D4.1 Data Management and Coupling Technologies (M11)

Date: December 12, 2023

EuroHPC
Joint Undertaking

Document Identification			
Status	Final	Due Date	30/11/2023
Version	1.0	Submission Date	12/12/2023

Related WP	WP4	Document Reference	D4.1
Related Deliverable(s)		Dissemination Level (*)	PU
Lead Participant	ATOS	Lead Author	Jesús Gorroñogoitia (ATOS)
Contributors	SZE, UNISTRA, PSNC, MTG, ATOS, USTUTT	Reviewers	Flavio Galeazzo (USTUTT)
			Dimitrios Tsoumakos (ICCS)

Keywords:
data management, data ingestion, data transfer, coupling technologies

Document Information

List of Contributors	
Name	**Partner**
Jesus Gorroñogoitia Cruz	ATOS
Marcos Sánchez	MTG
Ángela Rivera	MTG
Luis Torres	MTG
Piotr Dzierżak	PSNC
Marcin Lawenda	PSNC
Sameer Haroon	USTUTT
Luca Berti	UNISTRA
Christophe Prud'homme	UNISTRA
Zoltán Horváth	SZE

Document History			
Version	**Date**	**Change editors**	**Changes**
0.1	04/10/2023	ATOS	Table of Contents
0.2	24/10/2023	ATOS	Section 2.4, 2.5.1
0.3	13/11/2023	USTUTT, PSNC, MTG, ATOS	Sections 2.1, 2.2, 2.3, 3.2
0.4	17/11/2023	PSNC	Section 2.1, Annex
0.5	21/11/2023	MTG, PSNC, UNISTRA	Sections 2.3, 3.1, 3.2 and Introduction Version for peer-review
0.6	28/11/2023	ALL	Corrections after peer-review
0.7	29/11/2023	ATOS	Version for QA
1.0	12/12/2023	ATOS	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	**Who (Partner short name)**	**Approval Date**
Deliverable Leader	Jesús Gorroñogoitia (ATOS)	12/12/2023
Quality Manager (deputy)	Dennis Hoppe (USTUTT)	12/12/2023
Project Coordinator	Marcin Lawenda (PSNC)	12/12/2023

Table of Contents

List of Figures

List of Tables

List of Acronyms

Abbreviation / acronym	Description
ACID	Atomicity, Consistency, Isolation, Durability
AI	Artificial Intelligence
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CFD	Computational Fluid Dynamics

CKAN	Comprehensive Knowledge Archive Network
CLI	Command Line Interface
CPU	Central Processing Unit
CRUD	Create Read Update Delete
CSV	Comma separated values
DAG	directed acyclic graph
DML	Data Manipulation Language
DMS	Data Management System
ETL/ELT	Extract Transform Load
GB	Gigabyte
GC	Global Challenges
GIS	Geographic information system
GUI	Graphical User Interface
HDFS	Hadoop File System
HDP	Hortonworks Data Platform
HPC	High Performance Computing
HPDA	High Performance Data Analytics
HTTP	Hypertext Transfer Protocol
IAM	Identity Access Management
IDM	Identity Management
IO	Input Output
IP	Internet Protocol
JSON	Java Script Object Notation
LTS	Long Term Support
MML	Multiscale Modelling Language
M<X>	Month <x>
OSS	Open-Source Software
PDF	Portable Document Format
PNG	Portable Network Graphics
PubSub	Publication Subscription
RAM	Random Access Memory
RDBMS	Relational database management system
RDF	Resource Description Framework
RDMA	Remote Direct Memory Access
RES	Renewal Energy Sources

REST	Representational State Transfer
SCP	Secure Copy Protocol
(S)FTP	(Secure) File Transfer Protocol
SQL	Structured Query Language
SSH	Secure Shell
SSO	Single Sign On
TB	Terabyte
TIFF	Tagged Image File Format
UAP	Urban Air Pollution
UB	Urban Building
UML	Unified Modelling Language
UQ1	Uncertainty Quantification
URL	Uniform Resource Locator
UTF	Unicode Transformation Format
UUID	Universally Unique Identifier
VCS	Version Control System
VDF	Version Definition File
VM	Virtual Machine
VTK	Visualization Toolkit
WP<X>	Work-package <X>
WMS	Web Map Service
WRF	Weather Research and Forecasting
XML	Extensible Markup Language
YARN	Yet Another Resource Negotiator
yMMSL	YAML version of the Multiscale Modeling and Simulation Language

Executive Summary

Achieving HiDALGO2's objectives requires the use of advanced environmental simulation systems to estimate and predict the evolution of the different pilots defined in the project.

Due to the complexity of these systems, it is necessary to use HPC resources (such as the EuroHPC supercomputers) to enhance simulation performance. Additionally, employing HPDA and AI techniques is essential for improving the accuracy of results. These approaches require the agile and efficient management of large volumes of data. The resulting data management must address the flow of data to and from HPC systems, from and to the HiDALGO2's centralized storage that ensures its availability for analysis.

This requires knowing the characteristics of the data to be used in each use case and defining how this data will be handled in the HiDALGO2 ecosystem.

This report defines data required by the pilots, specifies data structures as well as fetching and delivering techniques depending on processing methods, propose data handling solutions, and additionally, it introduces coupling solutions applicable to project requirements. The document is a living document that will be further updated and reported in Deliverable D4.2 due in M11.

This document also addresses the requirements for coupling among various models within the pilots. These models pertain to different areas of interest, operating with distinct temporal and spatial scales. Harmonizing them for interoperability is necessary. Depending on each case, this can be achieved through loose or tight couplings. The document outlines a proposed solution and references various new technologies that can be integrated for this purpose throughout the project.

1 Introduction

1.1 Purpose of the document

The purpose of this document is twofold: on the one hand, to serve as a reference for the creation of a technological solution capable of responding to the data management needs in HiDALGO2, from the collection of data from various sources, its storage in easily accessible locations, its availability in the different HPC systems used, its management and modification as well as the necessary maintenance of the associated metadata; and, on the other hand, to define the architecture to support the coupling between models from different pilots of the project.

1.2 Relation to other project work

This deliverable establishes the data management needs and proposes a solution to address them and thus is related to WP2, which establishes the requirements to be fulfilled and the workflow orchestration that has to be integrated with the data management solution proposed, with WP4, in which this activity is integrated and poses some requirements on data exchange for HPDA, Visualization, and UQ, and with WP5, as the data flow aims to address both the data management of the pilots developed in this WP and provide a solution for their coupling within the same WP.

1.3 Structure of the document

This document is organized into several chapters, the contents of which are shown below.

Executive summary. It frames the context of the document in the project, summarizes the objectives of this document and briefly describes its main contents.

Chapter 1. *Introduction*. It establishes the purpose of the document, how its fit within the project framework and the structure of the document.

Chapter 2. *Data Management*. A description of pilot requirements, functional specifications, the main technologies that can respond to the needs raised, and the proposed solution that best suits them are described in this chapter.

Chapter 3. Coupling Technologies. A description of the coupling scenarios within HiDALGO2 and the State of the Art on coupling technologies that can fit the information exchange and message interchange among them.

Chapter 4. *Conclusions*. The main foreseen solutions for data management and coupling technologies are presented in the form of conclusions.

2 Data Management

This section introduces the functional and technical specifications of the overall HiDALGO2 solution for the data management system (DMS). The functional specification of the DMS that is proposed in Section 2.2 addresses the requirements collected from the pilots' stakeholders, and other technical work-packages, and reported in section 2.1. In Section 2.3, we collect the baseline technologies that have been selected to build up the HiDALGO2 DMS in accordance to the architecture design introduced in Section 2.4. Section 2.5 describes the initial M11 implementation of the HiDALGO2 DMS, consisting of some baseline technologies deployed into the HiDALGO2 Cloud development infrastructure (see D2.4 [11]) and an initial development for data transfer.

2.1 Requirements

Considering the demanding requirements of the Global Challenges (GC) system in terms of performance, efficiency and reliability, the choice of a Data Management System (DMS) must be carried on following a specific set of principles. DMS is an essential part of the workflow in large-scale processing systems. It should be emphasized that the DMS module is intensively utilised by many applications at various stages of scenario implementation. Moreover, data access should be enabled in parallel mode for multiple services to cope with application demands for simultaneous operations e.g., performed on the same file(s).

The analysis should also consider the increase in the amount of data (scalability) and their heterogeneity (format and purpose), taking into account various data sources, including e.g.: data exchange between pilot applications within coupled scenarios. This brings us to two main problems that should be solved in the proposed data management framework: efficient data management and reliable computation.

This section presents the requirements considering the above assumptions. Below is a list of the main requirements the DMS system must meet.

1. **FAIR principles.** When analysing the requirements of data storage systems, the FAIR (Findability, Accessibility, Interoperability and Reusability) principles come to the fore [22]. On the one hand, they emphasize the possibility of free data exploration in the context of their use in large data warehouses, and on the other, they emphasize the automation of data analysis processes. This last aspect is especially important in the face of increasing information resources difficulties in understanding their meaning without an appropriate computational support.

1. **Data preservation.** Due to the handling of valuable scientific data, which are often the result of long-term research, the data must be provided with an appropriate level of security through archiving, including the ability (supported by framework) to transfer data to other locations (physical and geographical).

2. **Interoperability.** Despite mentioning interoperability among the basic principles of FAIR, it is necessary to highlight this feature in a separate chapter. In a generally available definition, this characteristic is the ability to access and process data from multiple sources without loss of meaning, and then integrate that data for mapping, visualisation, and other forms of representation and analysis. For the HiDALGO2 project, cooperation with external systems (e.g., Copernicus, GeoPortal, OpenStreetMap) is necessary. This applies to both inflow (importing data from external sources for further processing) and outflow (e.g., when we want to make data available to generally recognized external repositories). Data collected from many different sources must be stored and organized in a consistent way so as not to lose their meaning. Therefore, the design environment should provide uniform data transport mechanisms that enable the preservation of their values along with their contextual meaning (e.g., appropriate data structure).

2. **Flexible/Extensible Interfacing.** In the context of data access, two basic ways of accessing data should be distinguished: access via the web interface and the API interface. A web interface is necessary taking into account the specificity of access to both services and data of the HiDALGO2 environment. The user gains access to the infrastructure and offered functionality through a set of web tools, which on the one hand enables easier operation and, on the other hand, guarantees an appropriate level of security (main entry point for the user is portal). Access via the API interface is necessary taking into account interoperability at the level of services exchanging data (e.g., REST [3] or GraphQL[1]), which is to ensure efficiency in their transport.

3. **Data exploration.** Effective data exploration (especially in the face of their large accumulation) is crucial for efficient calculations and is also one of the fundamental principles of FAIR. This process also concerns aspects such as initial visualization (on the web access level), obtaining data from external

[1] GraphQL - A query language for API, https://graphql.org/

sources and structuring storage, which are consistent with the workflow in the organization.
The ability to browse data (via the web) to find the appropriate data set (e.g., required at a given stage of processing) also includes an appropriate description of the data, which facilitates this process. This requires defining a set of metadata that uniquely defines the content of the stored files, taking also into account the scientific domain they concern.

