FENIX

RESEARCH INFRASTRUCTURE

ICEI Infrastructure / Documentation for RFI Meetings

The ICEI project has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement No 800858.

Contents

1 Disclaimer

This document does not signify the beginning of a procurement procedure or constitutes a commitment by the public procurers involved in preparing this document to undertake such exercise at a later stage. Any technical goals, objectives, requirements or other details may be subject to changes.

2 Introduction

The European ICEI project is considered for being funded by the European Commission and is formed by the leading European Supercomputing Centres such as BSC (Spain), CEA (France), CINECA (Italy), ETH Zuerich/CSCS (Switzerland) and Forschungszentrum Juelich/JSC (Germany) with EPFL (Switzerland) acting as project coordinator.

The ICEI project plans to deliver a set of e-infrastructure services that will be federated to form the Fenix Infrastructure. The distinguishing characteristic of this e-infrastructure is that data repositories and scalable supercomputing systems will be in close proximity and well integrated. The European Human Brain Project will be the initial prime user of this research infrastructure. It will take care of developing the community-specific services on top of the Fenix infrastructure services. Part of the resources within the ICEI infrastructure will be provided to European researches at large through PRACE.

For the realisation of the ICEI infrastructure missing components need to be developed and existing technologies need to be enhanced. The results of the needed development services must be deployable, maintainable and supportable for the lifetime of the ICEI infrastructure. Due to the long-term focus of the infrastructure, sustainability of the developments beyond the project lifetime are of high interest for the project participants. The ICEI project currently estimates the value of the required development services to be EUR 1.8 million.

3 Nomenclature

Requirements (RQ)	Requirements are considered essential for the target solution and must be fulfilled by all final proposals.
Target Capabilities (TC)	Target Capabilities are desirable features and desirable performance levels for the procured system. Failure to provide Target Capabilities will not lead to the rejection of the proposals. Proposals that provide the Target Capabilities will receive a higher score. Target Capabilities are prioritized. Level-one priority Target Capabilities (TC-1) are considered of higher importance than level-two Target Capabilities (TC-2).

4 FURMS

4.1 Introduction

The Fenix User and Resource Management Service (FURMS) will be a one-stop-shop for users of the Fenix infrastructure. It will support the management of resource allocations for communities, projects, and users. At the same time, it will cover the related topics of accounting and resource provisioning for the Fenix resource providers.

FURMS will be based on a well-defined model of the stakeholders and assets involved, the details of which will be documented below.

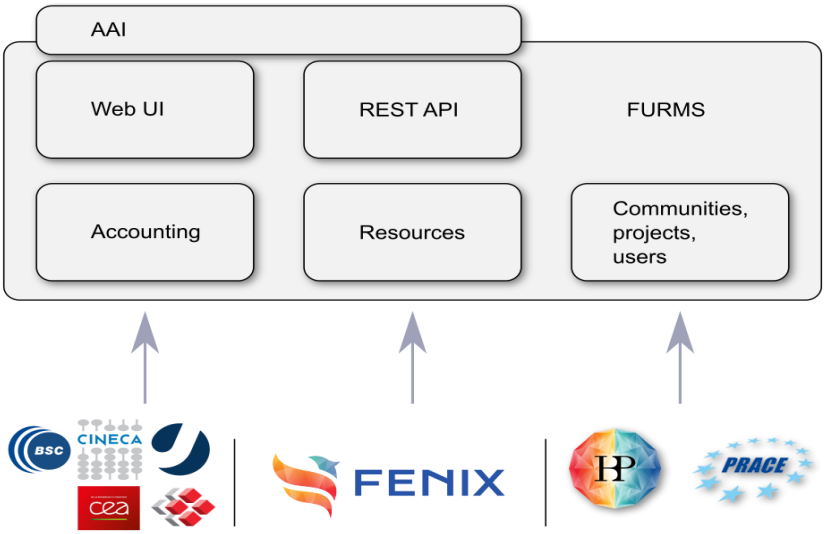


Figure 1: Schematic view on FURMS

4.2 Concepts

4.2.1 Fenix Community

From the point of view of Fenix a community is a virtual organisation of scientists that is entitled to use a given fraction of resources available within the Fenix Infrastructure. Initially two Fenix communities are foreseen: The Human Brain Project (HBP) as well as all PRACE.

4.2.2 Fenix User

A Fenix User is a person that has an identity provided by any of the Identity Providers (IdP) accepted within the Fenix AAI. Ordinary users will be associated with communities and projects, in which they can be members. Privileged users will have additional rights to manage Fenix infrastructure.

4.2.3 Fenix Project

A Fenix Community is responsible for distributing the resources, which have been allocated to this Community, on the basis of a peer-review process. Users have to apply within their community for resources. Once an application is approved, a Fenix Project can be instantiated.

Each Project belongs to a single Fenix Community and is managed by this Community. Privileged users of the community are able to create projects and allocate resources to them.

An optional extension to this could be nested projects that would help to further break down the distribution of resources among groups of users. As with a flat project hierarchy, projects at the top level would receive their share of resources from the community, subordinate projects would receive their share from the superior projects.

4.2.4 Fenix Site

Fenix sites provide resources of various types to Fenix for consumption by the communities, which in turn distribute the resources to their projects. Resources at a site can be consumed up to the amount provided by that site. The consumption will be reported back to FURMS for accounting purposes.

