EDIH4UrbanSAVE

Research Computing Infrastructure Platform for Experiments – Version 1

Deliverable D2.3, Version 1.9, 30.10.2023

This project has received funding from the European Union's Digital 2021 research and innovation program under grant agreement No 101083713.

DIGITAL-2021-EDIH-01-101083713

EDIH
For urban interconnected supply and value Ecosystems

www.edih-hamburg.de

Hamburger Informatik Technologie-Center e.V., Germany	HITeC eV
Artificial Intelligence Center Hamburg e.V., Germany	ARIC eV
Digital Hub Logistics GmbH, Germany	DigiHub
Hochschule fuer Angewandte Wissenschaften Hamburg, Germany	HAW
Technische Universitaet Hamburg, Germany	TUHH
Handwerkskammer Hamburg	HWK
Handelskammer Hamburg (assoc.)	HK
City of Hamburg, Ministry for Economy, Transport and Innovation, Germany (assoc.)	BMWI
Innovation Kontakt Stelle Hamburg, Germany (assoc.)	IKS

Research Computing Infrastructure Platform for Experiments – Version 1

Work package	WP2
Task	T2.3, T2.4, T2.5
Document number	D2.3
Deliverable type	DEM
Title	Research Computing Infrastructure Platform for Experiments – Version 1
Author(s) and Contributor(s)	HITeC: Kai Himstedt, Daniel Speck, Stephanie von Riegen
Reviewer(s)	ARIC: Florian Vogt
Location	Teams Collaboration Platform: WP2 EDIH_Deliverable_2_3_RCI_Platform_for_ Experiments_Version_1
Version	1.9
Status	Final
Dissemination Level	Public

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

30.10.2023

History of changes

Date	Ver.	Author(s)	Change description
04.10.2023	1.1	HITeC eV K. Himstedt	Document creation
05.10.2023	1.2	HITeC eV K. Himstedt	Document revision, overview added
11.10.2023	1.3	HITeC eV K. Himstedt	Document revision, architectures added
12.10.2023	1.4	HITeC eV K. Himstedt	Document revision, summary added
13.10.2023	1.5	HITeC eV K. Himstedt	Figures added
16.10.2023	1.6	HITeC eV K. Himstedt	Document revision, layout improved, glossary items added
24.10.2023	1.7	HITeC eV D. Speck	Document revision, AI library architecture added
26.10.2023	1.8	HITeC eV K. Himstedt	Layout improved
27.10.2023	1.9	ARIC F.Vogt	Document revision, final review

Executive summary

The Research Computing Infrastructure (RCI) is an essential building block for working with customers of the European Digital Innovation Hub for urban interconnected supply and value Ecosystems[1] (EDIH4UrbanSAVE) to solve their compute-intensive problems with powerful hardware in a reasonable short time. Typical scenarios include the use of the RCI in the context of developing use cases and implementing proof-of-concepts for solving AI problems.

AI workloads in particular can easily place demands on computing power, such that they can no longer be solved in a reasonable amount of time (for example, not within a few days) using ordinary PCs. Here, the use of the RCI can enable even comparatively large workloads to be processed in a reasonable time without customers having to procure their own powerful hardware or – with the corresponding additional effort and costs – switch to cloud systems.

The architecture of the RCI, which is still under development, is initially based on the architecture of typical High-Performance Computing (HPC) clusters. A workload manager handles the concurrent accesses of the users to the resources of the cluster (CPUs, GPUs, ...). HITeC as a partner of EDIH4UrbanSAVE has procured the appropriate server components to set up the system: As a head/login node, one server provides users with access to the RCI and performs other central tasks, such as acting as a file server. Another server, which is equipped with four powerful NVIDIA L40 GPUs (Graphical Processing Units), takes on the role of the workhorse as a compute node in the cluster. At ARIC another partner of EDIH Hamburg, the procurement of essentially identical server components is planned in order to roughly double the computing power of the cluster system. While HITeC and ARIC work particularly closely together on the technical implementation of the RCI, the close cooperation with the partner DigiHub makes it possible to operate the RCI in two server rooms that DigiHub has made available for this purpose.

For the realization of first projects, it is planned that EDIH Hamburg experts in the field of AI and HPC assist the users in the use of the RCI. For a further development stage, it is planned that the users can also use the RCI on their own, for example via a web interface to access a JupyterLab development environment. There are several ways to manage the RCI resources for simultaneous use as a non-interactive and interactive system in a way that access conflicts are avoided. Which of the options is best suited for the RCI to achieve this goal is a major topic of the current research at HITeC. A version 2 of this deliverable is planned (due July, 2024) in which the selection of the most appropriate option will be described in more detail.

[1] In the following text, EDIH4UrbanSAVE is also referred to as EDIH Hamburg.

Table of contents

List of figures

List of tables

1. Introduction

In the service portfolio architecture of EDIH Hamburg, the Research Computing Infrastructure (RCI) is embedded at the third level "Enabling & Testing" of the "Test before Invest" pillar under the entry "Testing Infrastructure". The third level supports the measures and offerings of the first two levels, as shown in Figure 1.

