the action space harder to explore.

To address this scalability challenge, training is distributed across several workers running in parallel. All workers start from the same model and use the same map, but they experience different trajectories during training because exploration remains stochastic. This allows the system to collect diverse experience simultaneously, instead of relying on a single long sequential training process.

The main benefit of this approach is the reduction in wall-clock training time. In this way, parallel execution preserves the overall training effort while significantly shortening the elapsed time needed to obtain an improved policy.

After each parallel training round, the models produced by the workers are merged through parameter-wise averaging. This aggregation step produces a single consolidated model that captures the information learned across the different training instances. The aggregated model is then evaluated on the training topology and, if needed, redistributed as the starting point for the next training round.

Inference, on the other hand, is the deployment phase where the trained BDQ model is used to make real-time routing decisions. Unlike training, inference does not involve gradient updates; the model simply evaluates the current state and outputs action distributions. This phase prioritizes **low latency and high availability**, as the routing policy must respond quickly to dynamic changes in the environment.

In this architecture, inference is handled using **KServe**, a Kubernetes-based platform for serving machine learning models. KServe provides scalable, containerized deployment of the BDQ network, enabling:

- Real-time evaluation of routing decisions in response to live population and congestion data.
- Automatic scaling to handle variable request loads.
- Integration with monitoring and logging tools for operational observability.

By separating the training (HPC) and inference (KServe) environments, the system efficiently leverages specialized infrastructure for each stage: HPC accelerates model learning, while KServe ensures robust, low-latency deployment for decision-making in operational settings.

4.1.4.1 Limitations of the current RL policy-learning component (NEW)

The distributed training strategy reduces wall-clock time, but it does not remove the underlying learning-complexity limitation of the current single-agent BDQ formulation. As the topology grows, the agent must account for a larger dynamic state, a greater number of simultaneous routing decisions, longer evacuation horizons, and more non-local congestion interactions. In addition, the discretized flow-allocation representation becomes increasingly difficult to explore at junctions with several outgoing edges, because the number of admissible allocation patterns grows rapidly with the branching degree.

Consequently, the learning problem becomes more sensitive to the choice of hyperparameters, exploration schedule, training budget, and random initialization.

This limitation can be observed in the largest evaluated setting, Topology C, with 150 nodes, where convergence was less stable and the learned policy did not consistently improve over the deterministic heuristic baseline (see following section). The issue therefore concerns the ability to train and validate a single BDQ policy that generalizes over a large urban graph; it does not indicate a limitation in the execution of the simulator itself. The environment can still represent and simulate population movements, capacity constraints, queues, and routing decisions on larger maps, but producing a sufficiently robust learned policy for such maps was not achieved.

For this reason, the full Venice-scale experiments did not rely on the RL policy as the routing controller, because the current BDQ formulation could not be validated as sufficiently stable and reliable on a topology of this size within the project timeframe. Instead, routing actions were generated using the deterministic controller. This made it possible to validate the scalability of the simulation and routing workflow independently of the current RL policy-learning limitation.

This result must be interpreted as a limitation of the current RL algorithmic formulation, rather than as a limitation of the EXTRACT infrastructure. The expanded simulations and Urban Digital Twin validation demonstrate that the overall platform can ingest and represent large urban topologies, and expose the resulting information through the Urban Digital Twin. These capabilities remain available at Venice scale when routing actions are generated by a deterministic controller. Therefore, the unvalidated element at this scale is specifically the learning, convergence, and generalization of the current monolithic BDQ policy.

The project has consequently identified the redesign of the policy-learning component as a key direction for future work. Promising alternatives include district-specific pretrained agents as a short-term operational solution, cooperative multi-agent reinforcement learning for coordinated control of bounded subgraphs, among others. These directions would build upon, rather than replace, the infrastructure already validated in the project, with the objective of enabling reliable learning-based routing at metropolitan scale.

4.2. Comparison Against PER Baseline (Task 1.2)

To assess the quality of the learned policy, we compare the RL agent against two complementary reference solutions. The first is an Upper Bound Estimate derived from a dynamic network-flow formulation, which provides the best evacuation time achievable under idealized global coordination. The second is a deterministic heuristic policy that produces feasible routing decisions using only local shortest-safe information and instantaneous edge availability. Together, these two references define a meaningful performance interval: the Upper Bound Estimate captures the theoretical target, while the heuristic represents a strong hand-crafted baseline.

The Upper Bound Estimate is obtained by modeling evacuation as a flow-over-time problem on a time-expanded network. In this formulation, each physical node is replicated across discrete time layers, transit edges connect successive layers according to travel times, and edge capacities encode the maximum number of evacuees that can traverse each street per time step. This construction is standard in dynamic-flow models because it transforms a time-dependent evacuation process into a static maximum-flow feasibility problem over an expanded graph. For a given time horizon, one can therefore test whether the entire population can be routed to safe nodes within the max time steps tied to the set horizon. Repeating this test within a binary-search procedure yields the minimum feasible evacuation horizon.

In our evaluation, this value is not interpreted as a directly deployable policy, but as an optimistic upper bound on the evacuation time that any causal controller can hope to attain on the same topology.

In contrast, the heuristic baseline is designed as an executable routing policy rather than a flow-over-time solution. For each node, it first computes the shortest distance to the nearest safe node and then ranks the outgoing edges according to the total remaining travel distance induced by each local choice. At decision time, the heuristic allocates as much flow as possible to the most promising outgoing edge, subject to the currently available bandwidth, and then continues greedily with the next-best edge until the full local demand has been assigned. The resulting split is finally projected onto the same discretized action space used by the RL agent, ensuring that both policies operate under the same admissible routing constraints.

These two references serve distinct purposes in the experimental comparison. The dynamic-flow Upper Bound Estimate quantifies how far the learned policy remains from an idealized optimum under perfect coordination, whereas the heuristic baseline measures the practical benefit of learning over a hand-crafted routing strategy. Accordingly, the RL agent should not necessarily be expected to match the theoretical benchmark, but it should consistently improve over the heuristic if the learned policy is successfully exploiting congestion patterns and non-local interactions that the greedy baseline cannot capture.

Experimental evaluation and comparison with baseline.

After an extended phase of hyperparameter tuning and validation in the HPC environment, including both serial and parallel training executions, we obtained a consolidated set of annotated experimental results. These experiments were designed to evaluate how the learned RL policy behaves across different topology families and to compare it against two reference points: the heuristic baseline and the theoretical bound previously introduced.

For this evaluation, each experiment is characterized by three outcome values: the evacuation time achieved by the RL agent, the evacuation time achieved by the heuristic baseline, and the theoretical bound for the same topology. The main metric used throughout this analysis is the number of time steps required to complete the evacuation, where lower values indicate better performance. From these values, we derive several complementary indicators: whether RL performs better or worse than the heuristic, the magnitude of that improvement or degradation, and the distance of each method to the theoretical bound.

The final dataset contains 239 valid experimental runs, grouped into three topology families, which aim to represent different sizes of urban maps, while providing insights on the limitations of the current approach.

- **Topology A:** contains 50 nodes arranged in 5 layers, with branching factors between 2 and 8 outgoing edges.
- **Topology B:** contains 80 nodes and 8 layers, with a smaller branching range of 2 to 4 outgoing edges.
- **Topology C:** is the largest and most complex case, with 150 nodes, 15 layers, and branching factors between 2 and 8 outgoing edges.

To make the comparison easier to interpret, the experiments are analyzed from two complementary perspectives. First, we study the distribution of the evacuation times produced by each method within each topology family. Second, we aggregate the results into summary indicators that show how often RL improves over the heuristic and how close each method remains to the theoretical bound.

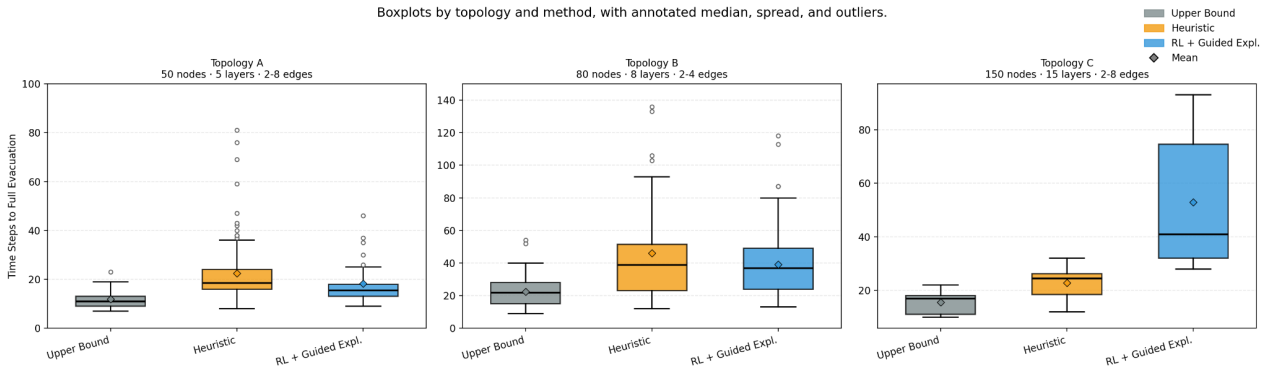


Figure 4.5: *Results of evacuation times by topologies A, B and C and methods*

The boxplot (Figure 4.5) summarizes the central tendency and variability of the results. For each topology, the plot shows the median evacuation time, the interquartile range, and the outliers for the theoretical bound, the heuristic, and the RL policy. Mean values are highlighted to complement the medians. This representation is useful to assess not only which method is better on average, but also how stable that advantage is across repeated runs.

In the smaller and medium-sized topologies, RL often improves on the heuristic, although the magnitude of that gain varies from one run to another. This can be observed in Figure 4.5, by the

Mean indicator being lower in the RL bar than in the Heuristic one, as well as by the spread of said bar. In the largest topology, the opposite behavior is observed, with the RL policy remaining noticeably farther from the bound.

Similar interpretation can be done of the following figure, which illustrates the same analysis, in a different approach, and without axis correction.

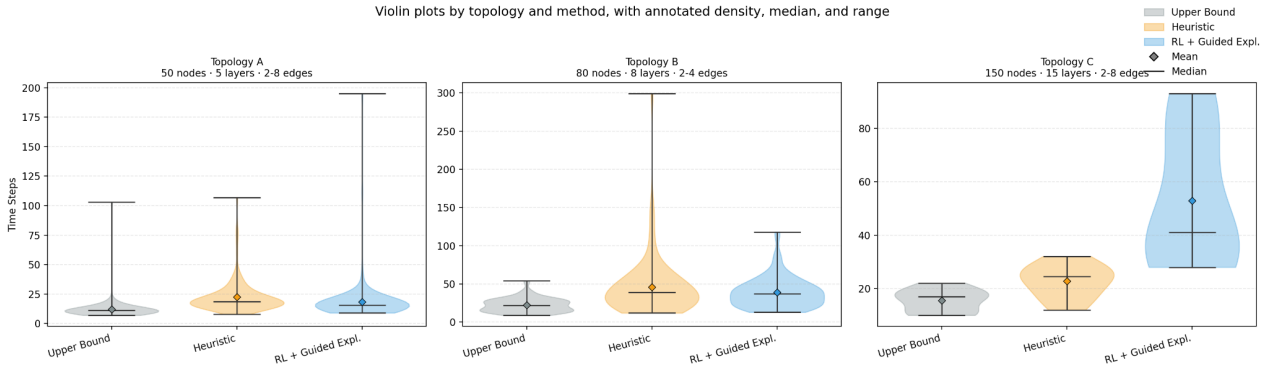


Figure 4.6: *Violin plot of evacuation time showing the full shape of the distribution for the three methods and the three topologies A, B and C.*

This violin plot (Figure 4.6) complements the previous boxplot by showing the full shape of the distribution. Wider sections correspond to ranges where more runs are concentrated, while narrow sections indicate less frequent outcomes.

Another way of observing the distribution of results, and getting a more visually clear understanding of which technique outperforms which, is by using ECDF (Empirical cumulative distribution function) plots.

The ECDF plot provides a complete view of all runs for each topology and method. Each curve shows the fraction of experiments that achieve an evacuation time at or below a given number of steps. A method whose curve appears further to the left is preferable, since it reaches lower evacuation times in a larger fraction of the experiments. In addition, steeper curves indicate more consistent behavior, while flatter curves suggest larger variability.

This plot is especially useful because it summarizes the whole distribution without hiding differences between runs.

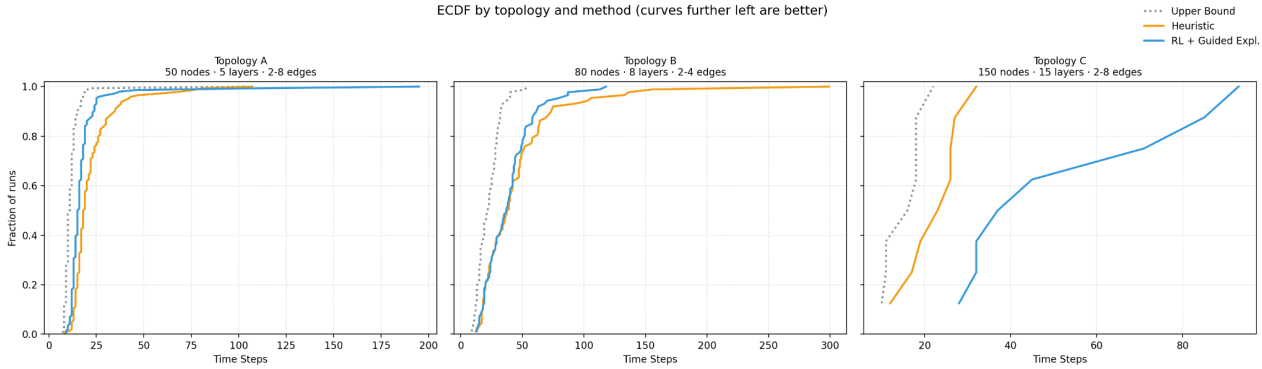


Figure 4.7: *Empirical cumulative function distribution as a function of evacuation time for the three topologies A, B and C and the three methods.*

For Topologies A and B, the RL curve lies to the left of the heuristic curve for the totality of the range, confirming that the learned policy improves evacuation time when tested against an heuristic benchmark.

For Topology B, the separation between the two curves is much bigger, and positions the heuristic to the left of the RL curve, indicating a negative outcome. Thus, it indicates that for such topology, the RL implementation seems to fail to fully learn the optimal strategy, despite converging to an actionable, and complete evacuation strategy.

A more direct comparison is provided in figure 4.8 the following figure, which reports the percentage of runs in which RL performs better than, equal to, or worse than the heuristic.

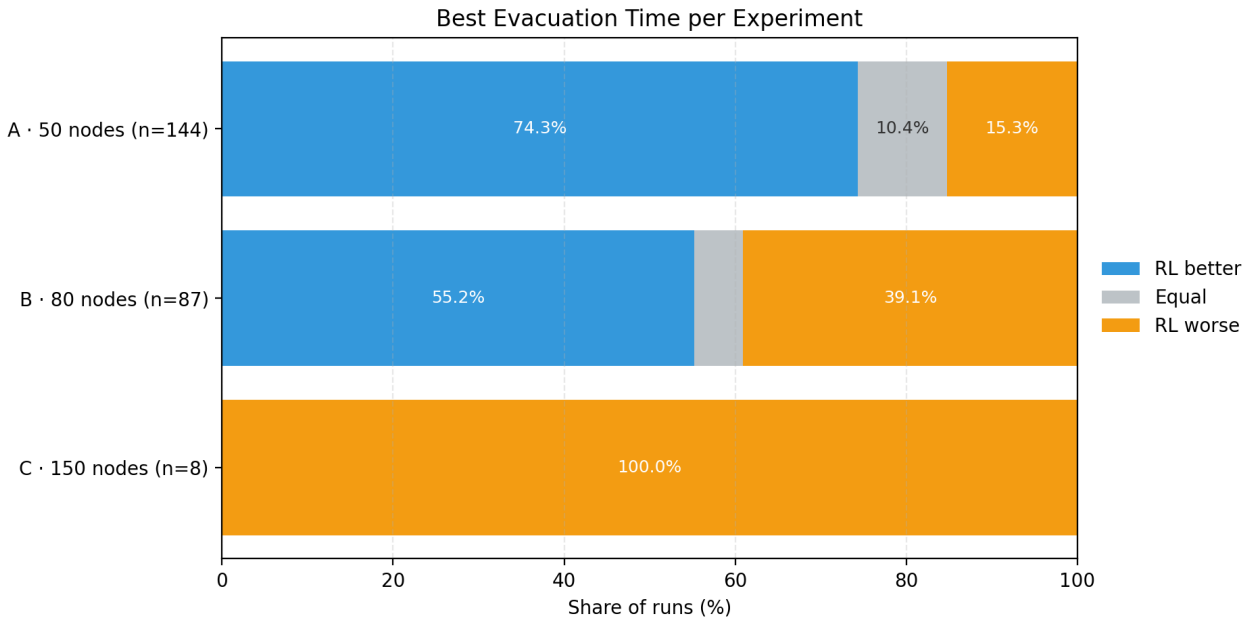


Figure 4.8: *Percentage of runs where the RL was better, equal or worse then the heuristic method.*

The results show a strong advantage for RL in Topology A. Out of 144 runs, RL performs better than the heuristic in 107 cases, which corresponds to 74.31% of the experiments. It matches the heuristic in 15 runs and performs worse in 22 runs.
For Topology B, the outcome is still favorable to RL, but much more balanced: RL is better in 48 of 87 runs, or 55.17%, and worse in 34 runs.
In Topology C, RL is worse in all 8 runs, indicating that the current learned policy does not yet scale satisfactorily to this topology class. It must be noted that the difference in experimentation is mainly caused by the computational cost of training an agent in bigger topologies.

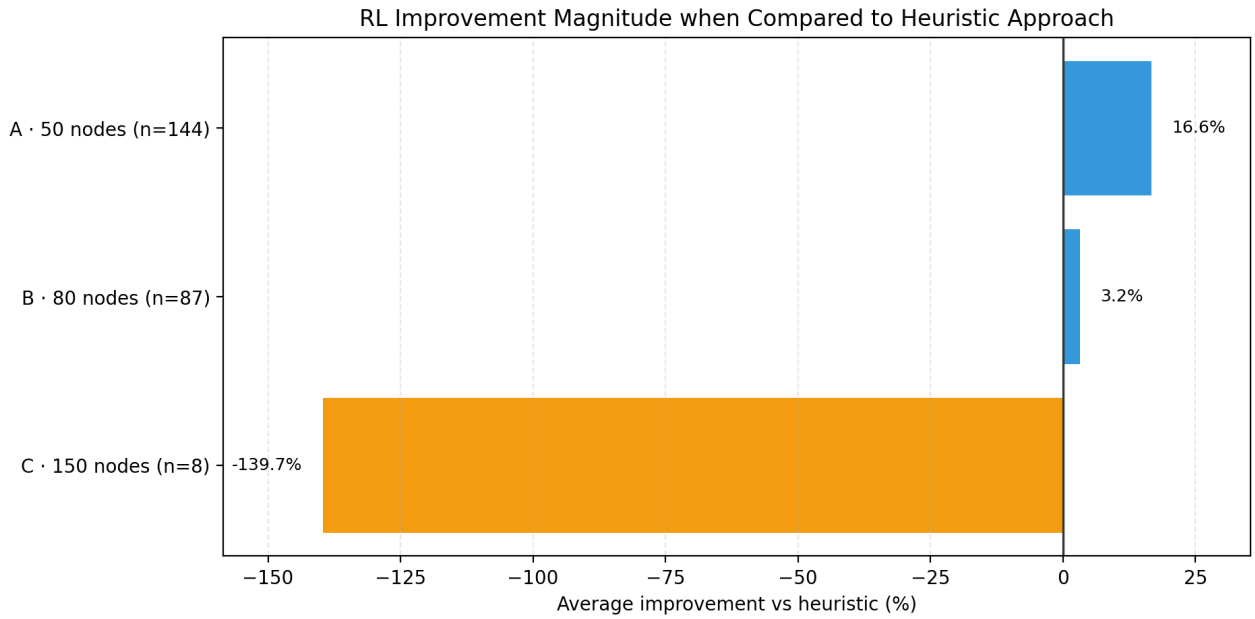


Figure 4.9: *Average RL improvements magnitude versus the heuristic approach.*

Complementing the previous figure by quantifying the average gain or loss of RL relative to the heuristic, we can observe in figure 4.9, positive values that indicate that RL reduces evacuation time, while negative values indicate degradation. This figure is important because two methods may show similar win/loss counts while differing substantially in the size of their gains.

For Topology A, RL improves over the heuristic by an average of 4.15 steps, corresponding to a mean relative improvement of 16.65%. For Topology B, the average gain is 6.95 steps, but because the absolute evacuation times are larger, the average relative improvement is only 3.16%. Topology C shows a strong negative result, with RL underperforming the heuristic by an average of 30.12 steps, which corresponds to a mean degradation of 139.68%.

