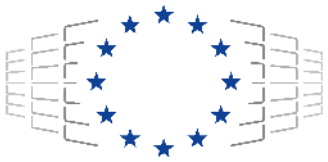
The IO-SEA Software Stack

Philippe DENIEL (CEA) – philippe.deniel@cea.fr

with the help of the IO-SEA WP leaders: Eric Gregory (e.gregory@fzj-juelich.de), Philippe Couvée (philippe.couvee@eviden.com), Céline Lemarinier (celine.lemarinier@eviden.com), Sébastien Gougeaud (sebastien.gougeaud@cea.fr), Maike Gilliot (maike.gilliot@cea.fr)



EuroHPC Joint Undertaking

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955811. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, the Czech Republic, Germany, Ireland, Sweden, and the United Kingdom.

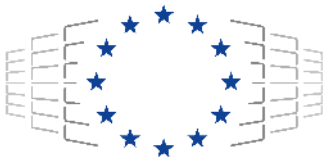
What you'll find here

This training covers the different topics

- A quick big picture about mass storage systems in the Exascale era and related issues
- The motivations behind the IO-SEA project and the chosen solutions
- Focuses on the produced software solutions
  - Phobos and the HSM feature
  - Workflow and ephemeral services
  - The DASI API
  - Available ephemeral services
  - Importance of monitoring

IO-SEA

IO challenges in the Exascale Era



EuroHPC
Joint Undertaking

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955811. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, the Czech Republic, Germany, Ireland, Sweden, and the United Kingdom.

The Exascale IO Challenges

The Exascale era brings new challenges or make some know situations even harder to manage

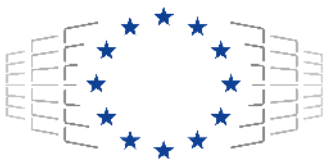
- **Challenge 1:** the volume of data and metadata will increase a lot
  - Future storage systems will have to handle up to ~1-10 EB and ~10-100 billions of inodes
- **Challenge 2:** The size of the client system will increase a lot
  - Client exascale system may contains up to ~10 000 – 100 000 nodes/clients
- **Challenge 3:** Inversion of the memory/number of cores ratio
  - Less memory is available to run the system stack, GPU technologies increase this trend
- **Challenge 4:** Non locality of the client systems
  - Supercomputer have complex network topologies, but “distance” has impact on performances
- **Challenge 5:** data heterogeneity and storage resources heterogeneity
  - Different storage media: SSD, HDD, Tapes, each with advantages and drawbacks
  - Different use-cases: from IA to High Energy physics, many different profiles and IO behaviors

New paradigms for IO and storage

The Exascale era brings new challenges or make some know situations even harder to manage

- **Object Stores:** they have many advantages
  - Objects are fully independent from each other
  - The CRUD semantics is simple and scales well
  - Object Stores are to be used with Key-Value Store (KVS) for metadata management
- **Smart Data Placement**
  - Hierarchical Storage Management can be easily extended to Object Stores
  - AI framework can be used to advise about better placement of data
- **Ephemeral Services and Data Nodes**
  - Data Nodes are identified in supercomputers's "islets", close to compute nodes
  - Storage servers, dedicated to compute jobs, run on this nodes: the ephemeral services
  - Ephemeral Services are associated to computes jobs, they start and end with them

The IO-SEA software stack

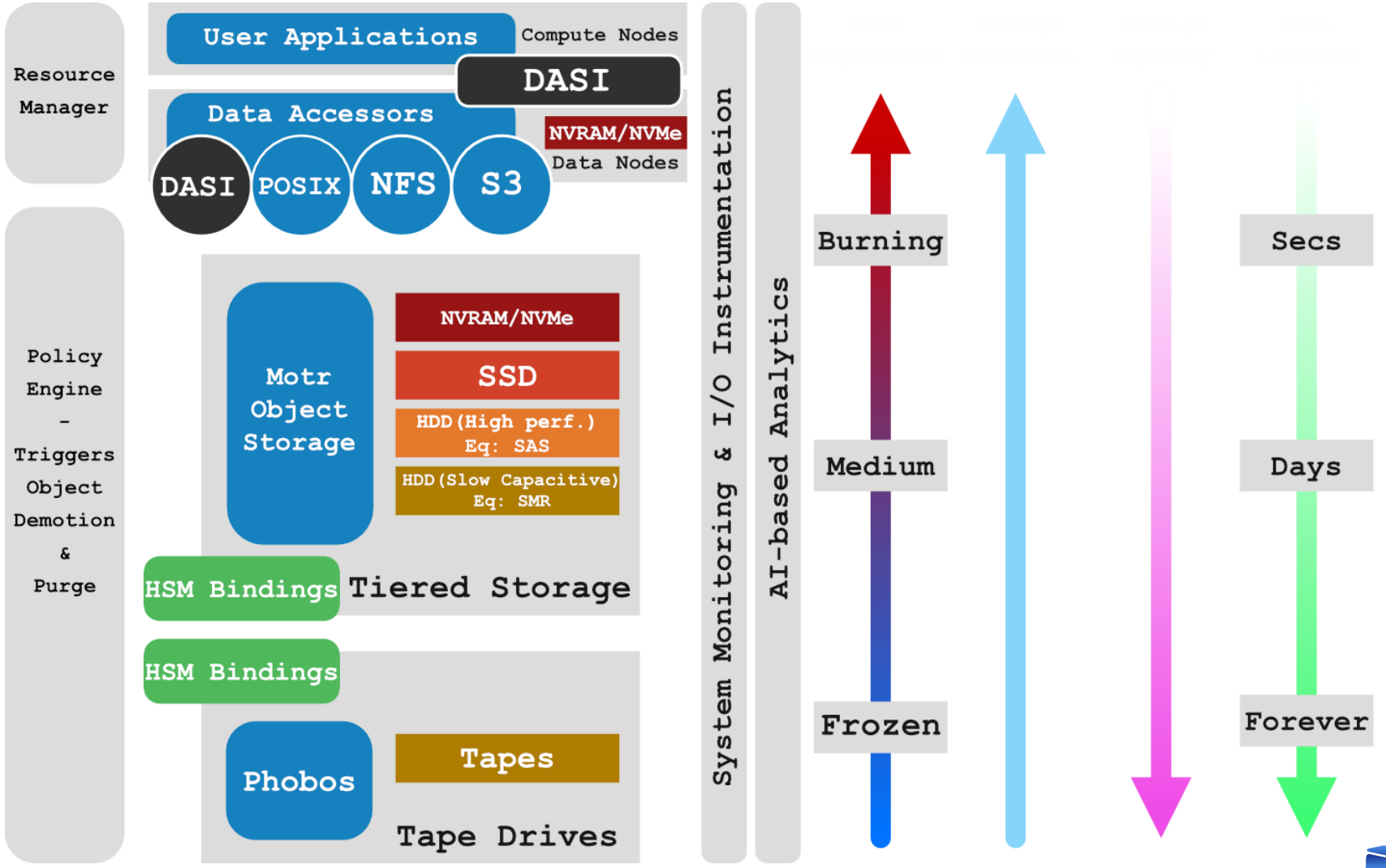


EuroHPC
Joint Undertaking

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955811. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, the Czech Republic, Germany, Ireland, Sweden, and the United Kingdom.

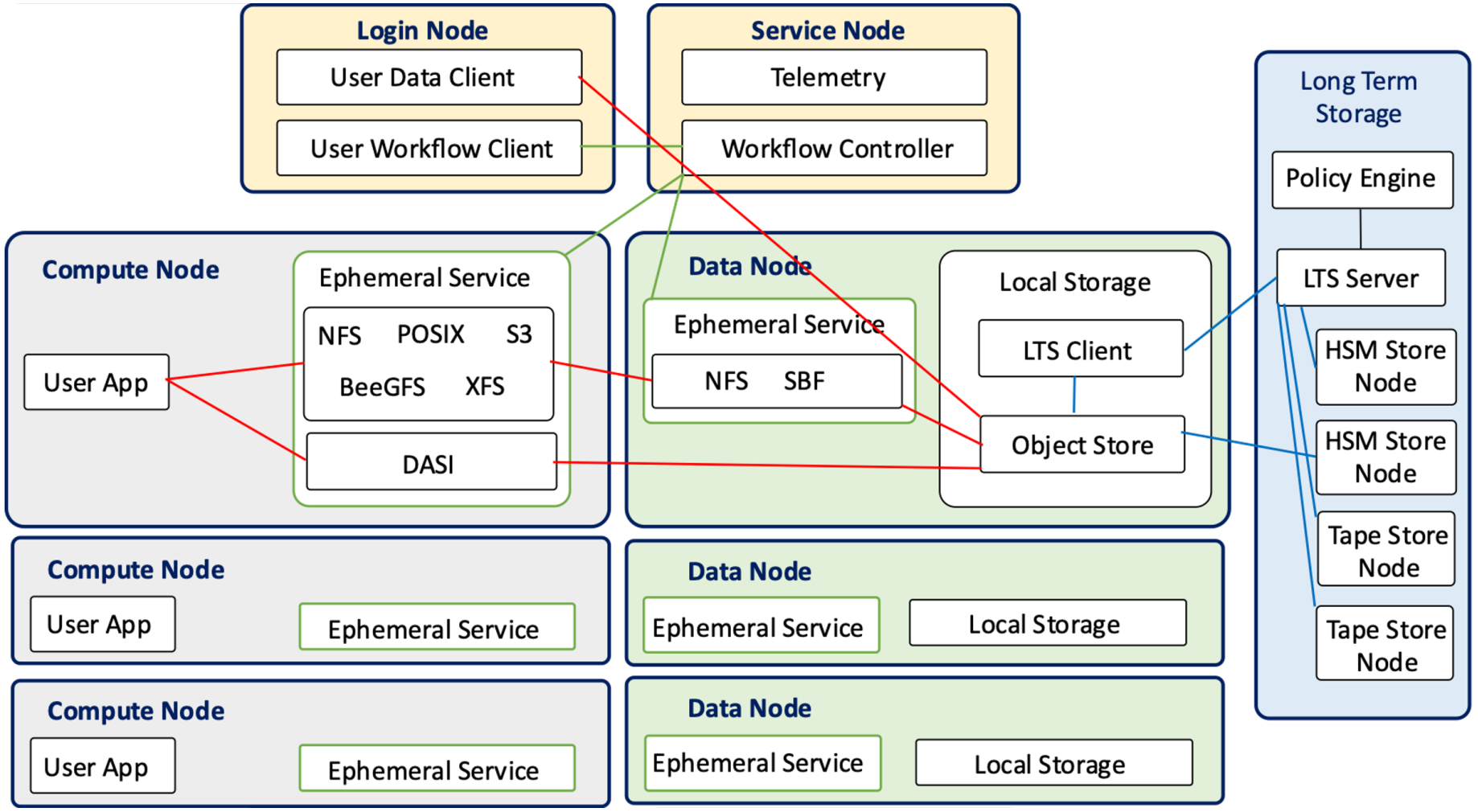
IO-SEA Project - Architecture

Storage I/O and Data Management for Exascale Architectures



IO-SEA Project - Architecture

Storage I/O and Data Management for Exascale Architectures

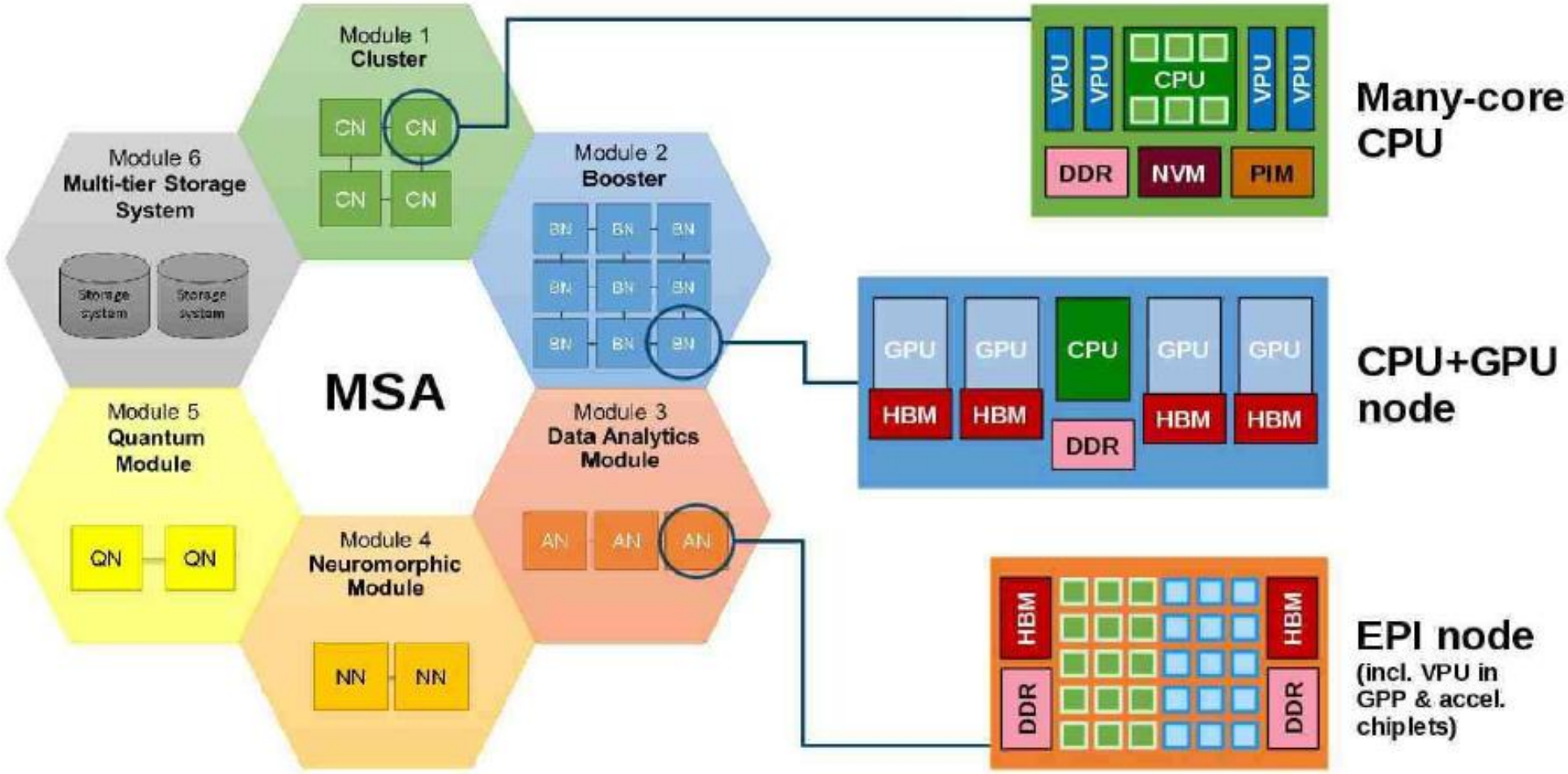


Concepts

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955811. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, the Czech Republic, Germany, Ireland, Sweden, and the United Kingdom.

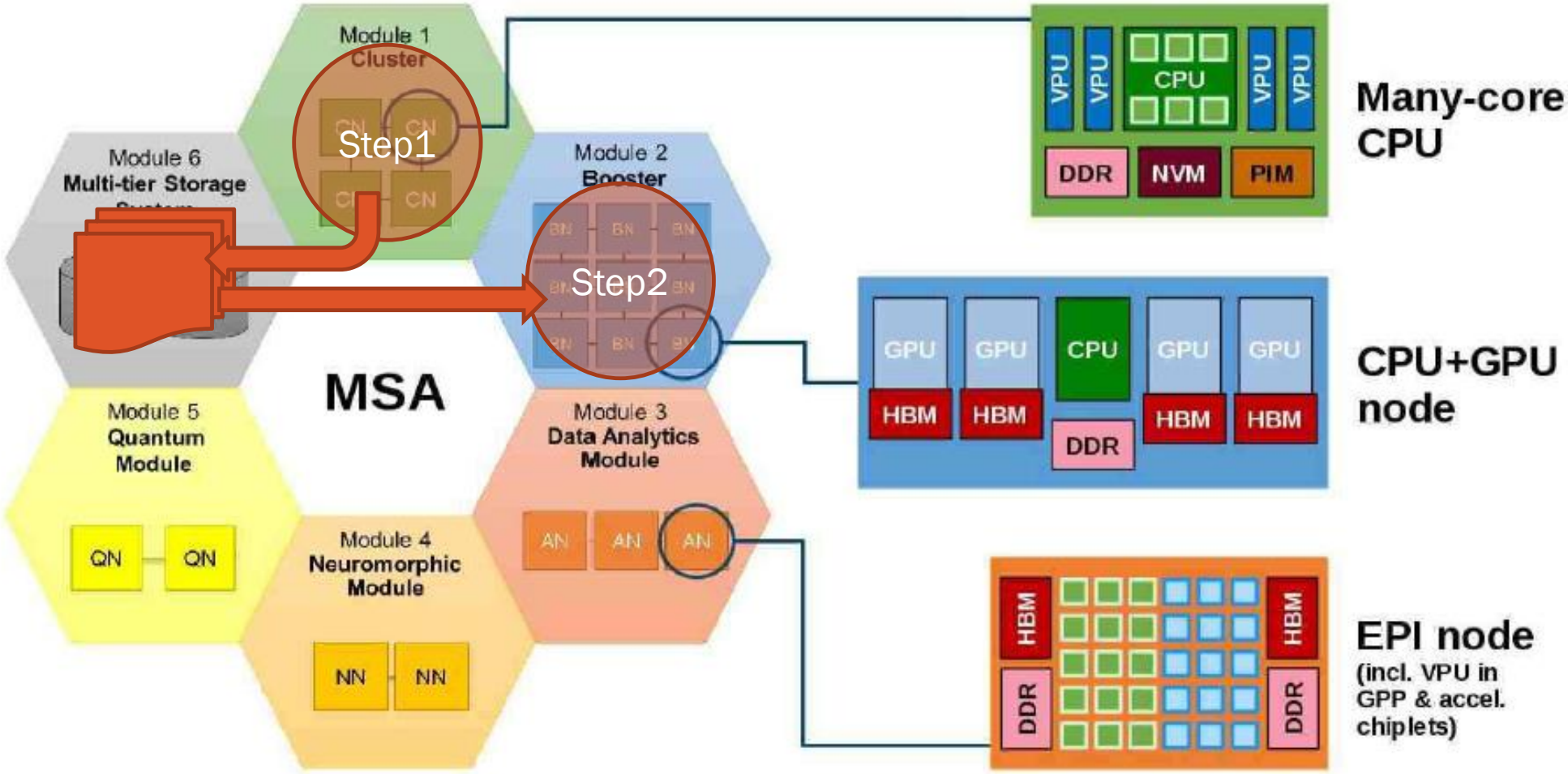
Modular SuperComputer Architecture

- Heterogeneous Compute modules
- « Long Term » Storage module with multiple technologies
- No cluster-wide interconnect
  - High speed Ethernet to connect modules



Modular SuperComputer Architecture

- Complex workflows running simultaneously and/or sequentially on multiple modules
  - Exchanging data through files