4. **Efficient data transfer.** Additional performance requirements are placed on the DMS when data must be efficiently delivered to the processing site. It should be considered that in GC systems we are dealing with large-capacity data sets. We often deal with a multidimensional parameter space, processed in parallel, which forces the transfer of many files at the same time. This often excludes access via an inefficient web channel and imposes effective solutions based on the API interface.
Moreover, one of the main goals of the HiDALGO2 project is the effective convergence of HPC and HPDA systems. This constitutes additional requirements for the data management system, related to interoperability with the infrastructure of both systems, enabling effective access and collection of data sets required by pilot applications.

3. **Scalability** – Due to the developing system, it is difficult to predict the target size of storage resources at the beginning. Therefore, it is assumed that the system can be initially configured with reasonable (medium-sized) resources, and over time, if necessary, it would adapt (expand) to emerging requirements. This means that the offered framework should be scalable and flexible in terms of performance and capacity.

4. **Security** - Ensuring an appropriate level of security for stored and transmitted data is a fundamental requirement. However, this requirement should be extended to include other aspects ensuring appropriate access to data for specific user groups. The most important include structuring permissions, granting access to specific files in order to make them available to a specific group in the project, and sometimes even publicly.

HiDALGO2 collects requirements in two categories: scenarios and requirements. Scenarios define user stories collected from project partners so they define high-level requirements for the project environment and are related to pilots, the platform, data, the HPC/HPDA infrastructure, or other aspects of the project.

Requirements are derived from scenarios and are categorised as functional (e.g., requirements for the functionality of the system or the created data) and non-functional (e.g., look & feel, user experience, performance, etc.).

An initial list of both (scenarios and requirements) is delivered along with Deliverable D2.1 [4]. DMS scenarios are defined from SCO-DMA-001 to SCO-DMA-005, while requirements are specified in REQ-DMA-001 - REQ-DMA-016.

However, it should be assumed that as work on the HiDALGO2 system progresses, new requirements will be defined, which will be implemented in order to meet the requests of individual developer groups in the project.

In this iteration of requirements gathering, several more demands were indicated, mainly related to specifying certain operations in terms of data format, access protocols, and technological conditions. Newly collected requirements are included in the range from REQ-DMA-017 to REQ-DMA-041 and specified in Annex 1 of this document.

2.2 Functional Specification

Drawing from the breakdown of requirements collected in the previous section, we have a clear-cut listing of features that the HiDALGO2 Data Management Infrastructure must support, in order to cover all the possible use cases that Pilot Partners and End Users could need during and after the project duration.

These features have been split across two main areas of requirement, namely Data Storage and Data Transfer.

2.2.1 Data Storage

2.2.1.1 Data Types

The Data Management System will support various data types for storage and processing purposes. These include all three classical types of data: structured, semi-structured and unstructured data, so that the system covers use cases such as storing data in tables, (e.g., graphs, XML/JSON documents). This also ensures that unstructured data, e.g., text logs files, are also equally managed.

The system will also support the online storage of streaming data and messages, e.g., from sources like Weather Stations, GIS datasets. The system should also take into consideration the use of supporting serialization formats such as Protocol Buffers, Avro, or Parquet.

2.2.1.2 Data Formats

The system will formally support all required file and data formats, as specified in the requirements collected in Annex1, although it will consider as well new requirements reported in the future. Formats include both textual, such as CSV, JSON, ASCII doc, with all required encodings (e.g., Univode UTF8), as well as binary formats e.g. TIFF (GeoTIFF), h5, shapefile (.shp, .shx, .dbf), GRIB2, WRF, NETCDF4, PNG, VTK, MSH.

2.2.1.3 Dataset Features

The system will support several features to make it as convenient as possible in dealing with the diverse sets of data in HiDALGO2.

This includes features like supporting the versioning of datasets and tracking these versions transparently. All datasets will be annotated with the required metadata and allow for standard and custom scheme definitions of the metadata, as well allow for the browsing through of datasets with the help of this metadata.

The System will also allow data querying based on the internal schema of relational data, e.g., SQL-like querying for RDBMS storage, and based on the internal indexing of unstructured and (semi) structure data.

Aside from querying, the system will expose multiple APIs to support the transfer and use of these data sets. These include APIs such as:

- API for CRUD Operations

  The system enables operations such as “Create”, “Read”, “Delete”, on the stored datasets. (“Update” will generally not be possible, as most stored datasets will be immutable).

- API for Search/Query

  The system exposes an API to allow for searching/browsing through stored datasets, based on hierarchical dataset structures.

- CLI/REST/Web Channels for APIs

  The system will support all; CLI, REST and other Web API channels for storage e.g., for remote interoperability with data storage (CRUD) and associated services, such as search and querying.

2.2.2 Data Transfer

Aside from storage of data, the efficient transfer of data, internally and externally to the project is also a fundamental requirement. The three main features that the system provides here are:

- HPC to/from Central Storage

The system will support fast and reliable bi-directional data transfer from and to HPC and the central storage system. In order to achieve this, SSH-based transfer mechanisms between HPC and central storage, such as SCP, SFTP, smount, will be applied. Identity Management solutions such as Keycloak and Vault will be leveraged to ensure that the correct user credentials are used for HPC access and data transfer.

- Web Providers to Central Storage

  The system will allow downloading datasets from Internet data providers, such as Copernicus, to the central storage, using web channels such as HTTP(S), REST, (S)FTP.

- Internal Servers to Central Storage

The system should allow downloading/transferring datasets from other internal data providers to the central storage, using server-specific channels.

2.3 Baseline Technologies

To best serve users and developers with the requested set of features, the ideal collection of tools is to be selected, which future-proof HiDALGO2's data management requirements, and which complement and work well together with each other, as well as with components external to the immediate Data Storage and Transfer Components.

2.3.1 Selected Technologies

2.3.1.1 CKAN

CKAN[2] (the Comprehensive Knowledge Archive Network) is a free solution to manage and publish data collections. Collections of user data are organized in datasets, which can contain datastores (files in various formats: CSV, XML XLS, image file, PDF document and SQL records).

Each dataset can be described with additional metadata and tags. In the CKAN platform, data can be uploaded via the GUI or REST API. CKAN supports many extensions, for example Datapusher (for converting data from CSV, XLS files to SQL), authentication plugins (OIDC, Oauth2) and visualization (Interactive data visualizer). There is an official directory of extensions for CKAN and also a Python library that allows you to integrate your own applications with the CKAN instance.

[2] https://ckan.org/

Layers included in the CKAN system are presented on below Figure 1.

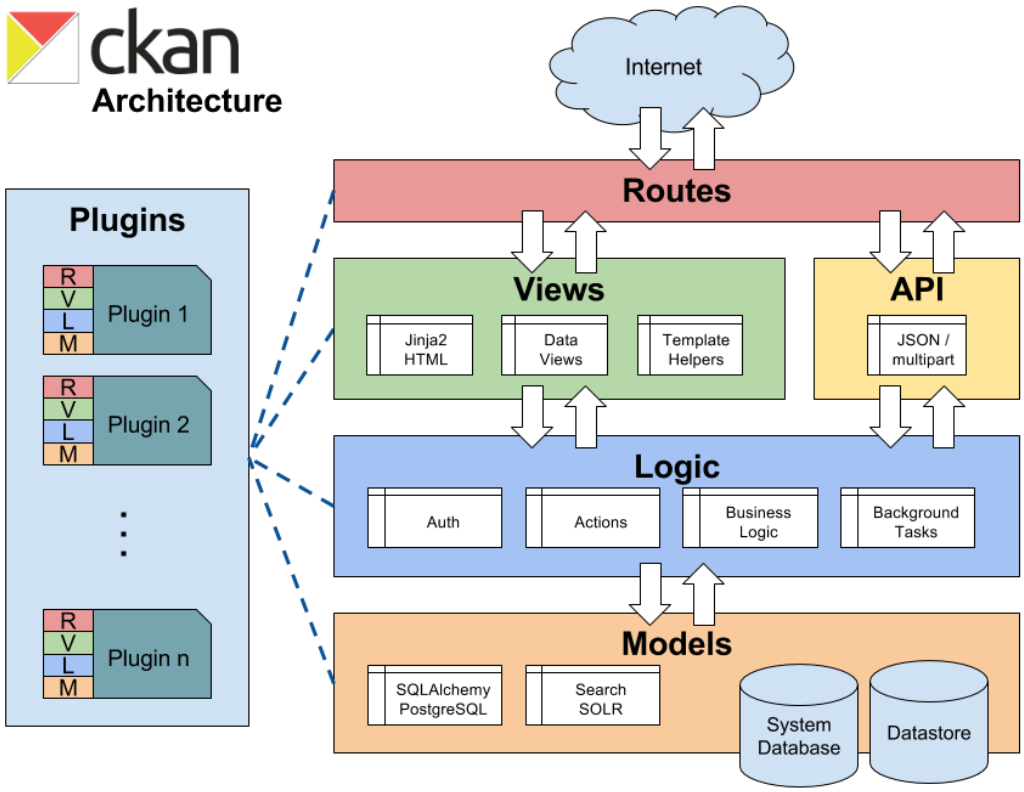

Figure 1. Architecture of CKAN[3]

In HiDALGO2 CKAN instance users are assigned to their home organizations as well as to the project organization (see Figure 2). User data (datasets) can be visible as public, private within the organization or private within the HiDALGO2 project.

CKAN is integrated with the HiDALGO2 SSO IDM (Keycloak) and it's easy to use. Unfortunately, CKAN is not scalable solution. By default, CKAN supports files up to 10MB. We made some modifications to speed up and support files up to 10GB. In the collaboration with FTP/SCP server we can share files larger than 10GB.

[3] Picture taken from https://ckan.org

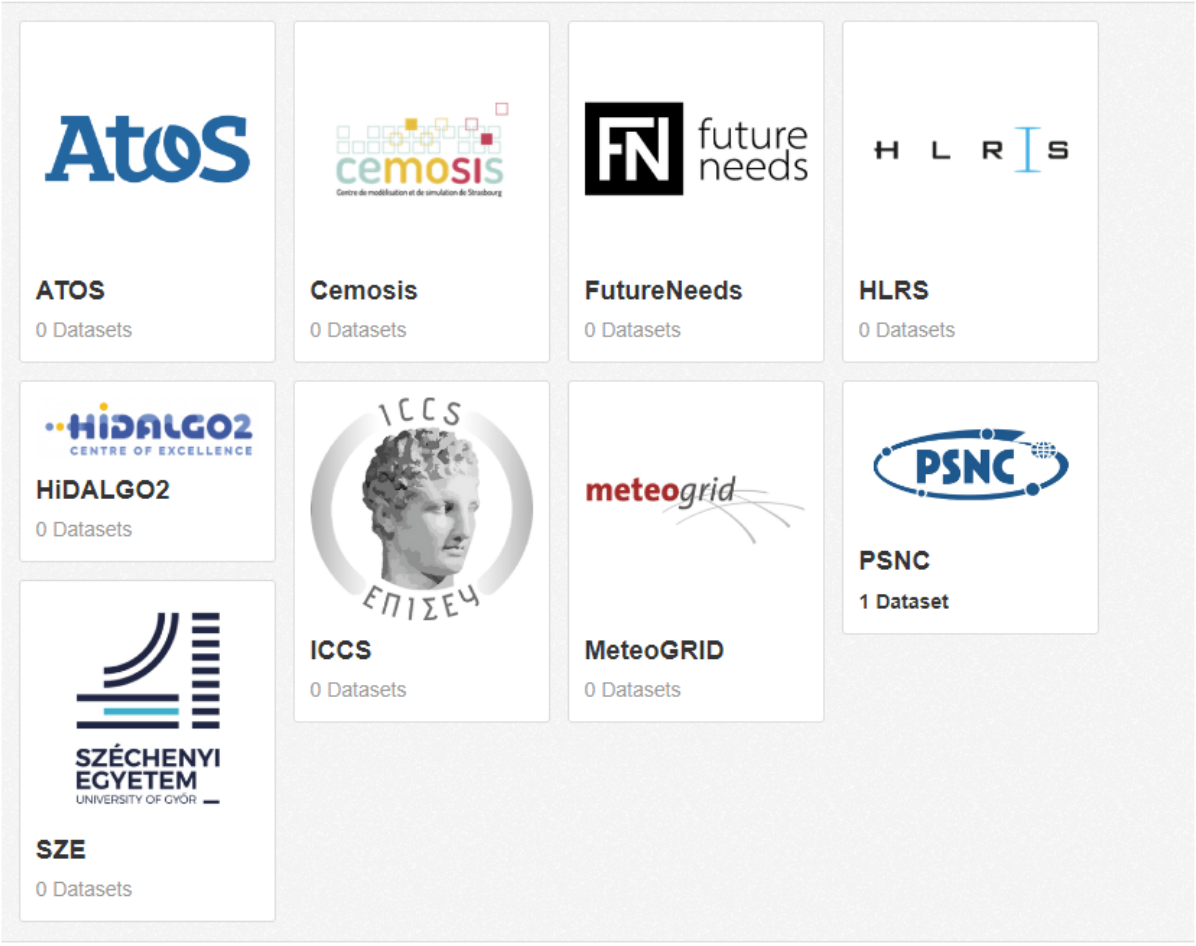

Figure 2. HiDALGO2 organizations in CKAN[4]

