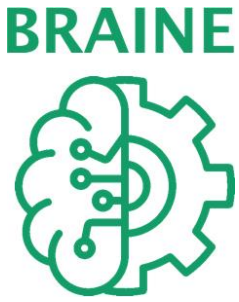


BRAINE - Big data Processing and Artificial Intelligence at the Network Edge

Project Title:	**BRAINE - Big data Processing and Artificial Intelligence at the Network Edge**
Contract No:	876967 – BRAINE
Instrument:	ECSEL Research and Innovation Action
Call:	H2020-ECSEL-2019-2-RIA
Start of project:	1 May 2020
Duration:	36 months

Deliverable No: D4.4

Final project report on the status of WP4 Part 2

Due date of deliverable:	30 N2022
Actual submission date:	17 April 2023
Version:	1.0

ECSEL Joint Undertaking

Project funded by the European Community under the H2020 Programme for Research and Innovation.

Project ref. number	876967
Project title	BRAINE - Big data Processing and Artificial Intelligence at the Network Edge

Deliverable title	Final project report on the status of WP4-Part1
Deliverable number	D4.4
Deliverable version	Version 1.0
Previous version(s)	-
Contractual date of delivery	28 Feb 2023
Actual date of delivery	17 April 2023
Deliverable filename	Final project report on the status of WP4-Part2
Nature of deliverable	Report
Dissemination level	PU
Number of pages	45
Work package	WP4
Task(s)	T4.1, T4.2, T4.3, T4.4
Partner responsible	DELL
Author(s)	Javad Chamanara (LUH), Ahmed Khalid (DELL), Sean Ahearne (DELL), Adam Flizikowski (ISW), Md Munjure Mowla (ISW), Vojtěch Janů (CTU), Hemant Mehta (UCC), Martin Ron (FS), Roberto Bifulco (NEC), Amihay Tabul (MLNX), Ilya Vershkov (MLNX), J.J. Vegas Olmos (MLNX), Philippe Nguyen (SIC)
Editor	Javad Chamanara (LUH)

Abstract	Report on the set of BRAINE platform components developed in WP4 for data management, 5G vRAN, factory motif discovery, and AI profiling.

Keywords	WP4, Data management, 5G, Industry 4.0, AI

Copyright

Deliverable history

Version	Date	Reason	Revised by
00	01.10.2022	Document Outline	Javad Chamanara
01	06.02.2023	List of Components and contributions	Javad Chamanara
1.0	17.04.2023	Final	Javad Chamanara, Filippo Cugini

List of abbreviations and Acronyms

Abbreviation	Meaning
5G	5th Generation
AI	Artificial Intelligence
API	Application Programming Interface
CPU	Central Processing Unit
CU	Centralized Unit
DU	Distributed Unit
EMDC	Edge Mobile Data Center
EPC	Evolved Packet Core
EU	European Union
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
IoT	Internet of Things
IT	Information Technology
KPI	Key Performance Indicator
MES	Manufacturing Execution Systems
MOD	MOtif Discovery
RAN	Radio Access Network
UE	User Equipment
USRP	Universal Software Radio Peripheral
DoA	Description of Action
K8S	Kubernetes
DB	Database
DLCM	Data LifeCycle Manager
JSON	JavaScript Object Notation
GUI	Graphical User Interface
REST	Representational State Transfer
VNF	Virtual Network Function
MQTT	Message Queuing Telemetry Transport

Table of Contents

List of Figures

1. Executive summary

Work Package 4 delivers 15 software components that not only interact with each other to build part of the EMDC platform, but also, they are utilized by the use-cases. This deliverable is the technical report that communicates the final outcome the first part 2 of the WP4 components. In D4.3 seven components were present. In this deliverable the remaining 8 components including their technical details, state-of-the-art as well as advancements are presented. Some of the components may have been reported in D4.3 and appear here too. This is because the partners have either made changes or improvements or provided more technical details in D4.4.

2. Introduction

WP4 has produced 14 components, which their specification, features, and performance is to be reported in two deliverables, D4.3 presented seven of the components (see table below) and this document, D4.4, explains the rest. Based on the DoA's requirements, D4.4 includes the following sections and results, which cover 8 of the components, belonging to the following categories:

- **Workflow definition language and authoring tool:** This deliverable section provides a working ontology, suitable to describe data workflows as well as an authoring tool to visually create these workflows.
- **Flow-based telemetry with ML-based network analyzers:** This deliverable section is a software system providing a framework for telemetry collection of specific network infrastructure metrics, effectively streaming selected measurements to ML-powered network analyzers.

To clarify the components' distribution among the two deliverables, table below indicates partners responsible for each of the components and the deliverable (D4.3 or D4.4) each component will be reported in.

Partner	Components	Deliverable
ISW	vRAN with adjustments (C4.12)	D4.3
FS	Motif Discovery Tool (C4.11)	D4.3
UCC	Data Placement (C4.9)	D4.3
DELL	Data lifecycle manager (C4.1)	D4.3
	Policy manager (C4.2)	D4.3
	Global File System (C4.3)	D4.3
NEC	AI platform profiling engine (C4.13)	D4.3
LUH	Active Data Product (C4.4)	D4.4
	Monitoring Dashboard (C4.8)	D4.4
SIC	Catalyzr tool (C4.5)	D4.4
ECC	Authoring tool (C4.6)	D4.4
	Service Orchestrator (C4.7)	D4.4
IMC	Healthcare Assisted Living (C4.10)	D4.4
MLNX	Network Telemetry Framework (C4.14)	D4.4

Overall System Architecture

Figure 2.1 illustrates the overall BRAINE's architecture and locates the WP4 components. Note that while C4.11 and C4.13 are developed by use case partners, the features provided are capable of acting as a service which could be consumed by any other use cases where desired, hence their inclusion in WP4. C4.12 integrates as part of the overall workload placement framework, influencing workload distribution of vRAN components based on the overall systems current state.

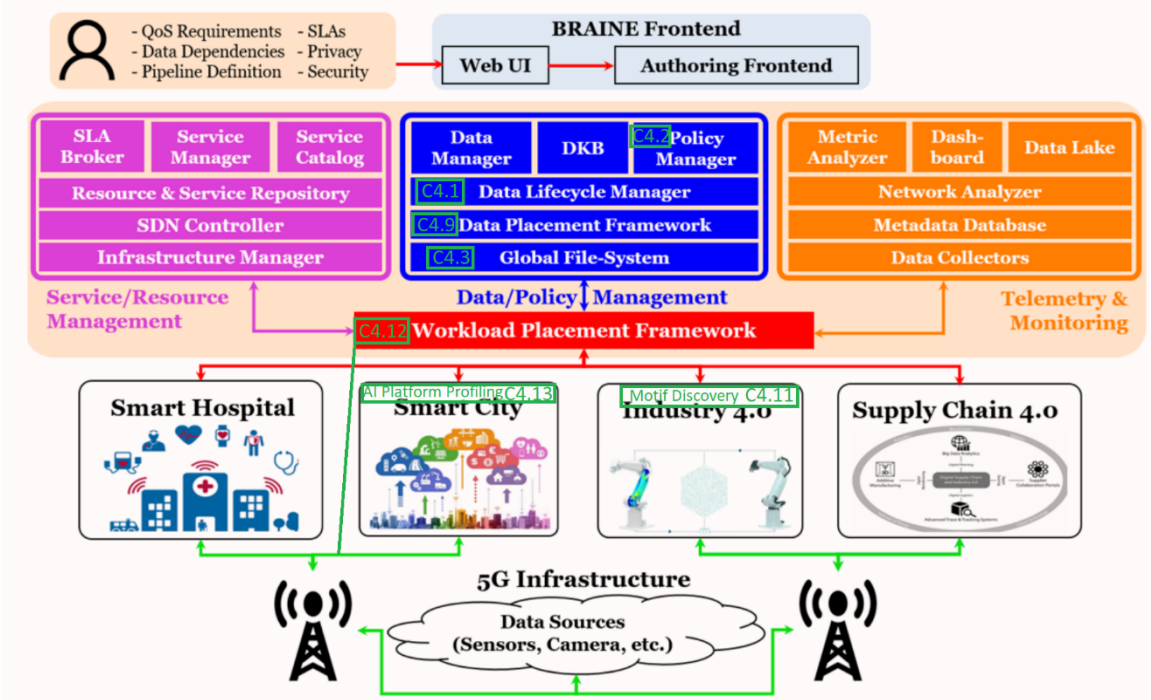


Figure 2.1 BRAINE Architecture Diagram showing components developed in D4.3

3. Status of the components

3.1. Global File System (C4.3)

Component ID		Component Name	Development	Owner
C4.3		Global File System	100%	DELL

GitLab Repository: https://gitlab.com/braine/dmf
Containerized: Yes
Registered on BRAINE platform image registry: Yes
Deployed as a pod and is functional on BRAINE platform: Yes
Integrated with other platform components: Yes
Status: The development of data storage system based on Apache Ozone is complete and the sub-components have been deployed as pods and services in the BRAINE platform. C4.3 is a storage solution that provides a unified filesystem and object store for applications and workloads running on the platform. It has been integrated with other system components such as data placement framework and policy manager, as well as use-case applications that require persistent storage. Further integration activities may be carried out in WP5, based on use-case requirements.

3.1.1. Technical description

In this period, we studied two encryption techniques and here we are discussing them along with adopting suitable encryption techniques with the constraint-based data placement technique (named as CATER) developed in the component C4.3 in a previous report. The discussion focused on Apache Ozone's default encryption technique called Transparent Data Encryption (TDE) and Attribute Based Encryption (ABE).

Transparent Data Encryption (TDE) in Apache Ozone, encrypts the data at rest (i.e., data on the disk). TDE encrypts the data on disk in way that is transparent to the user, meaning the user accesses the Ozone data encrypted on disk identical to accessing non-encrypted data. No knowledge or implementation changes are needed on the client side and the user sees the data in its unencrypted form. In Apache Ozone we can enable TDE at bucket level during bucket creation. All files written to the designated encrypted bucket are encrypted on disk. At present, TDE is the only encryption technique supported by Ozone.

Attribute-based encryption (ABE) is a type of public-key encryption in which the secret key of a user and the ciphertext are dependent upon attributes (e.g., the country in which they live, their group, or the kind of subscription they have) [1]. In such a system, the decryption of a ciphertext is possible only if the set of attributes of the user key matches the attributes of the ciphertext.

Example Scenario: Consider the healthcare use-case involving data of several internal applications like patients' records, administration data, inventory data and insurance data. The data should be encrypted so that only authorized user will be able to access them.

To use TDE in such situation we may like to use one key per application so that the only users from patient management system (doctors, nurses, internal medicine members) can decrypt the data using application's key. Similarly, the administration data will be accessible by admin team, inventory data may be access by the members of inventory data management and the insurance team can access and use the insurance related data.

The ABE encrypts/ decrypts the data using user attributes and it also has a data access policy. For example: if the patient record management application creates the access policy using the attributes [Doctor ∨ Internal Medicine] and then encrypts the data. Only users with attributes Doctor or Internal Medicine can decrypt the data.

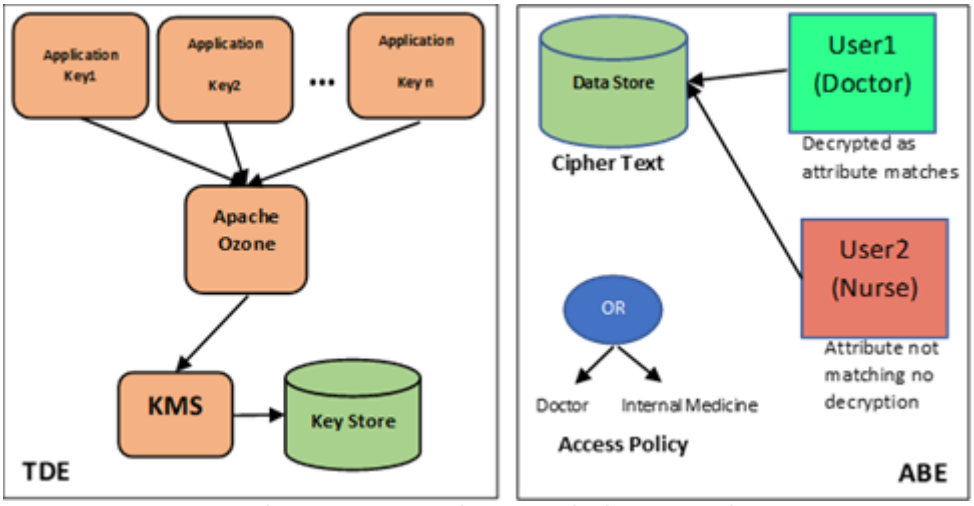


Figure 3.1 Example Scenario (TDE, ABE)

3.1.1.1. TDE in Apache Ozone

Using Ozone's Transparent Data Encryption (TDE) the data on the disk can be encrypted-at-rest and decrypted during access. Ozone supports TDE at Bucket level. We need to setup a Key Management Server (KMS) and provide its URI to Ozone using a configuration in hdfs-sites.xml.