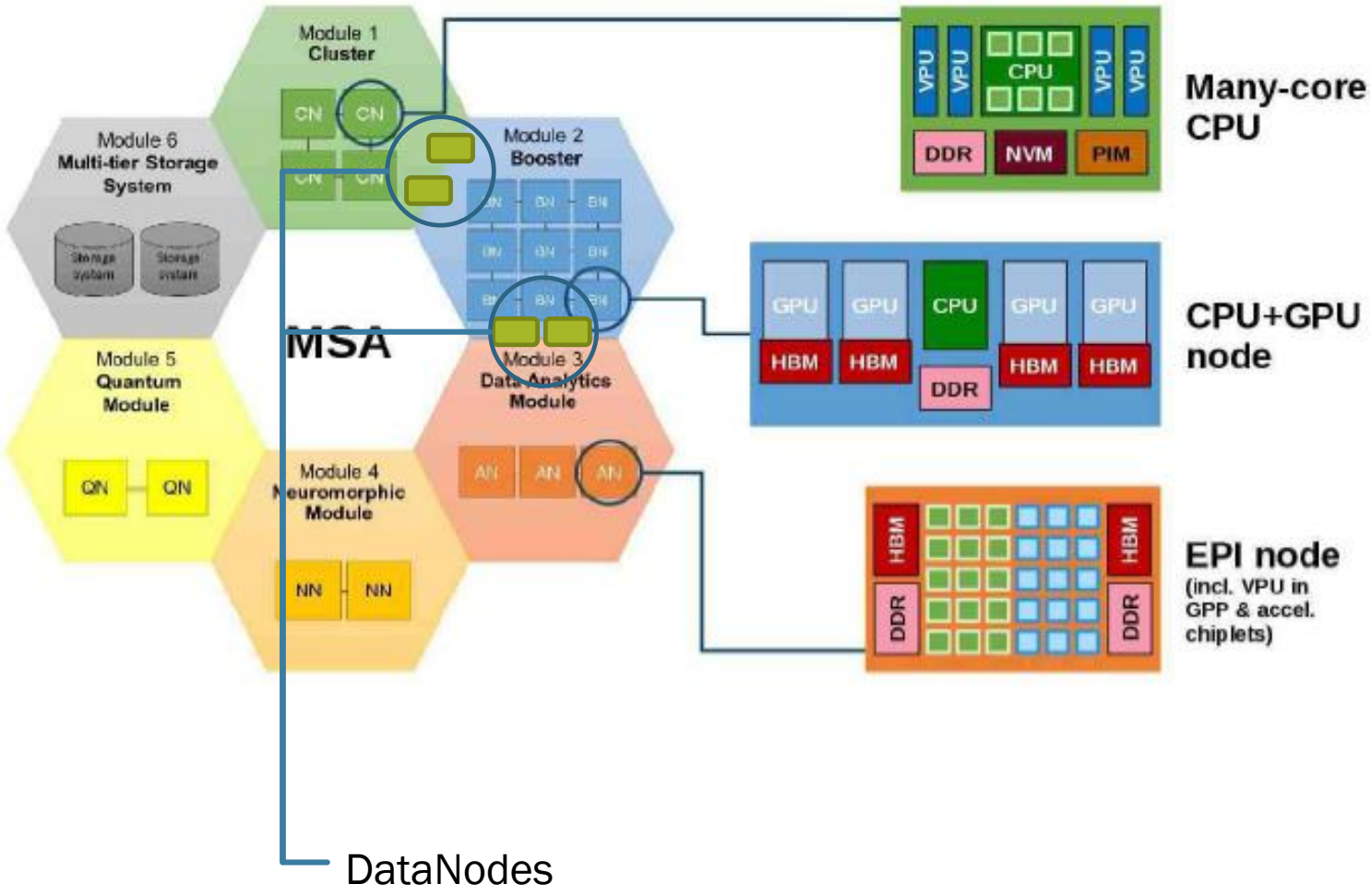


The IO-SEA Solution

IO-SEA Main Concepts 1: Data Nodes

Add **Data Nodes** between compute nodes and Long Term Storage Module

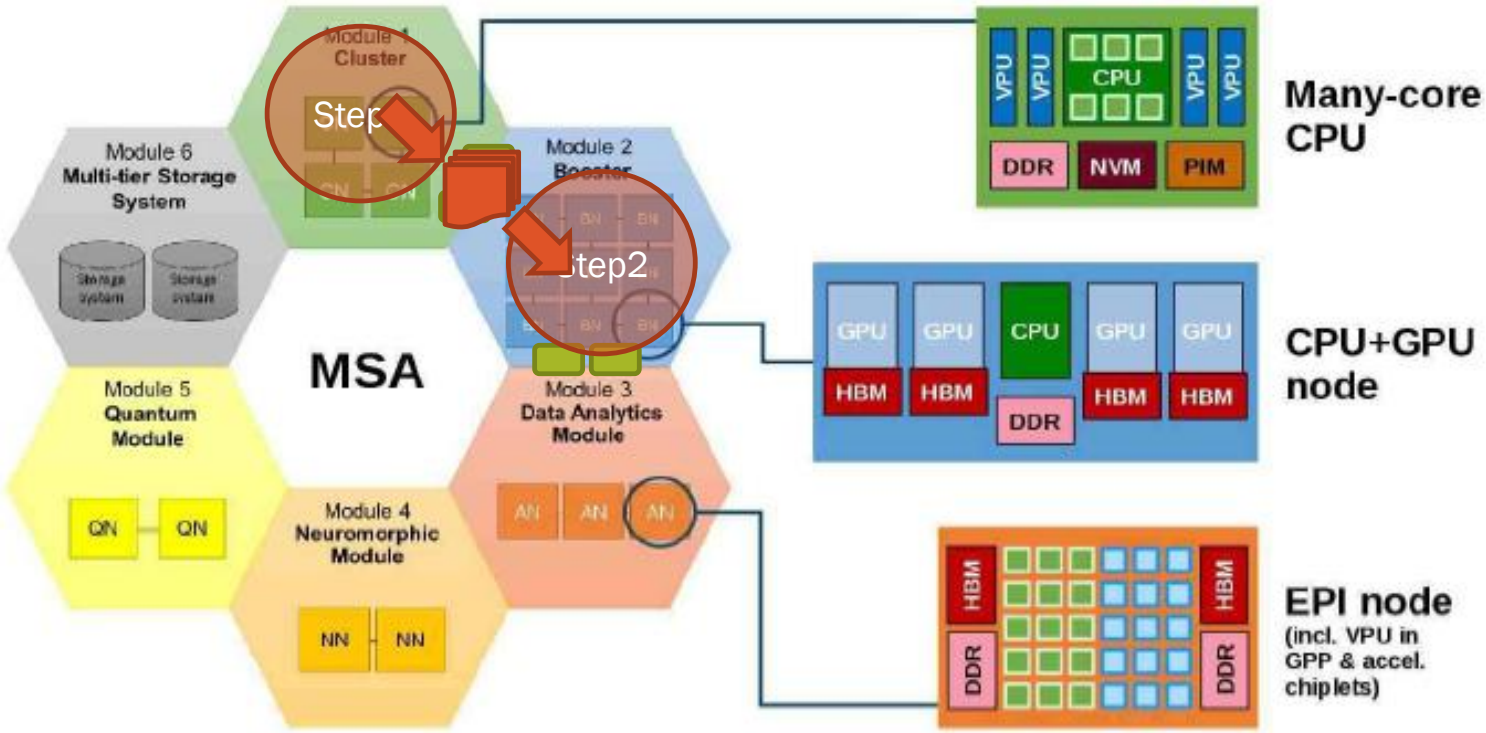
- connected to Compute Modules on their High Speed interconnects, and to the Long Term Storage Module
- Equipped with high speed storage devices
  - NVMe disks
  - NVRAM



IO-SEA Main Concepts 2: Workflows Scheduling

Consider **Workflows** rather than individual applications for scheduling

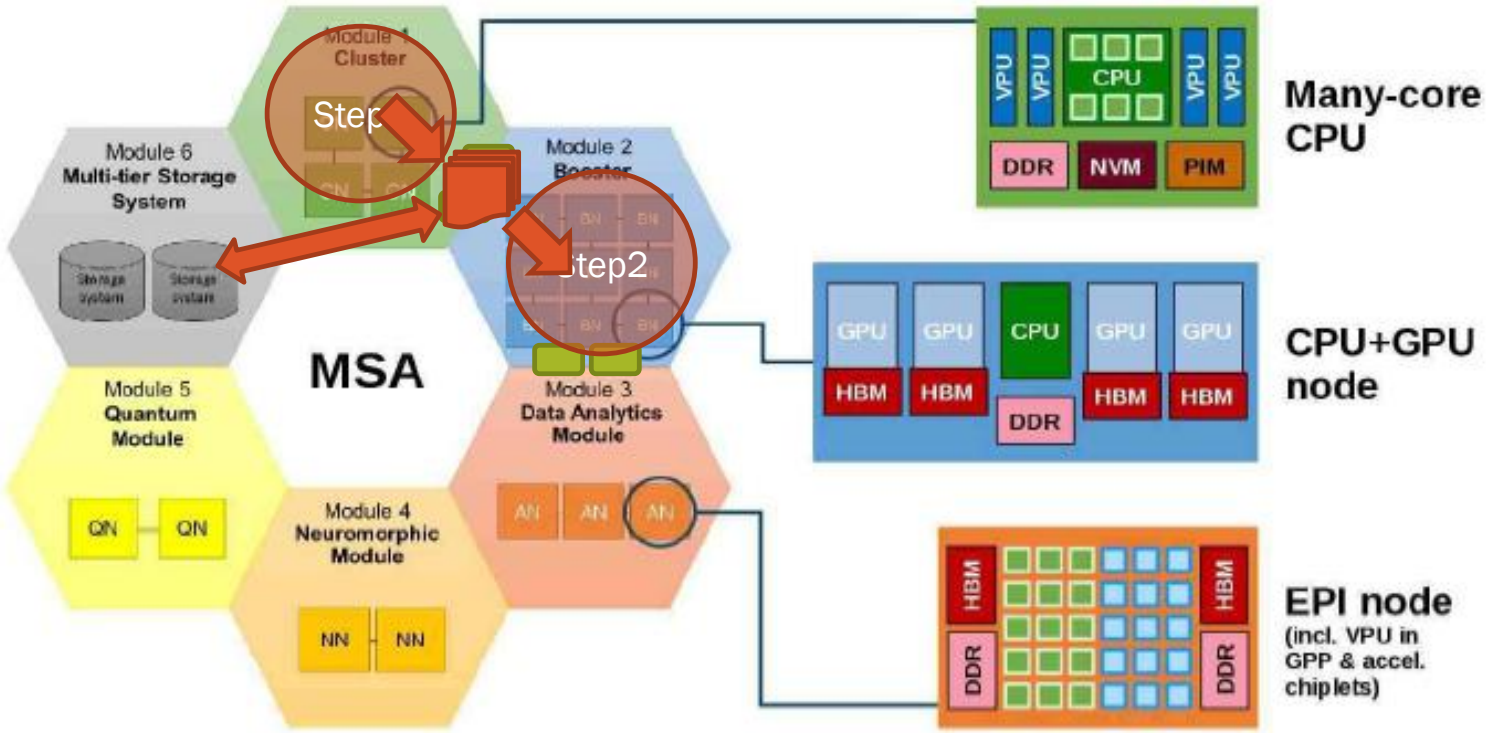
- Workflows composed of many applications (steps) running on different Modules
- Allocate data nodes resources in addition to Compute nodes



IO-SEA Main Concepts 3: Datasets & Ephemeral Services

Store Workflow data in **Datasets**

- Datasets are « data containers » stored in the **long term storage as « objects »,** exposed to compute nodes by **Ephemeral Services** running on Data Nodes



IO-SEA Stack

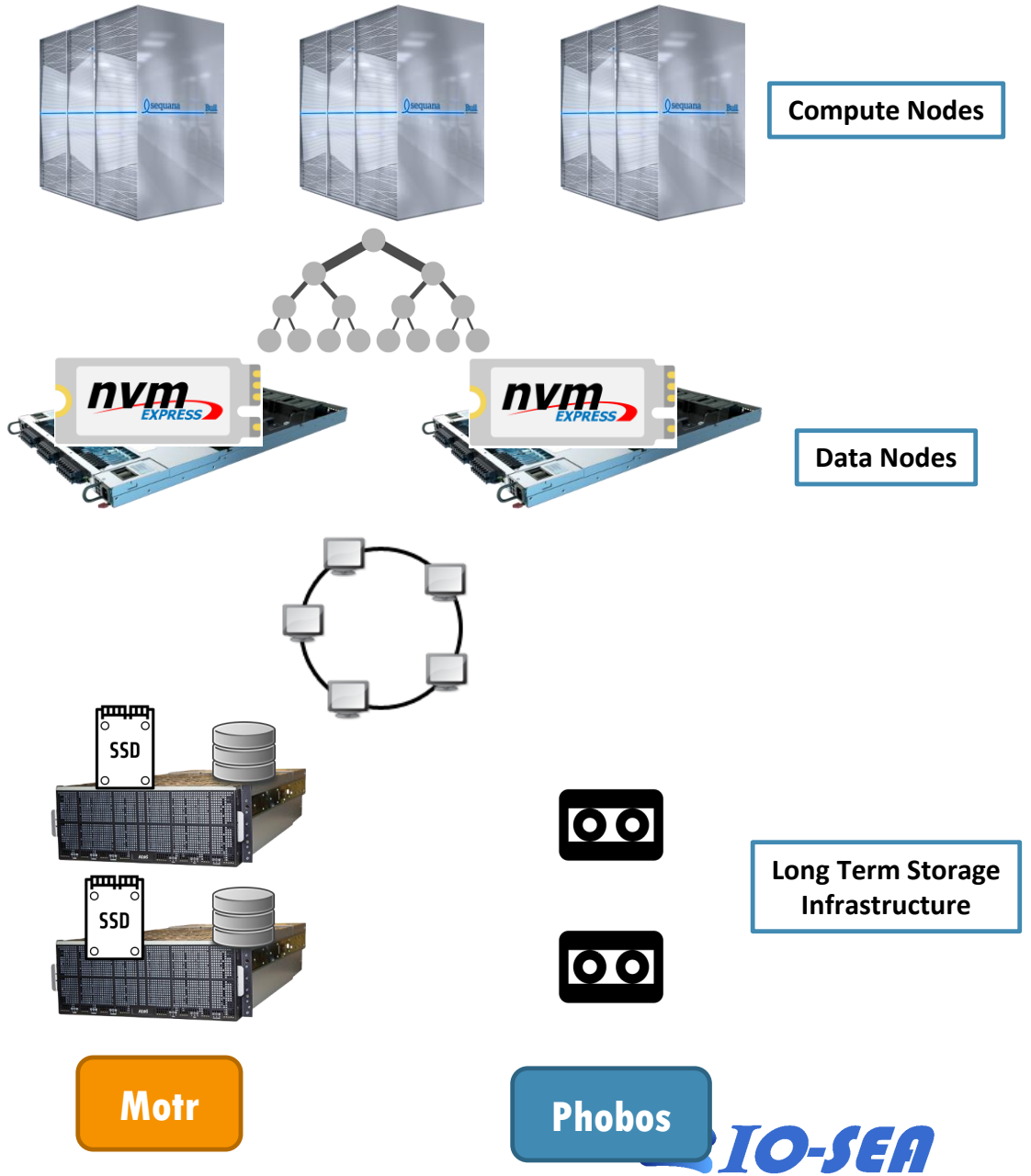
2 HW layers

- Data nodes equipped with high-speed storage devices (NVMe, NVRAM)

Ephemeral

- Long Term Storage composed of different tiers unified by an HSM software solution

