GLACIATION

Green responsibLe privACy preservIng dAta operaTIONs

Deliverable D3.1 – Secure AI Enabled Placement Engine (Intermediate)

GRANT AGREEMENT NUMBER: 101070141

This project has received funding from the European Union's HE research and innovation programme under grant agreement No 101070141

Project acronym:	GLACIATION
Project full title:	Green responsibLe privACy preservIng dAta operations
Call identifier:	HORIZON-CL4-2021-DATA-01-01
Type of action:	RIA
Start date:	01/10/2022
End date:	30/09/2025
Grant agreement no:	101070141

D3.1 – Secure AI Enabled Placement Engine

Executive Summary:	D3.1 outlines the progress and status of research and development on WP3: AI-enabled Data movement Engine		
WP:	3		
Author(s):	Javad Chamanara		
Editor:	DELL		
Leading Partner:	LUH		
Participating Partners:	LUH, HIRO, DELL, ECOM, ETH, LAKE, SOGEI, UCC, UNIMI, ENG		
Version:	0.1	Status:	Ready
Deliverable Type:	OTHER	Dissemination Level:	(PU) Public
Official Submission Date:	M18 (March 2024)	Actual Submission Date:	M18 (March 2024)

Disclaimer

The GLACIATION consortium consists of the following partners:

No.	Partner Organisation Name	Partner Organisation Short Name	Country
1	MINISTERO DELL'ECONOMIA E DELLE FINANZE	MEF	IT
2	EMC INFORMATION SYSTEMS INTERNATIONAL UNLIMITED COMPANY	EISI (DELL)	IE
3	HIRO MICRODATACENTERS B.V.	HIRO	NL
4	GOTTFRIED WILHELM LEIBNIZ UNIVERSITAET HANNOVER	LUH	DE
5	THE LISBON COUNCIL FOR ECONOMIC COMPETITIVENESS ASBL	LC	BE
6	UNIVERSITA DEGLI STUDI DI MILANO	UNIMI	IT
7	UNIVERSITA DEGLI STUDI DI BERGAMO	UNIBG	IT
8	GEIE ERCIM	ERCIM	FR
9	EURECOM	EURECOM	FR
10	SAP SE	SAP SE	DE
11	UNIVERSITY COLLEGE CORK - NATIONAL UNIVERSITY OF IRELAND, CORK	UCC	IE
12	SOGEI - SOCIETÀ GENERALE D’INFORMATICA S.P.A.	SOGEI	IT
13	LAKESIDE LABS GMBH	LAKE	AT
14	ENGINEERING - INGEGNERIA INFORMATICA SPA	ENG	IT
15	EIDGENOESSISCHE TECHNISCHE HOCHSCHULE ZUERICH	ETH	CH

Document Revision History

Version	Description	Contributions
0.1	Table of Content	LUH
1.0	Partners provided content	ALL
1.1	UCC Reviewed	UCC
1.2	Lake Reviewed	LAKE
2.0	LUH addressed review items	LUH

Author

Author	Partner
Javad Chamanara	LUH
Oleksandr Chepizhko	LAKE
Péter Forgács	LAKE
Diego Carraro	UCC

Reviewers

Name	Organisation
Oleksandr Chepizhko	LAKE
Péter Forgács	LAKE
Ken Brown	UCC

Table of Contents

1. Introduction 11
2. Hardware optimized processing of distributed knowledge graphs 12
2.1 Objectives 13
2.2 Research 13
2.2.1 Findings 13
2.2.2 Technology Selection 14
2.2.3 Satisfying the GLACIATION Requirements 16
2.3 Advancements 17
2.3.1 Developments Performed 17
2.3.2 Current Status 18
2.4 Roadmap 18
3. AI/ML-based workload placement 19
3.1 Objectives 21
3.2 Research 21
3.2.1 Forecasting component 21
3.2.2 Technology Selection 26
3.2.3 Satisfying the GLACIATION Requirements 27
3.3 Advancements 27
3.3.1 Developments Performed 27
3.3.2 Current Status 27
3.4 Roadmap 27
4. Swarm intelligence-based orchestration 28
4.1 Objectives 28
4.2 Research 28
4.2.1 Findings 28
4.2.2 Technology Selection 30
4.2.3 Satisfying the GLACIATION Requirements 30
4.3 Advancements 30
4.3.1 Developments Performed 30
4.3.2 Current Status 31
4.4 Roadmap 31
5. Secure data management framework for AI 32
5.1 Objectives 32
5.2 Research 33
5.2.1 Findings 33

5.2.2 Technology Selection 36
5.2.3 Satisfying the GLACIATION Requirements 37
5.3 Advancements 39
5.3.1 Developments Performed 39
5.3.2 Current Status 40
5.4 Roadmap 40
6. Integration and Lab test 41
6.1 Objectives 41
6.2 Advancements 41
6.2.1 Developments Performed 41
6.2.2 Current Status 42
6.3 Roadmap 43
Conclusions 45

List of Tables

Table 1: GPU specification and server details composing the Alibaba cluster. 24
Table 2 Verdelix Requirements 32
Table 3 Big Data Success Factors 33
Table 4 Maturity Levels for Semantic Interoperability 34
Table 5 RDF Data Management Approaches 34
Table 6 RDF Data Management Technologies 35
Table 7 Big Data Management Solutions 35
Table 8 Taxonomy of Big Data Management...35
Table 9 Data Management Requirements 37
Table 10 Verdelix Requirements Alignment...38
Table 11 Verdelix Milestones..39
Table 12 Remaining Verdelix Milestones...40

List of Figures

Figure 1: Workload prediction and placement scheme 19
Figure 2: Results for scenario PM_{off} (Persistent Mode off). We report the ten trade-off values corresponding to different SLT in {0.9, 0.91, ..., 0.99} (from left to right) for each predictor. The models with the best trade-off should be on the top right hand of the plot. The red horizontal line indicates the 0.95 SLT, for reference. 25
Figure 3: Results for scenario PM_{on} (Persistent Mode On). We report the ten trade-off values corresponding to different SLT in {0.9, 0.91, .., 0.99} (from left to right) for each predictor. The models with the best trade-off should be on the top right hand of the plot. The red horizontal line indicates the 0.95 SLT, for reference. 26
Figure 4: Network structure. Three layers: core (orange), fog (green), edge (blue)...............29
Figure 5: Pheromone map of a network with 30 nodes. Red R marks nodes with resources that answer the query. 29
Figure 6: Hit rate as a function of time. The number of resources in network is given in legend. Network size is 64 nodes. The different colors are for the different total number of resources that answer the query in the system..30

List of Terms and Abbreviations

Abbreviation	Description
CC0	Creative Commons 0
C-LSTM	Long-short term memory neural network with inferred variance
DKG	Distributed Knowledge Graph
DMP	Data Management Plan
DoA	Description of the Action
DPIA	Data Protection Impact Assessment
DRI	Decentralised Resource Identifier
EAB	External Advisory Board
FAIR	Findable-Accessible-Interoperable-Reusable
GA	Grant Agreement
GDPR	General Data Protection Regulation
GEP	Gender Equality Plan
GLACIATION	Green responsibLe privACy preservIng dAta operations
GPU	Graphical Processing Unit
H2020	Horizon 2020
HBNN	Hybrid Bayesian Neural Network
HBNN++	HBNN extended with 2nd bayesian layer
HE	Horizon Europe
LSTM_Q	Long-short term memory neural network with quantile outputs
MILP	Mixed-integer linear programming
ML	Machine Learning
Mxy	Month xy of the project’s duration
PA	Public Administration
PM	Persistence Mode
R&I	Research and Innovation
RRI	Responsible Research and Innovation
SLT	Service-level target
SR	Success Rate
UCx	Use Case x
WP	Work Package

Executive Summary

This report details the development of the platform level software for the GLACIATION project in the context of WP3's key objectives:

- **Developing secure embedded software (Obj. 3.1)** for control, processing, acceleration, storage, and networking.
- **Enabling secure edge node connectivity (Obj. 3.2)** through research on hardware-optimized processing and network intelligence.
- **Resource management and service deployment (Obj. 3.3)** using machine learning for workload placement, distribution, and Deep Reinforcement Learning for dynamic edge environments.

Key functionalities achieved so far in the scope of the work package and in line with the project include:

- **Hardware-optimized processing:** efficient execution of knowledge graph processes on various platforms (CPU, GPU, FPGA).
- **Workload orchestration**: initial implementation of dynamic workload prediction and placement across edge and cloud resources based on demand prediction and constraints.
- **Swarm intelligence**: agent-based swarm for scalable and self-organizing edge/core/cloud orchestration.
- **Autonomous data management**: predicting the data movement based on usage and power consumption, with a focus on transparency, understandability, and ethical algorithms.
- **Data management framework**: Addressing data lifecycle, storage, provenance, sovereignty, and vocabularies for the edge environment.

1. Introduction

This document provides the status, selected technologies satisfying the project requirements, and the current development status as well as future roadmaps of the components that WP3 will contribute to the project.

Chapter 2 presents the result of their research and experiments regarding the hardware-optimized processing of Distributed Knowledge Graphs (DKGs), which contains a range of evaluated DKGs as well as various optimization possibilities.

Chapter 3 describes the AI/ML-based solution to workload placement, comprised of two main components. The first component is the forecasting model, which predicts the future workload demand. The second component, i.e. the orchestrator, plans and prepares the resources to meet the predicted workload and ultimately provide for the real incoming workload.

Chapter 4 elaborates on the distributed data search and data management (movement) in the dynamic environment of the DKG using a Swarm Intelligence-based Orchestration.

Chapter 5 is dedicated to the details of the development of a secure data management framework specifically tailored to the GLACIATION platform's edge computing focus. The task highlights the challenges of managing data in a distributed AI environment and suggests solutions.

Chapter 6 outlines the integration procedures, requirements, and protocols and provides the details of a lab test to enable all partners to formally declare their dependencies, announce their services, and provide containerized versions of the components. These components will go through a quality control process including dependency checking, container deployment, and dry runs.