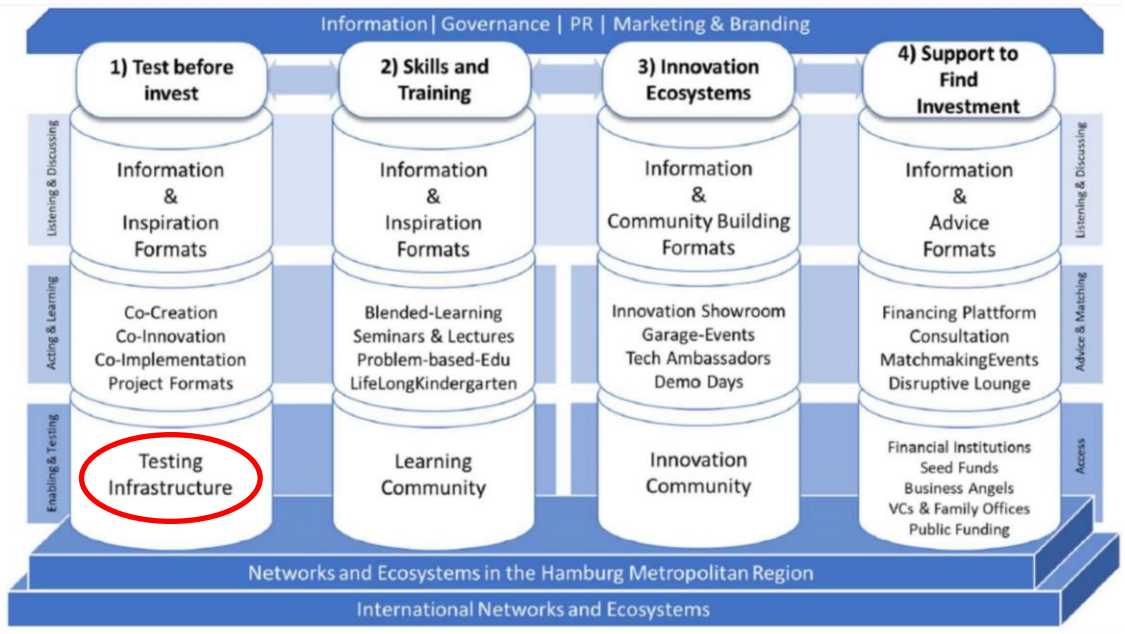

Figure 1: Embedding the RCI in the service portfolio architecture of EDIH Hamburg[2]

The target groups of EDIH Hamburg, i.e. Small and Medium sized Enterprises (SMEs) and Public Sector Organizations (PSOs), are to be supported in testing the technical feasibility of their digital transformation projects before making (further) investments by gaining access to appropriate testing infrastructures and relevant expertise. At EDIH Hamburg, the focus is on Artificial Intelligence (AI), Digitisation, Cybersecurity, and High-Performance Computing (HPC) – ADCH, which can be supported in terms of testing with cloud solutions, on premise solutions, access to tech labs etc. The RCI is an essential building block in this context, which is designed to support the testing especially in the fields of AI on dedicated hardware. As a rule of thumb, it can be assumed that such a stand-alone system that is used to some extent over a longer period of time is more cost-effective than purchasing the same computing power in the cloud. Compared to a cloud solution, the RCI as an on premise solution is also particularly well suited for dealing with data privacy concerns.

For the use of the RCI in the scope of the services offered by the EDIH Hamburg, the emphasis is on the execution of compute-intensive experiments (also see GA). A typical scenario is shown in Figure 2: After a corresponding use case has been developed with the customer, the customer needs the computing power of the RCI, e.g. to process large data sets or to validate a new AI method (also see path of yellow arrows). In another customer journey (not shown in

[2] Adapted from figure appearing in the grant agreement (GA)

Figure 2), it would be just as well conceivable that a customer would like to use the RCI as part of a Proof of Concept (PoC).

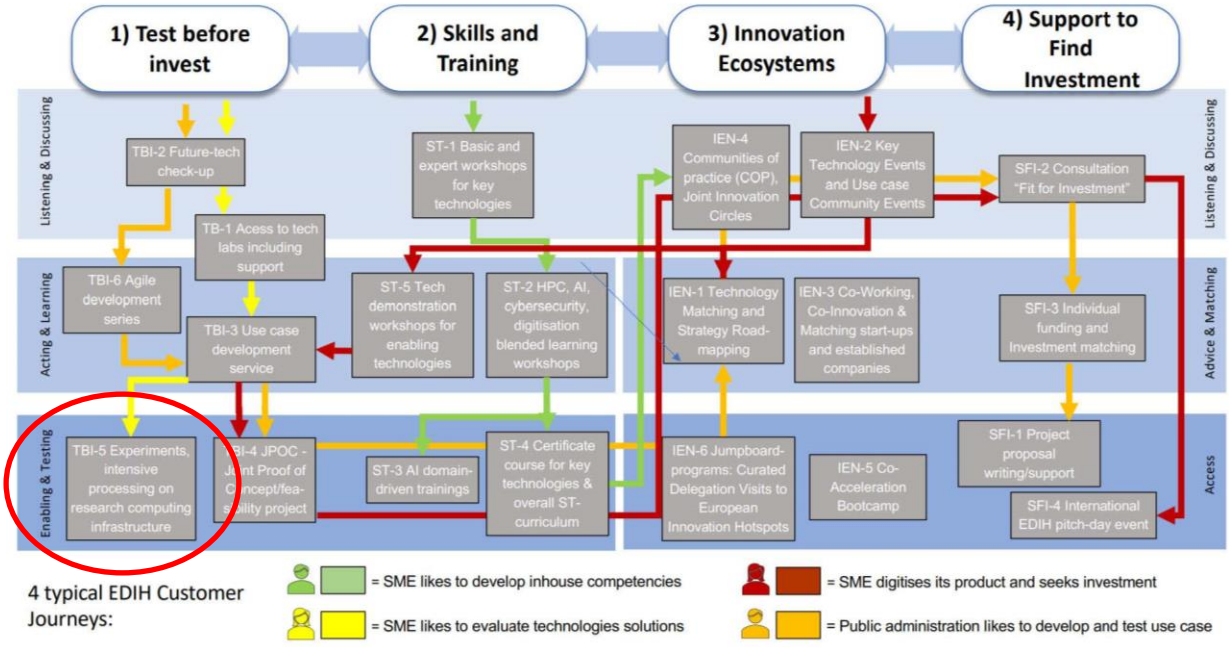

Figure 2: Examples for customer journeys[3] in the EDIH Hamburg: From use case development to the RCI

1.1 Scope and Objectives of this Deliverable

The RCI is still under development. A short overview of the typical requirements associated with HPC systems for AI workloads provides the basis for our hardware components selection and architectural decisions to date. The aim of this document is to present the current state of the technical implementation, as well as already planned extensions to the RCI. For the description of the technical implementation of the planned extensions, another deliverable with the same title will be provided in a version 2 (due July, 2024).