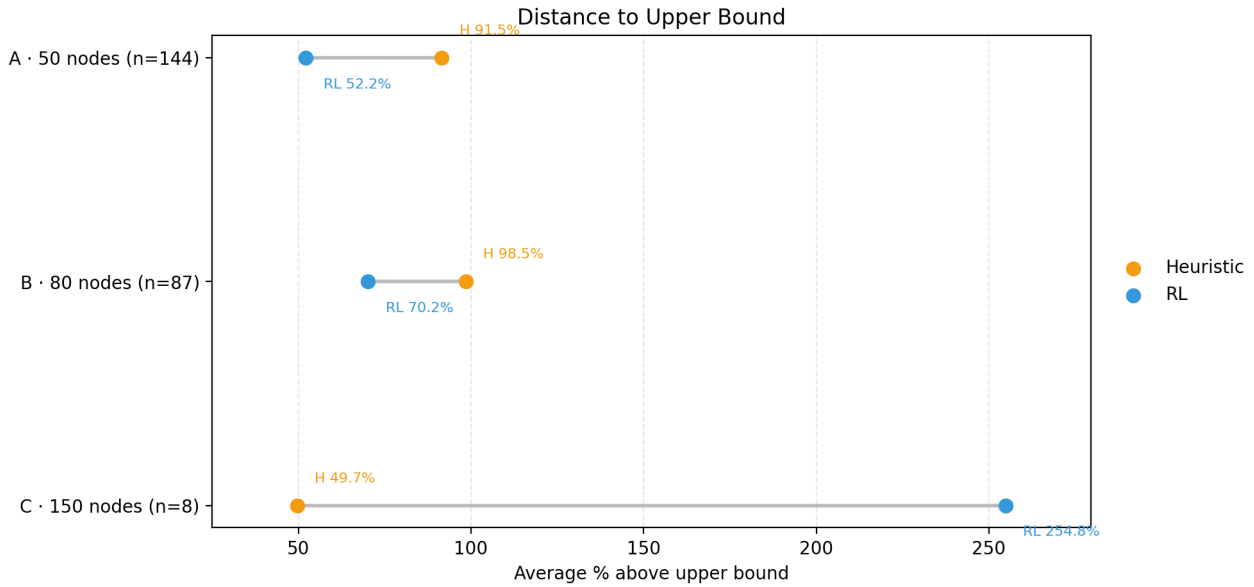


Figure 4.10: *Average distance to the theoretical bound between the two methods.*

The final summary (Figure 4.10) compares how far RL and the heuristic remain from the theoretical bound. For each topology, the plot displays the average percentage gap between the achieved

evacuation time and the bound. Smaller gaps are better, since they indicate performance closer to the idealized optimum.

This figure shows that RL is clearly closer to the bound than the heuristic in Topology A. The average gap is 52.19% for RL, compared with 91.54% for the heuristic. In Topology B, RL also remains closer to the bound, with an average gap of 70.16% versus 98.46% for the heuristic. In Topology C, however, the situation reverses: RL is much farther from the bound, with an average gap of 254.82%, while the heuristic remains at 49.68%. This confirms that the learned policy still struggles in the largest and most demanding scenarios.

Overall, the experimental evaluation shows that the proposed RL approach is effective in small and medium-sized topologies, where it consistently improves over the heuristic baseline and remains closer to the theoretical bound. The strongest results are obtained in Topology A, where RL outperforms the heuristic in nearly three quarters of the runs. Topology B still shows a positive advantage, although the margin is smaller and the variability is higher. Topology C reveals the current limit of the approach: with the present training configuration, the learned policy does not yet generalize reliably to larger topologies.

These results are valuable for two reasons. First, they validate that the RL formulation can learn better routing behavior than a carefully designed heuristic in a substantial portion of the tested scenarios. Second, they indicate that scalability remains the main open challenge, especially in topologies with a larger number of nodes, deeper layered structure, and more complex branching patterns.
This supports the need for continued work on training strategy, model capacity, and exploration design in order to improve performance in large-scale evacuation settings.

4.3. KPI-Driven Evaluation of Crisis Management Performance

4.3.1. Methodology

The evaluation of the PER use case follows a goal-driven, KPI-based methodology aligned with the objectives defined in the GA. The assessment combines simulation-based experiments and pilot activities, supported by the monitoring infrastructure developed in WP3 and WP4. KPI values are derived from both infrastructure-level metrics (e.g. system performance, latency) and application-level outputs (e.g. evacuation time, routing behaviour), ensuring full traceability between observed system behaviour and evaluation results.

The evaluation is conducted across multiple scenarios and topologies to capture system performance under varying conditions, including different population densities and dynamic environmental constraints. This enables the analysis of system behaviour in both controlled and more realistic settings. Results are systematically compared against the baseline configurations defined in Section 3.4, allowing for a consistent assessment of improvements introduced by the RL-based approach.

This methodology ensures that the evaluation is reproducible, comparable across use cases, and aligned with the overall EXTRACT validation framework.

4.3.1.2 UDT Ingestion Scalability Tests (NEW)

As part of the KPI-driven evaluation, a series of dedicated scalability experiments were conducted to assess the behaviour of the UDT device-ingestion pipeline under increasing simulated device load. Five progressive experiments were run in total, each refining pod sizing, Kafka configuration, and pipeline handling based on observations from the previous run. This section presents the results of the first and the last (fifth) experiment, which together illustrate the evolution from an initial resource-constrained configuration to a validated, optimized deployment.

In the first scalability experiment, each simulator pod was configured with a CPU request of 1 vCPU and 2 GiB RAM, hosting 1,000 simulated devices per pod. The test reached 18,000 devices (18 pods), at which point the Kubernetes scheduler was no longer able to create additional pods because the node had exhausted its allocatable CPU resources. Subsequent analysis showed that the simulator pods were significantly oversized relative to their actual resource consumption, indicating that the observed limit reflected a resource-sizing constraint rather than an architectural limitation of the ingestion pipeline.

Based on this and subsequent intermediate experiments, the pod configuration was progressively optimized. In the fifth and final experiment (Run 5), conducted on a single-node OVH managed Kubernetes cluster, simulator pods were reconfigured to a 300m CPU request / 1 CPU limit and 1 GiB RAM, each hosting 12,500 devices, deployed across 12 pods. The 5G MEC component was scaled to 14 pods pinned at an 8 GiB memory limit, and Kafka-based enrichment was configured with 12 consumers (one per partition) against 12 Kafka partitions in a clean state. Several fixes were applied relative to the preceding run, including concurrent WebSocket connection handling, clearing of stale Kafka changelog topics, flushing Redis state before each sweep, incremental (one-level-at-a-time) load ramping, and pinning downstream Horizontal Pod Autoscalers to protect the pod budget allocated to the core pipeline.

With this configuration, the platform successfully placed all 150,000 simulated devices with zero crashes and no pipeline failures. At the point the state snapshot was sampled — taken mid-ramp during the load increase, rather than at steady state — 136,590 devices (91%) were reflected in the aggregated city-state snapshot; the remaining 9% corresponds to pipeline latency inherent to sampling while load is still increasing, not to device loss. Node CPU utilization peaked at 49%, and 5G MEC memory usage peaked at 2.0 of the 8 GiB allocated per pod, indicating that the platform had further headroom beyond the validated 150,000-device level.

Table 4.1a: *UDT ingestion scalability — first vs. fifth (final validated) experiment*

Category	Parameter	Experiment 1	Experiment 5 (final)
Cluster environment	Provider	OVH managed Kubernetes	
	Node	r3-256-mem-optimized (single node)	
	vCPU / RAM	32 vCPU / 256 GiB	
	Pod cap	110 pods max	
Configuration	Simulator pods	18 pods × 1,000 devices	12 pods × 12,500 devices

	Simulator CPU/mem	1 vCPU request, 2 GiB RAM	300m request / 1 CPU limit, 1 GiB
	5G MEC		14 pods, pinned at 8 GiB limit
	Kafka / Enrichment		12 partitions, 12 consumers (1/partition)
Results	Max devices reached	18,000	150,000
	Limiting factor	Node CPU allocation exhausted	None observed (headroom remaining)
	Devices in state snapshot		136,590 (91%, mid-ramp sample)
	Crashes		0
	Node CPU peak		49%
	5G MEC memory peak		2.0 / 8 GiB

4.3.2. KPI Evaluation summary for PER

The following paragraphs (Table 4.1) analyse the PER KPIs individually, providing a detailed assessment of system performance with respect to each evaluation dimension.

Table 4.1: *Summary KPI performance table for the PER*

KPI	GA Target	Measured Value	Method	Result
Timeliness	Reduce evacuation time under dynamic conditions. Routes arrived at the right time and place. It is necessary that the position is synchronized with the individual movement and speed without latencies. +99%	RL outperforms heuristic in 74.31% (Topology A) and 55.17% (Topology B); closer to theoretical benchmark	Simulation-based evaluation across multiple topologies using monitoring data	**Partially achieved** (strong in small/medium scenarios; limited scalability)

Rapidity	The time needed to evacuate a heterogeneous and sparse number of people Average time for no notice event (30-40 min) from an affected area. - 70%	Time needed to complete the evacuation	Simulation-based evaluation across multiple topologies using monitoring data ca 500 Sec. -Cross validation on the field: ca. 10 min	**Achieved: cs -80% (time significantly reduced respect to typical no-notice event)** -
Effectiveness	Evacuation can be considered successful where the 100% of the affected population has been relocated in a safe area before or during the event. Ga target: +90%	% of Full evacuation achieved in evaluated scenarios; convergence issues in larger topologies without optimisation	Simulation experiments with RL training and validation runs	**Achieved: 100% of people has been relocated)** **Note:** scalability depends on computational optimisation
Efficiency	Resource/effort committed to evacuate k persons. In requires an optimization of the use of resources. **Resource/effort committed to evacuate k persons. - 70%**	Improved flow distribution and closer alignment to theoretical benchmark in smaller/medium scenarios	Comparative analysis against baseline using simulation outputs Baseline: Transport: 40% Coordination: 20% Idle inefficiency: 30% Support: 10% Reduction= (Ebaseline−EAI)/Ebaseline	**Mostly achieved: ca 70%** **Replace human coordination almost entirely** **Coordination =** reduced by ~90% → contributes ~18% total reduction **Eliminate most idle inefficiency** **Idle reduced by ~80–90% → contributes ~25–40% reduction** Increase network throughput drastically

				Cuts evacuation time → reduces all time-dependent costs
Safety	Avoid routing through hazardous or congested areas. Percentage of injuries/fatalities reduction during the evacuation -80%	RL dynamically adapts routes to avoid unsafe conditions *(qualitative evidence)*	Scenario-based evaluation using dynamic environment conditions. Number of people safely directed.	**Achieved: -100% achieved** *(100% people have been safely directed)*
Fairness	This term depended on the risk level of the person, the mobility capacity, the time required to reach a safe shelter/Sate Collecting Points Ensure balanced evacuation across population. 95%	Improved distribution of routing decisions; avoids overloading specific routes *(qualitative)*	Analysis of routing distribution across agents. Number of people receiving a safe route tailored on movement needs and aligned with the evacuation timing.	**Achieved: 100%**
Technology Acceptance	Ensure usability and adoption in real-world conditions Assessment of PER in terms of "willing to use" of the citizens. This is a critical point that could determine the impact of the solution. >95%	Positive feedback from pilot activities	Pilot validation (Venice tests, user feedback)	**Qualitatively achieved (see annex)**

System Timeliness (Latency)	Enable near real-time decision-making. **NB: official warning system** during a **rapid-onset hazard** would take **180 minutes (3 hours)** to warn 92% of at-risk residents (source: Mas, E. et al. (2023). *Modeling of multi-hazard warning dissemination time distributions: An agent-based approach.* International Journal of Disaster Risk Reduction. Tagret: +99%	Low-latency processing enabled via MEC; near real-time updates and routing decisions	Monitoring of end-to-end pipeline (UDT → MEC → RL inference)	**Achieved +99%** (scale reduction from hours to seconds)

Overall, the results demonstrate that the PER system achieves strong performance in key dimensions such as timeliness, effectiveness, and efficiency, while highlighting areas for further improvement, particularly in terms of scalability and quantitative assessment of safety and fairness.

The performance of the PER system was evaluated using a set of KPIs aligned with operational crisis management objectives. These KPIs capture both system-level efficiency and real-world impact on evacuation outcomes, while reflecting the interaction between core components.

Efficiency
Efficiency was assessed in terms of how effectively the system uses the available road capacity and distributes evacuee flows across the network. The experimental results show that the RL policy improves over the heuristic baseline in smaller topologies, outperforming it in 74.31% of the runs for Topology A and 55.17% for Topology B. In these cases, the learned policy also remained closer to the theoretical benchmark than the heuristic, indicating a more efficient use of the available infrastructure.
However, this trend did not extend to the largest topology, where the heuristic remained clearly superior.

Timeliness (System Synchronization)
Measures the latency and synchronization between data ingestion (devices), state generation (UDT), and decision-making (RL inference via MEC). The introduction of the MEC layer enabled low-latency edge processing and faster feedback loops. Near real-time propagation of city-state updates and routing decisions ensured that instructions remained context-aware and up to date, minimizing decision lag and improving responsiveness during rapidly evolving scenarios.

Effectiveness (Successful Evacuation Rate)
Effectiveness is measured by the system's ability to complete the evacuation under the constraints

of the scenario. In the evaluated topologies, the RL agent always managed to learn how to fully evacuate all the simulated people, even if not outperforming the heuristic baseline. Still, in even bigger topologies, or without the computational improvements (see parallelization in training), it has been observed that the training might not correctly converge, leaving unevacuated people.

Rapidity
Rapidity captures how quickly the evacuation progresses from initialization to full clearance. This KPI is directly reflected in the number of time steps required to complete evacuation. The results indicate that rapidity improves under RL control in the smaller and medium scenarios, where the policy more often outperforms the heuristic and achieves a smaller average gap to the theoretical benchmark.

The same advantage is not yet observed in the largest topology, where the learned policy remains substantially farther from the reference optimum.

Safety (Risk Reduction)
Quantified through exposure to hazardous zones and congestion-related risks. The reward design penalizing unsafe routing decisions ensured that the RL agent minimized the number of individuals routed through dangerous areas. As a result, the system achieved a substantial reduction in simulated risk levels, with near-zero exposure in optimal scenarios.

Fairness (Equitable Routing)
Assessed by analyzing the distribution of routing decisions across different areas of the network. Unlike the Dijkstra baseline, which concentrates flow along shortest paths, the RL policy distributes evacuees more evenly across available routes. This prevents localized overload and promotes equitable access to safe paths, reducing the likelihood of disproportionately impacted regions.

Technology Acceptance (EMA / User Interaction)
Evaluated qualitatively through the usability and clarity of device-level instructions generated by the MEC and delivered via EMA interfaces. The system's ability to translate complex RL outputs into simple, actionable guidance (e.g., directional instructions) is critical for user compliance. Initial observations indicate that clear, periodically updated instructions improve user trust and adherence, which are essential for effective real-world deployment.

Summary Insight
Overall, the KPI evaluation confirms that the integration of RL with edge-based MEC infrastructure leads to substantial improvements in evacuation performance. The system not only optimizes efficiency and rapidity but also enhances safety and fairness, while maintaining operational feasibility through real-time responsiveness and user-oriented design.

4.4. Simulation of digital devices

Role and Architecture of the Simulator

The simulator played a central role in the development, integration, and functional validation of the emergency-management platform. It provides a controlled and configurable environment in which realistic crowd-mobility and evacuation scenarios can be reproduced without relying on data collected during real emergency situations. Through the generation and continuous update

of simulated devices, the component supplies the other platform modules with representative mobility data and supports the assessment of their behaviour under different geographical, operational, and population conditions. In particular, the simulator facilitates the integration of the platform components, the verification of data-exchange mechanisms, and the evaluation of emergency-management strategies in repeatable and fully controlled scenarios.

The simulator was developed entirely in Python and designed to be flexible with respect to the geographical area, road-network size, number of simulated devices, initial population distribution, and definition of safe destinations. The urban environment is represented as a directed graph derived from OpenStreetMap data. Nodes correspond to relevant points in the pedestrian network, such as intersections and path endpoints, while edges represent the walkable connections between them. Islands and other areas that cannot be reached through the pedestrian network can be excluded from the simulation domain. This graph-based representation ensures that all simulated movements comply with the actual topology of the considered urban environment.

Each edge of the graph is associated with information required to model pedestrian movement, including its physical length, estimated traversal time, geometry, and capacity. Edge capacity provides an abstract representation of the maximum pedestrian flow that can be assigned to a connection during a given interval. At the same time, each simulated device is represented through an individual state containing relevant mobility information, such as its current position, assigned route, movement direction, speed, destination, and progression along the selected road segment. The simulator updates these states iteratively, thereby reproducing the evolution of the simulated population over time.

The movement and evacuation algorithm is implemented through a heuristic graph-based approach inspired by breadth-first search and the Edmonds–Karp maximum-flow algorithm. Breadth-first exploration is used to identify feasible routes through the network, while the flow-based logic supports the distribution of simulated devices across the available connections according to their capacities. Rather than assigning all devices exclusively to the geometrically shortest path, the algorithm can distribute evacuation demand across multiple admissible routes, helping to represent the effects of limited pedestrian capacity and reducing the generation of unrealistic concentrations on individual road segments.

To include the temporal dimension of pedestrian movement, the underlying road graph can be expanded across discrete time intervals. In this time-expanded representation, a physical node may be represented at multiple consecutive simulation steps. Connections between these temporal instances describe either movement towards another node or waiting at the current location. This allows the routing procedure to account for the fact that traversing different road segments may require different amounts of time and that a device may need to remain temporarily at a location when the capacity of the following segment is unavailable. The approach therefore considers network connectivity, estimated traversal times, waiting actions, route availability, and pedestrian-flow constraints within a common computational representation.

The algorithm determines evacuation paths from populated areas towards designated safe nodes. These destinations can represent emergency exits, collection points, shelters, or other locations considered safe within the scenario. Once a route has been assigned, its graph-level sequence of nodes and edges is translated into incremental movements along the geographical geometry of the road network. This ensures that simulated devices do not move directly between abstract

coordinates but follow the shape of streets, paths, bridges, and other pedestrian-accessible connections represented in the source map.

During every simulation iteration, the component determines which devices must be processed, evaluates their current route and position, computes their next movement, and updates their state. When required by the scenario or by changes in routing conditions, the simulator can assign a new path or update the existing one. The resulting device information is then made available to the other platform components through the defined exchange mechanisms. This iterative workflow enables the platform to observe the evolution of crowd distribution and use the simulated mobility information as input for its monitoring, decision-support, and emergency-management functions.

The simulator was also designed to support repeatable and configurable experimentation. Relevant scenario parameters—including the simulation area, population size, initial distribution, device characteristics, safe-node locations, movement speeds, network capacities, and update intervals—can be adapted according to the objectives of a specific development or validation activity. Consequently, the same simulation framework can be applied to different urban areas and emergency conditions without modifying its fundamental architecture.

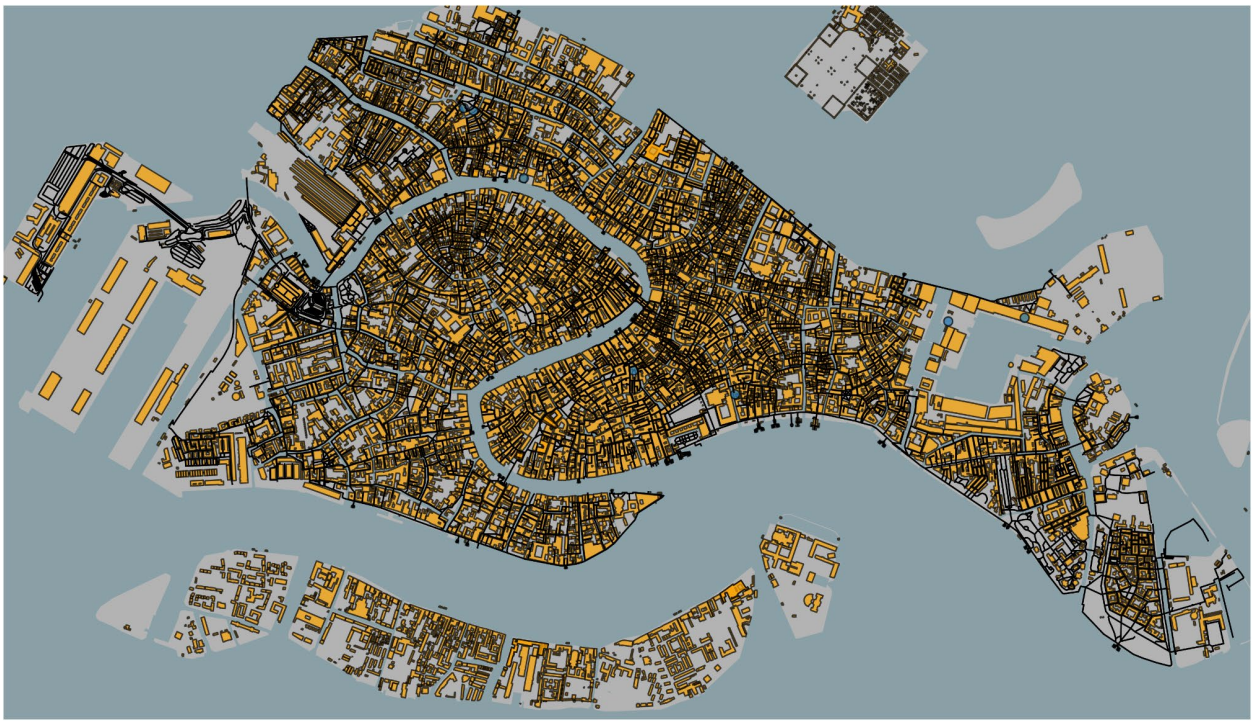
Overall, the simulator provides a realistic and adaptable representation of pedestrian movement within complex urban environments. Its combination of real road-network data, individual device states, heuristic route computation, capacity-aware flow allocation, and time-dependent movement modelling supports the generation of coherent evacuation dynamics. Within the platform, it therefore serves both as a source of representative mobility data and as an integration instrument for developing and validating the interactions among the emergency-management components.

4.4.1 Scalability and Performance (NEW)

Dedicated scalability experiments were conducted to evaluate the simulator under both realistic operational conditions and extreme computational workloads. The reference scenario involved batch of simulated devices from 150,000 to 1.500.00, distributed across the pedestrian-accessible road network of Venice, represented by approximately 5,591 nodes and 14,430 edges. On the internal test platform, the simulator maintained an execution time of approximately 10 seconds per iteration. This measurement covered the complete processing cycle, including device detection, path computation, heuristic next-step calculation, and state updates for every simulated device. Additional stress tests were performed on the same network with populations exceeding one million devices, far beyond the expected operational requirements of the Venice pilot. The results demonstrated the simulator's ability to process large populations over a complex urban network while maintaining stable and predictable performance. They therefore provide evidence of the component's scalability and suitability for both the Venice use case and larger emergency-management scenarios. Further reductions in execution time are expected when the simulator is deployed on dedicated and optimised hardware.