2.3.1.2 Hadoop Ecosystem

Requirements expressed by pilots for the HiDALGO2 DMS (see Section 2.1) suggest the need to offer a centralised data storage to host large-volume datasets, collected from multiple data providers, and showing different internal formats, including raw data, semi-structured data, relational data, and possibly other types of data (e.g., images). Besides, the increasing size of data to be hosted by the HiDALGO2 DMS storage suggests the need to select a highly scalable, highly distributed data storage that also offers high-throughput data transfer from the data sources and from/to HPC clusters. This kind of data storage solutions are described under the concept of Data Lakes, defined by Google[5] as "*a centralised repository designed to store, process, and secure large amounts of structured, semi-structured, and unstructured data*".

[4] CKAN organizations in HiDALGO2 - https://ckan.hidalgo2.eu/organization/

[5] https://cloud.google.com/learn/what-is-a-data-lake

One of the most popular OSS Data Lake ecosystems in the Big Data domain is Hadoop, a Big Data storage and processing ecosystem provided by Apache and other third-party vendors, offering i) a high-distributed, high-performance file system, HDFS, ii) YARN, to manage distributed resources and schedule user's applications, iii) a MapReduce implementation for large-scale data processing, and more.

To realise the HiDALGO2 DMS data lake, we have considered adopting Hadoop HDFS as one main possible solution for the data storage system (the other one is CKAN, see Section 2.4 describing the HiDALGO2 DMS architecture for further justification), together with other components of the Hadoop ecosystem we will describe in the following subsections. Other alternatives for HDFS as the main data lake have been considered, mostly some distributed object/blog storage, such as Ceph[6] or MinIO[7], which are also described in following subsections. The main reasons for adopting HDFS as main DMS storage are: i) HDFS is quite popular distributed data storage among the OSS communities, with large community support and integration with other tools, ii) Several ETL/ELT OSS solutions considered for data transfer, such as Apache NIFI, Airflow or Flume (see descriptions below) offer native support for HDFS transfer, iii) HDFS distributed deployment, configuration, and scalability is easily managed by the Ambari tool, iv) Scaling up/down storage is quite straightforward by adding/removing nodes, v) HDFS is fully compatible with HPDA and AI technologies considered by T4.2 HPDA and T4.3, such as Apache Spark[8] or Flink[9]. On the downside, HDFS security is based on Kerberos, whose compatibility with the Hidalgo SSO, based on Keycloak, needs to be further investigation.

2.3.1.2.1 Hadoop HDFS

HDFS is a highly distributed, fault-tolerant, high-performance, portable file system, organised into multiple data nodes where datasets are stored, with the data split across fix-size blocks. A master node or name node is the entry point to the HDFS cluster, as it contains the metadata that describes all the datasets stored within, their location (across the data nodes) and the location of the redundancy copies. A secondary name node (also known as the checkpoint node, manages the metadata checkpoints of the file system. HDFS follows a data-locality principle, by clustering the data splits across nearby local nodes and by permitting data processing to be executed in the same nodes where data resides.

[6] https://ceph.io/en/
[7] https://min.io/
[8] https://spark.apache.org/
[9] https://flink.apache.org/

2.3.1.2.3 Ambari

Apache Ambari[10] is a Hadoop management tool. It can be used i) to deploy HDFS into a cluster, ii) to configure its name, secondary name and data nodes, adding/removing nodes, iii) to automate the replication of the cluster deployment, iv) to automate the configuration of the cluster security, v) to supervise (through monitoring) and maintain the HDFS cluster.
Ambari will be the main HiDALGO2 DMS storage delivery and maintenance tool.

2.3.1.2.4 HUE

HUE[11] is a popular GUI for databases and data warehouses, which can be used to browse HDFS datasets, but also to query HDFS data by issuing HIVE queries and browsing the results. When interfacing a HDFS with YARN installed, HUE can be used to monitor the state of submitted jobs. HUE can be used to upload local files into the target HDFS, organise the HDFS content into folders, manage the user's browsing history, manage file access permissions, and more. HUE will be delivered as the main GUI interface to the HiDALGO2 DMS storage based HDFS, combined with Ambari for HDFS administration.

2.3.1.2.5 Atlas

Apache Atlas[12] offers metadata management and governance, enabling data owners and organisations to define and manage their catalogue of data assets. Atlas includes predefined schemas and types to define metadata but also supports the definition of custom types. Atlas also permits the definition of classifications, a way for tagging datasets. Classifications enable Atlas to browse through catalogues, and can also be used to enforce security restrictions. Atlas shows visual representations of dataset linages going through transformation pipelines. Atlas will be used in Hidalgo DMS as the catalogue system for HDFS.

2.3.1.2.6 Kerberos and Apache Ranger

Kerberos[13] is a network protocol that provides strong users' authentication in a distributed computing environment like Hadoop HDFS. Kerberos is primarily used in HDFS to authenticate and authorise users to access HDFS.

Apache Ranger: Apache Ranger[14] is a Hadoop's centralised security and access control framework. It offers tools for defining, managing and enforcing security access policies, auditing, and authorising Hadoop components like HDFS, Hive, and others.

[10] https://ambari.apache.org/
[11] https://github.com/cloudera/hue
[12] https://atlas.apache.org/
[13] https://web.mit.edu/kerberos/
[14] https://ranger.apache.org/

To secure Hadoop HDFS effectively, both Kerberos and Apache Ranger should work together:

- **Kerberos for Authentication**: Kerberos is primarily responsible for authenticating users to get access to Hadoop clusters. It doesn't provide fine-grained access control or authorisation rules, though.
- **Apache Ranger for Authorization and Access Control**: Apache Ranger complements Kerberos by providing fine-grained authorisation and access control policies that specify what users can access HDFS data, what data and what actions they can perform on it. Ranger enforces these policies at HDFS and other Hadoop component levels.

The adoption of Kerberos and Ranger for HDFS authentication and authorisation needs to be integrated with the main HiDALGO2 AAI based on Keycloak. Further investigation is required.

2.3.1.2.7 Hive

Apache Hive[15] is a distributed, fault-tolerant, highly scalable and good-performing data analytics platform. Hive provides a tabular interface to HDFS datasets, similar to RDBMS, that permits to group, consult and analyse datasets. Queries are propagated in batch through the distributed HDFS data storage as jobs managed by YARN that leverage the MapReduce strategy for high-performance data analytics. Hive can be installed on top of the HDFS when users require fine-grained access to tabular data stored in HDFS, or for instance, to perform HPDA.

2.3.2.2 NIFI

Apache NIFI is a tool used to automate the transfer of data between data sources and systems. It is a distributed system that has the capability to extract, transform and load data (ETL tool[16].) The tool is used with a browser-based interface where data flows can be designed visually using a library of more than 300 connectors that perform a wide variety of operations on the data in an easy and intuitive way with no need of software programming knowledge, and it also allows for a complete tracking of all the steps involved in a data stream.

NIFI is easily scaled and can be used within a cluster, it can use TLS encryption to ensure secure communications, and it can be extended by users adding new connectors.

[15] https://hive.apache.org/

[16] https://en.wikipedia.org/wiki/Extract,_transform,_load

2.3.2 Alternative Technologies

2.3.2.1 Alternatives for Object Data Storage

2.3.2.1.1 MinIO

MinIO[17] is a fully compatible with Amazon S3 (Simple Storage Service) economical object storage. It can be used for data archiving, big data analysis, or for backup and disaster recovery. MinIO implements a micro-storage architecture. It allows you to create a cluster to ensure scalability and high availability. MinIO is best suited for storing unstructured data such as photos, videos, log files, etc.

2.3.2.1.2 CEPH

CEPH[18] is reliable, easy to manage, and free software-defined storage. This solution is based on a cluster consisting of Ceph monitors and Ceph OSD daemons. This makes Ceph a highly available and scalable solution (up to many Petabytes). It provides object storage, block storage and file storage. Ceph allows clients to interact directly with Ceph OSD daemons. Ceph OSD daemons replicate objects on other Ceph nodes to ensure information security and high availability.

2.3.2.2 Alternatives for Metadata storage

2.3.2.2.1 THREDDS

The THREDDS[19] (Thematic Realtime Environmental Distributed Data Service) Data server is a tool that aims to provide data providers to create a catalogue of scientific data with useful metadata and easy access to the data. The catalogue is dynamically generated in XML format and it can be created reading data in multiple formats (NetCDF, HDF5, Grib, etc.) and provides access to the data collection via HTTP or WMS protocols.

2.3.2.3 Alternative LakeHouse solutions

Lake House[20] combines the benefits of data lakes to store heterogeneous data sources in their original format into a centralised storage with data warehouse features such as ACID transactions (for concurrent IO), time travel versioning, schema enforcement and governance for data integrity, and others. On demand, a Lake House platform can be delivered on top of the HDFS for the HiDALGO2 DMS, particularly to support concurrent HPDA on results generated by the pilots' simulations. Several OSS Lake House are available (see following paragraphs) and will be eventually evaluated in order to complement the DMS storage with Lake House benefits.

[17] https://min.io/
[18] https://ceph.io/en/
[19] https://www.unidata.ucar.edu/software/tds/
[20] https://www.databricks.com/blog/2020/01/30/what-is-a-data-lakehouse.html

2.3.2.3.1 Hudi

Apache Hudi[21] is an OSS Lake House solution that offers ACID transactions, with support for incremental streaming workloads, time travel version browsing and rollback, integration with multiple data sources and query engines, tabular high-performance analytics, and more.

2.3.2.3.2 DremIO

DremIO[22] is another Lake House solution specialised on self-service data analytics without the need of ETL and compatible with any BI tools. DremIO can be connected to any Data Lake provider to permit interactive and efficient data analytics. Dremio offers features like federated query processing, data caching, data reflections, fine-grained access control, and a user-friendly interface for data exploration. DremIO offers an OSS community edition.

