Software layout plan for a unique AI framework

Table of Contents

List of Figures

List of Tables

Executive Summary

High-Performance Computing (HPC) become more broadly available for researchers, engineers, and scientists through the transition process from the Partnership for Advanced Computing in Europe (PRACE) towards EuroHPC Joint Undertaking (JU) systems[1] such as LUMI, LEONARDO, VEGA, MELUXINA, KAROLINA, DISCOVERER, DEUCALION, or MARE NOSTRUM 5. The first European Exascale system Joint Undertaking Pioneer for Innovative and Transformative Exascale Research (JUPITER)[2] will be installed at the Jülich Supercomputing Centre (JSC) in 2024. All these systems are heterogeneous in nature with respect to their underlying hardware and software technologies. This raises challenges for HPC and Artificial Intelligence (AI) users to seamlessly use these systems, and to easily port their applications, depending on their grant of computing time, between the systems. The European Center of Excellence in Exascale Computing "Research on AI- and Simulation-Based Engineering at Exacale" (CoE RAISE) develops a Unique AI Framework (UAIF) design that guides users of these systems to specifically identified HPC/AI technologies that are possible to be used at scale. The goal is to lower the barrier of using and porting HPC/AI applications at Exascale, especially helping those AI users that have not used HPC systems before.

The UAIF converged to a consistent set of components available for a broad range of HPC/AI applications. The Work Package (WP)2 activities have analyzed, benchmarked, and checked such components with respect to their suitability for Exascale execution. The adoption plans of the UAIF started with potential integrations at EuroHPC JU hosting sites or jointly working together with National Competence Centers (NCCs) of the EuroCC-1 and EuroCC-2 projects[3] on industrial use cases. Other planned UAIF adoptions are within Digital Twin (DT) projects, e.g., Destination Earth[4], or other Euopean Centers of Excellence (CoEs). A core component of the UAIF is available as Load AI Modules, Environments, and Containers (LAMEC) Application Programming Interface (API). This interface enables a streamline access to HPC systems by abstracting from low-level software versions and module environment details. Through the adoption plans, this LAMEC API is continously extended to support more European Union (EU) HPC systems over time. The LAMEC API also paves the way for an easier adoption by new AI users that have previously not used HPC systems for speeding up their trainings or inference, or have not used Hyperparameter Optimization (HPO) before. Examples include the integration of the LAMEC API within established AI community platforms such as OpenML[5] or ClearML[6].

[1] EuroHPC JU HPC systems https://eurohpc-ju.europa.eu/about/our-supercomputers_en
[2] JUPITER Exascale HPC system https://digital-strategy.ec.europa.eu/en/news/eu-enters-Exascale-era-announcement-new-supercomputing-hosting-sites
[3] EuroCC projects https://www.eurocc-access.eu
[4] Digital Twin Destination Earth https://digital-strategy.ec.europa.eu/en/policies/destination-earth
[5] OpenML platform https://www.openml.org/
[6] ClearML MLOps platform https://clear.ml/

1 Introduction

This document "Deliverable D2.13 - Software layout plan for a unique AI framework" of project month M26 provides an updated description of the UAIF software layout plan and its components based on co-design activities contributed by WP3 “Compute-Driven Use-Cases at Exascle” and WP4 “Data-Driven Use-Cases at Exascle” of CoE RAISE. The monitoring and interaction of these co-design activities by WP2 continues to be executed via the Interaction Room Methodology [2, 3] using Mural Boards[7] from the project inception. A list of the boards for each CoE RAISE is provided in Appendix B (Mural Board List of CoE RAISE Use Cases).

The structure of this Deliverable is as follows. Section 2 describes the overall framework software layout plan with a list of all considered components and an updated set of requirements. More details on selected updated UAIF components are given in Section 3. Sec. 4 describes the adoption plans of the UAIF with an emphasis on external stakeholders and the CoE RAISE Certification program. Finally, a short summary of this Deliverable is provided and some conclusions are drawn in Sec. 5.

An initial software layout plan of the UAIF has been provided in "D2.12 - Software Layout Plan for a unique AI Framework" (M9) and a short update of its components is also available in "D2.10 - Monitoring Report (M18)". Both previous versions of the UAIF have been added to Appendix A (Previous Framework Versions) for the readers convenience. This Deliverable is thus intended to provide an updated and comprehensive description for both project-internal and external adopters of the UAIF. To facilitate external adoption of the UAIF, this Deliverable also describes the plan of the adoption across EuroHPC JU hosting sites, EuroCC(-2) NCCs, and domain-specific communities through DTs and CoEs.

This document is primarily intended for the WP3 and WP4 use case application experts of CoE RAISE, potential adopters of the UAIF from the larger HPC community, and CoE RAISE stakeholders. It aims at similar HPC/AI application use case developers such as in CoE RAISE that are part of the larger EuroHPC and international HPC and AI community, and ecosystem. Additionally, the document serves as an initial information for AI communities that never used HPC before, but want to adopt certain elements of the UAIF. Already several projects, initiatives, and researchers have started to adopt certain findings of CoE RAISE and the UAIF in the context of their own use cases outside of the project's application domains.

[7]Mural Boards https://mural.co

2 Overall Framework Software Layout Plan

This section provides a comprehensive overview of the framework software layout plan, including major updates since the last reporting period. It briefly describes each component of the framework. Previous versions of the UAIF have been added to Appendix A, including the initial version. They were also published in Riedel et al. [4]. WP2, in cooperation with WP6, provides different trainings on a wide variety of UAIF components or its approaches. They are all available in CoE RAISE's YouTube Channel[8] and are listed below in the context of the UAIF components.

Figure 1 below shows the current software layout plan of the UAIF at project month M26 with several new components and in-depth refinements on its core building blocks. Updates from the previous Deliverable include merging several components to reduce the UAIF complexity, necessitating a re-ordering of the alphabetic characters for each component. New components or major updates in the UAIF design layout are marked with *'NEW'* in Fig. 1. The figure includes several components that are not directly under the control of CoE RAISE, but are identified as important dependencies for the UAIF running on HPC systems. Furthermore, any type of use case application software such as simulation sciences codes (e.g., using numerical methods based on known physical laws) that integrate AI codes or cutting-edge Deep Learning (DL) models can be used in conjunction with the UAIF components. Those use case applications and simulation sciences codes have been kept out of Fig. 1 since CoE RAISE's WP2 main focus is on supporting AI.

The structure of this section follows the main items depicted in Fig. 1. That is, Sec. 2.1 describes the components of the application layer and Sec. 2.2 the reference architecture layer of the UAIF. This is followed by the a description of the components of the software infrastructure layer in Sec. 2.3. Finally, Sec. 2.4 provides details on the hardware infrastructure components.

2.1 Applications

The application layer contains components on compute- and data-intensive CoE RAISE use cases (component (A) in Fig. 1), domain-specific CoE RAISE use cases (component (B) in Fig. 1), NCC and industrial use cases (component (C) in Fig. 1), and DT use cases (component (D) in Fig. 1). These components are described in detail in the following Sec. 2.1.1, Sec. 2.1.2, Sec. 2.1.3, and Sec. 2.1.4. In Fig. 1, the large red arrow represent the co-design activities that influence the project reference architecture components. The large green arrows represent the benefits and adoption potential of the UAIF for external project activities in the larger HPC ecosystem.

2.1.1 A - Compute- and Data-Intensive CoE RAISE Use Cases

Component (A) in Fig. 1 represents the co-design efforts of the UAIF based on compute- and data-intensive use cases[9]. Fact Sheets for each use case have been produced to describe what novel AI methods correlate to available UAIF components. They foster general understanding of the contributions that have been added over time to the UAIF and include scalability and utility for Exascale aspects. Several different tasks in WP2 contributed to benchmarking and proof of scalability of selected components of the UAIF on various production and prototype HPC systems in this context, see also related details in Sec. 3. More information on the Fact Sheets for each CoE RAISE use case are provided in "Deliverable D2.10 - Monitoring Report" (M18).

[8] CoE RAISE YouTube Channel https://www.youtube.com/@coeraise6339
[9] CoE RAISE Use Cases https://www.coe-raise.eu/use-cases

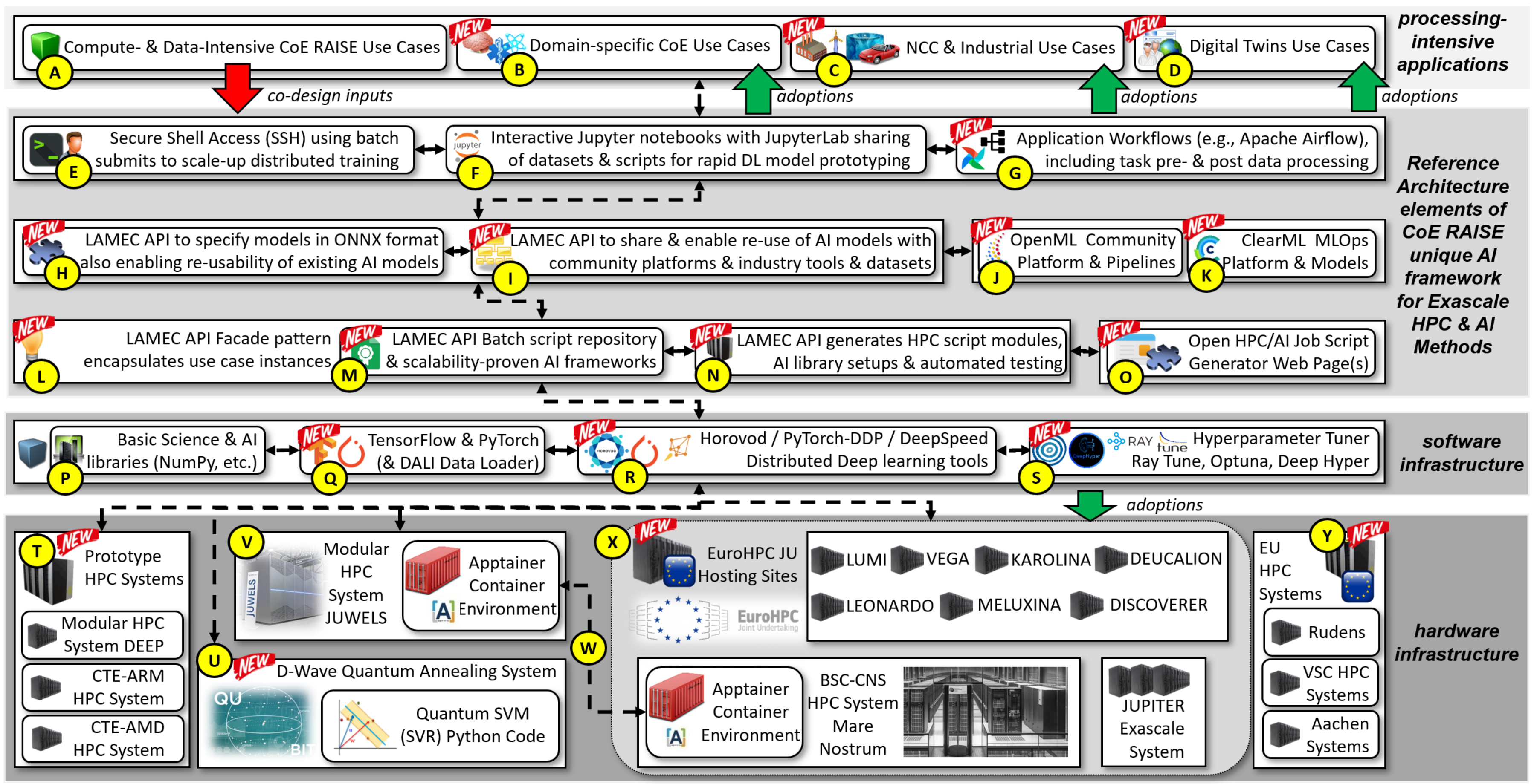

Compute- & Data-Intensive CoE RAISE Use Cases
A
NEW
Domain-specific CoE Use Cases
B
NEW
NCC & Industrial Use Cases
C
NEW
Digital Twins Use Cases
D
processing-intensive applications
co-design inputs
adoptions
adoptions
adoptions
Secure Shell Access (SSH) using batch submits to scale-up distributed training
E
Interactive Jupyter notebooks with JupyterLab sharing of datasets & scripts for rapid DL model prototyping
F
NEW
Application Workflows (e.g., Apache Airflow), including task pre- & post data processing
G
NEW
LAMEC API to specify models in ONNX format also enabling re-usability of existing AI models
H
NEW
LAMEC API to share & enable re-use of AI models with community platforms & industry tools & datasets
I
NEW
OpenML Community Platform & Pipelines
J
NEW
ClearML MLOps Platform & Models
K
Reference Architecture elements of CoE RAISE unique AI framework for Exascale HPC & AI Methods
NEW
LAMEC API Facade pattern encapsulates use case instances
L
NEW
LAMEC API Batch script repository & scalability-proven AI frameworks
M
NEW
LAMEC API generates HPC script modules, AI library setups & automated testing
N
NEW
Open HPC/AI Job Script Generator Web Page(s)
O
Basic Science & AI libraries (NumPy, etc.)
P
NEW
TensorFlow & PyTorch (& DALI Data Loader)
Q
NEW
Horovod / PyTorch-DDP / DeepSpeed Distributed Deep learning tools
R
NEW
RAY Tune
Hyperparameter Tuner Ray Tune, Optuna, Deep Hyper
S
software infrastructure
adoptions
T
NEW
Prototype HPC Systems
Modular HPC System DEEP
CTE-ARM HPC System
CTE-AMD HPC System
V
Modular HPC System JUWELS
JUWELS
Apptainer Container Environment
U
NEW
D-Wave Quantum Annealing System
Quantum SVM (SVR) Python Code
W
X
NEW
EuroHPC JU Hosting Sites
EuroHPC Joint Undertaking
LUMI
VEGA
KAROLINA
DEUCALION
LEONARDO
MELUXINA
DISCOVERER
Apptainer Container Environment
BSC-CNS HPC System Mare Nostrum
JUPITER Exascale System
EU HPC Systems
Y
NEW
Rudens
VSC HPC Systems
Aachen Systems
hardware infrastructure

Figure 1: UAIF software layout plan at project month M26. There have been several changes and additions marked as *'NEW'* during the last reporting period. One core component update is the introduction of the UAIF LAMEC API.

Detailed co-design activities have been performed via the Interaction Room methodology and Mural Boards. A list of each board per use case is provided in Appendix B (Mural Board List of CoE RAISE Use Cases). During the project and especially in the last reporting period, a clear picture is provided on what components are relevant for the UAIF.

2.1.2 B - Domain-Specific CoE Use Cases [New]

A wide variety of CoEs[10] have been funded in different domain-specific areas providing use cases that leverage simulation sciences or AI/HPC methods to utilize the emerging Exascale computing. At the time of writing, another EuroHPC JU Work Programme (WOPRO)[11] outlining future funding of CoEs addresses the needs of large user communities in four specific areas of application domains. As shown in Fig. 1 (B), the UAIF is recommended to CoEs to adopt the UAIF to prevent AI developers in domain-specific sciences wasting a lot of efforts, cf. Sec. 4.3.

2.1.3 C - NCC and Industrial Use Cases [New]

A pan-European network of NCCs has been created under the EuroCC-1 and EuroCC-2 project umbrella to enable industry and Small and Medium Enterprises (SMEs) to leverage HPC resources made available via EuroHPC[12]. Component (C) of Fig. 1 has been added to represent adoptions of the UAIF by NCCs and the significant potential to governmental, academic, industry, and SME partners to speed-up and scale-up their applications towards Exascale. More information about potentials for NCC adoptions is provided in Sec. 4.2.

2.1.4 D - Digital Twins Use Cases [New]

DTs and corresponding workflows as they are developed, e.g., in the Destination Earth[13] or interTwin[14] projects, are becoming important for scientific and engineering HPC users in Europe. Component (D) has been added to Fig. 1 to represent the processing-intensive applications of DTs that are also highly relevant for CoE RAISE, either the DTs adopting parts of UAIF components or including new use cases in CoE RAISE. More information about potentials for DT adoptions is provided in Sec. 4.4.

2.2 Reference Architecture Elements

This section describes the reference architecture components relevant for the UAIF for Exascale HPC/AI methods, which are listed in Fig. 1 in the second layer (components (E) – (O)). This covers descriptions of the Secure Shell (SSH) low-level access (see Sec. 2.2.1 – (E)), Jupyter notebooks high-level access (see Sec. 2.2.2 – (F)), application workflows (see Sec. 2.2.3 – (G)), LAMEC API Open Neural Network Exchange (ONNX) standard elements (see Sec. 2.2.4 – (H)), LAMEC API community platform integration (see Sec. 2.2.5 – (I)), community platform OpenML interoperability (see Sec. 2.2.6 – (J)), ClearML MLOps platform interoperability (see Sec. 2.2.7 – (K)), LAMEC API facade pattern implementation (see Sec. 2.2.8 – (L)), LAMEC API batch script repository (see Sec. 2.2.9 – (M)), LAMEC API batch script generator (see Sec. 2.2.10 – (N)), and open HPC/AI script generator web page(s)(see Sec. 2.2.11 – (O)).

[10] EU HPC Centres of Excellence https://www.hpccoe.eu/eu-hpc-centres-of-excellence2/
[11] EuroHPC JU Work Programme 2023 https://eurohpc-ju.europa.eu/documents_en
[12] EuroHPC JU HPC Systems https://eurohpc-ju.europa.eu/about/our-supercomputers_en
[13] Digital Twin Destination Earth https://digital-strategy.ec.europa.eu/en/policies/destination-earth
[14] InterTwin https://www.intertwin.eu

2.2.1 E - Secure Shell (SSH) Low-Level Access

As shown in Fig. 1 (E), the first reference architecture element includes the use of the SSH protocol into the plan. Principally, as a means to remotely log into HPC systems and submit batch scheduler scripts, e.g., via the Simple Linux Utility for Resource Management (SLURM) [5] tool, it remains one of the integral access methods for HPC applications. It hence needs to be provided to researchers. One example of relevance for CoE RAISE is that AI researchers often use batch scripts for distributed training of DL models to leverage the high number of Graphical Processing Units (GPUs) that are available on HPC systems.

2.2.2 F - Jupyter Notebooks High-Level Access

AI researchers frequently require some form of interactive access to HPC systems to facilitate quick and rapid prototyping of Machine Learning (ML) and DL algorithms and models. Component (F) in Fig. 1 represents the acknowledgement of this need in the UAIF by offering Jupyter notebooks and JupyterLabs[15]. In addition to interactive graphical access via web interfaces, it may extend the SSH component presented in Sec. 2.2.1 to create SSH sessions out of Jupyter notebooks on HPC systems, while at the same time being complemented by the Jupyter environment and a wide variety of useful extensions. CoE RAISE offers access to JupyterLab instances running at JSC through its service portal[16].

2.2.3 G - Application Workflows [New]

The component (G) in Fig. 1 is a new addition to the UAIF and supports application workflows and workflow automation, including task pre- and post data processing capabilities. The UAIF recommends the Apache Airflow[17] tool that is a platform to programmatically author, schedule, and monitor workflows. The UAIF application workflow set may include more workflow automation tools, such as Elyra[18], in the future. More technical information about application workflows, including application use cases, is provided in Sec. 3.9.

2.2.4 H - LAMEC API ONNX Standard Elements [New]

As shown in Figure 1 (H), a key component of the overall UAIF LAMEC API is the fast portability between different DL frameworks and reproducibility achieved by using the standard ONNX format wherever possible. While one short ONNX report was developed in CoE RAISE and that is available in the internal document repository of WP2, the implementation of the overall UAIF LAMEC API using ONNX is still work in progress.