1.2 Structure of this Deliverable

Section 2 first gives an overview of the relation of the RCI to general HPC system architectures in Section 2.1. Section 2.2 notes the importance of GPUs (Graphic Processing Units) for the hardware architecture. Section 2.3 discusses various file systems used in the RCI. Section 2.4 briefly describes the collaboration of HITeC, ARIC, and DigiHub for the technical implementation and operation of the RCI. The placement of hardware components in the server rooms at DigiHub and their integration with the existing IT infrastructure at DigiHub is discussed in more detail in Section 2.5. Section 3 discusses the software architecture against the background of time sharing vs. batch systems and gives an overview of the AI Library Architecture being developed by HITeC for EDIH Hamburg. Finally, the considerations of the planned interactive and non-interactive usage features of the RCI are presented in Section 4.

[3] Adapted from figure appearing in GA

1.3 Intended Audience

This deliverable is aimed at three main audiences: 1) Consortium members, 2) the commission services and independent reviewers of the project, and 3) external organisations and participants of external projects, especially those with an interest in EDIHs. The primary audience are the first two groups, but as a public deliverable, its content is intended to be made available to other interested parties.

2. Research Computing Infrastructure Overview

For the customers of the EDIH Hamburg, it can be assumed that they usually just work with typical desktop PCs or notebooks, but do not have their own powerful hardware that would be suitable for processing compute-intensive problems. However, typical AI workloads such as Natural Language Processing (NLP) or Machine Learning easily require significantly more computing power than a conventional PC can provide in order to keep the waiting times until the result is available within reasonable limits (the goal is to reduce the time-to-solution). It is interesting to note that the computing power of a single core of a powerful PC is approximately equivalent to the computing power of a single core of an HPC system. Therefore, in classical HPC systems, a significant speedup cannot be achieved without some form of parallel computing. On the hardware side, several compute nodes – each roughly comparable to a very powerful PC – are connected as workhorses to form an HPC cluster.

Two types of using the cluster system can be distinguished: non-interactive and interactive. In the case of non-interactive use, which remains widespread in classic HPC for the efficient use of the expensive cluster resources, the compute jobs of the users, at this point described somewhat simplified, are placed in a queue and are processed one after the other, also overnight. For interactive use, users have quasi-immediate access to the powerful hardware and can interact with an application program in a similar way to using a PC. For the RCI, it is planned to support both application scenarios, which is a certain challenge. For the implementation to date, the focus was initially on the non-interactive usage of the RCI, in order to be able to cover scenarios such as those that are first to be expected in the context of EDIH Hamburg.

2.1 General System Architecture

A classical HPC cluster system is built out of a few to many servers that are connected by a high speed communication network.[4] The servers, typically mounted in racks, are called cluster nodes. HPC clusters use batch systems for running compute jobs. The nodes of an HPC cluster are named in accordance with their tasks. Essentially, these are a

- head node (or login/gateway node), on which the user can start its activities after login and which is used for administration purposes of the cluster, and
- compute nodes, as true workhorses.

In addition, there may be further nodes, e.g. to provide the global file system, i.e. a file system that can be used on all other kinds of nodes. In small cluster systems, a node can also take on several roles. A global file system is a central component of an HPC cluster, since it allows direct access to the same files from all nodes. This functionality is known from network file systems, but can go beyond that with the use of parallel file systems. On the one hand, the I/O performance can be increased with parallel file systems compared to classic network file systems and on the other hand they enable the simultaneous writing from different processes into the same file.

Another characteristic of HPC systems is that the nodes are connected via a fast communication network (e.g. InfiniBand[5] or at least 10 GbE (Gigabit Ethernet) instead of 1 GbE). On the one hand, this allows parallel programs running on several compute nodes of the cluster to exchange

[4] also see Kai Himstedt, Nathanael Hübbe, and Hinnerk Stüben, "Getting Started with HPC Clusters", introduction available at https://www.hhcc.uni-hamburg.de/files/pecoh-gswhc201912-text.pdf

[5] https://www.infinibandta.org/

data very quickly, and on the other hand, it enables high I/O performance of the file system. In practice, the communication network is designed against the background of the expected requirements and the available budget.

2.2 Hardware Architecture

The basic components described above also form the essential basis for the RCI. A special feature is that for some time now, especially for processing AI workloads, the computing power of CPUs[6] has been combined with the parallel computing power of GPUs (Graphical Processing Units) or GPGPUs (General Purpose Graphical Processing Units) to further accelerate the computations. Originally, GPUs were used for image processing and displaying images on screens. Then people started to utilize the parallel compute power of GPUs for other purposes and special GPUs were built for computing purposes only. The computing power of GPUs typically has the major impact in reducing the time-to-solution for AI workloads. Therefore, a major focus in the implementation of the RCI has been the procurement of powerful GPUs. The relevant details are given in Appendix A.2.

2.3 I/O Architecture

In HPC environments, the I/O architecture has a significant importance for the file systems used.

2.3.1 Local File System

A local file system can be provided on nodes that are equipped with disks or nowadays especially SSDs (Solid State Drives)/NVMes (Non-Volatile Memory express). Only the node itself has access to this local storage. Local file systems are useful for heavy scratch I/O to store intermediate results. The advantage of local disks/SSDs/NVMes is that I/O performance scales perfectly with the number of nodes. Because hard disks are a major source of failures many clusters have diskless nodes. However, experience at HITeC from other projects has shown that AI workloads in particular can benefit greatly from fast accesses to the local file system for storing intermediate results. For the RCI, it is therefore planned to also interconnect several very powerful NVMes to form a local RAID (Redundant Array of Inexpensive Disks) network, which can further increase I/O performance. For current NVMes, the risk of failure is considered low compared to the risk of failure of a conventional hard disk. Since only scratch data would be lost anyway, we expect the benefit of the high I/O performance to outweigh the possible disadvantages of a short-term outage to replace a failed NVMe.