2.3.2.3.5 DeltaLake

Delta Lake[23] is another popular OSS Lake House solution offering common Lake House features, such as ACID transactions, time travel versioning management, schema evolution and enforcement, DML operations and more. Delta Lake is also compatible with popular HPDA tools such as Flink or Spark.

2.3.2.4 Alternatives for Data Transfer

2.3.2.4.1 Airflow

Apache Airflow[24] is a workflow manager used as a service orchestrator. It is used to automatize jobs using Python that are then managed and executed by Airflow. It is commonly used to handle data ingestion, periodical administration tasks or events triggered by external conditions. Airflow has a web user interface to monitor and manage the execution of all the jobs and tasks configured.

2.3.2.4.2 Flume

Apache Flume[25] is a distributed service part of the Hadoop ecosystem, that is used to collect, aggregate and move large quantities of data from different sources to data storage systems. It enables the flow of data and logs into a Hadoop environment.

[21] https://hudi.apache.org/
[22] https://www.dremio.com/
[23] https://delta.io/
[24] https://airflow.apache.org/
[25] https://flume.apache.org/

2.4 DMS Architecture

The technical architecture of the HiDALGO2 data management system (DMS) has been designed aiming to address the functional specification of the DMS described in the previous section. This architecture, depicted in Figure 3, consists of two main DMS components, namely:

- A single-entry point, distributed data-lake-like **data storage** component, which stores multiple kinds of (semi) structured and raw datasets, complemented by number of associated services for data browsing, searching and cataloguing, etc., and
- A **data transfer** component, which supports the high-performance transfer of any-size data-sets, from/to providers (e.g., Cloud, HPC) to/from the above data storage.

The *data storage* component can leverage multiple baseline technologies. Continuing the HiDALGO1 approach, it can be based on CKAN[26] tool, an open source DMS, or, by adopting a big-data approach, it can be based on an open-source distributed storage, such as Apache Hadoop HDFS[27].

Both CKAN and HDFS shows downsides to their adoption in HiDALGO2. In particular, CKAN faces the following challenges:

- Storage scale-up/down on demand requires more complex configuration and maintenance, which requires further investigation,
- Data transfer through the exposed CKAN single-entry-point REST API shows poor performance for larger files,
- CKAN requires the development of a custom integration with HPDA and AI tools,
- Data transfer with CKAN requires the development of custom processor for Apache NIFI (or custom source/sink for Flume).

[26] https://ckan.org/

[27] https://hadoop.apache.org/

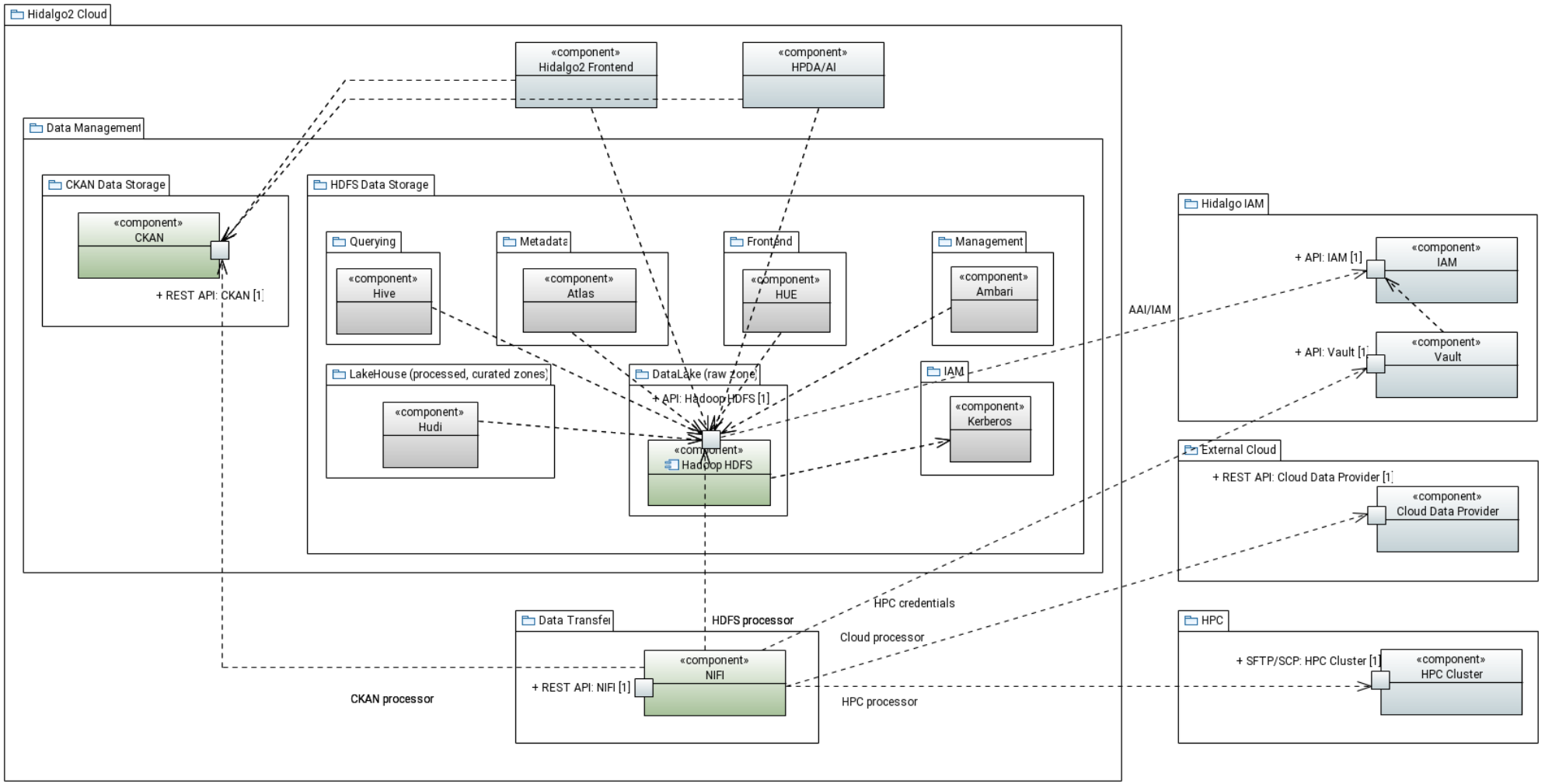

Figure 3. HiDALGO2 Data Management Architecture

Similarly, HDFS faces the following challenges:

- HDFS storage do not suit small-medium datasets well, being optimal for medium-size to large ones[28],
- The HDFS deployment, setup and maintenance shows additional complexity[29],
- HDFS IAM is based on Kerberos, whose deployment, setup and maintenance shows high complexity, and whose integration with Keycloak is still an open issue that requires further investigation (see [5]),
- HDFS requires the integration of third-party services for Web frontend (e.g., HUE[30]) and metadata support (e.g., Apache Atlas[31])

By continuing the HiDALGO1 approach, CKAN will be adopted as main storage for immutable datasets, and those to be shared with external data consumers. Acknowledging the limitations of CKAN, the HDFS storage will be adopted for datasets requiring HPDA and AI data pre-processing. How both datasets coexist and are synchronized on demand, will be further investigated.

CKAN data storage will be extended by installing available extensions according to the pilot's demands, along the next releases of the HiDALGO2 infrastructure.

HDFS data storage will be complemented with the following additional services:

- **HDFS Management, Apache Ambari**[32]: will be used to install and maintain the HDFS cluster. The cluster will consist of one name-node, one secondary name-node, and a flexible number of data-nodes,
- **HDFS Frontend, HUE**: will be used a main Web frontend to the HDFS cluster,
- **HDFS metadata catalogues, Apache Atlas**: will be used to manage the metadata associated to datasets and the creation of catalogues for the HDFS storage,
- Optionally, the HUE SQL-like search could be completed with other **HDFS searching** solutions, including **Apache Hive**[33]. Other alternatives such as **Apache Impala**[34] will be explored as well,
- Optionally, the HDFS data-lake storage could be extended with lake-house ACID transactions and time travel dataset versioning and traceability support, if demanded by HPDA (T4.2) and AI (T4.3), and pilots (WP5). Open-source **lake-**

[28] https://blog.cloudera.com/the-small-files-problem/
[29] This challenge can be overcome with the adoption of Ambari
[30] https://gethue.com/
[31] https://atlas.apache.org
[32] https://ambari.apache.org/
[33] https://hive.apache.org/
[34] https://impala.apache.org/

house solutions like **Apache Hudi**[35] will be considered, as well as **DeltaLake**[36] or **Dremio**[37],
- **HDFS IAM, Kerberos**[38]: will be used to restrict the access to datasets and folders located in the HDFS storage to granted users

The *data transfer* component provides an ELT solution for HiDALGO2, enabling the high-efficient transfer of medium to large volumes of data from data sources to targets, including, cloud data providers, HPC clusters and the HiDALGO2 data storage. In current architecture, the **Apache NIFI**[39] solution is adopted. If required, other ELT solutions for data transfer pipeline, such as **Apache Airflow**[40] and **Apache Flume**[41] will be evaluated, even considering the possibility to adopt a hybrid data transfer approach, combining NIFI with Airflow, for instance.

As depicted in the architecture in Figure 3, NIFI will rely on different processors to transfer data, namely:

- A CKAN processor to transfer data from/to CKAN through its REST API. This processor requires to be implemented and registered in NIFI as a custom processor,
- A HPC processor to transfer data from/to a HPC cluster through its SSH connectivity. The NIFI SFTP processor or a custom processor to implement SCP or another SSH based transfer technology (e.g., RSYNC) will be evaluated and eventually implemented,
- A Cloud processor to transfer data from Cloud data providers to central storage, NIFI provides processors to get data from Cloud using REST API and HTTP,
- A HDFS processor to transfer data from/to the HDFS storage. NIFI provides processors to get/put data into HDFS storage.

The transfer of data from/to HPC requires user's accounts to transfer data through a SSH connection. Therefore, the user's credentials, stored into the HiDALGO2 Vault, needs to be passed to the *data transfer* component. This dependency, explicit in the architecture (Figure 3), also requires of user's authentication in the HiDALGO2 Keycloak, so that the user's secret account can be retrieved from Vault.

[35] https://hudi.apache.org/
[36] https://delta.io/
[37] https://www.dremio.com/
[38] https://web.mit.edu/kerberos/
[39] https://nifi.apache.org/
[40] https://airflow.apache.org/
[41] https://flume.apache.org/

HiDALGO2 services such as the front-end or the HPDA and AI services, will make use of the data management services, namely the *data storage* and *data transfer* services, through the APIs and CLIs offered.

In the next section, a description of the initial deployment of this architecture and a roadmap to release a more complete versions of the architecture will be defined.

2.5 Solution Deployment

A first implementation of the HiDALGO2 DMS has been deployed into the HiDALGO2 development Cloud (see D2.4) consisting of the main data storage (CKAN and HDFS) and a data transfer solution based on Apache NIFI. The following gives details of the deployment of these tools and the first data transfer implementation for HDFS to/from HPC.

2.5.1 CKAN

The CKAN is installed in the PSNC OpenStack infrastructure. The virtual machine with 8 vCPU and 16GB RAM is running on Ubuntu Server 22.04 LTS Linux. The CKAN web interface is accessible via HTTPS protocol. Additional storage for database and data is located on the CEPH infrastructure. The volume for the storage is 2TB with opportunity to extend the size.