Before encrypting a bucket, the client needs to create a bucket encryption key using the following command

Hadoop key create encKey

After the key creation this key can be used by ozone to create an encrypted bucket by using following command.

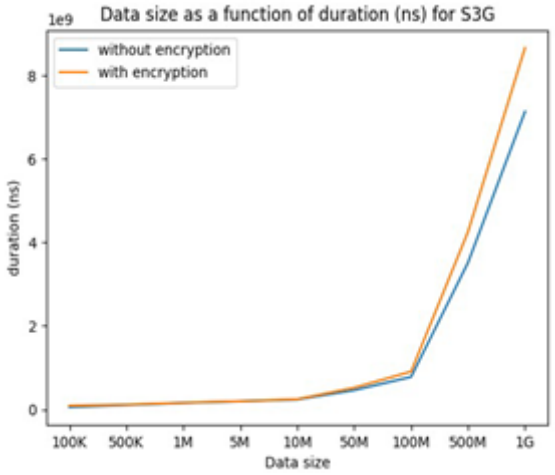
ozone sh bucket create -k encKey /vol/encryptedBucket

Now onwards, all data written to the *encryptedBucket* will be encrypted via the encKey and while reading the clients will talk to Key Management Server and read the key and decrypt it. In other words, the data stored inside Ozone is always encrypted.

While reading the client will talk to key management server to get the key and decrypt the data. If using Ranger (as KMS) then we can enable the ACLs support in Ozone and set the ACL authorizer class to Ranger Authorizer.

We perform the analysis of TDE performance by comparing when data is written with encryption and without encryption. We collected the data while writing files of different size (100KB, 500KB, 1MB, 10MB, 50MB, 100MB and 1GB).

From the results in Figure 3.2, we observe that the while writing with encryption it takes 17% longer time and uses 30% more CPU as depicted in the following two figures.



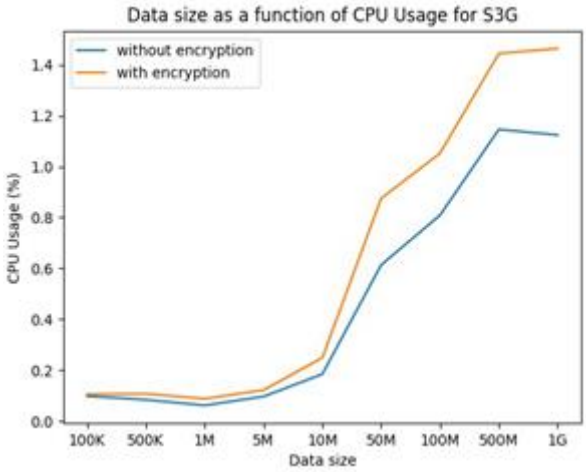


Figure 3.2 Ozone Performance (with encryption and without encryption)

3.1.1.2. Attribute Based Encryption

ABE is a type of public-key encryption in which the secret key of a user and the ciphertext are dependent upon attributes (e.g., the country he lives in, or the kind of subscription he has). In such a system, the decryption of a ciphertext is possible only if the set of attributes of the user key matches the attributes of the ciphertext. A crucial security feature of Attribute-Based Encryption is collusion-resistance: An adversary that holds multiple keys should only be able to access data if at least one individual key grants access.
There are mainly two types of attribute-based encryption schemes: Key-policy attribute-based encryption (KP-ABE) and ciphertext-policy attribute-based encryption (CP-ABE) [2], [3], [4].
In ciphertext-policy attribute-based encryption (CP-ABE) a user's private-key is associated with a set of attributes and a ciphertext specifies an access policy over a defined universe of attributes within the system. A user will be able to decrypt a ciphertext, if and only if his attributes satisfy the policy of the respective ciphertext. Policies may be defined over attributes using conjunctions, disjunctions and (k,n)-threshold gates, i.e., k out of n attributes have to be present.
KP-ABE is the dual to CP-ABE in the sense that an access policy is encoded into the users' secret key, e.g., (A∧C)∨D, and a ciphertext is computed with respect to a set of attributes, e.g., {A,B}. In this example the user would not be able to decrypt the ciphertext but would for instance be able to decrypt a ciphertext with respect to {A,C}.
An important property which has to be achieved by both CP- and KP-ABE is called collusion resistance. This basically means that it should not be possible for distinct users to "pool" their secret keys such that they could together decrypt a ciphertext that neither of them could decrypt on their own (which is achieved by independently randomizing users' secret keys).
Advantages of ABE: ABE offers following advantage:

1. Secure both data in rest and data in transit.
2. Fine grained (Policy-based) access control and flexible access policies.
3. Easier Key Management
4. Efficient on Small Form Factor computers (like Raspberry Pi) [5].

Implementation: We have compiled and tested following open-source ABE implementation, we may integrate them with Ozone.

- **https://github.com/junwei-wang/cpabe**

3.1.1.3. Encryption with the proposed data placement policy

Distributed data and object stores manage multiple storage nodes and can ensure availability and fault-tolerance through replication. However, current solutions consider

limited constraints and policies when making data placement decisions. As multiple stakeholders provide storage nodes to multiple consumers, it is essential to manage placement according to security and privacy constraints. Furthermore, increasing heterogeneity in the edge-cloud continuum gives rise to additional hardware requirements, e.g., processor or storage type. Moreover, operational policies from regulatory bodies must also be considered, e.g., GDPR which may restrict where certain data can be located.

Brief Details of CATER

To address the challenges discussed in previous paragraphs, we developed CATER, a modular poliCy-bAsed daTa placEment fRamework that can integrate with existing storage systems. We formulate the data placement problem as an optimization model, constrained by sharing and hardware constraints. We also develop heuristics for real-time computation. We implement a prototype of CATER and integrate it with Apache Ozone. The disk sharing and hardware constraints are defined by the applications and given as the input to the optimization model.

CATER an external data placement framework to optimize the number of nodes used while respecting all the constraints of various applications deployed on the cluster. Upon receiving a request to store the data, modified Apache Ozone invokes the placement API to suggest the list of suitable nodes to be used to store the data.

The placement algorithm has two underlying components as:

- An optimization model that minimizes the number of datanodes while respecting all application constraints. The problem is formulated as an Integer Linear Program and solved using a Constrained Programming with Satisfiability methods (CP-SAT) solver. This solution enables the system to enforce the desired constraints using the minimum number of nodes. Lesser active nodes lead to lower energy consumption.
- A heuristic algorithm to cope with the real-time evolution of the data store. This works as an add-on to the optimization model, after the initial placement, to efficiently handle the modifications in the applications and constraints, and to minimize data movement operations.

This framework is implemented using an external REST API, independent of a specific data store. This independence allows the user to integrate the API with any data store.

We changed the part of the data store (Apache Ozone) that selects the list of datanodes. The modified Ozone uses the API to obtain the datanode list to fulfil the demands of the application. In the stateful mode, after creating the list, API updates its records for current allocation to have the details of this new request.

We may have an input parameter for using encryption just like application's sharing constraints, hardware, location etc. When an application requires encryption then all the buckets for the application will be encrypted. For TDE we recommend to have one key per application. Access policy within the ABE enforces access control using the user's attributes.

If all the underlying file systems are the same of all the nodes, then choosing TDE or ABE is not a placement problem and CATER behaves normally. However, if the file systems on the nodes support different encryption techniques (TDE/ ABE) then CATER takes should take care of such situation accordingly.

CATER and TDE

The following figure depicts client and Ozone interaction when TDE is in effect. Client Interacts with Ozone to request a key. In response Ozone requests KMS to create an encKey for the client. This encKey is used by the client to create an encrypted bucket. Once an encrypted bucket is created the files added to this bucket will be encrypted. If the client tries to read the file, it will be decrypted using the encKey.

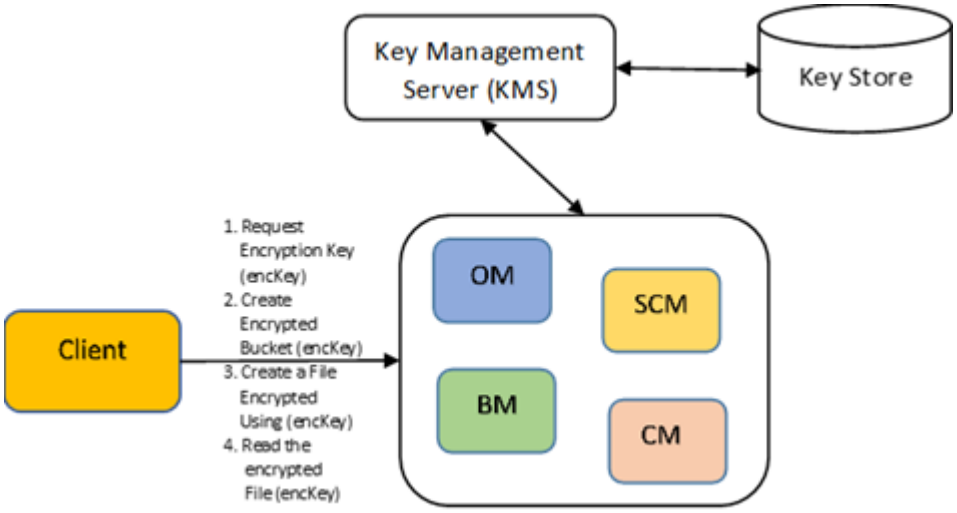


Figure 3.3 TDE in Apache Ozone

CATER and ABE

Following figure depicts the proposed system flow to integrate ABE with Ozone. The applications define corresponding attributes and associated access policy. The Key Management Server keeps the application wise details of the attributes and access policies. The clients will be provided the key for decryption if their attributed matches with the access policy defined by the data owner (Applications).

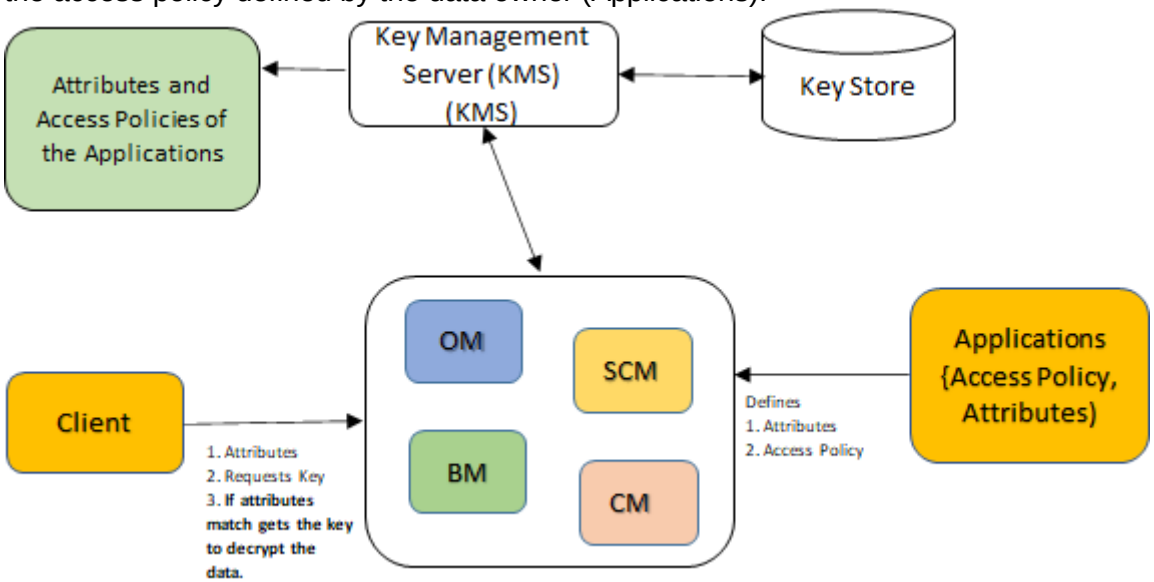


Figure 3.4 ABE in Apache Ozone

Encryption methods are quite useful in securing the data and it adds a further layer of security over the secure placement policy (CATER). In this period, we studied and discussing both TDE and ABE in details. TDE secure the data at rest and ABE is capable of securing the data at rest and during the data in transit. We also suggest guidelines on how to adopt TDE/ ABE along with the placement policy (CATER) in the Apache Ozone in context of the BRAINE project.

References:

[1] Goyal, V., Pandey, O., Sahai, A., & Waters, B. (2006, October). Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security* (pp. 89-98).