2.2.5 I - LAMEC API Community Platform Integration [New]

Another element of the overall UAIF LAMEC API shown in Fig. 1 is component (I), which represents a seamless integration with other tools. The goal is to use the LAMEC API to share and re-use existing AI models of CoE RAISE with community platforms (see also Sec. 2.2.6 and Sec. 2.2.7), industry tools, datasets, and to enable Transfer Learning (TL). While initial discussions with community platforms have taken place, the implementation of the overall UAIF LAMEC API integration and

[15] Project Jupyter https://jupyter.org/
[16] CoE RAISE Service Portal https://www.coe-raise.eu/service-portal
[17] Apache Airflow https://airflow.apache.org/
[18] Elyra https://github.com/elyra-ai/elyra

provisioning of AI models is still work in progress.

2.2.6 J - Community Platform OpenML Interoperability [New]

OpenML[19] is an open community platform for sharing datasets, algorithms, models, and experiments in the realm of AI with a wide variety of traditional ML approaches. One ansatz to enlarge the user community of CoE RAISE is to integrate UAIF components into the OpenML platform such that experiments can be also run on cutting-edge HPC systems where available. Hence, component (J) in Fig. 1 represents how this community might leverage the LAMEC API components described in Sec. 2.2.5. Initial integration has started with OpenML, including one joint training with OpenML that is available on the CoE RAISE YouTube Channel: *RAISE CoE Training: Using OpenML for sharing datasets, algorithms, and experiments*[20]. At the time of writing, the CoE RAISE team focuses on the core functionality of the LAMEC API. An integration within OpenML is planned for the second half of the year 2023.

2.2.7 K - ClearML MLOps Platform Interoperability [New]

ClearML[21] is an Machine Learning Operations (MLOps) platform that can be used to develop, orchestrate, and automate ML workflows at scale. The CoE RAISE consortium provides one installation of ClearML for its internal and external users, see Sec. 3.2. Another approach to enlarge the user community of CoE RAISE is to integrate UAIF components into MLOps platforms such as ClearML that are often used in industry such that its tasks can be also run on cutting-edge HPC systems, where useful. Hence, the component (K) in Fig. 1 represents how this community might leverage the LAMEC API components described above in Sec. 2.2.5 through integration with MLOps platforms. Experience with ClearML exists in CoE RAISE. The project offers the service to users, and a training on ClearML is available on the CoE RAISE YouTube Channel: *RAISE CoE Training: MLOps with ClearML*[22]. Full integration with ClearML is planned in the second half of the year 2023.

2.2.8 L - LAMEC API Facade Pattern Implementation [New]

To map the abstract specifications of software and hardware needs by AI researchers to specific software and hardware HPC infrastructure elements, a facade pattern is used by the UAIF LAMEC API general design. Hence, as represented by component (L) in Fig. 1, the UAIF software layout design employs an abstract wrapper functionality that maps the abstract specifications from users to specific software and hardware configurations. The framework element is marked as *'NEW'*. It is, however, rather a refinement through implementation of the general abstract idea. Apart from the above mentioned parts of the LAMEC API, the core of this API is split into the two following elements. The first core element is a batch script repository explained in Sec. 2.2.9 and the second is an API using this repository to generate new batch script elements, which is described in Sec. 2.2.10. More technical information about the current implementation status of these two core elements of the LAMEC API is provided in Sec. 3.6.

[19] OpenML https://www.openml.org/
[20] CoE RAISE Training on OpenML available on YouTube https://www.youtube.com/watch?v=xAuXDxGQsxo
[21] ClearML https://clear.ml/
[22] CoE RAISE Training on ClearML available on YouTube https://www.youtube.com/watch?v=lowmDcR5qL8

2.2.9 M - LAMEC API Batch Script Repository [New]

As described above in Sec. 2.2.8, the first core element of the UAIF LAMEC API is a batch script repository. It consists of batch scripts for specific HPC systems with a correct setup of modules needed for using specific UAIF AI tools (see Sec. 2.3.1, Sec. 2.3.2, Sec. 2.3.3, and Sec. 2.3.4). The framework element is marked as *'NEW'*. It is, however, rather a refinement through implementation of the general abstract idea already reported in previous Deliverables. Earlier elements of the implementation of this component are also part of a training on the UAIF that is available on the CoE RAISE YouTube Channel: *RAISE CoE Training: Towards a CoE RAISE Unique AI Software Framework for Exascale*[23]. As previously described and represented by component (M) in Fig. 1, one idea is to use this repository with the UAIF LAMEC API (see also Sec. 2.2.10). It quickly becomes clear that the repository in itself is also a great resource for AI/HPC researchers that already know how to deal with changing HPC modules in batch scripts. More technical information about the current implementation status of this first core element of the LAMEC API is provided in Sec. 3.6.

2.2.10 N - LAMEC API Batch Script Generator [New]

As referenced above in Sec. 2.2.8, the second core element of the UAIF LAMEC API, which is represented by component (N) in Fig. 1, is using the above mentioned repository to generate new batch script segments. This lowers the barrier for entry to leveraging HPC systems for AI researchers that may not have much experience working with modules in HPC environments, as well as saving valuable time through automation for experienced users. Additional components beyond verified site AI modules and libraries, such as AI model scripts or datasets for training and inference, are planned for later addition (although these are usually elements of a job script that inexperienced AI researchers do not find challenging). While the implementation is work in progress, both core elements of the UAIF LAMEC API were already demonstrated on selected HPC systems during the "all hands meeting" at the European Organization for Nuclear Research (CERN) in January 2023. More technical information about the current implementation status of the core elements of the LAMEC API is provided in Sec. 3.6.

2.2.11 O - Open HPC/AI Script Generator Web Page(s) [New]

The open HPC/AI job script generator web page(s) shown as component (O) in Fig. 1 uses the implementation of the UAIF LAMEC API. The general concept is derived from existing job script generators available at the Swiss National Supercomputing Centre (CSCS)[24] or the National Energy Research Scientific Computing Center (NERSC)[25], where the difference to these existing tools lies in the use of UAIF components with a specific focus on AI toolsets. The job script generator might be hosted at several sites to provide seamless access to AI tools on a variety of HPC systems, not only one per HPC site.

2.3 Software Infrastructure

The software infrastructure layer components (P) – (S), which are depicted in Fig. 1. are presented in this section. This layer contains four components, i.e., basic science libraries (P), DL libraries (Q), distributed DL tools (R), and hyperparameter tuner (S), which are described in the following

[23]CoE RAISE Training on UAIF available on YouTube https://www.youtube.com/watch?v=cCkfcrXhNdU
[24]CSCS job script generator https://user.cscs.ch/access/running/jobscript_generator/
[25]NERSC job script generator https://my.nersc.gov/script_generator.php

Sec. 2.3.1, Sec. 2.3.2, Sec. 2.3.3, and Sec. 2.3.4. Again, a green arrow represents adoptions (see the connection to the hardware layer in Fig. 1, presented in Sec. 2.4).

2.3.1 P - Basic Science Libraries

Despite the massive increase of DL tools and packages, and their uptake in the AI communities, there remains a core of basic science libraries heavily used by CoE RAISE communities. Examples of these basic science libraries for AI are NumPy[26] and scikit-learn[27]. This building block (P) in Fig. 1 of the UAIF also includes simulation science codes, e.g., those using numerical methods based on known physical laws and that have the potential to benefit from coupling to AI models. Since the CoE RAISE project focuses primarily on AI models, the various relevant simulation science codes have been kept out of the UAIF software layout plan. Instead, the reader is referred to the Fact Sheets of the CoE RAISE use case applications that have been described in "D2.10 - Monitoring Report" (M18). They include simulation science codes where relevant. More technical information about basic science libraries is provided in Sec. 3.4.

2.3.2 Q - Deep Learning Libraries [New]

The UAIF recommends the use of PyTorch[28] and TensorFlow[29]. CoE RAISE has tested their performance and scalability in depth using various applications during the last two years. Although these two libraries were featured in previous UAIF software layout plans, this component (Q) is marked as *'NEW'* in Fig. 1 due to the inclusion of the NVIDIA Data Loading Library (DALI)[30] since the last reporting period. DALI further increases the performance of PyTorch and TensorFlow. This inclusion is represented by component (Q) in Fig. 1 in parenthesis due to the proprietary nature and support for NVIDIA GPUs. At the time of writing, CoE RAISE continues to investigate libraries of other GPU vendors such as from Advanced Micro Devices (AMD). More technical information about the DALI data loader is provided in Sec. 3.1.

2.3.3 R - Distributed Deep Learning Tools [New]

Component (R) in Fig. 1 outlines three supported libraries used for accelerating distributed AI model training by leveraging the large number of GPUs available at cutting-edge HPC sites today. Earlier implementations of this component are available as a part of a training on CoE RAISE's YouTube Channel: *RAISE CoE Training: Distributed Deep Learning*[31]. PyTorch-Distributed Data Parallel (DDP)[32] and Horovod[33] was already included in earlier UAIF software layout plans. This component is marked as *'NEW'* due to the addition of DeepSpeed[34]. More technical information about the newly added DeepSpeed tool is provided in Sec. 3.5.

[26] NumPy https://numpy.org/
[27] scikit-learn https://scikit-learn.org/stable/
[28] PyTorch https://pytorch.org/
[29] TensorFlow https://www.tensorflow.org/
[30] DALI https://developer.nvidia.com/dali
[31] CoE RAISE Training on Distributed Deep Learning available on YouTube https://www.youtube.com/watch?v=q_J5HGCdiW8
[32] PyTorch Distributed Data Parallel https://pytorch.org/tutorials/beginner/dist_overview.html
[33] Horovod https://github.com/horovod/horovod
[34] DeepSpeed https://www.deepspeed.ai/

2.3.4 S - Hyperparameter Tuner [New]

One of the most successful aspects of the current adoptions of the UAIF are the hyperparameter tuning or HPO tools represented by component (S) in Fig. 1. Trainings reflecting this component and its implementation are available on CoE RAISE's YouTube Channel: *RAISE CoE Training: Hyperparameter Tuning with Ray Tune*[35]. In addition to the previously included Ray Tune tool[36], this component is marked as *'NEW'* due to the addition of the Optuna[37] and DeepHyper[38] tools. More technical information about the newly added DeepHyper tool is provided in Sec. 3.3.2.

2.4 Hardware Infrastructure

The hardware infrastructure layer components (T) – (Y) depicted in Fig. 1 are presented in this section. This layer contains components on prototype HPC systems (see Sec. 2.4.1 – (T)), the D-Wave Quantum Annealing (QA) system (see Sec. 2.4.2 – (U)), the Modular Supercomputing Architecture (MSA) Juelich Wizard for European Leadership Science (JUWELS) (see Sec. 2.4.3 – (V)), container technologies (see Sec. 2.4.4 – (W)), EuroHPC JU hosting sites (see Sec. 2.4.5 – (X)), and EU HPC systems (see Sec. 2.4.6 – (Y)).

2.4.1 T - Prototype HPC Systems [New]

The benchmarking and porting activities of WP2 have been performed on a number of interesting prototype HPC systems that feature new and emerging technologies. Since the beginning of the project, the Dynamical Exascale Entry Platform (DEEP)[39] system has been used to experiment with the MSA type of HPC architecture [6, 7]. This component (T) is marked as *'NEW'* in Fig. 1 to highlight the activities performed on the two new prototype systems, the Advanced Reduced Instruction Set Computer Machine (ARM)-based CTE-ARM and CTE-AMD, hosted at the Barcelona Supercomputing Centre (BSC) in Spain. The CTE-ARM is a supercomputer based on 192 A64FX ARM processors, with a Linux Operating System (OS) and an Tofu interconnect network (6.8GB/s)[40]. CTE-AMD is a cluster based on AMD EPYC processors, with a Linux OS and an Infiniband interconnection network. Its main characteristic is the availability of two AMD MI50 GPUs per node, making it an ideal cluster for GPU applications[41]. The reader is referred to "Deliverable D2.7 - Support Report" (M18) for more details.

2.4.2 U - D-Wave Quantum Annealer System [New]

Quantum Computing (QC) is gaining momentum as the EuroHPC JU recently funded, together with national contributions, several QC systems[42]. Multiple CoE RAISE use case applications [8–10] have successfully engaged in QC by utilizing the D-Wave QA system available via the Juelich UNified

[35] CoE RAISE Training: Hyperparameter Tuning available on YouTube https://www.youtube.com/watch?v=Ylt7Htnl1zY
[36] Ray Tune https://www.ray.io/ray-tune
[37] Optuna https://optuna.org/
[38] DeepHyper https://deephyper.readthedocs.io/en/latest/
[39] DEEP Prototype HPC System hosted by JSC https://www.fz-juelich.de/en/ias/jsc/systems/prototype-systems/deep_system
[40] CTE-ARM HPC System https://www.bsc.es/innovation-and-services/technical-information-cte-arm
[41] CTE-AMD HPC System https://www.bsc.es/innovation-and-services/technical-information-cte-amd
[42] EuroHPC JU Quantum Computers https://eurohpc-ju.europa.eu/selection-six-sites-host-first-european-quantum-computers-2022-10-04_en

Infrastructure for Quantum computing (JUNIQ)[43] at JSC in Germany. As represented by component (U) in Fig. 1, the quantum AI models implemented were Support Vector Machines (SVMs). They were used for regression tasks via Support Vector Regression (SVR). An implementation of this component is also available as part of a training on SVMs on the CoE RAISE YouTube Channel: *RAISE CoE Training: Quantum Support Vector Machine Algorithms*[44]. More details about these models can be found in Sec. 3.8 and in "D2.8 - Benchmarking and Support Report" (M24).

2.4.3 V - Modular HPC System JUWELS

The MSA-based HPC system JUWELS is massively used within CoE RAISE for co-designing the UAIF and performing necessary speed-up and scaling benchmarks of its components, see component (V) in Fig. 1. It is an ideal HPC system for AI workloads as described by Kesselheim et al. in [11]. More details on its usage in the project can be obtained from "D2.10 - Monitoring Report" (M18) in the context of the use case Fact Sheets.

2.4.4 W - Container Technologies

Container technologies are an important tool within larger AI communities to facilitate porting of applications and datasets between systems. One such example in CoE RAISE is shown as component (W) in Fig. 1, where the porting operation of a containerized application from JUWELS at JSC to the MARE NOSTRUM 4 system at BSC is realized. This transparent deployment of containerized code is made possible by the support of Apptainer[45] (previously named Singularity[46]) available at both sites. Initial test have been performed with containers on both HPC platforms. More application use case uptake is foreseen in the last half of the year 2023. This component of the UAIF is crucial to support more industry applications and to enable easy porting of data science applications that have not used HPC systems before.

2.4.5 X - EuroHPC JU Hosting Sites [New]

Component (X) in Fig. 1 covers the major EuroHPC JU hosting sites[47] that represent stakeholders to adopt the UAIF. Several European HPC systems that are available within CoE RAISE contributed to co-design with applications to the UAIF software layout and design. It is the goal of CoE RAISE is to support as many as possible EuroHPC JU systems during and beyond the lifetime of the project by building on the sustainability strategy developed in WP5 (Business Development). Initial discussions with some some of these sites have been started by WP2 partners to encourage the adoption of the UAIF, and to engage the HPC sites in a CoE RAISE certification process jointly with WP5. At the time of writing, the broader adoption strategy is in its initial stages, while components such the LAMEC API are considered to be further developed adding more EuroHPC JU systems support over time. For additional detail, see Sec. 4.1. One highlight of the planned adoption will be the integration to the first European Exascale system JUPITER[48], which will be installed at JSC in 2024.

[43] JUNIQ
https://www.fz-juelich.de/en/ias/jsc/systems/quantum-computing/juniq-facility

[44] CoE RAISE Training on Quantum SVMs available on YouTube https://www.youtube.com/watch?v=WBRfpBRSepg

[45] Apptainer https://apptainer.org/

[46] JUWELS Container Runtime
https://apps.fz-juelich.de/jsc/hps/juwels/container-runtime.html

[47] EuroHPC JU Hosting Sites https://eurohpc-ju.europa.eu/about/our-supercomputers_en

[48] Path to JUPITER https://www.fz-juelich.de/en/ias/jsc/news/news-items/news-flashes/2023/path-to-jupiter

2.4.6 Y - EU HPC Systems [New]

It is important to consider that the whole landscape of European HPC systems is broader than the EuroHPC JU hosting sites described above. It is observed that new users of the UAIF are often starting using regional or university-level systems before scaling up to larger systems. Component (Y) in Fig. 1 contains examples such as the university-level systems Rudens[49] of the Riga Technical University (RTU), or the HPC systems of Rheinisch-Wesfälische Technische Hochschule Aachen - RWTH Aachen University (RWTH)[50]. Both of these sites are in the process of adopting parts of the UAIF framework and are in discussions with CoE RAISE concerning certification steps. Another example are Belgium regional HPC systems such as the Vlaams Supercomputer Centre (VSC)[51] that are in use by CoE RAISE. There is a wide variety of other HPC system providers, such as commercial and industrial systems in Iceland (e.g., Responsible Compute[52]) that are not shown in Fig. 1 (Y), but are in discussions with CoE RAISE to adopt elements of the UAIF.

[49] Rudens HPC System
https://www.rtu.lv/en/research/science-and-innovation-centre/scientific-equipment-unit/hpc-center

[50] RWTH Aachen University HPC Systems
https://help.itc.rwth-aachen.de/en/service/rhr4fjjutttf/

[51] VSC HPC Systems https://www.vscentrum.be/

[52] Responsible Compute https://responsiblecompute.com/

3 Updates of Selected Framework Components

This section details components that have been recently updated or added to the framework. This also includes major updates of components that have been planned since the beginning of the project. More specifically, Sec. 3.1 describes DALI from NVIDIA and Sec 3.2 ClearML. This is followed by sections on HPO, see Sec. 3.3 and on some basic science libraries, see Sec. 3.4. The tool DeepSpeed is introduced in Sec. 3.5 before the LAMEC API is described in Sec. 3.6. Subsequently, Google JAX is discussed in Sec. 3.7. Finally, Sec. 3.8 and Sec. 3.9 provide details on quantum SVR and setting up workflows with Apache Airflow[53].

3.1 DALI Data Loader

DALI[54] is an open-source framework designed to accelerate the data loading process in DL applications. Usually, a GPU runs computations much faster than a Central Processing Unit (CPU) can provide data, resulting in data starvation. GPU starving can be prevented by moving the data-loading process to the GPU at an early stage. DALI supports multiple data formats and with its unified interface, it is easy to integrate into all common DL frameworks. A large-scale benchmark with the ImageNet dataset [1] on the JUWELS Booster machine comparing the native PyTorch and the DALI (CPU and GPU based) data loader across different distributed data-parallel frameworks has been performed. The results depicted in Fig. 2 show the DALI data loader to achieve high data throughput DT (in images i per second s) on all node configurations for the CPU and GPU based version. As shown in Fig. 3, it outperforms the native data loader in terms of data throughput, especially on a large number of GPUs.

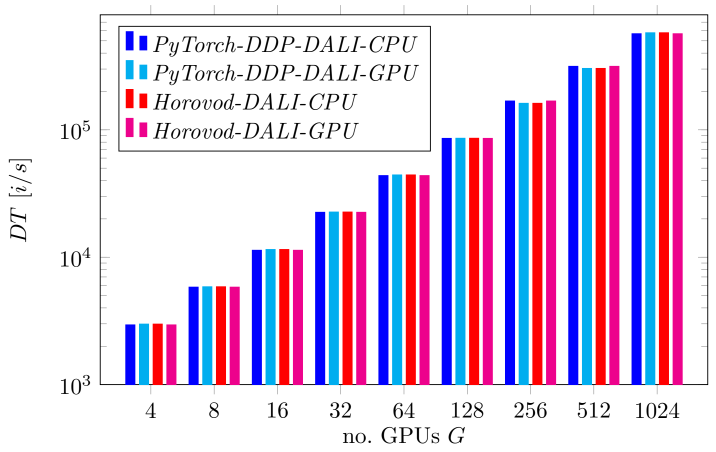


Figure 2: Throughput DT of *Horovod* and *PyTorch* with the DALI data loader CPU and GPU version on the compressed ImageNet dataset [1] in images i per second s over the number of GPUs G.