2. Hardware optimized processing of distributed knowledge graphs

The emergence of AI-enabled placement engines has revolutionized decision-making processes in numerous domains, including logistics, supply chain management, and resource allocation. These engines leverage advanced algorithms to analyze vast amounts of data and derive optimal placement solutions. One promising approach involves the utilization of DKGs, which represent complex relationships among various entities. However, efficient processing of these graphs poses significant computational challenges, necessitating innovative hardware optimization techniques.

This section provides an overview of AI-enabled placement engines and the role of DKGs in enhancing their functionality. Additionally, it explores existing research on hardware optimization for graph processing tasks, highlighting the need for tailored solutions to address the unique characteristics of DKGs.

2.1
2.1 Objectives

1. Challenges in Processing Distributed Knowledge Graphs:

Processing distributed knowledge graphs involves dealing with immense volumes of interconnected data, leading to challenges such as high computational complexity, memory requirements, and communication overhead. This section delves into the specific challenges encountered when processing Distributed Knowledge Graphs within the context of AI-enabled placement engines.

2. Hardware Optimization Techniques:

We propose a set of hardware optimization techniques tailored to address the challenges associated with processing distributed knowledge graphs. These techniques encompass both architectural enhancements and algorithmic optimizations aimed at improving performance, scalability, and security.

2.1. Parallel Processing Architectures:

Utilizing parallel processing architectures, such as GPUs (Graphics Processing Units) and FPGAs (Field-Programmable Gate Arrays), can significantly accelerate graph processing tasks by exploiting parallelism at various levels. We explore the suitability of these architectures for handling the unique characteristics of DKGs and discuss optimization strategies tailored to maximize their efficiency.

2.2. Memory Hierarchy Optimization:

Efficient utilization of memory resources is critical for handling large-scale DKGs. We investigate techniques for optimizing memory hierarchies to minimize latency and maximize bandwidth, thereby improving overall system performance.

2.3. Network Communication Optimization:

Communication overhead poses a significant bottleneck in distributed graph processing systems. We propose optimizations at both the hardware and software levels to minimize communication latency and bandwidth utilization, facilitating seamless interaction among distributed components.

3. Experimental Evaluation:

To assess the efficacy of the proposed hardware optimization techniques, we conduct comprehensive experimental evaluations using real-world datasets and representative placement scenarios. Performance metrics such as processing time, scalability, and resource utilization are analyzed to quantify the benefits of hardware optimization in enhancing the efficiency of AI-enabled placement engines.

2.2 Research

2.2.1 Findings

Through our investigation into hardware-optimized processing of DKGs for secure AI-enabled placement engines, several significant findings have emerged. These findings shed light on the efficacy of hardware optimization techniques in addressing the challenges associated with graph processing while enhancing both performance and security aspects. Below are the key research findings:

1. Performance Enhancement:

Hardware optimization techniques, including parallel processing architectures such as GPUs and FPGAs, demonstrate substantial performance improvements in processing DKGs. These architectures exploit parallelism inherent in graph algorithms, resulting in accelerated computation and reduced processing times.

Memory hierarchy optimization plays a crucial role in improving performance by minimizing latency and maximizing memory bandwidth utilization. Techniques such as cache-aware data structures and memory prefetching contribute to more efficient memory access patterns, leading to enhanced overall system performance.

2. Scalability:

The scalability of AI-enabled placement engines is significantly enhanced through hardware optimization. Parallel processing architectures enable seamless scalability by distributing computation across multiple processing units, thereby accommodating larger graph sizes without sacrificing performance.

Optimization of network communication further facilitates scalability by minimizing communication overhead among distributed components. Efficient data exchange mechanisms ensure seamless interaction, enabling the placement engine to scale efficiently with increasing workload demands.

3. Security Enhancement:

Hardware optimization techniques offer inherent security benefits by leveraging hardware-based security mechanisms. Features such as hardware-based encryption and secure enclaves provide robust protection against security threats, including data breaches and unauthorized access.

By integrating security mechanisms at the hardware level, AI-enabled placement engines can mitigate vulnerabilities associated with distributed graph processing, thereby enhancing overall system security posture.

4. Real-world Applications:

The findings have significant implications for real-world applications where AI-enabled placement engines are deployed. Industries such as logistics, supply chain management, and resource allocation stand to benefit from improved performance, scalability, and security offered by hardware-optimized processing of DKGs.

The ability to process large-scale knowledge graphs efficiently enables more accurate and timely decision-making, resulting in optimized resource utilization, reduced costs, and enhanced operational efficiency.

2.2.2 Technology Selection

In the pursuit of hardware-optimized processing of DKGs for secure AI-enabled placement engines, careful selection of technologies is paramount to achieving desired performance, scalability, and security objectives. The following outlines the key technologies identified for realizing efficient and resilient placement engines:

1. Parallel Processing Architectures:

Graphics Processing Units (GPUs): GPUs offer massive parallel processing capabilities suitable for handling the computational demands of DKG processing. Their highly parallel architecture enables concurrent execution of graph algorithms across numerous processing cores, leading to significant performance gains.

Field-Programmable Gate Arrays (FPGAs): FPGAs provide customizable hardware acceleration tailored to specific graph processing tasks. Their flexibility allows for the implementation of custom graph algorithms directly in hardware, offering unparalleled performance and energy efficiency for complex computations.

2. Memory Hierarchy Optimization:

Cache-aware Data Structures: Utilizing cache-aware data structures optimizes memory access patterns, minimizing cache misses and reducing memory latency. Techniques such as graph partitioning and vertex reordering enhance locality of reference, improving overall memory hierarchy efficiency.

Memory Prefetching: Prefetching techniques anticipate memory access patterns and fetch data proactively, mitigating memory access latency. Hardware support for prefetching mechanisms enhances memory subsystem performance, particularly in scenarios with irregular memory access patterns inherent in graph processing.

3. Network Communication Optimization:

High-speed Interconnects: Leveraging high-speed interconnect technologies such as InfiniBand or Ethernet with Remote Direct Memory Access (RDMA) capabilities minimizes communication latency and overhead. RDMA facilitates direct data transfers between nodes without CPU intervention, improving communication efficiency in distributed environments.

Message Passing Libraries (MPI): MPI libraries offer optimized communication primitives for efficient data exchange among distributed components. Implementing communication protocols tailored to distributed graph processing minimizes synchronization overhead and maximizes network throughput.

4. Hardware-based Security Mechanisms:

Secure Enclaves: Secure enclaves provide isolated execution environments within the hardware, protecting sensitive data and computation from unauthorized access. Utilizing secure enclaves ensures confidentiality and integrity of critical operations, enhancing the security posture of AI-enabled placement engines.

Hardware-based Encryption: Hardware-accelerated encryption mechanisms safeguard data in transit and at rest, mitigating the risk of data breaches and unauthorized access. Integration of hardware-based encryption accelerators ensures robust data protection without compromising performance.

5. Software Ecosystem:

Graph Processing Frameworks: Leveraging scalable graph processing frameworks such as Apache Spark GraphX, Apache Giraph, or GraphLab accelerates development and deployment of distributed graph algorithms. These frameworks provide high-level abstractions for graph computation tasks, enabling seamless integration with hardware-optimized processing architectures.

Security Libraries: Integration of security libraries and protocols, such as OpenSSL or Intel SGX SDK, enhances the security posture of AI-enabled placement engines. These libraries offer standardized interfaces for implementing encryption, authentication, and access control mechanisms, ensuring compliance with security requirements.

6. Evaluation and Benchmarking Tools:

Graph Analytics Benchmarks: Utilizing benchmark suites such as Graph500 or LDBC Graphalytics facilitates performance evaluation and comparison of hardware-optimized graph processing systems. These benchmarks assess various aspects of graph processing, including traversal performance, scalability, and memory efficiency.

Performance Profiling Tools: Profiling tools such as NVIDIA Nsight Compute or Intel VTune Amplifier enable in-depth analysis of system performance and resource utilization. Profiling hardware performance counters and memory access patterns provides insights into potential bottlenecks and optimization opportunities.

2.2.3 Satisfying the GLACIATION Requirements

The GLACIATION requirements refer to the necessity of ensuring long-term storage and accessibility of historical data within AI-enabled placement engines. Meeting these requirements entails adopting strategies and technologies to preserve the integrity, availability, and usability of data over extended periods. Below are the key considerations for satisfying the GLACIATION requirements:

1. Data Archiving Policies:

Establishing robust data archiving policies is essential for preserving historical data while managing storage costs effectively. Define criteria for identifying data eligible for archiving based on relevance, retention periods, and compliance regulations.

Implement automated archiving mechanisms to systematically move inactive or historical data to long-term storage repositories, such as tape archives or cloud-based cold storage solutions. This ensures efficient utilization of primary storage resources while maintaining accessibility to archived data when needed.

2. Immutable Data Storage:

Adopting immutable data storage mechanisms safeguards against unauthorized modifications or deletions of historical data, preserving its integrity and auditability. Utilize technologies such as Write-Once-Read-Many (WORM) storage or blockchain-based ledgers to enforce data immutability.

Implement access controls and encryption mechanisms to secure archived data against unauthorized access, ensuring compliance with data privacy regulations and protecting sensitive information from unauthorized disclosure.

3. Data Lifecycle Management:

Implement comprehensive data lifecycle management strategies to govern the entire lifespan of data within the placement engine. Define clear policies for data ingestion, processing, retention, archiving, and eventual disposal in alignment with business requirements and regulatory mandates.

Utilize metadata management frameworks to catalog and index historical data, facilitating efficient retrieval and analysis. Metadata enrichment enhances the discoverability and contextuality of archived data, enabling users to extract valuable insights for decision-making purposes.

4. Data Retention Compliance:

Ensure compliance with data retention regulations and industry standards governing the preservation of historical data. Stay abreast of evolving regulatory requirements and adjust data retention policies accordingly to maintain compliance.

Implement data anonymization and pseudonymization techniques to protect the privacy of individuals whose data is archived within the placement engine. Anonymized data minimizes the risk of regulatory non-compliance while preserving the utility of archived datasets for analytical purposes.