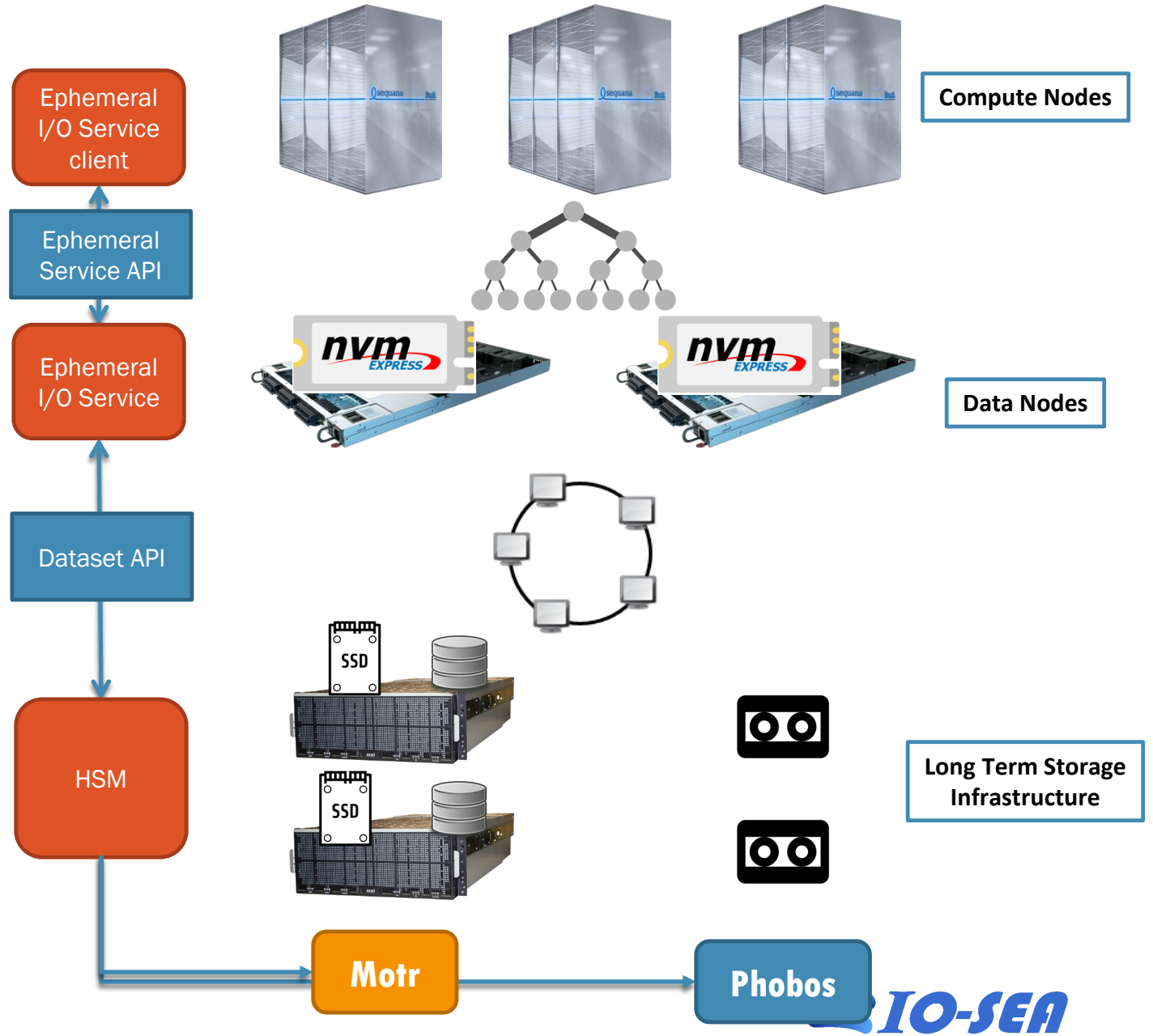
Long Term



IO-SEA Stack

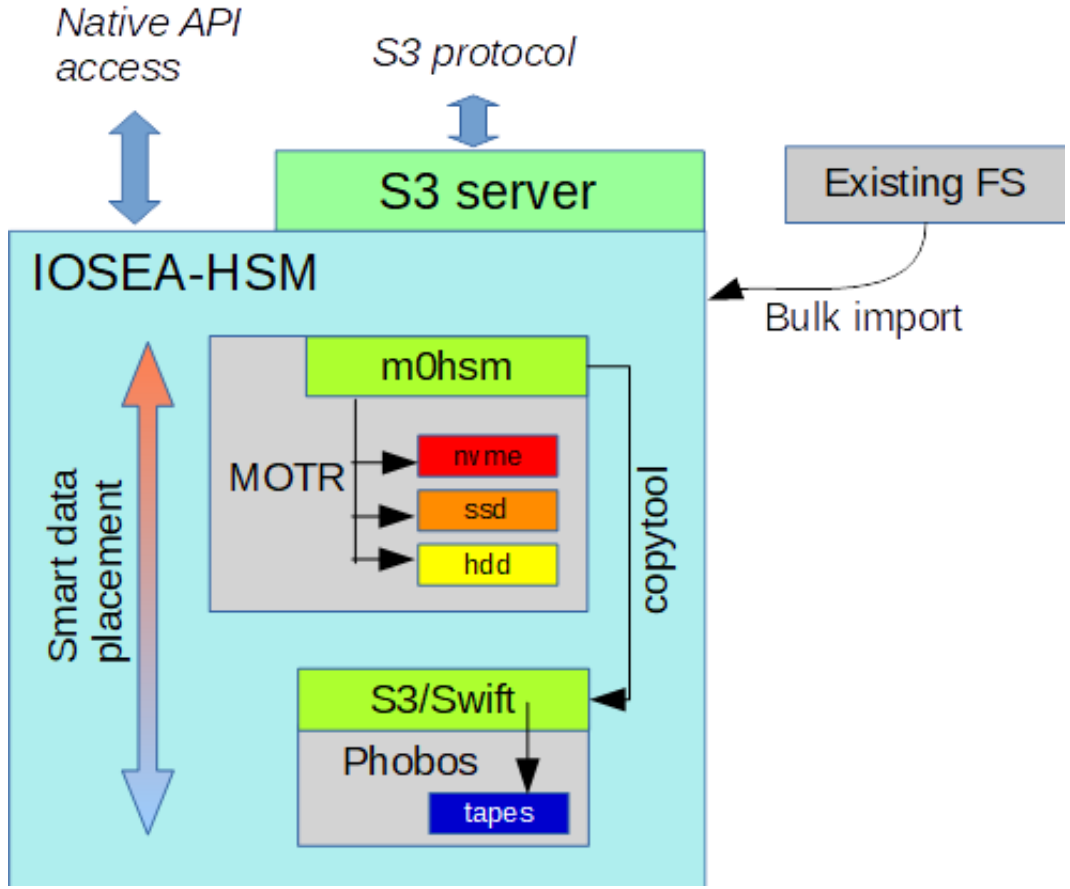
- Datanodes resources (cores, RAM, NVMe) are allocated for a **workflow** to run Ephemeral I/O services to give access to Datasets from compute nodes
  - POSIX, S3, DASI protocols

- Long Term Storage composed of different tiers unified by an HSM software solution
  - **Object Storage based**, no POSIX limitations



IO-SEA Long Term Storage : Hierarchical Storage

- Leverage a wide variety of **storage technologies** in a single system
  - from NVMe to tapes
- Implement **transparent** data migration between many storage tiers
- Smart **data placement** / **movements**:
  - ➢ Place data according to application needs
  - ➢ Gather needs from all running applications and develop a global data placement strategy
  - ➢ Prevent tiers from being fullfilled
  - ➢ Arbitrate conflicting requests / resource usage



DataSets & Namespaces

- Datasets are **data containers** hosting objects
  - Could be seen as private file systems or object stores
  - No data organization, just collections of objects…
- Objects in Datasets are organized with Namespaces
  - Different Namespaces can be created for each Dataset
  - Namespaces can expose the same dataset through different protocols
    - POSIX, S3

Datasets/Namespaces

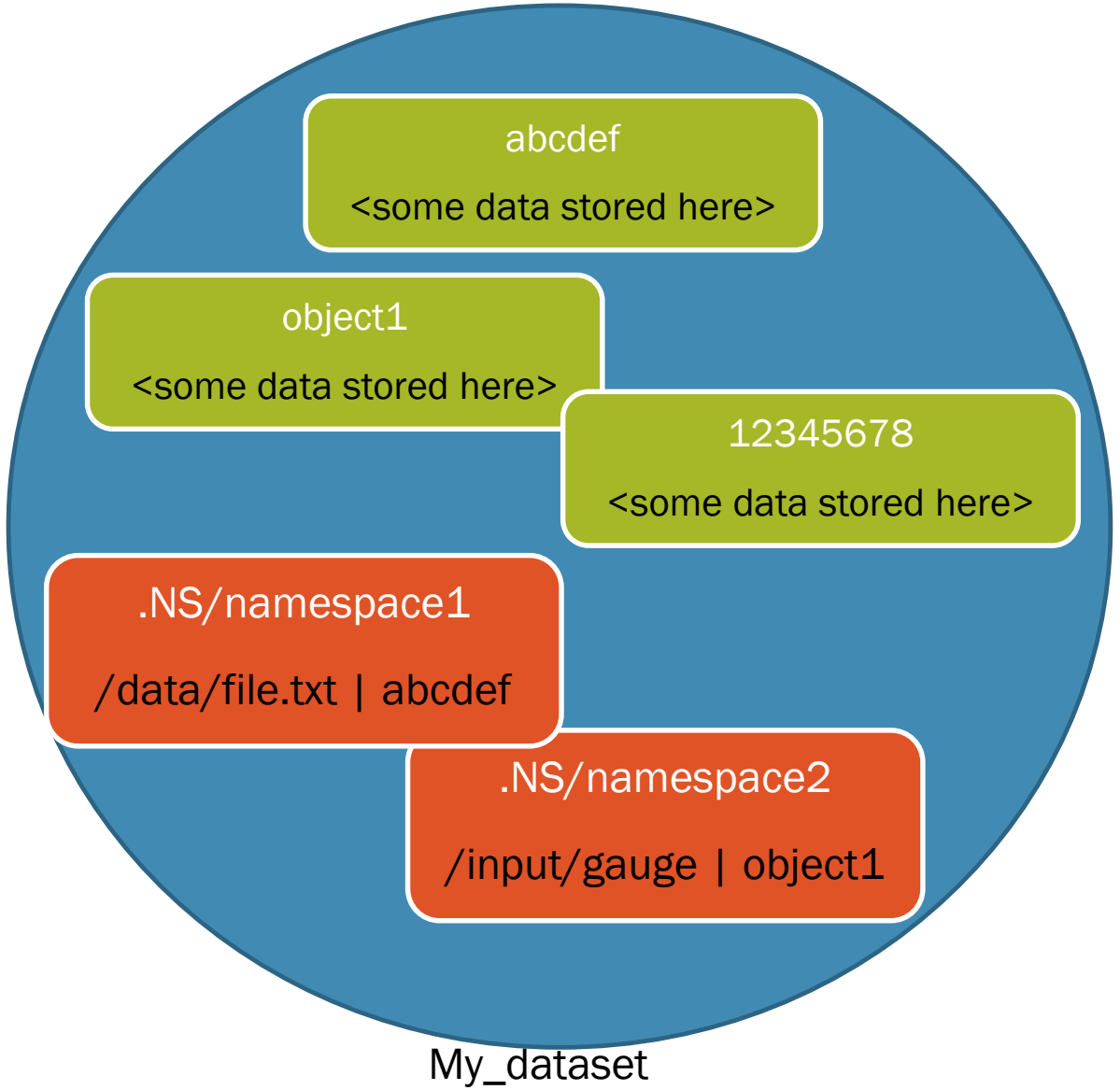
A Dataset is stored as 1 S3 Bucket

Files stored as S3 object

All namespaces stored in « metadata » objects

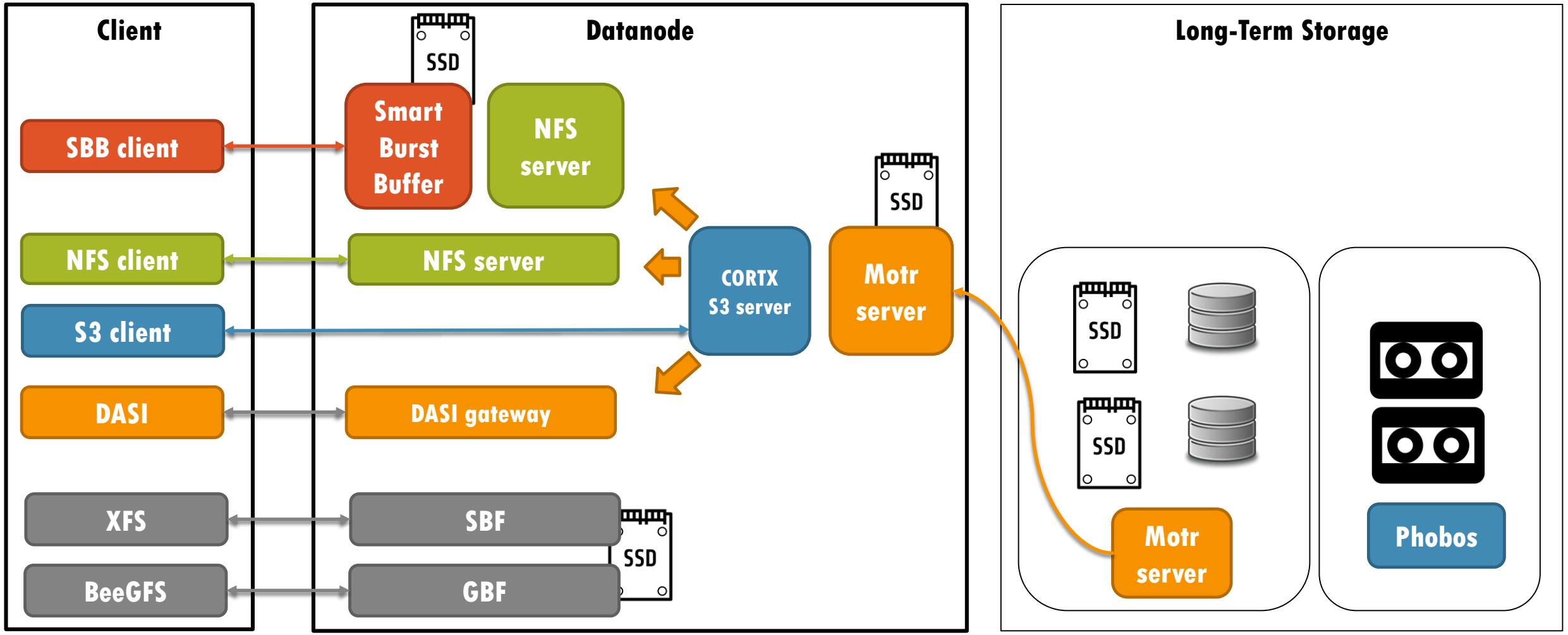
Think of it as a set of directories and symbolic links to objects in dataset

- .NS/namespace1
- .NS/namespace2



6 Ephemeral I/O Services

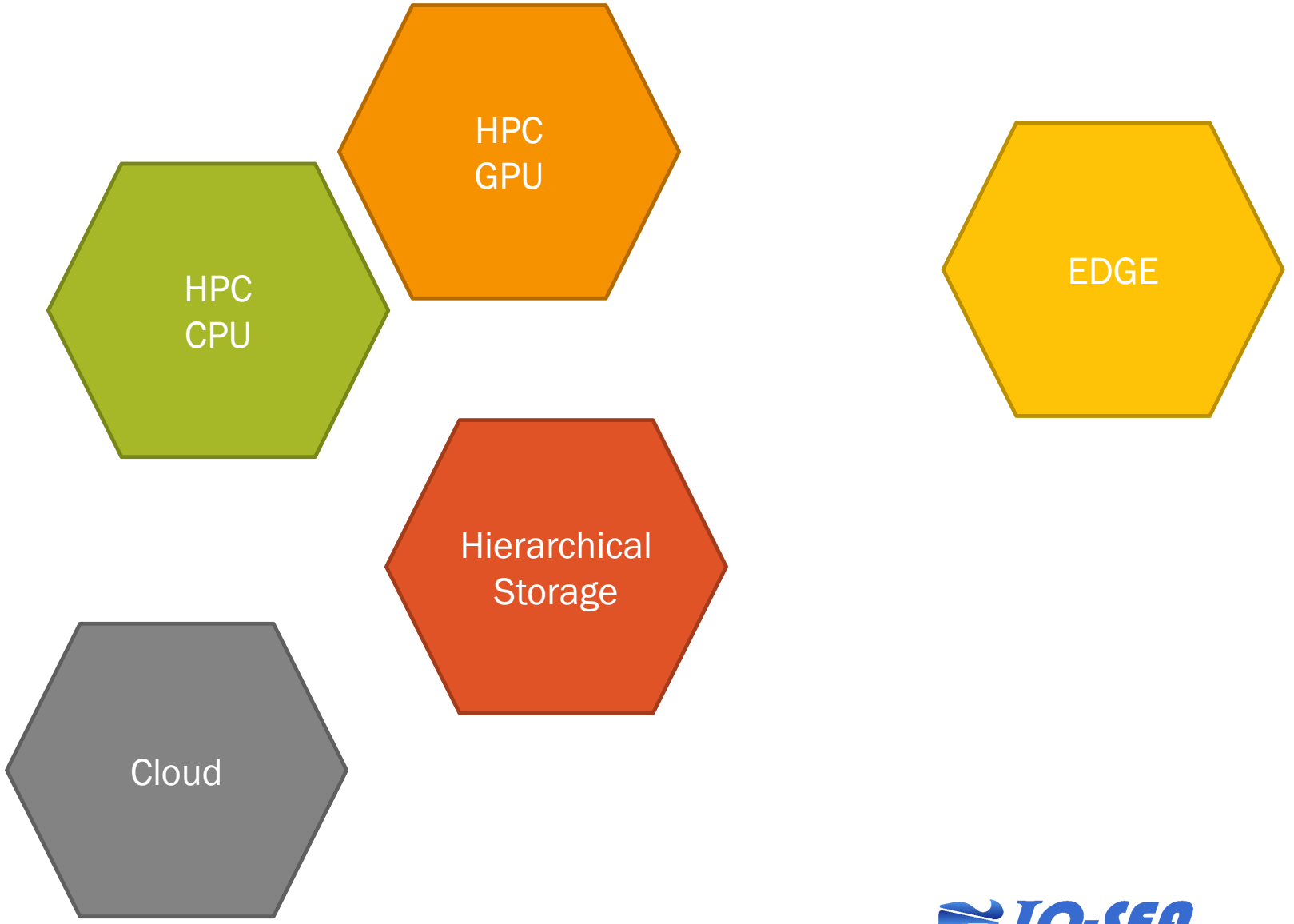
- NFS, BB-NFS, S3, DASI, SBF, GBF



WORKFLOWS

The Computing Continuum

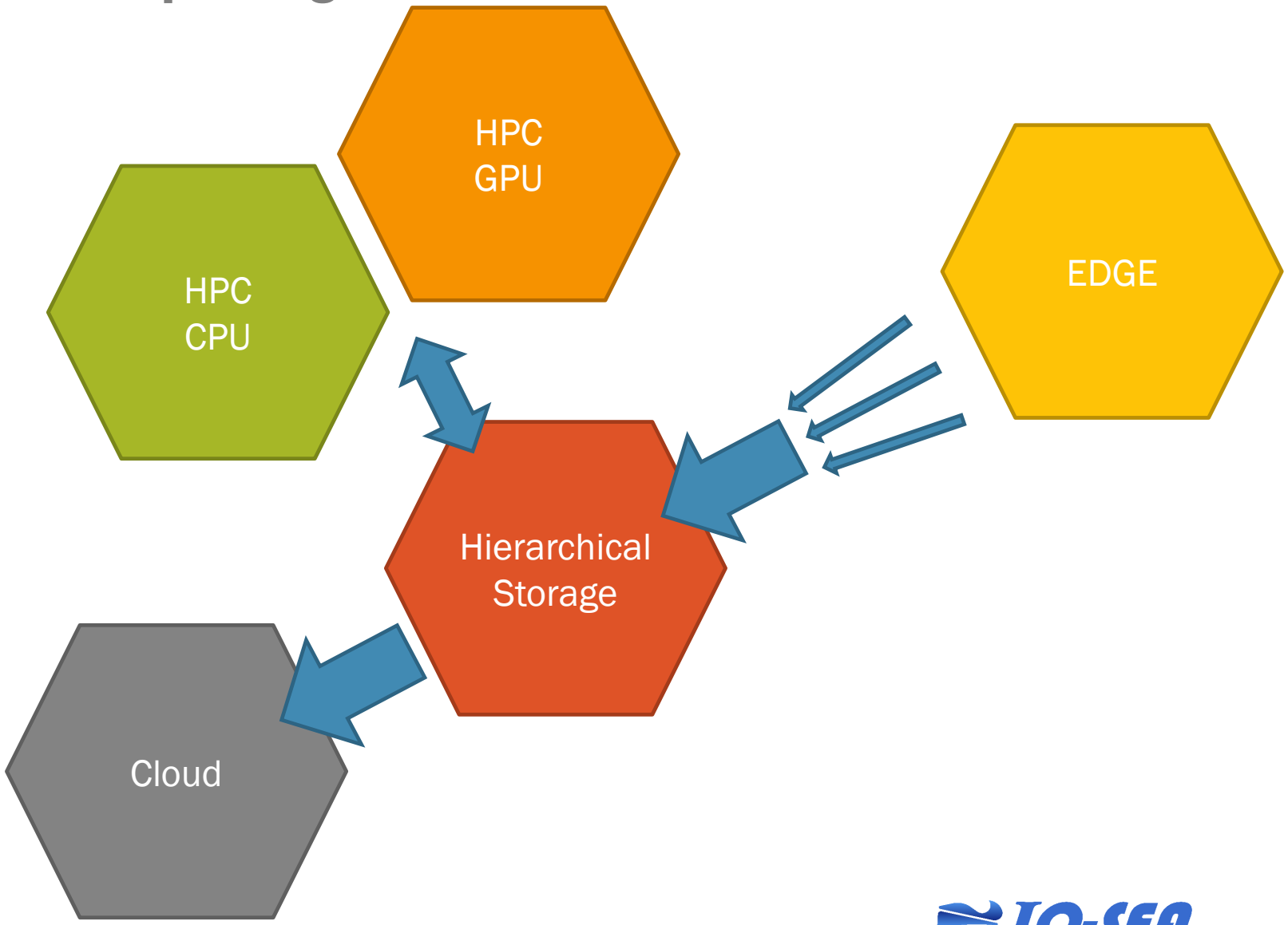
- Heterogeneous compute modules
  - Even on HPC
- « Long Term » Storage module with multiple technologies
  - Hierarchical storage
- Low/medium bandwidth, high latencies connections between modules