3.2 ClearML

This section describes the usage of ClearML. It is subdivided into Sec. 3.2.1, explaining ClearML and its purpose, and Sec. 3.2.2, explaining the deployment architecture.

[53] Apache Airflow https://airflow.apache.org
[54] DALI https://developer.nvidia.com/dali

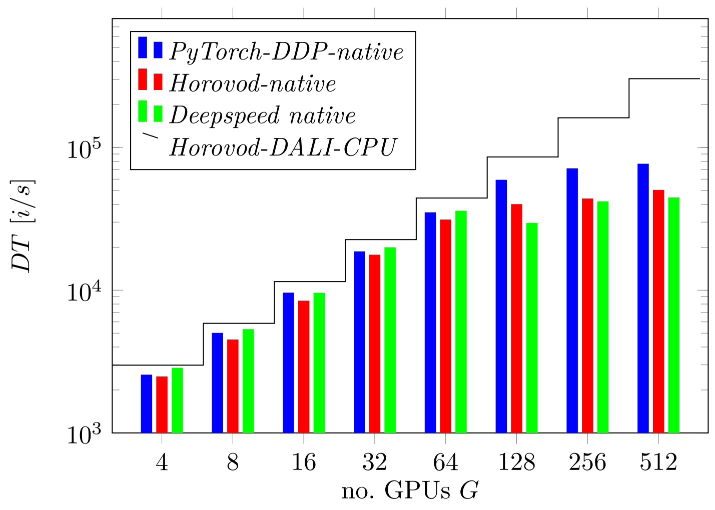


Figure 3: Throughput DT of *Horovod*, *PyTorch-DDP*, and *DeepSpeed* with the native *PyTorch* data loader on raw ImageNet dataset, including comparison with *Horovod-DALI-CPU* throughput. The largest configuration only features 512 GPUs in this case as no significant additional speed-up of the native data loader is expected on larger configurations. The quantity DT is given in images i per second s over the number of GPUs G.

3.2.1 Purpose

MLOps, a name borrowed from Development Operations (DevOps), began as a set of best practices for developing AI models, and similarly has evolved into a methodology to increase automation and improve the quality of production models. In contrast to the otherwise ad-hoc approaches in data sciences, it provides methodology in line with DevOps and data engineering to structure development processes. MLOps tools provide automation of ML/DL pipelines, enable orchestration of all tools needed for the pipelines, and offer reproducibility of the solution to be used in other use cases, the latter being especially important for good science.

ClearML is all-round MLOps software that consist of client/server components. AI model training scripts are instrumented with the client library, which logs training details to a server. Later, source code version, selected data sets, hyperparameters, runtime parameters, or the resulting learning curves and performance metrics can be inspected, reviewed, and analyzed, see Fig. 4 for a screenshot of the ClearML Graphical User Interface (GUI).

3.2.2 Deployment architecture

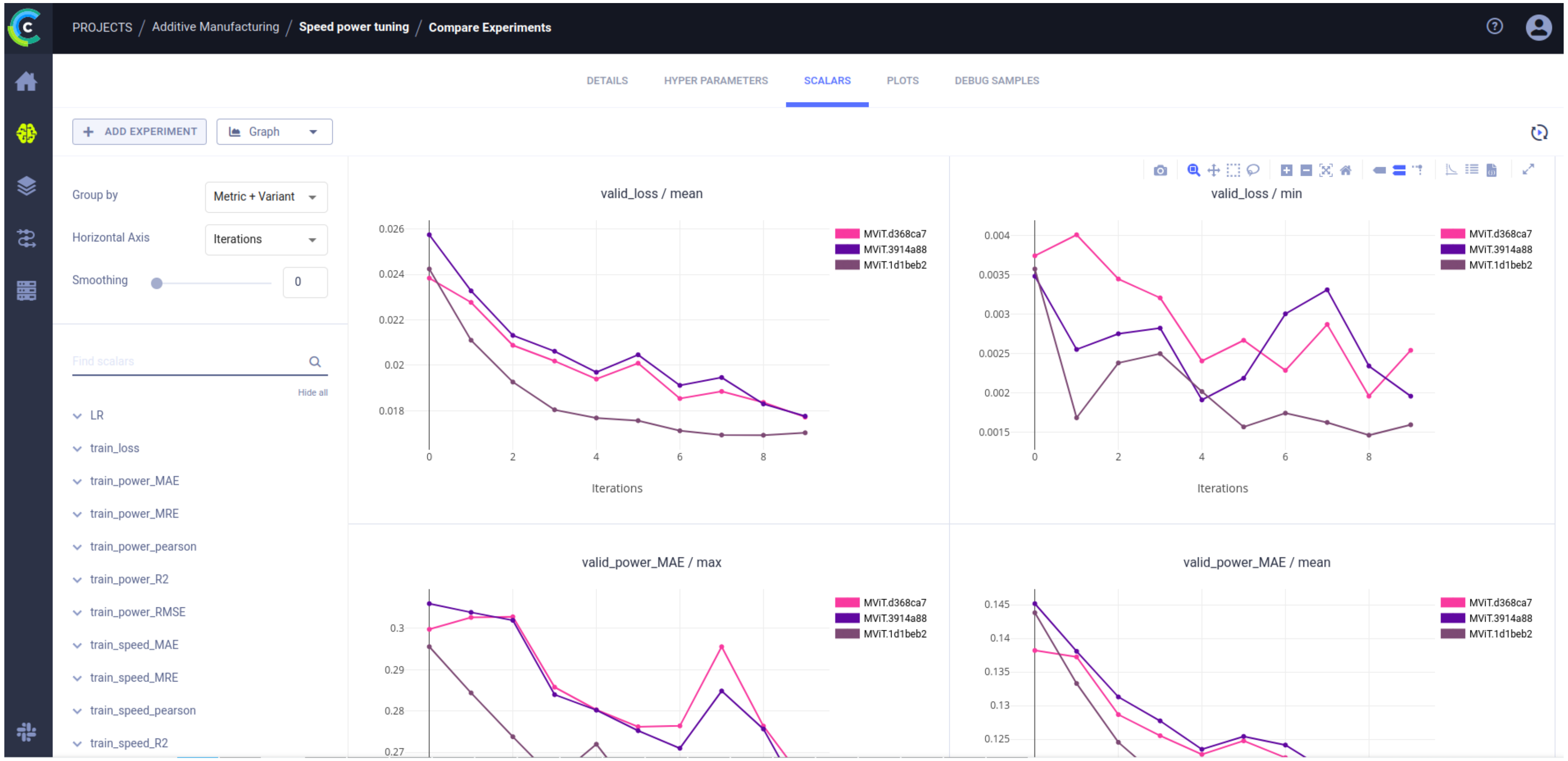
In CoE RAISE, ClearML is used in HPC environments. The client is the open source[55] Python package `clearml`. The server ("ClearML Server") is available as Software as as Service (SaaS) and as source-available[56] software for self-hosting. To achieve full control over performance and scalability aspects, the server components are deployed on a private OpenStack cloud at an HPC center, see Fig. 5. The stack can be considered part of a modular supercomputer service and is well connected to most European HPC centers via the BelNet[57] and GÉANT[58] networks.

[55]The client bears the permissive Apache 2.0 license.

[56]The license for ClearML Server is the Server Side Public License (SSPL), which is not Open System Interconnection (OSI) approved open source. SSPL is best known as the license of MongoDB since 2016. Although somewhat controversial, the terms of SSPL are not peculiarly arduous for CoE RAISE or HPC centers wanting to host it. The main condition of concern is strict copyleft that extends to maintenance and integration scripts such as backup routines.

[57]BelNet is the Belgian academic network provider https://belnet.be/en/networks

[58]GÉANT is the European high-speed research network https://geant.org/



Group by: Metric + Variant
Horizontal Axis: Iterations
Smoothing: 0

Figure 4: Comparing experimental results for variants of a Transformer model in the ClearML GUI

.

The particular implementation used by CoE RAISE consists of various containers that are hosted in a Kubernetes cluster, which in turn, runs on the Virtual Machines (VMs) provisioned by the OpenStack environment of the VSC Tier-1 cloud. Transport Layer Security (TLS) termination is implemented with a Kubernetes Nginx[59]-based ingress component instrumented with an Automated Certificate Management Environment (ACME) tool, see Fig. 5, Kubernetes box. The networking environment of Kubernetes is configured in Neutron[60], the OpenStack networking component.

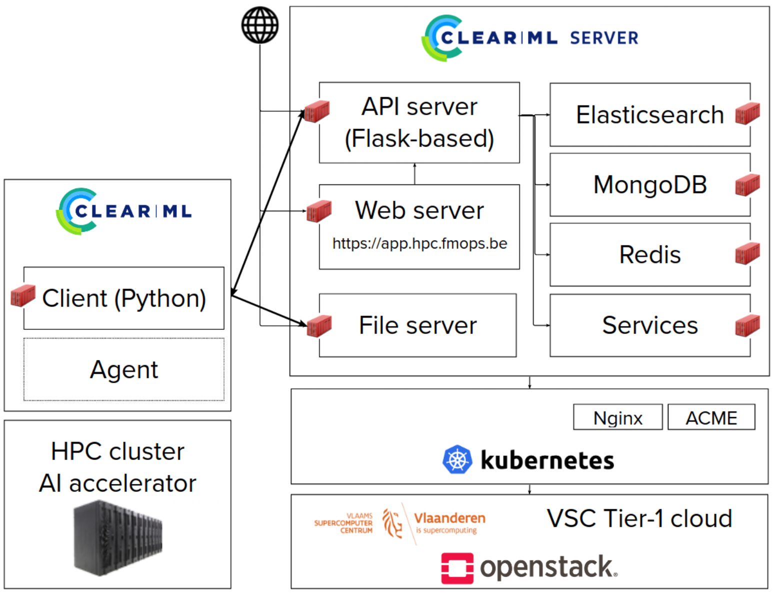


Figure 5: Architecture of CoE RAISE's implementation of ClearML on an HPC platform.

The client can run inside an Apptainer (formerly Singularity) container if the HPC center supports containerization, or on the bare OS of the HPC cluster. On some clusters, it is necessary to configure firewall and Network Address Translation (NAT) traversal rules to use this service, restricted to Hypertext Transfer Protocol Secure (HTTPS) between specific subdomains (*.hpc.fmops.be in this example).

3.3 Hyperparameter Tuning

Performance and reliability of neural networks are highly dependent on the selection of hyperparameters. Finding the optimal setting of hyperparameters is a notoriously tough task, and is often performed manually. Hyperparameters are either related to model the architecture, such as the number of layers or neurons, or related to the optimizer such as the learning rate and batch size. Designing neural network architectures manually can be time-consuming and prone to human bias. Automated Neural Architecture Search (NAS) can lead to discovering non-obvious architectures that yield better results in multiple tasks [12]. Finding the optimal combination of hyperparameters is often impractical as the the search space is too large to explore exhaustively, and thus is limited by computational resources. Recent developments in HPO can overcome this limitation by using scalable and efficient scheduling methods. In the following, two methods for HPO used in CoE RAISE are presented, i.e., Ray Tune in Sec. 3.3.1 and DeepHyper in Sec. 3.3.2.

[59]Nginx https://www.nginx.com
[60]OpenStack Neutron https://docs.openstack.org/neutron/latest/

3.3.1 Ray Tune

One of the most common libraries used to perform HPO is the open-source library Ray[61]. Its sub-package Ray Tune can run distributed hyperparameter tuning at scale. As HPO involves running a lot of trials with different sets of hyperparameters, allocating and launching each trial manually can be costly. With Ray Tune, only a head node needs to be launched and all the worker nodes via a SLURM script. The head node then connects to the worker nodes and launches the trials. During training, the worker nodes report their current status, including performance metrics, to the head node, which can then make informed decisions on terminating low performing trials or launches new ones. The user specifies the number of resources to use per trial, the hyperparameters and their range, and a scheduling or optimization algorithm. The head node takes care of communication and scheduling tasks.

Ray is compatible with all common ML frameworks, including TensorFlow and PyTorch. It also supports distributed DL libraries like Horovod and PyTorch-DDP. This way, two levels of parallelism can be used during the HPO process, see Fig. 6. That is, first, the training of the models runs in data-parallel fashion, e.g., one model is trained across four GPUs on a node using Horovod. Second, the trials are distributed across different nodes, e.g., on the GPU partition of the Jülich Research on Exascale Cluster Architectures (JURECA)-DC at JSC, 24 trials run in parallel on 24 nodes using Ray Tune. These two levels are necessary to accelerate the HPO process, as the models and the data-sets they are trained on are usually extremely large.

For running HPO at large scale, it is crucial to use efficient algorithms to avoid the waste of compute resources. The three most commonly used HPO algorithms are:

1. Bayesian Optimization Hyperband (BOHB) [13];
2. Asynchronous Successive Halving Algorithm (ASHA) [14];
3. Population Based Training (PBT) [15].

BOHB and ASHA are both based on the Hyperband algorithm [16], which seeks to approximate the performance of a hyperparameter configuration by evaluating it on a smaller budget, e.g., by running a trial with fewer epochs. Worse performing trials are cut early and only the best-performing trials are kept. This procedure is known as successive halving [17]. In BOHB, the Hyperband al-

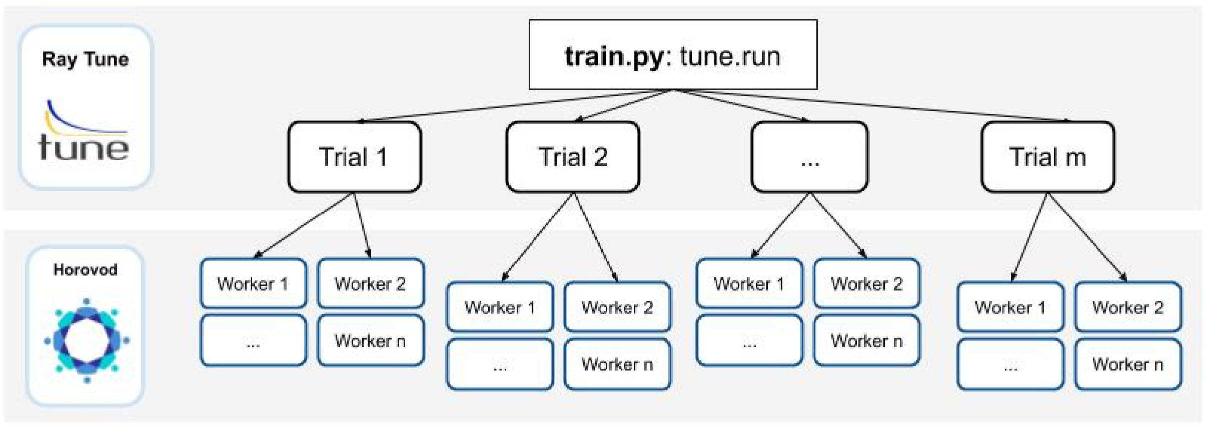


Figure 6: Two levels of parallelism in distributed HPO. The single model training runs distributed across workers with Horovod while different trials run in parallel with Ray Tune. Image taken from the Horovod website.

[61] Ray https://www.ray.io/

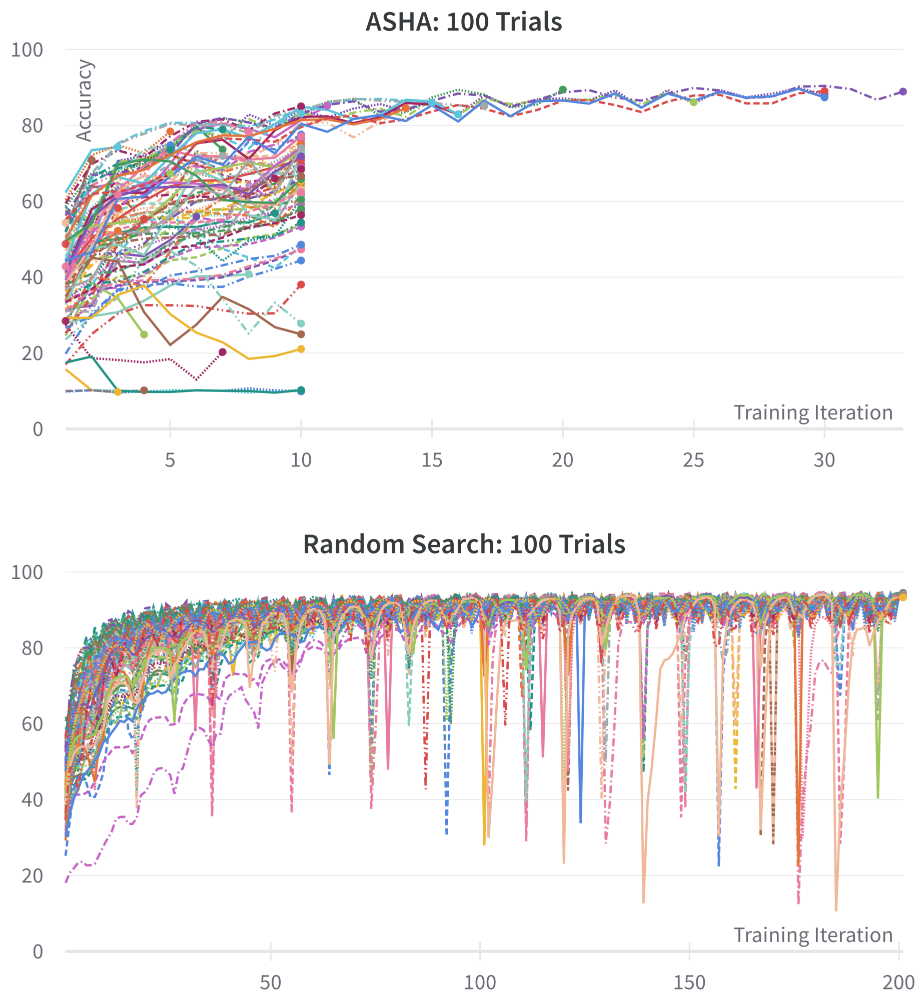


Figure 7: Comparison of the accuracy over the number of epochs (training iteration) for the ASHA algorithm (top), where under-performing trials are stopped early, to a simple random search algorithm (bottom), where every sampled configuration is fully trained on cifar-10. In the ASHA case, one training iteration corresponds to six training iterations in the random search case. 100 trials are evaluated in parallel on the JURECA-DC-GPU partition.

gorithm chooses the number of hyperparameter configurations and their assigned budget. Bayesian Optimization (BO) is used to choose the values for the hyperparameters by deploying a tree parzen estimator [18].