2.3.2 Distributed/Network File System

A network file system is provided by a file server which is integrated into the cluster. The file system is available on all nodes. For example, on a Linux based cluster system, each user typically has a home directory that only he or she can access. A network file system now gives a user access to his or her home directory on all nodes. While all nodes can read from a distributed file system simultaneously without interfering with each other, caution is required with concurrent writes. In general, a file should only be written or modified by a single process at a time. A classical distributed file system is the Network File System (NFS). Network file

[6] Although CPU stands for Central Processing Unit, there is no central, i.e. single, processing unit any more. Today, all CPUs have multiple compute cores which all have the same functionality.

systems are not designed for (very) high I/O loads. For the RCI, the head node also takes on the role of the NFS server until otherwise specified. To improve reliability, the local SSDs or NVMes can be connected to form a RAID. As a further precaution, a (smaller) number of the available SSDs or NVMes can be configured as hot spare so that they are automatically activated in the event of a failure in order to seamlessly replace a failed SSD.

2.3.3 Parallel/Cluster File System

A parallel file system is a global file system like a distributed file system, i.e. it can be used on any node in the cluster. For example, on a Linux based cluster system, there is typically a work directory, that is available for performing the experiments and for storing the possibly larger (temporary) data sets with high I/O bandwidth. There is a twofold cluster aspect to this. First, the hardware itself is parallel, i.e., the file system is provided by multiple servers working in a federated fashion with each other. Secondly, parallel I/O accesses are made possible, i.e. several processes can write to the same file simultaneously in a consistent manner. For the RCI the file system BeeGFS[7] was chosen, as a hardware-independent POSIX (Portable Operating System Interface) parallel file system developed with a strong focus on performance and designed for ease of use, simple installation, and management.

2.4 Collaboration of the Partners HITeC, ARIC, and DigiHub

An advantageous feature of cluster architectures is that they can be easily expanded, for example, to increase the overall computing power of the system by adding more compute nodes as needed. This also applies to the RCI, which is an ideal prerequisite for the collobaration between ARIC and HITeC – the main partners responsible for the technical implementation of the RCI. HITeC has already procured and made operational central server components and ARIC is now planning to procure compatible server components that can be seamlessly integrated into the RCI as compute nodes. EDIH Hamburg partner DigiHub is providing significant support for the operation of the RCI by providing HITeC and ARIC with one server room each and the corresponding infrastructure (air conditioning, high speed interconnect of the server rooms based on fibre optics, high-speed Internet access, etc.) as colocation.

2.5 RCI Components: Planned Setup and Colocation

Figure 3 shows the hardware components of the RCI and the planned integration into the IT infrastructure at DigiHub.

[7] https://www.beegfs.io/c/

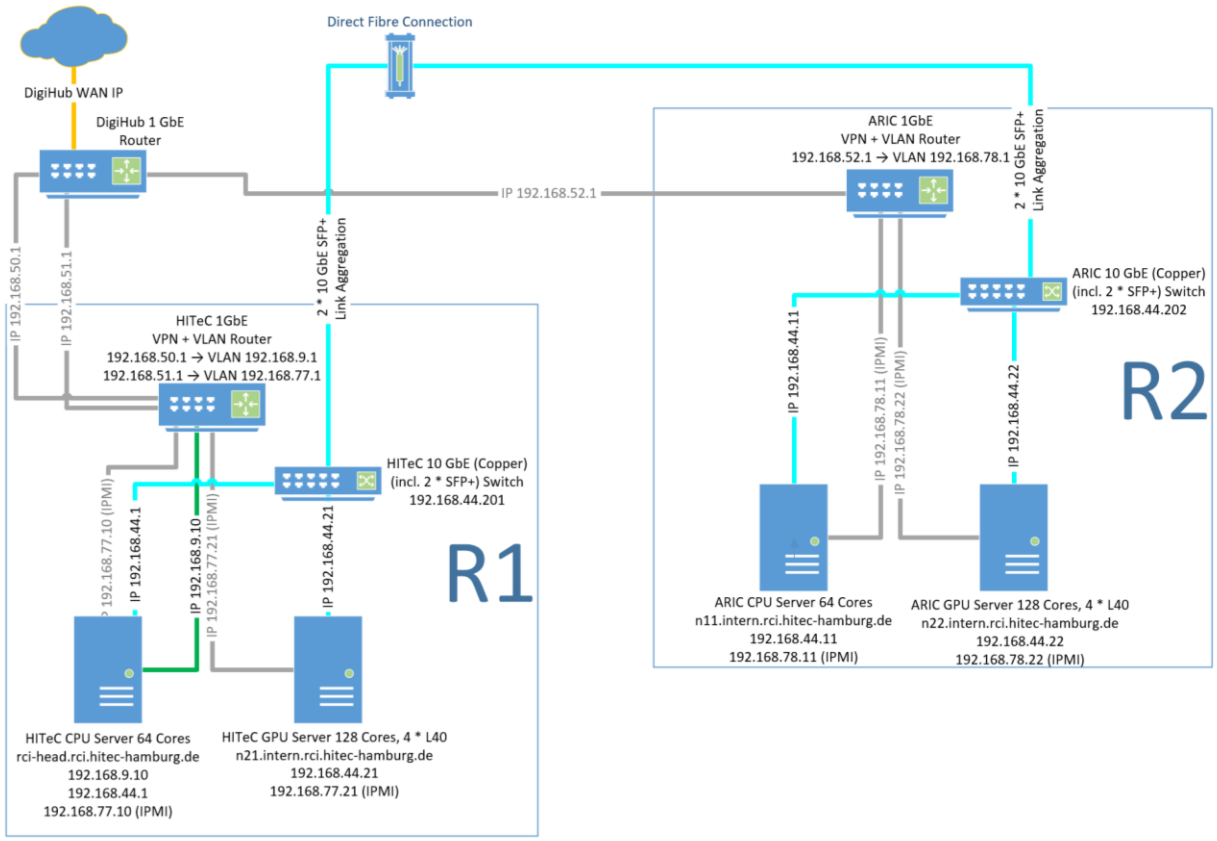

Figure 3: Placement and interconnection of the RCI hardware components at DigiHub (planned)