PSNC has implemented a new version of the OAUTH2 plugin allowing integration with HiDALGO2 IDM service. Configuration in the ckan.ini file related with the plugin:

```
ckan.oauth2.authorization_endpoint = https://idm.hidalgo2.eu/realms/hidalgo2/protocol/openid-connect/auth
ckan.oauth2.token_endpoint = https://idm.hidalgo2.eu/realms/hidalgo2/protocol/openid-connect/token
ckan.oauth2.profile_api_url = https://idm.hidalgo2.eu/realms/hidalgo2/protocol/openid-connect/userinfo
ckan.oauth2.client_id = ****
ckan.oauth2.client_secret = *******
ckan.oauth2.scope = email openid profile
ckan.oauth2.profile_api_user_field = preferred_username
```

The default size for an uploaded single file has been increased from 10 MB to 10 GB. Additionally, caching of uploaded data has been moved to RAM memory. Configuration changes in the ckan.ini and nginx site:

```
ckan.max_resource_size = 10000
client_max_body_size 10G
proxy_temp_path /dev/shm
client_body_temp_path /dev/shm
```

2.5.2 Ambari and HDFS

A HDFS cluster has been commissioned in in the PSNC OpenStack infrastructure, consisting of five virtual machines with Ubuntu Server 16.04 LTS Linux. The resources assigned to the individual nodes are as follows:

- hidalgo-ambari - 8 CPUs, 16 GB RAM, 40GB + 500GB storage
- hidalgo2-nn-1 (primary name node) - 8 CPU, 16 GB RAM, 40GB + 500GB storage
- hidalgo2-nn-2 (secondary name node) - 8 CPU, 16 GB RAM, 40GB + 500GB storage
- 2 Data Nodes (hidalgo2-dn-1, hidalgo2-dn-2) - 8CPU, 16GB RAM, 10GB + 10TB storage

Apache Ambari server was installed in the hidalgo-ambari node, by building it from sources. Ambari agent was also installed and configured in both the name and data nodes. However, Apache HDFS could not be installed with Ambari, since it required a version definition file (VDF) that describes the HDFS software stacks available for installation. Until 2021 a software stack provided by Cloudera for Hortonworks HDP was freely available with Ambari, but presently it requires a paid subscription. Packaging and defining a new Hadoop VDF requires significant effort which is beyond the purpose of simplifying the installation of the Hadoop HDFS cluster with Ambari. Eventually, further investigation on the usage of Apache Bigtop[42] for defining the Hadoop VDF[43] can be taken in the future.

Therefore, the Hadoop HDFS cluster has been manually installed from binaries into the above described HDFS nodes.

The HDFS main entry point is the master name-node, *hidalgo2-nn-1*, with public URL *sophora-42.man.poznan.pl*.

The secondary name-node, *hidalgo2-nn-2*, has public URL *sophora-103.man.poznan.pl*.

The two data-nodes, *hidalgo2-dn-1* and *hidalgo2-dn-2*, are not exposed in public URLs.

As this is a development HDFS cluster, it has been configured with file replication equal to 1.

The HDFS cluster needs further configuration and extensions, including user authentication and authorization with Kerberos and Apache Ranger, the installation of

[42] https://bigtop.apache.org/

[43] https://cwiki.apache.org/confluence/display/AMBARI/How-To+Define+Stacks+and+Services

a GUI based on HUE and a metadata/catalogue GUI based on Apache Atlas. Furthermore, the integration between Kerberos and the main HiDALGO2 AAI/IAM based on Keycloak also needs further investigation.

2.5.3 NIFI

The installation of NIFI in this stage of the project is being considered as a proof of concept and a practical test to assess the better configuration and approach in the long term. For this installation it has been decided to use a Virtual Machine (VM) located at the Poznan infrastructure[44]. This VM is running an Ubuntu 22.04 Linux distribution. This version has been selected due to compatibility, ease of use, and long-term support (meaning that it will receive updates and patches until 2027).

At this stage it has been decided to use a standard standalone installation of Apache NIFI downloading the application and Toolkit, as described in the official documentation[45], instead of a cluster deployment or a Docker installation. Both of these options will be revisited and reassessed once the service has been running for some time. The version selected is the last release available 1.23.2[46].

For the configuration and set up, the connection via HTTPS protocol has been configured in the container machine, so access to the NIFI interface running in the VM can only be done via HTTPS protocol. This has been considered as the best approach in terms of security when accessing the VM and the tool. Further measures will be taken later on regarding user management and authentication.

2.5.4 HDFS – HPC data transfer

HiDALGO2 requires a performance-efficient way to transfer medium to high volumes of data from/to the main HiDALGO2 data (i.e., either the CKAN or HDFS based) storage to/from target HPC clusters. This data could be input data required by pilot's simulations or their outcomes, which must come back to the main storage for subsequent data processing, analysis and visualization by other HiDALGO2 services, namely HPDA (T4.2), AI (T4.3) or visualization (T4.4).

According to the designed architecture, data transfer leverages the Apache NIFI as the main ELT solution. Based on NIFI, a first prototype for HDFS2HPC and HPC2HDFS data transfer, using SFTP as main transfer protocol, has been implemented. Data transfer to/from HPC clusters is limited to SSH-based communications, therefore few options are available, including SFTP, SCP, RSYNC, and remote file system mounting

[44] sophora-105.man.poznan.pl

[45] https://nifi.apache.org/docs/nifi-docs/html/walkthroughs.html

[46] https://nifi.apache.org/download.html

using SSHFS and FUSE[47]. None of these options offers high performance for a single data transfer process, so further investigation will be required to rely on the high performance of NIFI to split the data transfer into multiple concurrent transfer channels to speed up the entire process. SFTP-based transfer has been selected for this first prototype as NIFI includes a processor for SFTP transfer in its collection of available processors[48]. Processors for SCP and RSYNC are not included and would require the development of a custom processor or leveraging existing processors (e.g., ExecuteProcess, ExecuteScript, ExecuteStreamCommand) to process a SCP-based transfer with external scripts. We don't expect a significant increment in performance by adopting a SCP strategy over SFTP. Using RSYNC strategy would speed up the performance transfer, as only data updates are transferred from the source to the target, but in HiDALGO2, we don't expect pilots will update consumed or produced datasets. Therefore, rather than investigating on the low-level transfer technology, we should focus on ways to design concurrent transfer strategies, by leveraging the multiprocessing capabilities of NIFI or similar ELT solutions.

The code of this first data-transfer prototype based on SFTP is located at the HiDALGO2 Git repository[49].

The prototype offers a CLI interface, intended to be used, as a client, by users and programmatically by other HiDALGO2 services. The CLI interfaces the NIFI REST API to request the instantiation and execution of a specific process group from a registered template. That is, the CLI acts as a client, and NIFI acts as a data transfer service. Offering the CLI as REST API as well would introduce another service in the middle, not designed nor implemented to offer the same concurrency support and performance as the offered by NIFI. Therefore, this prototype plays exclusively the role of the main HiDALGO2 data transfer client.

[47] https://www.debuntu.org/how-to-mount-a-remote-filesystem-using-ssh-sshfs-and-fuse/

[48] https://nifi.apache.org/docs.html

[49] https://git.man.poznan.pl/stash/projects/HiDALGO2/repos/hid-data-management/browse/data-transfer/nifi/data-transfer-cli?at=refs%2Fheads%2Fnifi

The CLI data transfer client is implemented in Python 3[50]. Poetry[51] is used as main package management tool. The configuration of the target NIFI is done in the *cli.cfg* file located in the *data_transfer_cli/conf* module:

```
[Nifi]
nifi_endpoint=http://localhost:8443
nifi_login=nifi
nifi_passwd=nifinifinifi
nifi_server_user_name=yosu
nifi_server_private_key=~/.ssh/id_rsa
nifi_upload_folder=/opt/nifi-data/upload

[ProcessGroups]
hdfs2hpc_group_id = ffd29515-018a-1000-5d0e-15aeca8f0d14
hpc2hdfs_group_id = 1a10c94a-018b-1000-4a84-ed5e565431fc
```

The section *[NIFI]* in the configuration describes the endpoint and account of an NIFI user. This NIFI account is provided by the NIFI administrator. It also defines the user account (*nifi_server_user, nifi_server_private_key*) to temporary transfer HPC keys to the NIFI server. This user's account is provided by HiDALGO2 infrastructure provider and it is user's specific.

Within the same NIFI section, we also specify a folder (*nifi_upload_folder*) in NIFI server where to upload the keys. This folder must be accessible by the NIFI service. The section *[ProcessGroups]* provides the UUIDs for registered templates for HDFS2HPC and HPC2HDFS processes.

The prototype consists of several modules, namely:

- *conf*: manage the NIFI configuration described above,
- *nifi*: main module containing the classes and methods required to interface with the target NIFI through its remote REST API,
- *parser*: parsers the CLI invocation and processes the main command and associated input parameters. Manages the CLI help report,
- *utils*: offers helper methods required by other modules

[50] https://www.python.org/
[51] https://python-poetry.org/

The main python script, *data_transfer_cli.py* reads the target NIFI configuration, parsers the CLI invocation, and delegates on the associated command method to process the data transfer.

CLI client usage requires Python 3, a local instance of NIFI and a local instance of HDFS. Support for remote instances of NIFI and HDFS will be implemented in next release.

CLI client is invoked as follows:

```
python data_transfer_cli.py command <arguments>
```

To get help use it as follows:

```
python data_transfer_cli.py –h
Obtaning:
usage: data_transfer_cli.py [-h] {check-status,cleanup,hdfs2hpc,hpc2hdfs} ...

positional arguments:
  {check-status,cleanup,hdfs2hpc,hpc2hdfs}
                        supported commands to transfer data
    check-status        check the status of a command
    cleanup             cleanup a processing command
    hdfs2hpc            transfer data from hdfs to target hpc
    hpc2hdfs            transfer data from hpc to target hdfs

optional arguments:
  -h, --help            show this help message and exit
```

Current CLI prototype offers the following commands:

- *hdfs2hcp*: transfer data from a local HDFS location (e.g., /users/yosu/data/genome-tags.csv) to a remote HPC (e.g., LUMI, at $HOME/data folder)
- *hpc2hdf*: transfer data from an HPC location (e.g., LUMI, at $HOME/data/genome-tags.csv) to a local HDSF location (e.g. HDFS location (e.g., /users/yosu/data/target/)

- *check-status*: checks the status (and possible warnings/errors) of the issued command identified by its returned UUID
- *cleanup*: cleans up the NIFI artefacts created for the execution of the command identified by its returned UUID

To get help on a concrete command, use it as follows:

```
python data_transfer_cli.py hdfs2hpc -h
Obtaning:
usage: data_transfer_cli.py hdfs2hpc [-h] -s DATA_SOURCE [-t DATA_TARGET] -
n HPC_HOST -u HPC_USERNAME -k HPC_SECRET_KEY

optional arguments:
 -h, --help            show this help message and exit
 -s DATA_SOURCE, --data-source DATA_SOURCE
                       HDFS file path
 -t DATA_TARGET, --data-target DATA_TARGET
                       HPC folder
 -n HPC_HOST, --hpc-host HPC_HOST
                       target HPC host
 -u HPC_USERNAME, --hpc-username HPC_USERNAME
                       HPC username
 -k HPC_SECRET_KEY, --hpc-secret-key HPC_SECRET_KEY
                       HPC secret key path
```

When the CLI triggers a command execution in NIFI the following procedure is performed:

- The NIFI template associated to the command (e.g., Hdfs2hpc or hpc2hdfs) is located in the NIFI registry,
- A process group is instantiated from the template,
- A parameter context is created and populated with the input parameters provided by the CLI caller,
- The parameter context is associated to the process group,
- The sequence of processors in the group is determined,
- Following this sequence, each processor in the group is executed once, and data flows the sequence. Next processor is executed upon the termination of previous one.