ASHA addresses the problem of large-scale HPO by improving the scalability of the Hyperband method. To assess the most promising trials, the Hyperband algorithm waits for *all* trials to reach a certain threshold in time before applying successive halving. This leads to idling workers as some will be faster than others. ASHA circumvents this by deciding on a rolling basis which trials are promising. When two trials reach the time barrier, the trial with the better performance is continued while the other is paused until the performance of the next completed trial can be juxtaposed. This asynchronous comparison leads to large speed-ups. A comparison of ASHA to a simple random search method is shown in Fig. 7. The goal is to optimize the number of layers, filters, batch size, learning rate, and momentum of the training of a small Convolutional Neural Network (CNN) on cifar-10[62]. It

[62]cifar-10 https://www.cs.toronto.edu/~kriz/cifar.html

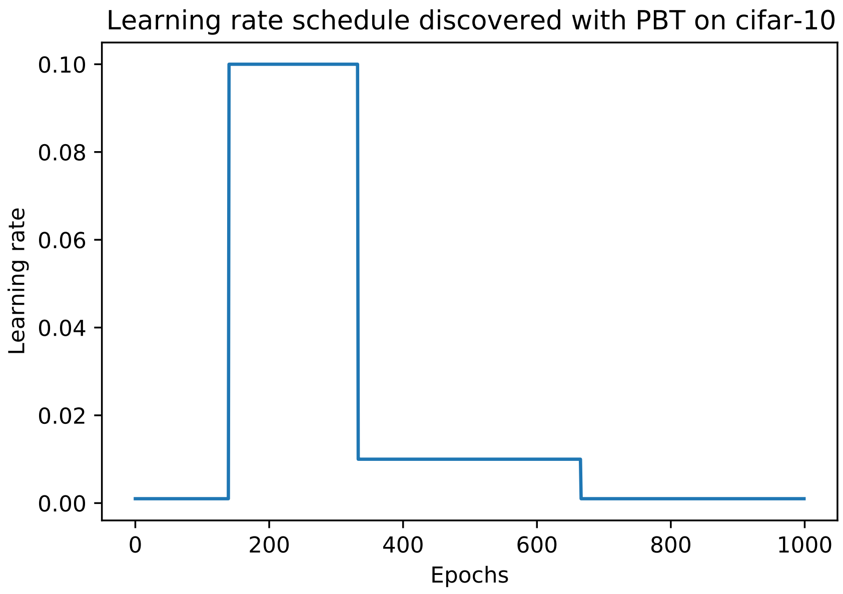


Figure 8: Example of using the PBT algorithm to discover an optimal learning rate schedule over the number of epochs for a CNN trained on cifar-10.

can be seen that ASHA successfully terminates the trials that do not show good performances and only runs the top performing ones to the end. In conctrast, random search completely calculates all sampled trials, even the ones that exhibit poor performance, and therefore does not use the computing resources efficiently.

Evolutionary methods in HPO such as PBT seek to mimic the process of evolution and natural selection by using Genetic algorithms for finding optimal hyperparameters. At the beginning, an initial population of different ML models with randomly sampled hyperparameters is initialized and trained for a few epochs (a generation). Subsequently, the performance is measured and the models are ranked according to their results. In the next step, different genetic operations such as mutations are applied. In the case of mutation, the worst performing trials copy the state and hyperparameters of the best performing models and apply small perturbations to these parameters. This way, only the best-performing models continue training. The worst-performing ones are early-stopped and their resources are reassigned to the perturbed configurations. As multiple models are trained in parallel and the performance evaluation only takes place at certain points in time, the framework is highly scalable. Due to the iterative nature of the optimization process, the framework is suitable for discovering unsteady series of hyperparameters, e.g., learning rate schedules. An example is shown in Fig. 8, where PBT is used to find an optimal learning rate schedule for training a CNN on cifar-10.

In CoE RAISE, Ray Tune and its HPO algorithms have been successfully used across different use cases to improve the performance of ML models:

- In the use case of Task 3.1 (AI for turbulent boundary layers), the hyperparameters of a Convolutional Autoencoder (CAE) have been analyzed and ranked according to their significance. As the learning rate was found to be the most important parameters, several approaches for optimizing it were explored.
- In the use case of Task 4.1 (Event reconstruction and classification at the CERN HL-LHC), a Graph Neural Network (GNN) for particle reconstruction in a High-Energy Physics (HEP) domain was improved by the use of the ASHA and BOHB algorithms.

- In the use case of Task 4.2 (Seismic imaging with remote sensing for energy applications), the ASHA algorithm was deployed to optimize the hyperparameters of a Transformer (TF)-based ML model for the prediction of Land Cover (LC) change, based on time-series satellite image data. The model was able to achieve a validation accuracy of 96% and, as depicted in Fig. 9, the speed-up is close to linear, even when scaled to a large numbers of GPUs. Therefore, ASHA has been verified as a suitable choice for this use case running HPO on supercomputers.

Beyond internal use cases, the Ray Tune component of the UAIF has been adopted by use cases outside of CoE RAISE, e.g. in the medical field [19].

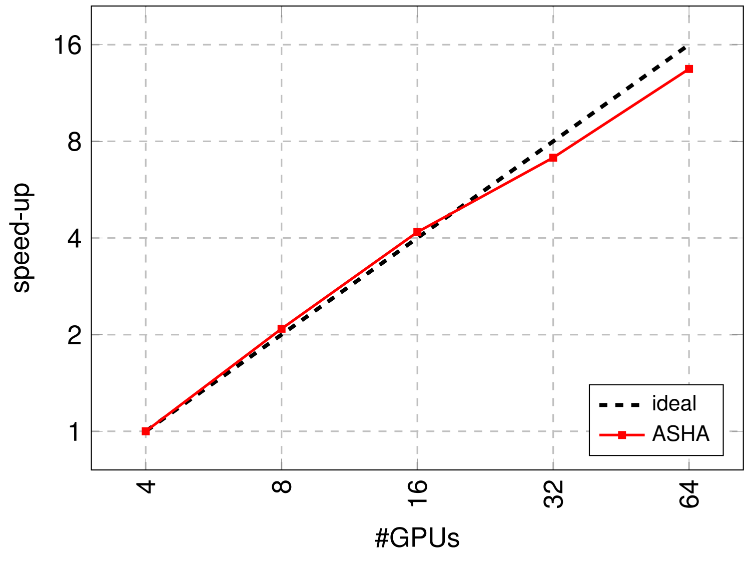


Figure 9: Scalability of ASHA, optimizing the hyperparameters of the TF model from Task 4.2 of CoE RAISE. Each trial runs on one node (four GPUs of the GPU partition of JURECA-DC and the time to evaluate 50 trials is measured.

3.3.2 DeepHyper

DeepHyper[63] is another distributed ML package for automated development of deep neural networks. It adopts an asynchronous BO approach for HPO and NAS at HPC scale. The BO algorithm aims to fit a dynamically updated cheap-to-evaluate surrogate model to identify promising regions in the search space. High-performing hyperparameters configurations are eventually found by exploitation and exploration of the search space. In addition, DeepHyper's automated deep ensemble for Uncertainty Quantification (UQ) evaluates the model reliability by automatically generating a catalog of neural network models through joint neural architecture and hyperparameter search, and selecting a set of high-performing models to construct the ensembles, and estimating aleatoric and epistemic uncertainties from the generated ensembles.

DeepHyper is comprised of three modules:

- `deephyper.problem`: tools for defining hyperparameter search problems;
- `deephyper.evaluator` : interface to dispatch model evaluation tasks;
- `deephyper.search`: search methods.

[63] DeepHyper: https://deephyper.github.io

The basic workflow for DeepHyper is to first define a black-box function $f(x)$ to be optimized, which represents the performance of the network. Then, the search space as input variables to the black-box function can be defined through the `deephyper.problem` module. An `evaluator` object wraps the black-box function, distributes the computation, and adapts to different back-ends, e.g., to the Message Passing Interface (MPI) or Ray. The final step is to define the search algorithm. Centralized Bayesian Optimization (CBO) search in DeepHyper is currently used in CoE RAISE, which has the advantage of being asynchronous to keep a good utilization of the resources when the number of available workers increases as a trial has finished and the worker become idle.

DeepHyper is being applied to a new use case in medical imaging[64]. Best-practice for using DeepHyper will be compiled and shared in upcoming publications, highlighting performance, scalability, and accuracy, as well as comparing DeepHyper to similar tools such as Ray Tune.

3.4 Basic Science Libraries: NumPy and scikit-learn

This section covers some of the common Python libraries used in CoE RAISE for ML and data analysis tasks. For brevity, only the NumPy[65] and scikit-learn[66] packages are decribed in the following Sec. 3.4.1 and Sec. 3.4.2.

3.4.1 NumPy

NumPy is a general-purpose open source Python package for scientific computing primarily supporting multi-dimensional arrays and matrices and including high-level mathematical functions to operate on these arrays. NumPy as a fundamental package is a dependency for many higher-level Python packages used in CoE RAISE including scikit-learn, PyTorch, TensorFlow. It is used in many project use cases, for example, using its powerful library support working with arrays (see below).

The main data type of NumPy is `ndarray` representing a multi-dimensional array object and providing fast array-oriented arithmetic operations, as well as flexible broadcasting capabilities. The NumPy arrays have the following benefits:

- efficient storage and data operations;
- ability to perform complex computations on large blocks of data.

NumPy provides various methods for working with arrays:

- indexing and slicing;
- sorting and reshaping;
- combining and splitting;
- adding and removing elements;
- descriptive statistics.

3.4.2 scikit-learn

scikit-learn represents a set of simple and efficient tools for predictive data analysis and ML. scikit-learn contains algorithms and models for classification, regression and clustering, as well as for supervised and unsupervised learning. The package is extensively used, for example, in WP3 Task

[64] HPO use case with DeepHyper: https://jlesc.github.io/projects/hyperp_sr_project/
[65] NumPy https://numpy.org
[66] scikit-learn https://scikit-learn.org

3.2 on BSC and RTU HPC clusters for wind turbine simulation and ML model data preprocessing, as well as for the accuracy estimation of the results.

The main functional components of scikit-learn are the following:

- *Supervised learning algorithms* - popular supervised learning algorithms, for example, linear regression, SVM, decision trees.
- *Unsupervised learning algorithms* - popular unsupervised learning algorithms, for example, clustering, factor analysis, Principal Component Analysis (PCA).
- *Clustering* – grouping unlabeled data such as KMeans.
- *Cross validation* – methods for estimating the performance of supervised models on unseen data.
- *Dimensionality Reduction* – methods for reducing the number of attributes in data for summarization, visualization, and feature selection.
- *Ensemble methods* - for combining the predictions of multiple supervised models.
- *Feature extraction* - methods for defining attributes in image and text data.
- *Feature selection* - methods for identifying meaningful attributes from which to create supervised models.

3.5 DeepSpeed

Data-distributed training is a simple but powerful approach to accelerate ML training, where the input dataset is distributed to separate GPUs, and the trainable parameters between the GPUs are exchanged occasionally. DeepSpeed [20], developed by Microsoft, is one of many open-source frameworks that includes such data-distributed training algorithms. It targets resource-effectiveness and memory-efficiency. This framework is designed to train large distributed models with better parallelism on existing computer hardware. DeepSpeed is optimized for low-latency and high-throughput training. It includes the Zero Redundancy Optimizer (ZeRO)[67] for training large models with 100 billion or more parameters, which is especially useful for CNNs and Natural Language Processing (NLP) applications. DeepSpeed extends the capabilities of PyTorch by taking care of the distribution of the dataset to different workers, e.g., CPUs, GPUs, or Intelligent Processing Units (IPUs), and the update of network parameters (gradients and weights) between these workers.

DeepSpeed was tested on various heterogeneous HPC systems available to CoE RAISE. Initially, this framework was ported to two prototype systems, i.e., to the DEEP and CTE-AMD systems at Forschungszentrum Jülich GmbH (FZJ) and BSC. With these systems, the performance of DeepSpeed was tested for various hard- and middlewares, including NVIDIA/AMD GPUs running Compute Unified Device Architecture (CUDA)[68]/Radeon Open Compute platforM (ROCm)[69] frameworks. These findings have been made available in Deliverables D2.6 and D2.7. With the knowledge gathered from these prototype systems, DeepSpeed was then deployed on JURECA, with no additional complications. Full details of porting and testing activities can be found in Deliverable D2.2.

Performance results of DeepSpeed are depicted in Fig. 10. Briefly, the first analysis of DeepSpeed has shown that the data distribution and network update of the DeepSpeed framework greatly resemble those of the original PyTorch-DDP. The lack of improvement from DeepSpeed likely stems

[67] ZeRO https://www.deepspeed.ai/tutorials/zero/
[68] CUDA https://developer.nvidia.com/cuda-toolkit
[69] ROCm https://www.amd.com/en/graphics/servers-solutions-rocm

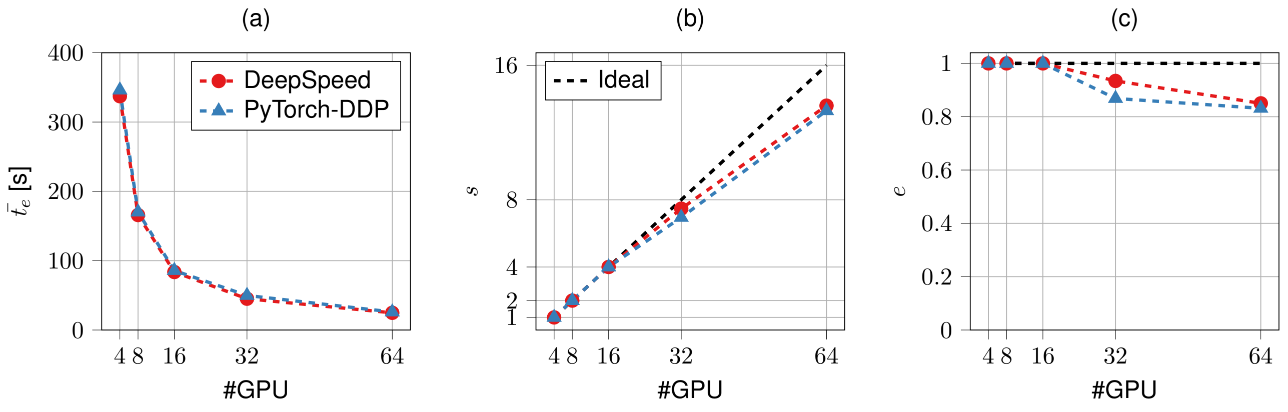


Figure 10: Performance of distributed training with DeepSpeed vs. PyTorch-DDP API on JURECA. Depicted are average epoch time $\bar{t_e}$ (a), speed-up s (b), and efficiency e (c). The black dashed line represents the ideal scenario (perfect scaling).

from the shared data and parameter distribution algorithms of PyTorch-DDP. It could be that DeepSpeed is developed to train exceptionally large ML models, which require model-distributed parallelism, i.e., it necessitates to distribute parts of the model to different GPUs and communicate the gradients/weights between all GPUs. Therefore, the developers of DeepSpeed might have focused solely on model-distributed parallelism and used the standard implementation for data-distributed parallelism from PyTorch-DDP. This likely assumption comes from the fact that DeepSpeed has only been used for model-distributed parallelism in the literature, see, e.g., the work by Rasley et al. [20]. Thus, any improvement to the standard PyTorch-DDP algorithms should not be expected.

These findings suggest to rely solely on DeepSpeed if model-distributed parallelism is required, noting that such a use case with a large ML model is at present not part of CoE RAISE. That is, the lack of speed-up indicates that PyTorch-DDP is already sufficient for the use cases in CoE RAISE that already benefit from PyTorch-DDP's data-distributed parallelism. Nevertheless, it was important to understand the performance and applicability of DeepSpeed to ML training and to make the software available to others with potentially more suited applications on JSC's and BSC's machines.

3.6 LAMEC API

Dependency management on HPC systems can be a source of frustration for many users. Software updates can cause unexpected and cryptic errors, sometimes taking hours or even days to fix. Therefore, having an automated way to deal with dependency issues could save a significant amount of working hours on a yearly basis. HPC systems rely heavily on module environments to load software packages and libraries for users. It is with these module commands, found practically in all HPC jobscripts, that dependencies tend to break: an unannounced or silent update is made to the module environment that breaks compatibility and results in errors. Having a tool that generates these module commands automatically, based on the software libraries that users need, could be a first step in combating this problem.

Load AI Modules, Environments, and Containers (LAMEC) is a Python API and command-line tool that generates SLURM scripts to start jobs on HPC systems, leveraging the SLURM workload manager. Two basic components have been implemented in LAMEC: dependency resolution and resource allocation. First, based on the software and system being used, commands are generated to load modules with the correct dependencies. This is done by extracting information from a Gitlab repository[70] developed as a part of the UAIF, which contains example start scripts for using mainstream ML frameworks on multiple HPC systems. The Gitlab repository is constantly maintained so it can provide up-to-date dependency information for the LAMEC API. Second, SLURM commands are generated to start jobs with the required allocation. Currently, the LAMEC API batch script generator has support for JUWELS, JURECA, and the DEEP system at FZJ, and the following software frameworks and libraries:

- DeepSpeed (DEEP, JURECA)
- PyTorch-DDP (All)
- Helmholtz Analytics Toolkit (HeAT)[71] (DEEP, JURECA)
- Horovod (DEEP, JURECA)
- Ray Tune (JURECA)
- TensorFlow (DEEP).

The above mentioned ML frameworks are widely available on HPC systems and have all been benchmarked and evaluated in CoE RAISE. The reader can refer to previous sections for application of PyTorch-DDP, TensorFlow, Horovod, Ray Tune 3.3 and DeepSpeed 3.5 in CoE RAISE. HeAT is mainly in developed in the Helmholz Association and is suited for distributed and high-performance data analytics.

The command-line tool is simple and has two subcommands: `lamec gen` and `lamec eval`. The `gen` sub-command allows users to generate module commands and prints to standard output.

```
$ lamec gen deep ddp

# MODULES BEGIN deep ddp heat
ml -force purge
ml use $OTHERSTAGES
ml Stages/2022 GCC/11.2.0 OpenMPI/4.1.2
cuDNN/8.3.1.22-CUDA-11.5 NCCL/2.11.4-CUDA-11.5
Python/3.9.6
# MODULES END
```

A module block begins with `#MODULES BEGIN`, followed by the HPC system and a space-separated list of software libraries exposed through modules, and ends with `#MODULES END`. These module blocks serve a purpose when used with the `eval` sub-command, as it evaluates the listed HPC system and software libraries, and if there has been a change to the module environment, the block will be updated. In addition, `eval` allows adding and removing software, as well as updating the module block to be used on a different HPC system, e.g., transitioning from DEEP to JURECA. By default, changes are printed to standard output but can also be written to a file.

The method for the LAMEC API to generate module commands with the correct dependencies requires that the Gitlab repository be synchronized with the software packages installed on HPC systems. Currently, this requires manual intervention from a maintainer, however, communication systems are under development for automated reporting of module changes and bug reports from users.

[70] AI for HPC repository https://gitlab.jsc.fz-juelich.de/CoE-RAISE/FZJ/ai-for-hpc
[71] HeAT http://www.helmholtz-analytics.de/helmholtz_analytics/EN/GenericMethods/HeAT/_node.html

The second component in the LAMEC API is resource allocation. SLURM manages clusters by allocating access to computing nodes for users with a certain duration of time. It provides tooling for users to allocate, submit, and monitor jobs, as well as monitor queuing of pending jobs. The user can take the output from the LAMEC API and manage jobs in batch systems using SLURM commands:

- `sbatch` – submit a job script;
- `squeue` – check status of jobs;
- `scancel` – delete job from the. queue

The following is an example of a SLURM script generated by LAMEC for using DDP on JURECA's GPU partition:

```
#SBATCH -job-name=testjob
#SBATCH -account=raise-ctp2
#SBATCH -output=test.out
#SBATCH -error=test.err
#SBATCH -time=01:00:00
#SBATCH -partition=dc-gpu
#SBATCH -nodes=2
#SBATCH -gpus-per-node=4
#SBATCH -ntasks-per-node=1
#SBATCH -cpus-per-task=32

srun ./executable.x
```

Among the above SLURM parameters, the `gpus-per-node`, `ntasks-per-node` and `cpus-per-task` are not straightforward to determine, and are dependent on the system and software utilized. Given a system and partition, LAMEC sets reasonable default values as suggestions for users. For example, the default value of `gpus-per-node` is set depending on hardware settings: `4` for JURECA's GPU partition and JUWELS booster, `1` for the DEEP system. `ntasks-per-node` is dependent on software requested. It is equal to `gpus-per-node` for Horovod and HeAT, but set to `1` when using DDP, Ray Tune, and DeepSpeed. On JURECA's GPU partition, `cpus-per-task` is set to `32` because there are 2*64 core CPUs and 4 GPUs per node. Therefore, `cpus-per-task` is best set to the total number of CPU cores divided by `gpus-per-node`. Currently, LAMEC is being developed on FZJ's HPC systems and soon will be extended to systems on other partner sites, including CTE-AMD at BSC and PizDaint at CSCS.