Data Movement in The Computing Continuum

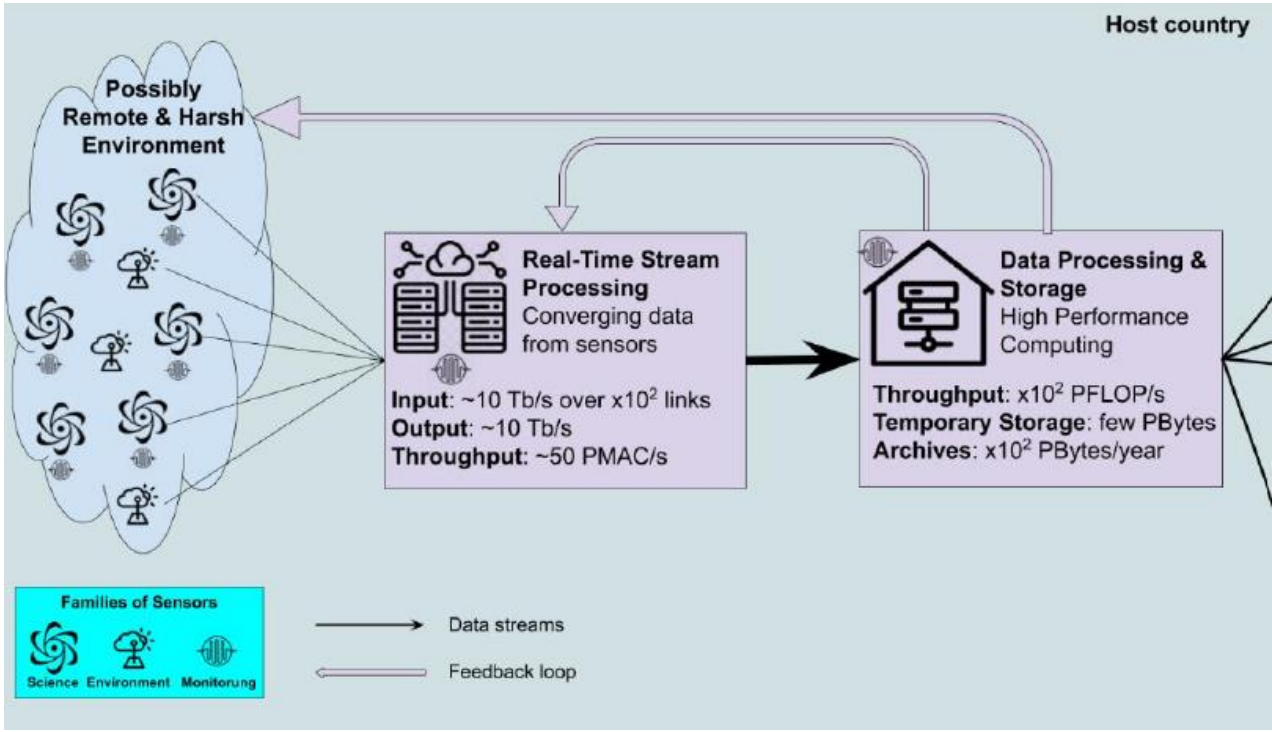
- Heterogeneous workflows
- Data produced/gathered outside of HPC datacenters
- Main processing steps on HPC implies data movement
- Post processing in the Cloud

- Challenges :
  - **Organize data**
  - **Limit data movements**
  - **Give control to users**

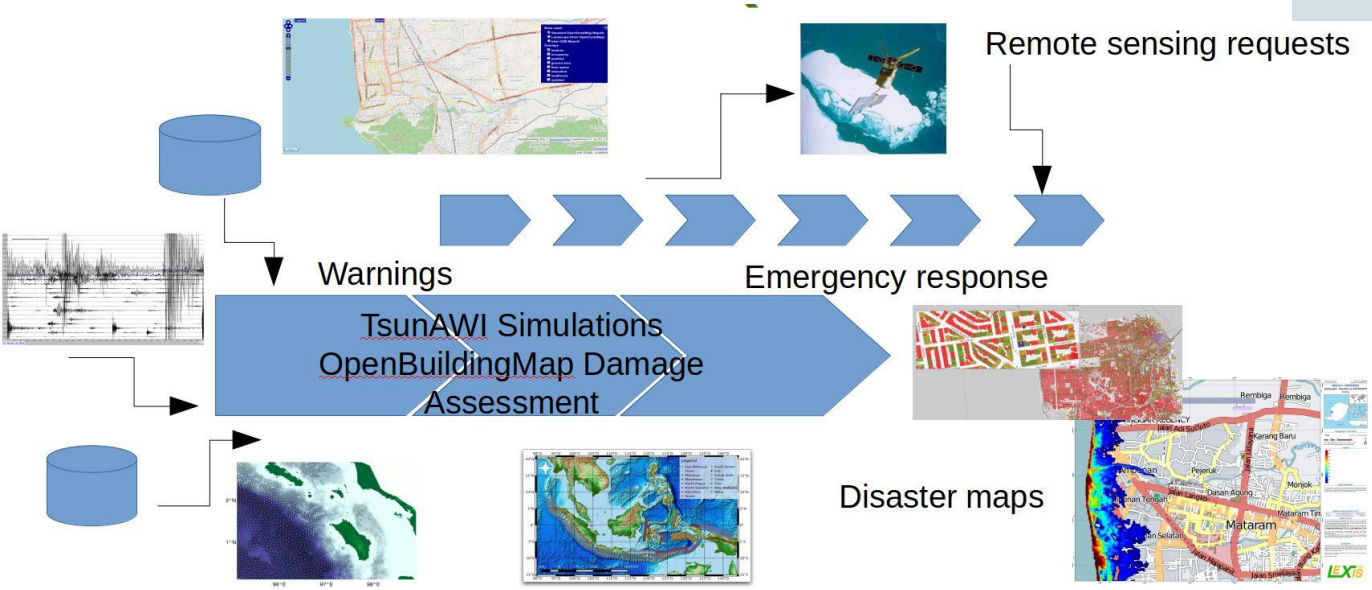


Example of such workflows

- Not IO-SEA workflows
- Tsunami prediction in LEXIS
- SKA in the « ECLAT » joint CNRS/INRIA/Atos laboratory



SKA Host Country infrastructure

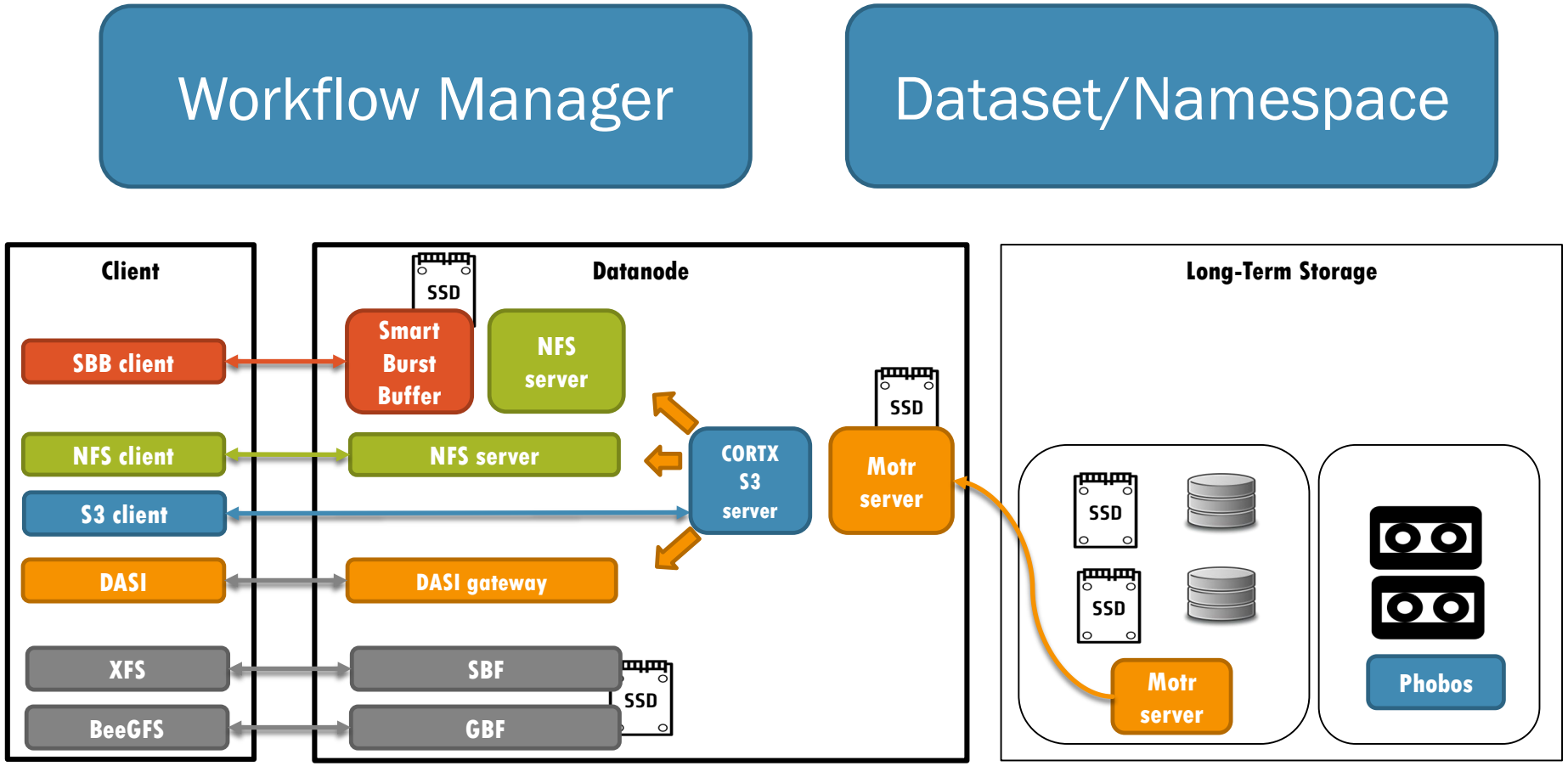


Tsunami prediction

Workflow: End User Interfaces & APIs

User Interfaces

- Workflows are described in a « Workflow Description File » . yaml, and managed through command line tools
- Datasets/Namespaces are created with command line tools



Workflow Sessions

- The steps processing the same data are run within a « **session** »
  - **Access tokens** protect against dataset access by multiple sessions in parallel
- Sessions have a user provided name
  - (UID, Session_Name) as identifier

```
# start a session for my workflow described in the WDF.yaml file
iosea-wf start WORKFLOW=WDF.yaml SESSION=My_Session

#run the workflow steps
iosea-wf run SESSION=My_Session STEP=step1
iosea-wf run SESSION=My_Session STEP=step2
iosea-wf run SESSION=My_Session STEP=step3

#stop the session and release the datanode
iosea-wf stop SESSION=My_Session
```

IO-SEA

Workflow Session Management

- **status** & **list** commands to report information
- **access** command to launch an interactive access environment
  - Slurm salloc to launch a shell in which Ephemeral Services clients will be configured
  - Will be « shareable » with team members

```
# display info about jobs & ephemeral services
iosea-wf status SESSION=My_Session

# list all active sessions of the user (report the session-names)
iosea-wf list

# Start an interactive access environment for all or limited to [<service>]
iosea-wf access SESSION=My_Session [NS=service]
```

Workflow Description File (WDF)

- **services** describe the ephemeral services needed to run the workflow

- **steps** describe how to configure the run time environment to run the steps

```
workflow:
  name:      My_Workflow

services:
  - name: ephemeral_service_1
    type: NFS
    attributes:
      namespace: dataset.My_namespace
      mountpoint: /mnt/USER/My_Workflow
      flavor: medium

steps:
  - name: step_A
    location:
      - gpu_module
    command: "srun My_Step_A"
    services:
      - name: ephemeral_service_1
```

Parametric Workflow Description File (WDF)

- To run multiple sessions in parallel, or to make the WDF more generic, variables can be used for most fields
- Variables must be defined « before use »

```
services:
  - name: ephemeral_service_1
    type: NFS
    attributes:
      namespace: {{ NS1 }}
      mountpoint: /mnt/USER/{{ SESSION }} My_Workflow
      flavor: medium
```

```
# start a session for my workflow
iosea-wf start WORKFLOW=WDF.yaml SESSION=My_Session NS1=My_namespace

#run the workflow steps
iosea-wf run SESSION=My_Session STEP=step1 NS1=My_other_namespace
```

Status command

- to monitor steps progress

```
> iosea-wf status SESSION=My_session
Workflow Name       : My_Workflow
Workflow SessionID : My_session_<timestamp>

Pending Steps
Step Name    Slurm ID     Tag      Command
step_B       2767         last     srun My_Step_B
SaveResults ----                   srun My_Copy_Script

Active Steps
Step Name    Slurm ID     Tag      Command
step_B       2763                  srun My_Step_B

Terminated Steps
Step Name    Slurm ID     Tag      Command
step_A       2760         init     srun My_Step_A

Ephemeral Services....
```

Location in the WDF

```
steps:
  - name: step_A
    location:
      - gpu_module
    command: "sbatch My_Step_A"
    services:
      - name: ephemeral_service_1
 - name: step_B
    location:
      - gpu_module
      - cpu_module
    command: "sbatch My_Step_B"
    services:
      - name: ephemeral_service_2
 - name: step_C
    location:
      - datanodes
    command: "sbatch My_Step_C"
    services:
      - name: ephemeral_service_2
```

- Steps indicate the « location » of their compute nodes
- Multiple locations are possible for complex jobs (MSA architecture)
- Location can be also « datanodes » for « on the fly » processing

IO-SEA

Data Movers in the WDF

```
services:
  - name: ephemeral_service_1
    type: NFS
    attributes:
      namespace: {{ NS1 }}
      mountpoint: /mnt/USER/{{ SESSION }}
      flavor: medium
    datamovers:
      - name: datamover1
        trigger: step_start
        target: flash
        operation: copy
        elements:
          - "gauges/*.hdf5"
          - "input/*"
```

- To ensure the elements of a namespace are located in the proper storage tier, a data_mover can be activated before (step_start) and after (step_stop) a step
- Operations are either move or copy
- Data Movers are defined at the service level, but activated per step

```
steps:
  - name: step_A
    location:
      - gpu_module
    command: "sbatch My_Step_A"
    services:
      - name: ephemeral_service_1
        datamovers:
          - datamover1
```

IO-SEA

Hints

"Hints" are optional information given by users about their future use of data, when the workflow is terminated

- **intended_access**: Intended access in the short term (will access, won't change…)
  e.g. intended_access="wont_change"

- **estimated_lifetime**: Estimated time until the object will be deleted (in seconds)
  e.g. estimated_lifetime=2592000 (1 month)

- **estimated_atime**: Estimated time the object will be used (in seconds)

- **access_period**: How often the object will be accessed (in seconds)
  e.g. access_period=60 (every minute)

- **predefined_policy**: Name of a pre-configured policy
  e.g. predefined_policy="temporary_data"

Setting hints

- User Control: CLI
  - `iosea-ns {locate|move|copy|release}`
  - `iosea-ns hints DATASET=My_dataset hint1=value1 hint2=value2 ...`
- User Control: through POSIX Ephemeral Services
  - `Set_attr on files and directories`

Datasets/Namespaces : Command line tools

```
# create namespaces
iosea-ns create my_dataset.my_namespace
iosea-ns create DATASET=my_dataset my_2nd_namespace
# fill my_namespace with a file
io-sea-ns put DATASET=my_dataset NAMESPACE=my_namespace ./gauge.cfg
```

- Enables leveraging standard POSIX tools and scripting

ls, find, …

PHOBOS

Context

Next scale of mass storage

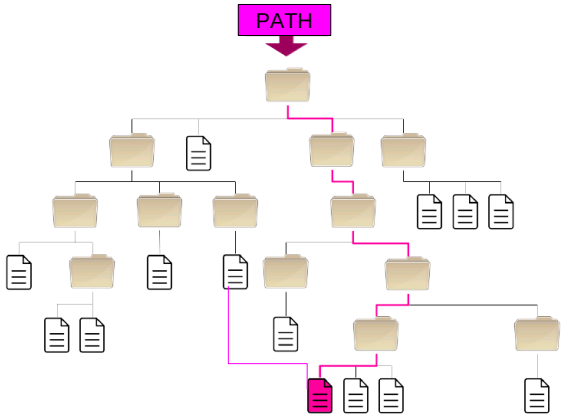
- Exaflopic supercomputers in the 2020's
- Huge amounts of data to ingest: petabytes per day
- Huge amounts of data to store: exabytes

	Today	Tomorrow
Daily production	Hundreds of TB	Petabytes
Storage system capacity	Hundreds of PB	Exabytes

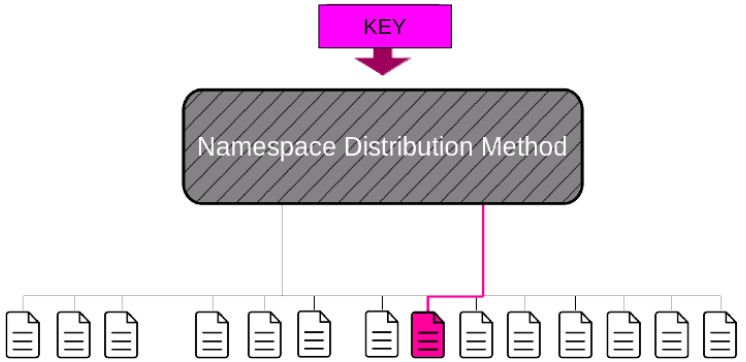
IO-SEA

Benefits of object-based storage

- Suppressing POSIX filesystem's bottlenecks



Addressing entries in a POSIX file system

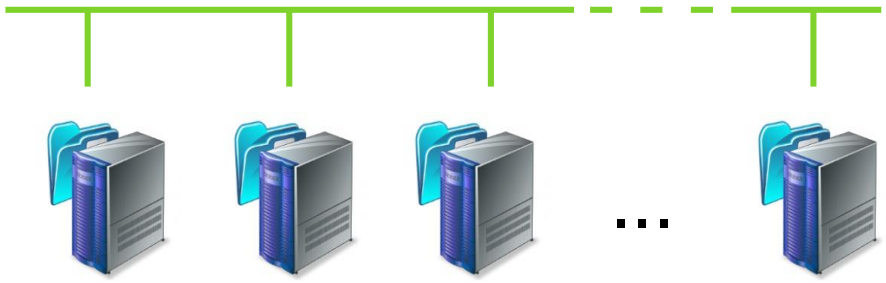


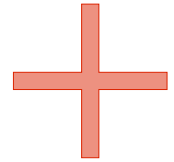
Addressing entries in an object-store

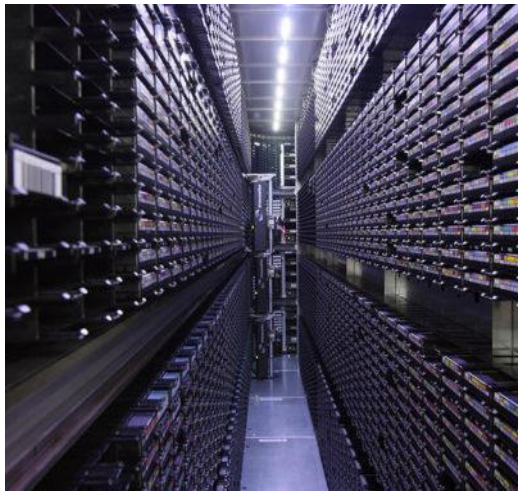
- Object stores have proved their scalability
- Widely adopted for Internet services, Cloud computing, social networks...