[2] Bethencourt, J., Sahai, A., & Waters, B. (2007, May). Ciphertext-policy attribute-based encryption. In *2007 IEEE symposium on security and privacy (SP'07)* (pp. 321-334).
[3] Ostrovsky, R., Sahai, A., & Waters, B. (2007, October). Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM conference on Computer and communications security,* (pp. 195-203).
[4] Goyal, V., Jain, A., Pandey, O., & Sahai, A. (2008, July). Bounded ciphertext policy attribute based encryption. In *International Colloquium on Automata, Languages, and Programming* (pp. 579-591). Springer, Berlin, Heidelberg.
[5] Ambrosin, M., Anzanpour, A., Conti, M., Dargahi, T., Moosavi, S. R., Rahmani, A. M., & Liljeberg, P. (2016). On the feasibility of attribute-based encryption on internet of things devices. *IEEE Micro*, *36*(6), 25-35.
[6] Hwang, Y. W., & Lee, I. Y. (2020). A Study on CP-ABE-Based Medical Data Sharing System with Key Abuse Prevention and Verifiable Outsourcing in the IoMT Environment. *Sensors*, *20*(17), 4934.

3.2. Active Data Product (C4.4)

Component ID	Component Name	Development	Owner
C4.4	Active Data Product	65%	LUH

GitLab Repository: https://github.com/braine-project/WP4R/tree/main/T41
Containerized: Yes
Registered on BRAINE platform image registry: Yes
Deployed as a pod and is functional on BRAINE platform: No
Integrated with other platform components: No, waiting for UC integration
Status: ...

3.2.1. Technical description

An active data product is indeed a dataset encapsulated in a secure container that allows access via a well-defined access point that conforms to the terms of an agreed-upon contract. A data product is said to be active as it can operate in an external environment. It is indeed a self-contained, secure executable package that must be run in order to allow for the utilization of the data it contains. When requested by external agents to access the data, it will ensure that the request and the response comply with contract terms, usage, sovereignty regulations, and boundaries defined.
A contract definition language using YAML is under development, by which the data owner can define the terms and conditions for accessing the data. The contract is then enforced by the contract controller. Any legitimate access to the data will be recorded in a blockchain for contract term enforcement as well as for auditing and accounting. The ADP component is a prototype and may not be secure enough for production environments with high-sensitive data.
Each UC that aims at sharing data with other UCs or external systems/parties may benefit from this component.
The general user interaction with the ADP is depicted in Figure 3.5 below.

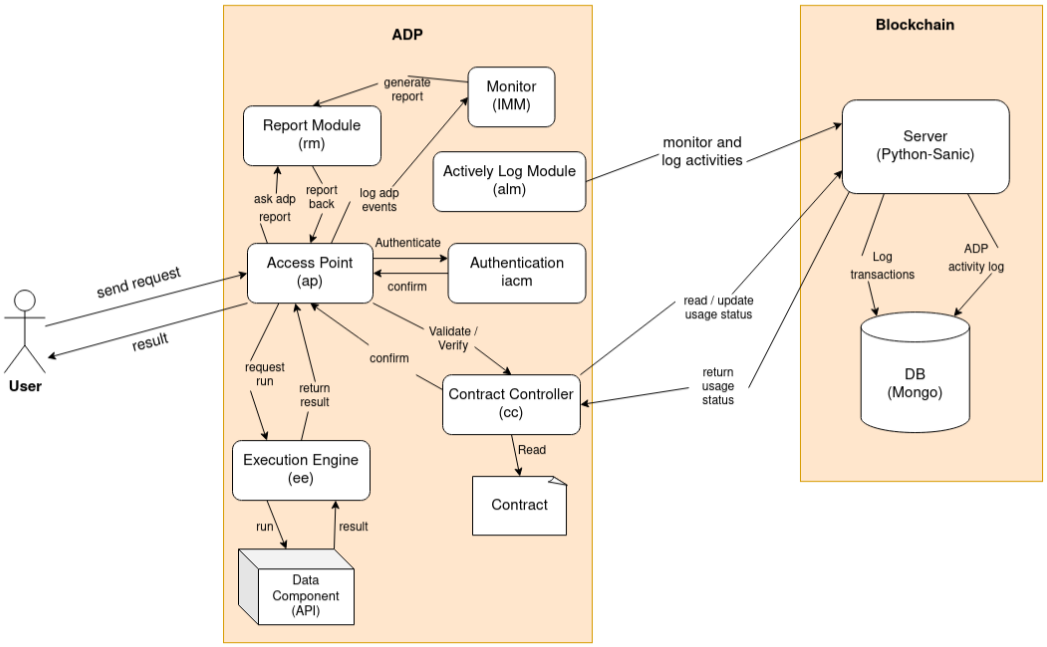


Figure 3.5 Interaction model of the ADP and a user (agent)

The ADP encapsulates a dataset (or an AI/ML model) and a contract that governs the access and usage terms of the dataset. When an agent requests to access data, it needs to provide the querying parameters. The access point first validates the request by checking authentication and authorization policies as well as usage policies and terms as in the contract. Therefore, if the request violates any usage of the terms, the request will be discarded and a cost associated with it will be submitted to the blockchain. This is to mitigate denial of service and data exploitation attacks. Any successful request will also be written to the blockchain for usage tracking. The access point validates the parameters against the contract terms. For example, an agent may be prevented from filtering data based on the gender of people in the dataset, or it can be prevented from requesting data of people under 18 years old. After parameter validation, the request will be passed to the execution engine to construct the result set from the dataset/model. The result set will also be validated against contract terms. For example, the input parameters may not explicitly ask for data about females, but the result may contain such data. If there is a restricting term in the contract, those records will be dropped from the final response. As both served and discarded resources are logged in a blockchain via the activity Logging Module, usage can be tracked not only by the ADP instance but also by the parties of the contract as well as third parties. At the moment a prototype of the ADP has been implemented and is under test. The blockchain is simulated by MongoDB, which will be replaced after the contract controller reaches stable status. Research and development on securing the whole ADP against malicious agents and contract parties is ongoing.

3.2.2. Advancements

The ADP was utilizing a MongoDB database for storing the transactions produced by the activities. In this period, LUH has enhanced the component to utilize a blockchain. Now, ADP can be configured to either use MongoDB (mainly for testing, debugging, development) or IOTA blockchain for operational purposes.

In brief, IOTA is a blockchain network comprised of a set of nodes that can communicate to each other via a messaging protocol. The transaction are written to the nodes, so that each node has the record of the entire network. Usually blockchains have a main network

and one or more test/development networks[1]. In this project LUH has established a connection to an IOTA test network as shown in Figure 3.6. The ADP instance that runs on the data consumer facility, will use this connection to submit transactions (contract data, usage record, and activity logs) to the blockchain, that is outside of the data consumer realm.

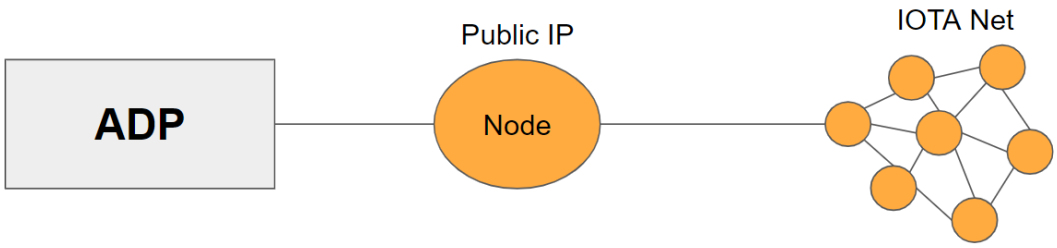


Figure 3.6 ADP connection to the IOTA network

To work with IOTA, ADP utilizes the iota_client[2] python library. It relies on the IOTA's messaging system as explained in the documentation[3], which is brief includes a string **index** to identify the message and a **data** element that contains the actual message content encoded in UTF-8 format. ADP loads three types of transactions into the message payload, namely: contract, usage, activity.

Contract transaction type:

On the ADP first run, the contract's manifest is submitted to the blockchain with a unique contractID so that the contract itself in maintained in the distributed ledger.

Usage Transaction type

This is the transaction type ADP uses to keep track of all the contractual usages of the data encapsulated into the ADP container. The transaction contains the contract term/condition that has been invoked, the counters, limitations, and remaining amount of the usage topic and any other quotas, and the result of the invocation of the terms. These transactions are used by the ADP's contract controller for follow up usage requests, as well as visible to all parties having access to the blockchain.

Activity Transaction type

All activity and events that have been configured in the ADP's Activity Log Module will use this transaction type to submit messages to the blockchain. These transactions are usually used for administrative, monitoring, or error handling.

3.2.3. Performance Evaluations and comparisons

As mentioned before, ADP now is capable of storing transactions either in MongoDB or IOTA blockchain. The choice can be made proving the storage endpoint URL to the Docker container of the ADP.

[1] https://v2.iota.org/how-it-works/introduction
[2] https://wiki.iota.org/iota.rs/examples/running_examples/
[3] https://wiki.iota.org/iota.rs/examples/data_message/

LUH conducted a set of initial evaluations assessing the functionality of ADP as well as comparing performance of the system when running on MongoDB versus IOTA blockchain.

Both variants were tested with identical contracts and the target was to compare the runtime. It was clear that MongoDB version was faster than the blockchain, first because the blockchain was hosted remotely and second that a single transaction on a blockchain takes longer as it requires multiple validations by different nodes and includes a census too. However, the blockchain performed better than expectation, as summarized in Table 3-1.

Table 3-1 ADP initial performance evaluation on document store and blockchain (milliseconds)

Indicator	MogoDB	IOTA blockchain
Login	020	916
Handshake	031	634
Contract term 1	0106	3377
Contract term 2	0130	5583
Contract term 3	0118	4551
Report Module	0072	1328
Get history	0027	1649

In addition to the performance evaluation, LUH tried several times to install and run the both of the variants. The results suggest that implementation, running, and maintaining the ADP over IOTA is easier and more stable, bearing lower maintenance cost. The higher transaction time can be justified by the non-replaceable role of the distributed leger in establishing trust between the parties allowing them to use the usage transactions as a basis for licensing and costing.

3.3. Catalyzr Tool (C4.5)

Component ID	Component Name	Development	Owner
C4.5	Catalyzr Tool	50%	SIC
GitLab Repository: No Containerized: No Registered on BRAINE platform image registry: No Deployed as a pod and functional on BRAINE platform: No Integrated with other platform components: No Status Report: Catalyzr is a tool for joining work between cryptographer and software developer for hunting security vulnerability. The capability to track microarchitecture-induced vulnerabilities (i.e., cache timing attacks) is operational. The Catalyzr has been deployed for joint work between SIC and ISW to secure ISW software cryptographic libraries.			

3.3.1. Technical description

Introduction

The component is a detection tool designed to analyse C source code and to detect potential vulnerabilities related to micro-architectural side channel attacks.
Those vulnerabilities are basically code statements that cause private data leakages on certain computer architectures. In the presence of such vulnerabilities, an attacker can use cache timing attacks to spy on private data. Such attacks rely on the differences of access time between cache memories and main memories and allow attackers to “guess” if some data is loaded in the cache memory and being used by the processor.
Such attacks usually target crypto libraries but are applicable to any type of software program manipulating sensitive data (private keys, passwords, pin codes, etc.). Infamous exploits like Spectre and Meltdown rely partly on cache timing attacks.

Input and output

The vulnerabilities related to cache timing attacks can be detected at source code level. Indeed, two types of code lines may lead to such attacks:

- conditional statements testing sensitive data (such as “if” or “switch” in C)
- table accesses using sensitive data as index.

Therefore, the Catalyzr tool is designed as a static code analyser: it takes as input source code, analyses it, detects vulnerable statements with regards to a sensitive variable tagged in the code, and finally lists the found lines in a report.
The C language being the most used for crypto libraries development and the most common target of cache timing attacks (in terms of software executables), it is therefore chosen as main target language. C++ language can also be supported with minor modifications of the tool.
The motivation for the tool design is to work hand in hand with software developers. The user can perform detection checkpoints and iterate all along the development of a software to ensure that no vulnerability is introduced in the development.

Internal workflow

The internal workflow of the tool in described in Figure 3.7. First, the user must tag sensitive variables in the source code using a pragma. Sensitive variables include any data that could compromise the security of a system if known by external attackers (keys, passwords). Second, the tools build the Abstract Syntax Tree (AST) of the program using a compiler, and extract paths related to the tagged sensitive variables. Third, it detects all the statements using direct sensitive variables, or indirect sensitive variables (related to the tagged one) that could lead to cache timing attack. Finally, the tools gather its findings in a report, which can be exported in PDF or HTML.