The LAMEC API is currently available as a command-line tool, a web form interface is being developed and shall be made available through the service portal on CoE RAISE's website[72] and FZJ's APPS server for better visibility. The APPS server at FZJ is a content management system to host web applications for science or administration. The reader can refer to an existing tool hosted on APPS server as example[73]. In the web form interface currently under development, users can select the HPC system and software they want to use from a drop-down list and modify input job configuration data such as the account name, wall time, or the executable. By clicking a button, a start script will be generated in a text area, which can simply be copied.

[72] CoE RAISE Service Portal https://www.coe-raise.eu/services
[73] Pinning tool on APPS server https://apps.fz-juelich.de/jsc/llview/pinning/

3.7 Google JAX

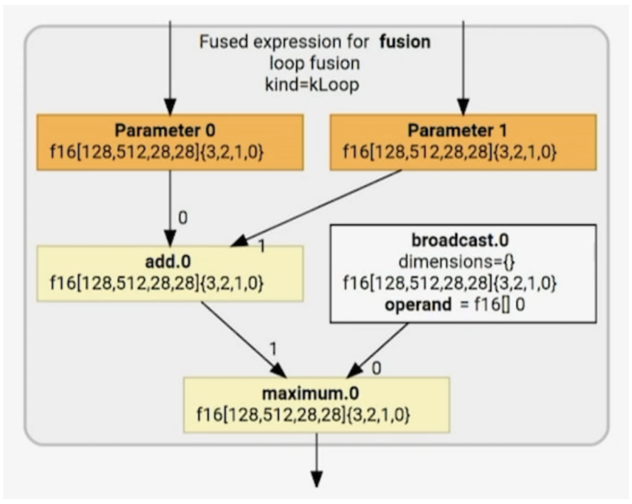
Google JAX[74] is an ML framework for transforming numerical functions. It brings together a modified version of Autograd[75] (automated obtaining of the gradient function through differentiation) and TensorFlow's Accelerated Linear Algebra (XLA)[76]. It is designed to follow the structure and workflow of NumPy, see Sec. 3.4, as closely as possible and works with various existing frameworks such as TensorFlow and PyTorch. The primary functions of JAX are:

- `grad`: automatic differentiation;
- `jit`: just-in-time compilation;
- `vmap`: auto-vectorization;
- `pmap`: Single-Program Multiple-Data (SPMD) programming.

JAX provides two different APIs for ML and simulation:

- `jax.numpy`: lightweigth numpy-like API, build to parallelize numpy operations by simply replacing the import;
- `jax.lax`: stricter API, for tailored operations.

JAX is built on the XLA library, which works by producing an Intermediate Representation (IR) code of a computational graph. Then, XLA's compiler applies optimizations on this IR code to build kernels tailored for the specific hardware architecture. More specifically, it fuses kernels whenever possible in order to avoid data copies and buffering by fusing multiple operations inside the same vectorization loops. Figure 11 shows a kernel fusion example including a code block, where the image was taken from slides of the 2019 European LLVM Developers Meeting[77].


Fused expression for fusion
loop fusion
kind=kLoop
Parameter 0
f16[128,512,28,28]{3,2,1,0}
Parameter 1
f16[128,512,28,28]{3,2,1,0}
0
1
add.0
f16[128,512,28,28]{3,2,1,0}
broadcast.0
dimensions={}
f16[128,512,28,28]{3,2,1,0}
operand = f16[] 0
1
0
maximum.0
f16[128,512,28,28]{3,2,1,0}

```
__global__ void fusion(float *lhs,
    float *rhs, float* output) {

  int i = blockIdx.x * blockDim.x +
          threadIdx.x;

  if (i < 128*512*28*28) {
    output[i] = max(0.0, lhs[i] + rhs[i]);
  }
}
```

Figure 11: Example of kernel fusion: fusing multiple operations inside the same vectorization loop.

.

[74]Google JAX https://github.com/google/jax
[75]Autograd https://github.com/hips/autograd
[76]XLA https://www.tensorflow.org/xla
[77]Slides 2019 European LLVM Developers Meeting https://llvm.org/devmtg/2019-04/slides/TechTalk-Joerg-Automated_GPU_Kernel_Fusion_with_XLA.pdf

Algorithm 1 Snippet of code comparing JAX to numpy.

```
from jax import jit
import jax.numpy as jnp

def selu_np(x, alpha=1.67, lambda=1.05):
  return lambda np.where(x > 0, x, alpha * np.exp(x)-alpha)

def selu_jax(x, alpha=1.67, lambda=1.05):
  return lambda jnp.where(x > 0, x, alpha * jnp.exp(x)-alpha)
```

NumPy vs. JAX	Compute time [ms]	Speed-up
NumPy on CPU	7.6	-
JAX on CPU	4.8	1.58
JAX on GPU w/o Just-In-Time Compilation (JIT)	1.21	6.28
JAX on GPU w/ Just-In-Time Compilation (JIT)	0.13	58.46

Table 1: JAX speed-up compared to NumPy.

JAX requires functionally-pure procedures to be compiled into low-level kernels. That is, JAX can compile procedures that are:

- stateless;
- without Input/Output (IO), random generators...

In the case of NumPy codes, this imposes a few constraints:

- tensors should be immutables and modified by setters, e.g., by using `y.at[0].set(1)`;
- slicing assignments are forbidden, e.g., `a[a < 0] = ...`.

JAX unifies different hardware since it runs seamlessly across CPUs, GPUs, and Tensor Processing Units (TPUs). It can scale easily over hundreds of cores. As part of CoE RAISE's Task 3.3 (AI for data-driven models in reacting flows) and Task 3.4 (Smart models for next-generation aircraft engine design), BULL investigated JAX as a framework working both for Computational Fluid Dynamics (CFD) simulations and ML. Most DL frameworks are based on Python, while CFD solvers are usually based on `Fortran` or `C/C++`. This creates compatibility issues when trying to couple both. JAX helps to alleviate this issue by having a NumPy-like framework to solve partial equations and ML derivatives like shown in Alg. 1. Table 1 shows the speed-up that one can have when implementing JAX, also with and without Just-In-Time Compilation (JIT).

At the time of writing this Deliverable, this component is not included in the UAIF, but investigation continues and potential addition of this component will be evaluated in the next software layout plan iteration.

3.8 Quantum Support Vector Regression

Quantum machines are novel hardware devices that exploit the quantum mechanical properties of matter to perform computations in a faster and more energy-efficient way. For the UAIF, it is planned

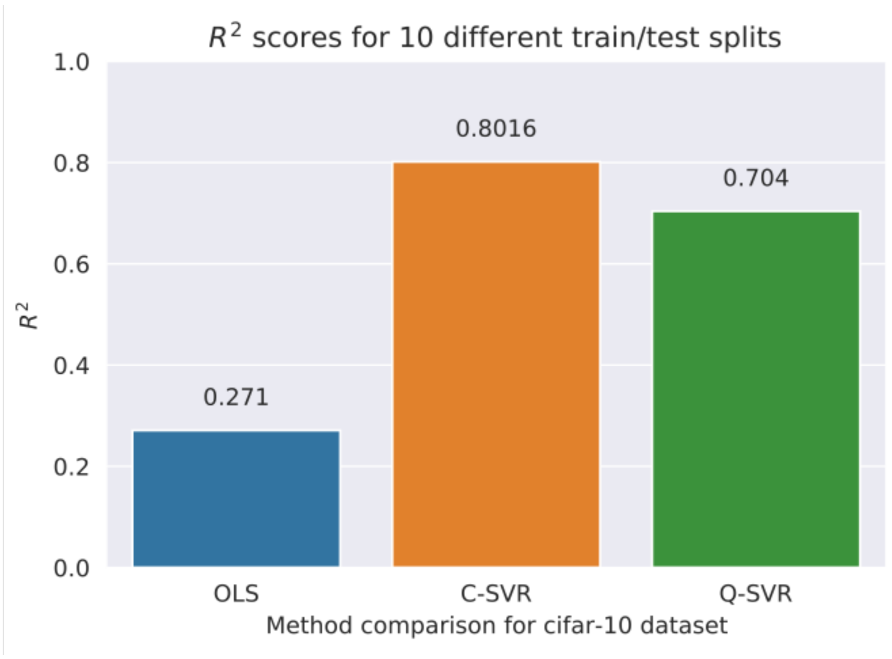


Figure 12: Comparison of regression methods for predicting the learning curves of neural networks trained on cifar-10.

to provide methods that researchers can directly use to run their problems on a D-Wave QA. In terms of algorithms, the focus has been on the Quantum Support Vector Regression (Q-SVR) method. This method takes as input a set of samples formed by a pair of one or more independent variables and one dependent variable. The objective of the training phase is to learn the relationship between the independent and dependent variables. The training phase amounts to an optimization problem in which the objective is the determination of the coefficients that define the estimated regression function.

The original formulation to run Q-SVR on a QA was proposed in [9]. Specifically, the QA was used for the optimization problem related to the training phase of the SVR. To be solved by the QA, the problem must be reformulated as a Quadratic Unconstrained Optimization Problem (QUBO), in which each variable can take its value from a binary set and the terms of the equation are either linear or quadratic. The reformulation of the original optimization problem to a QUBO problem is carried out by applying an encoding procedure to the original problem variables that are intrinsically continuous, to turn them into discrete variables. The set of possible values that the encoded variables can take depends on the encoding hyperparameters that are selected by the user. The QA returns as output a set of solutions whose number is selected by the user. To obtain the final solution, the individual solutions are combined. A first benchmark of the performance of Q-SVR in comparison to classical SVR (C-SVR) and linear regression (OLS) is depicted in Fig. 12. The task of the benchmark was to extrapolate the learning curves of different neural networks trained on cifar-10.

The results show that the Q-SVR algorithm does achieve R^2 scores above 70%, but is still outperformed by classical SVR by roughly 10%. Therefore, more research into the stability of the predictions of the QA is required. The code used for performing the calculations will be available open-source soon.

3.9 Workflow with Apache Airflow

The workflow used in CoE RAISE is based on a modified version of the classification system proposed by Paris *et* al. [21]. It is based on a scalable and parallelizable, tile-based approach [22] designed for the fast and efficient production of LC maps at the large scale. It can automatically update the LC maps using available Remote Sensing (RS) images and ML and DL models based on parallel and scalable algorithms. In this version of the workflow, a DL TF model is considered as a classifier [23]. The entire workflow can be run on an HPC system, enabling the rapid generation of LC maps at a large scale, such as country, continental, or at a global level.

RS data can be complex to process due to its large volume, diverse sources and formats, and scale of physical area with asynchronous or unreliable reporting. Challenges include data storage and management, data integration and harmonization, and processing data for different applications. These factors can make processing RS data more complex, particularly for large-scale applications. To address these needs it is necessary to develop processing workflows using parallel algorithms that can scale on heterogeneous and HPC technologies, including HPC platforms, clusters and clouds, and hardware accelerators such as GPUs.

Workflows are a way of organizing and automating a series of computational and data manipulation steps. They can be represented visually as a series of building blocks, and their formalization allows for enhanced reuse and portability of processes. Workflow managers are tools that help to design and execute workflows, and they can optimize processing through conditional rules such as scheduling and parallelization. However, designing and implementing workflows often requires specialized training and expertise. Currently, there are challenges in building scalable workflows due to increasing data volumes and demands, and the need for portability and distributed workflows.

Apache Airflow[78] is a workflow manager to programmatically author, schedule, and monitor workflows. It uses a message queue to orchestrate the workers and has a modular architecture, which facilitates scalability and extensibility. Moreover, tasks and dependencies may be defined using Python. This allows users to maintain full flexibility in a familiar language when building workflows. Figure 13 shows an example on how to integrate Apache Airflow into HPC systems.

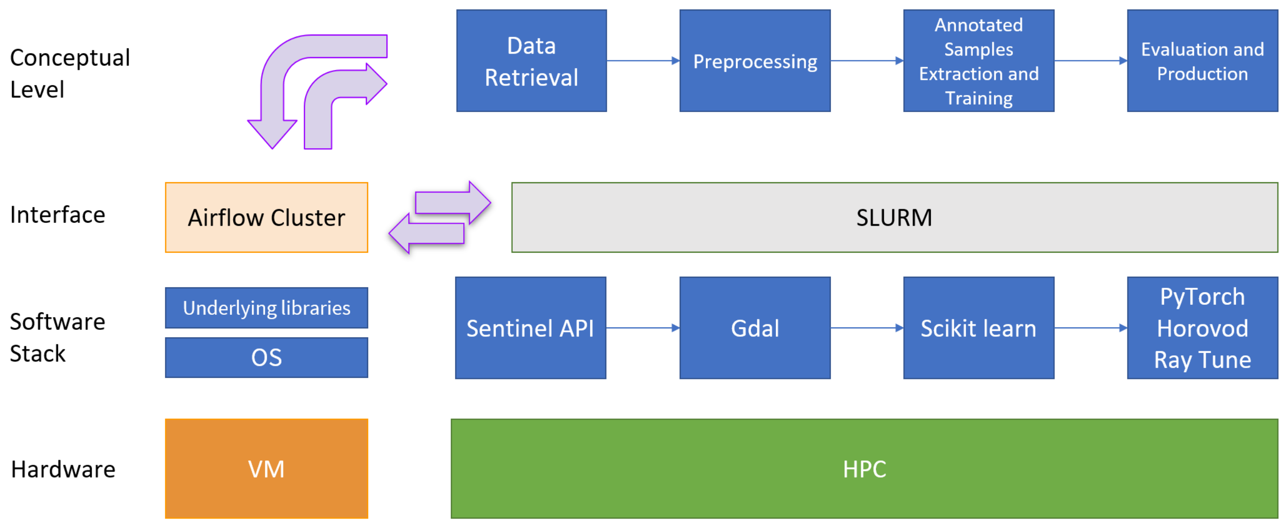


Figure 13: How Apache Airflow can be integrated into HPC system.

[78]Apache Airflow https://airflow.apache.org

Algorithm 2 shows how Apache Airflow manages the individual scripts. After the tasks are defined, an SSH connection to HPC is built up and a batch script can be executed. By using Airflow, the management and execution efficiency of the workflow can be enhanced.

Algorithm 2 Example of how Apache Airflow-based workflows are created.

```
# This bash script is executed on the HPC system. It accesses the folder, where
   the script is found and then executes it. It exits once the job fails.
task_ssh_bash """
pwd
cd /p/project/deepacf/deeprain/kreshpa1/test &&
JID=$(sbatch run_script.sh)
echo $JID
sleep 10s
ST="PENDING"

while [ "$ST" != "COMPLETED" ]; do
   ST=$(sacct -j ${JID##* } -o State | awk 'FNR == 3 {print $1}')
   sleep 1m

    if [ "$ST" == "FAILED" J; then
        exit 122
    fi

echo $ST
done
"""

# Instantiating a DAG by passing to it a dag_id which serves as unique
   identifier.
dag = DAG(dag_id='test1'
        default_args=default_args,
        dagrun_timeout=timedelta(seconds=120))

# Test case: Instantiating an SSHOperator which creates the SSH connection to an
    HPC system. After that, it executes a simple command.
1s = SSHOperator(
        task id='1s',
        command='ls -1',
        ssh_hook=sshHook,
        dag=dag)

# Instantiating an SSHOperator, which sets up an SSH connection to an HPC system
    and then executes a batch script located on that system.
test = SSHOperator(
        task_id='test',
        command=task_ssh_bash,
        ssh_hook=sshHook,
```

4 Adoption Plans of the Framework

This section outlines the plans for the adoption of the UAIF for different stakeholders during the remainder of the project and beyond based on the sustainability strategy that is being discussed in WP5. That is, it is the goal of this section to focus on the technical realization on the possible adoption while respecting commercialization plans regarding sustainability of the project, business plans, and certification as provided in complementary WP5 Deliverables that follow the same project impact vision. The adoption plan will provide guidance for the remaining time of the project to proceed with the adoption jointly performed with the different listed EU activities and the contacts established through them, e.g., SMEs, industry or government, academic or industrial HPC centers, etc.

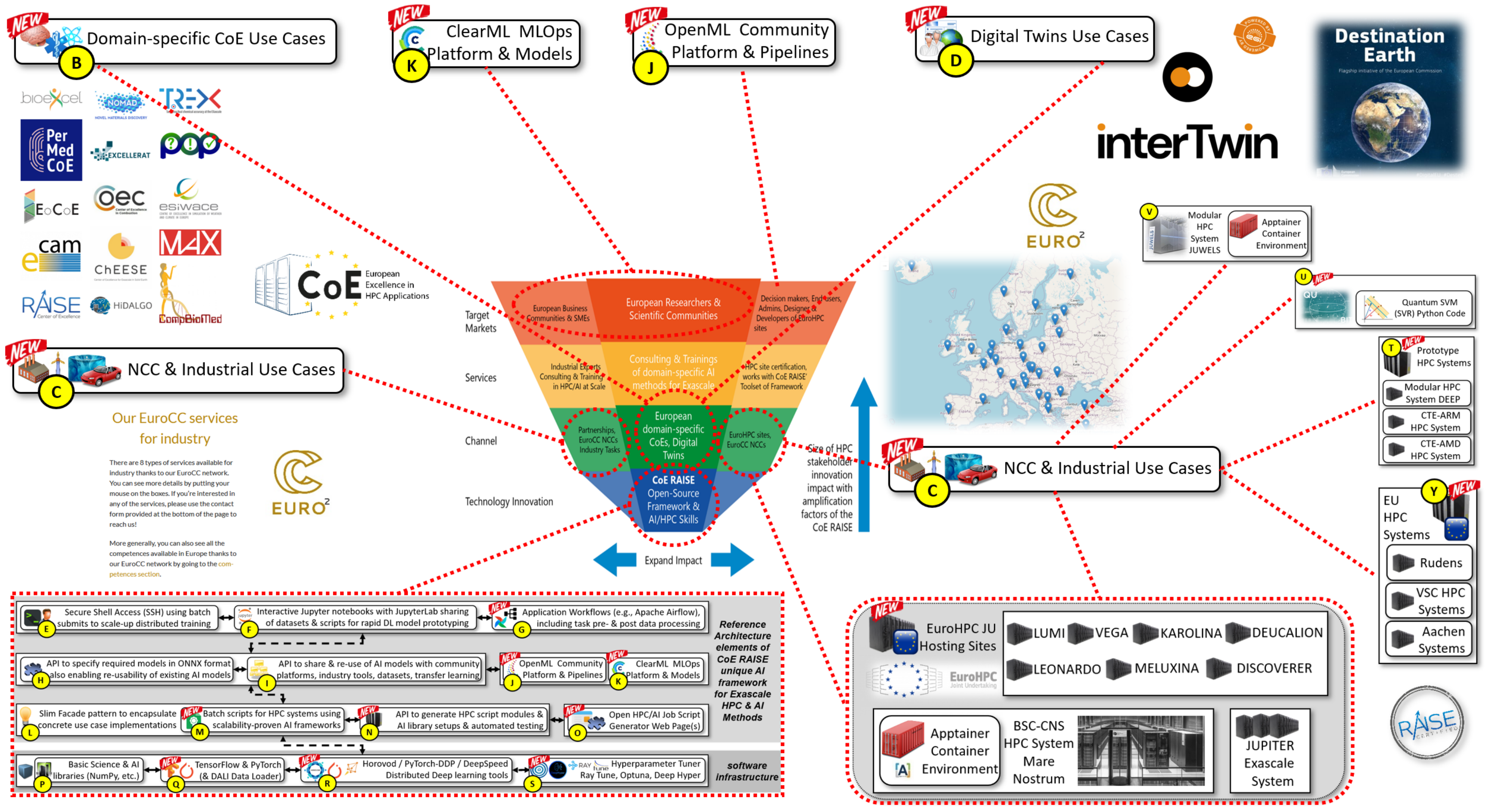
Figure 14 summarizes the adoption plan within the larger context of the project impact plans. Each subsection will address the relevant parts of this figure and will connect it with other ongoing EU activities that can enable an amplification factor of impact for the CoE RAISE project. In more detail, Sec. 4.1 and Sec. 4.2 present the adoption plans for EuroHPC JU hosting sites and for the NCCs of EuroCC-2. Sunsequently, Sec. 4.3 and Sec. 4.4 discuss them for other CoEs and DT endeavors. Finally, Sec. 4.5 provides a plan for the wider AI community.