Once the last processor in the sequence has been triggered, typically the one processing the real transfer (e.g., from HDSF to HPC), the CLI returns the UUID of the process to the caller, not blocking the process termination, as the transfer can take long times. The, the caller can use the check-status CLI command to pull the state. Upon process termination, the CLI caller invokes the CLI clean-up command to clean up the created NIFI artefacts, namely the parameter context and process group. This approach is asynchronous, non-blocking design, that requires the caller collaboration to check the transfer termination status and the subsequent clean up. An alternative design consists on a synchronous CLI command invocation, where the caller is blocked until the termination of the data transfer, moment when NIFI is automatically clean up by the CLI, without requiring an explicit caller collaboration. The following sequence diagram in Figure 4 describes a typical caller-CLI interaction.

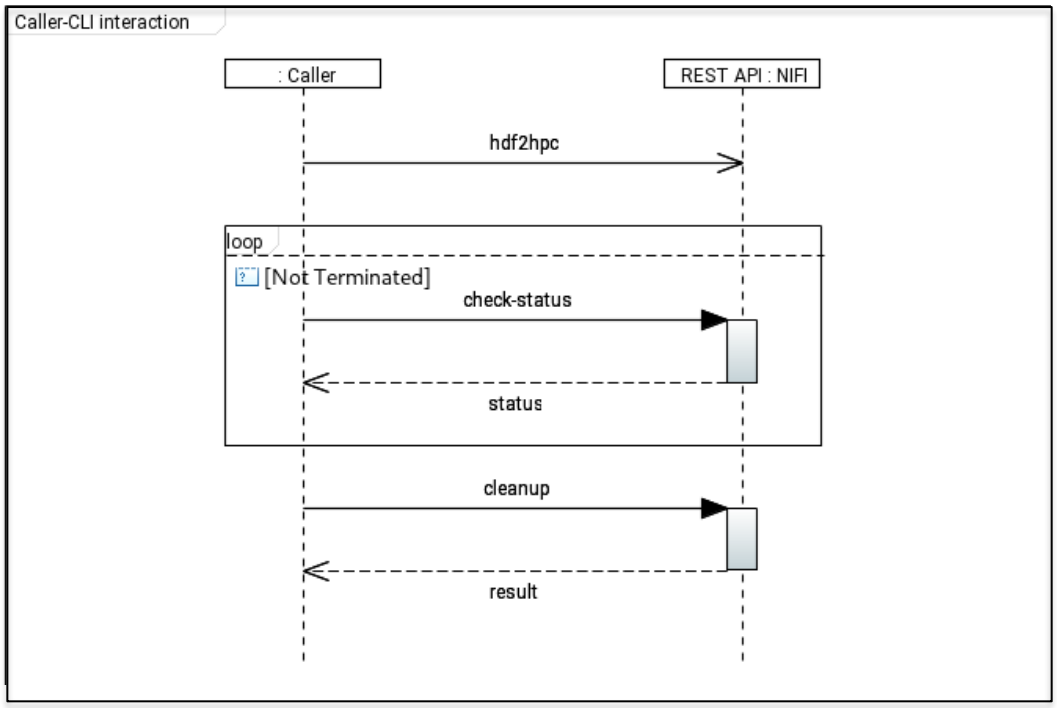

Figure 4. Caller-CLI Interaction

3 Coupling Technologies

Coupling technologies are those that facilitate the intra-application interoperability in collaborative scenarios that involve several pilot's simulations. In the following, we will justify with examples the need for these scenarios and conduct a state-of-the-art analysis of the available, suitable coupling technologies for future provision and adoption.

3.1 Pilot Coupling Scenarios

The HiDALGO2 CoE pilot applications model environment related physical processes:

- airflow and gaseous pollutants' dispersion in urban domains (UAP),
- urban buildings' thermal flow and emission of greenhouse gases (UB),
- airflow and weather simulation around renewable energy systems and their effect on energy production estimations (RES), and
- airflow simulation in rural area and smoke propagation (WILDFIRES).

The modelled physical processes interact to each other and even several of the main physical variables of the applications are the same: wind velocity, temperature, and pollutant concentrations. The main difference between the 4 applications is in their computational domains that are as follows below.

- UAP models airflow and dispersion in cities in the exterior of buildings,
- UB processes air temperature and composition inside the buildings of a city,
- RES' domains are industrial area with industrial structures,
- WILDFIRES simulates processes in forests and other rural domains.

At the boundary of its domain, each application assumes some values for their variables. Since several domains are neighbouring thus it is natural to consider one simulation to prescribe boundary conditions to the other for the common variables on the contacting surfaces or make a nesting with overlapping domains. The decision of whether it is worthwhile to perform the coupling of various applications depends on whether the resulting coupled code represents greater societal value, models a realistic situation, or increases the accuracy of the codes to be attached in themselves.

In HiDALGO2, the following couplings have been already investigated and are under development.

3.1.1 UAP and UB coupling

It was found in the literature, see e.g. [6], that taking building walls as heat sources into account for the modelling of the airflow near buildings, significantly modify the airflow. Conversely, since UAP operate at high spatial resolution at street level (1-5 meters), this allows a more accurate modelling of the air exchange in the buildings modelled by UB than the current air-exchange of the building models in the current UB. As a conclusion, UB will provide to UAP heat sources at building external walls which will result in higher accuracy for the urban airflow simulation. The coupling between UAP and UB will be a dynamic two-way coupling, i.e., data exchange from UAP to UB and conversely, from UB to UAP during the simulation time steps. For the terminology of couplings, see e.g. [7].

The UB application will interact mainly with the UAP pilot by providing data on the production of CO2 and other greenhouse gases. The models will output a temporal series of gas concentrations that will be mapped onto the building surfaces in order to ensure the coupling with the UAP solver. Temperature data can also be exchanged between the two pilots for a tighter coupling between air flow and building performance.

Since the two pilot applications build their respective solvers on the same geometrical representation of the district/city, it is natural to use it as the common base for the data exchange. The coupling will be file-based initially, we may investigate other types of coupling depending on the performance and IO demand.

Using an appropriate parallel file format (i.e., HDF5), the UB pilot will project its outputs (gas concentration and temperature, as stated before) onto the building surfaces, which will be exploited by the UAP application as boundary conditions for their model. The synchronisation between the two applications could be ensured by creating a stamp file for each application, containing its current state, and having the other application reading and waiting for a precise event. The orchestrator could help in the creation of these files and in setting up the dialogue between the two applications.

Two types of coupling scenarios are imagined: loose coupling, in which, first, the UB pilot generates the boundary conditions for the UAP application (for the whole simulation time interval), and second, the UAP application is executed; tight coupling, in which the two applications run simultaneously and the UAP application provides the UB one with wind and temperature data influencing the building energy simulation.

3.1.2 WILDFIRES and UAP coupling

The WILDFIRES case study aims to integrate simulations with the Urban Air Pollution (UAP) atmospheric pollution pilot in two different scales:

- **Landscape** scale (spanning several tens of kilometres): Simulations conducted with WRF-SFIRE-CHEM yield emissions, transport, and deposition of air pollutants, including PM2.5 particles, resulting from the development of large wildfires. These outputs will be incorporated into the UAP modelling as background pollution. To achieve this, the simulations utilize coarser spatial resolutions (on the order of tens of meters) and a broader temporal scope (on the order of minutes).
- **Local Urban** scale: The goal is to incorporate simulations of combustion results from a wildfire, performed using OpenFoam-FireFoam, affecting and progressing within an urbanized area. This includes the generation and dispersion of smoke columns containing health-toxic components. To accomplish this, a detailed scale (a few hundred meters) is employed, with a much finer spatial and temporal resolution (sub-meter spatial resolution and a temporal resolution on the order of a few seconds).

In the **landscape-scale** scenario, simulations of large forest fires will be conducted under various meteorological configurations, with a particular focus on wind speed and direction. To achieve this, the WRF-SFIRE model will be employed, capturing the dynamics of the atmosphere and the spread of forest fires. Additionally, the WRF-CHEM model will be integrated to simulate the emission, dispersion, and deposition of smoke and its components. This includes a detailed examination of compounds with potential health implications (COx, NOx, SOx) and fine particles (PM10 and PM2.5).

The outcome of this modelling effort will be volumetric distributions of smoke components at each computational point in space, especially emphasizing urban areas and cities. These concentration profiles will be stored in *netcdf* format. The integration with urban air pollution (UAP) simulations will occur through these files (loose coupling), serving as background pollution for the modelling process. Consequently, both simulations will run at different moments, though they will correspond to the same time frame.

Prior to UAP simulations, necessary adjustments will be made to spatial and temporal resolutions. Additionally, there may be a need for file format conversion into the model's input format.

In the context of **urban-scale** simulations, an exhaustive analysis encompassing airflow, vegetation and other materials combustion, heat emission, as well as smoke and its components' emission and dispersion due to the main fire front, spot fires and

local combustions will be conducted. This comprehensive study will delineate a volumetric calculation domain of 1 km³, intricately detailing the corresponding geometry of topography, buildings, and vegetation.

The ensuing models will be incorporated into OpenFoam-FireFoam, along with the relevant parameterization, and a detailed description of anticipated combustion sources will be provided. The calculation of airflow will be executed under various meteorological configurations, specifically focusing on wind direction and speed. Additionally, the simulation will encompass the combustion of vegetation, structures, and other materials at multiple instances throughout the simulation period.

For each time step, a volumetric matrix of scalar values (temperature, pressure, and smoke component concentrations) will be saved in either ASCII or binary format. Simultaneously, a wind vector field will be stored for every spatial point, also in ASCII or binary format. These simulation results will serve as the foundation for coupling with the UAP simulation, wherein they will be employed as boundary conditions.

As seen, it is also a loose coupling trough file sharing. Furthermore, the coupling between models requires a previous precise synchronization of the commonly utilized 3D models, boundary conditions, and temporal framework.

3.1.3 All pilots and WRF coupling

In addition, all of the pilot applications need weather parameters, which are currently modelled by the numerical weather forecast software WRF [8] for the RES and WILDFIRES applications while for UAP and UB the weather parameters are input parameters. In HiDALGO2, weather parameters for all 4 pilot applications will be modelled employing WRF and using the HiDALGO2 infrastructure. WRF will be run in parallel to the solvers. As usual, weather simulations predict weather parameters several hours in advance, and thus WRF-results might be stored in files. Coupling with WRF again needs a one-way coupling.

3.1.4 UAP and weather sensor data coupling

Whenever sensor data are available for the wind and other weather data, UAP uses different these data for setting up boundary conditions. When running the digital twin model of the urban airflow and pollutant concentration, the wind speed and wind direction from sensor data are assimilated to the UAP simulation. The dataflow for this data assimilation procedure can be considered as a dynamic one-way coupling of the sensor data stream into the CFD-solver.