DigiHub operates small server rooms in the role of a server housing provider, which can be made available to DigiHub customers, e.g. startups, to run their own hardware. A central component of the infrastructure provided by DigHub is a broadband access to the Internet, which is shown in Figure 3 with the WAN (Wide Area Network) access in the upper left. Currently, a download bandwidth of 1 GBit/s and an upload bandwidth of 200 MBit/s is available there, which is shared by all users in all of the computer and server rooms. From a technical and organizational perspective, the operation of the RCI in the two DigiHub server rooms R1 and R2 is comparable to the operation of startups' server hardware in the other server rooms. DigiHub has the option to increase the upload bandwidth to 1 GBit/s at the WAN interface with the current Internet Service Provider (ISP) if a corresponding demand for this arises. The purchase of bandwidth from another ISP to bundle transmission capacities and improve reliability on the WAN side is also under discussion at DigiHub. For the time being, however, bandwidth bottlenecks are not expected with the existing WAN connection.

Since DigiHub currently only obtains a single global IP address from its ISP, all other IT components in the various server rooms are addressed on the LAN (Local Area Network) side via NAT (Network Address Translation). To improve security during the transmission of potentially sensitive data, the individual subnets of the startups and the RCI are divided into different VLANs (Virtual LANs). The VLANs are used to separate the data traffic of the different startups so that the confidentiality of the data is guaranteed. The Figure 3 shows that two separate VLANs are used to operate the RCI components procured by HITeC (with gateway IP addresses 192.168.50.1 and 192.168.51.1, each in a /24 subnet). This is for additional separation of the administration network for remote access to the server systems in R1 via BMC/IPMI (Baseboard Management Controller/Intelligent Platform Management

Interface) from regular access to the server systems. For remote access via BMC/IPMI to the server systems in R2, an additional VLAN (with the gateway IP address 192.168.52.1 in a /24 subnet) is in preparation. To further improve security, the administration network cannot be accessed directly via the web but only via a VPN (Virtual Private Network) connection. During the phase of the first launch of the RCI, the access for the users via the web is not yet enabled, but also takes place via a corresponding VPN connection. Within the DigiHub VLANs, a VPN and VLAN-capable router has been procured for this purpose, which in turn divides the administration network and the network for regular access via NAT into corresponding VLANs. This means that a typical router cascade with "NAT behind NAT" has been implemented.

The Figure 3 shows the cluster nodes procured by HITeC in R1. The HITeC CPU server takes over the role of the head node. The HITeC GPU server with its 4 powerful NVIDIA GPUs of type L40 is used as a compute node. For the access to rci-head and the access to the administration network the 1 GbE network seems to be sufficient. For the interconnection of the cluster nodes, a larger communication bandwidth offers a correspondingly larger potential, so that a 10 GbE network is used here. This fits well with the network infrastructure already in place at DigiHub: while within a single floor the IT components (horizontally) are usually connected via ordinary copper-based RJ45 network cables, server rooms on different floors (vertically) are connected via fiber optic cables and SFP+ modules. The switch in R1 procured by HITeC can thus first connect the two nodes to each other with a bandwidth of 10 GbE via (short) copper-based RJ45 cables. The rci-head node additionally takes the role of a NAT router to connect the subnets 192.168.9.x/24 and 192.168.44.x/24. This gives the compute nodes access to the services of the rci-head node such as DNS, NTP, NFS, to name a few. Furthermore, the compute nodes, which for defensive security reasons are not directly accessible from the outside via the DigiHub WAN, also gain access to the Internet. Since rooms R1 and R2 are located on different floors, the existing DigiHub fiber optic network is used for their interconnection. Using the link aggregation functionality of the switch, the bandwidth can be doubled by bundling two fiber optic connections if more than two nodes are involved in the communication of nodes in different rooms at the same time. For this purpose, no further active IT components of DigiHub need to be used, as it is described with the "Direct Fibre Connection" item in Figure 3.

In Figure 3, R2 shows in principle the compute nodes to be procured by ARIC. It is planned to design the server systems quasi identically or downscaled version to the two server systems in R1. In this case, the CPU server in R2 assumes the role of a compute node for classic HPC applications that cannot benefit from acceleration with a GPU. With the cluster architecture basis of the RCI, the throughput for processing AI problems can be doubled with the integration of the GPU compute node in R2, or the time-to-solution of a suitable application program can be halved in the ideal case with simultaneous utilization of a total of 8 GPUs for a specific problem.

At the current state of the hardware-technical implementation of the RCI, the components in R1 have meanwhile been completely procured and preliminarily set up at HITeC. For this test operation, the DigiHub WAN access was simulated at HITeC so that all other components could be configured as if they were already at DigiHub (IP addresses/ranges, VPN servers, VLAN IDs, ...). So far, the implementation is going so smoothly that the move of the hardware to the DigiHub server room R1 is currently being planned for October, 2023.

3. Software Architecture

The users of the RCI will potentially compete in a way for the expensive HPC resources of the cluster system. Therefore, an essential goal is to make the access to the computing resources efficient and fair. Appropriate software is required to implement this goal. The resources of the RCI can be distinguished as

- shared resources (e.g. a parallel file system that is shared across all cluster nodes and therefore shared between all users),
- not-shared resources (e.g. cluster nodes dedicated to a particular parallel program of an individual user).

The configuration of the cluster system matters as well: a compute node can also be a resource that is shared between several users. A major aspect of job scheduling is to manage these resources in a way that users are treated fairly. Accounting for users or user groups can later additionally support this.

3.1 Time Sharing vs. Batch Systems

A typical computer system like a PC provides the possibility for interactive communication between the user and the system. The user gives instructions via a keyboard and/or a mouse to the operating system or directly to a program, and the response is immediately displayed on the monitor. Such interactive systems are characterized by the users' desire for a short response time. In a typical cluster system such activities are performed simultaneously by a number of users on nodes explicitly provided for that purpose, e.g. on a head node. The Linux operating system (nowadays used in almost all cluster systems) provides a time sharing environment, such that all users logged in to a specific node (e.g. to the head node) have the impression of using a dedicated computer. With the 64 physical cores of the head node of the RCI, the computing power is dimensioned in such a way that the expected user demands in terms of interactivity in the startup phase of the RCI should be well covered. For further technical details of the hardware procured so far, see Appendix A.2. Nevertheless, it is important to keep in mind that the head node is a shared resource and all users are advised to use it carefully.