At the time of writing, many members of the CoE RAISE project are already engaged in various steps of this plan. It should be noted that not all of the listed EU activities in Fig. 14 have been contacted, nor collaborations with all of them have been established.

4.1 Adoption Plans for EuroHPC JU Hosting Sites

The general plan for the adoption of the UAIF by EuroHPC JU hosting sites was briefly described above in Sec. 2.4.5. The formula for beginning adopting requires assembling the relevant experts from both the hosting site and CoE RAISE. Along with experts from WP2 and WP5, administrators, technical, and software experts of the hosting site should be included in the collaboration meetings. After an initial presentation of the UAIF goals and potential benefits of the adoption to users and hosting site, the adoption steps are as follows. The first step is a survey and analysis of what software is already available on a specific hosting site compared the components of the UAIF. The second step is having discussions about adding or modifying the hosting site software availability over time, e.g., in Fig. 15 roughly three month are estimated – potentially even more for Exascale hosting sites, with selected UAIF components, where useful. In the third step, testing and certification runs of applications are performed to demonstrate the benefits gained from the UAIF components, where adopted. This also includes discussions on certification with hosting sites to understand what a CoE RAISE certification entails. The final step is to collect tailored job scripts customized to the hosting site and integrating them into the LAMEC API described in Sec. 3.6.

Figure 15 shows the projected adoption timeline for EuroHPC JU hosting sites for selected UAIF components. The timeline serves as an estimate and is subject to availability from the corresponding hosting site. At the time of writing, several hosting sites have been contacted and are pending discussions with systems administrators and application specialists. As shown in Fig. 14, in some cases, NCCs have close relations to hosting sites or are the hosting site partner directly. In these cases, a closer collaboration and faster adoption of the UAIF at the hosting sites is foreseen. Alongside WP2 experts, there will be also WP5 experts participating in the collaboration to understand the potential of certification for a hosting site. While WP2 provide technical discussion inputs, the WP5 experts provide insights on the certification process of the project.



Domain-specific CoE Use Cases
ClearML MLOps Platform & Models
OpenML Community Platform & Pipelines
Digital Twins Use Cases
Destination Earth
interTwin
NCC & Industrial Use Cases
Our EuroCC services for industry
EURO²
CoE European Excellence in HPC Applications
Target Markets
Services
Channel
Technology Innovation
European Business Communities & SMEs
European Researchers & Scientific Communities
Decision makers, End-users, Admins, Designers & Developers of EuroHPC sites
Industrial Experts Consulting & Training in HPC/AI at Scale
Consulting & Trainings of domain-specific AI methods for Exascale
HPC site certification, works with CoE RAISE' Toolset of Framework
Partnerships, EuroCC NCCs Industry Tasks
European domain-specific CoEs, Digital Twins
EuroHPC sites, EuroCC NCCs
CoE RAISE Open-Source Framework & AI/HPC Skills
Expand Impact
Size of HPC stakeholder innovation impact with amplification factors of the CoE RAISE
Modular HPC System JUWELS
Apptainer Container Environment
Quantum SVM (SVR) Python Code
Prototype HPC Systems
Modular HPC System DEEP
CTE-ARM HPC System
CTE-AMD HPC System
EU HPC Systems
Rudens
VSC HPC Systems
Aachen Systems
EuroHPC JU Hosting Sites
LUMI
VEGA
KAROLINA
DEUCALION
LEONARDO
MELUXINA
DISCOVERER
Apptainer Container Environment
BSC-CNS HPC System Mare Nostrum
JUPITER Exascale System
Secure Shell Access (SSH) using batch submits to scale-up distributed training
Interactive Jupyter notebooks with JupyterLab sharing of datasets & scripts for rapid DL model prototyping
Application Workflows (e.g., Apache Airflow), including task pre- & post data processing
API to specify required models in ONNX format also enabling re-usability of existing AI models
API to share & re-use of AI models with community platforms, industry tools, datasets, transfer learning
OpenML Community Platform & Pipelines
ClearML MLOps Platform & Models
Slim Facade pattern to encapsulate concrete use case implementations
Batch scripts for HPC systems using scalability-proven AI frameworks
API to generate HPC script modules & AI library setups & automated testing
Open HPC/AI Job Script Generator Web Page(s)
Basic Science & AI libraries (NumPy, etc.)
TensorFlow & PyTorch (& DALI Data Loader)
Horovod / PyTorch-DDP / DeepSpeed Distributed Deep learning tools
Hyperparameter Tuner Ray Tune, Optuna, Deep Hyper
Reference Architecture elements of CoE RAISE unique AI framework for Exascale HPC & AI Methods
software infrastructure

Figure 14: Adoption plans for the UAIF for potential European stakeholders with links to the overall project impact plans. The plan is aligned with plans of other relevant EU activities like the EuroHPC JU, the EuroCC-2 project, other CoEs, and DTs.

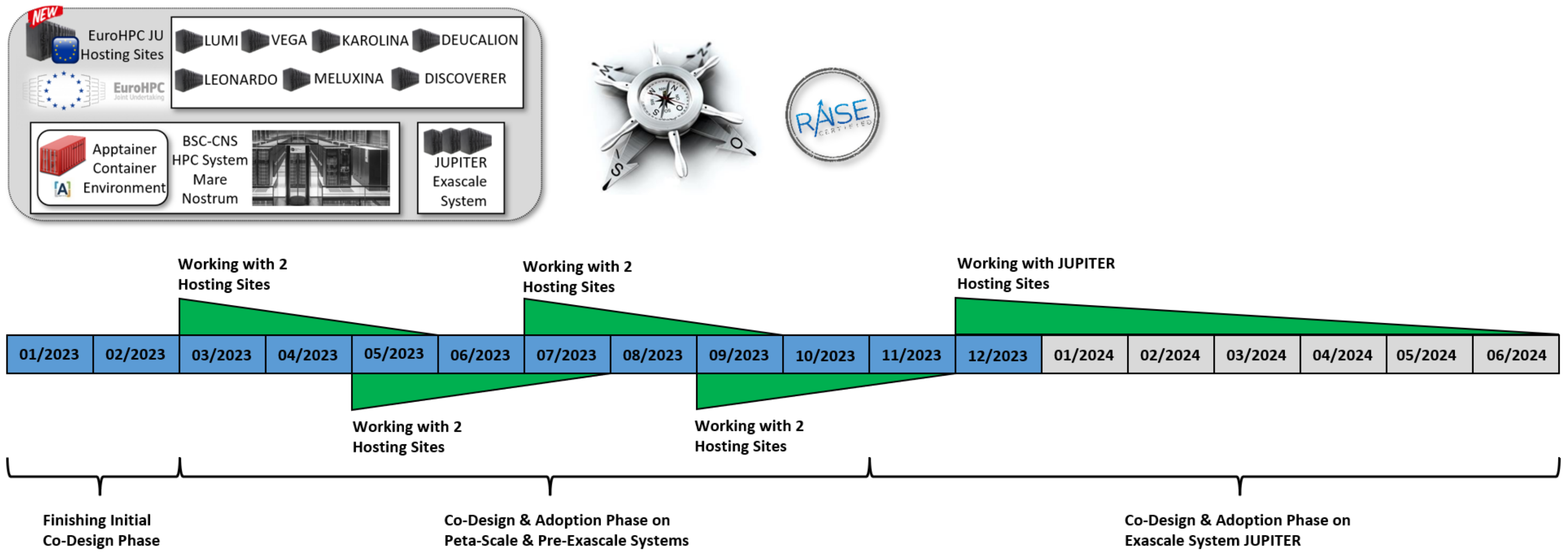

NEW
EuroHPC JU Hosting Sites
EuroHPC Joint Undertaking
LUMI
VEGA
KAROLINA
DEUCALION
LEONARDO
MELUXINA
DISCOVERER
Apptainer Container Environment
BSC-CNS HPC System Mare Nostrum
JUPITER Exascale System
RAISE CERTIFIED
Working with 2 Hosting Sites
Working with 2 Hosting Sites
Working with JUPITER Hosting Sites
01/2023
02/2023
03/2023
04/2023
05/2023
06/2023
07/2023
08/2023
09/2023
10/2023
11/2023
12/2023
01/2024
02/2024
03/2024
04/2024
05/2024
06/2024
Working with 2 Hosting Sites
Working with 2 Hosting Sites
Finishing Initial Co-Design Phase
Co-Design & Adoption Phase on Peta-Scale & Pre-Exascale Systems
Co-Design & Adoption Phase on Exascale System JUPITER

Figure 15: Adoption plans for the UAIF for potential European stakeholders with a particular focus on the EuroHPC JU hosting sites. The project aims to work with two new hosting sites every two month. Based on the sustainability strategy of WP5 and realization timeline of JUPITER, the most efforts towards adopting the UAIF on a European Exascale machine would be only in 2024. The blue timeframe reflects the runtime of the CoE RAISE project while the grey timeframe depends on the sustainability of the project and work performed in WP5.

The timeline shown in Fig. 15 is quite ambitious. However, several of the UAIF components are already available at many of the targeted sites. It is expected that this will accelerate the adoption of the UAIF in many cases, while preserving autonomy as not all components of the UAIF need to be adopted, i.e., it depends on the interest of the specific hosting site. CoE RAISE offers a significant speed-up through scaling and more efficient AI models for users through HPO. The incentive is quite high from all parties to save significant time for users in creating job scripts by cooperating with CoE RAISE by means of the LAMEC API described in Sec. 3.6. Hence, a primary goal of the collaboration with hosting sites entails the integration of tailored job scripts specific to the hosting site systems into the LAMEC API and the job script generator capabilities.

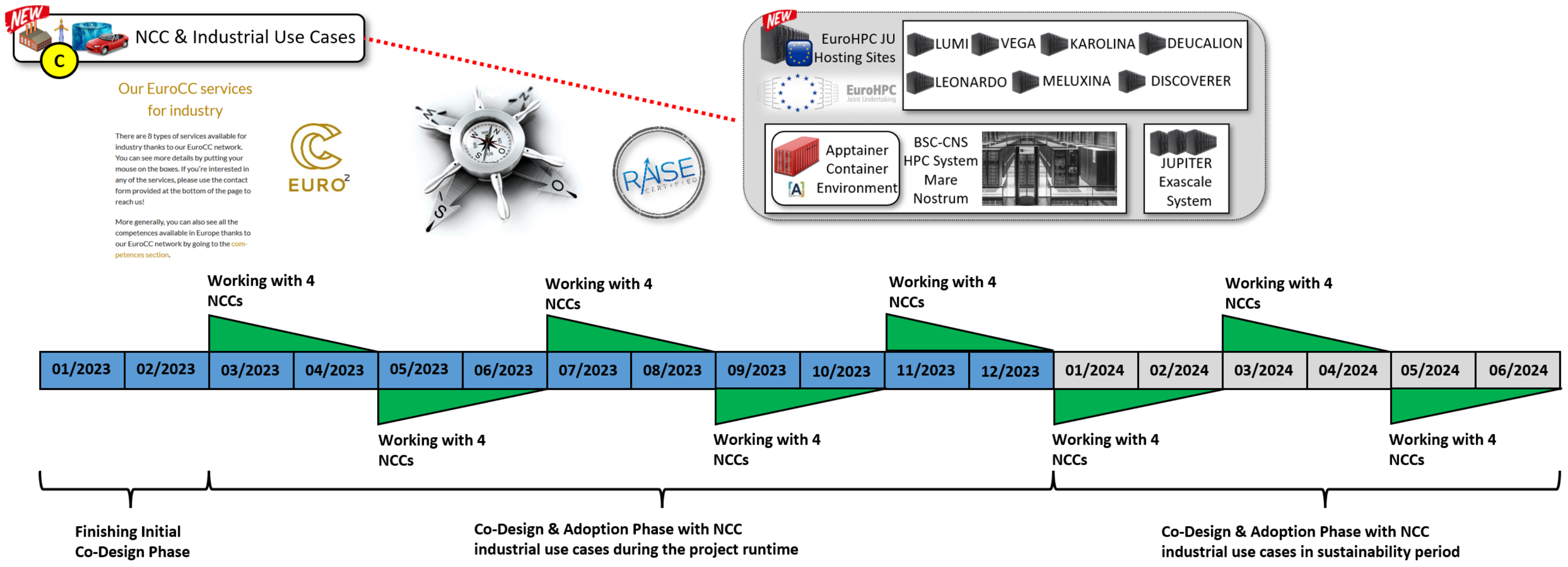
Finally, CoE RAISE does not compel or aim to influence hosting sites that have no time or resources to cooperate. This section has outlined a general plan of adoption. It should be noted that the plan does no guarantees that hosting sites will engage as projected in Fig. 15. This timeline remains subject to further discussion and evolution in the remainder of the project.

4.2 EuroCC-2 National Competence Centers (NCCs)

The general idea of adoption by NCCs of the UAIF has already been sketched in Sec. 2.1.3. As shown in Fig. 14 and Fig. 16, the roles of NCCs in the context of the adoption are twofold. Several NCCs are close to EuroHPC JU hosting sites or are both the same organization thus also being a link to the adoption of the UAIF on HPC systems. Secondly, NCCs have the goal to enable industrial use cases and are thus also good contacts to understand what role HPC plays in industry and where the UAIF components may help. Several NCCs might not be fully operational yet or are just being built in selected countries. As a consequence, the adoption plan in Fig. 16 only plans for working with 20 NCCs in the remaining project runtime. More work with NCCs in 2024 depends on the sustainability plan created in WP5.

Approaching the NCCs is similar to approaching the hosting sites. Apart from WP2 and WP5 experts, e.g., for certification, also the technical, industrial, and software experts of the NCC should be part of the collaboration meetings. After an initial presentation of the UAIF goals and benefits to NCCs, the adoption steps are as follows. The first step in the adoption is an analysis of what industrial use cases of the NCC can potentially make use of the UAIF. The second step includes discussions about which HPC sites are currently being used by these industrial use cases and if good connections between the corresponding NCC to some EuroHPC JU hosting site exist. Modifying the use case software incrementally, e.g., in Fig. 16 roughly two month are estimated, with selected UAIF components, where useful, is part of this step. In the third steps, test runs of industrial applications show again the benefits of the UAIF components where adopted. The third step includes, if needed, specific job scripts that work with the UAIF LAMEC API described in Sec. 3.6.

Figure 16 shows the rough timeline plan of working with NCCs on the potential adoption of selected UAIF components in specific industrial use cases. Naming specific NCCs is explicitly avoided as it is subject to the availability of corresponding technical and industrial experts. At the time of writing, discussions are going on with several NCCs that are open to the idea. As shown in Fig. 16, in some cases also NCCs have good contacts to hosting sites or are even the hosting site partner directly. In such a cases, a closer collaboration towards the adoption of the UAIF at hosting sites as well working jointly with corresponding NCCs is foreseen.


NEW
NCC & Industrial Use Cases
C
Our EuroCC services for industry
EURO²
RAISE
NEW
EuroHPC JU Hosting Sites
EuroHPC
LUMI
VEGA
KAROLINA
DEUCALION
LEONARDO
MELUXINA
DISCOVERER
Apptainer Container Environment
BSC-CNS HPC System Mare Nostrum
JUPITER Exascale System
Working with 4 NCCs
01/2023
02/2023
03/2023
04/2023
05/2023
06/2023
07/2023
08/2023
09/2023
10/2023
11/2023
12/2023
01/2024
02/2024
03/2024
04/2024
05/2024
06/2024
Finishing Initial Co-Design Phase
Co-Design & Adoption Phase with NCC industrial use cases during the project runtime
Co-Design & Adoption Phase with NCC industrial use cases in sustainability period

There are 8 types of services available for industry thanks to our EuroCC network. You can see more details by putting your mouse on the boxes. If you're interested in any of the services, please use the contact form provided at the bottom of the page to reach us!

More generally, you can also see all the competences available in Europe thanks to our EuroCC network by going to the competences section.

Figure 16: Adoption plans for the UAIF for potential European stakeholders with a particular focus on industrial use cases and the EuroHPC JU hosting sites where connections exist. The project aims to work with two four different NCCs every two month. Based on the sustainability strategy of WP5, the work with other NCC will continue on 2024. The blue timeframe reflects the runtime of the CoE RAISE project while the grey timeframe depends on the sustainability of the project and work performed in WP5.

The timeline shown in Fig. 16 is quite ambitious. However, several of the UAIF components are already used by NCC industrial use cases in the field of AI. This in turn will accelerate the adoption of the UAIF in many industrial use cases while also not all components of the UAIF need to be adopted. It thus depends on the interest of the NCC industrial use cases themselves.

Apart from a significant speed-up through scaling and better AI models for industrial users through HPO, the incentive is quite high to save users significantly time in creating job scripts by cooperating with CoE RAISE with respect to the LAMEC API. Hence, part of the collaboration with NCCs entails also the feedback of working with specific job scripts on specific hosting site systems with the LAMEC API and the job script generator capabilities that in turn might be another incentive.

Finally, it is clear that CoE RAISE has no influence on NCCs or any industrial users having time or not willing to cooperate. This section just outlines a plan and gives no guarantees that NCCs will engage as planned on this timeline that is subject to further discussions during the remainder of the project.

4.3 Other Centers of Excellence

This section sketches possible benefits of adopting the UAIF in other CoEs, cf. Fig. 14. From the beginning of the project, CoE RAISE was designed in such a way that the use cases are the drivers of cross-sectional AI and HPC developments. Two primary application classes, i.e., the compute- (WP3) and data-driven (WP4) use case categories were selected to drive developments towards large-scale AI application, with WP2 being responsible for the "AI- and HPC-glue in between".

Exchanges with other CoEs, such as the CoE for Targeting Real Chemical Accuracy at the Exascale (TREX)[79], CoE for Engineering Applications (Excellerat)[80], or the CoE for Combustion (CoEC)[81], have revealed AI-method development in some manner. However, these initiatives are not primarily focused on Exascale deployment, and approach AI as a tool for other processes. For example, in CoEC, super-resolution networks are developed to enhance the resolution in coarse-grained combustion simulations, where they replace under-resolved phenomena as subgrid-scale models for Large Eddy Simulations (LESs). The implementation of such models is, however, not in the mission of the CoEC developments, which is the same for the other old CoEs. This has been exposed in discussions with the coordinators of these CoEs.

AI has become a hot topic across various domains and is hence also part of the mission of the new CoEs, which receive funding starting in 2023. For example, Excellerat P2, CoE for Exascale in Solid Earth 2 (ChEESE-2P)[82], CoE for Computational Biomolecular Research 3 (BioExcel-3)[83], and CoE for HPC and Big Data Technologies for Global Challenges 2 (HiDALGO2)[84] all feature AI component developments. Regardless that AI now plays a bigger role in recent CoEs, their focus remains on advancing a specific domain codes towards Exascale, in which AI might be a tool for physics modeling and acceleration, but only few or non Exascale-AI knowledge is built up. That is, the core developments of the traditional CoEs push individual domain-specific codes to use new HPC architectures and to prepare them for hero runs using the complete resources on the next-generation of supercomputers but might lack expertise in training large-scale AI technologies.