Challenge

- Needs for extremely scalable storage systems
- at a reasonable price
  - Object store: horizontal scalability
  - Tape library: safe long-term storage at low cost







Existing solutions for scalable tape storage

Drawbacks of existing solutions:

- Vendor lock-in
  - Proprietary code, formats and protocols
  - Lacking integration to standards
- Expensive
  - Licenses
  - Complex (need local expertise)

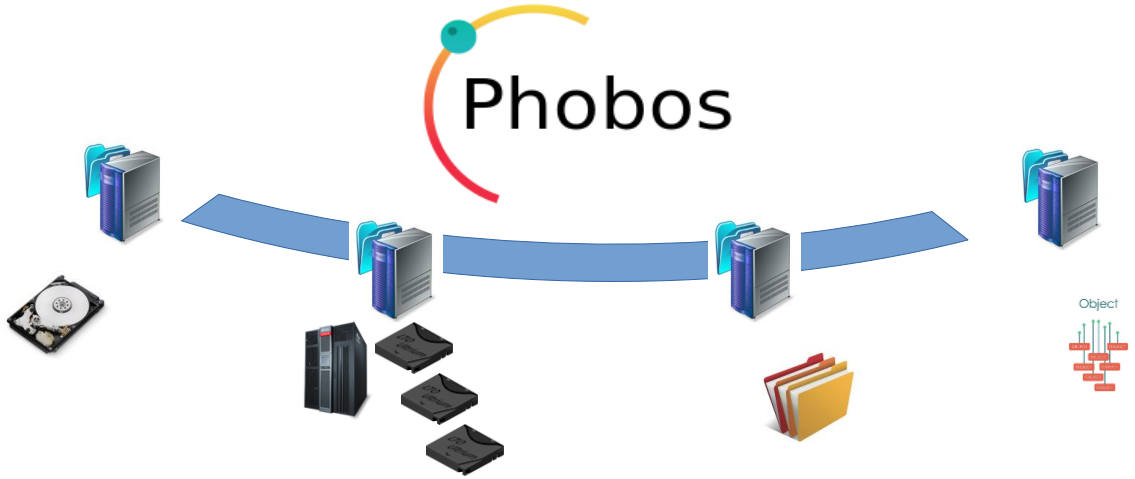
- Provide much more feature than needed (=> complex and expensive)
- Heavy installation and maintenance operations

IO-SEA

Phobos

Phobos: Parallel Heterogeneous Object Store

- Goals :
  - Manage a distributed set of storage resources on various storage technologies (HDD, **tapes**, object stores...)
  - Implement the best I/O optimizations for each technology without compromise
    - E.g. for tapes: minimize mounts and data sync

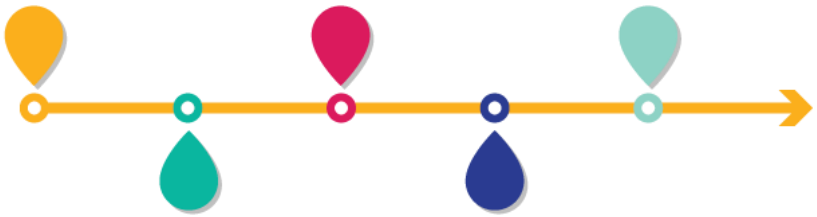


Guidelines

Design guidelines

- Scalability and fault-tolerance
- Based on open formats, open protocols, interoperable
  - E.g. LTFS as tape filesystem (ISO/IEC 20919:2016)
- Simple and common interfaces (CRUD API, REST)
- Simple administration (intuitive, admin-friendly CLI)
- Light, easy to deploy, easy to maintain
  - As of today: 48k lines of C and Python

History of the Project



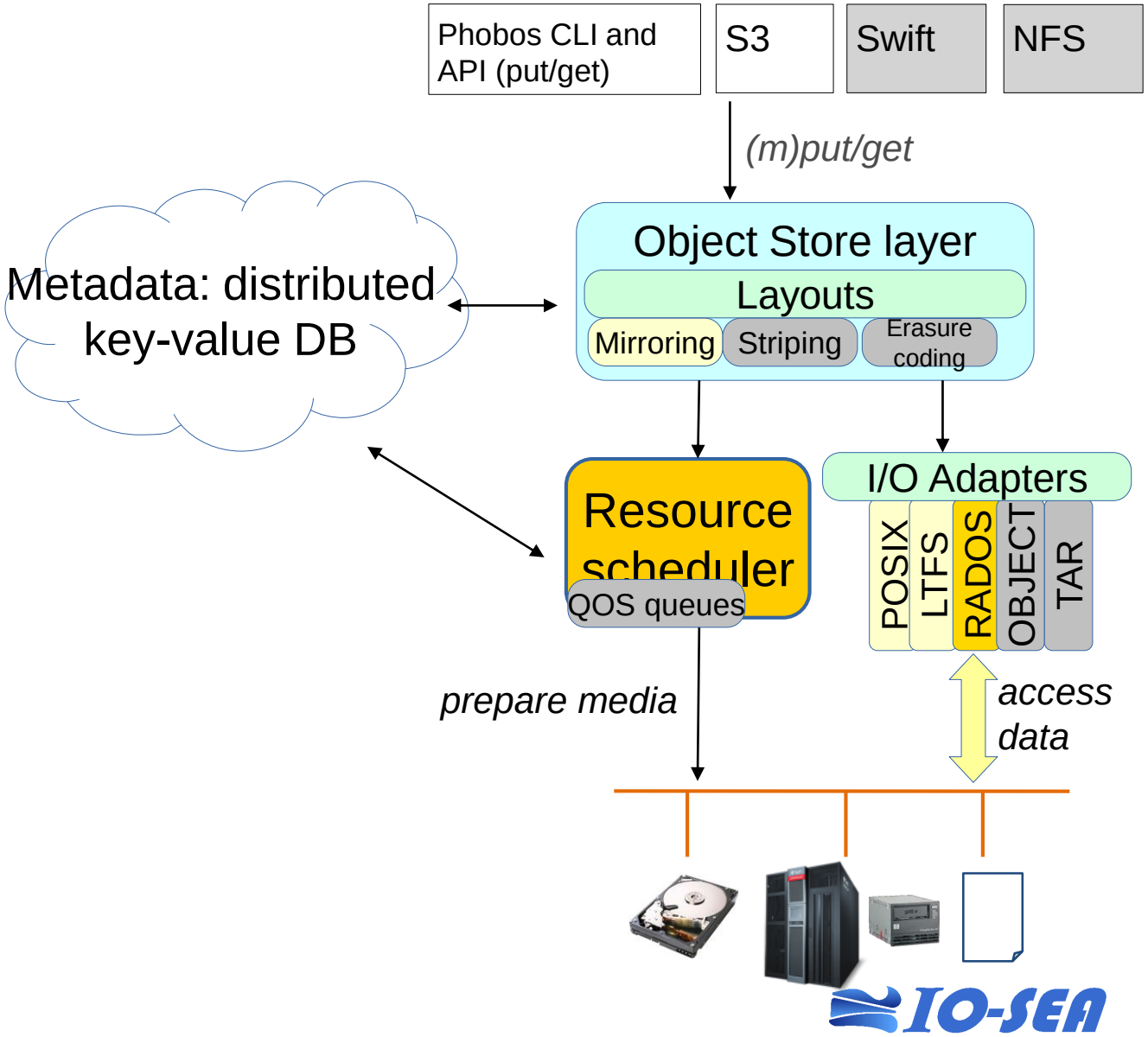
- 2013: first ideas
- 2014-2015: development of the initial version
  Scope:
  - Storage on tape using LTFS, or in a filesystem
  - SCSI-controlled tape library and LTO drives
  - Single server
- 2016: **Phobos in production**
  - Multi-Petabyte storage of genomics data
  - IBM TS3500 library, LTO5/6 drives
- 2019: Phobos made **open-source** (LGPL v2.1), available on github
- 2020-2022: Towards Phobos 2.0 → Parallelizing Phobos
- September 2022: First **parallel** version of Phobos **in production** as

FRANCE GÉNOMIQUE

IO-SEA

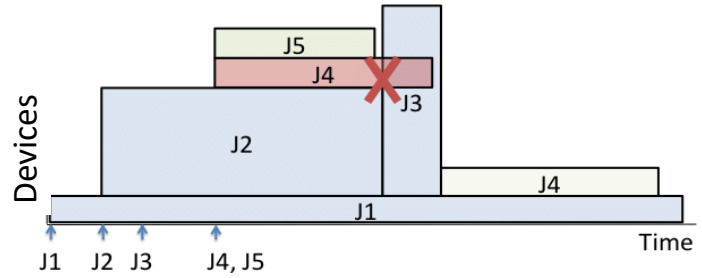
Phobos components overview

- **Front-ends**: CLI, API, S3, more to come
- **Key-value metadata schema**:
  - DB schema is NoSQL-ready
  - Currently uses PostgreSQL: can be parallelized thanks to sharding features
  - Backup copy of metadata stored with objects (recovery, tape import)
- **Layout plugins**: performance and fault-tolerance
- **Resource scheduling**: optimizes stream to tape drives, minimizes tapes mounts
- **IO adapters**: support of multiple storage backends (Posix, LTFS, RADOS)



Roadmap

- Current development: optimized I/O scheduling for tapes
  - Short term focus on grouping I/Os on tapes
  - Still much to do (local IO scheduling, global IO strategy, organization of device utilization over time...)
- 2H 2022: media life cycle (policy-based repacks...)
- 1H 2023: internal data migration (policy-based)
- Other planned enhancements:
  - Disaster recovery
  - Media import
  - New layouts (e.g. erasure coding)
  - New front-ends / new backends



In production use-case

FRANCE GÉNOMIQUE

- Multi-petabyte genomics datasets
- In production since 2016

DNA sequencers

Phobos
- IBM TS3500 tape library (SCSI)
- LTO6 and LTO8 drives

HPC data clusters

TGCC
Très Grand Centre de calcul du CEA

IO-SEA

Community

- Collaboration with DDN and ICHEC:
  - Implementing a S3 server for Phobos: Deimos
  - Contribution to Phobos: “alias” feature

- Collaboration with Atos, ECMWF, ICHEC, Seagate, Univ. of Mainz
  - In the framework of the EuroHPC project “IO-SEA”
  - Building a storage software stack for Exascale systems
  - Phobos used as the long-term storage component
  - New developments: scalability enhancements, erasure coding, media lifecycle management, administrative interface, LTFS tape import, smart tape request reordering, front-ends (Swift, POSIX)...

- Many contacts with interested HPC sites, companies...

Phobos configuration file

Config

- One configuration file:

```
/etc/phobos.conf
```

- Simple key-value format using INI syntax:

```
...
[lrs]
lib_device = /dev/changer
...
```

- It will contain parameters for:
  - The connection to the database
  - I/O scheduling, SCSI commands, LTFS operations, tape/drive compatibility, …
  - Store layer: layout parameters, alias, I/O parameters: block size, …

Phobos Dæmon

- Process that communicates with local Phobos clients through a UNIX socket
- Necessary to perform I/O: put, get, format
- Some operations can inform the local Daemon: dir/drive add, lock, unlock for example
  - They will not fail if the daemon is not available
  - These operations manage resources that are bound to a node such as a device
    - They have to be executed on the correct node
- Execution:

```
systemctl start phobosd
```

or

```
phobosd
```

Phobos features illustrated

Easy setup – add/format

- Drive setup

```
phobos drive add --unlock /dev/st1
```

- Tape addition & formatting:

```
phobos tape add -t lto6 [073200-073222]L6

phobos tape format --nb-streams 3 --unlock [073200-073222]L6
```

- The **add** and **format** commands are also available for **dir** and **rados_pool** families
- Unlike tapes, those families do not have an equivalent to drives

Phobos features illustrated

Lock/unlock

- Drive lock

```
phobos drive (un)lock /dev/st1
```

  - These operations will notify the local phobosd if online to update its internal information

- The same for tapes:

```
phobos tape (un)lock [073200-073222]L6
```

- Also works for POSIX directories (dir) and RADOS pools (rados_pool)

Phobos features illustrated

List

- List resources stored on the database:

```
phobos (tape|dir|drive|rados_pool|object|extent)
list
```

- The format of these commands can be modified:
  - --output: comma separated list of fields (see the help of each command for the exact list)
  - --format: output format. Any of human/xml/json/csv/yaml
- Some commands have additional query filters. Use phobos <type> list --help for more details. For example, list all the objects that have an extent on tape : `073200L6`

```
phobos extent list --name 073200L6 --output oid
```

Phobos features illustrated

Status

- Drive status

```
$ phobos drive status
|   address | device   | media    | mount_path       | name     | ongoing_io   |     serial |
|-----------|----------|----------|------------------|----------|--------------|------------|
|         4 | /dev/sg5 |          |                  | /dev/st4 |              | 1800984167 |
|         5 | /dev/sg6 | Q00000L6 | /mnt/phobos-sg6  | /dev/st5 | False        | 0749334872 |
|         6 | /dev/sg7 | Q00001L6 |                  | /dev/st6 | False        | 1296305820 |
```

- Display the current state of the drives handled by the daemon
- This command needs to communicate with the local daemon

Phobos features illustrated

Put

- Attaching arbitrary attributes to objects

```
phobos put /path/to/file objid1 --metadata \
    "cksum=md5:7c28...5e3e,user=foo"
```

- Consulting

```
phobos getmd objid1
```

- Filtering

```
phobos object list --metadata "user=foo" "obj*"
```

```
| oid    | user_md                                          |
|--------|--------------------------------------------------|
| obj01  | {"user": "foo", "crtime" : "132948897"}|
| obj02  | {"user": "foo"}                                  |
```


IO-SEA

Phobos features illustrated

- **Get**

```
phobos get objid1 /path/to/file
```

- **Mput**

  Writing multiple objects at once:

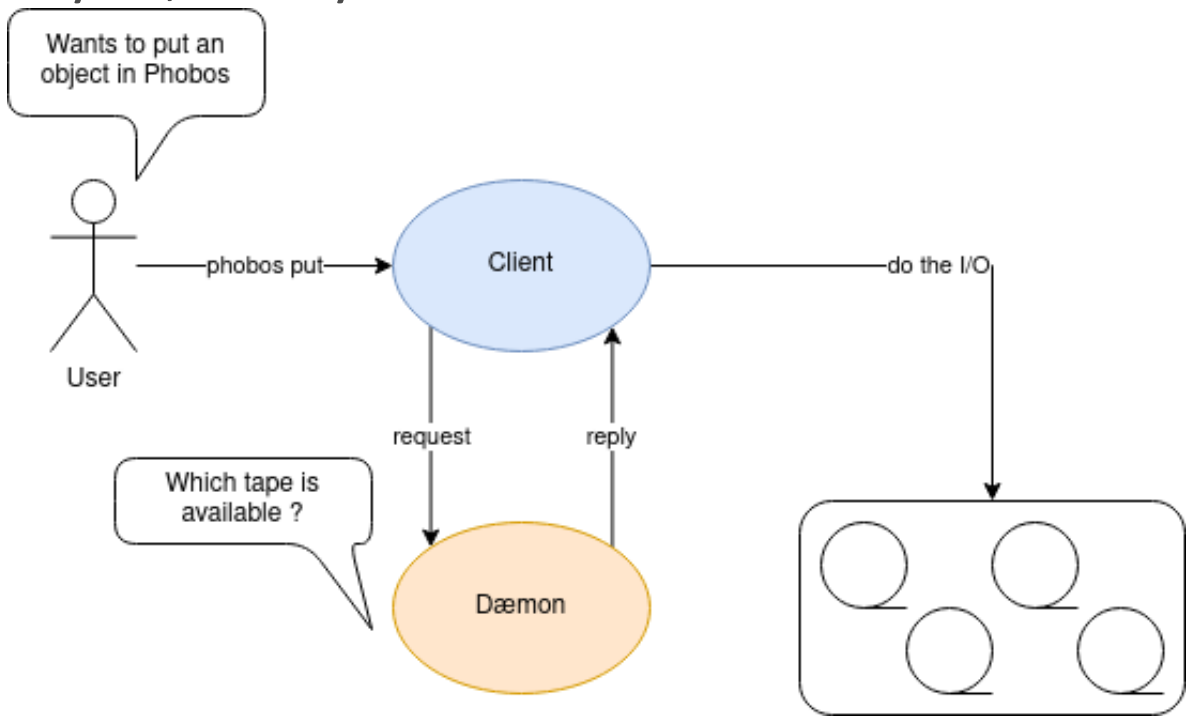
```
$ cat mput_file
/etc/hosts hosts foo=bar,baz=foobar
/etc/hostname hostname time=1669734861

$ phobos mput mput_file
2022-11-29 15:15:39,677 <INFO> PUT operation for oid:'hosts' succeeded
2022-11-29 15:15:39,765 <INFO> PUT operation for oid:'hostname' succeeded

$ phobos object list -o oid,user_md
| oid      | user_md                              |
|----------|--------------------------------------|
| hosts    | {"baz": "foobar", "foo": "...} |
| hostname | {"time": "1669734861"}               |
```

Phobos Dæmon

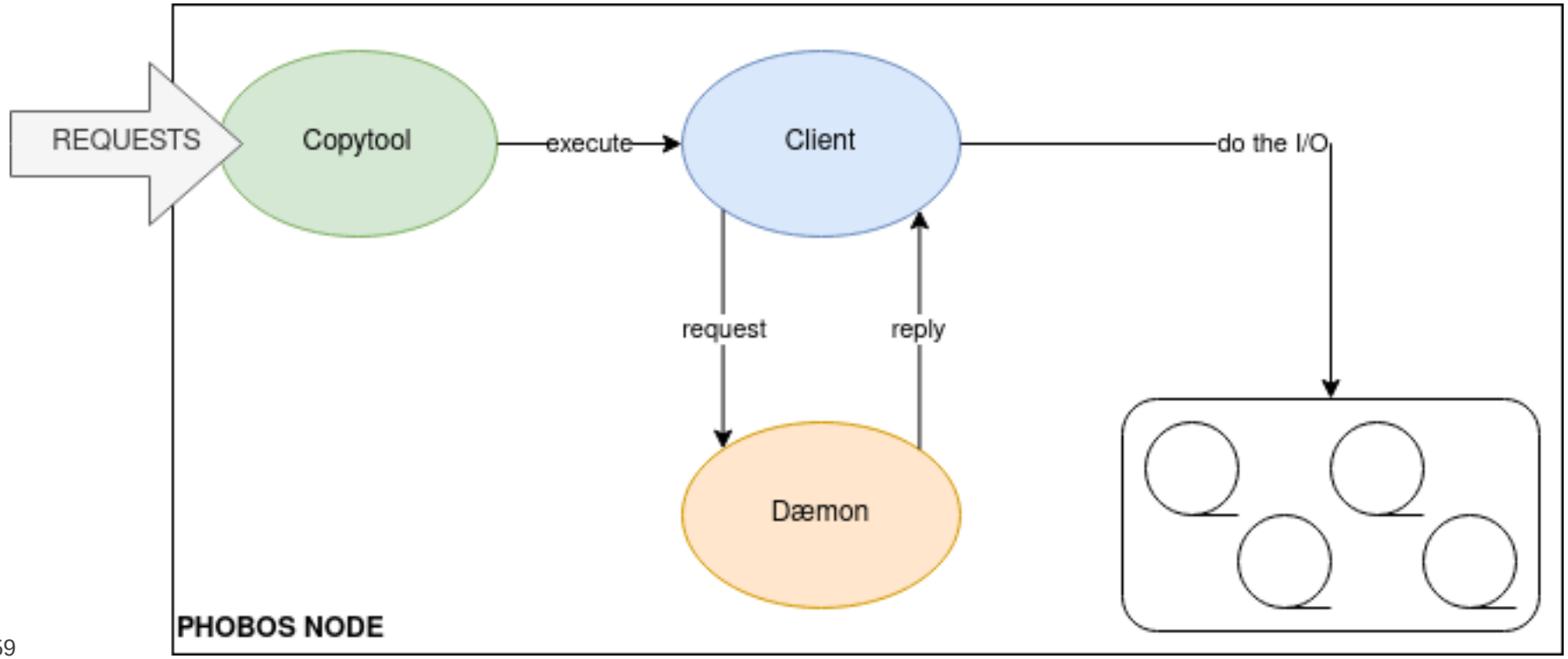
- To be effective, a service of Phobos, called **phobosd**, need to be started
- It centralizes all requests on the central node and schedules them in an optimized way
- It does not process any IO, it only reserves media