2.3 Advancements

2.3.1 Developments Performed

In the dynamic landscape of AI-enabled placement engines, a series of significant developments have been executed to advance the capabilities and functionalities of these systems. These developments encompass a broad spectrum of technological innovations, algorithmic enhancements, and strategic initiatives aimed at improving the performance, scalability, and security of placement engines. Key developments performed include:

Algorithmic Refinements: Researchers and practitioners, globally, have undertaken extensive efforts to refine and optimize algorithms used within placement engines. These efforts include advancements in machine learning techniques, graph analytics algorithms, and optimization methods tailored to specific placement scenarios. By fine-tuning algorithms, placement engines can achieve higher accuracy, faster processing times, and better adaptability to diverse datasets and use cases.

Infrastructure Scaling: Significant investments have been made to scale the infrastructure supporting placement engines, both in terms of computational resources and data storage capacity. This scaling encompasses the deployment of high-performance computing clusters, cloud computing resources, and distributed storage systems capable of handling large-scale datasets and intensive computational workloads. Scalable infrastructure enables placement engines to process and analyze vast amounts of data efficiently, facilitating real-time decision-making and analytics.

Integration with Emerging Technologies: Developments in AI and related technologies, such as Natural Language Processing (NLP), computer vision, and deep learning, have been integrated into placement engines to enhance their capabilities. Integration with these emerging technologies enables placement engines to leverage unstructured data sources, extract valuable insights from multimedia content, and make more informed placement decisions based on comprehensive data analysis.

Performance Optimization: Continuous optimization efforts have been undertaken to improve the performance and efficiency of placement engines. This includes optimizing codebase, streamlining data processing pipelines, and leveraging caching mechanisms to reduce latency and improve responsiveness. Performance optimization initiatives ensure that placement engines can deliver timely insights and recommendations, even when operating under high-demand conditions.

2.3.2 Current Status

As of the present time, research in the field of AI-enabled placement engines is characterized by a dynamic landscape of exploration, innovation, and collaboration. The current status of research reflects a multidisciplinary approach, with contributions from computer science, artificial intelligence, operations research, and domain-specific fields. Key aspects of the current status of research in AI-enabled placement engines include:

Advanced Algorithms and Models: Researchers are continuously developing and refining algorithms and models tailored to the specific requirements of placement engines. This includes advancements in machine learning algorithms, graph analytics techniques, optimization methods, and probabilistic reasoning models. Novel approaches such as deep reinforcement learning, transfer learning, and ensemble methods are being explored to improve the accuracy, efficiency, and adaptability of placement engine algorithms.

Scalability and Efficiency: Scalability and efficiency remain crucial areas of focus in research, particularly with the increasing volume, velocity, and variety of data processed by placement engines. Research efforts are directed towards developing scalable algorithms, distributed computing architectures, and parallel processing techniques to handle large-scale datasets and complex computational workloads efficiently. Innovations in data partitioning, caching mechanisms, and distributed storage systems are aimed at reducing latency and improving responsiveness in placement engine operations.

2.4 Roadmap

Collaboration with our partners is fundamental to the successful completion of all components, milestone achievements, and the overall maturation of each element. By aligning closely with our partners, we ensure that integration begins seamlessly within our dedicated integration environment, tailored to the unique needs of each component. This collaborative approach fosters efficient communication, smooth workflow coordination, and ultimately accelerates the progress towards our shared goals.

3. AI/ML-based workload placement

This section describes our AI/ML-based solution to workload placement, comprised of two main components (see Figure 1). The first component is the forecasting model, which predicts the future workload demand. Based on such predictions, the second component, i.e. the orchestrator, plans and prepares the resources to meet the predicted workload and ultimately provide for the real incoming workload.

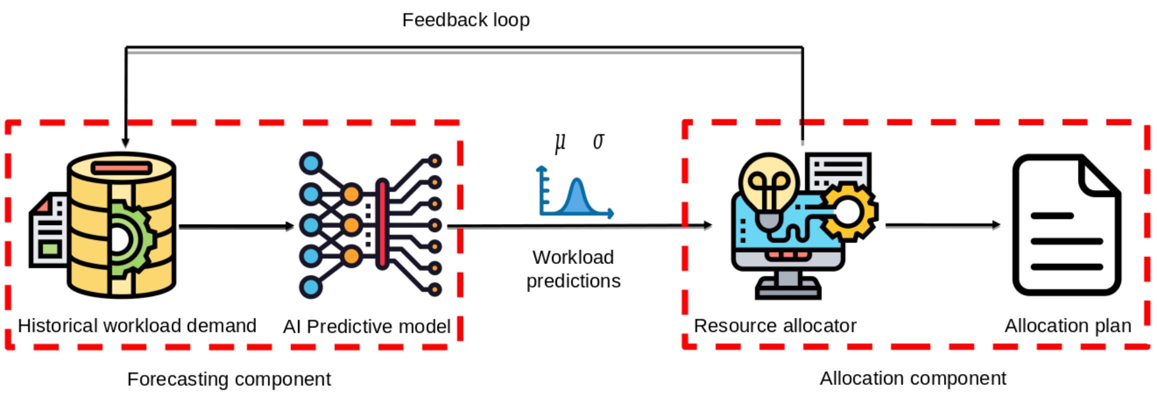


Figure 1: Workload prediction and placement scheme

3.1

3.1

3.1 Objectives

The main objective of this component is to forecast the incoming flux of workload and when arrived, to place them on various available worker nodes.

3.2 Research

3.2.1 Forecasting component

System managers and cloud orchestrators typically leverage future workload predictions to make informed decisions on resource/workload placement, where the ultimate goal of the allocation is to meet customers' demands while reducing the provisioning cost.

Workload forecasting is challenging due to workload patterns being highly irregular. The literature tackles the problem as a time series analysis task and has proposed state-of-the-art solutions using statistical, machine learning and deep learning approaches. These models often provide point estimate predictions for average future workload demand within a limited time window (typically five or ten minutes). However, recent research has proposed models that can provide uncertainty measures for such predictions, which tell how confident a model is about a prediction.

Uncertainty can help resource managers make more informed decisions when optimizing, for example, the performance vs energy trade-off, approaching the problem based on the degree of a prediction's confidence. For example, when a workload demand with low uncertainty is predicted, a proportion of the extra resources allocated to meet the predicted demand (i.e. the safety margin) can be placed in a low power state with minimal energy consumption (e.g. sleep state). When the uncertainty of predictions is high, the safety margin resources might be set to a more responsive state (e.g. idle state), sacrificing energy efficiency.

Motivated by the above considerations, we designed and implemented several ML-based uncertainty-aware workload forecasters. We also proposed an offline evaluation to assess the impact of uncertainty-aware predictions on the performance vs cost trade-off, where we express the cost in terms of energy savings. In the evaluation, we simulate two real-world cloud scenarios similar to and relevant to GLACIATION use cases, where an optimizer leverages workload predictions to allocate resources to satisfy demand while minimizing energy waste.

Our evaluation assumes the existence of N number of GPUs of different specifications distributed across several server machines I as resources available to cover workload demand. In particular, we consider *running* and *sleeping* states for a server machine. When sleeping, the server does not incur any energy cost, but its GPUs are not available for use. When running, its GPUs are available to cover the incoming workload demand, but there is an energy cost to keep the server awake. There is also the possibility of waking up a server, again with an energy cost. We consider three states for a GPU: *sleeping*, *idling* and *running*. When sleeping, GPUs do not incur any energy cost and are available for incoming workload with a negligible delay. When idling, GPUs consume a reduced amount of energy and are immediately available to cover the incoming workload. When running, GPUs cover a specific amount of workload and consume an energy cost depending on their specification.

We assume a workload predictor which predicts the average workload in a given time period t of length 5 minutes, i.e. W_t. The prediction is provided 10 minutes in advance to allow for resource allocation planning. The predictor is fed with the average historical workload of the previous 288 time windows, i.e. one day, that is used as the prediction's context.

Note that workload demand can be represented in different ways depending on the specific scenario at hand within the scope of GLACIATION. In our research, however, we represent workload as the aggregated average GPU workload demand appearing in the entire system/cluster in a 5-minute period t (expressed in GPU units per second).

At each period t, an independent allocator/orchestrator must allocate resources to cover the predicted workload demand W_t to meet a given Service Level Target (SLT), minimising the energy cost. In practice, it decides which servers and GPUs to run within the time window to cover the demand. If a required server is already running, there is no additional energy cost; if it is sleeping, it is woken up one minute before the period t starts and incurs the energy cost. Once the server runs, GPUs can be assigned to the incoming workload demand. Within t, we assume we cannot change the state of servers and GPUs based on real-time demand because the only prediction available is an average over the entire 5-minute window.

We implemented the allocation by solving an optimization problem where decision variables are:

- whether to run a server in t;
- whether to run a specific GPU in t.

And the following constraints:

- enough GPU computational power is allocated to meet demand $W_{t;}$
- if a server runs in period t, either it was already running in t-1, or it wakes up to run in the current period;
- a GPU is running only if its server machine is running too.

The problem can be solved optimally using a Mixed Integer Linear Programming (MILP) solver, which finds the expected energy cost if W_t is a predicted workload or an approximation of the actual energy cost if W_t is the real demand. In practice, the allocator must find a set of GPUs that are enough to service the upcoming request using the minimum amount of energy. The cost parameters and the constraints can be adapted to approximate the specific cluster system. The high number of binary variables (due to numerous GPUs and servers) makes it expensive for MILP to solve the problem.

Comparing different models on thousands of predictions requires considerable computational effort. To overcome this issue, we deployed a heuristic based on Dantzig knapsack problem algorithm that fixes which servers should be running to satisfy the predicted demand. This approach assumes that all the GPUs in a server machine are running workload, or none of them is, so there is no server using only part of their GPUs. The computational power a server provides can be seen as the volume, while the cost of waking and running it is the value/reward efficiency. The goal is to fill the knapsack while minimizing the total cost/reward. The solution found is an upper bound; if each server contains only GPU of one type, the energy cost before the last server is assigned is the lower bound. So, this heuristic over-allocates at most one server compared to the optimal solution; in instances with thousands of servers (where the computational cost of the MILP is excessive), the difference is negligible.