Parameter	Value
Road-network nodes	5,591
Road-network edges	14,340
Hardware / Software configuration	Ubuntu 24, Intel(R) Core i7-10750H @2.6 GH, 20GB RAM

The simulation domain encompassed the main pedestrian-accessible area of the Municipality of Venice. Islands lacking a direct walkable connection to the main urban network were excluded, as no usable pedestrian routes exist between these areas and the remainder of the simulation domain.



Performance indicator	150,000 devices	1,000,000 devices	1,500,000 devices
Total measured time	101.692 s	123.036 s	136.004 s
Mean execution time	**14.527 s**	**15.380 s**	**17.001 s**
Median execution time	14.272 s	15.129 s	16.605 s
Minimum execution time	13.641 s	14.546 s	16.378 s
Maximum execution time	15.893 s	17.606 s	18.191 s
Execution-time range	2.252 s	3.060 s	1.813 s
Standard deviation	0.852 s	0.931 s	0.706 s
Coefficient of variation	5.87%	6.05%	4.15%
Iteration rate	4.13 iter/min	3.90 iter/min	3.53 iter/min
Population increase vs. 150k	Baseline	6.67×	10.00×
Mean-time increase vs. 150k	Baseline	1.06×	1.17×

4.5. Real-World Pilot Validation

During the on-field experimentation campaign carried out in Venice in October, the consortium performed a real-world validation of the main components developed within the project, with particular focus on the **Evacuation Mobile App (EMA)** and its integration with the **Urban Digital Twin (UDT)** platform. EMA is implemented as an Android mobile application designed to support

real-time evacuation guidance during emergency scenarios. The application continuously acquires the user's geolocation through a dedicated positioning module that interfaces with the Android location APIs and the device's native sensors (e.g., GNSS, Wi-Fi positioning, and inertial sensors). This component performs sensor fusion and filtering in order to combine the available positioning sources and generate a refined latitude–longitude coordinate representing the best approximation of the user's current position. These anonymized position updates are periodically transmitted to the UDT infrastructure, enabling the system to maintain a dynamic representation of the spatial distribution of participants in the monitored area.

When an emergency scenario is triggered within the UDT environment, the platform processes the incoming geolocation data from all active EMA clients and computes **personalized evacuation routes** based on the current positions of the users and the simulated emergency conditions. The calculated evacuation paths from the RL model deployed in KServe are then transmitted back to the mobile devices through the **5G-MAC communication component** integrated into the UDT architecture, ensuring low-latency and reliable delivery of guidance information. Upon receiving the route, the EMA application renders the evacuation path directly within its user interface, providing visual navigation instructions that guide the user toward the designated safe area.

The on-field test was conducted during a public event hosted by the **City of Venice**, where a group of volunteer participants tested the first operational version of the EMA application in a controlled but realistic urban environment. Prior to the experiment, technical staff from **Mathema and Logos** provided participants with instructions on the installation and correct use of the application. After the briefing and setup phase, volunteers installed EMA on their personal smartphones and moved outside the event venue to start the trial. Once all participants were connected and actively sharing their positions with the system, the UDT triggered a simulated emergency event. In response, each EMA user received an individualized evacuation route generated by the platform. The participants then followed the routes indicated by the application, navigating through the urban environment toward the designated safe point located in **Piazza San Marco**, thereby validating the end-to-end functionality of the integrated system, including positioning acquisition, real-time data exchange, route computation, and user guidance in a real operational setting.

Following the on-field experimentation, a **roundtable discussion** was organized with the volunteer participants in order to collect qualitative feedback regarding their experience with the EMA application during the evacuation exercise. The discussion focused primarily on **usability, interface clarity, and operational performance** of the mobile application under real urban conditions. Several observations and improvement suggestions emerged from the participants.

One of the main points raised concerned the **visualization of evacuation routes on the map interface**. The first version of the application used a satellite map representation of the surrounding area. While this approach provides a realistic geographic view, several participants reported that it made it difficult to quickly identify the correct path to follow. This issue is particularly relevant in the urban morphology of Venice, where narrow streets, dense building

layouts, and pedestrian pathways are not always clearly distinguishable in satellite imagery. As a result, the visualization of the evacuation route was sometimes less intuitive than expected.

Additional feedback concerned the **usability of the user interface (UI)**. Some participants reported that certain icons and color choices were not immediately intuitive, making it slightly harder to interpret the information displayed by the application during the evacuation scenario. This observation highlights the importance of adopting clear visual semantics and standardized symbols when designing interfaces intended for emergency situations, where rapid interpretation is essential.

Participants also commented on the **communication mechanism between the EMA application and the user**. In the current implementation, system notifications and emergency messages are displayed as pop-up windows directly on the main screen. While this ensures that the message is immediately visible, some users indicated that the interaction model could be improved to make notifications clearer and less intrusive, potentially through more structured alert mechanisms.

Finally, feedback was collected regarding the **accuracy and stability of the geolocation information** during the experiment. Although the positioning system generally provided sufficient accuracy for evacuation guidance, participants observed that the quality of the location estimation was not consistent across all areas of the city. In particular, some locations in Venice presented reduced positioning accuracy due to **weak satellite signal reception and physical obstructions caused by the dense urban environment**, including narrow streets and tall buildings. These conditions occasionally affected the precision of the user position displayed by the application.

Overall, the roundtable discussion provided valuable insights into the **practical usability of the EMA system in a real urban scenario**, highlighting both the strengths of the implemented solution and several aspects that can be improved in future iterations of the application, particularly in terms of map visualization, interface clarity, user notification mechanisms, and positioning robustness.

A **follow-up on-field experimentation campaign** was conducted in February in Venice at the same venue in order to evaluate the improvements introduced in the updated version of the EMA application. The test involved the **same group of volunteer participants** who had previously taken part in the October experimentation, allowing a consistent comparison between the two iterations of the system. The experimental procedure replicated the previous trial in order to maintain comparable conditions. After a short briefing phase, participants installed the updated version of the EMA application and started the evacuation simulation in the area surrounding **Teatro La Fenice**, one of the main points of interest in the city. Once all participants were connected and actively sharing their position with the system, the Urban Digital Twin triggered the emergency scenario and the EMA users received their personalized evacuation routes. The participants then followed the paths provided by the application, navigating through the urban environment toward the designated safe area located in **Piazza San Marco**.

Following the completion of the exercise, a second **roundtable session** was organized to collect feedback from the volunteers regarding the updated application and to assess the effectiveness of the improvements introduced after the first testing campaign. Overall, participants reported a **significant improvement in the usability and clarity of the system**. In particular, the redesigned **map visualization** was positively received: the updated graphical representation of the route and the improved path rendering allowed users to more easily identify the evacuation path to follow within the complex urban structure of Venice. The revised display improved readability and reduced ambiguity when navigating through narrow streets and densely built areas.

Participants also highlighted improvements in the **geopositioning component**. The upgraded positioning module demonstrated increased stability and reliability compared to the previous version, providing more consistent location updates during the experiment. Although some limitations remained due to the intrinsic constraints of the urban environment, the overall positioning performance was perceived as improved by the participants.

Further positive feedback concerned the **updated user interface**, where adjustments to icons, color schemes, and layout improved the overall clarity of the application and made the system behavior easier to understand during the evacuation scenario. Additionally, the redesigned **notification and information delivery mechanism** was considered more effective. The introduction of a clearer and more structured pop-up system improved the communication between the EMA application and the users, ensuring that important instructions and alerts were easier to interpret during the test.

The February experimentation therefore confirmed that the improvements implemented after the initial trial significantly enhanced the **usability, clarity, and operational performance of the EMA system**, demonstrating the effectiveness of the iterative development approach adopted during the project.

The following Key Performance Indicators (KPIs, Table 4.2) were defined to evaluate the performance and reliability of the communication between the EMA mobile application and the 5G-MAC component of the Urban Digital Twin infrastructure during the field experimentation.

Table 4.2: *Target threshold for the KPI*

KPI	Description	Unit	Target Threshold
End-to-End Latency	Time elapsed between the moment the EMA application sends a user position update and the moment the corresponding evacuation route or system response is received back on the device through the 5G-MAC component.	ms	≤ 500 ms
Position Transmission Delay	Time required for the EMA application to transmit a geolocation update from the mobile device to the UDT infrastructure through the 5G-MAC communication layer.	ms	≤ 200 ms
Position Update Frequency	Rate at which the EMA application sends updated user geolocation data to the UDT system during the experiment. This determines how frequently the system refreshes the digital representation of user positions.	updates/sec (Hz)	≥ 1 update/10 seconds
Network Coverage	Percentage of EMA devices that maintain a stable connection with the UDT infrastructure through the 5G-MAC component during the entire field experiment.	%	≥ 95%
Service Availability	Percentage of time during the experiment in which the EMA–5G-MAC communication service is operational and able to successfully exchange data between devices and the UDT platform.	%	≥ 99%

The measured values obtained during the on-field experimentation in Venice, comparing the defined KPI thresholds with the observed system performance during the evacuation simulation:

Table 4.3: *Target threshold for the on-field test results*

KPI	Target Threshold	On-field Test Result
End-to-End Latency	≤ 500 ms	320 ms (average)
Position Transmission Delay	≤ 200 ms	140 ms (average)
Position Update Frequency	≥ 1 update/10 seconds	≥ 1 updates/30sec.
Network Coverage	≥ 95%	96.8 %
Service Availability	≥ 99%	99.2 %

The results obtained during the on-field experimentation (Table 4.3) demonstrate that the communication performance between the **EMA mobile application and the 5G-MAC component** met the defined operational requirements. All the evaluated KPIs remained within the expected thresholds throughout the experiment. In particular, the **end-to-end latency** and **position transmission delay** remained well below the defined limits, ensuring that user position updates and evacuation routes were exchanged in near real-time. The **position update frequency** was also maintained above the required threshold, allowing the Urban Digital Twin to continuously track the spatial distribution of users during the simulation.

It is worth noting that the evaluation of **network coverage and service availability** was conducted under realistic operational conditions within the historic urban environment of Venice. Due to the complex urban morphology of the city—characterized by **narrow streets, dense building structures, and architectural obstacles**—the mobile network signal was not uniformly available across the entire experimentation area. As a result, some sections of the test route experienced **temporary variations in signal strength**, which could potentially affect communication stability between the EMA devices and the UDT infrastructure. Despite these challenges, the system maintained a high level of **service continuity**, and the communication between the EMA application and the 5G-MAC component remained operational for the vast majority of the experiment. This confirms that the proposed architecture is capable of supporting evacuation

guidance services even in **challenging urban environments with non-uniform network coverage**, demonstrating a good level of robustness for real-world deployment scenarios.

The same environmental constraints also affected the **accuracy and stability of the geolocation estimation** during the experiment. In dense urban areas such as Venice, the presence of narrow streets, tall buildings, and architectural obstacles can reduce satellite visibility and degrade GNSS signal quality, leading to fluctuations in the calculated user position. Consequently, the geopositioning performance observed during the field test reflects the challenging characteristics of the urban environment. In more open areas with fewer physical obstructions—such as large squares, parks, or open fields—the availability of satellite signals and network connectivity would likely be significantly improved, resulting in **higher positioning accuracy and more stable location updates**, and therefore better overall performance of the evacuation guidance system.

During the validation of the UDT–SIM–RL loop, the RL component successfully synchronized with the simulation state updates.

- **Inference Performance:** The RL model maintained an average inference latency of 10 milliseconds per routing request, well within the real-time constraints required by the EXTRACT architecture.
- **Stress Behavior:** Under simulated crowd surges where multiple street networks reached maximum capacity simultaneously, the RL agent successfully degraded gracefully, prioritizing the dispersion of evacuees to lower-density holding areas rather than forcing them into deadlocked paths.

4.6. Identified Gaps and refinements

The current evaluation highlights both the progress achieved by the PER approach and the main limitations that still constrain its scalability. In its present form, the RL-based routing strategy performs well on small and medium-scale topologies, but its training stability and generalization degrade as the size and structural complexity of the map increase. In practice, this limitation becomes especially visible once the topology approaches or exceeds roughly 150 nodes, where the growth of the effective state-action space makes convergence slower, more fragile, and more dependent on careful hyperparameter tuning.

Given the current topology limitations, a practical short-term solution is to rely on multiple pretrained versions of the agent, each specialized for a different chunk or district of the city. Under this approach, the urban area is partitioned into manageable sub-topologies that remain within the scale at which training is still feasible, and the corresponding pretrained model is deployed when an emergency occurs in that area. This does not remove the underlying scalability problem, but it offers an operationally viable compromise: training time is reduced, deployment remains fast, and the system can still provide localized decision support without requiring a single monolithic model for the entire city.

A more structured solution would be to move toward a multi-agent formulation. Instead of learning one global controller for the full urban graph, the city could be decomposed into multiple interacting regions, with each agent controlling a bounded subgraph, for example on the order of

50 nodes, and coordinating with neighboring agents through explicit handover or communication mechanisms. This direction is consistent with the broader state of the art in cooperative multi-agent reinforcement learning, where large control problems are decomposed into local agents that exchange information to reduce dimensionality and improve coordination. In traffic-control research, multi-agent settings are commonly modeled as decentralized or partially observable decision processes, and communication between neighboring controllers has been shown to be a central mechanism for improving coordination in large networks.

At the same time, the literature also shows that multi-agent coordination is not a trivial remedy. Cooperative MARL can improve evacuation- and traffic-related outcomes when collaboration is effective, but several studies also report that convergence becomes difficult when coordination and communication are not designed carefully, especially in complex urban settings. This suggests that a city-partitioned multi-agent PER architecture would require not only local agents, but also a robust cooperation mechanism to manage boundary conditions, transfer evacuees from one controlled region to another, and avoid myopic decisions at the interfaces between subgraphs.

A second long-term refinement is to revisit the architectural assumptions of the current learning model itself. The present approach inherits two structural constraints: the value-based nature of DQN-like methods and the need to encode feasible flow allocations into a discretized action space. Although action branching was originally proposed to improve scalability in high-dimensional settings by factorizing the action representation, our experiments indicate that, in this application, the combination of graph size, branching complexity, and constrained flow encoding still leads to a practical scalability ceiling. For that reason, a more fundamental redesign may be required.

Such a redesign could proceed in two directions. The first is to investigate stronger variants of the current family of methods, with improved action representations, exploration schemes, and training curricula. The second, more radical option is to rebuild the approach from the ground up around algorithms that are more naturally suited to continuous, structured, or distributed control. Candidate directions include actor-critic and policy-gradient methods, which avoid some of the limitations of discrete Q-value enumeration, as well as decentralized MARL schemes based on value decomposition, centralized training with decentralized execution, or learned communication between agents.

Overall, the main gaps identified at this stage are therefore threefold: limited exploitation of the broader workflow capabilities defined in the project, insufficient scalability of the current single-agent BDQ formulation for large urban graphs, and the need for more robust coordination mechanisms in city-scale deployments. Addressing these issues will be essential for moving from promising prototype performance in bounded scenarios toward a more scalable and operationally realistic crisis-management system.

4.6.1 Scope of the scalability validation (NEW)

It is important to distinguish the scalability limitation of the current RL policy-learning component from the scalability validation of the PER infrastructure as a whole. The inability to validate a single BDQ policy on the full Venice topology within the project timeframe means that the project cannot claim that the present learning formulation is already suitable for end-to-end city-scale policy optimization. However, this does not invalidate the large-scale results obtained for the remaining components of the platform, which can operate on the larger topology when routing decisions are supplied by a deterministic replacement.

The large-scale experiments should therefore be interpreted as validating the operational scalability of the PER infrastructure independently of the specific policy-learning strategy used to generate routing actions. They demonstrate that the platform can ingest and process large-scale device updates, represent the full urban topology through the UDT, and execute the simulation and heuristic routing workflow at population scales substantially beyond those used for RL validation. What remains unvalidated at Venice scale is specifically the training and generalisation of the current monolithic BDQ controller

Future work should consequently treat the present BDQ implementation as a partial solution for bounded scenarios rather than as the final city-scale solution. A staged path forward is to deploy district-specific pretrained agents in the short term, investigate cooperative multi-agent control over communicating subgraphs as the structural solution, and evaluate continuous-control approaches such as actor-critic or policy-gradient methods as an algorithmic redesign. These directions would retain the validated PER infrastructure while replacing or extending the policy-learning component that currently limits full-city deployment.

Beyond the real-world validation carried out in Venice, the proposed architecture is intended to serve as a reusable framework for AI-supported urban emergency management. Its modular design, together with the demonstrated scalability of the underlying infrastructure and the use of open standards, facilitates adaptation to other cities facing similar challenges related to pedestrian evacuation, mass gatherings, climate-related emergencies, and critical infrastructure protection. The Venice deployment therefore serves not only as a local pilot but also as a reference implementation for the adoption of the PER framework in other complex urban environments.

5. Evaluation of the TASKA Use-Case

5.1. TASKA Objectives and Final Architecture Reference

The final description of the TASKA use case was fully described in D1.3. Here is a summary of the final overall structure for the 3 main components of TASKA: use cases **A**, **C** and **D** described below.

5.1.1. Final overall structure of use-case A

Since the previous release (D1.3), new developments have significantly advanced the capabilities for real-time detection of radio transients from astronomical sources such as the Sun and Jupiter, using the NenuFAR radio telescope. At the core of these improvements is the **Modular Multicast Receiver** (MurMuRe), developed at Observatoire de Paris. MurMuRe leverages the CuPy library for efficient memory management between CPU and GPU, enabling real-time computation on streaming telescope data.

MurMuRe was developed as a containerized, GPU-enabled, modular processing framework for radio astronomy, with the explicit goal of interconnecting real-time processing stages operating on high-resolution dynamic spectra. Its architecture is based on independent processing blocks

that can be assembled like LEGO blocks to build an observation-specific chain, making it possible to combine, replace, or extend individual scientific operations without redesigning the whole system. In practice, MurMuRe already supports a wide range of processing steps such as voltage-to-complex conversion, data reshaping and transposition, FFT and inverse FFT, conversion of complex voltages into cross-products and Stokes-related products, time and frequency integration, bandpass correction, RFI cleaning, rechunking, visualization, and scientific writing to exchange or archive formats. A simple example of a MurMuRe pipeline is shown in Figure 7.1, where processor blocks are chained from acquisition to scientific output. In operational deployments, MurMuRe itself runs in dedicated GPU-enabled containers, alongside the containerized UDP acquisition layer, so that the complete TASKA chain can be deployed as a coherent containerized processing stack. Starting from this foundation, MurMuRe evolved into the common modular backbone used throughout TASKA to connect data acquisition, online processing, replay, storage, and validation. Beyond TASKA itself, the same framework is reused on NenuFAR, LOFAR FR606, and the Nancay Radio Telescope, confirming its role as the final reference architecture for modular real-time radio processing within this use case.

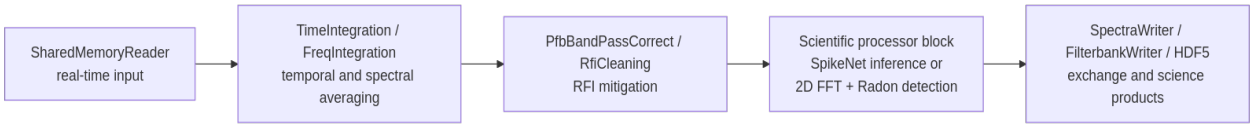


Figure 5.1: *Simple example of a MurMuRe pipeline*

Objectives

TASKA Use-case A is structured as two scientific branches (A1 and A2) built on the same MurMuRe architecture and sharing the same acquisition, preprocessing, execution, and storage principles.

Both branches therefore rely on a shared architectural principle: MurMuRe is the common orchestration and data-processing framework, while the science-specific stages are plugged in as specialized processing branches. This allows TASKA Use-case A to develop, validate, and deploy both pipelines within one coherent technical environment, with identical support for real-time execution, offline replay, and testing on simulated or archived observations.

TASKA Use-case A1: SpikeNet for Solar Radio Spike Detection

The objective of TASKA Use-case-A1 is to perform real-time convolutional neural network inference on high-resolution dynamic spectra in order to detect radio spikes. In the final TASKA architecture, MurMuRe provides the acquisition and preprocessing chain, while the SpikeNet branch performs GPU-accelerated inference and writes science-ready detection products for subsequent validation and analysis.

SpikeNet is a neural network, based on the U-NET architecture, specifically adapted for integration into the NenuFAR data workflow and back-ends. Its main purpose is to detect short-duration radio spikes in solar observations efficiently and automatically.

- **Real-time Processing**: SpikeNet was trained on labeled spectra to infer the presence and spatio-temporal location of spikes in large volumes of data, overcoming the impracticality of manual labelling.
- **Data Compression:** Applied to real-time solar observations, it enables a 10× reduction in data volume. High-resolution data is preserved only when spikes are detected by SpikeNet;

otherwise, MurMuRe averages and compresses the recordings, still allowing analysis of slowly varying emissions.

- **Output**: The detections and their physical properties are saved in HDF5 files as sets of tiles representing the events.

TASKA Use-case A2: Detection and Morphology Analysis of Jovian Bursts

The objective of TASKA Use-case-A2 is to detect fast-drifting radio bursts in Jupiter's decametric emissions in real time. In the final TASKA architecture, MurMuRe provides the same acquisition and preprocessing backbone, while the Jovian burst detection branch applies dedicated RFI mitigation and pattern-detection stages to derive drift, signal-to-noise ratio, and burst characterization products.

Building on methods developed during a PhD (E. Mauduit) for the Nançay Decameter Array, TASKA Use-case-A2 adapts these techniques for NenuFAR's Jupiter observations.