3.2 SoTA on Coupling Technologies

Coupling technologies are used to combine different (existing) applications to work together for a global purpose, incorporating new physics into the models and adapting the physical scale (spatial and time resolution) each application works with. A wide variety of generic technologies that attempt to solve a part or all of this problem have appeared in recent years, each bringing its own functionality and added value. As a consequence, coupling technologies have become a basic element in the resolution of multiscale and hybrid simulations. HiDALGO2 foresees the coupling between simulation models used in several of its Pilots, so it is advisable to review the available technologies in order to subsequently propose a solution in accordance with the proposed requirements.

3.2.1 Data exchange technologies

HiDALGO2 pilots run simulations in target HPC clusters, consuming input and producing output datasets. When pilot simulations are coupled, for instance in a sequential way, the dataset output of one simulation could be eventually consumed as the input datasets of another. In a broader sense, we can assume that datasets produced and consumed by coupled pilots' simulations are not located at the same location and the same time, therefore not being available before. Therefore, data sharing and exchange technologies are required by pilots to provision required data before simulations take place.
Different technologies have been provided in the state of the practice for data sharing and exchange, including data transfers technologies, cloud storage services, data broadcasting and streaming, data sharing and cataloguing, and more.
In the context of **data transfer technologies**, we encounter:

File transfer protocols, including FTP, and SFTP, its secure variant over SSH. They are protocols to transfer files over a TCP network. Another secure protocol to transfer data, files and commands over SSH is SCP. Additionally, remote file system mount over SSH is available with SSHFS[52] and FUSE, and can be used to securely transfer datasets to remote file systems.

- Open Web standards for data transfer over HTTP/S are also used to transfer datasets over the Web. Based on HTTP/S, RESTful APIs can be defined, with GET, POST, PUT, DELETE actions to provide CRUD support for management of Web resources (e.g., datasets), identified by URI/URLs.

[52] https://github.com/libfuse/sshfs

- In the HPC domain, GridFTP[53] is a high-performance data transfer technology, conceived as an FTP extension for grid, with support for authentication and data encryption, that achieves higher transfer rates by multiplexing the number of TCP connections. RDMA[54] is another high-throughput and low-latency data transfer technology that uses the shared memory among nodes in HPC clusters.
- There are other general purpose ETL/ELT technologies that can be used for data transfer, including Apache NIFI, Airflow or Flume, to cite a few, which offers high-throughput multiplexed data transfer, with multiple off-the-shelf NIFI processors, Airflow operators or Flume sources/sinks for popular data storage systems.

Cloud storage/sharing services can also be used to exchange or share datasets. Popular commercial Cloud storage services are Amazon S3, Cloud Storage, Azure Blob Storage and many others. Among those open-source Cloud storage solutions (see section 2.3 for a revision of some cited in the following), we can find NextCloud[55], Owncloud[56], MinIO, Ceph , CKAN , and also many others, all offering dataset sharing and synchronisation. There are also Cloud storage services specialised on sharing geospatial and environmental datasets, such as THREDDS, GeoNetworks[57] or DataONE[58], to cite a few, or applied to research dataset sharing, such as Dataverse[59], iRODs[60], Zenodo[61], Figshare[62], also to cite a few. Moreover, Version Control Systems (VCS) such as Git (e.g., GitHub[63], Bitbucket[64], etc.) can also be used to store and exchange datasets.

Some of the above data sharing services offer **data cataloguing** features (e.g., CKAN, THREDDS and others). There are, however, solutions specifically dedicated to metadata management and cataloguing, on top of existing distributed data storage platforms (e.g., Apache Hadoop HDFS) such as Apache Atlas, or DremIO, a

[53] https://github.com/gridcf/gct
[54] https://en.wikipedia.org/wiki/Remote_direct_memory_access
[55] https://nextcloud.com/
[56] https://owncloud.com/
[57] https://www.geonetwork-opensource.org/
[58] https://www.dataone.org/
[59] https://powerplatform.microsoft.com/en-us/dataverse/
[60] https://irods.org/
[61] https://zenodo.org/
[62] https://figshare.com/
[63] https://github.com/
[64] https://bitbucket.org/

LakeHouse solution that also offers metadata and catalogue support, as explained in section 2.3.

Data broadcasting and streaming platforms can also be used to **real-time exchange and transfer** datasets among peers. Among available platforms, we can cite Apache Kafka[65], which, thanks to its PubSub mechanism, can connect data providers (e.g., event publishers) to data consumers (e.g., event subscribers), or Apache Pulsar[66], another data messaging and streaming platform.

3.2.2 Data transformation technologies

Before datasets are exposed for public usage, they may eventually need to be transformed by applying certain techniques (to cite a few):

- Data mapping: change the data schema (for structured data),
- Data wrangling[67]: cleaning, enriching and validating the data,
- Data structuring: apply a schema (for unstructured/raw data),
- Data cleaning: fill data gaps, removing outliers, etc.,
- Data enriching: combine data features, data normalisation, etc.,
- Data slicing: create data views by selecting specific records (e.g., rows) and/or features (e.g., columns), etc.

There are technologies (tools and libraries) supporting data mapping or schema transformation, including ETL/ELT tools aforementioned such as Apache NIFI and Airflow, but also Talend[68], and other HPDA frameworks, such as Apache Spark or Flink. Both ETL/ELT and HDPA tools can be also used for data wrangling and slicing. They can be combined with other popular ETL and data processing libraries such as Bonobo[69], Pandas[70], Numpy[71], Polars[72], and others, combined with high scalable, parallel wrappers around some, such as Dask[73] or Modin[74] (for Pandas).

[65] https://kafka.apache.org/
[66] https://pulsar.apache.org/
[67] https://expressanalytics.com/blog/what-is-data-wrangling-what-are-the-steps-in-data-wrangling/
[68] https://www.talend.com/products/integrate-data/
[69] https://www.bonobo-project.org/
[70] https://pandas.pydata.org/
[71] https://numpy.org/
[72] https://www.pola.rs/
[73] https://www.dask.org/
[74] https://pypi.org/project/modin/

3.2.3 Specific simulation coupling tools

MUSCLE3

MUSCLE3[75] (Multiscale Coupling Library and Environment - version 3) is a software for creation and execution of tightly-coupled multiscale models, i.e., the composite models built from many single-scale models that interchange messages during computations [9].

The methodological heritage of the tool is quite significant. Its beginning can be linked with the publication of Multiscale Modelling Language (MML) in early 2010 [10]. The idea of UML-like description of computations was appealing, but it needed to be confirmed in practice. Unfortunately, early incarnations of MUSCLE (1 and 2) were too complex for its broader usage, and required to be replaced with the software developed from scratch in modern way and with more flexible technologies, The result is MUSCLE3, a relatively easy to use software package which allows to define models with easy yMMSL format and run compound simulations regardless of environment and their scale. That being said, MUSCLE3 supports execution of an entire simulation on a laptop, as well as it is capable to run it in a distributed way on an HPC system. It takes care not only of the starting, monitoring, and management of simulation, but it also provides check-pointing and Uncertainty Quantification features.

FabSim3

FabSim3[76] is a python-based automation toolkit, built on top of the Python Fabric library, whose purpose is the automation of scientific simulations and data processing workflows [11]. The FabSim3 functionality addresses common requirements of scientists interested in performing advanced computations on supercomputers. FabSim3 brings a set of command-line tools (based on Fabric[77]) and conventions that up to the needs of certain scenario can be leveraged and easily extended with custom plug-ins to facilitate execution of remote computations of complex structure, such as execution of workflows, ensemble scenarios or uncertainty quantification. Each step in the workflow is associated with a plugin where functions responsible for job preparation, submission, execution, monitor and data retrieval are contained[78] [79]. These functions can be called from the command line, or in a shell script, by using the "fabsim" command and specifying the machine where the plugin will be launched. A number of FabSim plugins are already available, and some of them use the EasyVVUQ

[75] https://muscle3.readthedocs.io/en/latest/
[76] https://fabsim3.readthedocs.io/en/latest/
[77] http://www.fabfile.org
[78] https://github.com/djgroen/FabDummy
[79] https://github.com/djgroen/FabSim3

Python library to perform sensitivity analysis[80]. Two patterns (Uncertainty Quantification, Verification & Validation) are discussed in the documentation as examples of multiscale applications that can compose and couple single-scale models. Hence, it may be possible to use FabSim3 to couple models loosely by running them one at a time and devising a pattern for each specific coupling.

QCG-PilotJob

QCG-PilotJob[81] is a lightweight python-based implementation of the Pilot Job paradigm [12]. The goal of the tool is to allow running multiple logical jobs inside an already running parent job in a scheduling system and in this way relieve the scheduler from maintaining some of jobs. This capability is particularly useful when a number of jobs or frequency of their submission exceed the limits imposed by the computing resource or its configuration. QCG-PilotJob offers two ways of submission of tasks, namely batch mode with a static configuration file, and dynamic mode through API. From the logical perspective, QCG-PilotJob tasks reassemble regular tasks (e.g., specifically crafted execution environment, resource binding), but through a tool-specific mechanisms like tasks dependencies or iterative tasks, they can be organised in higher-level structures such as DAG workflows [13].

3.2.4 Amuse/Omuse

Amuse framework [14], Astrophysical Multipurpose Software Environment, is an open-source framework that allow an easy way to couple different numerical codes and offers a set of services to run and interchange information among them. It has been developed at Leiden Observatory [15].

Amuse framework is based on a layered architecture with three layers: a user script layer that contains a set of Python scripts to resolve a specific problem; the Amuse Library layer that provides services to interchange information among numerical codes, to provide an object-oriented interface and to offer some common functionalities as unit handling; the last one, is the community codes layer that contains the numerical codes and the interfaces that handle the messages to use them.

A more detailed documentation of the Amuse architecture can be found in the web[82].

The main page to access to the Amuse code, documentation and news is available online[83].

[80] https://fabsim3.readthedocs.io/en/latest/plugins/
[81] https://qcg-pilotjob.readthedocs.io
[82] https://amuse.readthedocs.io/en/latest/design/architecture.html
[83] https://www.amusecode.org/

Using this technology, the Institute for Marine and Atmospheric research Utrecht (IMAU) developed Omuse, Oceanographic Multipurpose Software Environment. Numerical Codes integrated in Amuse are thought to resolve astrophysical problems while those in Omuse are related with Oceanographic ones. HyMUSE, right now under development, is focused to solve hydrological and meteorological based in the Amuse technology.

At this time there is no integration for WRF or OpenFOAM as community codes in HyMUSE, although it is possible that they will be incorporated in the future. Its main strength is that it offers a framework that allows combining different user codes / simulation models.

3.2.5 Workflow orchestration for coupling

Workflow orchestration is a crucial point of couplings whenever high-performance simulations are expected from the coupled simulations, in particular toward exascale performance, see e.g. [16-17] and the references therein for the experience of the US exascale project. In the European HPC community, there are significant contributions by several research groups, one of them is O-PALM (or OpenPALM) from CERFACS [18-19]. O-PALM serves as a dynamic coupler, allowing the implementation of couplings in scenarios where the scheduling of component execution and data exchange patterns cannot be fully predetermined before execution. Several massively scalable CFD projects applied O-PALM successfully, see the paper [19]. O-PALM applies the MPI message passing and process management standard library, with interfaces from several programming environments including C/C++ thus seems suitable for a deep study for dynamic two-way and one-way couplings in HiDALGO2, e.g., for UAP and UB couplings. Another popular coupler in environmental simulations is the OASIS3, or its more recent version OASIS-MCT-3.0 and successors, for a description and analysis see [18] and references therein. It allows for the coupling of numerous components (up to 10k) operating sequentially on the same set of tasks. Additionally, it supports coupling within a single component, facilitating connections between different grids or decompositions, such as those associated with physics, dynamics, and I/O.