We evaluate the allocation in terms of a trade-off between performance and energy cost. We use Success Rate (SR), i.e. the percentage of times the upcoming workload is successfully covered by the allocation, to measure performance. To measure energy cost, we use the energy required to allocate the real demand with the servers allocated according to the predicted one: when under-predicting demand, the unmet demand is simply dropped (the energy cost will be rewarded, and the SR will be penalised); when over-predicting, the SR will be rewarded, and the energy cost penalised.

All models we describe below, except for LSTM-Q, predict a workload probability distribution of the future workload demand as a Gaussian distribution with parameters μ (i.e. mean) and σ (i.e. standard deviation). From this probability distribution, we can compute the confidence interval (CI) corresponding to the desired SLT and take its upper bound as the final workload prediction W_t. The LSTM-Q provides predictions naturally associated with a specific SLT, i.e. quantile-based, which we take as the final workload prediction.

- Hybrid Bayesian Neural Network (HBNN): is a Long Short-Term Memory (LSTM)-based neural network optimised via variational inference where the last layer is Bayesian and models the epistemic uncertainty. The network's output, i.e. the distributional layer, captures the aleatoric uncertainty.

- HBNN++: derived from the HBNN, this architecture presents two Bayesian layers. The first one replaces the 1DCNN layer of the HBNN, and it captures the aleatoric uncertainty. The second one is positioned as the last layer of the architecture, and its output provides a non-deterministic pointwise prediction. We run the inference step 30 times and compute μ and σ of the Gaussian distribution.

- LSTM Quantile (LSTM-Q): this LSTM's variation outputs quantiles, thus making the model distribution independent from the Gaussian assumption. The benefit of quantile optimisation is to make a model more robust to outliers (common in workload forecasting) and to capture the aleatoric uncertainty. We train different models for different SLTs.

- C-LSTM: from Rossi et al. 2022, this traditional LSTM-based model outputs μ (deterministically), while σ is inferred from the training set as the standard deviation that a confidence interval should have to meet a specific SLT (and this value is constant for all the predictions).

- Prophet: uses an additive model to best capture trends and seasonality in the data. It has already been used for cloud workload prediction, but its performance in probabilistic predictions is still under-investigated in this domain.

Inspired by the Persistence Mode (PM, where Persistence Mode is the term for a user-settable driver property that keeps a target GPU initialized even when no clients are connected to it) option of real-world GPU servers, we implemented and ran two scenarios and used the offline evaluation presented earlier to assess the effectiveness of the uncertainty-aware forecasting models. These scenarios differ based on the default status of the server's GPUs when the server is running:

- PM_{off}. When the persistence mode is *off*, we consider GPUs to be in a sleep state by default while the server runs. GPUs are wakened up accordingly to meet workload (with some negligible delay of a few seconds); when no demand is available, there is no energy cost to pay. This scenario simulates a medium-responsive system where applications do not require real-time computations. In this case, the GPUs do not have to be idling even if the server that contains them is on.

- $PM_{on.}$ When the persistence mode is *on*, we consider GPUs idling by default while the server runs. When demand is assigned to a GPU, the GPU is immediately available to meet workload (with no delay); when no demand is assigned, there is an idle energy cost to pay, i.e. 20% of the maximum computational power in our experiments. This scenario simulates a high-responsive system where applications require real-time computations, e.g. gaming.

We also implemented two baselines:

- *oracle*, predicting the true demand -- the allocator provides maximum SR with minimal energy consumption;
- *naive*, which always predicts the maximum workload that the whole system can cover. The allocator thus keeps all servers running for all periods, resulting in the maximum SR with maximum energy consumption.

We preprocessed Alibaba GPU cluster data and partition data in time intervals of 5 minutes. The resulting dataset is a time series where each data point represents the aggregated average GPU workload demand appearing in the entire cluster in a 5 minutes period t (expressed in GPU units per second). We crafted a training set of 20 days (5760 data points) and a test set of 8.6 days (2476 data points). We train forecasting models on the training set and test them on the test set. We also crafted specifications of the GPUs in the cluster and their distribution across server machines (see Table 1). From that, we estimated energy consumption for a GPU running, sleeping or idling, and we use expert knowledge to estimate the cost associated with running and booting up servers.

Name	# GPUs	# servers	# GPUs per server	Max. comp. power (units/s)	Max. power cons. (kWh)
P100	1596	798	2	5821.61	0.0208
T4	994	497	2	4158.29	0.0058
V100	1912	239	8	8316.59	0.0233
MISC	2240	280	8	6513.58	0.0208
Total	6742	1814			

Table 1: GPU specification and server details composing the Alibaba cluster.

The results of our experiments are depicted in **Figure** 2 and Figure 3. **Figure** 2 shows results for the PM_{off} scenario, where the oracle achieves a 100% SR and a relatively small energy savings of 5.5%. Prophet is good at meeting its SLTs but is the worst regarding energy savings. Interestingly, all the other models can save more energy but at the expense of some SR percentage points. All the models show balanced and interesting trade-offs. All models are relatively close in energy savings (i.e. within 5.1% and 6.2%), but HBNN++ has the best trade-off (e.g. furthest right on the red horizontal line showing a 0.95 SLT) because it can meet most of its SLT while saving energy.

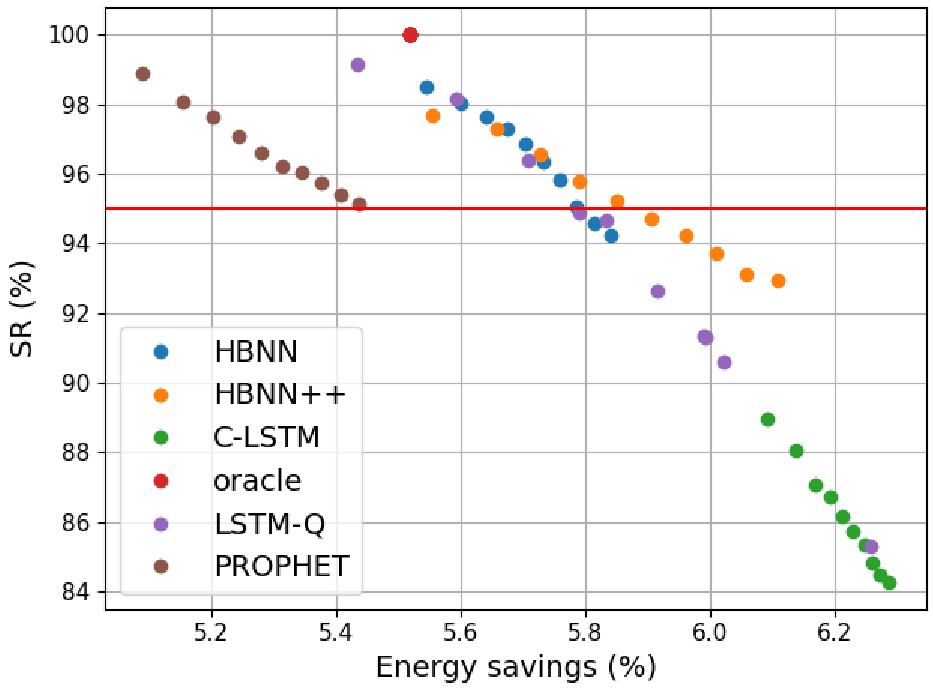


Figure 2: Results for scenario PM$_{off}$ (Persistent Mode off). We report the ten trade-off values corresponding to different SLT in {0.9, 0.91, ..., 0.99} (from left to right) for each predictor. The models with the best trade-off should be on the top right hand of the plot. The red horizontal line indicates the 0.95 SLT, for reference.

Figure 3 shows results for the PM$_{on}$ scenario, where the oracle achieves a 100% SR and energy savings of 30.1%. Prophet and C-LSTM behave similarly to the previous scenario and show less balanced trade-offs. Trade-offs for the other models are more subtle. LSTM-Q appears to provide greater energy savings, but it consistently undershoots its target SLTs. HBNN provides the best performance, as it consistently exceeds its SLTs while saving between 27% and 28% of the energy costs.

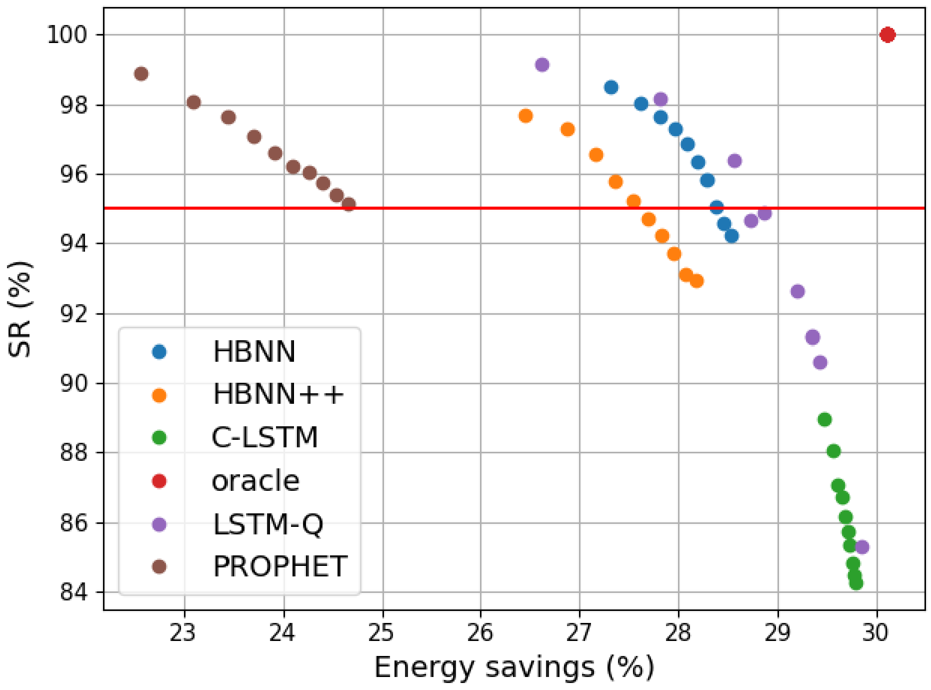


Figure 3: Results for scenario PM$_{on}$ (Persistent Mode On). We report the ten trade-off values corresponding to different SLT in {0.9, 0.91, .., 0.99} (from left to right) for each predictor. The models with the best trade-off should be on the top right hand of the plot. The red horizontal line indicates the 0.95 SLT, for reference.