4.2.5 Fenix credits

Resources are made available by sites as Fenix credits. Fenix credits represent units of value for consuming specific resources at site. They comprise four attributes:

- Type of resource
- Amount of this resource
- Fenix resource provider (Site, at which it can be consumed)
- Validity Period

4.3 Objectives

The objectives of FURMS are:

- Manage resource allocations to communities and projects.
  - Fenix staff distribute resources contributed by participating centres to communities according to a specified distribution, i.e. 25% HBP, 15% PRACE.
  - Communities establish scientific peer-review processes, according to which privileged community members can distribute resources among the projects.
- Accounting
  - Consumed resources are reported by computing centres.
  - Aggregated information is available according to user privileges, e.g. privileged community staff can see which projects consumed how much of their budget.
- Integration with AAI
  - Information about groups and their memberships will be consumed from the IdP through the authentication protocol.
  - We envision FURMS to be capable of manipulating the central IdP and make use of its RESTful management interface.

4.4 Requirements and target capabilities

Requirements and target capabilities are derived from the objectives above are collected in Table 1.

Table 1: Requirements and target capabilities for FURMS

No.	Category	Description
1	RQ	All functionality must be available through a web based user interface. Specifically, this comprises: Resource allocation to communities and projects Visualization of consumed budgets
2	RQ	Integration with AAI. In particular, all end users must be able to authenticate using a federated identity that is registered with the central Fenix identity provider (IdP). This integration holds for both the Web based UI and RESTful interface.
3	RQ	Sites must be able to retrieve information about budget allocations from FURMS.
4	RQ	Sites must be able to send their accounting information to FURMS.
5	TC	FURMS actively informs stakeholders about budget running low or being entirely consumed.
6	TC	FURMS can handle a 100 (TBD) concurrent logged in users.
7	RQ	Authorization of users actions is based on attributes provided by the central IdP.
8	RQ	FURMS handles user agreements and their expiration. Users need to agree to site specific policies before using resources at any given site. The bookkeeping of „signed“ agreements is done by FURMS. Agreements have a lifetime of typically one year and need to be „signed“ again upon expiry. The information about the agreements the user has signed is stored as an attribute in the IdP.
9	RQ	There must be a RESTful interface used for automating certain tasks. Specifically, these will comprise the following: Information retrieval by sites about resource distribution. Regular generation of budget reports. Sites reporting consumed budget (accounting).
10	RQ	User landing page. As a one-stop-shop, users will turn to FURMS to get an overview of their Fenix infrastructure and their allocated resources. Also, they may need additional information to use specific services. For example, they may need to look up their local UNIX Id to use the SSH service at a particular site.
11	RQ	Sites must be able to retrieve updated information about budget allocations. This is required when communities decide to rebalance or replenish project allocations, e.g. from so far unallocated resources.
12	RQ	The provider will need to provide support of the deployed solution until March 2023. The scope of the support will be subject to negotiation but must at least

include security fixes as well as fixes for bugs that affect the contracted functionality of the service.

4.5 Use Cases

Use cases are grouped by stakeholders or levels at which the use case is relevant. For that matter, we have the site, Fenix, community, project, and user levels.

4.5.1 Site

4.5.1.1 Create resource pool

There are several pools of resources available at the Fenix sites, each representing a certain capability to be consumed by projects. In order to make available a new type of resource, a site representative creates a new pool.

4.5.1.2 Delete resource pool

Resource types may change over time. Therefore, it is desirable to eventually remove such pools from FURMS. This may have implications on the accounting, so it must be considered carefully in the model.

4.5.1.3 Update resource pool

A site representative updates the resources available in a given resource pool. This could be the case, for instance, when additional resource become available at the site.

In rare cases, resources may need to be removed from the pool. In such cases, the system needs to ensure that resources are only removed at most down to the amount already consumed.

4.5.1.4 Send consumed budget

Consumed budget needs to be sent to FURMS for further processing. This is done by an automated sender at each site.

4.5.2 Fenix

4.5.2.1 Create community

Communities are managed by Fenix representatives. Initially, there will be two major communities: HBP and PRACE. However, in order to keep the system dynamic, new communities will need to be created.

The actual creation of the community can and should be delegated to the central Fenix IdP.

4.5.2.2 Delete community

After ceasing activity, communities are eventually deleted from FURMS.

The actual deletion of the community should be delegated to the central IdP.

4.5.2.3 Update community

Update general community information.

The actual update of the information should happen in the central IdP.

4.5.2.4 Allocate resources (to community)

Allocate resources of a given type to a community. Resources can only be allocated from resources available in the resource pools provided by sites.

4.5.2.5 Retrieve consumed budget

Fenix representatives retrieve total budget consumed over a given period by any of the communities. The system automatically aggregates budget consumed by projects within these communities and only makes available the sum.

4.5.2.6 Accept user application

After a user has applied for membership within the Fenix infrastructure group, a privileged representative may accept or reject this application.

4.5.3 Community

4.5.3.1 Create project

At the community level, projects are managed by privileged community representatives. They are created upon successful evaluation of a project proposal following a scientific peer review process. At creation time, a principal investigator (PI) is associated with the project. This user (PI) will have special privileges to manage users within the project.

The actual operation should be delegated to the central IdP.

4.5.3.2 Delete project

Projects can be deleted after their granting period is over.

The actual operation should be delegated to the central IdP.

4.5.3.3 Update project

Here, some general project information may be updated. One case may be the addition of another PI.

The actual operation should be delegated to the central IdP.

4.5.3.4 Allocate resources (to project)

Allocate resources of a given type from the community budget to the project. Resources can only be allocated from the resources available in the community's resource pools.

4.5.3.5 Retrieve consumed budget

Privileged community representatives can retrieve information about how many of their resources have already been consumed. This includes a breakdown of the information by project.

4.5.3.6 Accept user application

After a user has applied for membership within the community group, a privileged representative may accept or reject this application.

4.5.4 Project

4.5.4.1 Add user

The PI (or anyone in that role) associated with the project can add additional members to it.

The actual operation should be delegated to the central IdP.

4.5.4.2 Remove user

The PI (or anyone in that role) associated with the project can remove members from it.

The actual operation should be delegated to the central IdP.