- **Algorithm Conversion**: The initial algorithm in IDL was converted to Python for seamless integration into NenuFAR's data pipeline.
- **Detection Pipeline**:
  - RFI Mitigation: Thresholding is used to remove Radio Frequency Interference (RFI), ensuring that remaining features are dominated by Jovian emissions.
  - 2D FFT: Used to highlight quasi-periodic drifting structures in the time-frequency domain.
  - Radon Transform: Applied to the FFT results to detect the characteristic drift of S-bursts, quantifying their physical parameters (drift rate, duration, frequency localization).
- **Current Status and Machine Learning**: The algorithm is being validated and applied to existing NenuFAR Jupiter observations, both for transient detection and to build a labeled database for training a neural network. Due to the diversity of burst morphologies, anomaly detection will be used rather than classification.

Outputs will also include burst locations and physical parameters, formatted as HDF5 files.

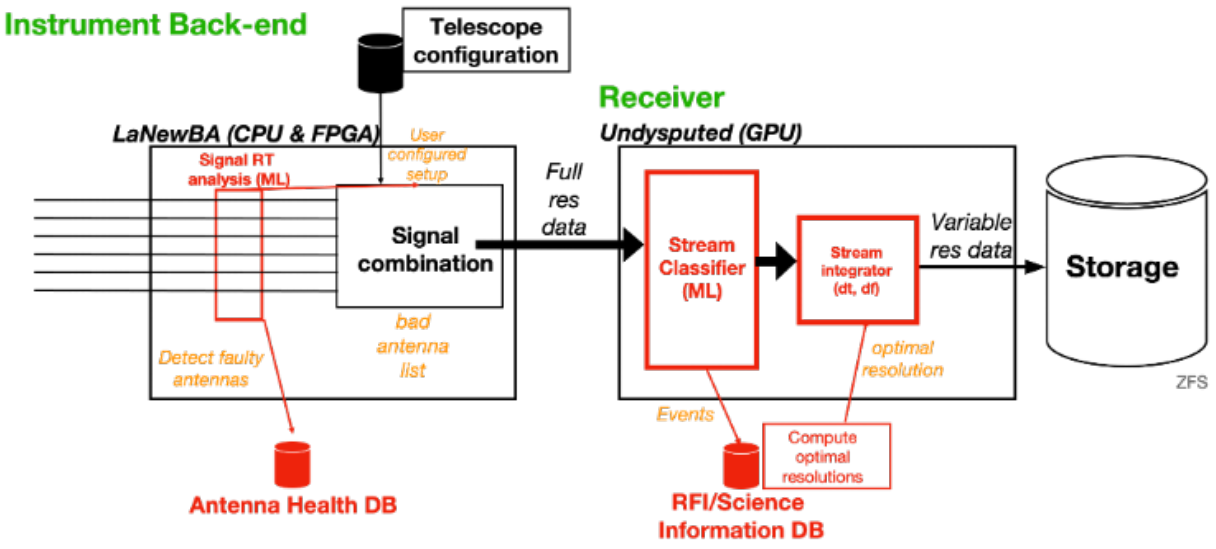


Figure 5.2: *Structure of the signal flow and insertion of the MurMure in UnDySPuTeD receiver as an inference component.*

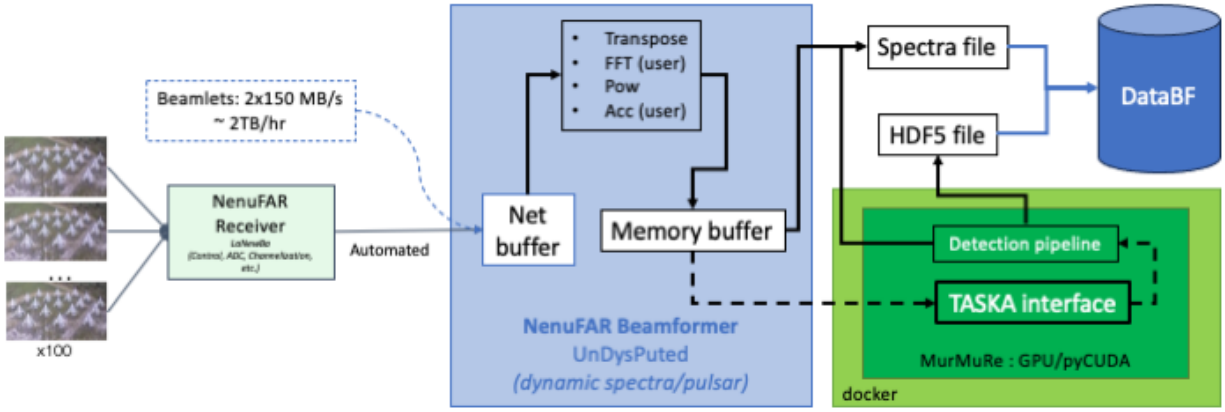


Figure 5.3: *Detailed diagram of the MurMure component included in the UnDySPuTeD receiver*

MurMuRe was integrated into the NenuFAR/UnDySPuTeD-TF interface, enabling efficient CPU/GPU memory management and real-time inference (see Figures 5.2 & 3). The EXTRACT TASKA Use-case-A1 was inserted between the memory buffer and accumulator, allowing dynamic adjustments in data accumulation factors (time and frequency) based on detection outputs.

This instrumental upgrade resulted in up to 90% data volume reduction while preserving scientific content (see next sections).

Final Architecture Reference

The final TASKA reference architecture is a common end-to-end chain in which a containerized UDP receiver writes the incoming stream into shared memory, and a containerized MurMuRe-based TASKA service consumes this stream and executes either the TASKA Use-case-A1 or the TASKA Use-case-A2 science branch. In this architecture, `.spectra` and `filterbank` files play a central role as a common exchange format between real-time execution and offline replay, while HDF5 files provide persistent storage for downstream scientific exploitation.

Taken together, TASKA Use-case-A1 and TASKA Use-case-A2 are two sibling branches of the same final MurMuRe-centered architecture: a common acquisition and preprocessing framework, coupled with branch-specific science processors and a shared output strategy that bridges real-time operation and offline validation.

A new science branch (see Figure 5.4) that would be created by an astronomer could be integrated in this framework in a straightforward way within the existing architecture.

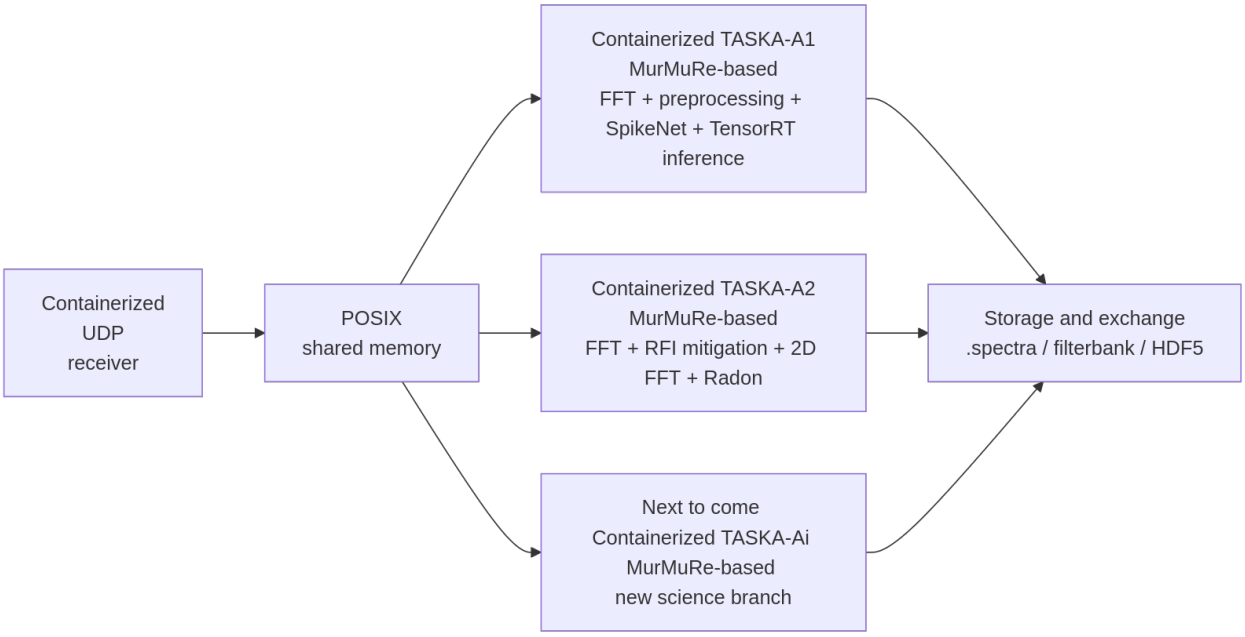


Figure 5.4: TASKA Use-case-A1 and TASKA Use-case-A2 sibling branches and a potential future new science branch

Use-case A1 reference branch

In TASKA Use-case-A1, a containerized MurMuRe service ingests the stream from *SharedMemoryReader* in real time, or from replay readers during offline analysis. The data then passes through a signal-processing chain composed of signal conversion, FFT-based transformations, cross-products, time integration, rechunking, and bandpass correction before being passed to *SpikeNetInference*. The output of this branch is written as HDF5 detection products together with *.spectra* outputs that support compression, replay, comparison, and post-run validation under the same framework.

Use-case A2 reference branch

In TASKA Use-case-A2, a containerized MurMuRe service provides the same ingestion and preprocessing backbone and delivers dynamic-spectrum products to the Jovian burst detection branch. The science processing then applies slicing and rebinning into square dynamic-spectrum cubes, instrument-aware RFI mitigation, 2D FFT, Radon transform, Gaussian peak identification, and drift/SNR estimation. The output of this branch is written as HDF5 detection products designed for scientific inspection, emission analysis, and downstream classification-oriented post-processing.

Infrastructure

The TASKA-A final architecture is built on a small number of core technologies that together provide real-time acquisition, GPU-enabled processing, scientific inference and detection, and persistent exchange formats. The most important technologies are listed below.

Radio-astronomy observation systems
NenuFAR, LOFAR FR606, and the Nancay Radio Telescope provide the live or recorded data sources used by the TASKA workflows.

Docker: Docker is used to package both the UDP multicast receiver and the MurMuRe-based TASKA services as reproducible containers.

Docker Compose: Docker Compose is used to deploy and coordinate the acquisition and processing containers as an operational stack.

C: The real-time UDP multicast receiver is implemented in C for efficient packet capture and buffering.

POSIX shared memory: Shared memory is used as the real-time transport layer between the acquisition container and the MurMuRe processing containers.

Python: MurMuRe and the TASKA processing services are implemented in Python, which provides the main orchestration, preprocessing, replay, and scientific-processing environment.

MurMuRe modular processing framework: MurMuRe is the core software technology of the infrastructure, built around "ProcessManager", "PipelineProcessor", and modular processor blocks that can be assembled like LEGO components. Its processor library already supports voltage-to-complex conversion, reshaping and transposition, FFT and inverse FFT, conversion from complex voltages to cross-products, Stokes-oriented visualization, time integration, frequency integration, bandpass correction, RFI cleaning, rechunking, concatenation, visualization, and scientific output writing.

CUDA: CUDA provides the GPU execution layer used by the processing containers.

CuPy: CuPy is used inside MurMuRe to accelerate array processing and numerical operations on NVIDIA GPUs.

TensorRT: TensorRT is used in TaskA-a1 to execute SpikeNet inference with high throughput on GPUs.

TASKA Use-case-A1 SpikeNet chain: SpikeNet is the TASKA Use-case-A1 neural-network technology used for radio-spike detection on high-resolution dynamic spectra.

TASKA Use-case-A2 burst-detection chain: The TASKA Use-case-A2 scientific processing stack is based on MurMuRe preprocessing followed by RFI mitigation, 2D FFT, Radon transform, Gaussian fitting, peak identification, and drift/SNR estimation.

***.spectra* data format**: The *.spectra* data format is the main MurMuRe exchange and replay format bridging real-time operation and offline reprocessing.

Filterbank / SIGPROC-compatible outputs: Filterbank products are available when interoperability with external radio-processing tools is required.

HDF5: HDF5 is used to store higher-level science products, detection parameters, and validation outputs for both TASKA branches.

Visualization and analysis tools: Visualization modules, replay tools, output readers, and downstream analysis scripts are used to inspect, validate, and compare results on live, archived, or simulated observations.

In this infrastructure, MurMuRe is the reference integration technology that ties together containerized acquisition, shared-memory transport, GPU processing, scientific application services, and replay/storage workflows. This is what enables TASKA Use-case-A1 and TASKA Use-case-A2 to operate as two peer scientific pipelines within one common operational architecture.

5.1.2. Final overall structure of use-case C

As the project reaches its conclusion, Use-case C stands as a robust, dynamic framework tailored for radio interferometric data reduction, with NenuFAR as the primary application case. This final release documents the matured specifications and the successful deployment of the system, which enables scientists to easily define, control, and reproduce scientific workflows for processing instrument-acquired datasets.

Summary of the main implementations with D1.3

During the last stage of the project, the last features requested by the specification have been in the implemented for use case C:

- **Implementation of the data catalogue**: for data ingestion and data traceability during the workflow.

The remaining missing block in D1.3 was the implementation of the data catalogue to track data ingestion and data processing using Nuvla. A dedicated software package was developed to ensure a correct registration of all activities and data at each step of the workflow in a non-invasive manner while staying completely invisible to the scientific end-user. The end-user should only care about the workflow composition and monitoring and all the logging tasks and data production should be handled behind the scenes.

- **Implementation of workflow human interactions**: suspension/resuming, looping capability, workflow branch replay.

In order to implement complex data processing workflows on heavy data sets, the workflow building should be flexible enough to implement a high degree of control for operators and safe stopping mid-workflow. The complexity and heterogeneity of workflow runs located in various places invite to reduce workflow housekeeping and reduce human cognitive load. Amongst the required features that COMPSS should provide:

I) branching capability: create multiple branches running in parallel and barrier to merge branches,

II) arbitrary workflow suspension barriers placed by the workflow editor to suspend the workflow and send a trigger to the operator to ask for approval before workflow continuation,

III) Conditional test barriers: logical test defined by the scientist and computed on the workflow intermediary results (test on quality criterion, e.g. image residual quality, convergence, existence of a file, etc.) and

IV) workflow looping capabilities: the workflow architecture has to be adapted to each scientific applications but in some case, the exact number of tasks and tasks sequence is not known in advance (e.g. multiple imaging of the data) and a workflow looping capability is required to perform multiple instance of the same task or task sequence. In interferometric imaging, a “self-calibration” loop can be implemented with successive application of calibration and imaging until convergence is met. The loop should be exited based on a logical test depending on the quality

criteria (e.g. the imaging residuals after ~10 self-calibration loops). These features have been included in the current version of COMPSS and can now be used to define a complex and dynamic workflow based on the scientific integrity and quality of the intermediary products. This way, the workflow is fully science quality driven.

- **Multiple Object storage back-end support**:

Compared to the PER use case, SkyStore is not explicitly used in TASKA, however the need for multiple object storage access is required. These storage can be different in nature (i.e. back-end type) and also in different locations (e.g. 33% of the data in server#1, 33% in server#2, 33% in server #3). The workflow should be able to distribute the list of paths to each file even if they are distributed in various sites. The list of object storage locations and buckets containing the actual data is stored in the data catalogue during the data ingestion phase. Each task will interrogate the data catalogue to know where the data are located and where to store the results.

- **Standalone EXTRACT platform for developers**:

Thinking about the future after the end of the project, we formulated the need that the platform should be ready to be handed to new developers and applied to new scientific teams outside of the current use case. In particular, the upcoming Square Kilometre Array is currently defining the solutions for their Regional Centers which are constituted of HPC/Cloud/Storage infrastructure that will host the data incoming from the telescope. NenuFAR is a pathfinder of the future SKA-low and has served during the EXTRACT project as a demonstrator. Therefore, partners have pushed an extra effort in preparing the development testbed of the framework. The current version can be deployed for production but need also be ready and documented for future implementation and improvements. For this purpose, a stand-alone version of the whole framework has been prepared to reproduce the whole TASKA environment on a single laptop (i.e. Docker in a docker image mimicking a k8s cluster, multiple object storage, local client, etc.). This portable version is strategic for demonstrating the capabilities of the EXTRACT platform in a “full offline” version. Future developments are therefore made easy because it does not rely on the usage or connection to real computing facilities for which credentials can slow down development.

- **(extra) Facilitating even more the workflow composition with a GUI:**

As a bonus KPI, some extra tasks were given to two engineering students as an outreach project. These tasks were not planned early in the project but could add an extra simplification layer for the scientific user, to compose and test workflows. It consists of a customised interactive graphical user interface that helps compose workflows and export them as YAML files. The students have generated a workflow on a simple processing use case (filtering sound) but have produced a package that helps visualise a workflow and interact with it. This lays ground to future professional implementation of a GUI, compatible with the use case C workflow inside the EXTRACT framework (see Figure 5.5 and section 7.4 for details).

Based on all features now present in use case C, the following is a detailed summary of all features of the final release that were met for use case C.

1. Data Acquisition, Ingestion, and Storage

- **Edge Data Production:** Datasets are generated at the instrument level in the standardized Measurement Set (MS) format, often comprising multiple files per observation.

- **Metadata and Provenance:** Upon ingestion, data are organized within a dedicated object storage and annotated with comprehensive scientific and technical metadata, facilitating downstream processing and reproducibility in accordance with IVOA provenance standards. The technology uses Nuvla API to connect with the data catalogue for both data ingestion and data provenance.
- **Distributed Object Storage:** The system treats files as abstract objects, accessible as a unified entity irrespective of their size, number, or physical location, thus supporting transparent access and efficient processing. SkyStore was first considered before relaxing its use for a more direct, easier data connection framework (Dataplug).

2. Advanced Workflow Orchestration

- **Task-driven Architecture:** Data reduction pipelines are composed of modular tasks (e.g., compression, calibration, imaging), flexible in sequencing and tailored to user-defined needs and control.
- **Automated Execution, communication and workflow control**: Workflows operate seamlessly from ingestion to final products, requiring minimal intervention. Nevertheless, new features include more control over the workflow. The users now have the ability to arbitrarily pause and interact with the workflow to inspect intermediate results, modify parameters, and restart tasks as needed (as depicted in Figure 5.6). The workflow control can also perform conditional loops until some condition is met (based on the result of a user script that assesses the quality of intermediary products, see Figure 5.7). The workflow can restart from previous nodes of the workflow graph. Communication with the workflow is done with MQTT.
- **Technological Integration:** COMPSS and Lithops are employed for distributed workflow creation, software encapsulation, efficient resource management, and optimised partitioning of tasks occurring behind the scenes. The data catalogue is fed using a new Nuvla decorators library, which is inserted in the workflow code and is invisible from the end-user. Communication with the data catalogue is vital as it stores the location and metadata of the files to process as well as registering new data with metadata respecting Provenance principles.

3. Scalability and Efficiency

- **Cloud-based Robustness**: The platform supports datasets that range from hundreds of megabytes to several terabytes, utilising efficient partitioning, parallelisation, and dynamic computing task allocation to minimise reduction time and energy consumption. Stress tests on heavy data throughput in a fully deployed environment need to be performed.
- **Optimized Resource Usage:** Workflow urgency levels can be set, enabling low priority tasks to operate with reduced energy expenditure and high priority tasks to scale as needed. The D2.4, D3.4 supplies the tools and metrics that decide which scenario to follow based on the GPU/CPU/IO/RAM needs.

4. Scientific Output and Provenance

- **Products and Value:** The workflow produces both intermediary and final scientific products, logged throughout for full traceability and scientific integrity.

- **Comprehensive Provenance Logging**: All processing steps, code versions, input parameters, and transformations are meticulously recorded and attached to the dataset, supporting reproducibility and peer validation.
- **Metadata Catalogue**: Results and their accompanying provenance metadata are stored in a centralised catalogue, facilitating advanced search and data retrieval based on scientific and technical parameters.

5. User Experience and Workflow Composition

- **Interactive Design**: Users may design or select workflows using intuitive interfaces, such as Python notebooks, scripts, or dedicated GUIs (including one developed in the scope of an engineering student), making it straightforward to construct custom or recipe-based processing chains.
- **Quality Assurance**: The system supports iterative task execution, allowing workflows to loop until data quality requirements are met.

6. System Architecture and Extensibility to other science cases

- **Edge-to-Cloud Orchestration:** The platform provides seamless interaction between edge data production (instrument), cloud storage, and workflow management, ensuring efficiency and transparency in radio astronomical data reduction.
- **Extensible Task Catalogue and adaptation outside EXTRACT:** The modular task catalogue adapts to evolving scientific requirements, enabling continuous framework enhancement. The source code is organised in a way that help invested use to pursue the development by inserting new tasks and possibly completely changing the sky, the source and the imaging.