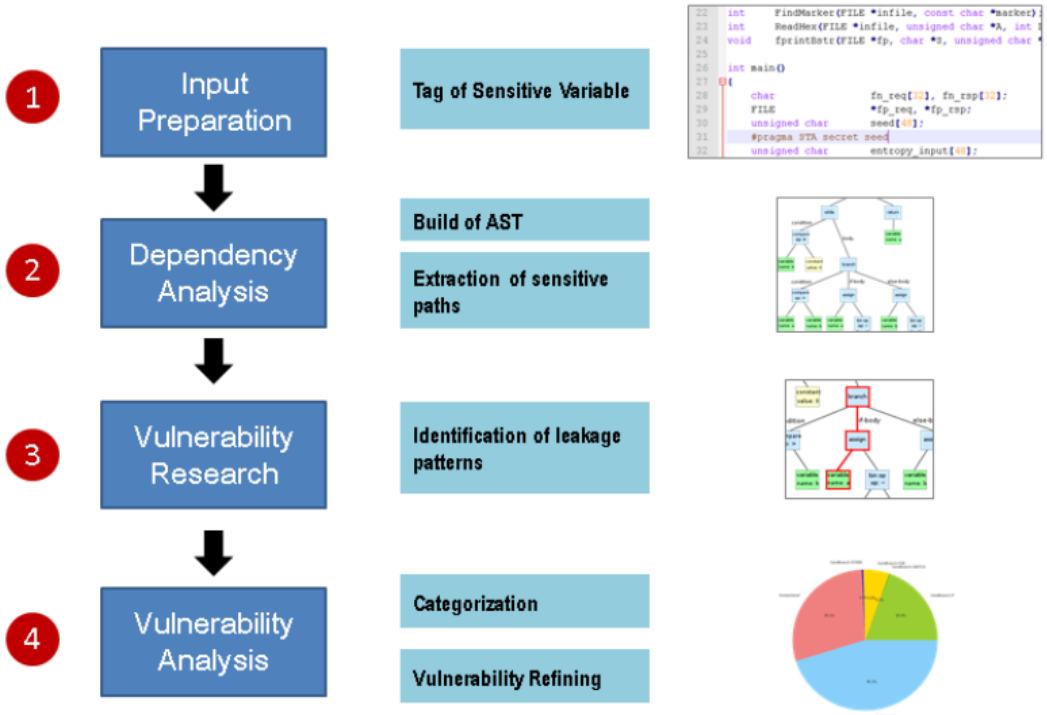


Figure 3.7 Internal workflow of the Catalyzr tool

Architecture and dependencies

The architecture of the tool is as follows:

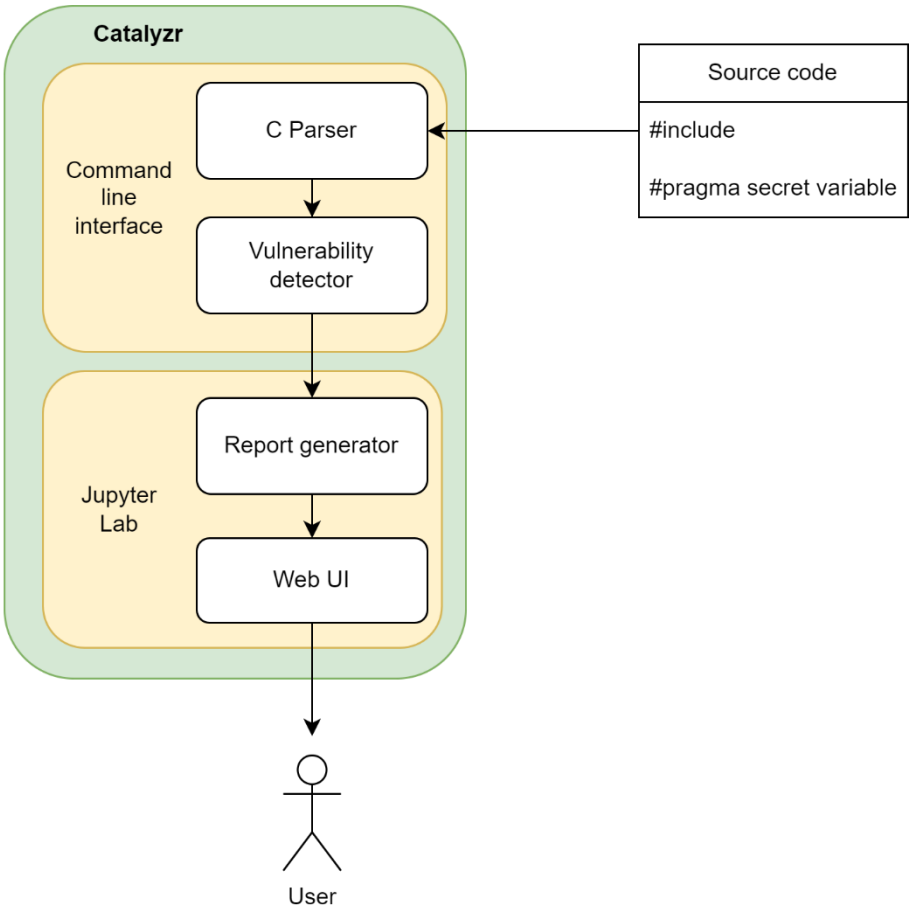


Figure 3.8 Catalyzr architecture

The tool is programmed in Python for the backend which includes the code parser and the vulnerability detection system. It requires the pycparser python package. Regarding the frontend, the command line interface is written in Java and therefore requires the Java Runtime Environment. The tool also proposes a web based graphical user interface based on the Jupyter Lab framework.

3.3.2. State of the Art (SOTA)

Static analysis tools are commonly used in software development. We can mention tools like Parasoft [1] or Coverity [2] which support different languages and allow to choose between various rulesets focusing on different aspects of the source code. Usually, good coding practices are verified with regards to general security, readability, and maintainability, but side channel security is not included in the scope of the usual tools. Therefore, this type of tools and rulesets can be seen as complementary to the verification done by the Catalyzr.

On the other hand, dynamic analysis tools and methods for side channel security exist, mostly in the academia. For example, a recent paper proposed the DATA (Differential Address Trace Analysis) methodology [3], which consists in executing the target software binary to generate memory access traces used to detect data leakages. This approach shows impressive results but is only working on x86 architecture. Indeed, dynamic analysis require the ability to execute the program under test and are therefore architecture dependant. Dynamic methods also rely on statistic methods and require multiple executions of the target software. The number of found vulnerabilities is dependent on this number of executions, while static methods are exhaustive.

References:

[1] Parasoft, S., & SAFE, D. (2016). Parasoft C/C++ test.
[2] https://www.synopsys.com/software-integrity/security-testing/static-analysis-sast.html
[3] Weiser, S., Zankl, A., Spreitzer, R., Miller, K., Mangard, S., & Sigl, G. (2018, August). DATA-Differential Address Trace Analysis: Finding Address-based Side-Channels in Binaries. In USENIX Security Symposium (pp. 603-620).

3.3.3. Advancements

In the recent months, a joint work was done with ISW. ISW is developing an application for their component and the application was using a custom crypto engine (at the time of the collaboration).
The Catalyzr tool allowed to assess the robustness of the cryptoengine provided by ISW. The tool was installed on ISW servers, and a remote access was given to SIC engineers to perform the source code analysis. The crypto engine was analysed in depth, and a report with the tools findings was delivered to ISW.
The findings of the reports are not detailed here for confidentiality reasons.

3.3.4. Performance Evaluations and comparisons

As explained above, the Catalyzr tool has a better base coverage than the SOTA dynamic SCA tools because the static approach is exhaustive: it will detect all potential leakages with a single analysis while dynamic methods require multiple executions of the target binary with no warranty of finding all vulnerabilities. Conversely, the Catalyzr tool generates more false positives which can be identified and discarded semi automatically.
The tool also has better portability across platforms as it requires only source code for analysis

3.4. Authoring tool (C4.6)

Component ID	Component Name	Development	Owner
C4.6	Authoring Tool	90%	ECC
GitLab Repository: https://github.com/eccenca/braine/tree/main/webclient Containerized: Y Registered on BRAINE platform image registry: Y Deployed as a pod and functional on BRAINE platform: Y Integrated with other platform components: Y – The Authoring Tool haven't been integrated with the Global Service Registry. Status Report: The Authoring Tool for service composition is under development. The data model to support persistence through the Resource & Service Catalog is already implemented while the development of the user interface has been already initiated. In the next iterations we expect to have a functional and integrated version working.			