4.5.4.3 Update user permissions

The PI (or anyone in that role) associated with the project can update user permissions. For FURMS, we envision ordinary user membership as well as the PI role. Other services within the infrastructure may require additional roles that will also be managed here.

The actual operation should be delegated to the central IdP.

4.5.4.4 Retrieve consumed budget

Project members can retrieve information about how many of their resources have already been consumed. For PIs, this may include a breakdown by individual users.

4.5.4.5 Request resources

Communities may have a process for replenishing projects with additional budget that has not already been allocated to any other project.

4.5.4.6 Accept user application

After a user has applied for membership within the project, the PI (or anyone in that role) may accept or reject this application.

4.5.5 User

4.5.5.1 Apply for group membership

Users may register for membership within a particular project, community, or Fenix as a whole.

The actual operation should be delegated to the central IdP.

5 Data Mover Service

5.1 Introduction

The ICEI architecture foresees two different categories of data storage tiers: Active and archival data repositories. Active data repositories, for example high-performance parallel file systems, are not directly integrated in the federated data infrastructure and may be architecturally diverse. The archival data repositories are fully federated, enabling access to geographically distributed data through a SWIFT-based access scheme. The active data repositories will typically not be accessible from the scalable compute resources at the sites. Instead, users will copy data from the archival data repository to the active data repositories by means of a Data mover service, see Figure 2. In case the data becomes modified in the active data repository, the user must be able to trigger an update (replacement) of the copy in the archival data repository.

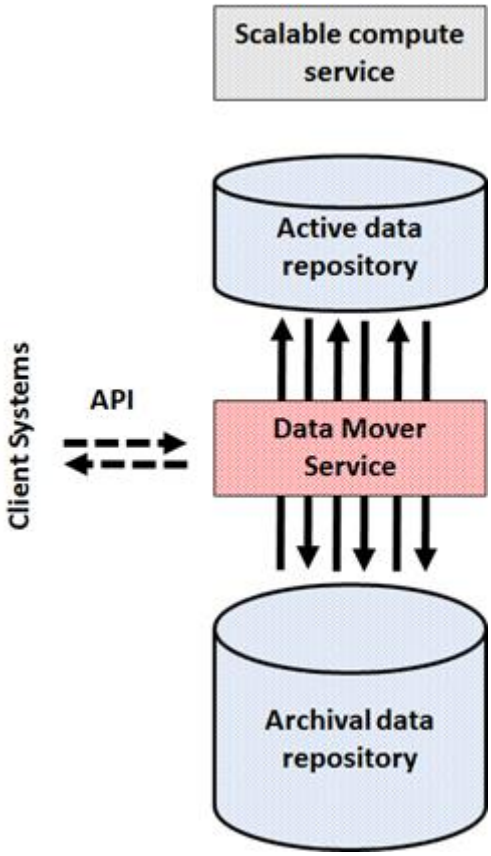


Figure 2: ICEI storage architecture abstraction

5.2 Objectives

The objectives of the data mover service are:

- The data mover infrastructure will be a programmable, secure high-performance service for the transfer between archival and active data repositories.
  - The data mover will transform user data between objects (archival data repository) and files (active data repository) and vice versa.
- A similar data mover service, with identical user interface, will be deployed at all ICEI sites.

    - The data mover infrastructures at different ICEI sites will likely be deployed on different hardware infrastructures in accordance with hardware guidelines developed in this project.
    - The data mover infrastructure should provide a basic feature set that can be utilized with minimal assumptions about the site infrastructure in order to enable an easy deployment of the service on new sites joining the federation.
    - The data mover infrastructure can take into account the specifics of the local infrastructure, in particular the specific technologies utilized for the active and archival repositories, for highest performance.
- The data mover service will cover the use cases described in Section 5.4
    - We note that support for the use case described in Section 5.4.2 may require development of additional features in the batch system.
- The data mover service should be built on sustainable components.
    - Where applicable, changes to open-source should be included in the "upstream" project.
    - For proprietary components, a business case must be provided that convincingly explains the long-term support and development prospect.
- The data mover service will be able to be integrated with a data location service by enabling the registration of data in the archival repository.
- Deployment and support of the solution at several, to be specified sites at least until the end of the ICEI project (March 2023).

5.3 Requirements and target capabilities

Requirements and target capabilities are derived from the objectives above are collected in Table 2.

Table 2: Requirements and target capabilities for the data mover services

No.	Category	Description
1	RQ	The outcome of the data mover development project will be ▪ A software stack; ▪ Design guidelines for hardware platforms. The data mover infrastructure will rely on commodity technologies (exceptions are possible only when duly justified). Different ICEI sites intend to deploy different hardware infrastructures in line with the design guidelines. The provider will need to provide support at different (to be defined) ICEI sites for five years. The scope of the support will be subject to negotiation but must at least include security fixes as well as fixes for bugs that affect the contracted functionality of the data mover service.

2	RQ	The data mover infrastructure will be designed for high-availability. To this end the underlying hardware and software infrastructure will feature little or no single points of failure. In addition, the data mover should be able to recover from hardware failures by restarting transfers that were interrupted due to internal hardware or software errors.
3	RQ	The data mover service will support archival data repositories offering object access through the OpenStack Swift Object Storage API.
4	TC	In the case where the archival data repository is based on an underlying file system (e.g., SWIFT on File on top of Lustre or IBM Spectrum Scale), the data mover service will additionally utilize the underlying file system for improved performance and/or an extended feature set.
5	RQ	The data mover infrastructure will support active data repositories based on an (almost) POSIX-compliant file system. All file systems currently utilized by the sites (see Section 6) must be support.
6	RQ	The data mover infrastructure will provide file system specific support for the active data repositories to the extent required to make the created data in the active data repository readily usable by applications on the scalable compute resources. The following file system specific settings need to be supported: ▪ Specific file striping configuration on Lustre.
7	TC	The support for specific active and archival data repository implementations is encapsulated in plugins that allow for easy adaptation of the service to other/new technologies.
8	RQ	The data mover software should provide support for registering hooks to be executed for specific events connected to data transfers. Among other use cases, this plugin mechanism can be utilized for the integration with a data location service.
9	RQ	The data movement service will provide operations with copy as well as move semantics. Data operations will perform the necessary (meta-)data transformation to convert between ▪ Containers and directories; ▪ objects and files; ▪ metadata items and extended attributes. Should access control lists be transferred, it should be performed in such a way that access allowances to data is never extended implicitly. Please note that the requirements regarding metadata and access control list handling are still under discussion.