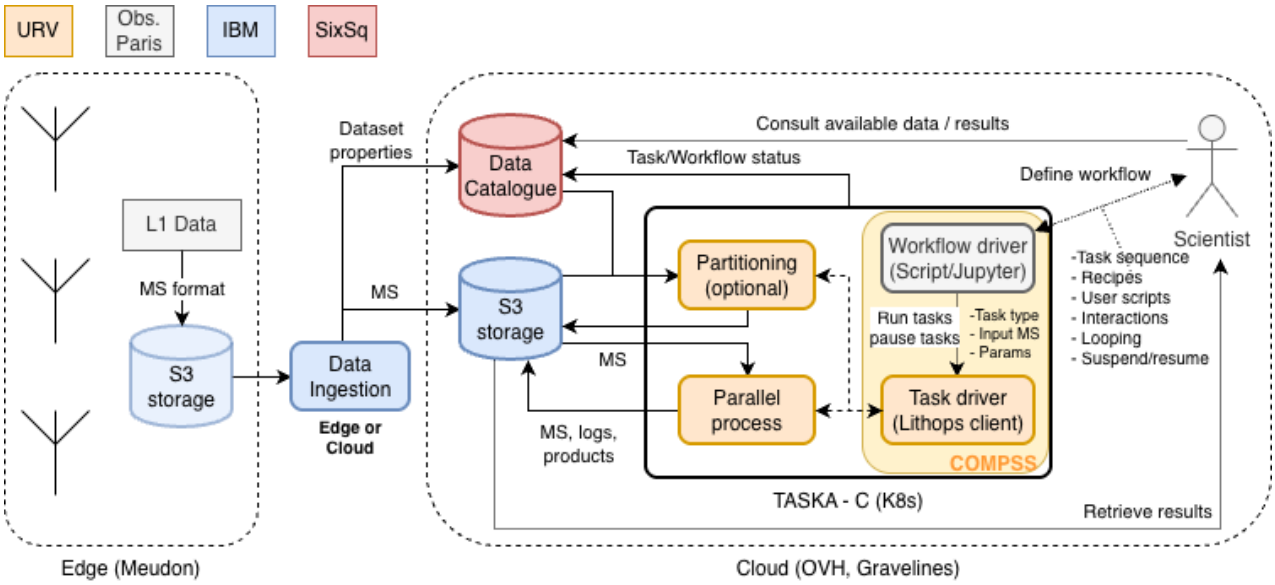


Figure 5.5: *Main view of the edge/cloud interaction system for data reduction orchestration in use-case C. See D3.4 for details on the architecture.*

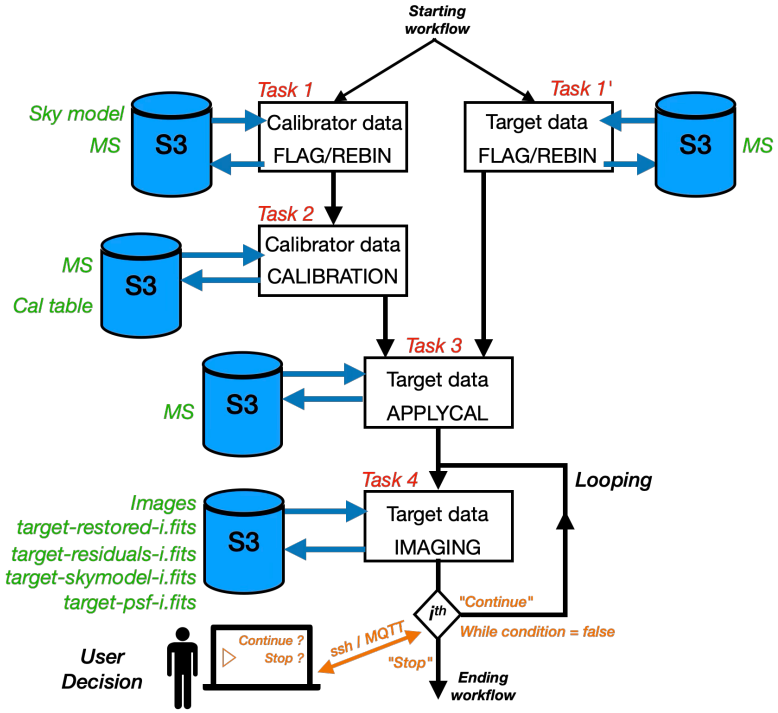


Figure 5.6: *Complex workflow with looping capability. Task #1, #1', #2 and #3 are parallel but linear. Task #3 will run depending on the success of tasks #2 and #1'. After task #3, the workflow enter a loop to perform an undefined number of runs of task #4 (imaging loop). The success of the loop is conditioned by a test on imaging output at iteration #i. The condition can be defined by a quality criterion on a given output at iteration #i automatically or after human approval through MQTT messaging.*

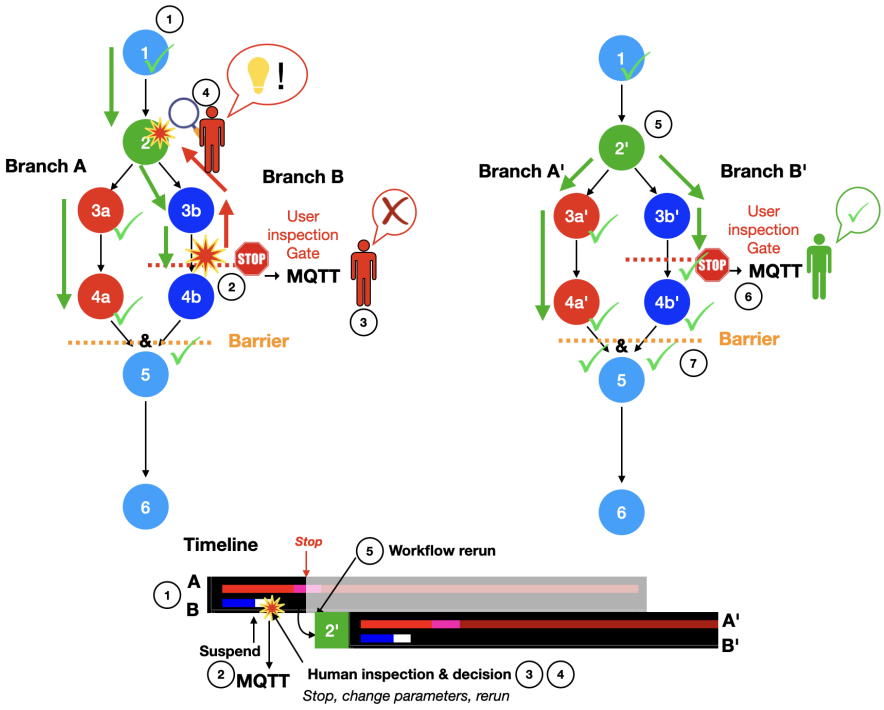


Figure 5.7: *Complex workflow with workflow control and rerun capability. (1) the workflow has started with task #1, then divide into two branches "A" and "B". While branch A takes longer than B, B advances to task #3b. (2) After task 3b, a gate was placed to hold the workflow and send a message to a user through MQTT. (3) User receives message and go inspect the output of task #3b. The user notice that the output is not correct and investigate the origins of the issue up to task #2. (4) the issue has been found and the decision is to stop the current workflow run (both branches) and start task #2 again with corrected inputs. (5) A new timeline is created (task #2' with branches A' and B'). (6) the*

branch B is stopped after task #3b' and a message is sent to the user for verification. The user approved the result and the workflow continue until the end of branches A' and B'. (7) Task #5 depends on the success of branch 4a' and 4b', since both branch have generated positive results, the workflow continues with task #5 and #6.

5.1.3. Final overall structure of use-case D

Use case D results go beyond expectations as this new imager was developed on a best-effort basis. Early results have demonstrated the feasibility of the method as well as the robustness and flexibility of the source code, making it applicable beyond EXTRACT and beyond NenuFAR. This new stand-alone imager can play a major role in the community by setting a new standard in dynamic imaging involving deep learning networks.

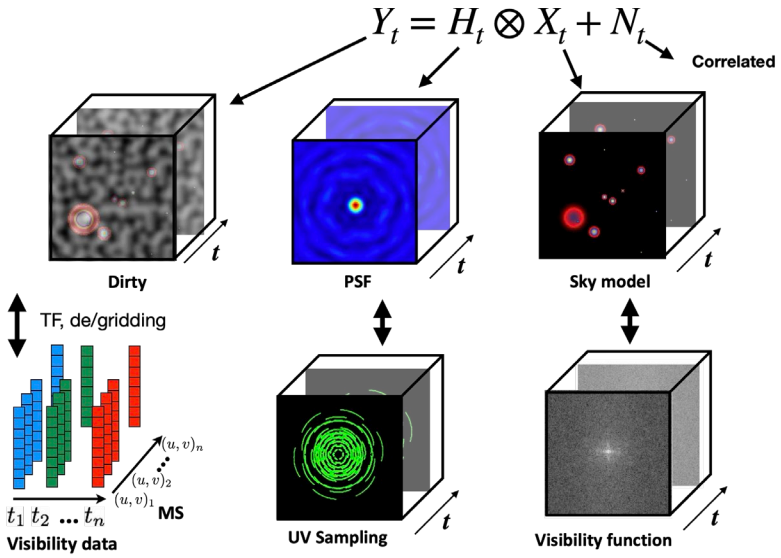


Figure 5.8: *Data model of use case D imager. The visibility data (Fourier samples) taken by the radio interferometers are stored as listing for each time and each baseline. Its Fourier transform is a raw image of the sky (Dirty). The measured sky image is the result of the convolution of the instrumental Point Spread Function (PSF) convolved to the true sky image. This convolution by the PSF is equivalent to the multiplication of the interferometer sampling function to the true visibility function.*

The developed package is organized as follow:

- **Imager:** a python package that allows radio image gridding and restoration. The modularity of this solution allows the design and use of an imager by combining a gridder object with a restorer object. The input of the imager is an MS object and the output is a FITS file. The imager has the capability of restoring transient sources using the Deflation Net model. It can be directly called from the CLI.
- **Astrogym:** a python package that allows the training and evaluation of radio imagers. The training is configurable using a YAML file. The data is loaded in webdataset format which is standard in the Deep Learning community. The trained model is saved with the corresponding code and configuration files for reproducibility purposes. For the evaluation, there are CLEAN scripts to generate CLEAN outputs for comparison. Advanced filters have been implemented to have a specific data evaluation, e.g. transient source only.
- **DataSim:** a python package for realistic radio data simulation. The solution is able to simulate point sources with various transient profiles (Gaussian, skewed). The data can be stored in MS format which is adapted for evaluation, or webdataset format which is adapted for Deep Learning. The simulation uses existing MS files such that the UVW coverage comes from real data. The noising and blurring processes are implemented using

exact physical formulas. The code has a CLI and uses YAML config files. The generated data is saved with the corresponding code and configuration files for reproducibility purposes.

- **MS-handler:** a python package that serves as a toolbox for MS file handling. It allows reading, writing and modifying MS files. It also allows preprocessing raw data and plotting it.

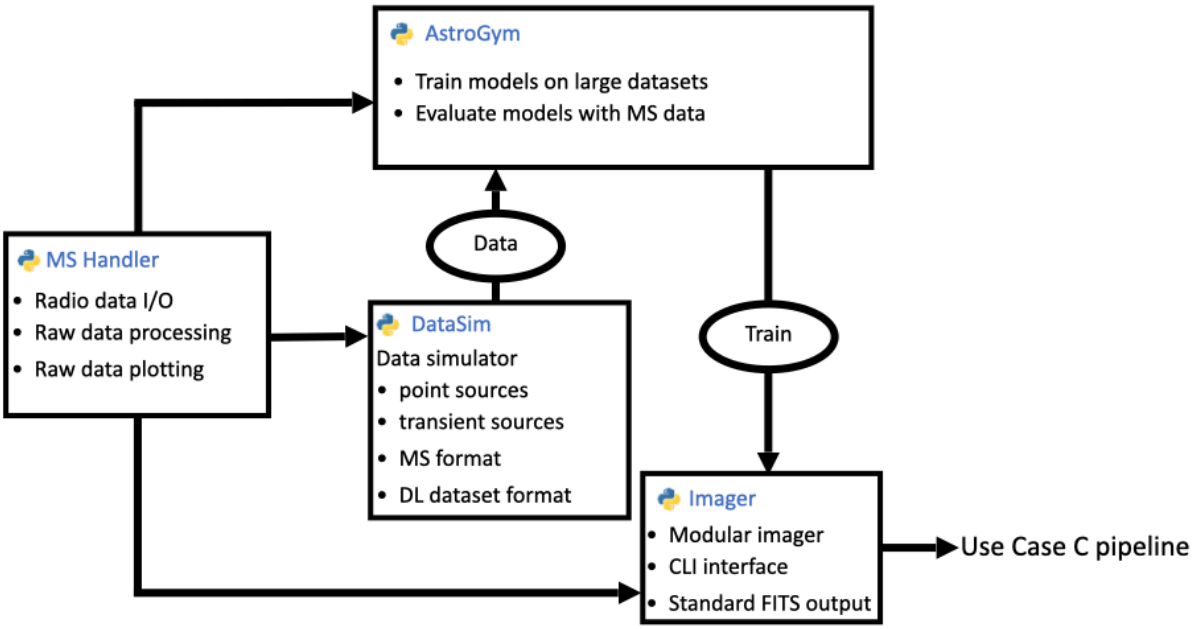


Figure 5.9: *Block diagram of the TASKA-D software package including the MS Handler library to handle MS format and I/O, DataSim to generate sky models, training data sets handling MS format. Astrogym is able to train the neural network from ground truth and generated data. The trained weight can then be used in the scope of the Imager which is a stand-alone application with a CLI and various gridders and restorers. It generates astronomical outputs as FITS image or cubes. The imager part can be inserted as a new imager directly in a use case C workflow.*

Connection with use case C

Thanks to the training environment provided by Astrogym, the imager part is only used for inference using the trained weights and network structures. This new imager can directly replace the current imager used in use case C (WSclean) as both tools can be called directly from terminal via a dedicated CLI with overridable options.

5.2. Baseline Comparison

The baseline comparison (Table 5.1) is done using a manual data reduction or a linear data reduction script performed on a single node with multiple CPUs as it was done before EXTRACT on the Nançay Radioastronomy Observatory. These numbers are average values on typical dataset measures with NenuFAR radiotelescope observing at full resolution (with (dt,df), A1/A2: (5ns,3kHz) & C/D:(1s,3kHz)) and for all available frequency channels of NenuFAR (2 x 244 192-kHz subbands).

Table 5.1: *Baseline comparison if none of the components developed in Use Case A1/A2, C and D were used*

Use case	Processing over observation (P/O)	Data delivery time	Scientific quality	Optimal usage of resources	Level of human intervention
Without use case A1 & A2 solution	~1 day / 3 hr observation	~2-3 days	Event selection biased by human	No	High and subject to error
Without use case C solution	~30min / 3 hr observation	1 day	Good	Thread parallelisation*	Medium when scripting is used
Without use case D solution	1h / 3h observation	~2-3 days	Low for data containing transients sources	Thread parallelisation on a single compute node	High level of human bias in the detection

*on a single compute node at Nançay Radioastronomy Observatory

5.3. KPI-Driven Evaluation of TASKA

5.3.1. Evaluation methodology

The evaluation of the TASKA framework relies on a combination of quantitative performance metrics and qualitative scientific assessments in order to capture both the technical efficiency of the platform (in its final design) and the scientific value of the produced results (some are still under refinement in preparation of corresponding scientific paper). Because the project addresses heterogeneous “sub” use cases (A, C and D), from real-time edge processing to large-scale interferometric workflows and deep-learning-based imaging, the evaluation methodology has been designed to remain comparable across cases while accounting for their specific operational constraints.

From a quantitative perspective, several Key Performance Indicators (KPIs) have been defined to measure the efficiency of the data processing infrastructure and workflow execution. These include metrics such as the Processing-over-Observation (P/O) ratio, which evaluates how quickly data products become available after observation, selective storage reduction factors resulting from intelligent data filtering or compression, and improvements in resource utilization and scheduling efficiency enabled by dynamic orchestration frameworks such as Lithops and COMPSs. These indicators allow the project to measure concrete improvements in data throughput, processing latency, and storage footprint compared to more traditional processing approaches.

Complementing these measurements, the evaluation also considers operational performance indicators, such as the timeliness of science data delivery and the ability of the infrastructure to support automated or near-real-time processing workflows. These metrics reflect how effectively the platform supports scientific operations, particularly in contexts where rapid analysis and data availability are critical. In addition to numerical performance indicators, qualitative evaluation criteria are used to assess the scientific integrity and usability of the produced data products. This includes verifying that the adoption of distributed workflows, data partitioning, and automated

processing does not degrade the scientific quality of the results. For some use cases, this evaluation also considers the behaviour of machine-learning models used for feature detection or imaging, assessing their capability to identify relevant structures in the data and their robustness with respect to observational variability. Together, these quantitative and qualitative evaluation dimensions provide a comprehensive framework for assessing the overall impact of the TASKA architecture. The following sections present the results of this evaluation for each use case, focusing on processing efficiency, data management improvements, workflow orchestration capabilities, and the scientific quality of the generated products.

5.3.2. KPI Evaluation summary for TASKA

The following table regroups a summary of the obtained KPI results in the scope of TASKA. The metrics are covering the various TASKA cases A, C and D (See the details in each KPI sections).

5.3.2.1 Scope of the TASKA use cases evaluation (NEW)

The TASKA use-case, was designed as a truly high volume use-case to validate the following characteristics:

- **Volume and velocity:** Telescopes generate up to 8 TB every hour of raw data. This data must be reduced and processed in less than 1 hour to make room for other observations.
- **Dispersed data storage***:* The data generated by the telescope is then stored on different storage locations in which science products are generated. In addition, the computing capabilities are not identical on each site.
- **Sparse and insufficient data, and extreme variations in values:** In each TASKA use case, data and information is sparse with extreme variations due to the transient nature of the events under analysis.

For TASKA, the EXTRACT project aimed to reach TRL 6 developments and results.

In the TASKA-A case, only the information related to the event must be kept at maximum resolution, the rest of information can be compressed.
In the TASKA-C and D cases, the information is sparsely sampled (inherent to the interferometry instrumentation) and the data is distributed. The TASKA-C and D pipelines are aiming at reconstructing the astrophysical information from this very sparse sampling.
The TASKA use-case allowed the EXTRACT platform capacity to deploy workflows across multiple cloud sites, supporting parallel computing in it. Before EXTRACT, only a portion of this data was considered for science as it had to be manually reduced and processed by humans.

EXTRACT enabled to automatically reduce and process 100 GB of data in a single site in less than 30 minutes. To do so, the consortium hired resources from OVH Cloud including: 32 virtual cores, 256 GB of RAM, 400GB of S3 storage and 8Gb/s of network. The project also considered 2 different sites (Meudon and URV facilities with similar characteristics) to process up to 600 GB of data in less than an hour. It is important to remark that this is not a platform limitation but an infrastructure/budget limitation.

This use-case exercised the following main platform components:

- COMPSs supported coarse-grain workflow parallelisation, supporting event-driven models to start/stop/resume workflow if certain conditions are not accomplished. This allowed to safe significant amount of useless computation.
- Lithops, integrated within COMPSs, supported and efficient, serverless-based parallel computation, that combined with Dataplug and Flexecutor, allowed to automatically split the dataset.

These components did not require refactoring the current scientific Python workflows.
Currently, OdP is working towards the integration of EXTRACT technology in production with the required computing facilities.
Furthermore, adapting astronomy pipelines and algorithms to HPC architecture is a huge effort, which has been started in parallel for selected libraries, which would clearly benefit from HPC optimization (e.g., DDFacet). These porting tasks require a lot of time, expertise and interaction between HPC experts and scientists, for each ported tool. The DataPlug solution brings data partitioning (with interaction between developers and scientist required only once, for the format-specific partitioning library) to any pipeline element without porting it to HPC, in a serverless computing context. Thanks to this scheme, OdP is now porting other astronomy pipelines to the EXTRACT framework, only focussing on the DataPlug partitioning tool, while the rest of the pipeline steps are already fully compliant through docker containers.

The following table summarize the key achievement of the TASKA use case in the scope of the EXTRACT project. Detailed results are addressed in their respective sections and/or deliverables

Table 5.2: *KPI summary of the performance if Use Case A1/A2, C and D were used*

KPI	GA Target	Measured Value	Method	Result
Processing Time (P/O ratio)	80% improvement (baseline: P/O~0.1-1)	P/O ≈ 0.01–0.075 depending on dataset and workflow	Measurement P/O representative datasets (Virgo A, CYGLOOP, SUN) and MVP/complex workflows executed with Lithops	**Target achieved** low latency processing achieved
Selective Storage Reduction	90% reduction	×10 reduction (A1) and ×6 reduction (A2) in stored high-resolution data	Comparison of output data volume between original pipeline and TASKA selective storage pipeline	**Partially achieved / close to target**
Efficient Scheduling	End-to-end workflow orchestration capability from raw data to scientific product	Dynamic resource allocation and workflow execution via Lithops/Dataplug and COMPSs demonstrated	Evaluation through pilot workflows and performance tests reported in D2.4 and D3.4	**Target achieved / final validation pending**
Detection Fidelity	99% (accuracy)	High detection accuracy demonstrated but some missed events observed in spike and S-burst detection tests	Qualitative scientific evaluation of neural network detections on solar and Jovian observations	**Partially achieved – AI models still under improvement**

Data Delivery	95%	Near-instantaneous delivery for A1/A2; few minutes for use case C products	Measurement of delay between observation completion and product availability	**Target achieved**
Restoration & Calibration (scientific integrity through the framework)	95% (accuracy)	No measurable degradation compared with manual processing; data integrity preserved (error below numerical ARCH precision)	Comparison between classical processing outputs and TASKA workflow outputs with partitioned datasets	**Target achieved**

Overall, TASKA demonstrates strong performance across the majority of KPIs defined in the Grant Agreement, particularly in detection fidelity and data processing efficiency, while some KPIs (e.g. scheduling or storage reduction) are partially validated due to the experimental setup. Some KPI final answers are also tied to the performance of the trained AI networks. The associated publication in astronomical journals are currently being prepared. In the following section, we revisit KPIs one by one for each use case of TASKA.

5.3.3. Processing/observation (P/O) ratio improvement

The processing over observation ratio (P/O ratio) is a useful metric that scales how the data processing architecture is able to quickly make the observed data (and scientific products) available right after the observation. Depending on the use cases, the P/O varies.

For use case A1-A2: the P/O is virtually close to 0 (i.e. theoretically the best possible) thanks to the live stream processing of the data. The recorded data (with variable resolutions) is literally recorded on disk with the adapted time and frequency resolution. The processing of the data on the Edge side amounts only to the speed (or dump rate) of the data recording on disk.