Phobos Node/Server

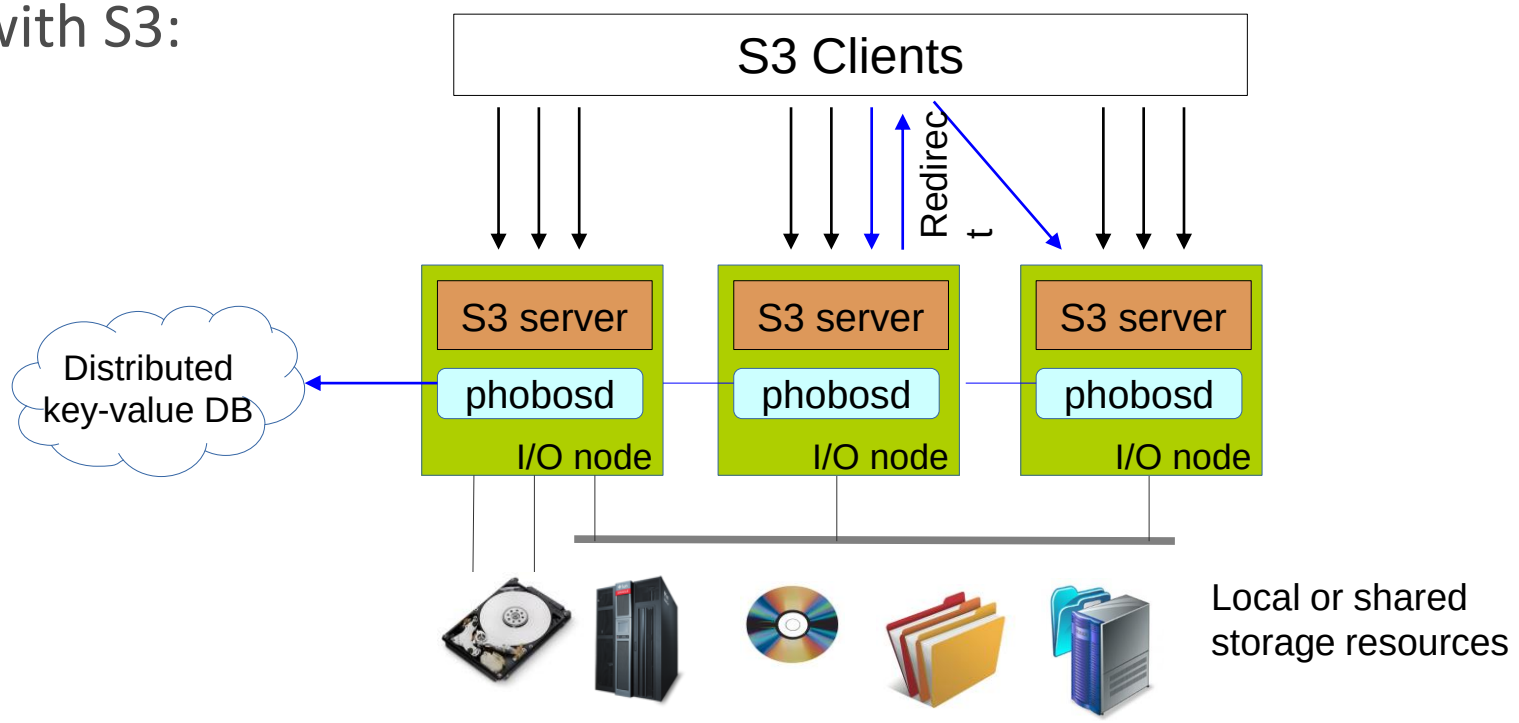
A Phobos Server is composed of:

- A set of devices to access a [shared] media library
- A Phobos service (phobosd) to schedule I/Os
- A tool/script which receives I/O requests and call Phobos client API



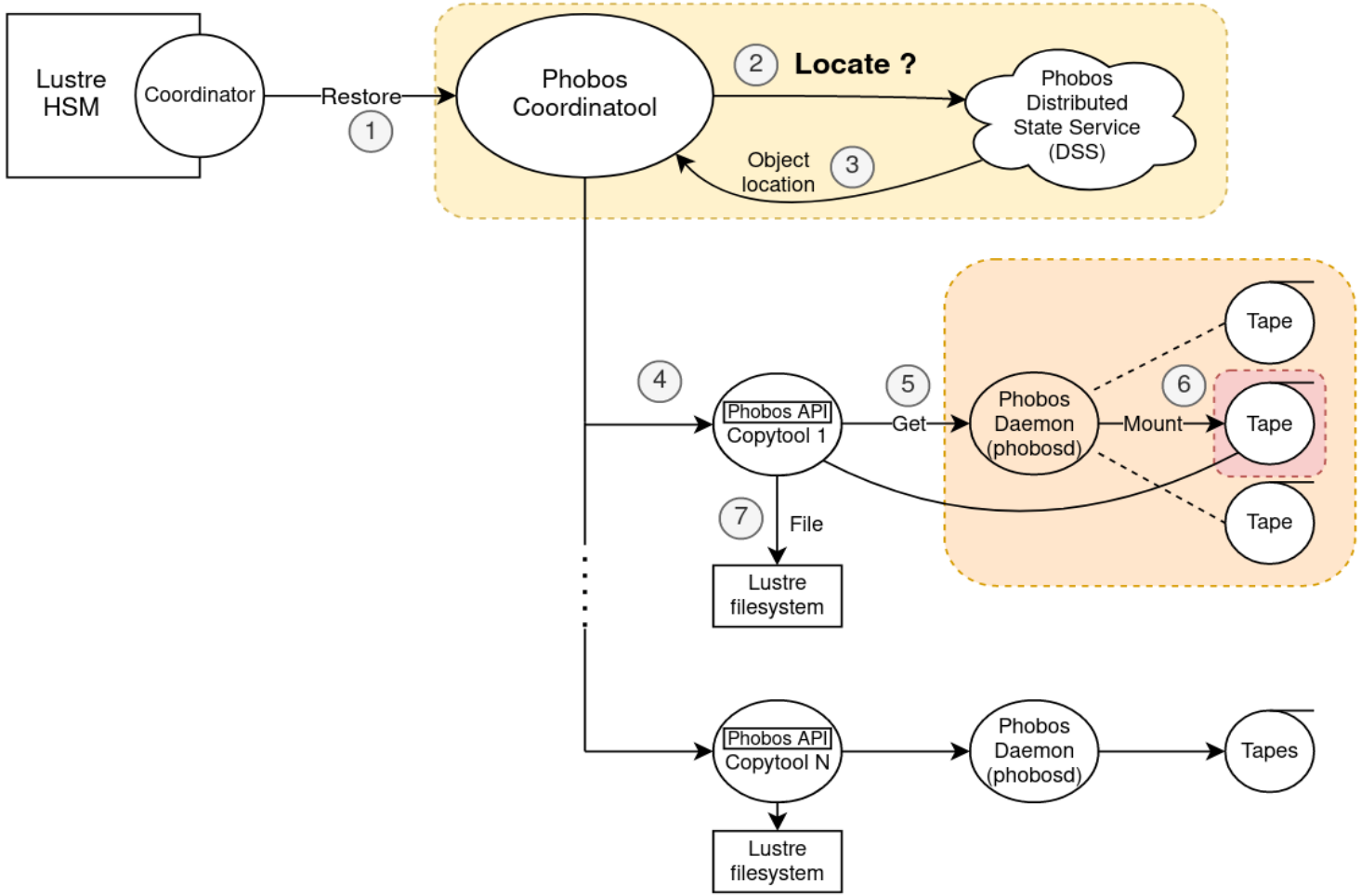
Distributing Phobos

- Phobos can run on multiple servers, using a common database
- I/O distribution relies on the “phobos_locate” feature to direct the I/O to the right Phobos server
- The use of this feature is up to the Front-end
- Example with S3:



Distributing Phobos

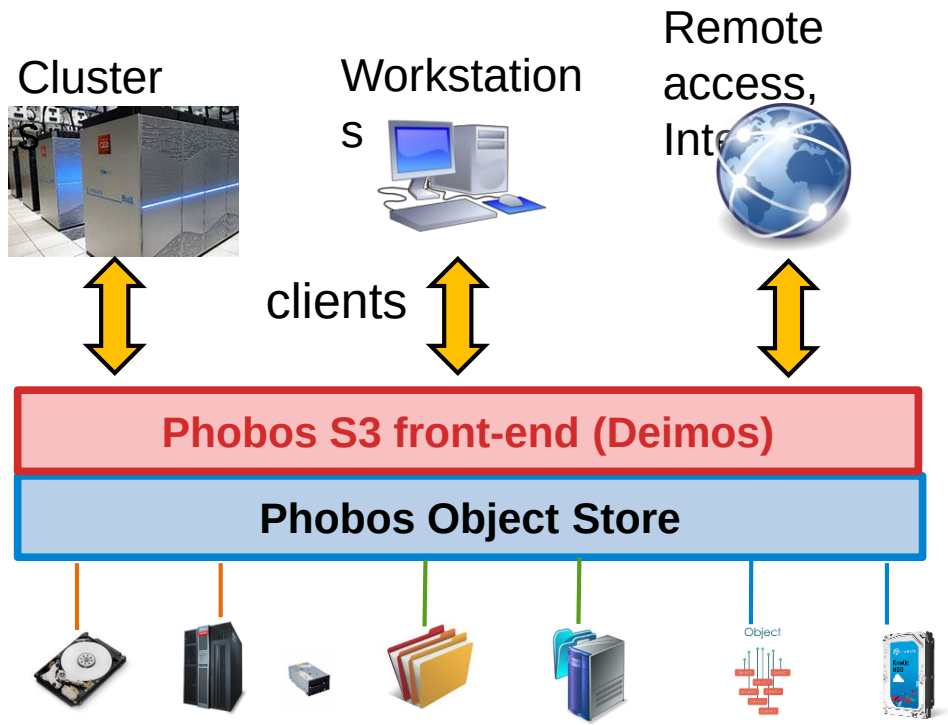
- Other example with Lustre/HSM:



S3 server

Object store with an S3 front-end

- S3 interface exposed to end-users
- Phobos: high-performance, scalable storage
  - Can manage a wide variety of high-capacity storage, including tape libraries
  - Provides an easy/uniform management of the storage resources



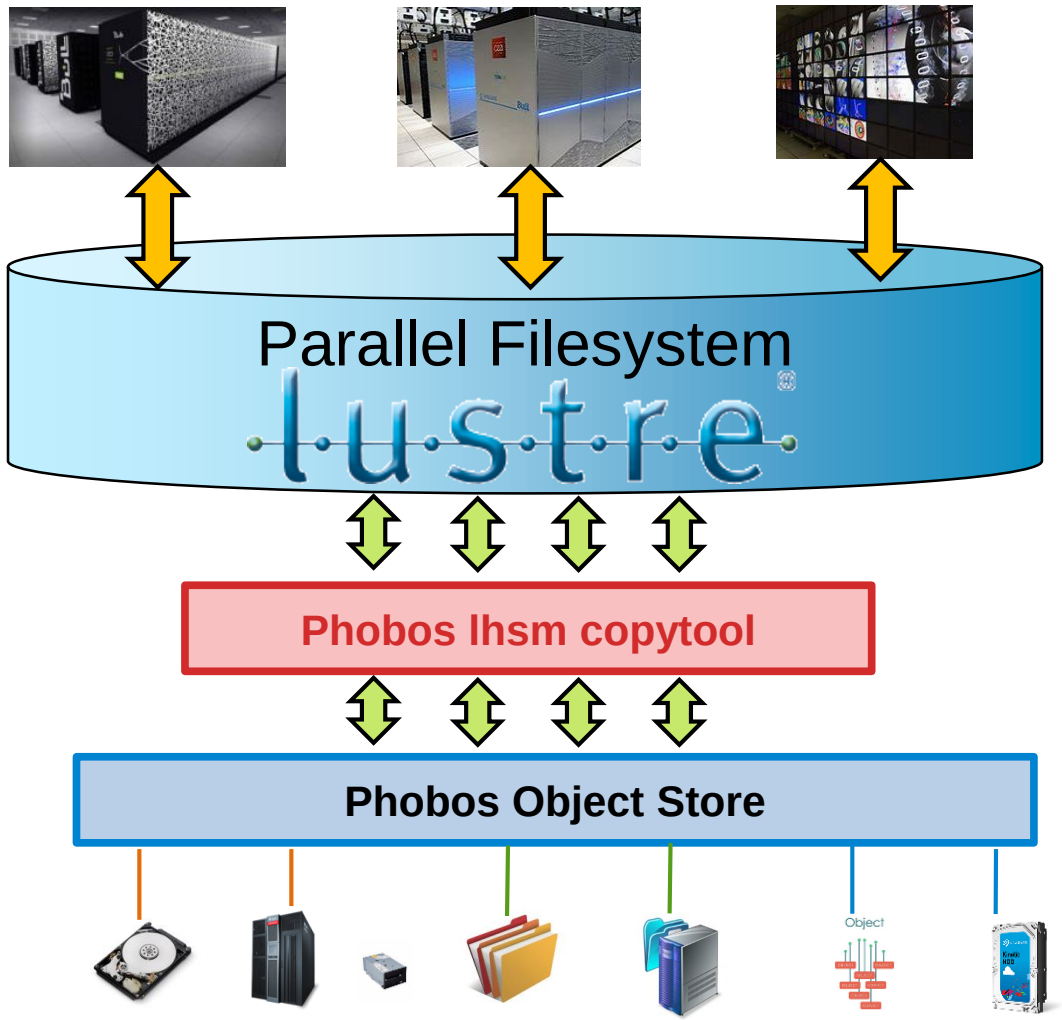
- Phobos' S3 front-end developed by ICHEC: https://git.ichec.ie/performance/storage/deimos

Lustre/HSM backend

Lustre HSM backend

- Lustre: filesystem user front-end
- Phobos as high-capacity backend (hierarchical storage)

- **In production this year at CEA**



Resource tuning: tags

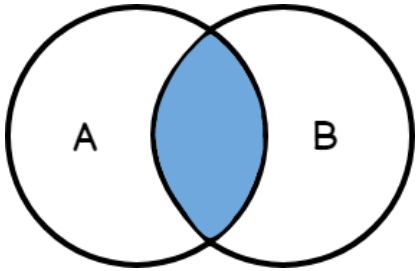
- Tagging resources

```
phobos tape update --tags project1,classB [073000-073099]L8
```

- Pushing data to specific resources:

```
# push data to any medium with tag "classB"
phobos put --tags classB /path/to/file  objid1

# push data to a medium with both tags "project1"
# and "classB"
phobos put --tags project1,classB /path/to/file objid2
```



Resource tuning: permissions

- Setting permissions

```
phobos tape set-access +PGD 07300[0-9]L8
phobos tape set-access -- -P 0730[10-99]L8
```

- Checking permissions

```
phobos tape list -o name,put_access,get_access,delete_access
```

```
| name   | put_access | get_access | delete_access |
| 073000 | True       | True       | True          |
...
| 073009 | True       | True       | True          |
| 073010 | False      | True       | True          |
...
| 073099 | False      | True       | True          |
```


IO-SEA

Put with family/layout

- Targeting a family:

```
phobos put --family tape file/to/put objid
```

- Using a layout:

```
phobos put --layout raid1 --lyt-params repl_count=2 \
    file/to/put objid
```

Aliases configuration

Multiple aliases defined through the configuration file

- Can set:
  - Target family
  - Target media tags
  - Data layout and its parameters

- Configuration example

```
[alias “simple”]
family = tape
layout = raid1
lyt-params = repl_count=1
tags = foo-tag
```

Put with alias/metadata

- Using alias

```
phobos put /path/to/file --alias simple objid1
```

- Using metadata

```
phobos put --metadata dataset=test,value=42 \
    /path/to/file objid2
```

- Listing objects with metadata

```
phobos object list --output oid,user_md
```

```
| oid    | user_md                                  |
| objid1 | {}                                       |
| objid2 | {"value": "42", "dataset": "test"} |
```

```
phobos object list --metadata dataset=test
```

```
objid2
```

Versioning

- Object uniqueness

```
phobos put /path/to/file objid
```

→ fails if *objid* exists

- Creating new object version

```
phobos put --overwrite /path/to/file objid
```

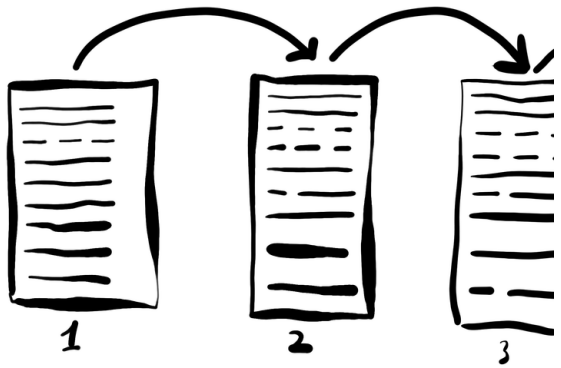
→ creates a new version of *objid*

- Listing object versions

```
phobos object list --deprecated objid
```

- Retrieving an old version

```
phobos get --version 1 objid file.out
```



IO-SEA

Object deletion

- Deletion

```
phobos del objid
```

- Canceling deletion

```
phobos undel oid objid
```

- Listing deleted versions

```
phobos object list --deprecated objid
```

- Retrieving a deleted version

```
phobos get --uuid ABC12312 --version 2 objid
```

Available until the media is “repacked”

IO-SEA

Object location

- Locating an object

```
phobos locate objid
```

- Proposing a suggestion in case a choice is possible

```
phobos locate --focus-host $HOSTNAME objid
```