10	RQ	A single data mover instance will support multiple separate archival and active data repositories.
11	RQ	The data mover infrastructure will be scalable, i.e., through an extension of the underlying hardware infrastructure the accumulated transfer bandwidth (multiple concurrent transfers) can be extended up to the bandwidth of the archival and active data repositories.
12	RQ	The data mover service will offer a REST web service API. The authentication mechanisms will be integrated with the token-based mechanism of the object storage API of the archival data repository.[DK1]
13	TC	The data mover service will offer an RPC-based API that relies on user IDs for authentication (utilizing technologies such as munge for credential verification).
14	RQ	Both data mover APIs (if applicable) will support asynchronous calls. Transfers will be represented by handles that enable users to query progress and status of ongoing transfers. The APIs will enable users to query their list of transfers and retrieve transfer information (success or error codes).
15	RQ	The data mover will log all transfers and provide per-transfer performance indicators for operators.
16	RQ	Command line interfaces for the data mover services will be available.
17	TC	The data mover service has an internal scheduling mechanism to maximize overall throughput by ordering of transfers. Please note that the data mover schedules data movement to improve transfer bandwidth. The data mover service does not keep track of active data repository utilization. The data mover will check for the availability of capacity prior to the execution of a transfer but does not guarantee the availability of resources throughout the execution of the movement.
18	TC	The data mover service provides QoS mechanisms to prioritize specific transfers (e.g., for the use case described in Section 4.3).
19	RQ	The data mover service should be able to manage a sufficiently large number of concurrent requests and outstanding transfers. Based on our current thinking, the following tentative numbers should be achieved: The data mover service can sustain at least 5,000 request per second and at least 10 million outstanding transfer requests.
20	RQ	The data mover service integrates with the batch system of scalable compute resources to enable the use case described in Section 5.4.2.
21	RQ	The data mover service can integrate with separate batch system instances for different systems.

5.4 Use cases

5.4.1 Manual data movement

Data movement is triggered manually by the user or by means of a workflow management system from frontend systems of the scalable compute resources or other server systems via the command line or the API. Since users also have access to the archival data repository directly from these systems, the focus in this use case is on asynchronous bulk large-volume transfer with few API invocations. Transfer requests may be issued at container/directory or object/file granularity level.

5.4.2 Data movement via the batch system

In this use case, the data movement between archival and active data repository is driven by the batch system. Prior to the job execution, a "stage in" step is performed by the data mover service that ensures the availability of necessary data in the active data repository. After the job execution, a "stage out" step can be performed to move generated data back to the archival data repository. Transfer requests may be issued at container/directory or object/file granularity level.

The specification of the data movement must be integrated in the batch job description in the form of annotations. This use case resembles existing mechanisms for the handling of burst buffer technologies. Note, however, that in in the presented use cases neither the batch scheduler nor the data mover service schedules active data repository capacity, i.e., the user is required to ensure availability of enough capacity reserves for the stage-in and stage-out destinations. The data mover will check for the availability of capacity prior to the execution of a transfer but does not guarantee the availability of resources throughout the execution of the movement.

5.4.3 Data movement from Applications

In this use case, applications utilize the data mover service directly to move data between archival and active data repository. To avoid waste of compute cycle such applications will be required to hide the data movement time via appropriate algorithmic mechanisms, e.g., pipelining. Nevertheless, this use case requires tighter guarantees regarding transfer times and will likely feature a pattern with a higher rate of transfer requests than the other use cases. Transfer requests in this use case will like be issued at the object/file granularity level.

5.5 Overview of relevant infrastructure components at the ICEI sites

The ICEI sites are using Lustre and IBM Spectrum Scale for the implementation of the active data repositories. The archival repositories at three sites are implemented with the cluster export services of IBM spectrum scales. Two ICEI sites will procure the archival data repository technology in the context of ICEI. Four of the five sites are using Slurm; the fifth site is likely going to transition to Slurm for one of the relevant compute platforms. Different security technologies are utilized at the sites. In particular, two ICEI sites are using Kerberos.

6 Improved SWIFT and S3 on top of OpenStack

6.1 Objectives

The ICEI architecture foresees two different categories of data storage tiers: Active and archival data repositories. Active data repositories, for example high-performance parallel file systems, are not directly integrated in the federated data infrastructure and may be architecturally diverse. The archival data repositories are fully federated, enabling access to geographically distributed data through a SWIFT-based access scheme. SWIFT will be a crucial component to access Object Storage that in ICEI nomenclature will mainly be the archival data repository that should provide an easy way to store long term data in a cost-effective and efficient way that may include cheaper media such as tapes.