A batch job mainly consists of a program or a set of programs being processed by the batch system of the computer. One major feature of a batch system is the lack of interaction between the user and the job. The programs to run are typically given by a sequence of operating system commands in a batch file. The user submits the batch file to the batch system for execution. In a cluster system there are typically more batch jobs submitted than can be executed immediately to prevent (expensive) resources from being idle. Submitted batch jobs are queued for later execution by the batch system. Basically, running jobs on a cluster only begins to be meaningful when a problem cannot be solved using a desktop PC within a few days. Batch systems are used for executing large (parallel) jobs which need no user interaction. The RCI is designed to support batch operation.

Scheduling is the process of selecting and allocating resources to a job waiting for execution. Queues are used to handle this. As a consequence, waiting times will typically arise before a job is executed. Linux itself is not a batch system. Workload managers like SLURM[8] (Simple Linux Utility for Resource Management) are used to provide this functionality in a Linux cluster system. For the first configuration stage of the RCI we implemented the batch system based on

[8] https://slurm.schedmd.com/overview.html

the free and open-source SLURM job scheduler because it is almost a de-facto standard nowadays and has a good reputation. For the functions of the various services of the first configuration stage, please refer to Figure 4 in Appendix A.1.

3.2 AI Library Architecture

To enable a wide variety of users testing different use-cases and especially allowing for an efficient evaluation and further development on the RCI libraries are needed. We define a library here as a modular piece of software we develop to (1) be used for demonstration purposes, i.e., allowing EDIH Hamburg to demonstrate key AI technologies to users and partners, (2) enable users to test these technologies in a Web UI, (3) supply customizable features for rapid prototyping to speed up development of specific use-cases, (4) and ultimately to reduce redundancies and lower implementation and teaching costs on both, EDIH Hamburg and partner sides (prevention of "reinventing the wheel").

At the start of the AI library architecture, we defined KPIs (Key Performance Indicators) and other objectives like coding standards to be met to ensure the quality and maintainability of the code. The libraries build the foundation for all succeeding experiments and technologies to be built, thus the software architecture must be reliable, fulfil high requirements of robustness and maintainability. After the initial definition of all requirements, we identified the functional parts of our KPIs that needed development by EDIH Hamburg and other functionality that can be met by existing Open-Source solutions. For the Open-Source software, after a research-phase, we developed a ranking schema and ranked all found software regarding the age, number and activity of contributors, number and age of releases, usage and/or funding by other big Open-Source projects or organizations. Finally, we checked potential licensing issues. This research allowed us to identify all the Open-Source software that our AI library architecture can build upon.

For the actual implementation phase of the AI libraries a proper software architecture plan is needed. Apart from development meetings and code reviews of different proposal, we also evaluated the overall design patterns to be used for implementing the AI library and demonstration services. To document the design decisions in a developer-friendly way, we decided to build a UML (Unified Modeling Language) diagram of our container architecture that will be used for demonstration purposes and the AI libraries within enabling the actual AI technology. The UML diagram is shown in Figure 5 in Appendix A.3.

4. Using the Research Computing Infrastructure

At the moment the RCI is still under construction. For the use of the RCI for the realization of first projects with compute-intensive AI problems, it is planned that the SMEs and PSOs as users will be guided by corresponding experts of the EDIH Hamburg. Access can be provided via interfaces such as VPN tunnels and/or SSH (Secure Shell) connections. Users who have reached a sufficient level of maturity – for example, on the basis of participating in eLearning courses within the framework of EDIH Hamburg – can also work on their projects on their own and be given access to the RCI for this purpose. From the user's point of view, this is roughly comparable to the possibility of computing in a compute cloud. One advantage for the customer in using the RCI is that no effort is required to set up or operate the computing instances in the cloud.

In a further extension stage, customers can also be offered a web-based access – as proposed in the GA – to the RCI. JupyterLab[9], which is in strong demand today, can be provided as a development environment with so-called notebooks, for source code and data. The flexible JupyterLab interface allows users to configure and arrange workflows, e.g. in data science, scientific computing, and machine learning. The operation of a JupyterLab environment can initially be implemented using techniques such as Docker containers[10]. The Docker environment enables lightweight management of an application based on prepared images for which, most importantly, no effort is required to consider special operating system dependencies. With the availability of a Docker environment on the RCI, all applications in a Docker container run there "out of the box", so to speak. Kubernetes[11] (also named K8s) further enables automated deployment of containerized applications. During interactive development and for short (interactive) test runs with low resource requirements, the computing power of the head node can be used.

For the scheduling of compute-intensive applications, the existing batch system can be used. A challenge now is to create a seamless transition from short interactive test runs to long production runs with an automated fair scheduling of the batch system.[12] Classical SLURM based HPC systems, like the RCI, have a quasi fixed size (of compute nodes) available for processing an "infinite" workload. SLURM covers several typical management tasks, such as job queuing prioritization, job accounting, user access control to compute resources, and launching large-scale jobs. K8s, as a cloud-native system, on the other hand, supports the operation of long-running processes in terms of scheduling, for which a large number of "infinite" resources can be made available on demand. To bridge the gap between classical HPC and cloud-native workloads, there are three obvious approaches. From the point of view of the SLRUM basic system, these are

- *Over*: In this approach, the K8s functionality is embedded below the SLURM workload manager. The approach is typical for large HPC computing centers where SLURM is the core component for operations. SLURM manages all resources of the cluster and the K8s functionality is created only ephemerally within batch jobs.