3.4.1. Technical description

Workflow definition language in the last iteration, the Workflow definition was replaced by Argos. Argos language is based on the YAML file and is an extension of the Kubernetes concepts. Providing required functionalities in the project scope and users can describe workflows in a declarative way using manifests (**Listing 3.2.1.**) in a similar fashion to those of Kubernetes and Docker. The use of Argos language to model and store workflows comes naturally, because Argos also provides the workflow engine which is compatible with the BRAINE architecture choices.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: hello-world
  labels:
workflows.argoproj.io/archive-strategy: "false"
  annotations:
workflows.argoproj.io/description: |
  This is a simple hello world example.
spec:
  entrypoint: hello-world
  templates:
  - name: hello-world
container:
  image: hello-world
```

Listing 3.2.1. Argo hello-world workflow example.

Argo Workflows Core Concepts The diagram below illustrates the core concepts in Argo Workflows (See Figure **3.9**). Argo language can be used for workflow definition while the Argo engine executes and manages its states. The workflow is defined through the workflow spec template which contains a list specifying the entry point and type of the workflow. The template defines the instructions to be executed while the entry-point specifies the primary instruction or template to execute before starting the workflow execution.

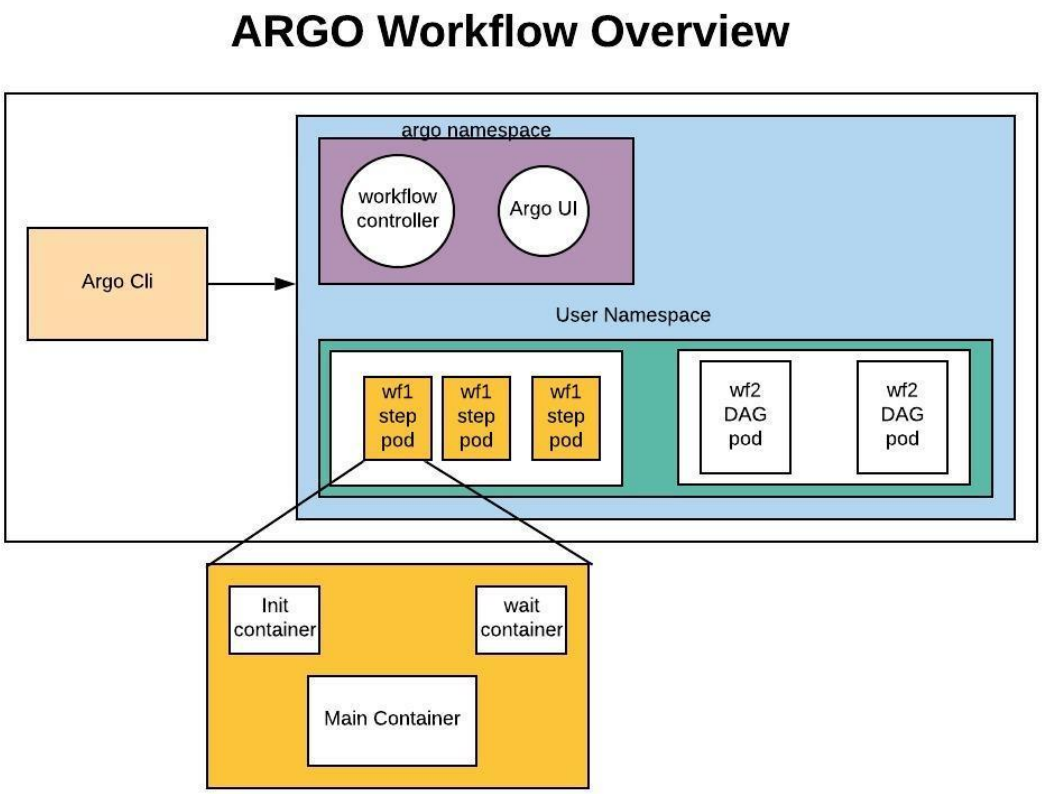


Figure 3.9 Argo Overview

Templates There are several types of templates in Argo workflow, they define the required functions of a workflow, typically in a container. Some of them are, but not limited to:

- Container — schedules a container.
- Resource — directly performs operations on a cluster resource such as GET, CREATE, APPLY, PATCH, REPLACE, or DELETE.
- Script — a convenience wrapper for a container. The script produces a result that automatically exports to an Argo variable, for instance:
    - {{tasks.<NAME>.outputs.result}}
    - {{steps.<NAME>.outputs.result}}
- Suspend — suspends the execution of a workflow for a specified duration until it is manually resumed.
- Invocaters — invokes or calls other templates and control their execution:
- Steps — allows to define workflow tasks as a sequence of steps.
- DAG — allows to define workflow tasks as a graph of dependencies.

Data model the workflow is handled through two main entities in the BRAINE knowledge graph:

Workflow Register: the workflow register stores the address of the Argo workflow endpoint to be used to deploy workflows (see Figure 3.10).

Workflow: the workflow contains the attributes manifest and variables inherited from the superclass deployable (see Figure 3.11). It is used to store the manifest from Argo workflow as well as default variables that may be used on its execution.

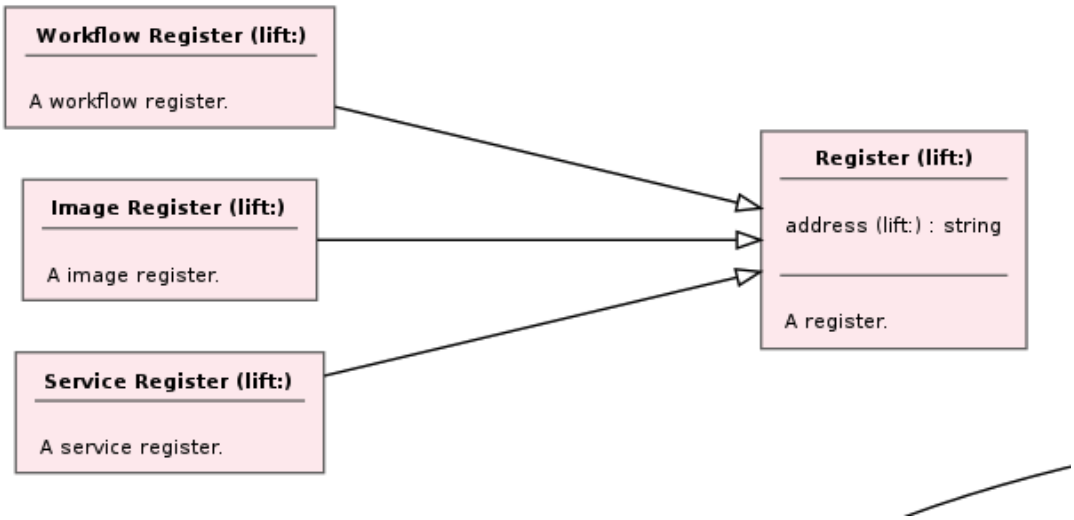


Figure 3.10 Excerpt of BRAINE vocabulary lift highlighting Workflow

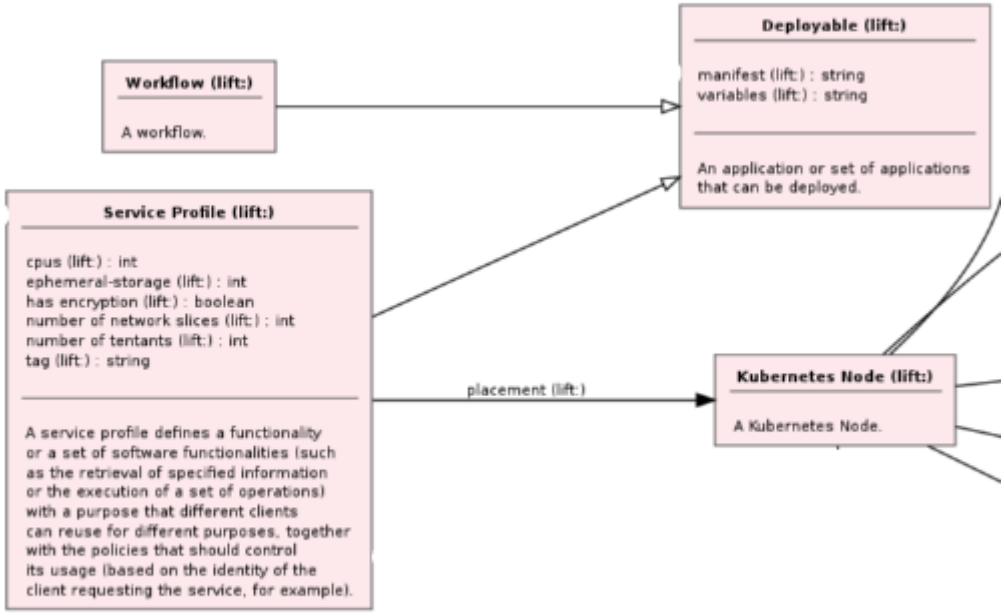


Figure 3.11 Excerpt of BRAINE vocabulary highlighting Workflow and Service Profile

Authoring tool

The authoring tool architecture was updated. Now a web-client communicates directly between the Corporate Memory and Argos, the workflow execution framework deployed in the BRAINE platform. Figure 3.12 and Figure 3.13 show respectively the BRAINE web-client Workflow and Workflow Registry Interfaces.

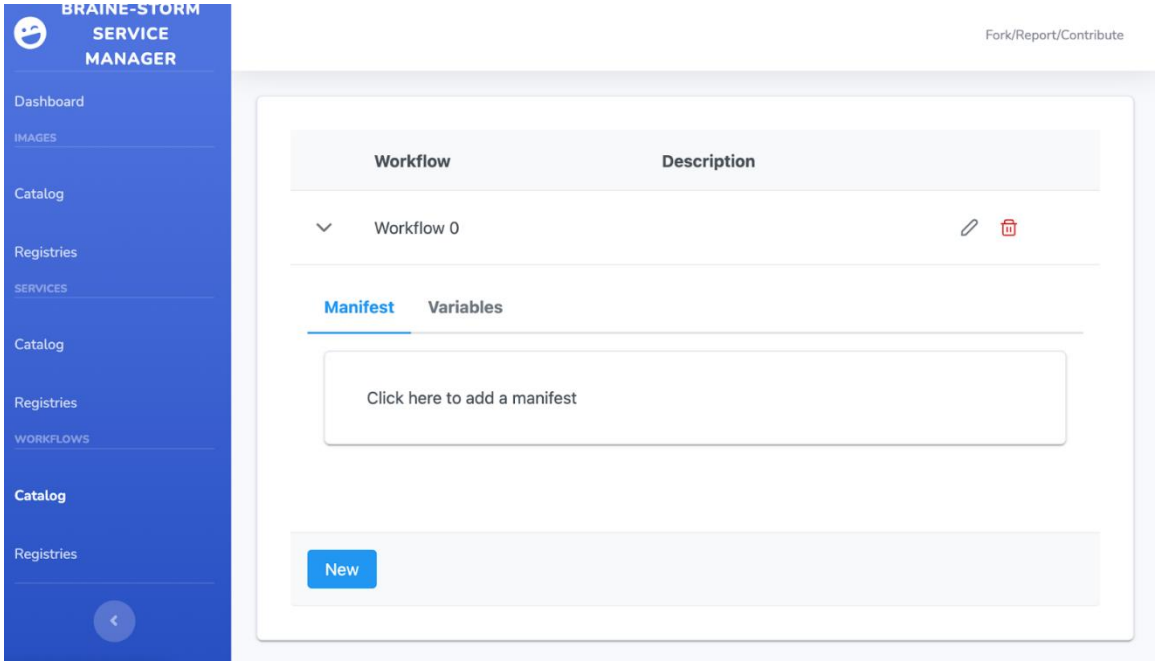


Figure 3.12 Workflow Specification Interface.

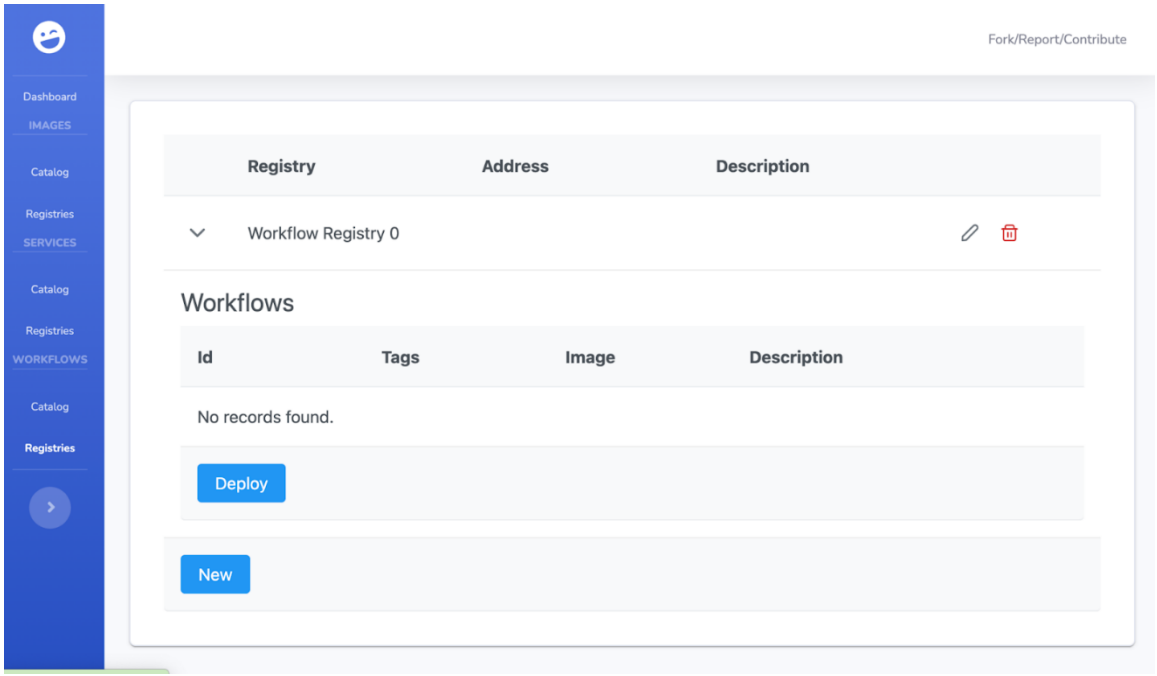


Figure 3.13 Workflow Registry Specification interface.

Deployment States

The new architecture makes the use of the formal deployment states deprecated. Now the web-client can check on the fly information of the running workflow, service or image on the fly directly on the respective registry. This also simplifies the management and removes the necessity of data duplication.

3.4.2. State of the Art (SOTA)

In this section, we discuss some SOTA frameworks and languages considered in BRAINE architecture. Section 4, presents a comparison among them:

Airflow is particularly suited to large scale batch processing of corporate data. However, Airflow has no event processing capabilities and is not well integrated into Docker containers or Kubernetes.
Dagster similar to Airflow, it is better suited to large scale batch processing.
Spark and Flink are stream and batch processing frameworks that can handle very large datasets. However, they are not best suited to providing end-to-end real-time dataflows.
Node-RED is based on Nodejs. It has the disadvantage of providing only limited support for remote distribution of workflows.
NiFi is an Apache open-source project. It is well suited for Workflows, but it is a standalone solution and does not integrate well with Kubernetes.
Kubeflow, is tightly integrated into Kubernetes. However, its primary purpose is to automate the lifecycle of ML models.
Spring Cloud Dataflow can be integrated with Kubernetes and execute workflows. However, because BRAINE implements services written in a variety of languages such as Python and prebuilt data processing engines like Spark, it is not well suited for the project.
Argo (argoproj.github.io) is a new open-source workflow engine tightly integrated into Kubernetes. Its template language and framework are built on top of Kubernetes.

3.4.3. Advancements

Over the last months we re-implemented the web-client adding support for Workflow authoring and deployment. The BRAINE web-client now has a completely new architecture. In addition, we enriched our data model with support for workflow storage management. The authoring tool communicates directly with the different components in the BRAINE architecture, simplifying the architecture by hiding complex communication APIs and protocols through friendly user interfaces. Figure 3.14 shows the old architecture while Figure 3.15 illustrates the new one. It is possible to see the replacement of the Service and Image Orchestrator by the BRAINE Web Client which communicates directly with the Workflow Engine as well as Global Service and Image registries. Further the Authoring Tool also facilitates BRAINE services usage as it allows users to access all BRAINE functions in a single interface.

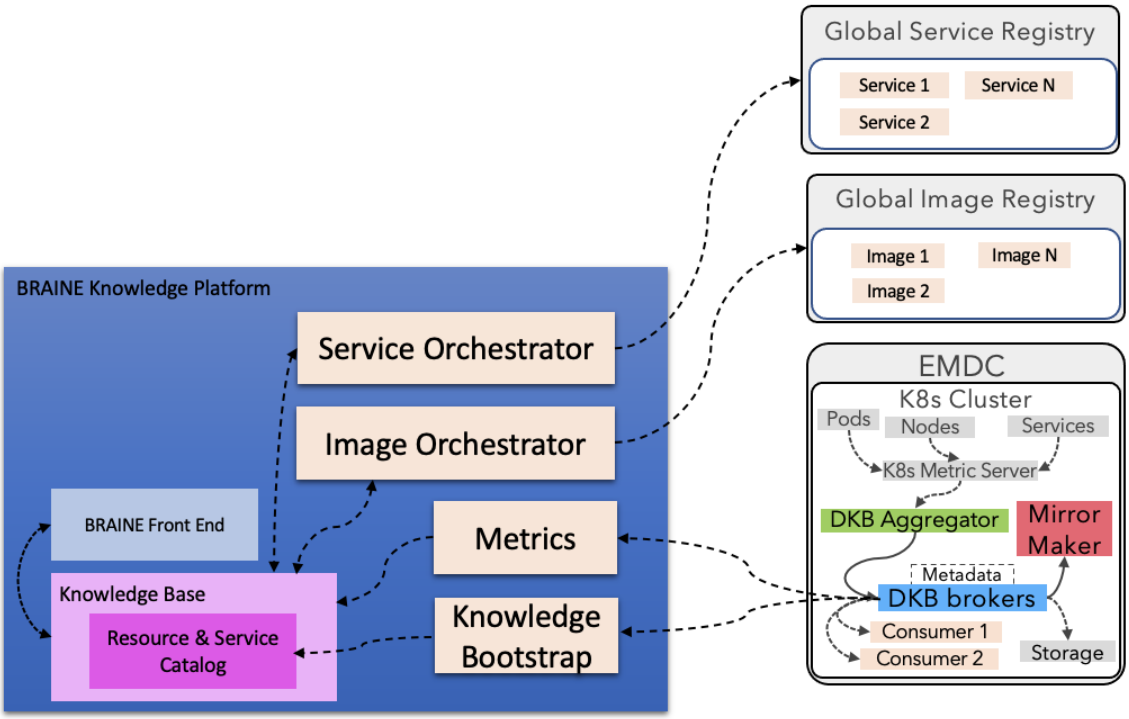


Figure 3.14 Old Architecture.

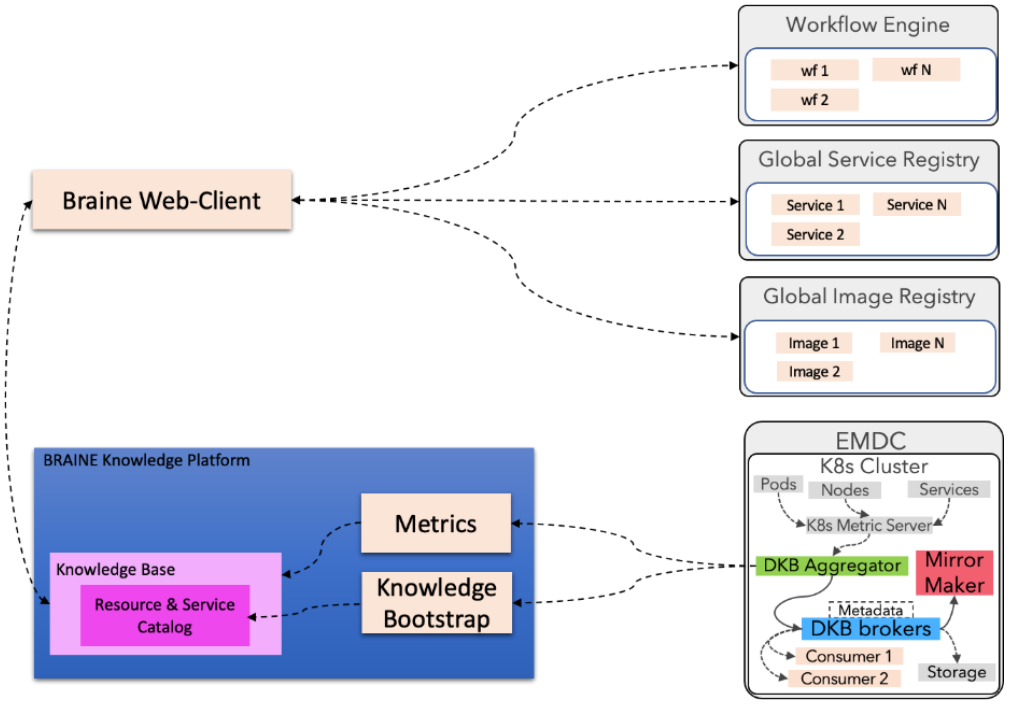


Figure 3.15 New Architecture.

3.4.4. Performance Evaluations and comparisons

The Table 3-2 lists some of the most applicable workflow languages and features, which were considered for the BRAINE project. In the table below, Language indicates the language used. Many workflow names are used to indicate the workflow language in the