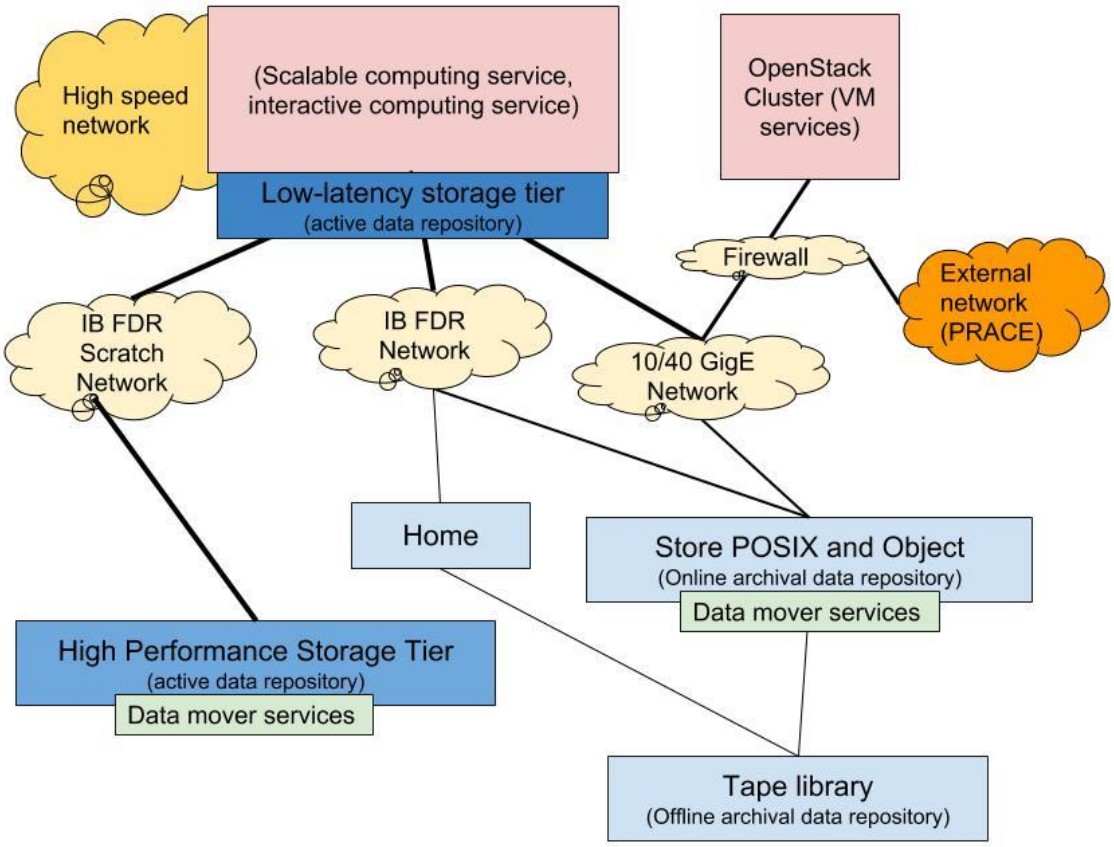


Figure 3: Example ICEI site infrastructure

The goal of this research and development effort is to implement a new set of APIs and features for SWIFT. The outcome of the R&D will consist of extending SWIFT enabling S3 workflow with existing or new features. A close collaboration with the OpenStack SWIFT community to submit reference implementation and to leverage standard would be an advantage.

6.2 Requirements

Requirements and target capabilities are derived from the objectives above are collected in Table 3.

Table 3: Requirements and target capabilities for improved SWIFT and S3 on top of OpenStack

No.	Category	Description
1	RQ	Release code integrated with the community Openstack release
2	RQ	The released features should be supported for at least the duration of the project
3	RQ	Add a user friendly interface to handle SWIFT ACLs
4	RQ	Availability of high-quality, full featured FUSE drivers like FuseS3.
5	RQ	Delegate a specific server or set of servers to operate on behalf of any user of a specific domain-project.
6	RQ	Wide availability of user-friendly GUI and Web GUI clients to handle Objects. Authentication of Openstack services through multi layer AAI ("more SAML/OAuth compliant").
7	RQ	Enable via GUI a simple user to list his own containers without being an operator
8	RQ	Improve SWIFT existing segmented uploads (large object support) in order to enable HPC workflow. Examples: If a transfer using segmented object last for too long it will hang and not be able to refresh the needed token. The same applies to recursive uploads (nested directories).
9	RQ	Make Swift upload resumable.
10	RQ	Implement rsync like functionality for object/containers like "s3rsync"
11	RQ	Improve Swift ACL management when domains are used. Example: Make possible to user User Friendly name instead of long ID.
12	RQ	Improve Swift ACL change. Add or delete without having to delete the previous and edit a new one.
13	RQ	Provide support for registering hooks to be executed for specific events connected to data transfers. Among other use cases, this plugin mechanism can be utilized for the integration with a data location service.

6.3 Use-Case Descriptions

The following use-cases will be addressed improving and adding new feature to SWIFT.

6.3.1 Enabling Collaboratory to use a federated AAI to access the SWIFT S3 interface

On cloud platform the most important feature is the ability to manage multi-tenancy.

In general, a tenant represents some organisational unit (legal entity) and it is the top level of all the authentication/permission management. The tenant is the responsible entity (at the legal level), and it is the one accountable. In our case, the top level tenant should be the HBP project. On Amazon AWS this is implemented by their IAM service.

The "best practice" are :

1) Create a "master account" tied to the tenant legal entity. This master account should be used only for the creation of the of other users.
2) Create some "super user" admin accounts tied to individuals persons. These admin accounts are used to manage almost everything
3) Create more individual users account with restricted privileges if needed
4) Create service accounts and roles for each project/environment

With IAM, you can create service accounts and roles for each platform/application improving the platform security with privilege separation for each resources.

For example you can create a role associated to a project (a group of VMs), and then create an S3 bucket with write permission associated to this role. At this point only the machine with the correct access rights will be able to write on this bucket.

Another point is that in general on cloud platform, individual user account are only used to manage resources. Every other thing is automated (headless, not interactive): this is why it's important to be able to create service accounts/roles.

It would be nice to be able to create such a workflow in Keystone/SWIFT.