[9] https://jupyter.org/

[10] https://www.docker.com/

[11] https://kubernetes.io/

[12] also see Tim Wickberg, "Slurm and/or/vs Kubernetes", SC'22, Slides available at https://slurm.schedmd.com/publications.html

- *Adjacent*: Both systems coexist on the same level. With the installation of the SLURM K8s scheduler plugin, SLURM supports the scheduling of SLURM and K8s workloads. K8s jobs have full access to K8s capabilities and SLURM jobs have full access to SLURM capabilities. There is a proof-of-concept in the form of a plugin[13], but so far it only supports node-level granularity for scheduling.
- *Under*: In this approach, K8s has sovereignty over the resources and the SLURM cluster runs – dynamically generated – within the K8s environment. There are hardly any restrictions on the use of SLURM from the user's point of view, but SLURM can no longer have any effect on the K8s workloads.

Which of the three approaches is best suited for the RCI – or whether possibly another approach may be particularly suitable for implementation – is the subject of current research at HITeC. The final decisions and results will be presented in version 2 of this deliverable with the same title (due July, 2024).

[13] https://gitlab.com/SchedMD/training/slurm-k8s-bridge

Appendix A

A.1 RCI Software Configuration (First Stage)

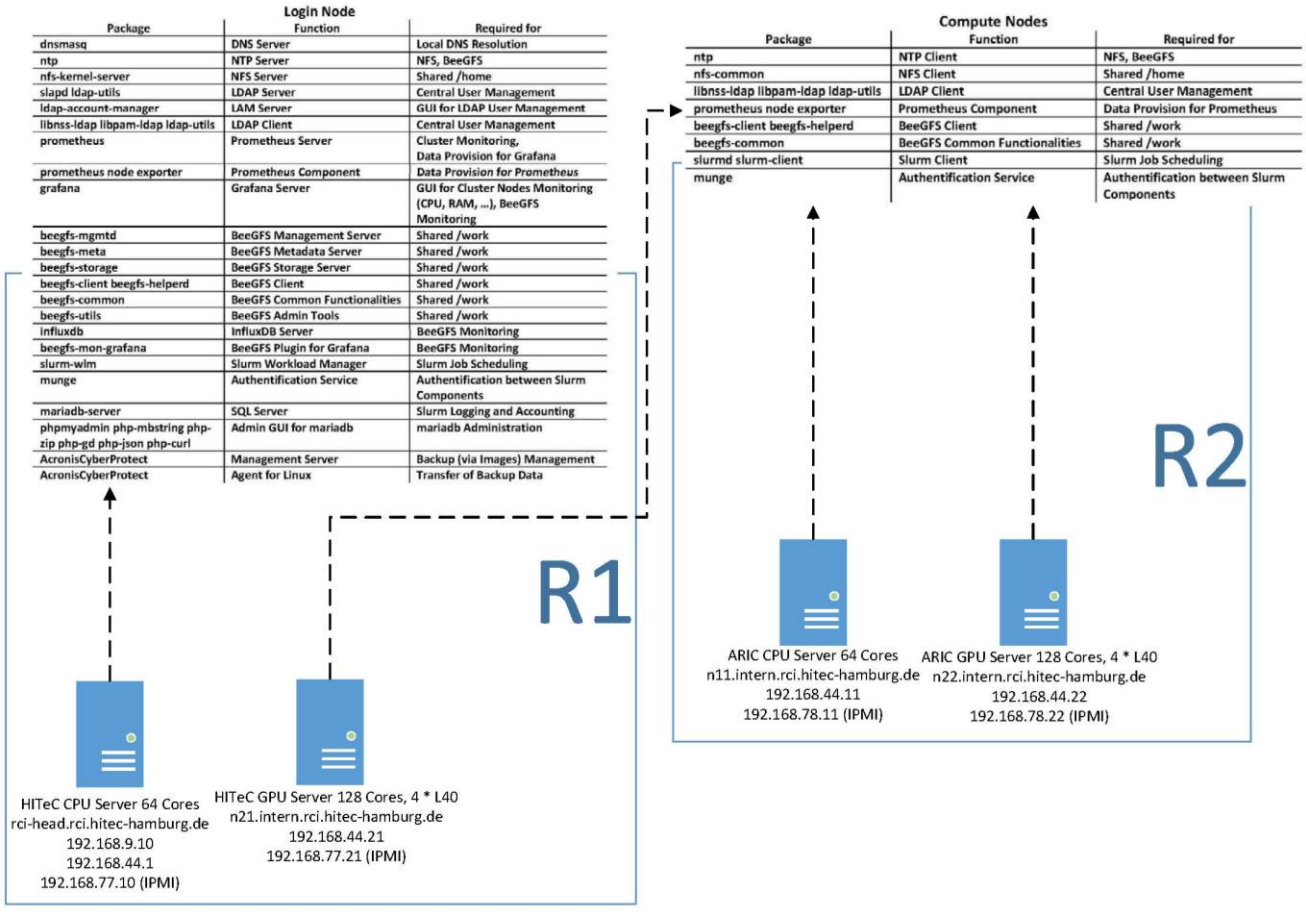

Figure 4: Services running on the login node and the compute nodes

The RCI software configuration (first stage) has been successfully tested in virtual environments and is currently being transferred to the physical hardware overlapping with the completion of this deliverable.

A.2 RCI Hardware Configuration (First Stage)

Table 1: RCI cluster nodes specifications.