- Getting the object only if we are on the best node

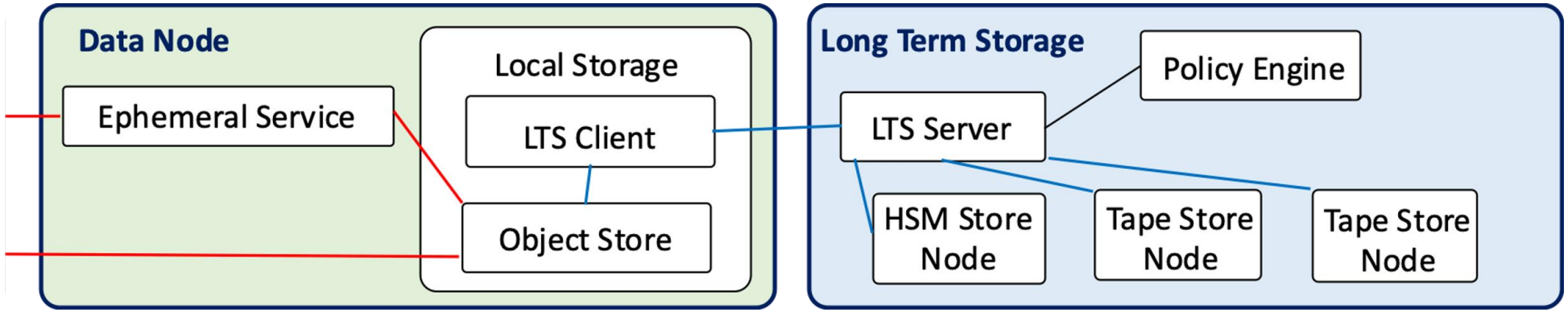
```
phobos get --best-host objid
```

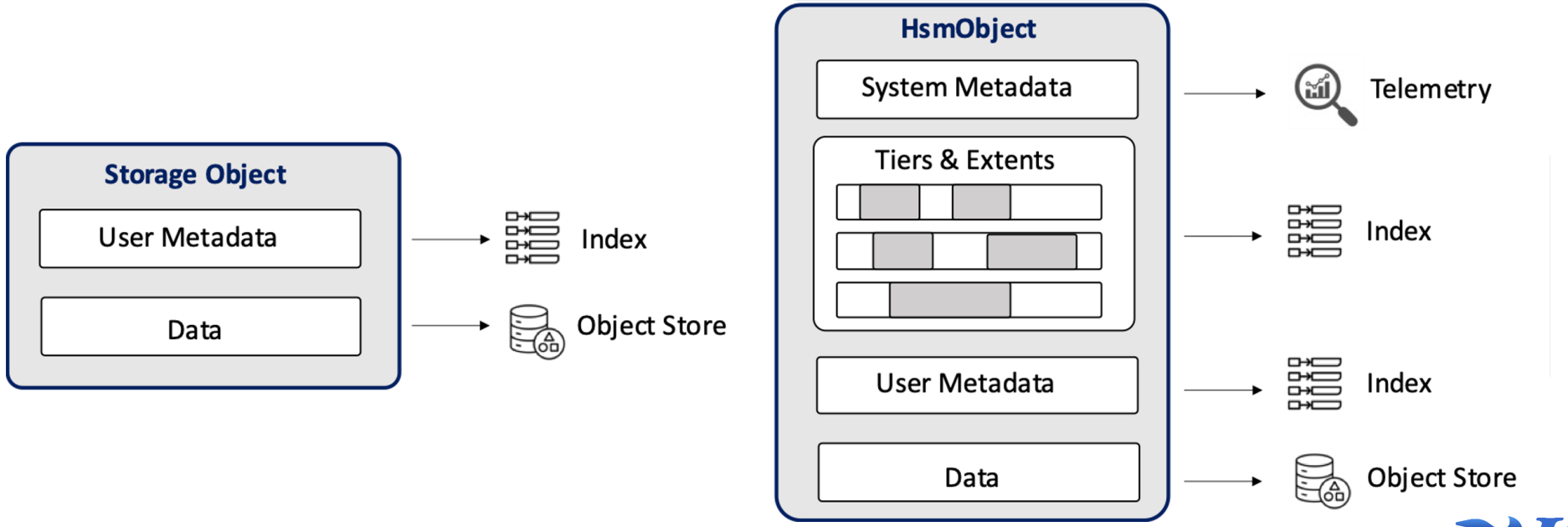
- Locating a medium

```
phobos tape locate 073002L8
```

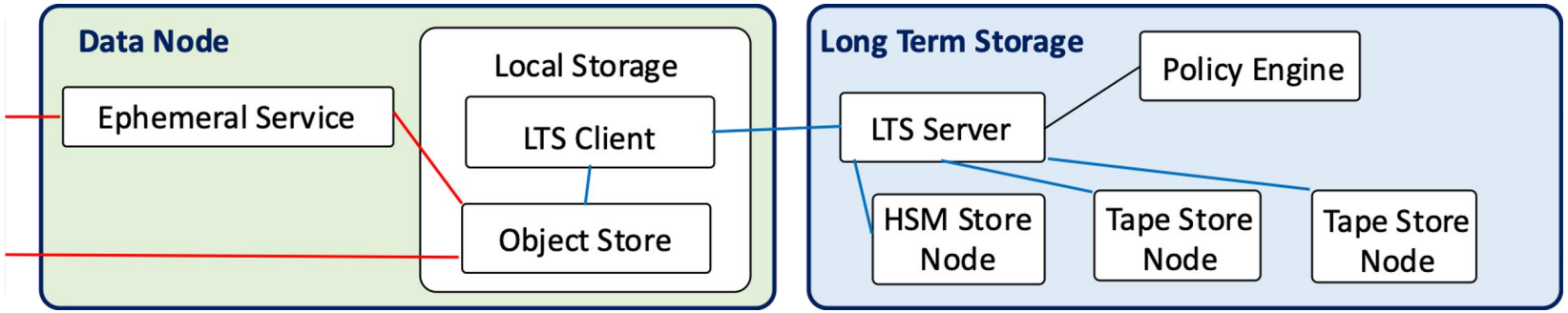
HSM for Exascale

HSM for Exascale





HSM for Exascale



SEAGATE CORTX

Motr

Distributed Object Storage Prototype System

https://github.com/Seagate/cortx-motr

cea

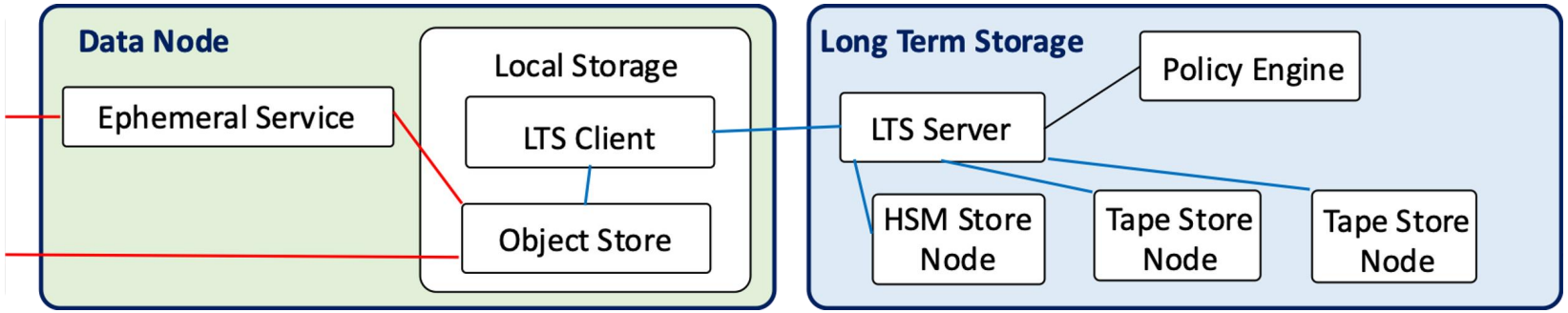
Phobos

Object Storage on Tapes

https://github.com/cea-hpc/phobos

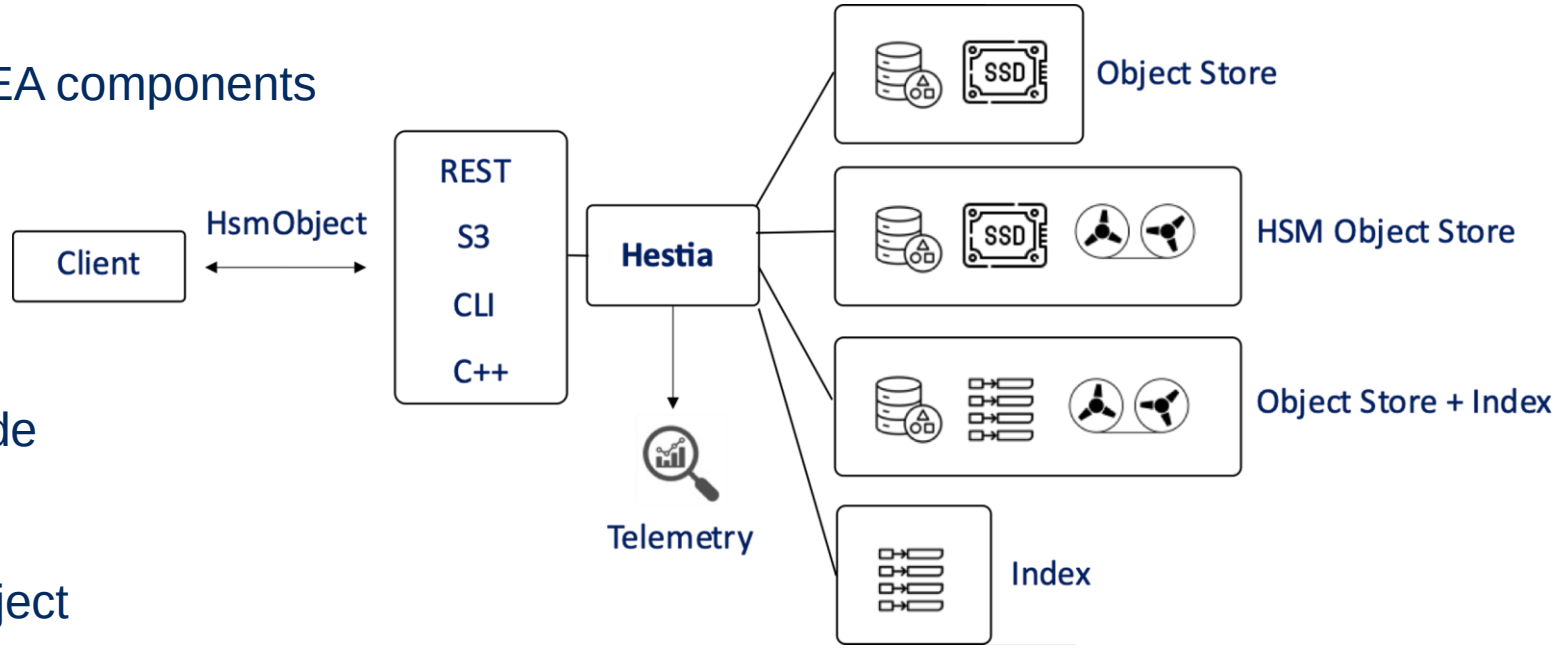
IO-SEA

HSM API for Exascale – Hestia Component



Goals:

1) Support interaction of IO-SEA components via a 'HSM API'
2) Minimal impact on:
   - Performance
   - IO-SEA component code
   - Introspection ability
3) Be extensible beyond the project



Hestia Component

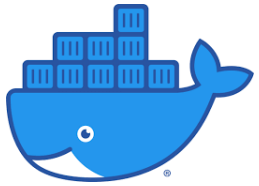
Hestia Implementation

Modern C++ Library – CMake Build

Standard Dependencies and Formats

- Curl, LibS3
- Interchangeable WebServer
- Yaml/JSON
- Spdlog/CLI11

CI with containerized builds

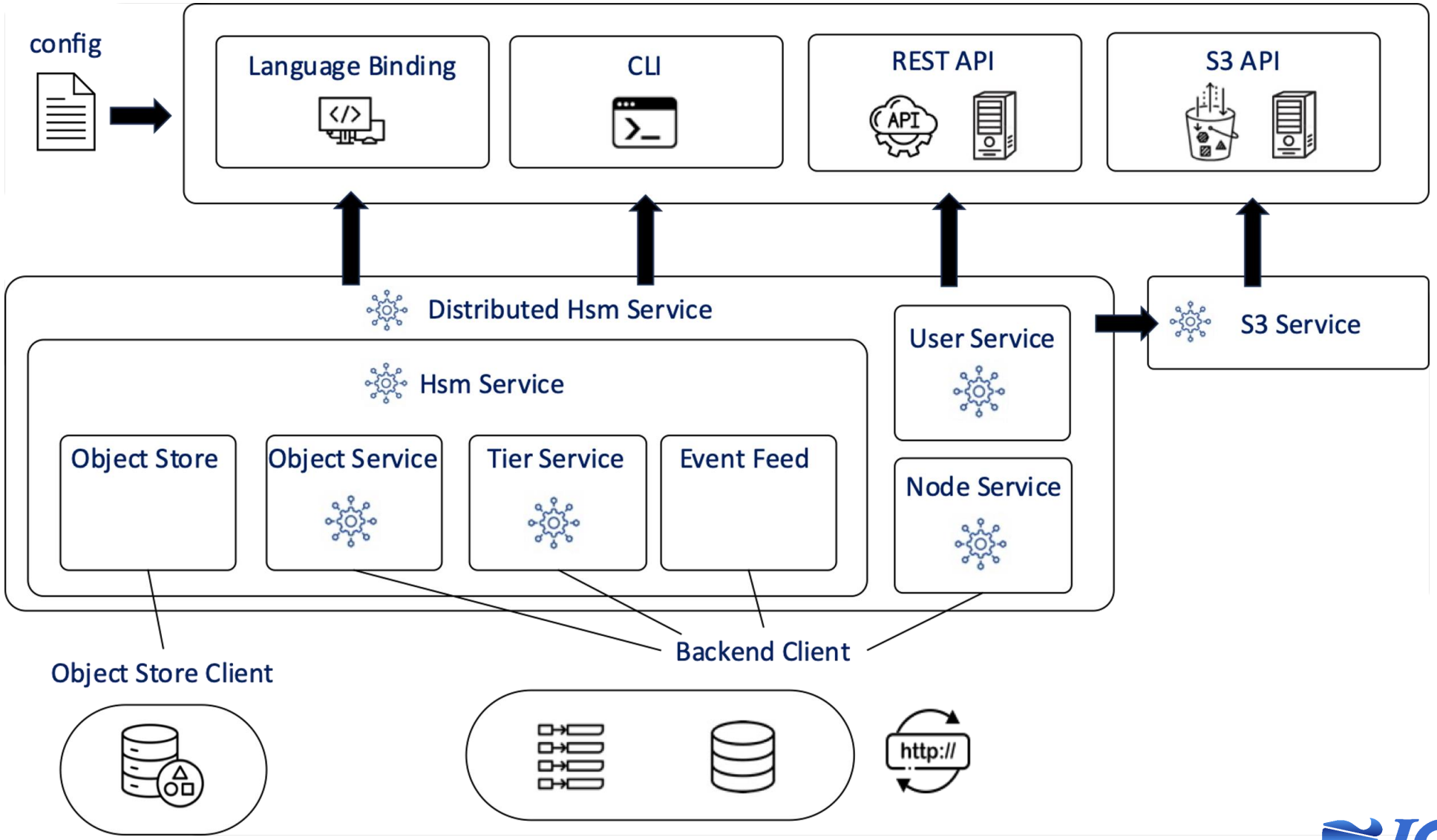


Open Source – MIT License

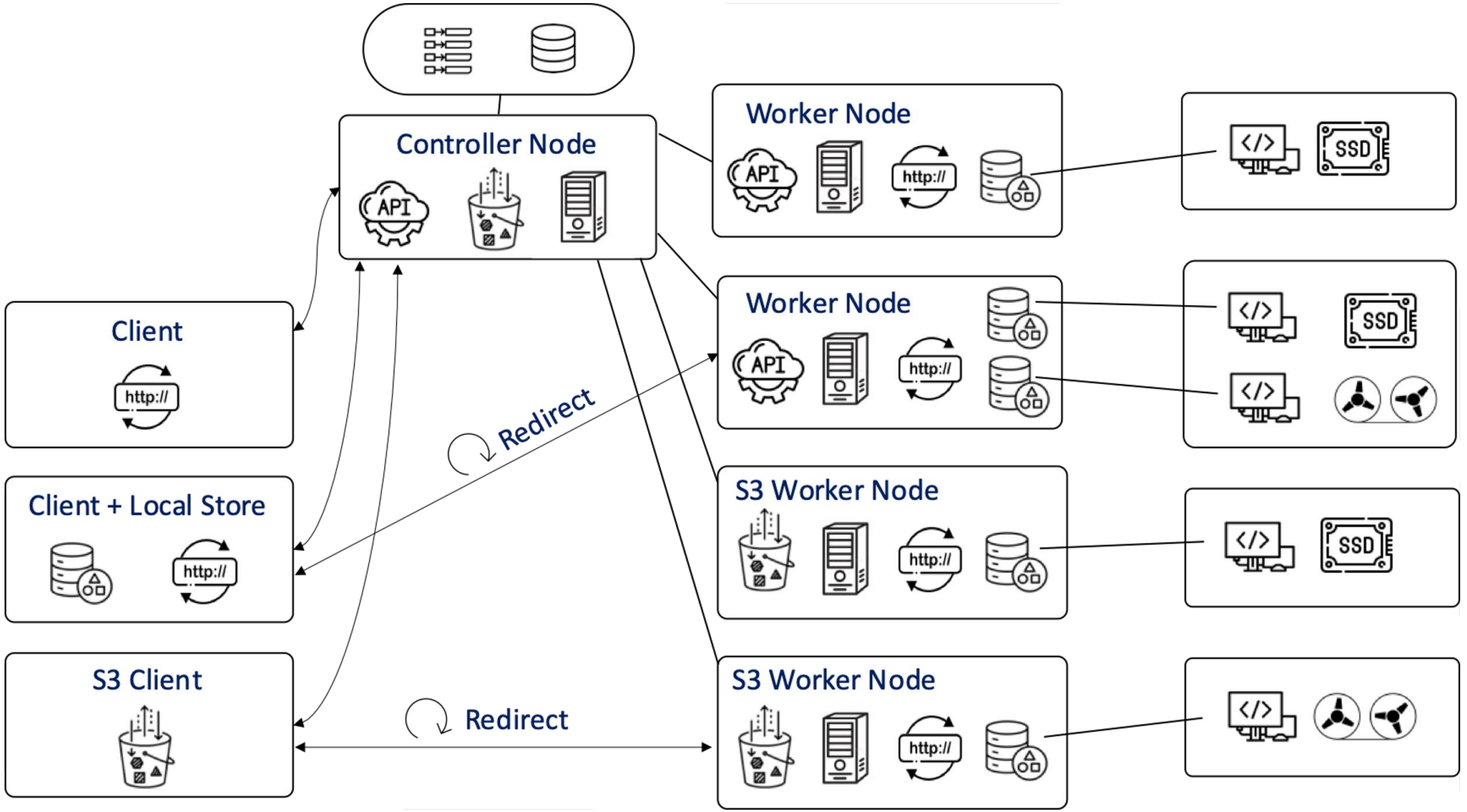
https://git.ichec.ie/io-sea-internal/hestia