Here below four use cases related to the Collaboratory/HBP:

6.3.1.1 Use Case 1

User JDC wants to transfer a large dataset (multiple files, 200 GB) securely from one HPC site to another without having to worry about site specific credentials. The protocol should be resilient to a transfer failure allowing transfer resume without starting over from the beginning:

- JDC logs in to a host (either at the source site, the destination site or a 3rd party) via a single sign-on solution.
- JDC issues a command via a CLI or GUI to initiate the transfer.
- The transfer can be performed in background and JDC is notified when the transfer is done or if it fails.
- In case of failure, JDC can resume the transfer by issuing a command via a CLI or GUI.
- During the file transfer, the partial file on the destination file is not accessible by any user, except possibly JDC in read mode.
- Ideally, the integrity of the data transfer should be checked automatically at the end of transfer.
- JDC can change ACL permissions on the destination site if needed via a CLI or GUI to share the dataset with other users.

6.3.1.2 Use Case 2

User JDC is working on a Jupyter notebook and he wants to access a large dataset (multiple files, 200 GB) located in the same data centre where Jupyter notebook is running:

- He wants to access the dataset with random read or write accesses.
- He wants to access the dataset as a POSIX filesystem in order to have good performance while using libraries that only supports POSIX-type access.
- JDC wants to be able to manage ACLs to possibly share the file with other Jupyter notebook users.

6.3.1.3 Use Case 3

User JDC has uploaded a large dataset (multiple files, 200 GB) to HPC storage and now wants to make dataset public:

- This dataset has to be made available to users outside HPC, anonymously, through one or several standard protocols.
- JDC signs on to the HPC site via a single sign-on solution and issues a command via a CLI or GUI to publish the dataset.
- JDC issues a command to list the datasets he has made public.

6.3.1.4 Use Case 4

As an HBP system administrator, Robert wants to manage services for the whole HBP project on the HPC sites. He wants to create service accounts and roles within the HBP master account for each platform/application which he manages in order to isolate these services. This will improve the platform security by implementing "least privilege" and "separation of roles" for each resource (VM, storage, network).

Examples:

- Robert creates a role associated to a project (a group of VMs), and then creates a storage bucket with write permissions associated to this role. Only the authorized VMs will be able to write to this bucket.
- The service accounts are only accessible by a limited set of scripts running on VMs at the HPC site. Each service account has its own key (e.g. SSH key, API key, ...). The scripts can access these service accounts without any human input.
- Robert has committed to use these service accounts only in this way, not to let them be used directly by a human user or by a user from outside the HPC site.

6.3.2 Improve Sysadmin interface and user experience

Another important use case is the limitation that sysadmin and normal user are experiencing while using SWIFT interface (GUI or not). The goal of this R&D is to improve SWIFT usability of both sides. Going in more detail even a simple admin user of a Swift container has a hard time managing SWIFT ACL or any admin operation due to a complete lack of user friendly approach.

Here a couple of scenarios:

1) To set an ACL. In case of domain usage the only way to properly set ACL to a specific user it to use a long not human readable ID instead of the human readable username that exist in SWIFT which is not usable in this kind of operation. The same applies to the most standard operation that will always requires tricks to get these ID.

In order to modify an ACL an admin has to delete the previous one and add a new one instead of making a little edit operation

7 Improved tape usage on top of SWIFT

7.1 Objectives

The ICEI architecture foresees two different categories of data storage tiers: Active and archival data repositories. Active data repositories, for example high-performance parallel file systems, are not directly integrated in the federated data infrastructure and may be architecturally diverse. The archival data repositories are fully federated, enabling access to geographically distributed data through a SWIFT-based access scheme. SWIFT will be a crucial component to access Object Storage that in ICEI nomenclature will mainly be the archival data repository that should provide an easy way to store long term data in a cost-effective and efficient way that may include cheaper media such as tapes.

The goal of this research and development effort is to implement a new set of APIs and features for SWIFT in order to enable the following components:

1) HSM feature for SWIFT
2) Periodic backup feature for SWIFT

7.2 Requirements

Requirements and target capabilities are derived from the objectives above are collected in Table 4.

Table 4: Requirements and target capabilities for tape usage on top of SWIFT

No.	Category	Description
1	RQ	Release the object HSM (SwiftHLM or similar) community feature as part of a supported product that will have official support for at least the duration of the project.
2	RQ	Enable Object HSM at object level.
3	RQ	Enable the Backup function of an object across the cloud concept in order to protect user from losing data
4	RQ	Enable backup restore at object level
5	RQ	Objects can be restored without invoking a multisite replica restore
6	RQ	Daily incremental Backup can be performed even in presence of millions of objects
7	RQ	HSM has to be a functionality that could be invoked also by a standard user
8	RQ	Backup could to be a functionality mainly restricted to privileged users

7.3 Use-Case Descriptions

7.3.1 HSM

HSM has been one of the most used features to enable a real user friendly experience for non-storage expert to benefit of cheap media such as tape when access latency is not crucial for the workflow.

In ICEI we clearly foresee this as an important feature in order to address the massive amount of data in the Archival storage. The use case we have in mind is a user that will want to store data for a long time on a low latency media and retrieve it when it will be needed. The best scenario will provide that functionality via web portal or in an automated fashion based on some configuration parameter such as access time.

7.3.2 Backup

There is a clear deficit with respect to any other data services that many HPC centres provide to their user community. There is no concept of standard backup as we know it. There is a concept of versioning that could be an interesting starting point but still it does not fulfil the backup safety criteria that an HPC user is used to have. In order to make the object storage service safer the HPC centre is in need of a comfortable way to ensure their user that each data that goes into the Object Storage will have a safe version such there are used with POSIX file systems with a file granularity and a daily frequency. Here the general use case to satisfy is the possibility for the sysadmin to tune a backup of each object based on tuning that could be following the base centre rules or the specific user/contract agreement.