absence of a name. DAG stands for Directed Acyclic Graph while K8 Int. Indicates whether the framework integrates with Kubernetes. Maturity indicates how old the language is. Framework indicates whether the language has an engine that can interpret it. It is possible to see that Argo, the framework chosen to manage the workflow execution, is the best option among them although it is relatively new. The main problem of OWL-S is that it has no industry-based workflow engine supporting it.

Table 3-2 Workflow Language Comparison

Language	DAGs	Maturity	Declarative/ Imperative/ UI	Framework	Handle Events	Data Size	K8 Int.
Airflow	**Y**	**Est.**	**Imper.**	**Y**	**No**	**Medium**	**N**
Dagster	**Y**	**New**	**Imper.**	**Y**	**Limited**	**Medium**	**N**
Spark	**N**	**Est.**	**Imper.**	**Y**	**Limited**	**Large**	**N**
Flink	**N**	**Est.**	**Imper.**	**Y**	**Limited**	**Large**	**N**
Node-RED	**Y**	**Est.**	**UI**	**Y**	**Yes**	**Small**	**N**
NiFi	**Y**	**Est.**	**UI**	**Y**	**Yes**	**Small**	**N**
Kubeflow	**Y**	**Est.**	**Imper.**	**Y**	**No**	**Large**	**Y**
Spring Cloud Dataflow	**Y**	**Est.**	**Imper.**	**Y**	**Yes**	**Small**	**Y**
Argo	**Y**	**New**	**Declar.**	**Y**	**Yes**	**Medium**	**Y**
OWL-S	**Y**	**Est.**	**Declar.**	**N**	**Yes**	**Medium**	**N**

3.5. Service Orchestrator (C4.7)

Component ID	Component Name	Development	Owner
C4.7	Service Orchestrator	50%	ECC

GitLab Repository: https://github.com/eccenca/braine/tree/main/service-orchestrator
Containerized: N
Registered on BRAINE platform image registry: N
Deployed as a pod and functional on BRAINE platform: N
Integrated with other platform components: Y – The Service Orchestrator synchronize service metadata such as status between the Global Service Registry and the Resource & Service Catalog.
Status Report:
The Service Orchestrator is under development being partially functional and integrated, however it needs further testing and development.

3.5.1. Technical description

The Service Orchestrator is deprecated and no longer maintained due to the introduction of the BRAINE Web client (See **Figures 3.2.6.** and **3.2.7.**). The Service Orchestrator (C4.7) was replaced entirely by component **C4.6** in section **3.4**

3.6. Monitoring Dashboard (C4.8)

Component ID	Component Name	Development	Owner
C4.8	Monitoring Dashboard	95%	LUH

GitLab Repository: https://gitlab.com/braine/wp4-monitoringsystem-luh
Containerized: Y
Registered on BRAINE platform image registry: Y
Deployed as a pod and functional on BRAINE platform: Y
Integrated with other platform components: Y
Status Report:
The monitoring dashboard is a visualization system for the time-series metric data that are stored in InfluxDB (and Prometheus). The dashboard comes with a pre-configured set of gauges and charts that display various system metrics scraped from node-exporter, including CPU, memory, disk I/O writing, and network traffic metrics. It is also able to visualize additional time-series data generated by the UC applications. The monitoring dashboard relies on the telemetry infrastructure components such as scraper and database from WP3.
Each device (network switch, compute node), platform component (operating system, scheduler, data lifecycle manager, etc), and use case may generate metrics and send them to the telemetry database for storage and processing. The monitoring dashboard can be tuned to extract general or specific (use case-related) metric data, filter and aggregate them and then display charts, gauges, or other visual forms of the data.

3.6.1. Technical description

The monitoring dashboard has been completed and the technical details as well as the source codes have been submitted in D4.2. However, it has remained open to receive potential feedback from the partners, especially from the UCs. No further development is anticipated.

3.6.2. Advancements

LUH is working on a multi instance Influx database to allow each of the UCs to store their monitoring and telemetry data in an isolated database.

3.7. Healthcare Assisted Living (C4.10)

Component ID	Component Name	Use Cases	Owner
C4.10	Exporter for the metrics for the UC1 application 'AI-driven Digital Twin solution for new digital ecosystems enabling Smart Healthcare in Medical and Caregiving Centres'	UC1	IMC

GitLab Repository: private repository
Containerized: Y
Registered on BRAINE platform image registry: N
Deployed as a pod and functional on BRAINE platform: Y
Integrated with other platform components: work in progress
Status Report:
The use 'Healthcare Assisted Living: AI-driven Digital Twin solution for new digital ecosystems enabling Smart Healthcare in Medical and Caregiving Centres'. The goal of the application to create a digital twin of patients using micro-services and continuous collection and analysis of patient data. As part of the task of 'WP4: User-oriented

utilization of the edge' additional component was designed and developed as part of the adaptation the Edge-based system for human-centric applications.

3.7.1. Technical description

In order to perform correctly UC1 application is needed to collect Key metrics required for the UC1 monitoring. Such metrics were defined and relevant for the UC1 application, monitoring tool— metric log connector in short 'exporter'—was designed and developed for the telemetry and application monitoring. The exporter for the metrics for the UC1 application connects to the C3.6 and provides an endpoint "/metrics" and sends GET metrics on request from the Prometheus server. A custom for UC1 application exporter is deployed as a pod and is functional on BRAINE platform.

The exporter written in Go language, deployed as pod in the system and added to the service which will be accessed by Prometheus to provide the set of metrics (as required by Prometheus' exporter implementation).

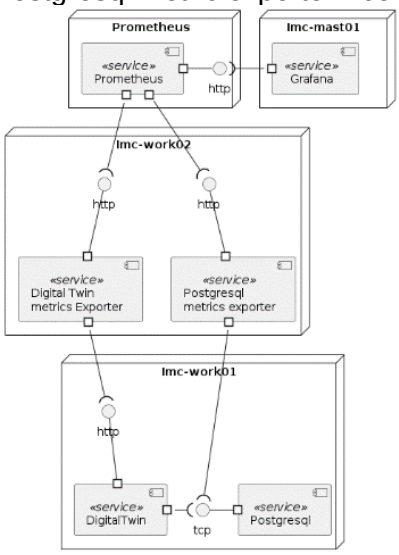
The overall view of metrics for the UC1 application which runs on EMDC is as follows (the number of actual metrics is bigger):

- Queue size for command execution, by using the label "queue_type" we make a separation into several queue types. In doing so we can use one metric and several labels to get a time series for all queues in the system;
- Command execution time in seconds;
- The number of imported values per execution;
- Number of indicator values calculated per command call;
- Number of object state values calculated in one call to the object state calculation command;
- Number of active generators;
- Total number of registered users in the system;
- Total number of models in the system;
- Total number of sensors;
- Total number of data elements;
- Total number of indicators;
- Total number of indicator values for all indicators;
- Total number of sensor values for all sensors;
- The total number of values of the data items for all data elements.

The following diagram represented internal workflow of the component on the BRAINE platform:

- imc-mast01: Grafana service.
- imc-work01: Smart hospital application and Postgresql
- imc-work02: Metric exporter collects metrics on calculations, indicators, imports - everything that is related to the work of the application itself.

- Postgresql metric exporter - collects metrics of the Postgresql itself