[79] TREX https://trex-coe.eu
[80] Excellerat https://www.excellerat.eu
[81] CoEC https://coec-project.eu
[82] ChEESE-2P https://cheese-coe.eu/
[83] BioExcel-3 https://bioexcel.eu/
[84] HiDALGO2 https://www.hpccoe.eu/hidalgo-2/

This is exactly, where CoE RAISE and its UAIF fit in. The AI and HPC developments in CoE RAISE are highly cross-sectional to use cases from other CoEs. It is a strong benefit in CoE RAISE that these methods are developed for general application, despite being driven by specific use cases. A special role in this play common AI architectures such as Fourier Neural Operators (FNOs), CNNs, and GNNs. CNNs are widely used for extracting critical features from images to make accurate predictions about their contents and GNNs are ideally suited to treat structured or unstructured (hierarchical) simulation meshes, also with Adaptive Mesh Refinement (AMR) features, that are used to discretize the space for multi-physics simulations. Such simulation methods are used by many CoEs. The other UAIF component of high importance is HPO, which has in CoE RAISE proven to be a method relevant to almost all AI trainings, and is particularly useful when the parameter space is large. There is a rapidly developing need for efficient methods to couple simulations and AI training and inference at runtime using heterogeneous architecture, or in other words MSAs. The European Center for Research and Advanced Training in Scientific Computation (CERFACS) pioneers this field and develops the Physics Deep Learning coupLer (PhyDLL)[85] as part of the UAIF. FZJ and RWTH are about to join the development and will contribute a `C++` interface to PhyDLL. Obviously, all these developments are relevant not only to CoE RAISE but – due to their general applicability – also for the other CoEs.

As a consequence, it makes sense to provide services from CoE RAISE to the other CoEs with respect to using the UAIF, further jointly developing models for specific CoE RAISE external use cases, extending UAIF components by those, work on best practice documents, and provide and/or organize joint trainings. Such a training has already taken place with CoEC and is planned with TREX. CoE RAISE is in this respect open to look at new use cases, find ways to generalize methods, and make new AI technologies ready for HPC at large scale. A special focus is on using novel hardware components and especially QA and QC technologies - all with respect to AI applications. Outreach in this direction is already taking place and more and more developers in CoE RAISE and also outside the project are attracted by CoE RAISE's developments. For example, a group from the Deutsches Elektronen-Synchrotron (DESY) are eager to join efforts with CERN and FZJ to work on Machine-Learned Particle-Flow (MLPF)-like algorithms or other groups from RWTH and Delft University of Technology (TUDelft) to explore QA systems for engineering applications.

To this end, CoE RAISE is going to foster the above described activities and ideas in the remainder of the project duration. Obviously, when it comes to services and commercialization, the experts of WP5 are involved and support detailing the strategies and planning the next steps.

4.4 Digital Twin Projects

DTs, or hyper-accurate digital representations of physical systems, have offered unprecedented insights for researchers and have an increasing demand for ready-to-use tools that are able to process and run complex and AI-based workflows in heterogeneous HPC environments. The DTs initiative is part of a larger EU strategy with diverse scientific applications, such as earth sciences with Destination Earth[86].

At the time of writing, options are explored to cooperate with DT projects like Destination Earth. Given the close interaction with overlapping project partners, CoE RAISE first focuses on possible collabo-

[85] PhyDLL https://gitlab.com/cerfacs/phydll
[86] Digital Twin Destination Earth https://digital-strategy.ec.europa.eu/en/policies/destination-earth

rations with the project interTwin[87]. In interTwin, the CoE RAISE partners CERN, CERFACS, and FZJ are contributing. It is funded by the European Union Horizon Europe Programme and devleops a DT blueprint architecture and an adaptable Digital Twin Engine (DTE) for various scientific areas through interdisciplinary collaboration. The engine will serve as an open-source platform that provides generic and tailored software components for modeling and simulation to integrate application-specific DTs. The DTE will deliver the generic capabilities of high-volume, high-speed, real-time data processing, forecast by domain-specific models, and validation tools. Figure 17 shows the different layers of the planned DTE.

The modularity and functionality of the DTE will be demonstrated in four different scientific domains – HEP (represented by CERN), radio astronomy, gravitational-wave astrophysics, as well as climate and environmental monitoring, where CERFACS contributes. FZJ brings in its expertise and experience in hosting and providing a world-class HPC infrastructure and corresponding services, and staff will assist in the development of novel AI technologies towards Exascale computing in a joint effort with CoE RAISE. Furthermore, FZJ will provide and develop software solutions to improve access to a wide range of heterogeneous computing and storage resources, including HPC and quantum systems. More specifically, CoE RAISE researchers from FZJ and CERN collaborate on the development of an AI workflow and method lifecycle. It is the aim to provide generalizable and widely applicable tools that can design and use AI workflows that are capable of connecting to HPC executions in the background. These tools will be based on JupterJSC, which is also offered by CoE RAISE as a service[88], Jupyter Dashboards[89], and on the AI-centric extension Elyra[90]. The framework is envisioned as modular system, where different modules represent a specific part of a flexibly designed AI workflow. A wide range of modules – such as a model training module, a model evaluation module, a distributed training frame work, and an HPO module – will be included to provide the users with a predefined set of tools. The modules will be interchangeable, providing the opportunity to easily integrate user-defined modules.

Obviously, there are similar development strands in interTwin and CoE RAISE and hence an intensive collaboration is envisaged. At the project kick-off of interTwin (EGI Conference 2022 in Prague, Sep. 19 to 20, 2022), intense discussions on possible synergies took place. Figure 17 highlights possible interactions and activities of CoE RAISE and interTwin. Especially the generalizable and scalable, as well as the user-specific AI methods developed in CoE RAISE (and being part of the UAIF) are of particular interest to interTwin, see (2) in Fig. 17. The interTwin project seeks to establish a fundamental database on AI technologies, ready to be used in Elyra, which could come from the UAIF for specific DTs. Here, HPO and parallel training is of particular interest as it is common to most data-intensive DT / simulation processes and is a core component of the interTwin developments. Such methods are also already components of the UAIF and CoE RAISE could provide trainings and support connecting methods in the AI / ML module of interTwin to the HPC backbone. When it comes to the usage of HPC / quantum components in the context of AI development, CoE RAISE alraedy provides system-adapted start scripts in the UAIF, which will also be made available as a service online on the CoE RAISE website, see (3) in Fig. 17. Considering that CoE RAISE is already offering JupyterJSC as a service, extending this development platform by Elyra will benefit both CoE RAISE and interTwin. Experiences in interTwin could be provided to CoE RAISE in this context. This CoE RAISE service

[87]interTwin https://www.intertwin.eu
[88]CoE RAISE Services https://www.coe-raise.eu/service-portal
[89]Jupyter Dashboards https://jupyter-dashboards-layout.readthedocs.io
[90]Elyra https://github.com/elyra-ai/elyra

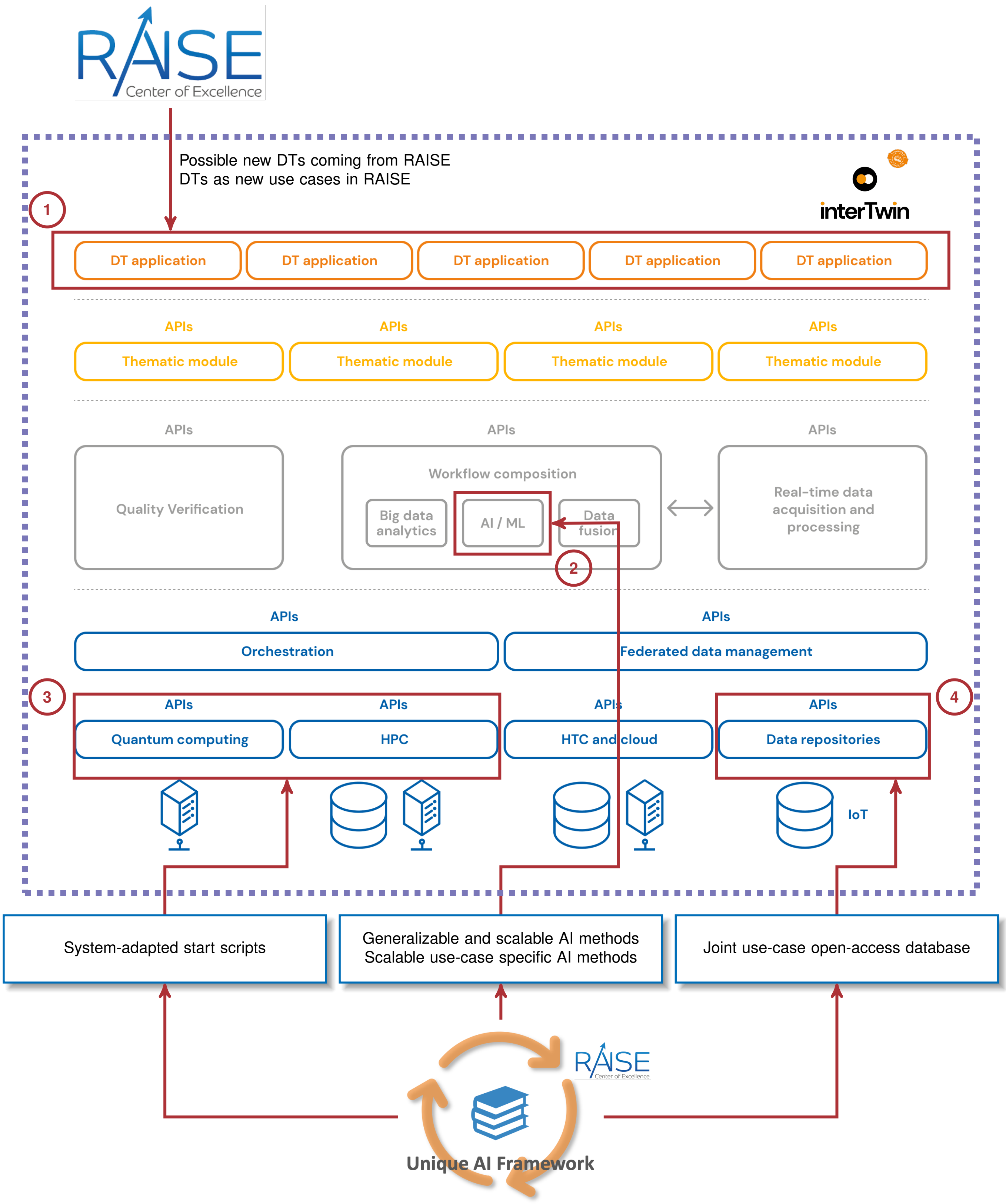


Figure 17: Foreseen interactions and activities between CoE RAISE and interTwin.

will serve as a blueprint to ease usage of AI development platforms on various HPC systems provided Euopean-wide – it will hence also benefit interTwin. Another aspect could be sharing data repositories for AI training, see (4) in Fig. 17. CoE RAISE has already started to provide the community with data, which is the base for AI training performed in CoE RAISE[91]. In case of an extension of the DT applications in interTwin towards CoE RAISE-like use cases, training data, pre-trained models, and best practice information would be available in the UAIF. Indeed, a mutual extension of applications for the two projects is in line with looking beyond the scope of the current work plan of both projects and with CoE RAISE's ambition to transfer gained knowledge, experience, and models to new domains. This is covered in (1) in Fig. 17.

To summarize, strong synergy potentials between the interTwin CoE RAISE projects have already been identified and it is the plan in the remainder of this project to exploit these synergies and to join the above mentioned development strands of the two projects. It should also be noted that interTwin will also reach out to other DT activities including Destination Earth and that CoE RAISE will also connect to this endeavor through interTwin.

4.5 AI Communities through OpenML and ClearML

The general idea of adoption by AI community platforms of the UAIF was already described above in Sec. 2.2.6 and Sec. 2.2.7. At the time of writing, the future adoption plan in this section is still abstract and considered to be refined during the remainder of the project period. Enabling interoperability between the UAIF with OpenML and ClearML (see Sec. 3.2) will provide an easier access to HPC for AI application experts, especially those not previously engaged in HPC. The work on the corresponding LAMEC API elements to enable interoperability with OpenML and ClearML is still work in progress. CoE RAISE aims for an availability of basic interoperability from the second half of 2023.

In the meantime, also other AI platforms and initiatives such as the AI-on-demand (AIOD) platform[92] are explored. One interesting element in this context is the AI4EU container specification that also includes code examples. CoE RAISE has explored an initial set of AI4EU experiments, which shown clear potentials for adopting HPC for both speed-up and improvement (in terms of accuracy) the AI through HPO.

[91] CoE RAISE Open Data https://www.coe-raise.eu/open-data
[92] AI-on-Demand Platform https://www.ai4europe.eu/

5 Summary and Conclusions

As shown in Fig. 1, the software layout plan for the UAIF has been updated for a wide variety of components, adding more details and implementation strategies. It can be concluded that the implementation process of the UAIF goes forward as planned with respect to scalability tests and benchmarking of components, but also with the implementation and design of the LAMEC API. As co-designed by the WP3 and WP4 use cases, corresponding components have also been adopted by the UAIF when they showed scaling capability towards Exascale. Future work includes the analysis of several possible UAIF components such as Apache MXNet, Paddle Paddle[93], or the general approach of some WP3 use cases in regard of coupling simulations with AI methods as described in "Deliverable D2.3 - Report on porting and performance analysis" (M24).

JSC will renew its module stages in March 2023 making a lot of job scripts again invalid. Using the LAMEC API of the UAIF instead, abstracts these changes and makes it easier for users to maintain accurate job scripts. At the time of writing, the hosting for the job script generator web page is already decided to be at JSC, providing also a simple GUI for the LAMEC API in the near future. It can further be concluded that one of the objectives of the UAIF to ease the use of modules is still considered a significant help.

The CoE RAISE project pursues an ambitious time schedule regarding the adoption of the UAIF and its different components as outlined in Sec. 4. CoE RAISE continues to collaborate with EuroHPC JU hosting sites, pursuing maximal support of the UAIF components to yield the largest impact for CoE RAISE, and to generate pathways for future sustainability models. Plans for working jointly with NCCs, other CoEs, DT projects, and often used AI community platforms have been shared. These plans will contribute to a significantly increased adoption of the UAIF components. Working jointly with these partners will further promote the uptake by industrial use cases. This in turn will bring the focus on UAIF components such as the containerization and future LAMEC API use case adaptations in this context.

Finally, it is important to note that the realization of the UAIF adoption plans jointly with external stakeholders are not in full control of the CoE RAISE project alone and will significantly depend on the time and expertise availability of the collaborating external partners. Initial discussions with the targeted adoption stakeholders show, however, a high interest and thus make the WP2 members confident in the adoption plans.

[93] Paddle Paddle https://github.com/PaddlePaddle/Paddle

A Appendix A - Previous Framework Layout

A.1 M9 - Initial Framework Software Layout Plan

Figure 18 below shows the initial software layout plan of the UAIF that is described in "D2.12 - Software Layout Plan for a unique AI Framework" (M9).

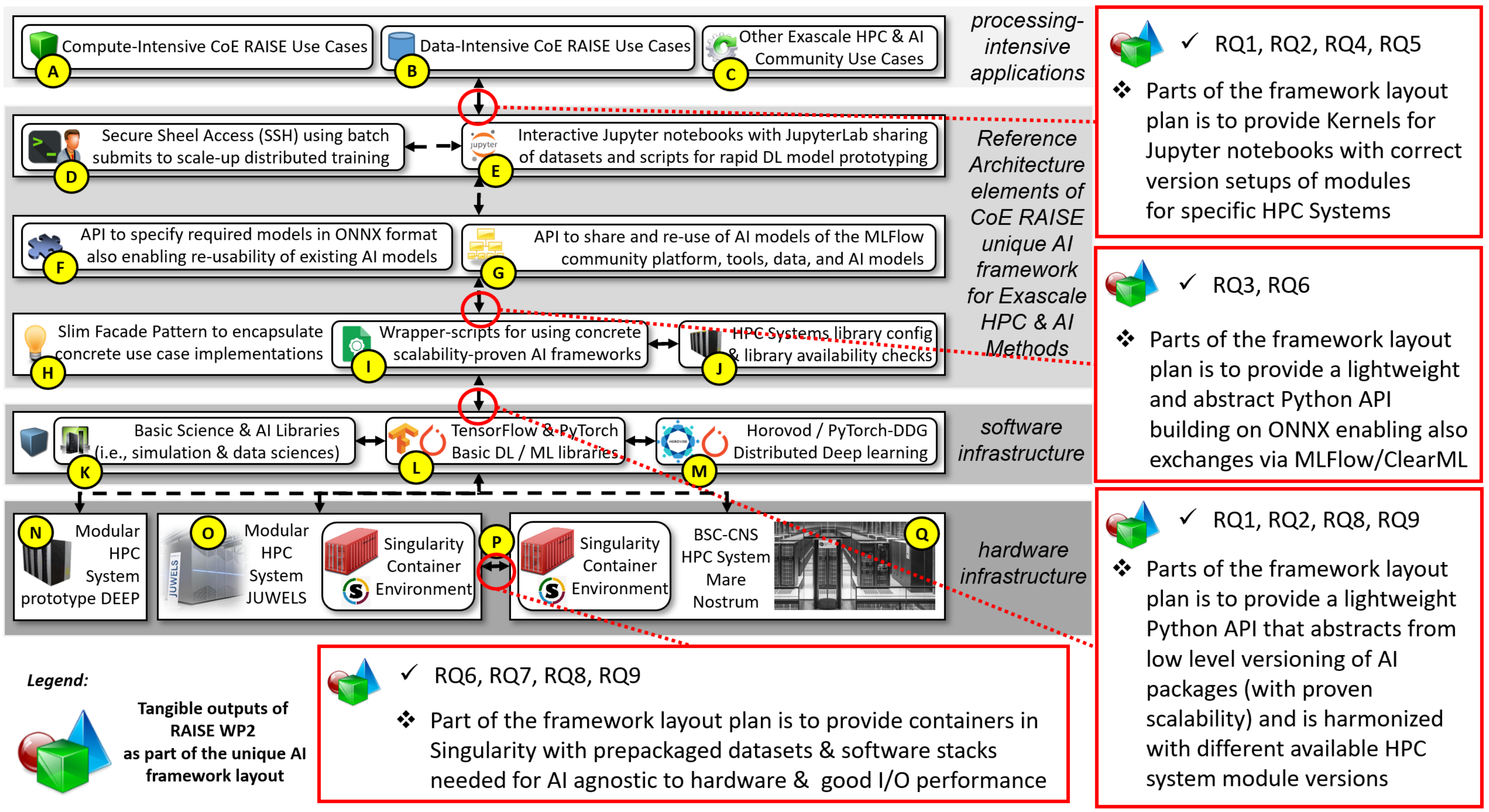

Compute-Intensive CoE RAISE Use Cases
A
Data-Intensive CoE RAISE Use Cases
B
Other Exascale HPC & AI Community Use Cases
C
processing-intensive applications
Secure Sheel Access (SSH) using batch submits to scale-up distributed training
D
Interactive Jupyter notebooks with JupyterLab sharing of datasets and scripts for rapid DL model prototyping
E
API to specify required models in ONNX format also enabling re-usability of existing AI models
F
API to share and re-use of AI models of the MLFlow community platform, tools, data, and AI models
G
Slim Facade Pattern to encapsulate concrete use case implementations
H
Wrapper-scripts for using concrete scalability-proven AI frameworks
I
HPC Systems library config & library availability checks
J
Reference Architecture elements of CoE RAISE unique AI framework for Exascale HPC & AI Methods
Basic Science & AI Libraries (i.e., simulation & data sciences)
K
TensorFlow & PyTorch Basic DL / ML libraries
L
Horovod / PyTorch-DDG Distributed Deep learning
M
software infrastructure
Modular HPC System prototype DEEP
N
Modular HPC System JUWELS
O
Singularity Container Environment
P
Singularity Container Environment
BSC-CNS HPC System Mare Nostrum
Q
hardware infrastructure
✓ RQ1, RQ2, RQ4, RQ5
❖ Parts of the framework layout plan is to provide Kernels for Jupyter notebooks with correct version setups of modules for specific HPC Systems
✓ RQ3, RQ6
❖ Parts of the framework layout plan is to provide a lightweight and abstract Python API building on ONNX enabling also exchanges via MLFlow/ClearML
✓ RQ1, RQ2, RQ8, RQ9
❖ Parts of the framework layout plan is to provide a lightweight Python API that abstracts from low level versioning of AI packages (with proven scalability) and is harmonized with different available HPC system module versions
✓ RQ6, RQ7, RQ8, RQ9
❖ Part of the framework layout plan is to provide containers in Singularity with prepackaged datasets & software stacks needed for AI agnostic to hardware & good I/O performance
Legend:
Tangible outputs of RAISE WP2 as part of the unique AI framework layout

Figure 18: Initial UAIF software layout plan at M9. A detailed description of its components alongside more details on the requirements RQ1-RQ7 are available in Deliverable D2.12.