Node Type	CPU	GPU	Memory	Storage
head/login in R1 (procured)	AMD 9554P, 64 cores	none	384 GB	12x 7.68 TB NVMe
compute (CPU) in R2 (procurement planned)	AMD 9554P, 64 cores	none	384 GB	12x 7.68 TB NVMe
compute (GPU) in R1 (procured)	2x AMD 9554, 128 cores in total	4x NVIDIA L40, each with 18,176 CUDA cores and 48 GB memory	384 GB	4x 6.4TB NVMe 8x 7.68 TB SSD
compute (GPU) in R2 (procurement planned)	2x AMD 9554, 128 cores in total	4x NVIDIA L40, each with 18,176 CUDA cores and 48 GB memory	384 GB	4x 6.4TB NVMe 8x 7.68 TB SSD

A.3 AI Library Architecture UML

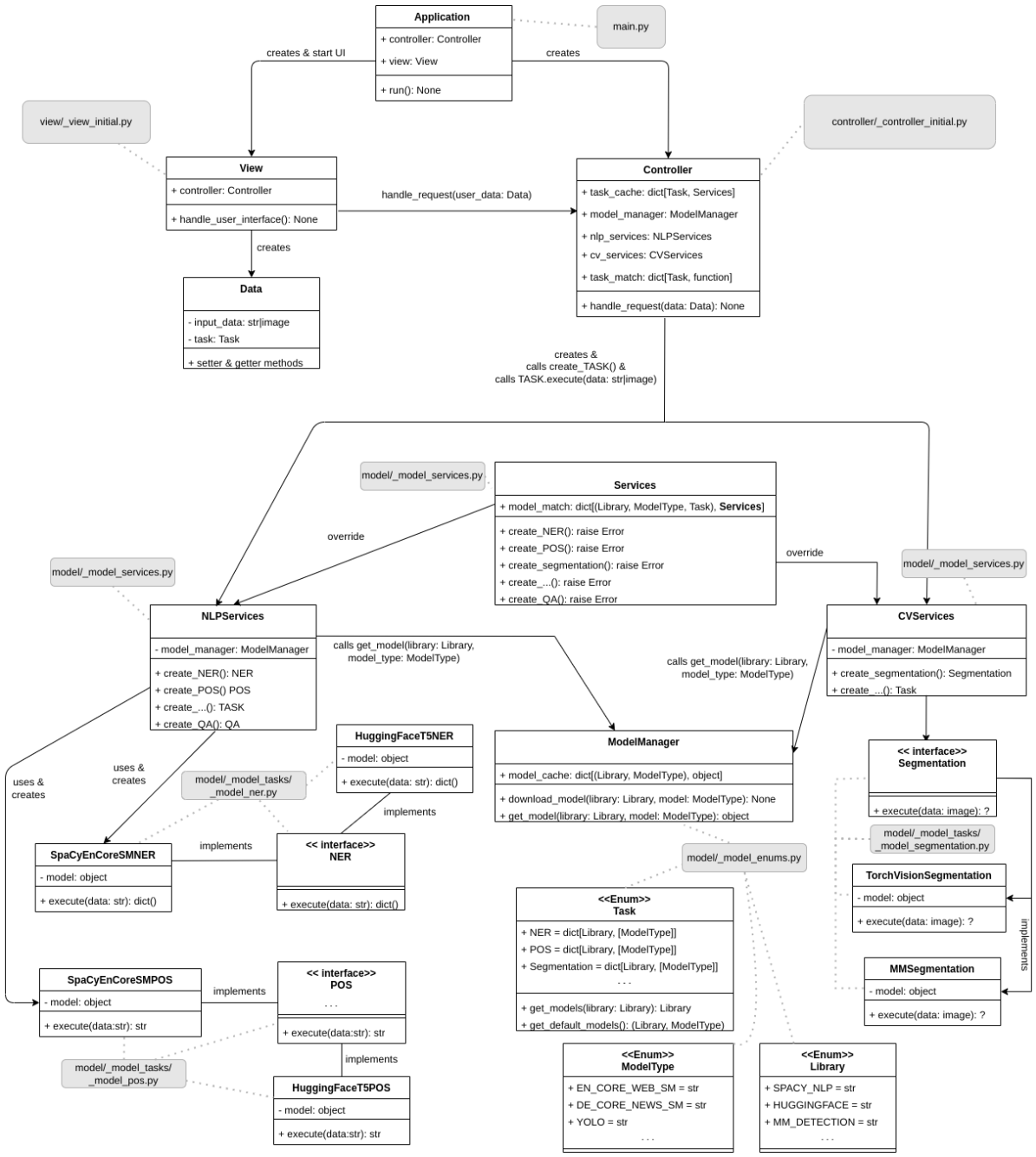

Figure 5: UML diagram of AI library demonstration container

This UML diagram describes the planning of the library for Natural Language Processing (NLP) and Computer Vision (CV) applications as well as the demonstration container application.

Glossary

Notions defined in this glossary are specific for this document only.

Notion	Meaning
AI	Artificial Intelligence
ADCH	Artificial Intelligence, Digitisation, Cybersecurity, and High-Performance Computing
ARIC	Artificial Intelligence Center Hamburg
BMC/IPMI	Baseboard Management Controller/Intelligent Platform Management Interface
BeeGFS	Bee Grid File System
CV	Computer Vision
DEM	DEM = Demonstrator, pilot, prototype
DigiHub	Digital Hub Logistics GmbH
DNS	Domain Name System
EDIH	European Digital Innovation Hub
EDIH4UrbanSAVE	European Digital Innovation Hub for urban interconnected supply and value Ecosystems
EU	European Union
GA	Grant Agreement
GB	GigaByte
GbE	Gigabit Ethernet
GPGPU	General Purpose Graphical Processing Unit
GPU	Graphical Processing Unit
HAW	Hochschule fuer Angewandte Wissenschaften Hamburg / University of Applied Science Hamburg
HITeC	Hamburger Informatik Technologie-Center
HPC	High-Performance Computing
HWK	Handwerkskammer Hamburg / Chamber of Crafts Hamburg
ISP	Internet Service Provider
KPI	Key Performance Indicator
LAN	Local Area Network
NAT	Network Address Translation
NFS	Network File System
NLP	Natural Language Processing
NTP	Network Time Protocol
NVMe	Non-Volatile Memory express
PC	Personal Computer
PoC	Proof of Concept
POSIX	Portable Operating System Interface
PSO	Public Sector Organisation
RAID	Redundant Array of Inexpensive Disks
RCI	Research Computing Infrastructure
SLURM	Simple Linux Utility for Resource Management
SSD	Solid State Drive
SSH	Secure Shell
SME	Small and Medium sized Enterprise
TB	TeraByte

TUHH	Technische Universitaet Hamburg / Hamburg University of Technology
UML	Unified Modeling Language
VLAN	Virtual Local Area Network
VPN	Virtual Private Network