8 Improved SWIFT and S3 on top of Lustre

8.1 Goals

The ICEI architecture is based on two different categories of data storage tiers: Active and archival data repositories. These tiers can be hosted on different type of storage systems like parallel file systems (Lustre, GPFS ...), archival systems (HPSS, IBM Spectrum, ...) or object store data warehouse. ICEI project choose to use a common interface to exchange data between all these systems: SWIFT.

To promote the use of open source solutions and avoid a vendor locking architecture, ICEI sees a need for developing an optimized SWIFT service over Lustre, an open source parallel file system used by ICEI partners. The goal is to study, develop the needed changes in the software and to work with different communities to integrate these changes to the main trees of these open source projects.

8.2 Use-Case Descriptions

The following uses-cases are addressed by this R&D project (from two point of view, sys-admin and end user):

- Sys-admin: Set up an Openstack/SWIFT environnement based on a standard Lustre cluster based on a best-practice guide
- Sys-admin: Set up an Openstack/SWIFT environnement based on a standard Lustre cluster with a deeper integration of OpenStack and Lustre
- User: Use Lustre as a SWIFT backend to manage objects (put/get/delete)
- User: Set security rules and check they are endorsed
- Sys-admin: Set quotas in SWIFT and check limits are mandatory

8.3 Statement of work

The following topics will be addressed improving and adding new features to SWIFT over Lustre.

8.3.1 Document Openstack usage of Lustre for SWIFT

Swift already supports Posix backend which is the basic Lustre interface. The goal of this effort is to make a best practice document on how to implement OpenStack/SWIFT over Lustre and how to map SWIFT mechanism over Lustre feature.

8.3.2 Lustre integration in SWIFT

Implement Lustre integration in Openstack, so Openstack users can easily deploy a Lustre cluster without a strong knowledge of Lustre administration. This is mainly the development of a “Lustre backup driver” for SWIFT.

8.3.3 Unify view between SWIFT and Lustre

The objective is to be able to map SWIFT namespace and Lustre namespace with a coherent access model (ACL based and quotas) and also to support:

- SWIFT segmentation (ideally no object size limit)
- SWIFT versioning
- SWIFT multi-region active-active replication

8.4 Requirements and target capabilities

Requirements and target capabilities are derived from the objectives above are collected in Table 5.

Table 5: Requirements and target capabilities for an improved SWIFT on top of Lustre

No.	Category	Description
1	RQ	Best practice document on how to setup SWIFT over Lustre
2	RQ	Best practice document on how to map SWIFT feature over Lustre feature
3	RQ	Best practice document on how to tune Lustre and SWIFT
4	RQ	Design of a Lustre backup driver
5	RQ	Implementation of a Lustre backup driver
6	RQ	Publication and review integration of Lustre backup driver
7	RQ	Design of unify view between SWIFT and Lustre and list of required changes in Lustre
8	RQ	Implementation of unify view: ACL
9	TC-1	Implementation of unify view: Quotas
10	RQ	Implementation of unify view: segmentation
11	TC-1	Implementation of unify view: versioning
12	TC-2	Implementation of unify view: multiregion
13	RQ	Publication and review integration of Lustre and SWIFT patches

8.5 References

[1] The Lustre community website, http://lustre.org/

[2] The SwiftStack S3 API support web reference page, https://www.swiftstack.com/docs/admin/middleware/s3_middleware.html