These results can be further explained in terms of under-allocation (undershooting the prediction) and overa-llocation (overshooting the prediction). In the first scenario, over-allocation has little impact on energy cost because it does not waste much energy (only a small, fixed cost for running a server), while the opposite is true for the second scenario (due to idling GPUs that waste more energy). Under-allocation always influences the trade-off, as under-allocating resources penalises SR but saves energy. Thus, in the first scenario models that overshoot predictions.

In conclusion, we showed that the proposed forecasting model could be adapted to fit different data centre settings. We provided a heuristic that quickly computes a near-optimal policy, making it usable for comparing prediction models on large datasets. Results show that our framework provides insights into the trade-off between service performance and energy cost of different predictors. Our analysis reveals that different predictions (with uncertainty) lead to different trade-offs under different scenarios (although they are quite consistent across different SLTs). While HBNN++ shows the best trade-off for PM$_{off}$, HBNN offers the best trade-off for PM$_{on}$.

3.2.2 Technology Selection

The components are developed mainly in Python, with the different ML models and training pipelines also in Python. However, the cluster status is obtained from telemetry system, synthetic data generators, and a temporal graph database that is built upon Neo4J.

3.2.3 Satisfying the GLACIATION Requirements

The forecasting and placement modules are working in experimental environments. Integrating them with Kubernetes scheduling mechanism, other platform services such as the DKG, the telemetry and power instrumentation components is under investigation and development. Also, integrating the component with the security system of the data movement engine as well as with the privacy enforcement component is under research.

3.3 Advancements

3.3.1 Developments Performed

Experimental setups are in place.

The placement module is capable of finding the proper worker node; however, the prediction accuracy is not stable yet. Also, the power consumption is not included in the selection criteria so far, mainly because the power consumption datasets at hand are not realistic.

3.3.2 Current Status

The placement module is a system of 4 containers that work as a workflow consisting of a data acquisition step, data transformation step, a temporal graph indicating the cluster (data center) status, and the placement model for training and serving phases. All the containers and the workflow are up and running.

3.4 Roadmap

The placement module must be trained using power data that is planned to be provided by DELL and HIRO. Also, the security and privacy interfaces are planned to be integrated into the Kubernetes scheduling pipeline, namely sorting and filtering steps.

4. Swarm intelligence-based orchestration

4.1 Objectives

The data search and data management (movement) in the dynamic environment of the Distributed Knowledge Graph (DKG) is a challenging problem. In the context of Swarm Intelligence-based Orchestration, we will refer to the devices containing a part of the DKG as nodes. To move data efficiently between nodes from where it is stored to where it is most frequently needed is a task that requires multilevel assessment. Here we discuss a possible solution to this problem based on the Ant Optimization Search Algorithm. We facilitate a search for results of a query in a DKG over a network using pheromone abstraction. Simple rules to move data from a node to a node can be devised utilizing the pheromone landscape emerged because of the execution of the search algorithm. Thus, search algorithm and data movement become entangled.

Below we describe our research and development regarding the search algorithm as the basis for our further design of data movement. The data movement development will be discussed in future deliverables.

We give a general overview of design principles and research results. A more detailed and technical manuscript is now under preparation to be submitted for a special scientific conference.

4.2 Research

4.2.1 Findings

To study a network configuration relevant in cloud-fog-edge context we came up with a three-layered tree-like topology. The core level represents cloud servers which are connected to each other in a full mesh. The intermediate level represents “fog” devices in between cloud and edge. The third level - the leaves of the tree-like structure – represent edge devices. This network structure is illustrated in Figure 4.

We employ the ant optimization mechanism to facilitate the search in such a network. Incoming query is represented by a Forward Ant abstraction. Forward Ants move through network following pheromone trails. The pheromone trails are created by Backward Ants. Once a Forward Ant finds a requested resource it sends a Backward Ant which drops pheromones on each node between the found resource and the query source.

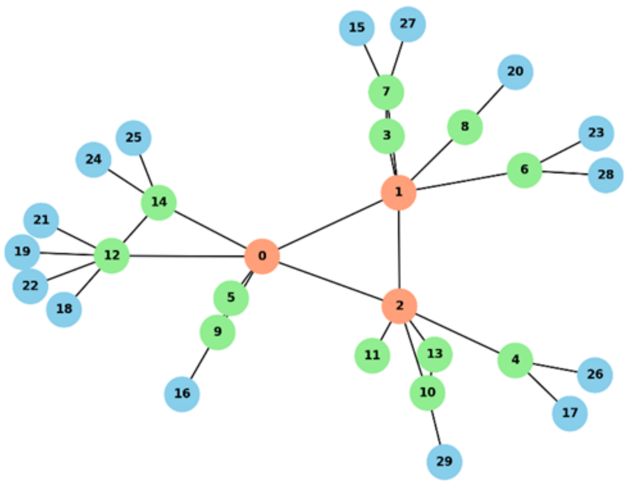


Figure 4: Network structure. Three layers: core (orange), fog (green), edge (blue).

The resulting pheromone map can be visualized if we color each node according to the total pheromone level it possesses, however discarding the directional information for the sake of simplified representation. An example of such a pheromone map is shown in Figure 5.

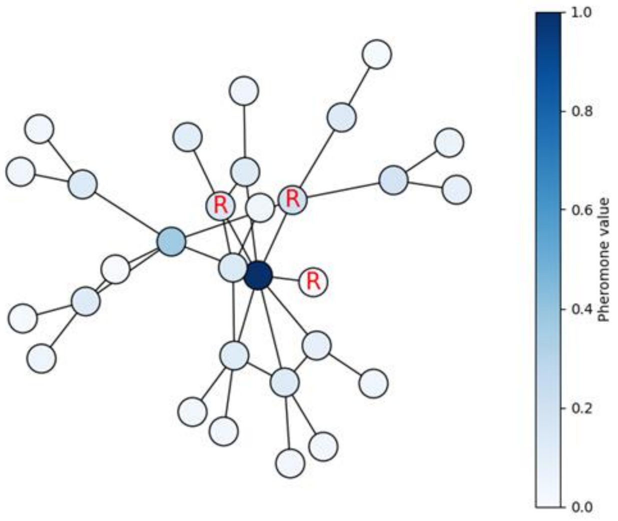


Figure 5: Pheromone map of a network with 30 nodes. Red R marks nodes with resources that answer the query.

To assess the efficiency of a search algorithm different measures can be employed. For the moment we have focused on the hit rate – ratio of the resources found during a query to the total amount of resources in the network and the number of visited nodes. In our studies we vary:

- pheromone evaporation coefficient – special tuning parameter to let the pheromone routes be removed with time and thus for the search algorithm to avoid being trapped within a sub-optimal solution;
- probability of switch between exploration and exploitation strategies is another parameter of interest.

Exploration strategy means that ants try to build new routes, while for exploitation strategy they stick to the strongest existing pheromone direction.

The time series of hit rate serves as primary data to be analyzed. An example of this time series is shown in Figure 6 for low values of pheromone evaporation and exploitation strategy probability.

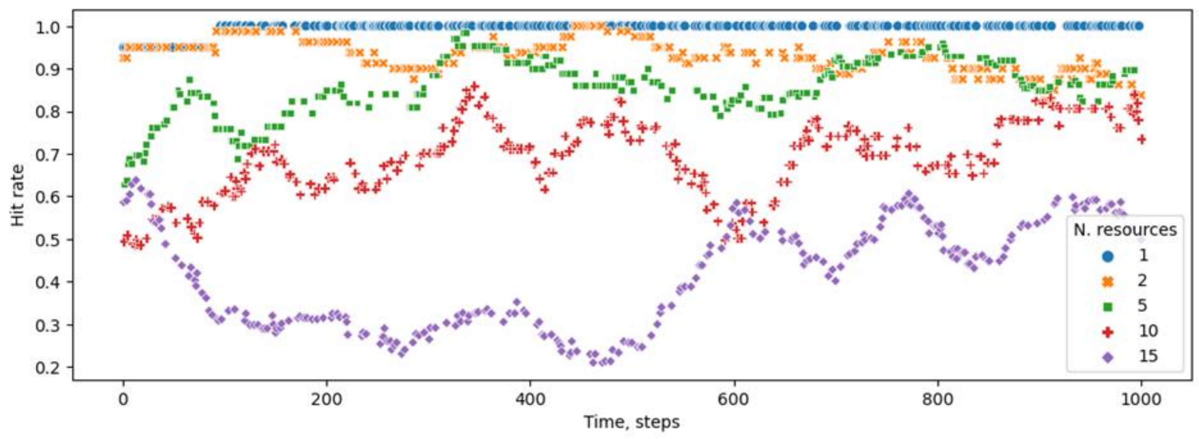


Figure 6: Hit rate as a function of time. The number of resources in network is given in legend. Network size is 64 nodes. The different colors are for the different total number of resources that answer the query in the system.

Further statistical analysis of this and similar time series is an ongoing investigation and will be presented in a manuscript which is right now in preparation.

4.2.2 Technology Selection

The swarming technology behind the ant optimization algorithm is well-developed at the Lakeside Labs and serves as the essential part of the competences. The algorithm is realized using MESA, an agent-based simulation package for Python programming language. The choice of Python pursues two goals. The first goal is to streamline the simulation stage and data analysis profiting from an extensive toolset of libraries available for Python. The second goal is to facilitate integration with the whole GLACIATION framework where the algorithm will be used for search on a network of real devices.

4.2.3 Satisfying the GLACIATION Requirements

We satisfy the GLACIATION requirements by providing energy efficient search solution that is very adaptive and can allow for seamless processing of privacy data concerns.

4.3 Advancements

4.3.1 Developments Performed

We developed simulation code for two network topologies. The first one was a half-mesh topology which we rendered unrealistic after initial assessment. Next, we have been developing a tree-like topology. Also, we are working on adopting our code to share with other partners for the purpose of integration into the GLACIATION framework.

4.3.2 Current Status

Currently simulations are up and running. We collect data and perform detailed statistical analysis. The goal of the analysis is to find the parameter values, which maximize desired outputs of the search algorithm. Also, we are studying which configurations are best suited for ant optimization algorithm and which configurations pose challenges to it.