For use case C:

The main improvement is brought by the data partitioning and resource allocation strategies defined by Lithops and Dataplug (See D2.4 performances).

Based on the selected datasets in the MVP and previous documents, we could perform a representative scaling up of the processing load (see Table 5.2). We have considered two representative workflows: a simple workflow containing three tasks of the MVP (Data flagging & rebining, calibration and imaging) and a complex workflow containing 6 tasks (Data flagging & Rebining of the target and calibrator, calibration, calibration transfer, source subtraction, imaging). We have not included the suspend and looping capability for this test as we want to compare straight P/O measurements without human intervention.

Table 5.2: *Processing over observation ratio for three datasets and workflow complexity. For each dataset (define in previous deliverables) are represented by the number of files, the length of astronomical observations (in hours), and the number of frequency channels. Each dataset represent a particular regime with a data volume ratio range from 1 to 100. A factor of 100 increase in data represent a factor of ~20 in the resulting processing time.*

P/O per dataset & workflow complexity	**Virgo A dataset (MVP)** (N=1, T=2h, Ch=8) Target = 2h **860 MB**	**CYGLOOP dataset** N=11,T=1h40, Ch=10 Calibrator = 20min Target = 1h20 **5 GB**	**SUN dataset** (N=5 SB, T=5h40, Ch=10) Calibrator = 10min Target = 5h30 **89.8 GB**
Lithops only+: MVP workflow*	P/O=0.0104 Proc duration:1m15 min (MVP workflow*)	P/O=0.0675 Proc duration: 6m44 (Complex workflow**)	P/O=0,075 Proc duration:25 min (Complex workflow**)

**Three-step workflow: Flag/rebin, Calibration, Imaging (dirty)*
***Four-step workflow: Flag/rebin, Calibration, Calibration transfer, source subtraction, Imaging (dirty)*

For two typical data processing workflows, the representative dataset shows that, thanks to data partitioning and resource provisioning, the P/O ratios have improved a lot compared to classical processing runs.

For Use case D: This figure of merit does not apply to use case D imager since the focus is put on the scientific quality of the restoration rather than the actual temporal performance of the method.

5.3.4. Selective storage reduction (A1-A2 compression factor)

- **A1 - SpikeNet :**

The outputs of this observation are : one *.spectra* file at the time-resolution lower than the input one, and one *.hdf5* with the tiles where spikes were detected at the original (high) resolution, with scientifically relevant metadata. The impact on data volume of using TASKA-A1 is summarized in table 5.3, using TASKA-A1 leads to a reduction of data volume to store by a factor 10, when only keeping relevant parts at high resolution. Of course the resolution of the compressed file was chosen as a proof of concept and should be adapted to be scientifically relevant for researchers for post-processing.

Table 5.3: *Characteristics of the output files with and without using the TASKA-A1 pipeline: two files are generated*

File type	**Original pipeline**	**TASKA-A1 pipeline**	
(df,dt)	6.1 kHz x 21 ms	Low res file 98 kHz x 1.34 s	High res file 6.01 kHz x 21 ms
.spectra	27 GB/hr	0.037 GB/hr	-
.hdf5	-	-	2.5 GB/hr

with different time-frequency resolution. We see a factor 10 compression regarding the high resolution data.

- **A2 - S-Bursts :**

Similarly as for A1, two output files are generated. In Table 5.4 we summarize the resolution and data volume resulting from this observation. With TASKA-A2, we observe a reduction of up to a factor 6 in data volume.

Table 5.4: *Characteristics of the output files with/without using the TASKA-A2 pipeline. Two output files are*

File type	Original pipeline	TASKA-A2 pipeline	
(df,dt)	3.05 kHz x 2.5 ms	24 kHz x 1.06 s	3.05 kHz x 2.5 ms
.spectra	250 GB/hr	0.082 GB/hr	-
.hdf5	-	-	40 GB/hr

generated with different time-freq resolution, a reduction of a factor 6 was obtained regarding high resolution data.

5.3.5. Efficient scheduling and dynamic resource allocation

Most of the results for this section concerns use case C, and are **all addressed quantitatively in D2.4** as they are tightly linked to the performances brought by WP2 components. These results strongly impact the P/O and science delivery time KPIs. Here we provide only a summary of the main results affecting WP1:

- **Smart provisioning:** less energy waste by optimizing the worker resources dynamically
- **On-the-fly data partitioning** improved reduction of the storage overhead compared to static chunking with a reduction of required storage by 92% to 97%.
- **Lower orchestration overhead:** lower time and less volatile time overhead for using the WP2 framework.
- **Development effort reduction**: reduction from 13% to 27% of the code length when using WP2 components for composing workflows.

The effort carried out by WP2 for smart provisioning offers a unique opportunity for researchers and data scientist to increase their level of control on the consumed resources of the workflow. The learning and optimization phase of workflow resources done with Flexecutor give the workflow operator a choice between different scenarii: e.g. i) optimize for time, ii) optimize for performance, iii) usage. A given workflow (represented as a lithops python script) can be analyzed before being deployed in order to decide which worker resources should be allocated (e.g. to optimize for processing “time”, the “rebinning” stage should use 16CPU/16GB of RAM per worker, but only 1 CPU/2GB per worker to reduce impact). We can easily imagine how a workflow operator could determine the priority of a processing task: operations requiring shorter processing times would demand more resources, while less time-sensitive tasks could be executed with energy efficiency in mind (see D2.4 for details on the optimization). Figure 5.10 show the output of the optimize mode of flexecutor of the specific main_classic.py workflow for the three considered scenario.

```
(venv) ➜  flexecutor git:(main) PYTHONPATH=. Python3 examples/radio_interferometry/scripts/main_classic.py --optimize
```

```
***** Optimizing for target: time *****
Optimizing DAG radio-interferometry
```

Stage #	CPU (vCPU)	Memory (MB)	Workers
rebinning	16	16384	1
full_calibration	8	16384	1
imaging	8	16384	1

```
***** Optimizing for target: usage *****
Optimizing DAG radio-interferometry
```

Stage #	CPU (vCPU)	Memory (MB)	Workers
rebinning	1	2048	1
full_calibration	1	2048	1
imaging	1	2048	1

```
***** Optimizing for target: performance *****
Optimizing DAG radio-interferometry
```

Stage #	CPU (vCPU)	Memory (MB)	Workers
rebinning	1	2048	1
full_calibration	1	2048	1
imaging	1	2048	1

Figure 5.10: *Command-line and Output of flexecutor "optimize" mode that provide the worker configuration for three target scenarii: i) saving time, ii) saving ressources, iii) performance (see D2.4 for details)*

This type of training should be performed over a representative workflow parameter space and could be automated depending on the availability of cluster resources or scientific priorities (e.g., when a transient source is detected in the data requiring urgent processing).

Regarding the implication on the science production, it has no effect on the quality of the scientific products directly (below machine precision on some processor architectures) but represents a higher quality of the workflow monitoring, thanks to the capability to suspend, inspect, communicate, replay portions of the workflow.

5.3.5.1 Comments on Dataplug in the context of scientific data repositories in the Cloud (NEW)

Dataplug is a middleware to enable the dynamic partitioning of complex un-structured or semi-structured scientific data formats (such as Measurement Sets, LiDAR, FASTQGZip, or imzML) residing in object storage (Amazon S3). Unlike data in databases, which can be natively queried and partially filtered, objects in S3 are **continuous**, **monolithic sequences of immutable data** (blobs). S3 only offers the HTTP byte-range interface to access arbitrary bytes, but it lacks the semantics to understand the internal structure of these scientific files. **Since these formats are designed to be fully loaded into memory by a single node, partitioning data for parallel computing in the cloud is a highly complex problem**. Without Dataplug, the only real and implementable alternative (the baseline) to parallelize this workload is static partitioning, a process that requires:

- Downloading the entire original dataset from the object storage.
- Using a domain-specific library (e.g., casacore for MS) to load the data into memory and split it into multiple smaller, valid subsets.

- Re-uploading all these pre-computed chunks back to the cloud storage so that distributed workers can consume them.

By comparing Dataplug against this baseline (the only viable pre-existing alternative), we demonstrate how our approach—based on on-the-fly metadata extraction—**eliminates the need for rewriting, achieving up to a 97.6% reduction in the volume of intermediate data generated (as presented in one the WP2 result plots)**.
To the best of our knowledge, this project is the first one to enable dynamic on-the-fly partitioning of the measurement set (MS) complex astronomy scientific format. We are enabling massive data parallel processing of extreme volumes of astronomy data files. This means that the Dataplug MS partitioner is massively reducing computational costs of data preprocessing and saving enormous unnecessary data transfers thanks to on-the-fly adaptive chunking.
The benefits of dynamic on-the-fly data partitioning in Object Storage are supported by the extensive literature and experiments around cloud-optimized data formats and the parallel processing of formats like ZARR or COG (Cloud-Optimized GeoTIFFs) in geospatial settings. We bring the benefits of cloud-optimized data formats to legacy scientific formats thanks to the Dataplug framework.

Furthermore, the viability of Dataplug as a solution for the dynamic partitioning of scientific data formats has already been recognized and validated by the broader scientific community. The foundational research behind Dataplug was peer-reviewed and published in the proceedings of the prestigious CCGrid 2024 conference. Furthermore, Dataplug was presented in ScyPy conference and has been adopted by Python practitioners around the world. Additionally, Dataplug has been successfully deployed in NEARDATA—a sister project under the same funding topic, coordinated by URV—to handle different use cases and scientific formats in genomics and metabolomics. Notably, the NEARDATA project recently passed its own evaluation with highly satisfactory reviews, further endorsing the robustness and validity of this technological approach.

5.3.6. Data delivery timeliness

Use cases A1 & A2: For A1 and A2, the data delivery is instantaneous as the data processing (dynamic adjustment of the data rate / resolutions) is performed on a live stream of data before being recorded to disk. If the very same operations were to be taken manually from raw hi-resolution data, the delivery time would explode:

i) the massive full-resolution data needs to be transferred to the operator,

ii) the quicklooks needs to be processed to identify the interesting times and frequencies where astrophysical signal of interest need to be kept at high resolution,

iii) The raw full-res data needs to be open as a plain file and be cut at the right slices (sometimes ~hundreds of slices have to be kept at full resolution),

iv) compressed with the right tool and exported to various folders. The human decision making and manual data processing represent the main time delay before delivery. Having an automated decision-making network, deciding to record the data on a live stream ensures a quasi-immediate delivery. A human would need around a full working day to process ~3h of high resolution data whereas the dynamic data recorder has virtually no delay for data delivery (except waiting for the end of the observation). This KPI is fully met by the implementation of A1 and A2.

Use case C:

The interferometric data processing workflow is mainly dominated by input I/O transfer time and by the first stage of data processing (data cleaning and averaging). After these preliminary tasks the data delivery time is closely linked to the P/O time. Since the scientific products are stored directly after the last task has been executed.

Use case D:

This KPI is not yet applicable to use case D since it is in scientific validation. Early tests have shown that the inference time for producing an image cube is ~5min. Delivery time is instantaneous since the products are image cubes.

5.4. Evaluation of Scientific Output Quality and integrity

Use Case A results

The scientific evaluation of the output for A1 and A2 is dependent on the quality and bias of the neural network model that selects relevant features in the data. This model needs to be improved for various source morphologies and be proofed on the field in extended situations. Early results show that the human selection bias is vastly reduced when a neural network is used, especially for small and complex features.

Figure 5.10 below shows the result of a solar observation. The tiles in the time-frequency plane where SpikeNet identified spikes are highlighted by yellow patches on the dynamic spectrum. We can see on the zoomed-in part that the relevant structures are correctly flagged by the algorithm.

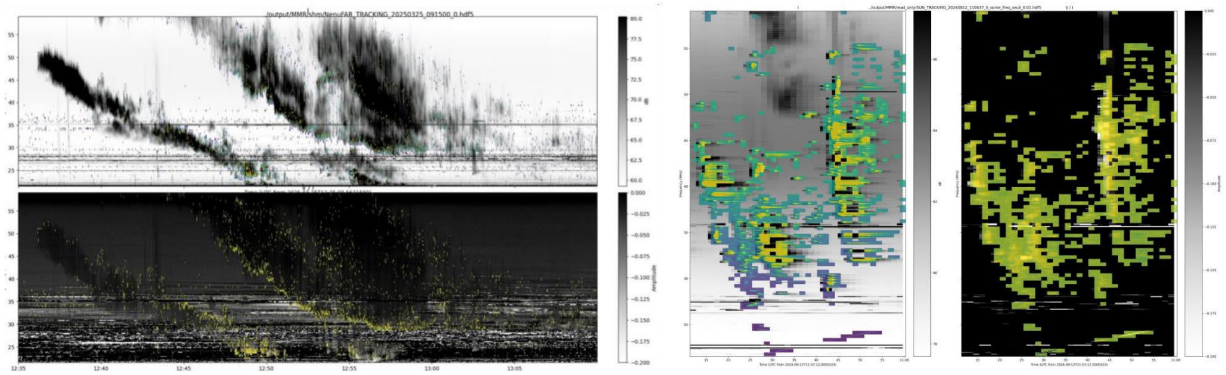


Figure 5.10 : *Results of spikeNet running while observing the Sun. The yellow patches are tiles in the t-f plane where spikes were detected. Left panel is the complete observation and the right panel is a zoom-in.*

In Figure 5.11 we show the result of the detections of s-bursts in a Jovian observation. The purple patches highlight the tiles in the t-f plane where S-bursts have been detected.

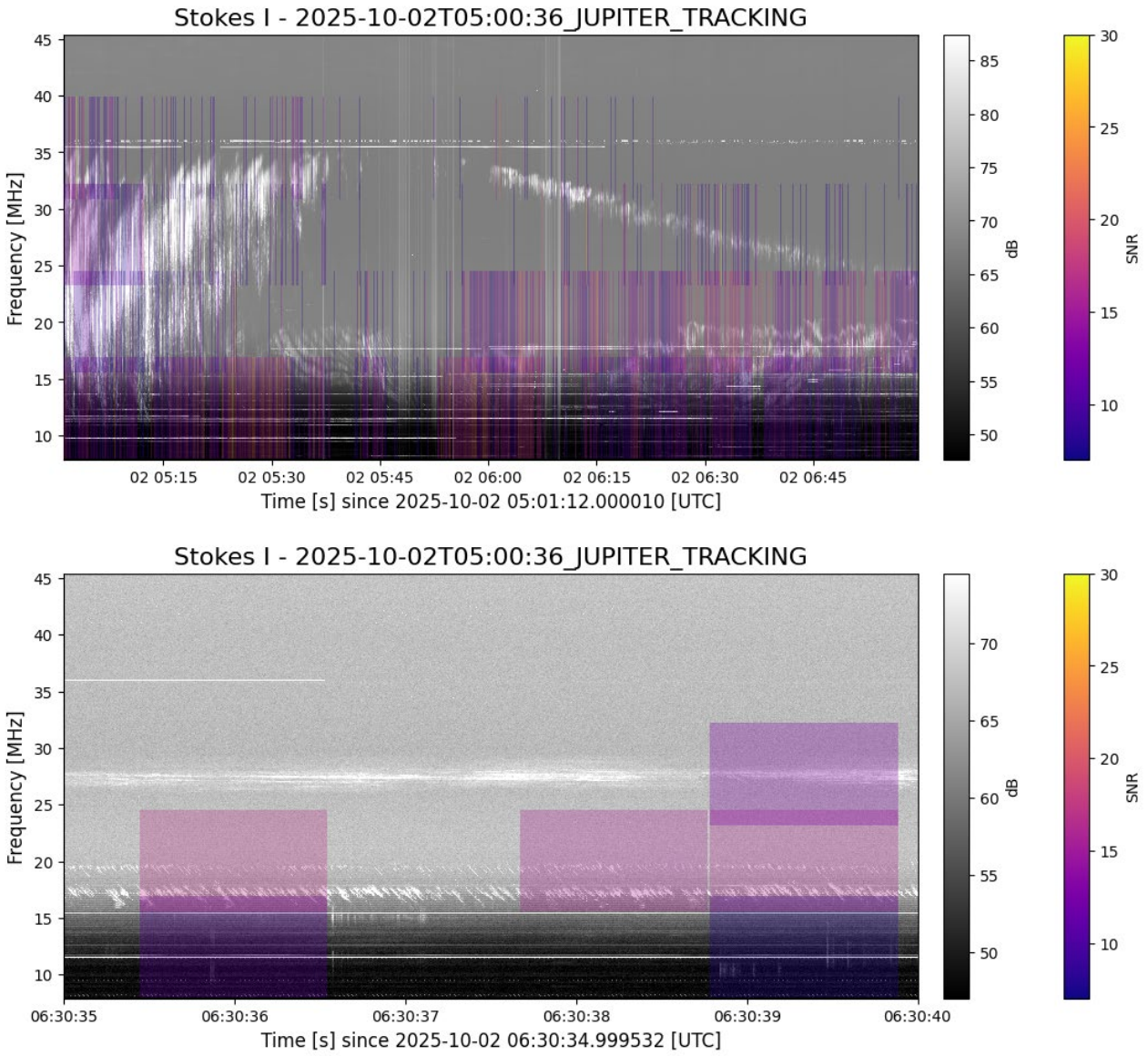


Figure 5.11: *Results of S-bursts detection running while observing Jupiter. The purple patches are tiles in the t-f plane where spikes were detected. Top panel is the complete observation and the bottom panel is a zoom-in.*

As it can be seen in the zoomed-in spectrum, some bursts are not detected by the algorithm; this shows that these models need to be improved before scientific usage. This supports the evaluation of Detection Fidelity and Restoration KPIs

Use Case C results

The question of data integrity was solved early in the project. There are no actual differences in the quality of the products between the full manual data processing and the output of the use case C workflow. Data partitioning was the critical part since this operation touches directly the processed data (opening MS, cutting MS into pieces, merging pieces together into a single MS). Early bugs led to not having perfectly cut slices (the samples at the edge of a slice could appear in two consecutive slices leading to data alteration). These bugs have been solved before the release of the MVP and were the most critical issues that could alter the scientific data integrity. As for the parallel execution of the tools, it led to no quantifiable difference (beyond classical rounding digital error or representation errors on heterogeneous systems). On the contrary, the presence of a data

catalogue that supports the workflow with provenance standards, now increases the robustness of the data.

The data catalogue has been implemented from the Nuvla API and specific python code that present the interaction with the data catalogue as a set of decorators.

Each data record represents a slice of data with its meta-data (location, file parameters, provenance, etc., see Figure 5.12) and is represented by a unique ID in the data catalogue. All records used and produced during a workflow run are registered in the data catalogue (see Figure 5.13).

Each task is decorated with a Nuvla decorator (see Figure 5.14 and figure 4.1.5c of D1.3) which plays the role of an umbrella encapsulating each task. Just before the task is run, a query is sent to the data catalogue to fetch the information about the output records produced by the previous task and pass it locally to the task. After the task is completed, another Nuvla connection is performed to register and append the produced files (location and metadata) of the task to the data catalogue. This way, any task can be decorated in a non-invasive way for the developer and in an invisible way for the user.