C4.10 component “Exporter for the metrics for the UC1 application ‘AI-driven Digital Twin solution for new digital ecosystems enabling Smart Healthcare in Medical and Caregiving Centres’” collects more than >250 metrics for the application.

3.7.2. State of the Art (SOTA)

Compared to the state-of-the-art, these advancements have significantly improved the monitoring and observability of containerized applications in micro-data center environments. By incorporating such advancements in monitoring and observability for healthcare applications, we can ensure the reliable and secure operation of our systems. By leveraging Docker, Kubernetes, Prometheus, and Grafana in UC1 in dealing with the healthcare environments, the BRAINE enables better visibility into application performance, proactive issue detection, and efficient resource management, ultimately enhancing the quality of care provided to patients.

3.7.3. Advancements

The component is linked to the UC1 and cannot be used standalone without other WP4 components e.g. C4.8 and a multi instance Influx database that will allow each of the UCs to store their monitoring and telemetry data in an isolated database.

3.7.4. Performance Evaluations and comparisons

3.8. Network Telemetry Framework (C4.14)

Component ID	Component Name	Development	Owner
C4.14	Flow telemetry agent	100%	MLNX
GitLab Repository: N Containerized: Y Registered on BRAINE platform image registry: N Deployed as a pod and functional on BRAINE platform: N Integrated with other platform components: Y – Info: Integrated with Flow P4 program (C4.14.1) Status Report:			

- The flow telemetry agent was tested to add/remove selected traffic flow, the P4 tables are updated with the add/remove entries and telemetry events are sent to Monitor & exported (C4.14.2).

Component ID	Component Name	Development	Owner
C4.14.1	Flow P4 Program	100%	MLNX

GitLab Repository: N
Containerized: Y
Registered on BRAINE platform image registry: N
Deployed as a pod and functional on BRAINE platform: N
Integrated with other platform components: Y – Info: Integrated with Flow telemetry monitoring and exporter (C4.14.2) for streaming telemetry data
Status Report:
The P4 program code is done, HW tables are created and new flows can be added to those tables.

Component ID	Component Name	Development	Owner
C4.14.2	Flow telemetry monitor & exporter	100%	MLNX

GitLab Repository: N
Containerized: Y
Registered on BRAINE platform image registry: N
Deployed as a pod and functional on BRAINE platform: N
Integrated with other platform components: Y – Info: Integrated with Telemetry Adapter for streaming telemetry data.
Status Report:
HW telemetry events are collected by the component and also exported to remote collector via gRPC.

3.8.1. Technical description

Considerations for AI-rich environments

Modern GPUs with improved computational capabilities are moving the bottleneck from compute elements to the communication infrastructure. Distributing AI workloads over multiple workers connected with high-speed network cause to enlarge the **bandwidth demand** and predictable latency that can be solved by increased cost of using networks with constant bisectional bandwidth or alternatively can be applied in cost effective shared multitenant oversubscribed networks.

Nature of ML algorithms create an asymmetric many-to-one or many-to-many workflows that may result in communication loss and **high task completion latency**. Oversubscription and resource limit can be resolved by changing the learning algorithms, adapting the synchronization approaches, optimizing the worker locations, using in-network computation, and providing efficient feedback from the network back to the transport and applications, so the latter can adapt its behavior.

Specification of the flow-based telemetry

Network telemetry can help to identify network failures, infrastructure malfunctioning, performance bottleneck and behavioral inefficiencies. Detailed analysis of those problems can help to find application that cause such problems or network configurations that limit the network performance.

Flow level telemetry (in-band or post-card based) will provide flow benchmarking in terms of latency, routing, and path load. Attaching application labeling to the network flows will

provide end-to-end context and enable application-level analysis and identify multitenancy dependencies.
The flows to be monitored are defined through specifically enforced flow entries. Flow definition is application and topology specific and should be defined in programmable way to enable verity of use-cases and setups. Flows can be simple (e.g., p2p) or can have a network span e.g., for many-to-many communication pattern.
After the flows are defined, every flow will produce a benchmark for the subset of packets that pass via specified flow. The benchmark results will be collected in the central location to aggregate the data for visualization and immediate and historical analysis.

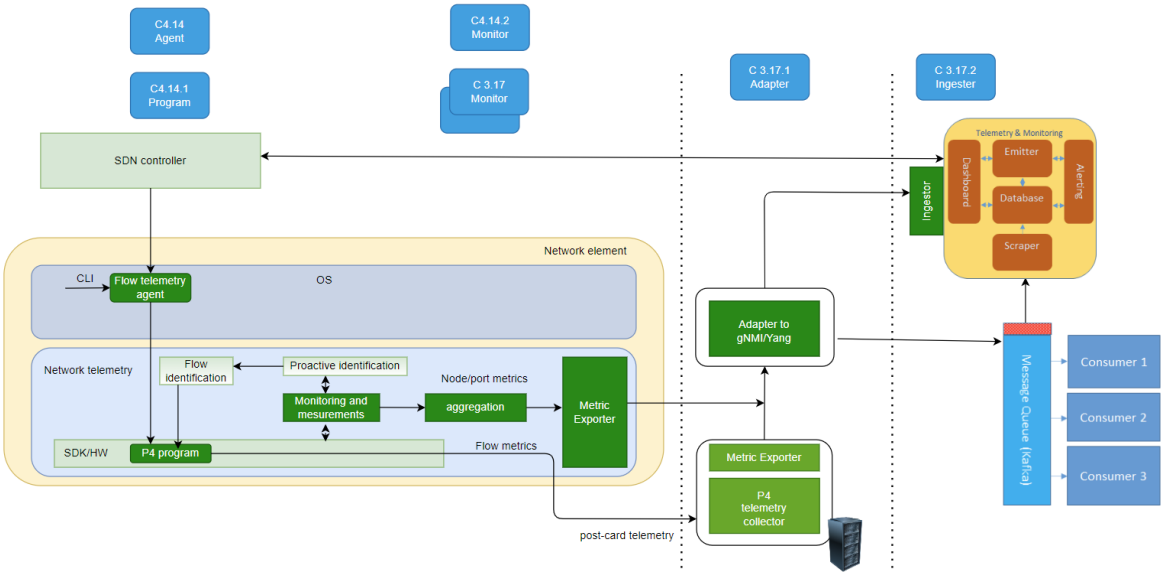


Figure 3.16 Network telemetry monitoring system

The telemetry framework is responsible to generate and collect telemetry information regarding the single network node.
The framework consists of the following sub-components:

- P4 program
  - Runs in HW so it doesn't affect the normal network behaviour
  - Identifies the flows of interest that should be monitored
  - Produces the subset of information that should be exported
  - Exports telemetry information and generating telemetry events
- P4 agent application
  - Interacts with higher layer controllers to for configuration and flow description
  - initializes and configures P4 program to add/remove flows that should be monitored.
- Monitoring and export unit
  - processes raw telemetry data
  - converts it to a format that can be used by collectors

Following interfaces are used to communicate between components and external interfaces:

- P4 programs based on P4-lang to program data plane of network elements and telemetry collector
- gRPC to export the data from Network telemetry framework to adapter unit (see above Figure 7.1)

Flow telemetry Agent (C.4.14)

The Flow Telemetry agent is responsible to initialize and configure the P4 program that was auto-generated by MLNX P4 backend compiler. This interface will be called by SONiC NOS CLI. The agent also configures the needed HW capabilities to enable telemetry reporting (Mirror). An example configuration is a Flow (5 tuples) that should be monitored. Once the selected network session/flow was added to P4 tables, the HW will send telemetry events to the Telemetry monitor & exporter components.

Flow P4 program (C4.14.1)

The Flow telemetry P4 program is responsible to configure the low-level HW to support the P4 program written in P4-lang. In the flow telemetry case, this is a P4 table monitoring 5 tuples and mirroring the sampled traffic to the switch's CPU for reporting to the remote collector.

The P4 table holds entries with 5 tuple keys and mirror actions. Below is the P4 table used for flow telemetry:

```
table table_flow_sampling {
    key = {
        headers.ipv4.src_addr : exact;
        headers.ipv4.dst_addr : exact;
        headers.ipv4.protocol : exact;
        headers.tcp.src_port : exact;
        headers.tcp.dst_port : exact;
        headers.ipv4.hdr_checksum : ternary;
    }
    actions = {
        NoAction;
        do_cpu_mirror;
    }
    default_action = NoAction();
    direct_counter = dc_flow_out_sampling;
    size = 1000;
}
```

Figure 3.17 P4 source code to define the flow telemetry table

Telemetry Monitor and exporter (C4.14.2)

The telemetry monitor & exporter is responsible to collect and report telemetry data from network elements regarding network node behaviours and the traffic passing over the network node.

This component will wait for selected telemetry events from HW (that was configured by the P4 program C4.14.1) and will generate a report via gRPC to the Adapter component (WP3.3 C13.17.1).

Below is the example gRPC proto3 example that is used to stream data from the telemetry monitor & exported to the Adapter component.

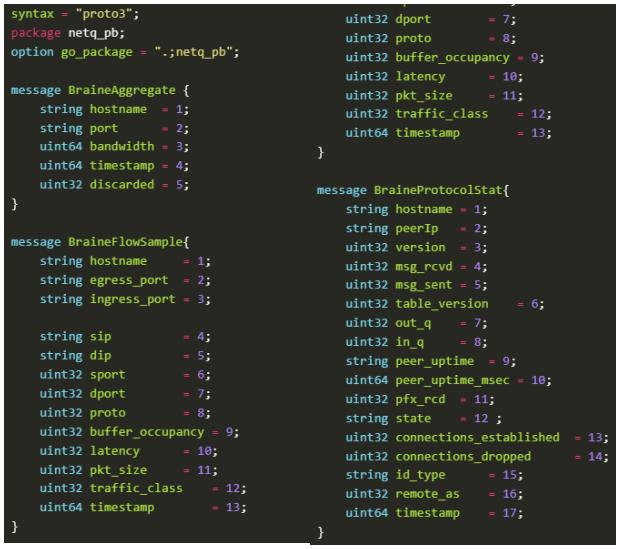


Figure 3.18 gRPC proto3 example used in the telemetry monitor & exporter

Telemetry Adapter
Developed in WP3 (C 3.17.1)

3.8.2. State of the Art (SOTA)

Current telemetry solutions relay on various sources that produces telemetry data in independent manner (e.g., draft-wu-t2trg-network-telemetry or pingmesh):

- SNMP traps and counters
- Syslog
- Flow information
- Topology
- Proactive probe metering

Those solutions provide limited visibility to the network resource state, are hard to correlate, don't care on the network operation state, work in periodic manner, are independent of actual network traffic pattern and add significant load to the network and network control plane.
Proposed approach is always on, can be provisioned on demand, very focused on real applications that are running in the given network and have low impact on underlying network as mostly run in HW.
For AI/ML workload P4 programmability and flow base telemetry is new option that can lead to better application performance utilizing detailed on-demand telemetry data produced in HW with application flow resolution.

3.8.3. Advancements

The key advancements for the network devices are based on the features shared in Section 3.6.1.
AI/ML workloads require high network bandwidth, low latency, and reliable transport to improve job completion time.
As mentioned above current telemetry solutions are too generic and doesn't enable to do efficient job scheduling based on the network performance, do not enable applications to receive updated network state to correlate their behaviour.
To implement advanced on demand network telemetry we propose to include:

- Programmable (P4) per flow telemetry – to provide HW generated application tailored data
- Per port statistics (e.g. bandwidth and packet drops) – to detect network outages
- Network protocol state (e.g., BGP) – to detect major network event
- Streaming telemetry data (communicate network events in real time, gRPC) – to reduce load on the network devices and provide advanced filtering to get only requested data

The streaming of telemetry data was done using YANG data model to structure the network information and enable fast integration with 3rd party tools. Structuring the data in a standard YANG model enable multi-vendor devices to stream telemetry data.
The key metrics collected per flow are:

- Latency (nsec)
- buffer-occupancy (Byte)
- Ingress/Egress ports

Using advanced telemetry in SLA brokers:

The advanced telemetry data sent by the network elements using the above infrastructure can be used by the SLA Broker from WP3.4 to enable the SDN controller to react to the network changes in real-time.

In WP3.4 the SDN controller can react on high latency and on network failures alarms, an example improvement can be to also act on high buffer-occupancy alarms and enable early detection of congestion buildup that impact latency.

Also, SDN controller based on bandwidth information collected from the network can select a better placement for the bandwidth hungry workloads.

In WP3.3 the BRAINE DKB delivers CPU and RAM telemetry to enable AI/ML models to be able to predict resource demand.

The proposed solution is not limited only for above metrics but is capable to accommodate more metrics like latency and also flow based telemetry, to improve the ML base workload predication models.

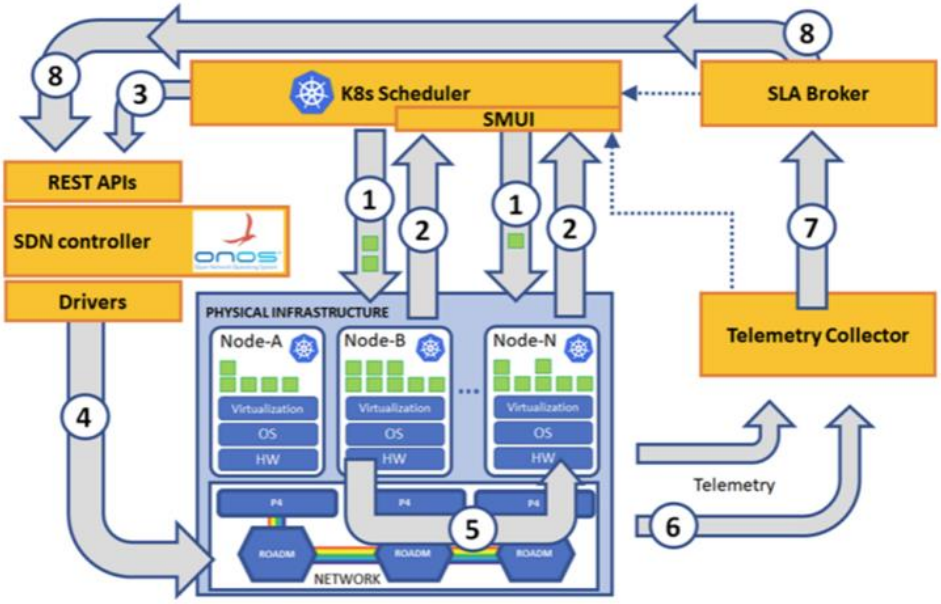


Figure 3.19 WP3.4 Main EMDC components and closed-loop telemetry workload.

3.8.4. Performance Evaluations and comparisons

In this task evaluation we used the Telegraf, InfluxDb and Grafana (TIG) stack.
This was used for viewing and analyzing the telemetry data.
TIG is constructed of the following components:

- Telegraf: The tool that collects the data from the input with a specific format and forwards it to the Influxdb
- Influxdb: The database where the data is stored (e.g.: dropped packets per port)
- Grafana: The visualization dashboard that presents the received data from the Influxdb in a graphical manner

In the below figure we can see an example for flow telemetry and the detection of increase in flow latency due to queue/buffer build up and jump in bandwidth.

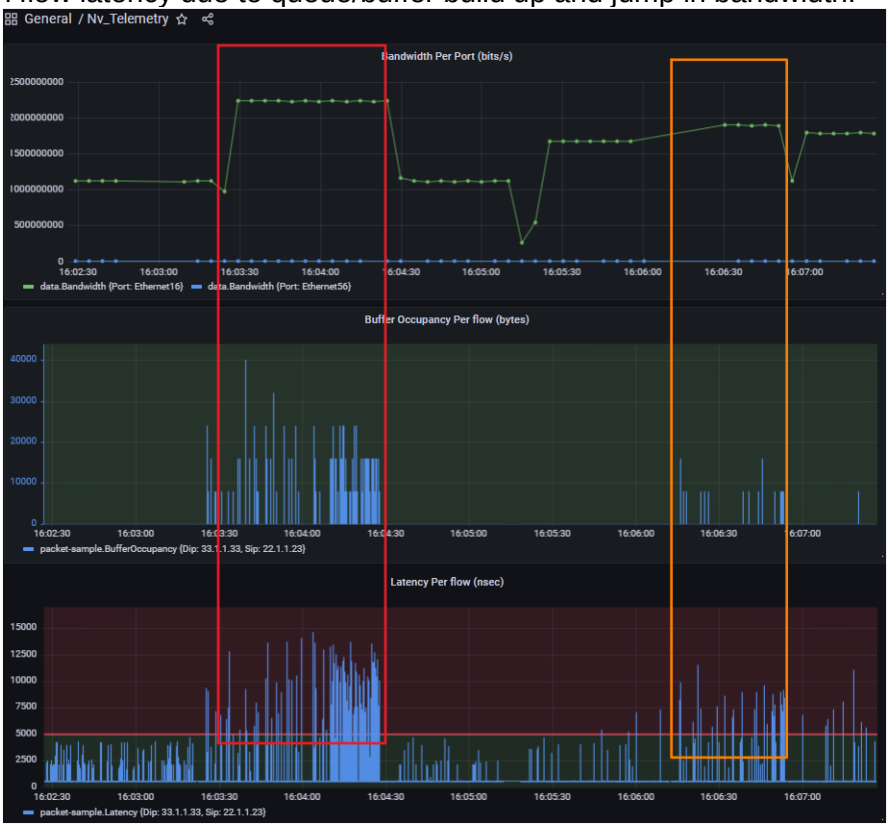


Figure 3.20 Grafana view for flow base telemetry

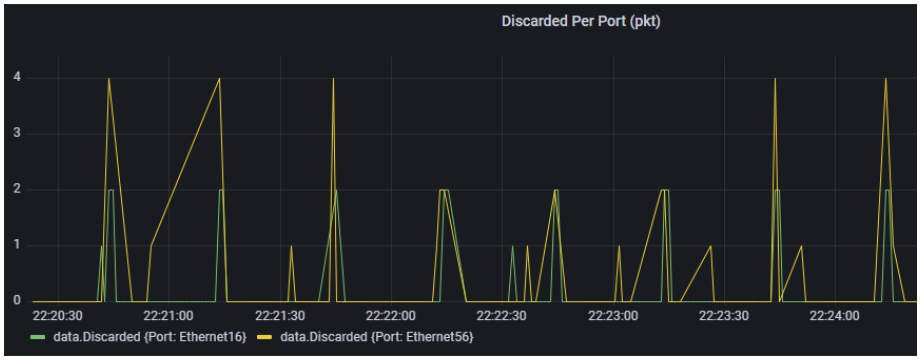


Figure 3.21 Grafana view for discarded packet

In the below figure we can see an example for BGP peer connection drop detection that can impact the network.

Figure 3.22 Grafana view for BGP protocol stat

4. Conclusion

This document provides the status report for the second half of the key software components developed under WP4 for development and integration as part of the overall BRAINE platform. Most of the development effort has been completed. Some components are still undergoing integration and testing with other WPs, more specifically with use-cases in WP5. Partners are planning the integration activities with corresponding use-cases.

5. References

[1] ETSI, “Multi-access edge computing (MEC); framework and reference architecture,” ETSI GS MEC 003, V3.1.1, pp. 1–29, 2022.

[2] 3GPP, “Technical specification group services and system aspects; management and orchestration; study on enhancements of edge computing management,” TR 28.814 V17.0.0, pp. 1–49, 2021.

[3] A. Giannopoulos et al., "Supporting Intelligence in Disaggregated Open Radio Access Networks: Architectural Principles, AI/ML Workflow, and Use Cases," in IEEE Access, vol. 10, pp. 39580-39595, 2022, doi: 10.1109/ACCESS.2022.3166160.

[4] E. Pateromichelakis, F. Moggio, C. Mannweiler, P. Arnold, M. Shariat, M. Einhaus, Q. Wei, . Bulakci, and A. De Domenico, “End-to-end data analytics framework for 5G architecture,”

IEEE Access, vol. 7, pp. 40 295–40 312, 2019.

[5] K. Papadakis-Vlachopapadopoulos, I. Dimolitsas, D. Dechouniotis, E. E. Tsiropoulou, I. Roussaki, and S. Papavassiliou, “Blockchain-based slice orchestration for enabling cross-slice communication at the network edge,” in 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2020, pp. 140–147.

[6] B. Ma, W. Guo, and J. Zhang, “A survey of online data-driven proactive 5G network optimisation using machine learning,” IEEE Access, vol. 8, pp. 35 606–35 637, 2020.

[7] D. Giannopoulos, P. Papaioannou, C. Tranoris, and S. Denazis, “Monitoring as a Service over a 5G Network Slice,” in 2021 Joint European Conference on Networks and Communications and 6G Summit (EuCNC/6G Summit), 2021, pp. 329–334.

[8] J. Prez-Romero, V. Riccobene, F. Schmidt, O. Sallent, E. Jimeno, J. Fernndez, A. Flizikowski, I. Giannoulakis, and E. Kafetzakis, “Monitoring and analytics for the optimisation of cloud enabled small cells,” in 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2018, pp. 1–6.

[9] A. Mudvari, N. Makris, and L. Tassiulas, “ML-driven scaling of 5G Cloud-Native RANs,” in 2021 IEEE Global Communications Conference (GLOBECOM), 2021, pp. 1–6.

[10] P. Veitch, J. Browne, and J. Krogell, “An integrated instrumentation and insights framework for holistic 5G slice assurance,” in 2020 6th IEEE Conference on Network Softwarization (NetSoft), 2020, pp. 247–251.

[11] F. Cugini, D. Scano, A. Giorgetti, A. Sgambelluri, P. Castoldi, and F. Paolucci, “P4 programmability at the network edge: the BRAINE approach [invited],” in 2021 International Conference on Computer Communications and Networks (ICCCN), 2021, pp. 1–9.

[12] L. Sanabria-Russo and C. Verikoukis, “A cloud-native monitoring system enabling scalable and distributed management of 5G network slices,” in 2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom), 2021, pp. 42–46.

[13] R. Casellas, R. Martnez, L. Velasco, R. Vilalta, P. Pavn, D. King, and R. Muoz, “Enabling data analytics and machine learning for 5G services within disaggregated multi-layer transport networks,” in 2018 20th International Conference on Transparent Optical Networks (ICTON), 2018, pp. 1–4.

[14] S. Hu, W. Shi, and G. Li, “CEC: A Containerized Edge Computing Framework for Dynamic Resource Provisioning,” IEEE Transactions on Mobile Computing, pp. 1–1, 2022.

[15] S. Pramanik, A. Ksentini, and C. F. Chiasserini, “Characterizing the computational and memory requirements of virtual rans,” in 2022 17th Wireless On-Demand Network Systems and Services Conference

(WONS) IEEE, 2022, pp. 1–8.

[16] T. Subramanya and R. Riggio, “Machine learning-driven scaling and placement of virtual network functions at the network edges,” in 2019 IEEE Conference on Network Softwarization (NetSoft), IEEE, 2019, pp. 414–422.

[17] X. Foukas and B. Radunovic, “Concordia: Teaching the 5G vRAN to share compute”, in Proceedings of the 2021 ACM SIGCOMM 2021 Conference, 2021, pp. 580–596.

[18] A. Flizikowski, E. Alkhovik, M. M. Mowla, and M. A. Rahman, “Data Handling Mechanisms and Collection Framework for 5G vRAN in Edge Networks,” 2022, IEEE Conference on Standards for Communications and Networking (CSCN), ACCEPTED.

[19] Flizikowski, Adam; Alkhovik, Evgeniy; Mowla, Md Munjure; Rahman, Md Arifur (2022): Importance of Workload Prediction of Virtualized RAN in the Edge Micro Data Center. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.21644708.v1

[20] R. Agrawal and R. Adhikari, “An introductory study on time series modeling and forecasting,” Nova York: CoRR, 2013.

[21] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, “N-BEATS: Neural basis expansion analysis for interpretable time series forecasting,” arXiv preprint arXiv:1905.10437, 2019.

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga et al., “Pytorch: An imperative style, high-performance deep learning library,” Advances in neural information processing systems, vol. 32, 2019.

[24] Y.-W. Cheung and K. S. Lai, “Lag order and critical values of the augmented dickey–fuller test,” Journal of Business & Economic Statistics, vol. 13, no. 3, pp. 277–280, 1995.

[25] L. N. Smith, “Cyclical learning rates for training neural networks,” in 2017 IEEE winter conference on applications of computer vision (WACV). IEEE, 2017, pp. 464–472

[26] Chen, Tianqi, et al. "{TVM}: An automated {End-to-End} optimizing compiler for deep learning." 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). 2018.

[27] Alsulbi, Khalil, et al. "Big data security and privacy: A taxonomy with some HPC and blockchain perspectives." International Journal of Computer Science & Network Security 21.7 (2021): 43-55.

[28] Rhahla, Mouna, Sahar Allegue, and Takoua Abdellatif. "Guidelines for GDPR compliance in Big Data systems." Journal of Information Security and Applications 61 (2021): 102896.

[29] Truong, Nguyen Binh, et al. "Gdpr-compliant personal data management: A blockchain-based solution." IEEE Transactions on Information Forensics and Security 15 (2019): 1746-1761.

[30] Chikhaoui, et al. Multi-objective optimization of data placement in a storage-as- a-service federated cloud. ACM Transactions on Storage (TOS) 17, 3 (2021), 1–32.

[31] Li, C., Bai, J., et al. “Joint optimization of data placement and scheduling for improving user experience in edge computing”. Journal of Parallel and Distributed Computing 125 (2019), 93–105.
[32] Long, S.-Q et al. “A multi-objective optimized replication management strategy for cloud storage cluster.” Journal of Systems Architecture 60, 2 (2014), 234–244.

[33] Padmanaban, R., et al. “HadoopSec: Sensitivity-aware Secure Data Placement Strategy for Big Data/Hadoop Platform using Prescriptive Analytics.” GSTF Journal on Computing (JoC) 5, 3 (2020).
[34] Revathy, P., et al. “Hadoopsec 2.0: Prescriptive analytics-based multi-model sensitivity-aware constraints centric block placement strategy for hadoop.” Journal of Intelligent & Fuzzy Systems 39, 6 (2020), 8477–8486.

[35] C. -C. M. Yeh et al., "Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets," 2016 IEEE 16th International Conference on Data Mining (ICDM), 2016, pp. 1317-1322, doi: 10.1109/ICDM.2016.0179.