4.4 Roadmap

Before the next deliverable we expect to have a full understanding of the search algorithm. This would result in publication of obtained results in scientific literature. Another important goal will be to develop the prototype of data movement mechanism. Search and data movement mechanisms working together in a synergetic manner would be our primal goal.

5. Secure data management framework for AI

This section details the development of a secure data management framework specifically tailored to the GLACIATION platform's edge computing focus. The task it is associated with, Task 3.5, highlights the challenges of managing data in this distributed AI environment. This section also details the solutions chosen to satisfy the requirements of managing data in this environment. Finally, we provide the current state of the implementation and roadmap of the Secure data management framework. Furthermore, the software solution implemented for the Secure data management framework for AI has been given the internal codename "Verdelix". The name is a synthesis of two core concepts: Verde: Derived from the Spanish and Italian word for "green," which implies environmental, sustainable, or eco-friendly connotations and Helix: A term used to describe the spiral structure of a double-stranded DNA molecule in biology. The helix can metaphorically represent structure, continuity, and interwoven elements. In a broader sense, it may signify intertwined data or a structured approach. Together, Verdelix suggests an eco-friendly, structured, and possibly intertwined or integrated approach, especially when applied to data or technology. It carries both an environmental consciousness and a sense of intricate, yet structured design. The name encapsulates the idea of a sustainable, green approach to structured or complex systems.

5.1 Objectives

Big data has a number of challenges (Wang, 2017). Big data generated by edge networks differs significantly from traditional data. Its rapid inflow, semi-structured/unstructured nature, distributed sources, and often real-time access requirements present unique complexities. Furthermore, edge devices may exhibit strong correlations across dimensions like time and location, and they may form social relationships. Content sharing among devices in a peer-to-peer fashion adds another layer of complexity, highlighting the need for sophisticated analytical methods to extract meaningful insights. Cloud-based analytics can handle the scale of IoT big data but face limitations due to its distinct structure, real-time demands, and the weak semantics of raw sensor data. The data analytics lifecycle involves raw data collection, aggregation, and transmission to the cloud. A critical goal lies in harnessing the patterns within edge big data (e.g., user behavior, data correlations) to improve network performance. Data interpretation remains a core challenge, requiring techniques to handle real-world noise, uncertainty, and the need for reliable sensor calibration to ensure the trustworthiness of decisions based on this data.

Each of the requirements for Verdelix software comes with its own challenges that the proposed solution must account for.

Table 2 Verdelix Requirements

Category	Requirement
Data Lifecycle Management	Data Availability: The framework shall enable the seamless provision of data to distributed AI algorithms running on edge compute resources with minimal latency.
	Data Destruction: The framework shall provide secure and automated mechanisms for data destruction at edge nodes while adhering to data retention policies. This must function in a dynamic environment with fluctuating compute resources.

	Access Control: The framework shall implement granular data access policies, governing user and algorithm permissions, and support auditing of access patterns.
	Cloud Archival: The framework shall determine when data is no longer actively needed at the edge and facilitate secure archival to cloud storage, complying with any applicable regulations.
Distributed Storage	Data Accessibility: The framework shall ensure data is accessible when and where required by edge AI algorithms, taking into account network constraints.
	Placement Optimization: The framework shall intelligently determine optimal data storage locations (edge, fog, or cloud) based on usage patterns, latency requirements, and security considerations.
	Replication and Resilience: The framework shall employ replication strategies to maintain fault tolerance, offering a configurable trade-off between redundancy and storage efficiency across edge resources.
Data Provenance	Tracking: The framework shall maintain a comprehensive history of data origin, transformations, and usage to support auditing and establish trust in AI outcomes.
Data Sovereignty	Ownership: The framework shall guarantee data producers retain ownership of their data throughout its lifecycle.
	Usage Control: The framework shall implement mechanisms for data producers to enforce strict control over how their data is accessed and processed, respecting jurisdictional and project-specific compliance requirements.
Data Vocabulary	GLACIATION Standardization: The framework shall define and manage consistent data vocabularies, including metadata and ontologies, to enable semantic interoperability within the GLACIATION platform.

5.2 Research

The Verdelix implementation relies on literature reviews, state-of-the-art reviews, and systematic reviews that already exist in various journals and conference proceedings as well as a review of existing data management solutions that exist in open-source form.

5.2.1 Findings

Based on reviewing (Al-Sai, 2020) (Antonios, 2023) (Özsu, 2016) (Siddiqa, 2016), we have made the following findings (see Table 2) relevant to the development of the Verdelix software. (Al-Sai, 2020) provides findings which require the focus on the success categories illustrated in Table 3 Big Data Success Factors.

Table 3 Big Data Success Factors

Category	Focus
Technology	Tools and systems for data collection, storage, processing, analysis, and applications. Emphasizes system performance, real-time availability, data quality, and integration with existing tools.
Data Management	Administrative processes involved in capturing, processing, validating, storing, and protecting data to ensure its secure accessibility, reliability, and timeliness.

Category	Focus
Governance	Social activities, processes, practices, and policies to manage Big Data projects effectively. Ensures appropriate conduct and compliance with legal frameworks and data security measures.

Furthermore, there are also factors based on the reliance of the GLACIATION platform on semantic interoperability and the Table 4 use of a Novel Metadata Fabric that must be accounted for. (Antonios, 2023) presents the maturity levels relevant to semantic interoperability solutions which are presented in Table 4 Maturity Levels for Semantic Interoperability.

Table 4 Maturity Levels for Semantic Interoperability

Level	Description
Level 1 – Defined	Modeling abilities prepared for semantic interoperability, with solutions available in isolated places and limited scalability.
Level 2 – Coordinated	Basic modeling abilities, with some data and integration management. Solutions available in a limited number of areas with basic scalability.
Level 3 – Integrated	Well-defined modeling abilities incorporating data and integration management. Solutions widely available and good scalability.
Level 4 – Optimized	Well-defined and optimized modeling abilities, with seamless incorporation of data and integration management. Highly scalable and optimized for performance.

(Özsu, 2016) presents the challenges and approaches to RDF Data Management which aligns with our Novel Metadata Fabric approach. Specifically, Table 5 RDF Data Management Approachespresents the challenges in RDF Data Management and also the technologies are presented in Table 6 RDF Data Management Technologies.

Table 5 RDF Data Management Approaches

Approach	Description
Cloud-based Approaches	Leveraging cloud platforms and MapReduce for managing RDF datasets, using HDFS for storage and MapReduce for query processing.
Partitioning-based Approaches	Dividing an RDF graph into fragments placed at different sites, with each site hosting a centralized RDF store and processing subqueries locally.
Federated Systems	Running SPARQL queries over multiple SPARQL endpoints, with precomputed metadata for each endpoint and decomposed subqueries.

Approach	Description
Partial Query Evaluation Approaches	Utilizing partial function evaluation for distributed SPARQL processing, where each site executes the query on its local RDF graph fragment.

Table 6 RDF Data Management Technologies

Technology	Description
Ontologies	For knowledge representation, structuring data for better accessibility and understanding.
Linked Open Data (LOD)	Connecting datasets across the web, making data more interlinked and usable.
Knowledge Graphs	Integrating and managing data through a graph-based structure, facilitating complex data queries and analysis.
Semantic Annotation	Enriching data with metadata for improved context and relevance, aiding in data discovery and utilization.
Semantic Reasoning	Inferring new knowledge from existing data, enabling more intelligent decision-making and insights.

Finally, (Siddiqa, 2016), presents solutions for big data management, Table 7 Big Data Management Solutions, and a taxonomy in Table 8 Taxonomy of Big Data Management.

Table 7 Big Data Management Solutions

Problem Domain	Solution
Clustering	SOHAC, K-Mean Algorithm, Artificial Bee Colony (ABC) optimization
Replication	Fuzzy Logic, ABC optimization algorithm, Dynamic Data Replication
Indexing	Composite Tree, Support Vector Machine, Fuzzy Logic
Transmission	Wavelength Division Multiplexing, Orthogonal Frequency-Division Multiplexing, Traffic Separating
Cleansing	BIO-AJAX, Minimum Covariance Determinant, Conditional Functional Dependencies
Classification	Statistics, Decision Trees
Prediction	Neural Network, Fuzzy Logic, Support Vector Machine
Privacy	Privacy-preserving cost reducing Heuristic Algorithm, Portable Data Binding, Expectation-maximization algorithm
Integrity	Scalable PDP, POSD, PDP
Confidentiality	DES, Triple DES, RC4
Availability	Erasure coding, Replication

Table 8 Taxonomy of Big Data Management

Component	Sub-components	Key Focus
Data Storage	Clustering, Replication, Indexing	Optimizing storage space and ensuring efficient data retrieval.

Component	Sub-components	Key Focus
Pre-processing	Transmission, Cleansing	Improving data quality and preparing data for analysis.
Processing	Classification, Prediction	Analyzing and making predictions from processed data.
Security	Privacy, Integrity, Confidentiality, Availability	Protecting data against unauthorized access, ensuring data integrity and availability.

Based on our review of existing literature, Verdelix must adopt a secure data management framework that emphasizes the success factors outlined in Table 2 Verdelix Requirements (technology, data management, governance). To leverage the GLACIATION platform, it's crucial to target high levels of semantic interoperability maturity (Level 3 - Optimized). The framework should incorporate suitable RDF data management approaches (Table 5 RDF Data Management Approaches), potentially combining cloud-based, partitioning, federated, and partial query evaluation methods. Lessons from big data management solutions (Table 7 Big Data Management Solutions) can be applied, while proactively addressing security across privacy, integrity, confidentiality, and availability (Table 8 Taxonomy of Big Data Management).