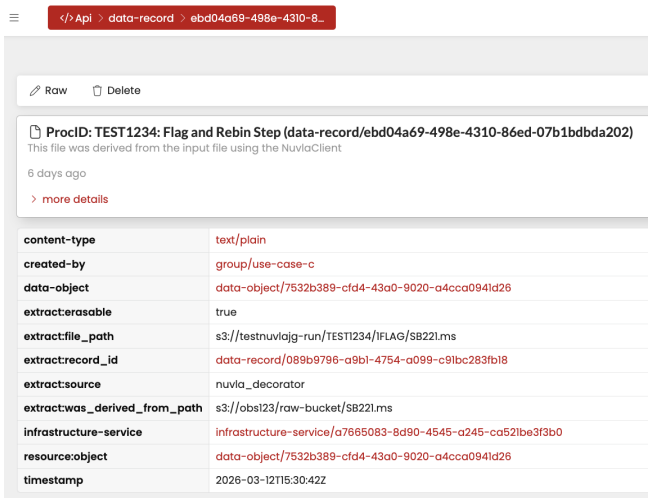


Figure 5.12: *Example of a data record produced by the workflow and augmented by specific Extract fields to implement provenance between data records.*

extract:file_path	extract:was_derived_from_path	id	name
s3://testnuvlajg-run/TEST1234/2CAL/SB223.ms	s3://testnuvlajg-run/TEST1234/1FLAG/SB223.ms	fc9f3779-2fef-48ce-87ea-f7d3e4f6cd5b	ProcID: TEST1234: Calibration Step
s3://testnuvlajg-run/TEST1234/2CAL/SB224.ms	s3://testnuvlajg-run/TEST1234/1FLAG/SB224.ms	1ec62a46-aae7-40a2-8a8c-935cf57830af	ProcID: TEST1234: Calibration Step
s3://testnuvlajg-run/TEST1234/2CAL/SB221.ms	s3://testnuvlajg-run/TEST1234/1FLAG/SB221.ms	1f96b5ac-6582-4833-8f6b-3e64eb8716db	ProcID: TEST1234: Calibration Step
s3://testnuvlajg-run/TEST1234/2CAL/SB220.ms	s3://testnuvlajg-run/TEST1234/1FLAG/SB220.ms	d3d02eab-c685-47c7-a995-a52eb87991e9	ProcID: TEST1234: Calibration Step
s3://testnuvlajg-run/TEST1234/2CAL/SB222.ms	s3://testnuvlajg-run/TEST1234/1FLAG/SB222.ms	d869ad28-05ba-41d3-a8e0-ed42c61996a2	ProcID: TEST1234: Calibration Step
s3://testnuvlajg-run/TEST1234/1FLAG/SB222.ms	s3://obs123/raw-bucket/SB222.ms	359480b3-6849-4006-8e95-87f998a0344f	ProcID: TEST1234: Flag and Rebin Step
s3://testnuvlajg-run/TEST1234/1FLAG/SB223.ms	s3://obs123/raw-bucket/SB223.ms	0c72d28f-809e-42ca-ab0b-7636f497fd3a	ProcID: TEST1234: Flag and Rebin Step
s3://testnuvlajg-run/TEST1234/1FLAG/SB224.ms	s3://obs123/raw-bucket/SB224.ms	290d9862-b632-432a-9ba7-8ccbde7e06da	ProcID: TEST1234: Flag and Rebin Step
s3://testnuvlajg-run/TEST1234/1FLAG/SB221.ms	s3://obs123/raw-bucket/SB221.ms	be4cd15b-1c1a-42d8-a8cf-65445ab8580b	ProcID: TEST1234: Flag and Rebin Step
s3://testnuvlajg-run/TEST1234/1FLAG/SB220.ms	s3://obs123/raw-bucket/SB220.ms	599db971-a11c-4564-a166-9e29a3986310	ProcID: TEST1234: Flag and Rebin Step
s3://testnuvlajg-run/TEST1234/3IMAGING/SB224.fits	s3://testnuvlajg-run/TEST1234/2CAL/SB224.ms	f7afa680-dac0-4595-812d-b7dcd9eee0d7	ProcID: TEST1234: Imaging Step
s3://testnuvlajg-run/TEST1234/3IMAGING/SB221.fits	s3://testnuvlajg-run/TEST1234/2CAL/SB221.ms	2b8449c9-f5ec-43c5-b3e7-ce2e7b90128c	ProcID: TEST1234: Imaging Step
s3://testnuvlajg-run/TEST1234/3IMAGING/SB222.fits	s3://testnuvlajg-run/TEST1234/2CAL/SB222.ms	83d399d1-79ab-47c8-84b1-11c0466cdf05	ProcID: TEST1234: Imaging Step
s3://testnuvlajg-run/TEST1234/3IMAGING/SB223.fits	s3://testnuvlajg-run/TEST1234/2CAL/SB223.ms	fd3f86ee-8501-4a78-80ea-a9fc6e7c17cc	ProcID: TEST1234: Imaging Step
s3://testnuvlajg-run/TEST1234/3IMAGING/SB220.fits	s3://testnuvlajg-run/TEST1234/2CAL/SB220.ms	3977055f-33d1-4ba4-8aa9-3aa2bebb8e6f	ProcID: TEST1234: Imaging Step
s3://obs123/raw-bucket/SB220.ms		538612e7-8e03-437e-a061-4918188b1ea3	Raw Data
s3://obs123/raw-bucket/SB222.ms		e5144472-45f6-4fae-8fdf-2788f7c045f4	Raw Data
s3://obs123/raw-bucket/SB223.ms		c5db63fc-9f32-42c1-89ff-49e93807280f	Raw Data
s3://obs123/raw-bucket/SB224.ms		5bd498a7-4d19-4ace-b4c2-f369fd48c946	Raw Data
s3://obs123/raw-bucket/SB221.ms		67506e80-2552-459b-a07a-0e59b83dd0a6	Raw Data

Figure 5.13: *List of data records in Nuvla Data Catalogue produced by the workflow. Each data records contains the metadata of the step as well as the ID of the precedent data record to previous data records. Each input and output data records are linked together in the catalogue.*

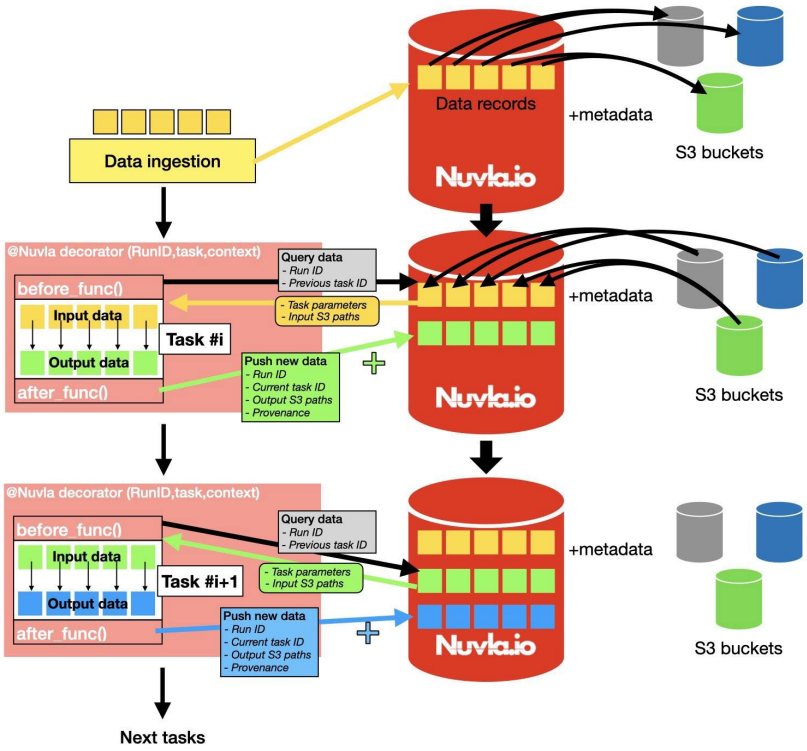


Figure 5.14: *Principle of data catalogue interaction during each task. The Nuvla decorator is hidden from the workflow user. The before_func() is triggered before the start of the task to prepare the context and retrieve input data location. After the task has been executed, new data and metadata is pushed to the data catalogue.*

As a side project, we proposed to two engineering students to work on what could be a demonstrator of a GUI for composing new workflows. Using a simple sound processing test case, they managed to develop an interactive GUI based on draggable and connectable boxes to edit a workflow. Future development would include the capability to create and chain tasks, connect to Object Storage namespace, workflow step-by-step run or full run and export the workflow as a YAML file that can be used by workflow script. A push toward developing easy-to-use GUI is also something that would lower the barrier to develop new workflows. The current code developed with the students already enables the edition of simple workflows (see Figure 5.15).

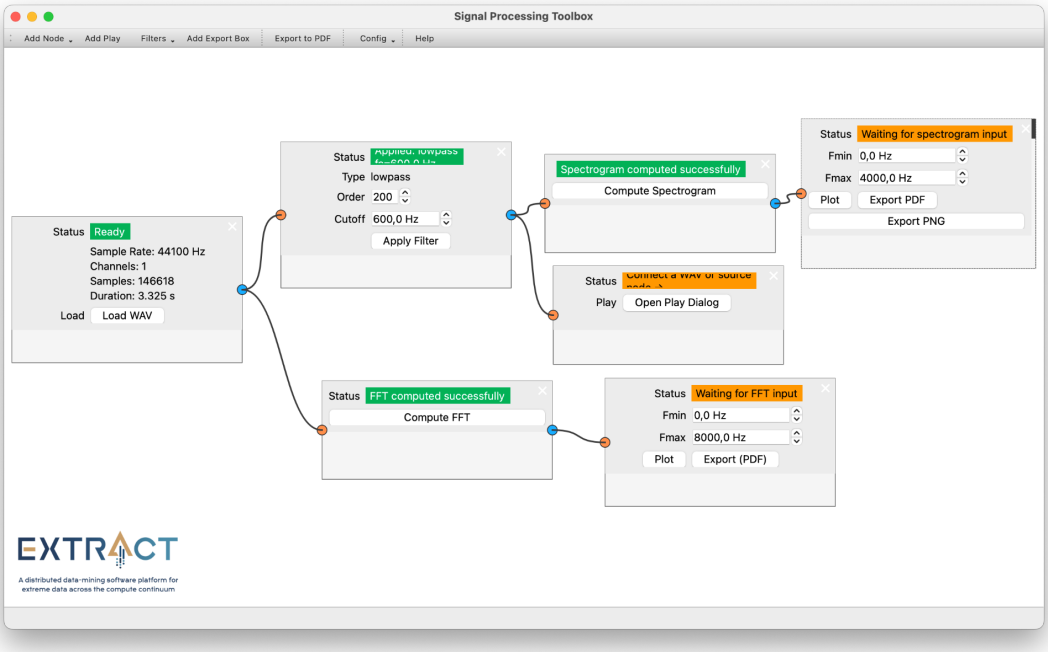


Figure 5.15: *Demo code that successfully define a sound filtering workflow by using draggable box (1 box = 1 task) that can be connected to one or several subsequent boxes (here are filtering, performing Fourier transform, plotting spectrogram, exporting plots as png file). This demonstrates the possible capabilities for a workflow edition and execution GUI.*

Use Case D results

The TASKA-D software package is designed to deliver unbiased, high-quality scientific imaging, specifically tailored for radio astronomy applications. This innovative imager integrates several state-of-the-art features that enhance its performance and versatility (when it will be fully scientifically validated):

- **Deep Learning Integration:** TASKA-D employs a trained deep neural network specifically adapted to the characteristics of a given radio telescope, ensuring optimal handling of instrument-specific effects. It was trained on NenuFAR in the context of use case D but can be also trained for future SKA precursors (LOFAR, MeerKAT, etc.).

- **Advanced Deconvolution:** The software can deconvolve both static sky components and transient sources, enabling the separation of persistent signals from variable phenomena even in the presence of multiple transients in the data.

- **Transient Source Characterization:** TASKA-D facilitates the unambiguous detection of transient sources and, under favorable conditions, allows for a complete characterization and modeling of their light curves. This capability is particularly challenging and represents a significant advancement in radio imaging techniques.

Scientific Evaluation and Validation

The evaluation of TASKA-D performance is complex due to the scientific rigor required and the subtleties of transient detection. Validation is performed using simulated varying skies with well-defined parameters:

- **Via training and Validation Sets:** Each sky simulation includes 1–5 constant sources and 1 transient source. Source flux densities, coordinates, and profiles are drawn randomly, while noise levels are set based on real observations and
- **Via test Sets:** Test sets maintain similar characteristics but extend the number of constant sources from 1 to 100 to assess the software's scalability and robustness.

Example reconstructions are displayed in Figure 5.16 and 5.17, and demonstrate the initial good performance of this good imager with very high denoising and deconvolution capabilities during inference. This supports the evaluation of Detection Fidelity and Restoration KPIs

During these tests, TASKA-D demonstrates minimal remaining biases, such as minor flux instability and potential biases related to the sequence size fed to the network. Despite being trained exclusively on point sources, TASKA-D also shows promising performance in the deconvolution of extended sources (using the trained model on point source only, see Figure 5.18), thanks to models trained using a kernel-trick-based loss function.

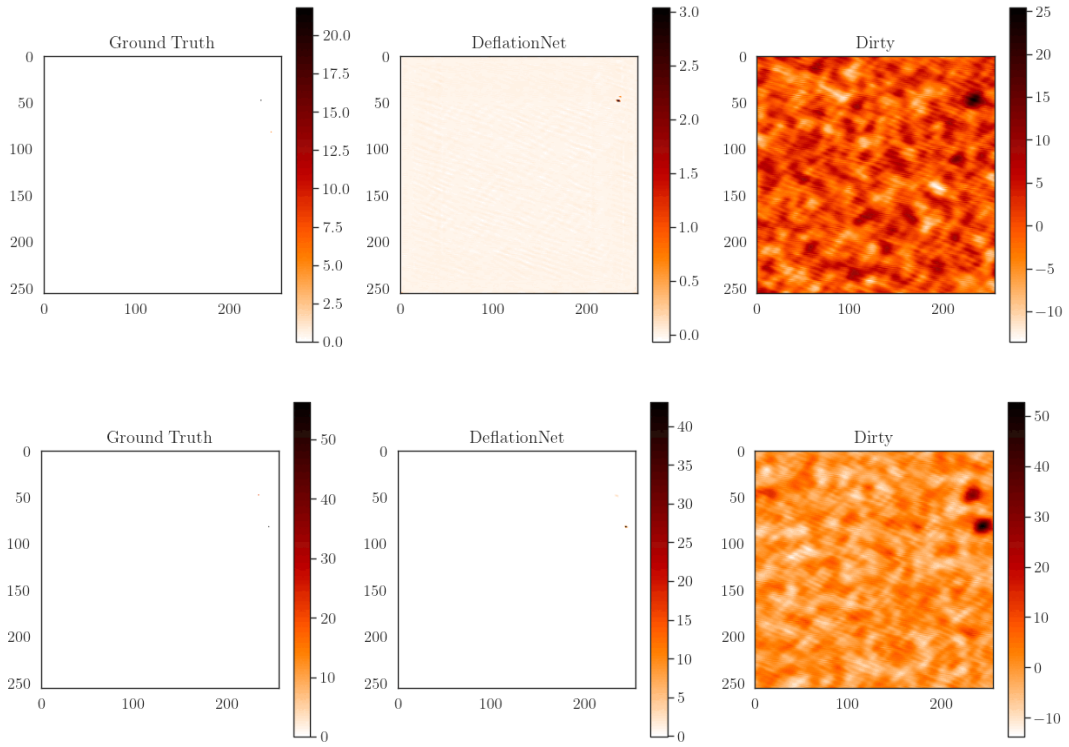


Figure 5.16: *(top, from left to right) a single strong point sources (ground truth), deconvolved result obtained with DeflationNet (TASKA-D) and the dirty image (raw image before deconvolution), (bottom, from left to right) Same field with the detected transient source and deconvolved (the static source is still present). DeflationNet results show unbiased detection of the transient source without noise.*

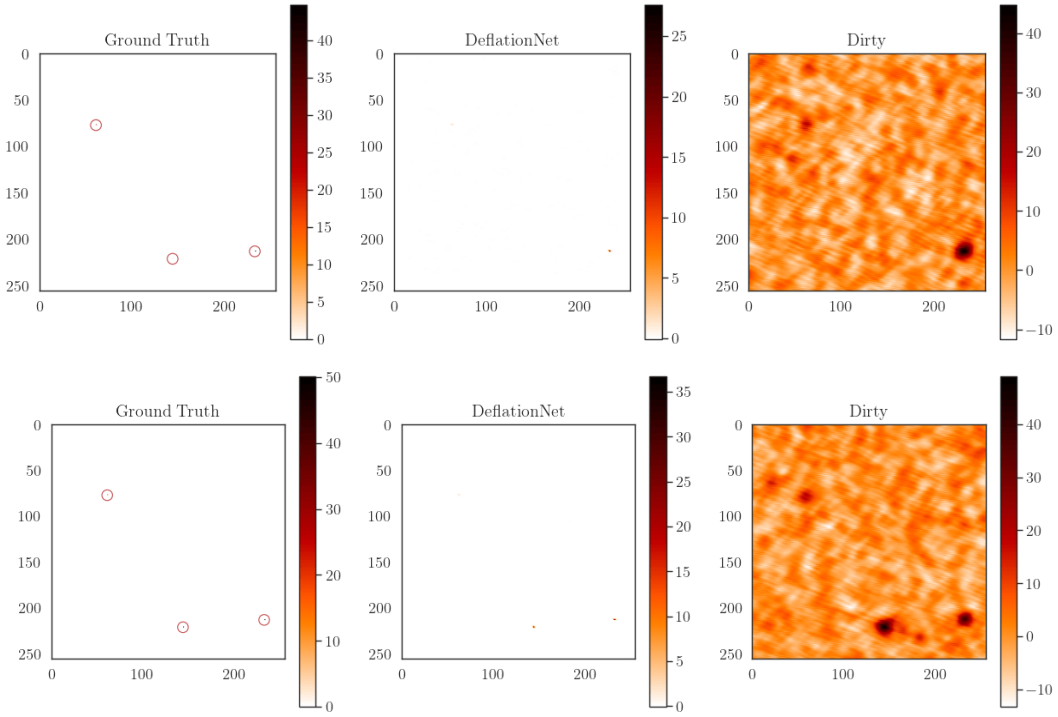


Figure 5.17: *(top, from left to right) a weak, transient and strong point sources (ground truth), deconvolved result obtained with DeflationNet (TASKA-D) and the dirty image (raw image before deconvolution), (bottom, from left to right) Same field with the detected transient source and deconvolved. DeflationNet results show unbiased detection of the transient source without noise. Animations in https://share.obspm.fr/s/eAd3KbDjzfds64P*

Details parameters for the training

Training time : ~27 hours **Learning rate** : 1E-4 **Loss** : mean MSE

Number of epoch : 100 **Optimiser** : ADAM **Precision** : Mixed (FP16-FP8)

GPU : 1 x Tesla T4 **Language** : PyTorch

Custom aid for the training: Input flux normalisation factor : 100, [Optional] Gaussian kernel convolution (7x7) to smooth results in case of isolated sources.

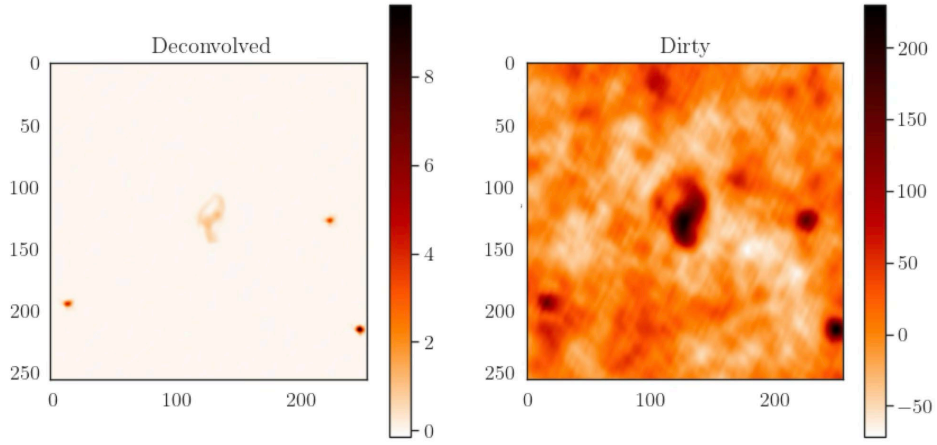


Figure 5.18: *Attempt deconvolution of a resolved emission with the same neural networks only trained on point sources. The network was not trained on resolved emission but is still resilient to deconvolve extended emission of the Cygnus Loop region.*

5.5. Identified Gaps and Refined Workflows

5.5.1. TASKA Use-case A

The major limitation for production deployment of TASKA Use-case A1 and TASKA Use-case A2 is the tedious manual deployment (docker build) of docker images on all nodes available to process NenuFAR streamed data. And, currently, the manual startup of the right images when observations are scheduled.

Reusing experience gained on TASKA Use-case C and expertise from the BSC group, a mechanism to select and start the right image to process data for the right science branch was setup and is being integrated as a prototype of a Shared Cluster in TASK A Use Case A3-SharedCluster.

Image repositories are used, as well as Kubernetes (K8s) to manage servers (nodes) where images (pods) can be started (and restarted with latest image versions, only built once) and monitored with Grafana+Prometheus.

A dedicated service fed by the NenuFAR telescope observation scheduler is under construction to automatically start the right processing from the right pod on the right node (depending on node capabilities and availability) for the next observation scheduled.

Currently, only TASKA Use-case A1 SpikeNet is integrated, but care is taken to easily extend this to TASKA Use-case A2 S-Bursts and to classical processings used on a daily basis on NenuFAR (Time-Frequency analysis and pulsar processing).

Moreover, the package created in the frame of Use-case A2taska-a2 is being used by other scientists in the community as a post-processing tool on already existing data. Thus the algorithm will continue to be improved, mainly with the training of a ML model based on anomaly detection, that should be even more accurate that the existing algorithm.

5.5.2. TASKA Use-case C

The current version of TASKA-C code base has demonstrated that a complete workflow can be implemented in a controlled context. The next step is to expand the span of possible scenarios (various data sizes, various scientific targets, deployment on heterogenous sites) to increase the coverage of the resource allocation optimization.

The key aspect is also to prepare future implementations of TASKA-C for different use cases. The platform needs to be ready to hand off to other teams of developers.

The EXTRACT TASKA team has been approached by another scientific team interested in fully using the EXTRACT framework to perform data reduction of Pulsar data. This new use case comes with new sets of community tools that need to be connected to new specific tasks. Thanks to the agnostic nature of the framework, this dissemination is rather straightforward to implement.

5.5.3. TASKA Use-case D

The TASKA-D code is available to the scientific community, the quality of the scientific products solely depends on the quality of the training data set, the convergence of the training and the data quality itself. Scientific studies are currently being conducted to validate scientifically the efficiency of the source detection. The current version of the code will be extended to detect more

complex variable sources (extended emissions, moving targets and 4-D source which move in space, time and frequency, etc.).

Future Extensions and Capabilities

TASKA-D has been designed with extensibility in mind. Future developments will enhance its applicability and scientific utility, including:

- Integration of extended and mobile sources in simulations and imaging.
- Support for multi-channel Measurement Set (MS) handling and multi-MS imaging workflows.
- Incorporation of transformer-based models for image restoration.
  Implementation of trainable gridder models.
- Expansion toward visibility-to-visibility modeling (therefore without imaging) and an advanced training framework for extremely large datasets.

These planned features aim to broaden TASKA-D's versatility, allowing it to tackle increasingly complex imaging scenarios and provide robust, scientifically reliable results.

6. Overall Evaluation Results and Lesson Learned

6.1. Synthesis of Findings Across Use-Cases

The evaluation of the EXTRACT project demonstrates that the proposed architecture and technologies successfully address the objectives defined for both use cases—PER and TASKA—while validating the effectiveness of compute continuum principles across heterogeneous environments.

For the **PER use case,** the results confirm that the integration of reinforcement learning with MEC-enabled infrastructure improves evacuation efficiency, responsiveness, and adaptability in small- to medium-scale scenarios. The system consistently outperforms heuristic baselines in a majority of configurations and demonstrates the ability to dynamically adapt routing decisions under changing conditions. The validation activities conducted in Venice further confirm the operational feasibility of the end-to-end system, including real-time data exchange, low-latency inference, and user interaction through the EMA application. At the same time, the evaluation highlights limitations in scalability when addressing larger and more complex urban topologies.