9 Efficient use of the interactive computing services

9.1 Introduction

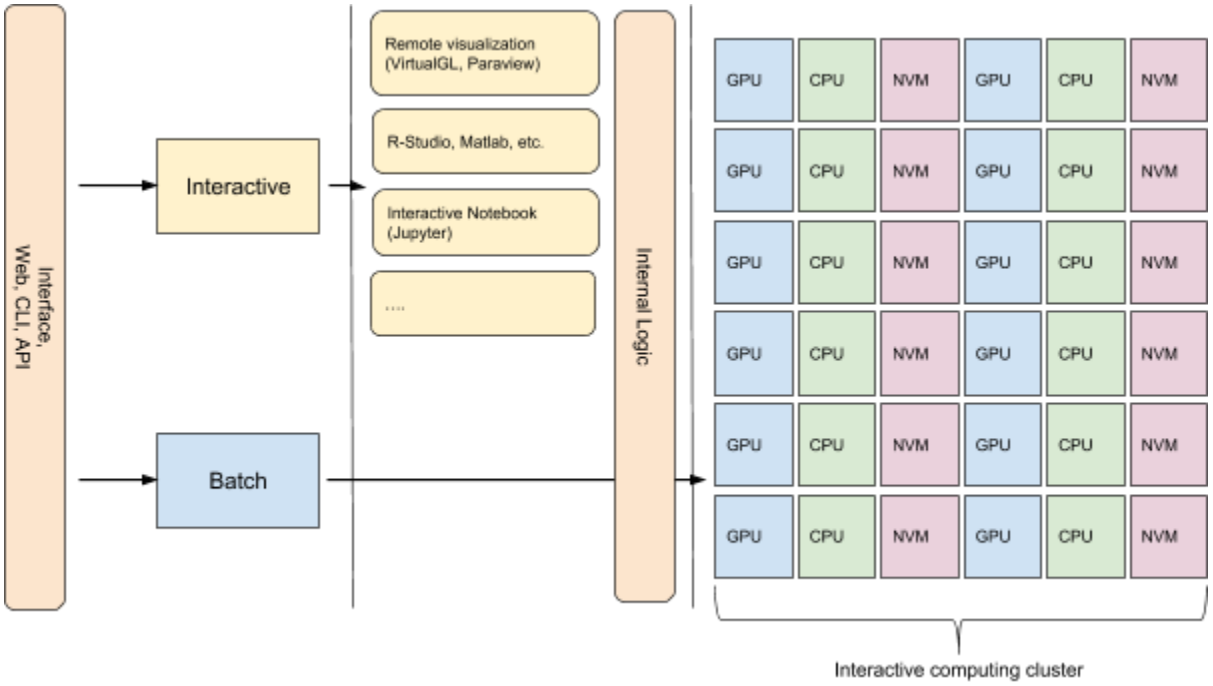
Nowadays, the main usage model for large scale HPC systems is based on the scheduling of batch jobs. Although this approach is the most adopted one, it does not originate from computational requirements, but it reflects the need to maximize the utilization of very expensive computational resources. Conversely, the situation differs when taking into consideration personal workstations or shared memory servers, where timesharing interactive executions are the norm. Interactive Computing refers to the capability of a system to support distributed computing workloads while permitting on-the-fly interruption by the user. The real-time interaction of a user with a program runtime is motivated by various factors, such as the need to estimate the state of a program or its future tendency, to access intermediate results, and to steer the computation by modifying input parameters or boundary conditions. The commonly agreed central components of interactive computing are, on the front-end, a sophisticated user interface to interact with the program runtime and, on the back-end, a separated steerable, often CPU and memory consuming application running on an HPC system. A typical usage scenario for interactive computing regards the visualization, the processing, and the reduction of large amounts of data, especially when the processing part cannot be standardized or implemented in a static workflow. The data can be generated by simulation or harvested from experiment or observation; in both cases, during the analysis the scientist performs an interactive process of successive reductions and production of data views that may include even complex processing like convolution, filtering, clustering, etc. This kind of processing could be easily parallelized to take advantage of HPC resources, but it would become clearly counterproductive to break-down a user session into separate interactive steps interspersed by batch jobs as their scheduling would delay the entire execution degrading also the user experience. Besides that, in many application fields, the computational scientists are starting to use interactive frameworks and scripting languages to integrate the more traditional compute and data processing application running in batch, e.g. the use of R, Stata, Matlab/Octave or Jupyter Notebook just to name a few. In this way, the time spent in this activity is a non-negligible component in the “time to science”. In that scenario, users will be presented with the possibility to load and visualize data resulting from simulations or collections, series or experimental data, in an interactive way. To properly support interactive supercomputing, the system needs to encompass all the connections and protocols needed to dynamically attach multiple visualization and steering front-ends to a running application and to enable transparent staging of data across multiple storage tiers, from distributed storage participation to node-central memory, to ensure front-ends receive a continuous stream of data. The Interactive Computing Services will be based on high-end servers with the following characteristics:

- Dual-socket processor configuration;
- High-end volatile memory configuration with a capacity of at least 256 GByte;
- Optional integration of non-volatile memory, which provide interfaces that facilitate high performance access to this memory (e.g. NVMe or DDR-like interfaces);
- Optional GPUs for visualisation; and
- Integration into a high-performance interconnect to facilitate fast access to Active Data Repositories as well as the Scalable Computing Resources.

9.2 Possible R&D activities

The Interactive Computing service will provide access to computational resources via interactive sessions, including visualization of large data sets, computing via notebook, data manipulation

and post-processing, etc. In order to be effective, this service would need to facilitate users' interaction with the infrastructure, support efficient handling of interactive sessions, maximize resources utilization while executing different workloads, support staging of data across multiple memory and storage tiers, improve energy consumption. To implement this goal, this R&D activity should work to develop a solution to provide users with a single interaction mechanism, i.e. Web, CLI, API capable to handle different utilization requests, from the submission of batch jobs, the creation of interactive visualization sessions, to the transparent set up of notebook environments, i.e. Jupyter Web Server.



9.3 Requirements and target capabilities

Requirements and target capabilities are derived from the objectives above are collected in Table 6.

Table 6: Requirements and target capabilities for efficient use of interactive computing services

No.	Category	Description
1	RQ	Permit the contextual execution of interactive sessions and batch jobs on the same infrastructure balancing the use of the resources on the base of pre-defined utilization policy, i.e. during the normal working hours interactive sessions may have the priority over batch jobs
2	RQ	Dynamically free resources when necessary, i.e. to guarantee or increase resource capacity for interactive sessions. This may include: ▪ suspension of running batch jobs, ▪ flushing of NVM drive, ▪ co-allocation of resources, ▪ exploitation of overbooking strategy,

		▪ automatic adaptation of resources allocation. Initial resource requests for interactive sessions may require adjustments over the duration of the session to better match workload requirements, i.e. increasing of memory
3	RQ	Ensure interactive sessions are spawn in a reasonable amount of time, order of magnitude is minutes;
4	TC	Guarantee simplified, efficient and scalable interactive sessions set-up, i.e. Web Service, SSH tunnelling, etc.
5	RQ	Support secure mechanism for users authentication based on standard protocols, i.e. Open ID Connect, SAML, etc.
6	RQ	Support the execution and/or the orchestration of containers;
7	RQ	capable to interoperate with the scale-out partition to enable computing steering;
8	TC	Ensure good performance for remote visualization sessions;
9	RQ	The outcome of this development project will be ▪ Software components; and ▪ Design guidelines for hardware platforms. Different ICEI sites intend to deploy these software components on different hardware infrastructures in line with the design guidelines. The provider will need to provide support at different (to be defined) ICEI sites until March 2023. The scope of the support will be subject to negotiation but must at least include security fixes as well as fixes for bugs that affect the contracted functionality of the service.

9.4 Expectations

- Specification of a scalable, HW agnostic, solution (possibly based on Open Source components) to maximize users interaction with the Fenix Interactive Computing Services
- Modular software package capable to meet requirements, compatible with Fenix systems and specification in terms of users authentication and authorization
- Training and documentation about developed software package(s)
- Maintenance/support services for whole project duration
- Long-term sustainability plan going beyond the end of the project