A.2 M18 - Updated Framework Software Layout Plan

Figure 19 below shows the initial software layout plan of the UAIF that is described in "D2.10 - Monitoring Report" (M18).

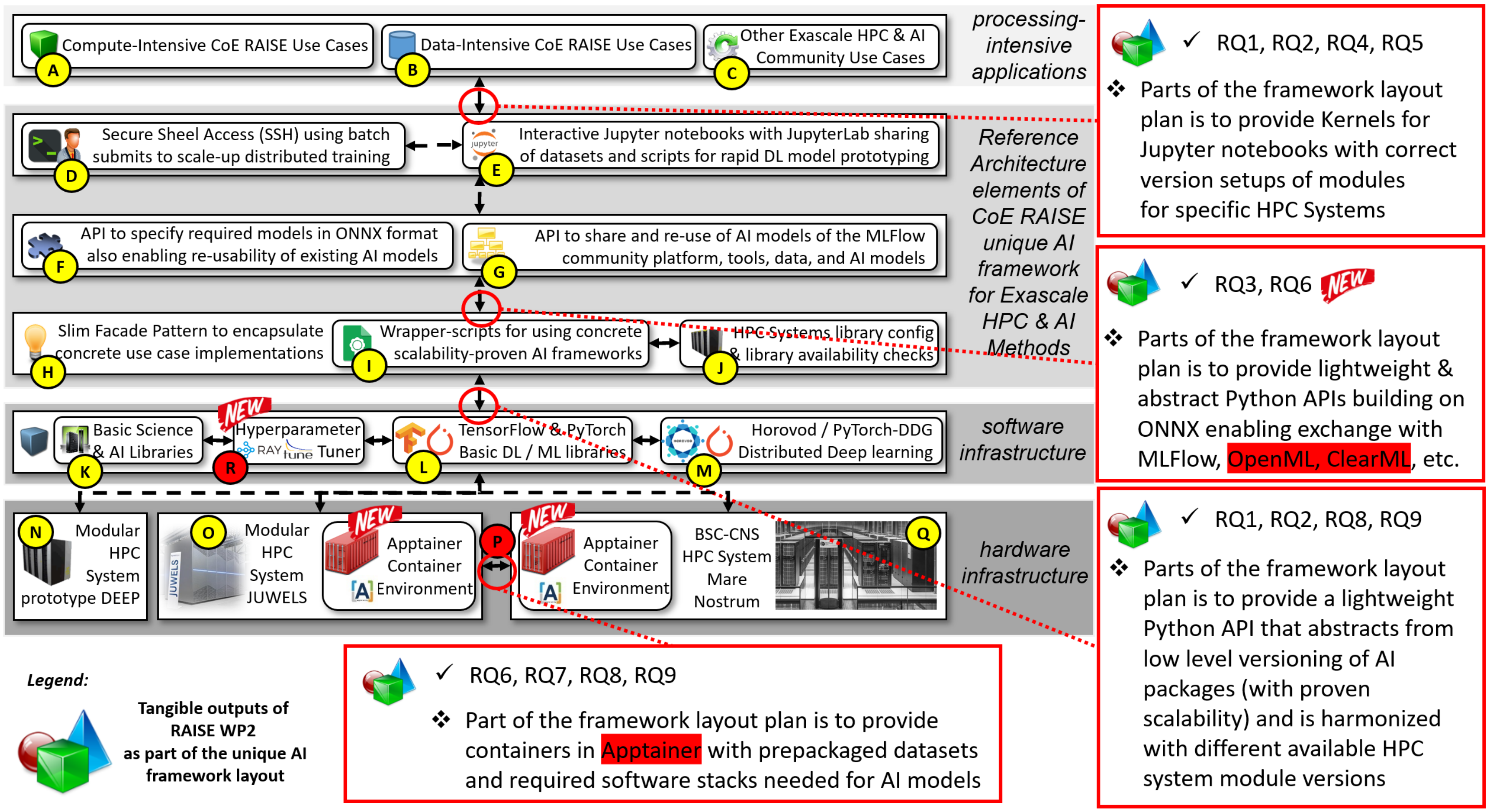

Compute-Intensive CoE RAISE Use Cases
A
Data-Intensive CoE RAISE Use Cases
B
Other Exascale HPC & AI Community Use Cases
C
processing-intensive applications
Secure Sheel Access (SSH) using batch submits to scale-up distributed training
D
Interactive Jupyter notebooks with JupyterLab sharing of datasets and scripts for rapid DL model prototyping
E
API to specify required models in ONNX format also enabling re-usability of existing AI models
F
API to share and re-use of AI models of the MLFlow community platform, tools, data, and AI models
G
Slim Facade Pattern to encapsulate concrete use case implementations
H
Wrapper-scripts for using concrete scalability-proven AI frameworks
I
HPC Systems library config & library availability checks
J
Reference Architecture elements of CoE RAISE unique AI framework for Exascale HPC & AI Methods
Basic Science & AI Libraries
K
NEW
Hyperparameter RAY Tune Tuner
R
TensorFlow & PyTorch Basic DL / ML libraries
L
Horovod / PyTorch-DDG Distributed Deep learning
M
software infrastructure
N
Modular HPC System prototype DEEP
O
Modular HPC System JUWELS
NEW
Apptainer Container Environment
P
NEW
Apptainer Container Environment
BSC-CNS HPC System Mare Nostrum
Q
hardware infrastructure
✓ RQ1, RQ2, RQ4, RQ5
❖ Parts of the framework layout plan is to provide Kernels for Jupyter notebooks with correct version setups of modules for specific HPC Systems
✓ RQ3, RQ6 NEW
❖ Parts of the framework layout plan is to provide lightweight & abstract Python APIs building on ONNX enabling exchange with MLFlow, OpenML, ClearML, etc.
✓ RQ1, RQ2, RQ8, RQ9
❖ Parts of the framework layout plan is to provide a lightweight Python API that abstracts from low level versioning of AI packages (with proven scalability) and is harmonized with different available HPC system module versions
✓ RQ6, RQ7, RQ8, RQ9
❖ Part of the framework layout plan is to provide containers in Apptainer with prepackaged datasets and required software stacks needed for AI models
Legend:
Tangible outputs of RAISE WP2 as part of the unique AI framework layout

Figure 19: Updated UAIF software layout plan at M18. A detailed description of its component updates are available in Deliverable D2.10. The major updates include the key role of Hpyerparameter Tuners such as Ray Tune and moving from Singularity to Apptainer for container solutions.

B Appendix B - Mural Board List of CoE RAISE Use Cases

The list of Mural Boards for each Interaction Room of WP2 vs. WP3/WP4 use cases was early made available in the CoE RAISE project as part of the following Basic Support for Cooperative Work (BSCW) folder:

https://bscw.zam.kfa-juelich.de/bscw/bscw.cgi/3591551

That same list is given here for readers convenience again per use case:

- Interaction Room Task 3.1 Turbulent Flow:
  https://app.mural.co/t/matthiasbook8855/m/matthiasbook8855/1621377866397/8613c384d54f66fb5e78599ff307a4ce8a9090c0?sender=u15e3008bb41d6628a5bb5701
- Interaction Room Task 3.2 Clean Energy:
  https://app.mural.co/t/matthiasbook8855/m/matthiasbook8855/1621377887905/cb44cca3eedd3bb9964fbfa36af16b1bfcce085f?sender=u15e3008bb41d6628a5bb5701
- Interaction Room Task 3.3 Reactive Flows:
  https://app.mural.co/t/matthiasbook8855/m/matthiasbook8855/1621377959022/0c363886f24833ecb19b025d87324b57fd50e2db?sender=u15e3008bb41d6628a5bb5701
- Interaction Room Task 3.4 Engine Design:
  https://app.mural.co/t/matthiasbook8855/m/matthiasbook8855/1621377976343/8d7aba6be09af3b2ffd305d2f709c53661ac889d?sender=u15e3008bb41d6628a5bb5701
- Interaction Room Task 3.5 Coating:
  https://app.mural.co/t/matthiasbook8855/m/matthiasbook8855/1621377991014/7a5d7e1eaf230178342d1e1d4a84d656d9055d52?sender=u15e3008bb41d6628a5bb5701
- Interaction Room Task 4.1 Fundamental Physics:
  https://app.mural.co/t/matthiasbook8855/m/matthiasbook8855/1621378007555/6f0d5285feaec5eafa515bd6676e84d8b4879d39?sender=u15e3008bb41d6628a5bb5701
- Interaction Room Task 4.2 Seismic Imaging:
  https://app.mural.co/t/matthiasbook8855/m/matthiasbook8855/1621378023838/a0b9503abb837ae3e28af4bb8d9adbec33874998?sender=u15e3008bb41d6628a5bb5701
- Interaction Room Task 4.3 Manufacturing:
  https://app.mural.co/t/matthiasbook8855/m/matthiasbook8855/1621378038069/93df6fa7a41093f4eaae7be9d72979de2ba42b9d?sender=u15e3008bb41d6628a5bb5701
- Interaction Room Task 4.4 Sound Engineering:
  https://app.mural.co/t/matthiasbook8855/m/matthiasbook8855/1621378050431/b5fa12219002404059f90a4bbb0101fa379a8503?sender=u15e3008bb41d6628a5bb5701

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, Li Fei-Fei, ImageNet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 248–255. doi:10.1109/CVPR.2009.5206848.

[2] M. Book, M. Riedel, H. Neukirchen, M. Goetz, Facilitating collaboration in high-performance computing projects with an interaction room, in: Proceedings of the 4th ACM SIGPLAN International Workshop on Software Engineering for Parallel Systems (SEPS 2017), 2017, pp. 46–47. doi:10.1145/3141865.3142467.

[3] M. Book, M. Riedel, H. Neukirchen, E. Erlingsson, Facilitating collaboration in machine learning and high-performance computing projects with an interaction room, in: Proceedings of the IEEE 18th International Conference on e-Science (e-Science), 2022, pp. 529–538. doi:10.1109/eScience55777.2022.00093.

[4] M. Riedel, M. Book, H. Neukirchen, G. Cavallaro, A. Lintermann, Practice and experience using high performance computing and quantum computing to speed-up data science methods in scientific applications, in: 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO), IEEE, 2022, pp. 281–286. doi:10.23919/MIPRO55190.2022.9803802.

[5] A. Yoo, J. Morris, M. Grondona, Slurm: Simple Linux utility for resource management, in: Proceedings of the Job Scheduling Strategies for Parallel Processing JSSPP - 9th International Workshop, 2003, pp. 44–60. doi:10.1007/10968987_3.

[6] N. Eicker, T. Lippert, T. Moschny, E. Suarez, D. project, The DEEP project an alternative approach to heterogeneous cluster-computing in the many-core era, Concurrency and computation: Practice and Experience 28 (8) (2016) 2394–2411. doi:10.1002/cpe.3562.

[7] E. Suarez, N. Eicker, T. Lippert, Modular supercomputing architecture: from idea to production, in: Contemporary high performance computing, CRC Press, 2019, pp. 223–255. doi:10.1201/9781351036863.

[8] M. Riedel, G. Cavallaro, J. A. Benediktsson, Practice and experience in using parallel and scalable machine learning in remote sensing from HPC over cloud to quantum computing, in: 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS, IEEE, 2021, pp. 1571–1574. doi:10.1109/IGARSS47720.2021.9554656.

[9] E. Pasetto, M. Riedel, F. Melgani, K. Michielsen, G. Cavallaro, Quantum SVR for chlorophyll concentration estimation in water with remote sensing, IEEE Geoscience and Remote Sensing Letters 19 (2022) 1–5. doi:10.1109/LGRS.2022.3200325.

[10] E. Wulff, M. Girone, D. Southwick, E. Cuba, J. G. Amboage, Hyperparameter optimization, multi-node distributed training and benchmarking of ai-based HEP workloads using HPC, Poster presentation in CERN, https://indico.cern.ch/event/1106990/contributions/4998112/attachments/2535527/4363652/ACAT2022_RAISE.pdf.

[11] S. Kesselheim, A. Herten, K. Krajsek, J. Ebert, J. Jitsev, M. Cherti, M. Langguth, B. Gong, S. Stadtler, A. Mozaffari, et al., JUWELS BOOSTER-a supercomputer for large-scale AI research,

in: High Performance Computing: ISC High Performance Digital 2021 International Workshops, Frankfurt am Main, Germany, June 24-July 2, 2021, Revised Selected Papers 36, Springer, 2021, pp. 453–468. doi:10.1007/978-3-030-90539-2.

[12] T. Elsken, J. H. Metzen, F. Hutter, Neural architecture search: a survey, The Journal of Machine Learning Research 20 (1) (2019) 1997–2017. doi:10.48550/arXiv.1808.05377.

[13] S. Falkner, A. Klein, F. Hutter, BOHB: Robust and Efficient Hyperparameter Optimization at Scale, in: Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 2018. arXiv:1807.01774.

[14] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, A. Talwalkar, A System for Massively Parallel Hyperparameter Tuning, in: Proceedings of Machine Learning and Systems, Vol. 2, 2018, pp. 230–246. arXiv:1810.05934.

[15] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, K. Kavukcuoglu, Population based training of neural networks (2017). arXiv:1711.09846.

[16] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization, The Journal of Machine Learning Research 18 (1) (2017) 6765–6816. arXiv:1603.06560, doi:10.5555/3122009.3242042.

[17] K. Jamieson, A. Talwalkar, Non-stochastic Best Arm Identification and Hyperparameter Optimization, in: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, Vol. 51, PMLR, Cadiz, Spain, 2016, pp. 240–248. arXiv:1502.07943.

[18] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, in: J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 24, Curran Associates, Inc., 2011, https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[19] C. Barakat, M. Aach, A. Schuppert, S. Brynjólfsson, S. Fritsch, M. Riedel, Analysis of chest X-ray for COVID-19 diagnosis as a use case for an HPC-enabled data analysis and machine learning platform for medical diagnosis support, Diagnostics 13 (3) (2023). doi:10.3390/diagnostics13030391.

[20] J. Rasley, S. Rajbhandari, O. Ruwase, Y. He, Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 3505–3506. doi:10.1145/3394486.3406703.

[21] C. Paris, L. Gasparella, L. Bruzzone, A scalable high-performance unsupervised system for producing large-scale hr land cover maps: The italian country case study, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 15 (2022) 9146–9159. doi:10.1109/JSTARS.2022.3209902.

[22] C. Paris, L. Bruzzone, D. Fernández-Prieto, A novel approach to the unsupervised update of land-cover maps by classification of time series of multispectral images, IEEE Transactions on Geoscience and Remote Sensing 57 (7) (2019) 4259–4277. doi:10.1109/TGRS.2018.2890404.

[23] A. A. Aleissaee, A. Kumar, R. M. Anwer, S. Khan, H. Cholakkal, G.-S. Xia, F. S. Khan, Transformers in Remote Sensing: A Survey (2022). arXiv:2209.01206.

List of Acronyms and Abbreviations

ACME	Automated Certificate Management Environment
AI	Artificial Intelligence
AIOD	AI-on-demand
AMD	Advanced Micro Devices
AMR	Adaptive Mesh Refinement
API	Application Programming Interface
ARM	Advanced Reduced Instruction Set Computer Machine
ASHA	Asynchronous Successive Halving Algorithm
BioExcel-3	CoE for Computational Biomolecular Research 3
BO	Bayesian Optimization
BOHB	Bayesian Optimization Hyperband
BSC	Barcelona Supercomputing Centre
BSCW	Basic Support for Cooperative Work
CAE	Convolutional Autoencoder
CBO	Centralized Bayesian Optimization
CERFACS	European Center for Research and Advanced Training in Scientific Computation
CERN	European Organization for Nuclear Research
CFD	Computational Fluid Dynamics
ChEESE-2P	CoE for Exascale in Solid Earth 2
CNN	Convolutional Neural Network
CoE	Euopean Center of Excellence
CoEC	CoE for Combustion
CoE RAISE	European Center of Excellence in Exascale Computing "Research on AI- and Simulation-Based Engineering at Exacale"
CPU	Central Processing Unit
CSCS	Swiss National Supercomputing Centre
CUDA	Compute Unified Device Architecture
DALI	NVIDIA Data Loading Library
DDP	Distributed Data Parallel
DEEP	Dynamical Exascale Entry Platform
DevOps	Development Operations
DESY	Deutsches Elektronen-Synchrotron
DT	Digital Twin
DTE	Digital Twin Engine
DL	Deep Learning
EU	European Union
Excellerat	CoE for Engineering Applications

FNO	Fourier Neural Operator
FZJ	Forschungszentrum Jülich GmbH
GNN	Graph Neural Network
GPU	Graphical Processing Unit
GUI	Graphical User Interface
HeAT	Helmholtz Analytics Toolkit
HEP	High-Energy Physics
HiDALGO2	CoE for HPC and Big Data Technologies for Global Challenges 2
HPC	High-Performance Computing
HPO	Hyperparameter Optimization
HTTPS	Hypertext Transfer Protocol Secure
IO	Input/Output
IPU	Intelligent Processing Unit
IR	Intermediate Representation
JIT	Just-In-Time Compilation
JSC	Jülich Supercomputing Centre
JU	Joint Undertaking
JUNIQ	Juelich UNified Infrastructure for Quantum computing
JUPITER	Joint Undertaking Pioneer for Innovative and Transformative Exascale Research
JURECA	Jülich Research on Exascale Cluster Architectures
JUWELS	Juelich Wizard for European Leadership Science
LC	Land Cover
LAMEC	Load AI Modules, Environments, and Containers
LES	Large Eddy Simulation
ML	Machine Learning
MLOps	Machine Learning Operations
MLPF	Machine-Learned Particle-Flow
MPI	Message Passing Interface
MSA	Modular Supercomputing Architecture
NAS	Neural Architecture Search
NAT	Network Address Translation
NCC	National Competence Center
NERSC	National Energy Research Scientific Computing Center
NLP	Natural Language Processing
ONNX	Open Neural Network Exchange
OS	Operating System
OSI	Open System Interconnection
PBT	Population Based Training
PCA	Principal Component Analysis

PhyDLL	Physics Deep Learning coupLer
PRACE	Partnership for Advanced Computing in Europe
QA	Quantum Annealing
QC	Quantum Computing
Q-SVR	Quantum Support Vector Regression
QUBO	Quadratic Unconstrained Optimization Problem
ROCm	Radeon Open Compute platforM
RS	Remote Sensing
RTU	Riga Technical University
RWTH	Rheinisch-Wesfälische Technische Hochschule Aachen - RWTH Aachen University
SaaS	Software as as Service
SLURM	Simple Linux Utility for Resource Management
SME	Small and Medium Enterprise
SPMD	Single-Program Multiple-Data
SSH	Secure Shell
SSPL	Server Side Public License
SVM	Support Vector Machine
SVR	Support Vector Regression
TF	Transformer
TL	Transfer Learning
TLS	Transport Layer Security
TPU	Tensor Processing Unit
TREX	CoE for Targeting Real Chemical Accuracy at the Exascale
TUDelft	Delft University of Technology
UAIF	Unique AI Framework
UQ	Uncertainty Quantification
VM	Virtual Machine
VSC	Vlaams Supercomputer Centre
WOPRO	Work Programme
WP	Work Package
XLA	Accelerated Linear Algebra
ZeRO	Zero Redundancy Optimizer