5.2.2 Technology Selection

Solutions which satisfy the requirements presented in the previous section exist in published literature. Based on a systematic review of such solutions, the closest which satisfies the Verdelix requirements as well as aligning with the overall GLACIATION architecture and tools is presented in (Wamhof, 2023). The paper discusses the Agri-Gaia project, which aims to enhance metadata management and asset exchange in agriculture through a distributed, open data architecture. It introduces a federated ecosystem architecture based on Gaia-X principles, emphasizing data sovereignty and secure data exchange. The approach utilizes an ontology-based metadata management system, leveraging RDF for a flexible, extensible metadata graph. This system supports dynamic metadata extensions, allowing data and service providers to adapt metadata as needed. The architecture also incorporates technology decisions for data storage, metadata storage, and data exchange, using tools like Apache Jena Fuseki for RDF graph storage and MinIO S3 for dataset storage. The paper highlights the importance of controlled vocabularies and ontologies for unambiguous, universal data descriptions, contributing to a common understanding and facilitating data exchange in the agricultural sector.

The architecture of the Agri-Gaia project is based on a federated approach that integrates several platforms, a marketplace, and a dataspace authority to foster a robust and sovereign ecosystem for data storage, processing, and exchange in agriculture. At its core, Agri-Gaia employs a federated architecture that includes multiple platforms providing sovereign data storage and processing services. This setup allows farmers to contribute data through any platform, while developers can use these platforms to process data and train AI models. Data storage is typically offered as object storage, turning each platform into a data lake. Metadata management within each platform is facilitated through an ontology-based data catalogue that describes the assets available on the platform. RDF-based graph databases are utilized to store the data catalogue, ensuring a flexible and extensible metadata management system. The project also focuses on the integration with the Gaia-X European data space. Such integration is also desirable for GLACIATION.

Apache Jena Fuseki is discussed in more detail in Work Package 6 which leverages Jena as the technology underpinning the Distributed Knowledge Graph. However, to ensure completeness of this deliverable, a short description of Apache Jena Fuseki is provided here. Apache Jena Fuseki is a server for RDF data, facilitating the storage and querying of metadata in a graph format. It supports SPARQL, an RDF query language, enabling the efficient retrieval and manipulation of stored data, including the GLACIATION metadata model and Shapes graph, directly within the Apache Jena Fuseki storage environment.

Following (Wamhof, 2023) MinIO is utilized as a scalable object storage server within the GLACIATION project, managing datasets and models in the form of buckets. This storage solution enables the handling of various data types, such as image and tabular datasets, without the need to incorporate additional technologies. MinIO's compatibility with S3 storage standards allows for efficient data management and interaction through different SDKs, supporting a wide range of use cases in the agricultural ecosystem.

GLACIATION also has strict privacy and security requirements. Open Policy Agent has been chosen to assist with satisfying these requirements. Open Policy Agent (OPA) is an open-source policy engine that enables unified policy enforcement across a wide range of software systems. It decouples policy decision-making from policy enforcement, allowing users to author policies in a high-level language and then query these policies at runtime. OPA utilizes policies written in Rego, a declarative query language designed specifically for expressing policies over complex hierarchical data structures. It is particularly useful for controlling access to data and APIs, ensuring compliance with specific rules, and automating policy enforcement across different layers of an infrastructure. Policy enforcement is implemented by utilizing OPA to choose appropriate tooling from Work Package 4 to enforce the policy. It should be noted that the RDF triple store follows a Partitioning-based Approach, presented in Table 5 RDF Data Management Approaches.

5.2.3 Satisfying the GLACIATION Requirements

Data Management requirements are presented in Table 9 Data Management Requirements. While requirements validation is out of scope for this deliverable (requirements validation is a task within Work Package 7), they are presented here to provide context with regards the technologies chosen and how those technologies will satisfy the GLACIATION requirements.

Table 9 Data Management Requirements

Requirements ID#	Requirements	Comments
REQ_001	Optimize data consumed near producer with distribution of data and sending back statistical data or essential results	To improve efficiency in computational and energetic resources required by the service. Saving in centrally large amount of data
REQ_007	Data lifecycle management according to policies and regulation (GDPR, GRI and Italian regulation)	So that improve or at least maintain actual security, compliances and rules
REQ_010	Services availability (24/7) and resiliency management	Governance and risk control, based on data propriety and privacy and incident management rule
REQ_011	Store only needed data for ML/AI workload placement and energy saving	Avoid unnecessary data
REQ_012	Ensure data protection with backup	Avoid losing important data.

REQ_019	Manufacturing cloud computing	Must be able to access and analyze the big data and deploys it for Manufacturing to offer better manufacturing services
REQ_020	Data movement	Should provide fast, efficient and secured movement of data generated from robots, cobots, other machines
REQ_021	Data wrapping	Ingestion mechanism should implement data wrapping functions as per the defined data governance model and different levels of sensitivity and risk, ensuring minimal latency
REQ_022	Data assessment	Should allow for data assessment of completeness ensuring data quality
REQ_023	Real-time data streaming	Ingestion mechanism should support real-time stream data handling and ingestion from multiple concurrent sources
REQ_024	Batch handling	Ingestion mechanism should support batch data handling, which will enable the best level of automation.
REQ_025	Data sharing / restrictions	Platform should allow data providers to share data or restrict with a particular data consumer or with multiple stake holders as needed while protecting sensitive data
REQ_027	Distributed data processing	Must have a distributed data processing system to enable data processing and analytics across multiple nodes or clusters to handle large volumes of data
REQ_031	Data ownership and control	Must provide clear data ownership and control mechanisms to ensure that each factory maintains control over its own data
REQ_032	Deletion process	Platform should provide guarantees over data deletion following necessary protocols
REQ_033	Policy configuration	Policies for data sets and platform users should be configurable by the data provider
REQ_034	Data Protection	Data must be protected at rest and in transfer, parameters should be configurable by the data owner protecting data controller rights
REQ_035	Data Risk assessment	System should be designed to evaluate the data provided by each contributor to determine the relative sensitivity of the data elements provided
REQ_036	Data Governance & License models	Platform should define data governance models for the data sets with clear language and set of definitions and must support various license model
REQ_037	Data Storage	Must be capable of secure and reliable storage, including local, cloud, or hybrid and retrieval of large volumes of data from edge computing devices and other systems
REQ_038	Storage architecture	Must have a scalable storage architecture that will consolidate and share data with every smart manufacturing application, including AI, computer vision and data analytics

Our technologies satisfy the data related requirements as illustrated in Table 10 Verdelix Requirements Alignment.

Table 10 Verdelix Requirements Alignment

Requirement ID	MinIO	Apache Jena Fuseki	Open Policy Agent	Interactions
REQ_001	Partial	Partial		Analytics Component, Swarm Intelligence Data Orchestrator
REQ_007			Yes	
REQ_010	Partial		Partial	
REQ_011	Yes			
REQ_012	Yes			
REQ_019	Yes	Yes		
REQ_020				Swarm Intelligence Data Orchestrator

REQ_021				Data wrapping provided by WP4
REQ_022				Data semantification & normalisation
REQ_023	Partial	Partial		Swarm Intelligence Data Orchestrator, Apache NiFi
REQ_024	Partial	Partial		Swarm Intelligence Data Orchestrator, Apache NiFi
REQ_025			Yes	
REQ_027	Partial	Partial		Distributed processing framework
REQ_031			Yes	
REQ_032			Yes	
REQ_033			Yes	
REQ_034			Yes	
REQ_035			Yes	Rego policy
REQ_036			Yes	Rego policy
REQ_037	Yes	Yes		
REQ_038	Yes			

5.3 Advancements

5.3.1 Developments Performed

The approach taken in development of the Verdelix software follows the methodology used in the integration tasks Task 2.4 and Task 7.2, more specifically breaking the development into milestones. The milestones for Verdelix are presented in Table 11 Verdelix Milestones.

Table 11 Verdelix Milestones

Milestone	Focus	Key Deliverables	Outcome
Milestone 1: Service Design and Planning	Understanding service requirements and interactions	Service Overview, Service Dependencies Diagrams, Service Architecture Diagrams	Shared understanding, reduces integration issues later
Milestone 2: API Contract and Data Model	Formalizing how services communicate, and the data structure	Domain Model/Terminology, REST/OpenAPI Specification	Parallel development, basis for automated testing
Milestone 3: Service Stubs and Integration Testing	Creating basic service versions to test in an isolated environment	Service Stubs (basic API, likely hard-coded responses), Integration Environment (even if basic)	Early detection of integration mismatches
Milestone 4: Demo-Ready Component	Showcase progress with a mature component	Demo-ready component (deployable in partner environment)	Demonstrates tangible progress, validates approach

5.3.2 Current Status

Verdelix has currently achieved Milestone 4 and has its own repository in the GLACIATION github. Furthermore, it has been deployed within the Use Case 2 (Dell Manufacturing) validation platform.

5.4 Roadmap

The remaining milestones for the Secure Data Management Framework for AI are presented in Table 12 Remaining Verdelix Milestones.

Table 12 Remaining Verdelix Milestones

Milestone	Description	Key Deliverables	Outcome
Milestone 5: Component Development & Unit Testing	Build out full component functionality, conduct isolated testing	* Fully implemented components * Unit test suites with good coverage	Robust, well-tested building blocks
Milestone 6: End-to-End Integration & Testing	Connect components, test system-wide data flow	* Dev environment deployment setup * End-to-end test scenarios	Identifies integration issues, baseline functionality
Milestone 7: Performance Testing & Optimization	Measure system performance, optimize where needed	* Performance benchmarks * Optimization strategies (code, hardware, architecture)	System meets performance requirements
Milestone 8: Staging Deployment & User Acceptance	Deploy in a production-like environment for user feedback	* Staging environment setup * User acceptance test plans * Feedback and bug tracking	Validates real-world needs, catches pre-production issues
Milestone 9: Use Case Deployment & Monitoring	Release the system live, establish monitoring	* Production deployment plan + rollback * Monitoring dashboards	System serves users, insights into usage
Milestone 10: Validation Against Requirements	Assess the deployed system against initial requirements	* Test report mapping requirements to cases * Stakeholder sign-off (if successful)	Documented success or identification of needed adjustments

The next milestones focus on transforming a demo-ready component into a fully deployed and validated solution. This involves thorough component development and unit testing to ensure individual pieces are robust. Then, end-to-end integration testing will examine how components work together as a system. Next, rigorous performance testing and optimization will ensure the system can handle expected loads and meet performance targets. A staging deployment, closely mirroring the production environment, will be crucial for user acceptance testing to gather feedback and address issues before going live. The penultimate milestone involves the production deployment itself, along with establishing robust monitoring to ensure ongoing health and performance of the system. Finally, a thorough validation against the initial use case requirements will assess the project's overall success and identify any remaining areas for improvement.