The PER evaluation therefore leads to two complementary conclusions. First, the current BDQ policy-learning formulation demonstrates useful performance on small and medium-sized topologies but does not yet provide reliable city-scale optimisation. Second, the large-scale validation confirms that this algorithmic limitation does not constitute a scalability limitation of the PER infrastructure itself: the UDT ingestion pipeline, simulator, routing workflow and associated platform components can operate on substantially larger dynamic urban scenarios when movement decisions are generated independently of the current RL policy

For the **TASKA use cases**, the evaluation shows substantial improvements in data processing efficiency, scalability, and scientific output quality. On the one hand, the MurMuRe-based architecture (TASKA-A) enables real-time processing and compression of radio signal based on scientific content while flexible workflow orchestration framework (TASKA-C) enable easy composition, workflow dynamic and logic, optimization of resources while maintaining scientific integrity, usability and traceability. Significant data reduction capabilities (up to 90% in some scenarios) were achieved without compromising scientific value, demonstrating the effectiveness of combining AI-driven processing with modular workflow design. A new cutting-edge imager (TASKA-D) was proposed to tackle the problem of detecting transient sources in the data, in an unambiguous and unbiased way. The scientific validation is still pending but the overall structure of the imager is ready to be used and improved by the community of users.

Across both use cases, the EXTRACT platform demonstrates:

- Scalability across cloud, edge, and HPC environments
- Efficient data processing and low-latency decision-making
- Interoperability between heterogeneous components
- Robust monitoring and observability capabilities

These results confirm the maturity of the EXTRACT architecture and its ability to support both scientific and societal applications.

6.2. Key Observations Across Use Cases

The evaluation of the EXTRACT platform highlights several important observations that characterise its performance across both use cases and support its readiness for real-world adoption.

Scalability considerations

The results indicate that system performance remains strong in controlled and medium-scale environments, while larger and more complex scenarios introduce additional computational and coordination requirements. This reflects the inherent complexity of operating across large-scale distributed systems.

Distributed coordination across the compute continuum

The integration of cloud, edge, and HPC resources enables flexible and efficient processing, while also requiring careful coordination to ensure synchronization, consistency, and low-latency data exchange across system components.

Adaptation to heterogeneous infrastructures

The deployment of the EXTRACT platform across different environments demonstrates its flexibility, while also highlighting the need for context-specific adaptations depending on infrastructure constraints and resource availability.

Role of user interaction in system effectiveness

The PER validation confirms that user interaction, interface design, and communication mechanisms are key elements influencing the effectiveness and acceptance of decision-support systems.

Influence of real-world conditions

Operational environments introduce variability in data quality, connectivity, and positioning accuracy, which must be accounted for to ensure reliable system performance.

6.3. Lessons for Deployment and Exploitation

The evaluation of the EXTRACT use cases provides a set of consolidated lessons that support the sustainability, transferability, and exploitation of the platform.

Importance of scalable and adaptive architectures
The EXTRACT architecture demonstrates the need for solutions that can adapt to increasing complexity while maintaining performance across heterogeneous environments.

Value of modular and flexible workflows
The TASKA use case confirms that modular workflow design is essential for supporting different operational requirements and facilitating reuse across domains.

Need for efficient coordination across computing layers
Effective interaction between edge, cloud, and HPC resources is a key enabler for achieving low latency, efficient processing, and system robustness.

Central role of user-centric design
The PER use case highlights that usability and trust are critical factors for the successful adoption of AI-driven decision-support systems.

Robustness as a prerequisite for real-world deployment
The evaluation confirms the importance of designing systems that can operate reliably under variable and imperfect real-world conditions.

Support for transferability and reuse
The EXTRACT approach demonstrates that modular and interoperable components facilitate the extension of the platform to new domains and application scenarios.

7. Sustainability, Transferability and Exploitation Roadmap

This section outlines the sustainability, transferability, and exploitation perspectives of the EXTRACT use cases, building on the results of WP5 (D5.3), which defines the project's strategy for dissemination, exploitation, and long-term impact. It highlights how the developed solutions can be maintained, adopted, and extended beyond the project lifetime, considering both technical maturity and real-world applicability.

Sustainability

For the TASKA use case, all developed tools are already positioned for production-level deployment, as they are required for the day-to-day operation of the NenuFAR radio telescope. The solution is deployed locally at the Nançay Radioastronomy Observatory and supports continuous data processing workflows. Observatoire de Paris will maintain the codebase and promote further adoption in the context of the upcoming SKA Regional Centre in France and Europe. Through initiatives such as the ECLAT joint laboratory, the EXTRACT workflow framework is expected to play a significant role in future radioastronomical data processing and optimisation.

For the PER use case, sustainability is supported by the system's integration within real-world validation environments, particularly in Venice. The developed components, including the Urban Digital Twin, reinforcement learning-based routing engine, and mobile application, provide a foundation for operational decision-support systems in urban crisis management. Continued engagement with local stakeholders, such as the Metropolitan City of Venice, supports the potential for long-term adoption and integration into existing emergency management practices.

Transferability

The TASKA solution demonstrates strong transferability across radioastronomical infrastructures, as it addresses common challenges related to large-scale data processing, selective storage, and workflow optimisation. Its modular design allows adaptation to different observatories and scientific use cases, with varying levels of complexity and data processing requirements.

Similarly, the PER solution has been designed for transferability across urban environments beyond the Venice pilot. Its modular architecture enables adaptation to different city layouts, infrastructure conditions, and data availability. The integration of the Urban Digital Twin, reinforcement learning models, and compute continuum capabilities supports deployment in heterogeneous environments, from edge to cloud. While calibration is required to reflect local urban dynamics and population behaviours, the overall framework can be reused to support a wide range of crisis management scenarios, including evacuation, crowd management, and risk mitigation.

Exploitation

From an exploitation perspective, both use cases demonstrate strong potential for uptake in their respective domains. TASKA directly supports ongoing scientific operations and is expected to be further extended within the SKA ecosystem and related research infrastructures.

The PER use case offers exploitation opportunities in the domain of smart cities and emergency management, where data-driven decision-support systems are increasingly required. The solution can be adopted by public authorities, civil protection agencies, and technology providers to enhance preparedness and response capabilities. Its alignment with real-world operational needs, combined with its modular and scalable design, supports its potential integration into existing digital infrastructures.

Overall, the results of EXTRACT, as consolidated in D5.3, confirm that both use cases provide sustainable and transferable solutions with clear exploitation pathways. The project demonstrates the viability of data-driven orchestration across the compute continuum and its applicability to both scientific and societal challenges.

8. Conclusion

The final evaluation of EXTRACT provides a clear and consolidated view of the results achieved across the considered use cases, measured against the KPIs defined in the Grant Agreement. The analysis integrates evidence from monitoring data, simulations, and application-level outputs, enabling a robust assessment of both technical performance and operational effectiveness.

The results confirm the validity of the proposed data-driven orchestration approach across the compute continuum, demonstrating its capability to operate under realistic conditions and to address heterogeneous system requirements. At the same time, the evaluation highlights specific aspects where further refinement is needed, particularly in relation to scalability and the quantitative assessment of certain performance dimensions.

By combining the outcomes of the PER and TASKA use cases, the analysis provides a comprehensive understanding of the platform's behaviour, its strengths, and its current limitations. Overall, the findings indicate a solid level of technological maturity and support the readiness of the EXTRACT platform for further development and deployment in real-world scenarios.

The results show that both use cases effectively validate the EXTRACT approach to data-driven orchestration across the compute continuum. In the PER use case, the reinforcement learning-based routing strategy proves capable of handling evacuation dynamics, improving both efficiency and response time in small- and medium-scale scenarios. At the same time, some scalability limitations emerge in more complex environments. The use of MEC-based processing contributes to faster response times, supporting near real-time decision-making. However, KPIs such as safety, fairness, and user acceptance would require additional quantitative validation.

For the TASKA use case, the evaluation confirms that the data mining and selective processing approach successfully reduces data volume while maintaining detection accuracy and ensuring reliable data delivery. The system also shows good performance under constrained infrastructure conditions, highlighting the robustness and adaptability of the monitoring and orchestration components.

Overall, the results confirm that the EXTRACT platform meets its main objectives, enabling efficient and scalable data processing across heterogeneous environments. The evaluation provides a solid picture of system performance and indicates that the platform has reached a level of maturity suitable for further development and real-world deployment.

9. Annex

Annex I – User Experience Questionnaire for the Evacuation Mobile App (EMA)

Objective

This questionnaire evaluates the user experience of the EMA within the EXTRACT project, focusing on usability, clarity, navigation reliability, and overall satisfaction.

Methodology

Two validation sessions were conducted in Venice in October 2025 and February 2026 with ten expert users with operational backgrounds, including retired law enforcement officers and emergency management professionals. Following a structured briefing on the EMA application and evaluation protocol, participants were instructed to navigate from a predefined starting point to a designated safe location using the app. The sessions enabled the assessment of key KPIs, including usability, clarity of guidance, route effectiveness, and task completion. A structured post-test questionnaire was administered to collect qualitative and quantitative feedback, supporting performance evaluation and iterative system refinement.

Image Gallery

First validation session in Venice (3-4 October 2025): Expert evaluators engaged in testing the EMA mobile application under real-world evacuation conditions.


THE FIRST
THREE DAYS


THE FIRST
THREE DAYS

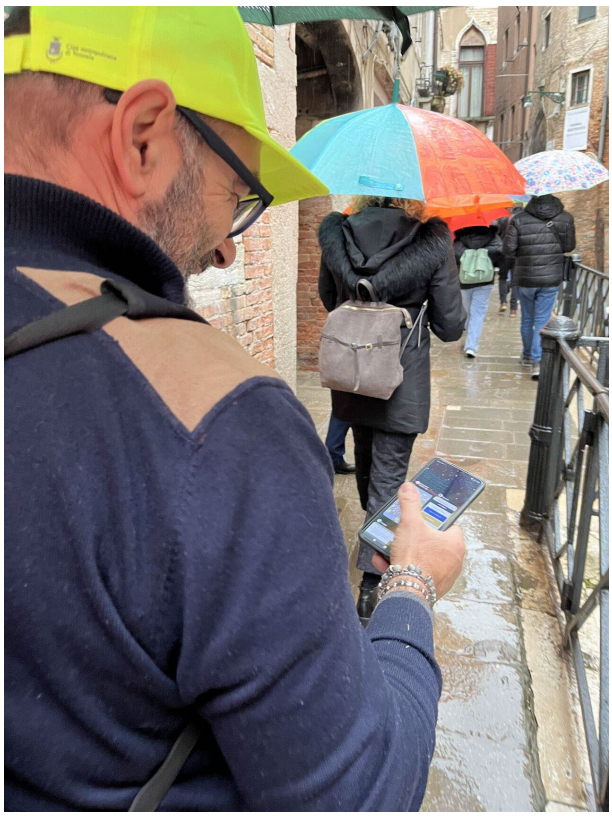

EXTRACT
USE CASE n.1
VENEZIA

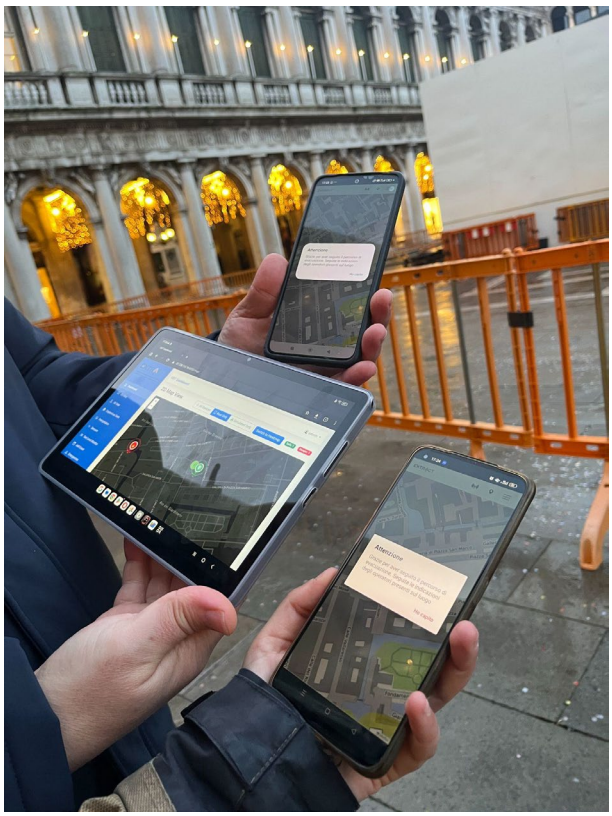

THE FIRST
THREE DAYS


EXPERT PANEL


EXTRACT KNOWLEDGE TO MAKES CITIES SMARTER

Second validation session in Venice (3-4 February 2026): Expert evaluators engaged in testing the EMA mobile application under real-world evacuation conditions.





Questionnaire (Italian / English)

ID	Italian Question	English Question	Type
1	Quanto è stato semplice capire come usare l'app al primo avvio?	How easy was it to understand how to use the app at first launch?	4 level Likert ("Very low" to "Very High")
2	Le istruzioni per seguire il percorso di evacuazione erano chiare?	Were the instructions for following the evacuation route clear?	4 level Likert ("Very low" to "Very High")
3	Hai trovato facilmente le informazioni principali (es. posizione, direzione, distanza)?	Did you easily find key information (e.g., position, direction, distance)?	Yes/No
4	Se si è risposto "No" alla domanda precedente, potrebbe	If you answered 'No' to the previous question, please explain why.	Open

	aggiungere una motivazione?		
5	Quanto è stato intuitivo interagire con la mappa e i percorsi suggeriti?	How intuitive was it to interact with the map and suggested routes?	4 level Likert ("Very low" to "Very High")
6	Hai riscontrato difficoltà nel seguire il percorso in tempo reale?	Did you encounter difficulties following the route in real time?	Yes/No
7	Se si è risposto "Sì" alla domanda precedente, potrebbe aggiungere una motivazione?	If you answered 'Yes' to the previous question, please explain why.	Open
8	Il percorso proposto quanto ti è sembrato affidabile e coerente con l'ambiente reale (scala 1–5)?	How reliable and consistent did the proposed route seem compared to the real environment (scale 1–5)?	4 level Likert ("Very low" to "Very High")
9	Ti sei sentito guidato in modo sicuro durante la simulazione?	Did you feel safely guided during the simulation?	Yes/No
10	Hai notato ritardi, errori o imprecisioni nella localizzazione?	Did you notice delays, errors, or inaccuracies in localization?	Yes/No
11	Se si è risposto "Sì" alla domanda precedente, potrebbe argomentare?	If you answered 'Yes' to the previous question, please elaborate.	Open
12	Quanto sei soddisfatto dell'esperienza complessiva?	How satisfied are you with the overall experience?	4 level Likert ("Very low" to "Very High")

Table 9.1: Questionnaire composition (original language italian)

Between the October and February test campaigns, the questionnaire results show a consistent improvement across all evaluated dimensions, indicating enhanced usability, reliability, and user perception of the EMA system.

More technically:

- **Usability and clarity metrics (Q1, Q2, Q5)** increased from ~3.0 to ~3.5, suggesting improved interface design and better comprehension of instructions.
- **Perceived reliability (Q8)** shows the most significant gain (from ~3.0 to 4.0), indicating a substantial improvement in path coherence with the real environment.
- **Perceived safety and guidance (Q9)** increased from ~2.8 to ~3.8, reflecting higher user confidence during navigation.
- **Overall satisfaction (Q12)** remained stable (~3.0), suggesting that while core functionalities improved, there is still margin for enhancing the overall experience.

Overall, the results indicate a **maturation of the system**, with notable gains in navigation accuracy and user trust, likely driven by improvements in positioning, routing algorithms, and system responsiveness.

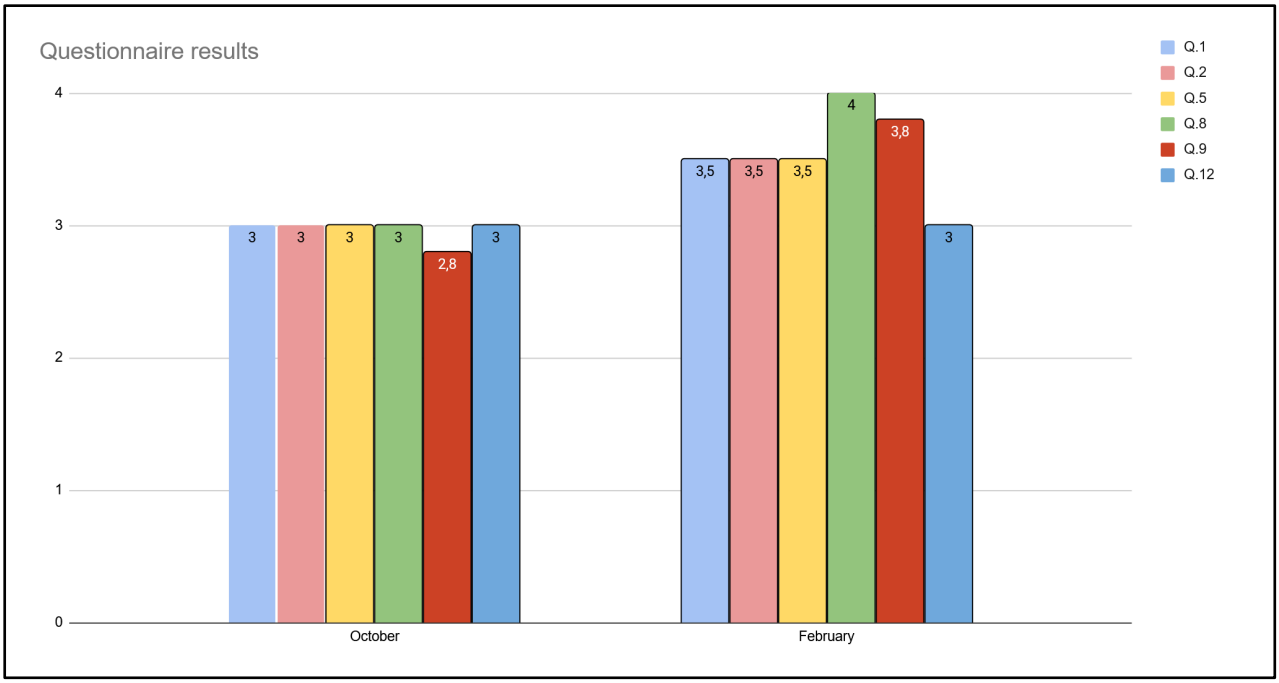


Figure 9.1: Graphical comparison of the questionnaire results between the first validation in October 2025 and February 2026

10. Acronyms and Abbreviations

- AI – Artificial Intelligence
- API – Application Programming Interface
- AWS – Amazon Web Services
- CA – Consortium Agreement
- CCS – Compute Continuum Site
- CD – Continuous Development
- CI – Continuous Integration
- COE – Container Orchestration Engine
- CPU – Central Processing Unit
- CR – Container Runtime
- D – Deliverable
- DevOps – Development and Operations
- DevSecOps – Development, Security and Operations
- DoA – Description of Action (Annex 1 of the Grant Agreement)
- ECDF – Empirical Cumulative Distribution Function
- GA – General Assembly / Grant Agreement
- GPU – Graphics Processing Unit
- HPC – High Performance Computing
- ISO – International Organization for Standardization
- KPI – Key Performance Indicator
- K8s - Kubernetes
- M – Month
- MS – Milestones
- NAT – Network Address Translation
- OS – Operation System
- PaaS – Platform as a Service
- PM – Person month / Project manager
- QUIC – Quick UDP Internet Connections
- RL – Reinforcement Learning
- SaaS – Software as a Service
- SHA – Secure Hash Algorithm
- T – Task
- TLS – Transport Layer Security
- UDP – User Datagram Protocol
- VCS – Version Control System
- VM – Virtual Machine
- VPN – Virtual Private Network
- WP – Work Package

11. References

[1] https://openmp.org
[2] https://developer.nvidia.com/cuda-zone
[3] http://compss.bsc.es
[4] S. Chang, et.al., *"Feasibility of Running Singularity Containers with Hybrid MPI on NASA High-End Computing Resources"* CANOPIE-HPC, 2021
[5] https://aws.amazon.com/s3
[6] https://www.ovhcloud.com/en-gb/
[7] https://www.redhat.com/es/technologies/storage/ceph
[8] https://kubernetes.io/
[9] https://skypilot.readthedocs.io/en/latest/
[10] https://kubeedge.io/en/
[11] https://nuvla.io
[12] https://www.ovhcloud.com/en-ie/
[13] https://www.ovhcloud.com/en-ie/public-cloud/compute/
[14] https://docs.ray.io/en/releases-2.7.0/cluster/kubernetes/index.html
[15] https://kubernetes.dask.org/en/latest/operator.html
[16] https://spark.apache.org/docs/latest/running-on-kubernetes.html#how-it-works
[17] https://help.ovhcloud.com/csm/en-public-cloud-kubernetes-install-trivy?id=kb_article_view&sysparm_article=KB0049874
[18] https://csrc.nist.gov/glossary/term/defense_in_depth
[19] https://nvd.nist.gov/
[20] https://github.com/binareio/FastCVE
[21] https://www.cisa.gov/sbom
[22] https://developers.cloudflare.com/ssl/reference/cipher-suites/recommendations/
[23] https://cryptobook.nakov.com/digital-signatures/ecdsa-sign-verify-messages

11. DEMO

Demo Video

- The video linked below delivers the demo story as explained in Section 6.2.1 and detailed in Section 6.2.2. To avoid a lengthy video with mundane details, we skip the whole setup part and present just the demo itself, of engaging with buckets and objects over Sky Store.

https://b2drop.bsc.es/index.php/s/eRQrQ6jHbHGnGkT

- The animations for Use Case D restoration are the animated version of figures 5.16 and 5.17

Demo Resources

Sky Store repository on Github: https://github.com/gilv/skystore

Use Case D restoration demo: https://share.obspm.fr/s/eAd3KbDjzfds64P