After a thorough analysis of the coupling tools, the HiDALGO2 application providers will select the most suitable tools and either apply them in the couplings, or select the necessary features to develop a HiDALGO2-specific lightweight coupler.

4 Conclusions

This document provides the initial functional and technical specification of the DMS, a key component of the HiDALGO2 architecture, which provisions the datasets required for the execution of pilots' simulations in target infrastructures, such as HPC clusters, and collects theirs results into a centralized storage, for further analysis and visualization. The functional specification of the DMS leverages the requirements elicited from pilots' stakeholders. Based on this functionality, a selection of suitable baseline technologies is made and justified. This provision endows the DMS with its required components, which are assembled according to the technical architecture specified in this document. In this architecture, two complementary main data storages have been included, namely CKAN and HDFS. CKAN offers a Web front-end for browsing and managing the data and a significant number of optional plugins. HDFS in complemented with HUE as its main front-end for browsing and with Atlas for metadata and catalogue management. A first deployment of the HiDALGO2 DMS is described in some detail as well. This first deployment includes CKAN, HDFS, NIFI, and a first implementation of the HDFS2HPC data transfer framework based on NIFI, which offers a CLI to end users.

This document introduces the notion of pilot's coupling scenarios and the associated coupling technologies. Then, it surveys the available technologies in the state of the art, facilitating the future provision of those suitable ones for the implementation of the HiDALGO2 coupling scenarios.

References

1. HiDALGO2 report. D4.2 Infrastructure Provisioning, Workflow Orchestration and Component Integration, 2023.
2. Wilkinson, M., Dumontier, M., Aalbersberg, I. *et al.* The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data* **3**, 160018 (2016). **Error! Hyperlink reference not valid.**
3. Fielding, Roy Thomas (2000). "Chapter 5: Representational State Transfer (REST)". Architectural Styles and the Design of Network-based Software Architectures (Ph.D.). University of California, Irvine. https://ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
4. HiDALGO2 report. D2.1 Requirements Analysis Definition (M6), 2023.
5. Spivey, B., & Echeverria, J. (2015). Hadoop Security: Protecting your big data platform. " O'Reilly Media, Inc.".
6. Didier G. Vernay, Benny Raphael, Ian F.C. Smith, Augmenting simulations of airflow around buildings using field measurements, Advanced Engineering Informatics, Volume 28, Issue 4, 2014, pp. 412-424, https://doi.org/10.1016/j.aei.2014.06.003.
7. Laurent Y.M. Gicquel, N. Gourdain, J.-F. Boussuge, H. Deniau, G. Staffelbach, P. Wolf, Thierry Poinsot, High performance parallel computing of flows in complex geometries, Comptes Rendus Mécanique, Volume 339, Issues 2–3, 2011, pp. 104-124, https://doi.org/10.1016/j.crme.2010.11.006.
8. Weather Research and Forecasting (WRF) Model Data: https://data.eol.ucar.edu/dataset/545.002#:~:text=WRF%20is%20a%20state-of-the-art%20atmospheric%20modeling%20system%20designed,hours%20at%2000%2C%2006%2C%2012%20and%2018%20hours.
9. Veen, L.E., Hoekstra, A.G. (2020). Easing Multiscale Model Design and Coupling with MUSCLE 3. In: Krzhizhanovskaya, V., et al. Computational Science – ICCS 2020. ICCS 2020. Lecture Notes in Computer Science, vol 12142. Springer, Cham. https://doi.org/10.1007/978-3-030-50433-5_33
10. Jean-Luc Falcone, Bastien Chopard, Alfons Hoekstra, MML: towards a Multiscale Modeling Language, Procedia Computer Science, Volume 1, Issue 1, 2010, Pages 819-826, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2010.04.089.
11. Groen, D., Arabnejad, H., Suleimenova, D., Edeling, W., Raffin, E., Xue, Y., ... & Coveney, P. V. (2023). FabSim3: An automation toolkit for verified simulations using high performance computing. *Computer Physics Communications*, *283*, 108596
12. Sfiligoi 2008 J. Phys.: Conf. Ser. 119 062044

13. Bosak, B., Piontek, T., Karlshoefer, P., Raffin, E., Lakhlili, J., Kopta, P. (2021). Verification, Validation and Uncertainty Quantification of Large-Scale Applications with QCG-PilotJob. In: Paszynski, M., Kranzlmüller, D., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M. (eds) Computational Science – ICCS 2021. ICCS 2021. Lecture Notes in Computer Science, vol 12746. Springer, Cham. https://doi.org/10.1007/978-3-030-77977-1_39
14. Pelupessy, I., van Werkhoven, B., van den Oord, G., Zwart, S. P., van Elteren, A., & Dijkstra, H. (2018, October). Development of the OMUSE/AMUSE Modeling System. In *2018 IEEE 14th International Conference on e-Science (e-Science)* (pp. 374-374). IEEE.
15. Pelupessy, F. I., A. van Elteren, N. de Vries, S. L. W. McMillan, N.Drost, and S. F. Portegies Zwart, 2013, “The Astrophysical Multipurpose Software Environment,” Astronomy and Astrophysics 557, 84.
16. Suchyta E, Klasky S, Podhorszki N, et al. The Exascale Framework for High Fidelity coupled Simulations (EFFIS): Enabling whole device modelling in fusion science. The International Journal of High Performance Computing Applications. 2022;36(1):106-128. doi:10.1177/10943420211019
17. Evans TM, White JC. Multiphysics coupling in the Exascale computing project. The International Journal of High-Performance Computing Applications. 2022;36(1):3-4. doi:10.1177/10943420211028943
18. Piacentini, A. [et al.]. O-Palm: an open-source dynamic parallel coupler. A: COUPLED IV. "COUPLED IV: proceedings of the IV International Conference on Computational Methods for Coupled Problems in Science and Engineering". CIMNE, 2011, p. 885-895. ISBN 978-84-89925-78-6. http://hdl.handle.net/2117/327474
19. Duchaine, F, Jauré, S, Poitou, D, Quémerais, E, Staffelbach, G, Morel, T, Gicquel, L, Analysis of high-performance conjugate heat transfer with the OpenPALM coupler. Computational Science & Discovery, Volume 8, Number 1, 2015. https://dx.doi.org/10.1088/1749-4699/8/1/015003

Annex 1 Extended requirements of Data Management System

These requirements are extension of collection delivered along with D2.1.

#ID	Title	Description	Type	Priority	Risk	Required by	Responsible	Dependencies
REQ-DMA-017	Data storage for unstructured and (semi) structured data	Data server must support the storage of unstructured and (semi) structured data (i.e., datasets)	F	H	L	All	ATOS	REQ-DMA-001
REQ-DMA-018	Data storage for relational data	Data server must support the storage of relational data (i.e., databases)	F	H	L	UNISTRA	ATOS	REQ-DMA-001
REQ-DMA-019	Data storage for streamed data	Data server must support the online storage of streamed data	F	L	L	UNISTRA, Meteogrid	ATOS	REQ-DMA-001
REQ-DMA-020	Textual format	Data server must support the storage of textual-format datasets	F	H	L	All	ATOS	REQ-DMA-017
REQ-DMA-021	Binary format	Data server must support the storage of binary-format datasets	F	H	L	All	ATOS	REQ-DMA-017
REQ-DMA-022	Data versioning	Data server must support the versioning of evolving instances of datasets	F	L	L	UNISTRA, SZE	ATOS	REQ-DMA-017
REQ-DMA-023	Data tracking	Data server must support the tracking of versions of datasets	F	L	L	UNISTRA, SZE	ATOS	REQ-DMA-024

REQ-DMA-024	High data volumes	Data server must support the storage of high volumes of datasets	NF	H	L	ALL	ATOS	REQ-DMA-038
REQ-DMA-025	Scalable data storage	Data server must be scalable across increasing storage infrastructure	NF	H	L	ALL	ATOS	REQ-DMA-026
REQ-DMA-026	Dataset metadata	Associate metadata to stored datasets	F	H	L	UNISTRA, MeteoGrid	ATOS	REQ-DMA-017
REQ-DMA-027	Browse/Search data	Allow data browsing/searching based on metadata	F	H	L	ALL	ATOS	REQ-DMA-026, REQ-DMA-028
REQ-DMA-028	Query data based on internal data structure (for structured data)	Allow data querying based on the internal schema of relational data	F	L	L	UNISTRA	ATOS	REQ-DMA-018
REQ-DMA-029	Query data based on internal indexing (for un-structured data)	Allow data querying based on the internal indexing of un-structured and (semi) structure data	F	L	L	ALL	ATOS	REQ-DMA-017
REQ-DMA-030	API for CRUD operations on datasets	Allow to create, read or delete operations on datasets stored in the central Data server	F	H	L	ALL	ATOS	REQ-DMA-017, REQ-DMA-018
REQ-DMA-031	API for search/query	Allow search/query operations on stored datasets	F	H	L	ALL	ATOS	REQ-DMA-017, REQ-DMA-018
REQ-DMA-032	CLI/REST/Web channels for APIs	Create CLI, REST, and Web API channels for supported storage and associated services	F	M	L	ALL	ATOS	REQ-DMA-017
REQ-DMA-033	HPC - Central Storage transfer	Allows bidirectional data transfer between central storage and target HPC	F	H	L	ALL	ATOS	REQ-DMA-003, REQ-DMA-036, REQ-DMA-038

REQ-DMA-034	SSH based data transfer	Allows ssh-based data transfer between central storage and target HPC	F	H	L	ALL	ATOS	REQ-DMA-037
REQ-DMA-035	User's credentials management for HPC access	Allows access to user's HPC credentials for data transfer	F	H	L	ALL	ATOS	REQ-DMA-015
REQ-DMA-036	High performance high data volumen HPC transfer	Allows bidirectional high-performance data transfer of high volumes of data from HPC and the central storage	NF	H	L	ALL	ATOS	REQ-DMA-003, REQ-DMA-035
REQ-DMA-037	Low latency data access	Allows low latency when access data stored in central storage	NF	H	L	ALL	ATOS	REQ-DMA-038
REQ-DMA-038	Data procurement from Web Data providers (HTTP, REST)	Allows to download datasets from Internet data providers into the central data storage, using standard Web channels, such as HTTP, REST, FTP	F	H	L	ALL	ATOS	REQ-DMA-002
REQ-DMA-039	Data procurement from other Internal data servers	Allows to download datasets from other internal data providers into the central data storage, using server specific channels	F	M	L	PSNC	ATOS	REQ-DMA-002, REQ-DMA-040
REQ-DMA-040	Support data access restrictions	Enable data access restrictions in central data storage	F	M	L	UNISTRA, SZE	ATOS	REQ-DMA-015
REQ-DMA-041	Integration with other data lakes	Enable support with Destination Earth data lakes through the Destination Earth data bridge	F	H	L	SZE, UNISTRA	ATOS	REQ-DMA-011, REQ-DMA-012

Table 1. Extended requirements for DMS of HiDALGO2