6. Integration and Lab test

6.1 Objectives

This encompasses the integration and adaptation efforts necessary for various modules, components, languages, APIs, and algorithms within the GLACIATION framework. The aim is to create a functional prototype suitable for network-wide system testing and validation. Throughout this process, rigorous testing of individual components, interfaces, and the overall architecture will be conducted, including functional, modular, inter-modular, scalability, and security tests. The objective is to ensure the accuracy and efficiency of the proposal through comprehensive testing processes inherent to software development projects.

6.2 Advancements

6.2.1 Developments Performed

To be able to achieve the previous objectives various activities have been conducted. A GLACIATION project has been created on GitHub with tasks for each partner to be able to track, show the progress and align our understanding of what need to be done. This tasks have been grouped in milestones. We have planned six milestones and four of them are in progress:

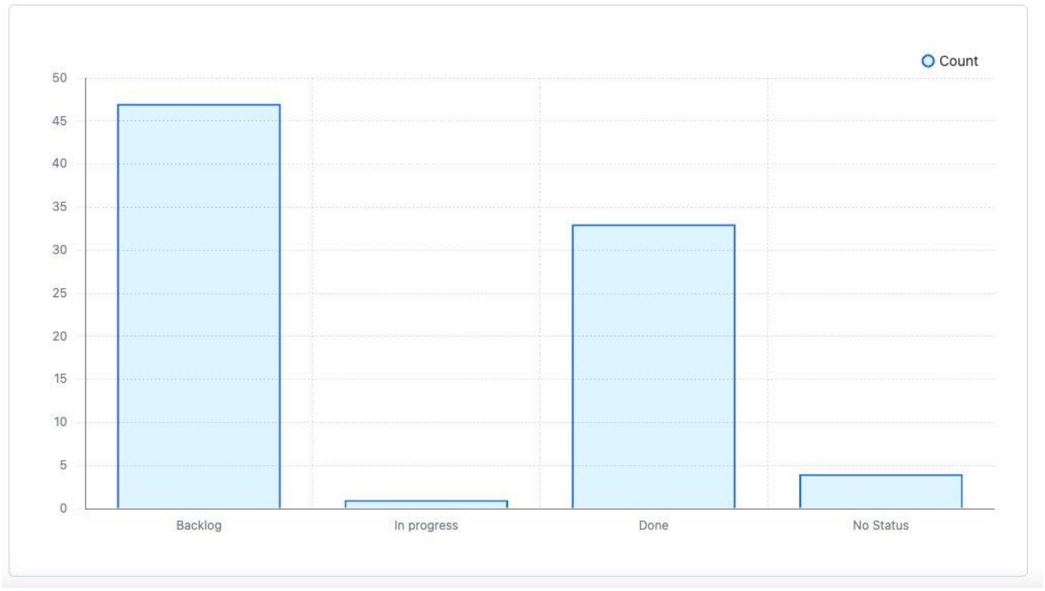
Milestone	Dates	Tasks
Milestone 5.1	Jan 25 - Feb 8	Create comprehensive documentation for the component including: - Component Overview - Service Dependencies - Service Architecture
Milestone 5.2	Feb 8 - Feb 22	Include Domain Model and REST/OpenAPI Specification in the Component Documentation including: - Domain Model/Terminology - REST/OpenAPI Specification
Milestone 5.3	Feb 22 - Mar 7	Make Service stubs available, capable of standalone run in an integration environment with dockerization and helm chart files. - Service Stubs Overview - Standalone Run - Dockerization service/component - Helm Chart Deployment
Milestone 5.4	Mid-March - Cork	Demo individual service/component on personal environment. (can be video recording)
Milestone 5.5	End of March	Gitops CI/CD: - automate helm chart deployment to the K8s cluster - Service implementation
Milestone 5.6	Beginning of April	-Fixing integration issues -Service implementation

We have also conducted more than 12 design sessions to make our architecture concrete, align the interfaces, and remove any ambiguity in the requirements. We also have the integration environment in place and ready.

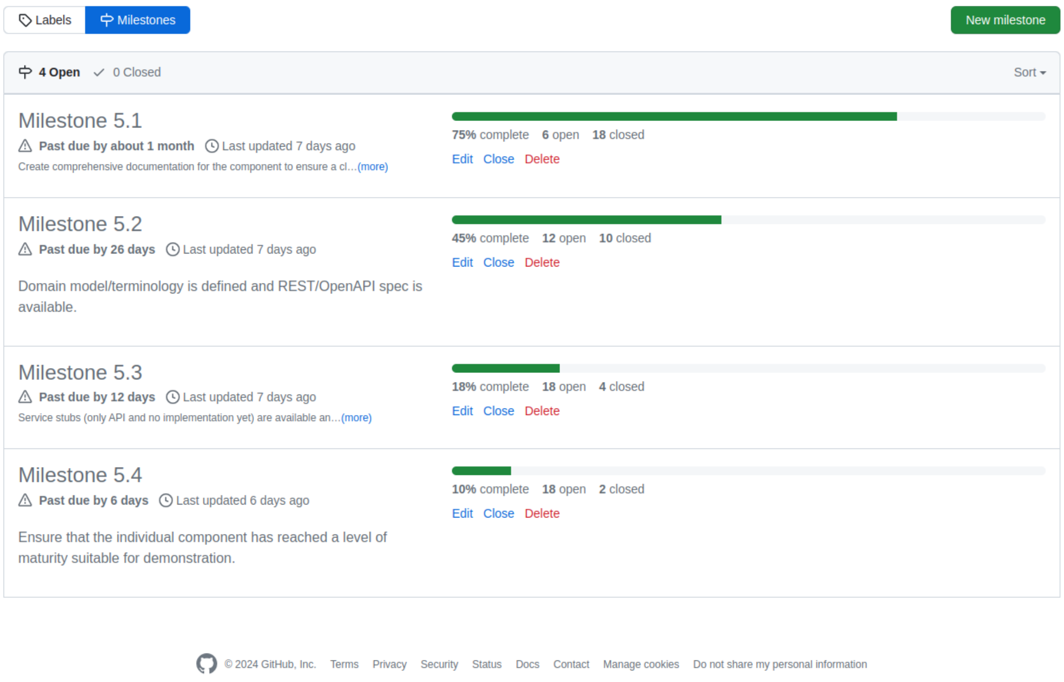
6.2.2 Current Status

Here is the status of the progress

- Prediction Service & Data Storage Service (DELL)
  - Design sessions are completed (Milestone 1)
  - Working on OpenAPI specifications (Milestone 2) and their implementations
- Metadata Service (LAKE)
  - M1 is completed. The service is designed to fulfill the requirements of the other services.
  - Working on OpenAPI specifications
- Trade-off Service (ENG)
  - M1 and M2 are completed. OpenAPI definitions are in the repository.
  - M3 is ongoing
- DKG Replica Service (HIRO)
  - Deprecated/On-Hold due to dropped requirement.
- Replica Service (HIRO)
  - M1 is done. M2, M3 to be done
- Data Processing and Monitoring Service (HIRO)
  - M1, M2 completed. M3



The status of integration tasks



4 Open ✓ 0 Closed

Milestone 5.1
Past due by about 1 month · Last updated 7 days ago
Create comprehensive documentation for the component to ensure a cl…(more)
75% complete 6 open 18 closed
Edit Close Delete

Milestone 5.2
Past due by 26 days · Last updated 7 days ago
Domain model/terminology is defined and REST/OpenAPI spec is available.
45% complete 12 open 10 closed
Edit Close Delete

Milestone 5.3
Past due by 12 days · Last updated 7 days ago
Service stubs (only API and no implementation yet) are available an…(more)
18% complete 18 open 4 closed
Edit Close Delete

Milestone 5.4
Past due by 6 days · Last updated 6 days ago
Ensure that the individual component has reached a level of maturity suitable for demonstration.
10% complete 18 open 2 closed
Edit Close Delete

© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

The Milestone completion progress

6.3 Roadmap

To ensure the system's overall quality and functionality, T3.6 plans to implement a robust CI/CD (Continuous Integration and Continuous Delivery) pipeline. This pipeline will automate lab testing for each software component throughout the development cycle. It aims to streamline development by enabling frequent integration and testing of individual components. Each partner's completed component will be automatically integrated into the central source control system, triggering automated lab tests to detect bugs or compatibility issues early on. The task will also organize meetings and integration sprints to bring together all the components developers to achieve internal integration milestones and enhance the maturity of each component.

Conclusions

T3.1 has provided the result of research and experiments regarding the hardware-optimized processing of distributed knowledge graphs (DKGs), including work on hardware optimization techniques, parallel processing architectures, memory hierarchy optimization, and network communication optimization.

T3.2 provided the latest status of the AI/ML-based solution to workload placement. It detailed the workload-forecasting model, which predicts the future workload demand and introduced the orchestrator, which plans and prepares the resources to meet the predicted workload and ultimately provide for the real incoming workload.

T3.3 explained the distributed data search and data management (movement) in the dynamic environment of the Distributed Knowledge Graph (DKG) using a Swarm Intelligence-based Orchestration. The swarm-based search has been modelled, simulated and evaluated. Algorithms and technologies have been chosen. Developing a prototype and publishing the findings are underway.

The outcomes of T3.4 on Ethical and Trustworthy Autonomy have been compiled into the Deliverable D3.3: Ethical and Privacy Impact Assessment and Recommendations, which are submitted alongside this deliverable at M18.

T3.5 has carried out the work related to development of a secure data management framework (Verdelix) specifically tailored to the GLACIATION's requirements. This management framework provides solutions for data lifecycle management, distributed storage, data provenance, data sovereignty, and data vocabulary.

T3.6 has outlined the integration procedures and protocols needed to establish a lab test to enable all partners to formally declare their dependencies, announce their services, and provide containerized versions of the components. Documentations, Helm Charts, REST API specifications, and Service Stub generations are in place and in use by the partners. Main architectural components and services are defined, the APIs are specified, and implementations are started.