IO-SEA

Hestia Components



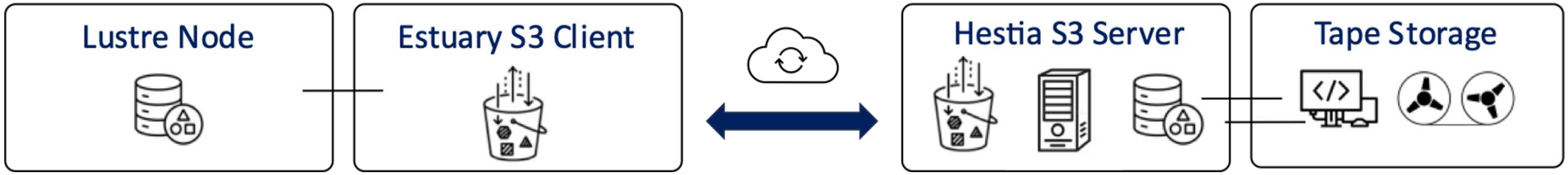
Hestia System



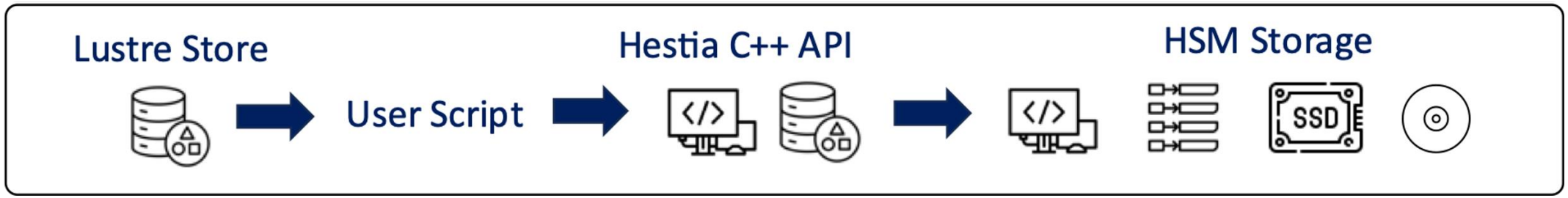
Hestia: Applications

Applications – Data Mover

Lustre to Remote Tape



Lustre to Local HSM Object Store

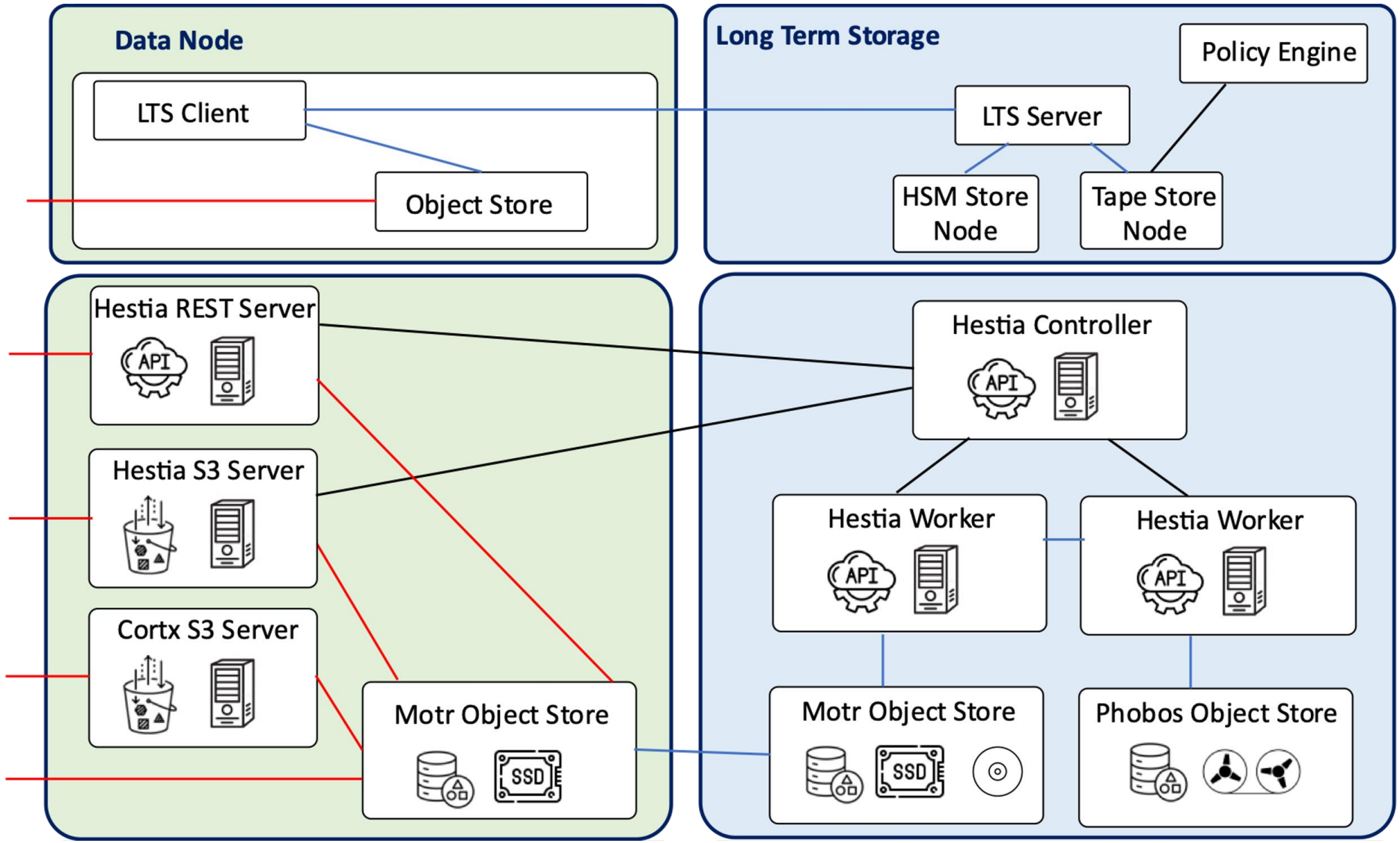


https://git.ichec.ie/performance/storage/estuary

https://github.com/ComputeCanada/lustre-obj-copytool

IO-SEA

Applications – IO-SEA



Resources

Software

IO-SEA: https://github.com/io-sea

Hestia: https://git.ichec.ie/io-sea-internal/hestia

DASI: https://github.com/ecmwf-projects/dasi

Motr: https://github.com/Seagate/cortx-motr

Phobos: https://github.com/cea-hpc/phobos

Contact

ICHEC: https://www.ichec.ie – james.grogan@ichec.ie

IO-SEA: https://iosea-project.eu/contact/

DASI: Data Acces Storage Interface

DASI

IOSEA CROSS-WP SESSION

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955811. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, the Czech Republic, Germany, Ireland, Sweden, and the United Kingdom.

Outline

- Concept and Design
- API
- Example Workflow
- Ephemeral Workflow Definition
- POSIX Interoperability

IO-SEA

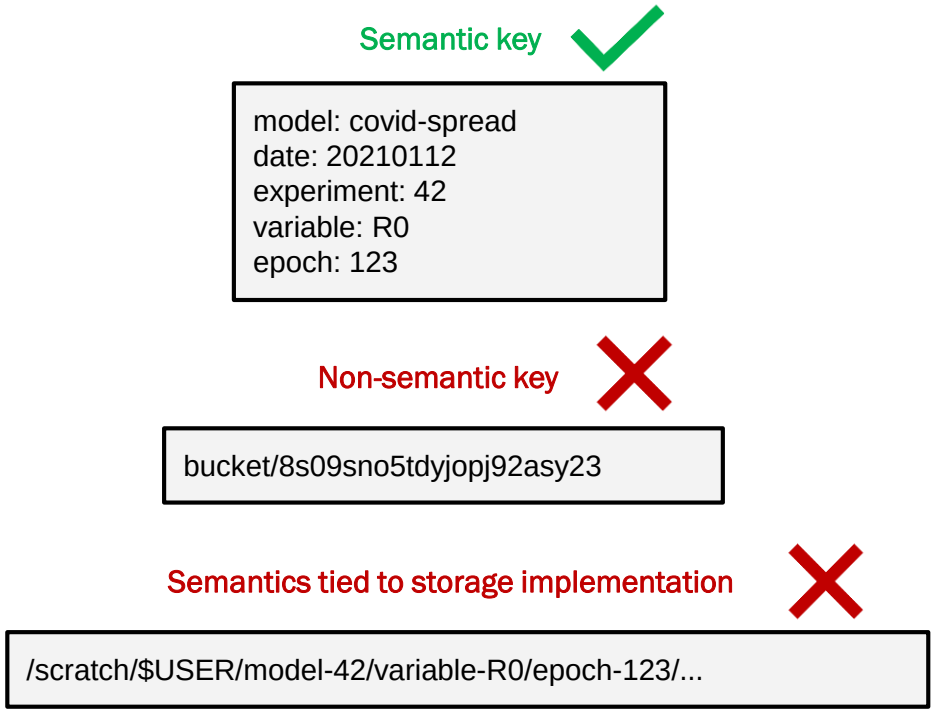
DASI Concept and Design

DASI Concept

- **Data Access and Storage Interface (DASI) will provide a scientifically-meaningful access to data**

- **The key used to index data is a semantic description of the data**
  - **Not just a UUID**
  - **The metadata is used to index and uniquely identify the data**
  - **Ensures data is findable and accessible**
  - **Domain-specific schemas of allowed keys are defined by configuration**

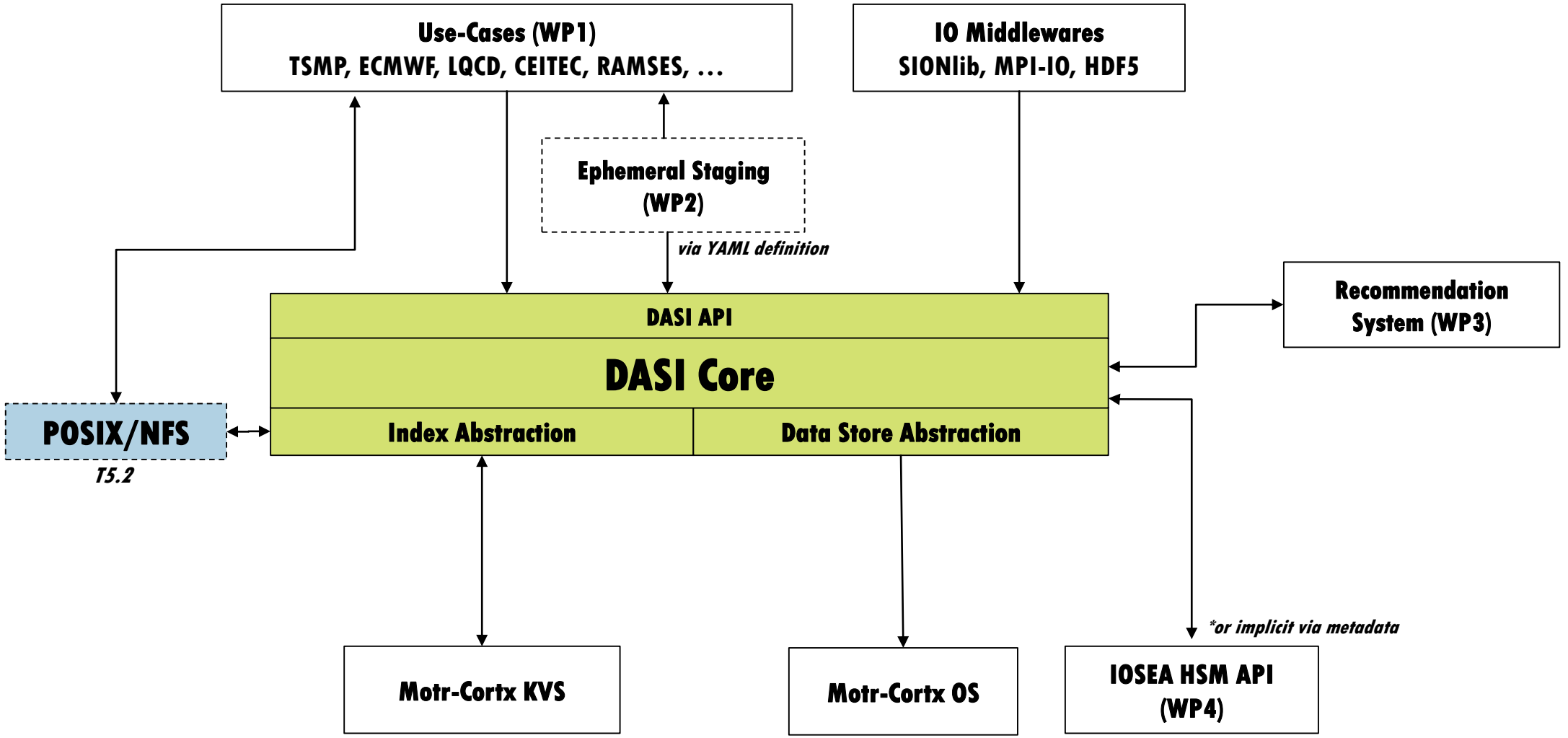
- **The key is detached from the underlying technology**
  - **Flexibility in storage backend with no impact on the user**
  - **Allows optimal usage of storage backend without leaking details to user-space** (***e.g.*** **grouping objects together**)



DASI Use Cases

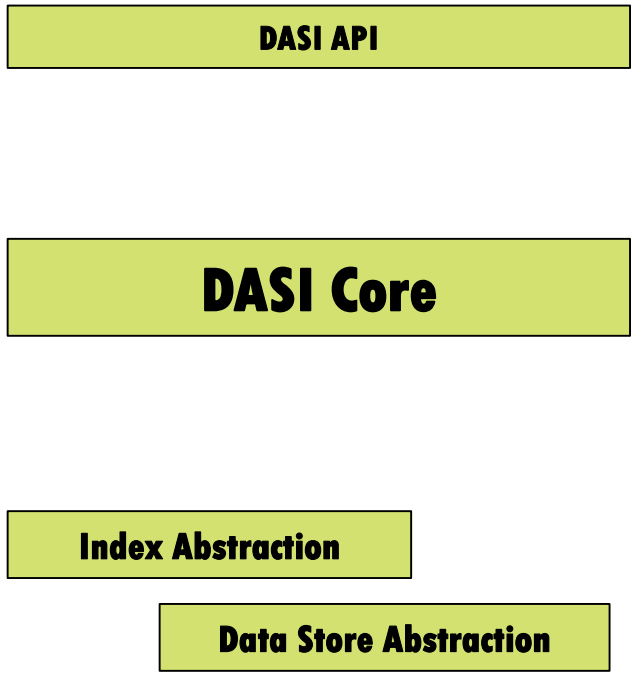
- **CEITEC** will use DASI for raw imagery and processed images
- **LQCD** will use DASI for markov-chain scientific checkpoint files
- **TSMP** will use DASI for output from TSMP model components (Parflow/COSMO/CLM)
- **ECMWF** will use DASI for IFS weather forecasting workflow (raw data and products)
- **RAMSES** will use DASI for post-processing data from Hercule
- **SIONlib** middleware will integrate with DASI

WP5 Requirements and Design



DASI Design

- **DASI API**
  - Frontend abstraction
  - Could allow directly implementing POSIX-like frontend
- **DASI Core**
  - Converts requests into indexable identifiers
  - Expands query requests (ranges, wildcards, etc.)
  - Dispatches between index, datastore, policy engine, user management, monitoring, etc.
- **DASI Index Abstraction**
  - Mapping between keys and object locations in datastore
  - Will be implemented using POSIX, Motr-Cortx KVS
- **DASI Datastore Abstraction**
  - Object-store-like API for raw storage objects
  - Often needs to adapt to be efficient
    - e.g. some backends may group many objects into single files
    - e.g. may need to store hierarchically, but not in the same hierarchy as the collection schema
  - Will be implemented on POSIX, Motr-Cortx OS
- **Other abstractions for Policy, User management, etc.**

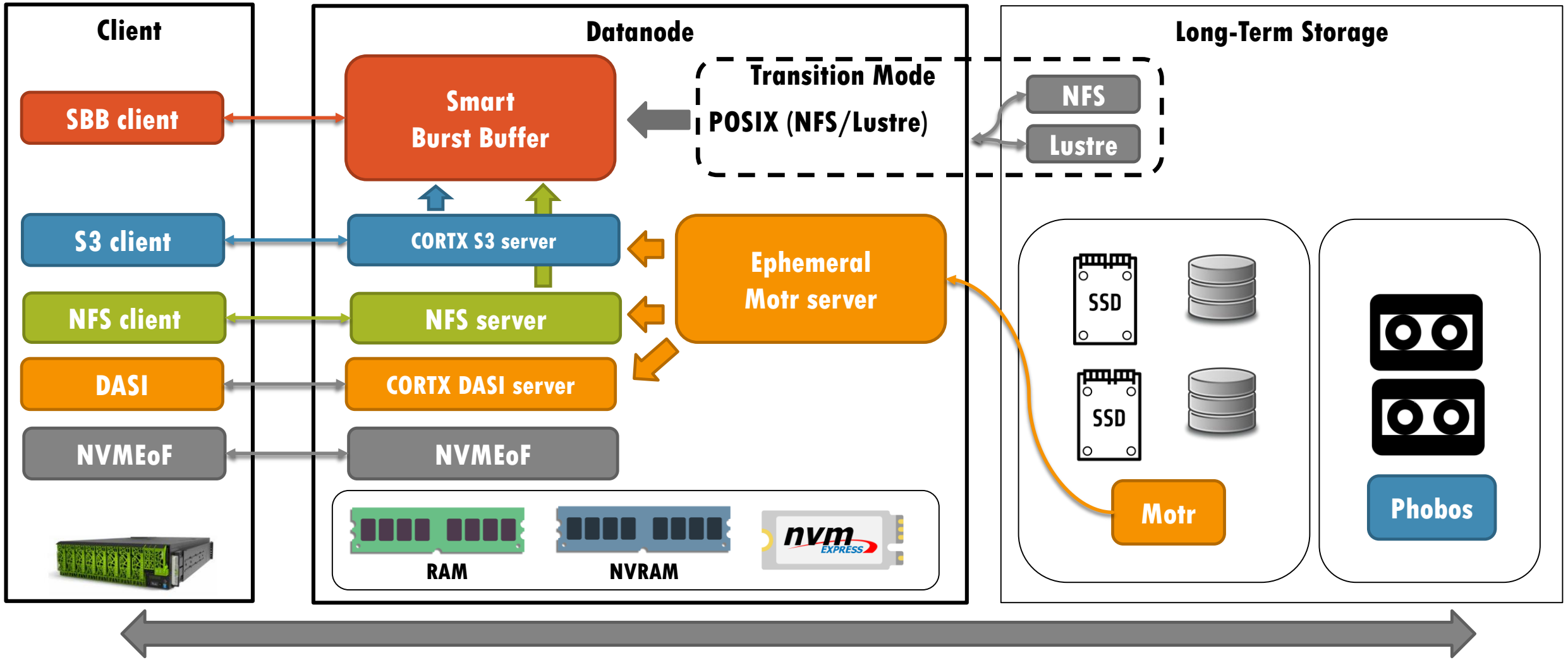


IO-SEA

DASI Deployment

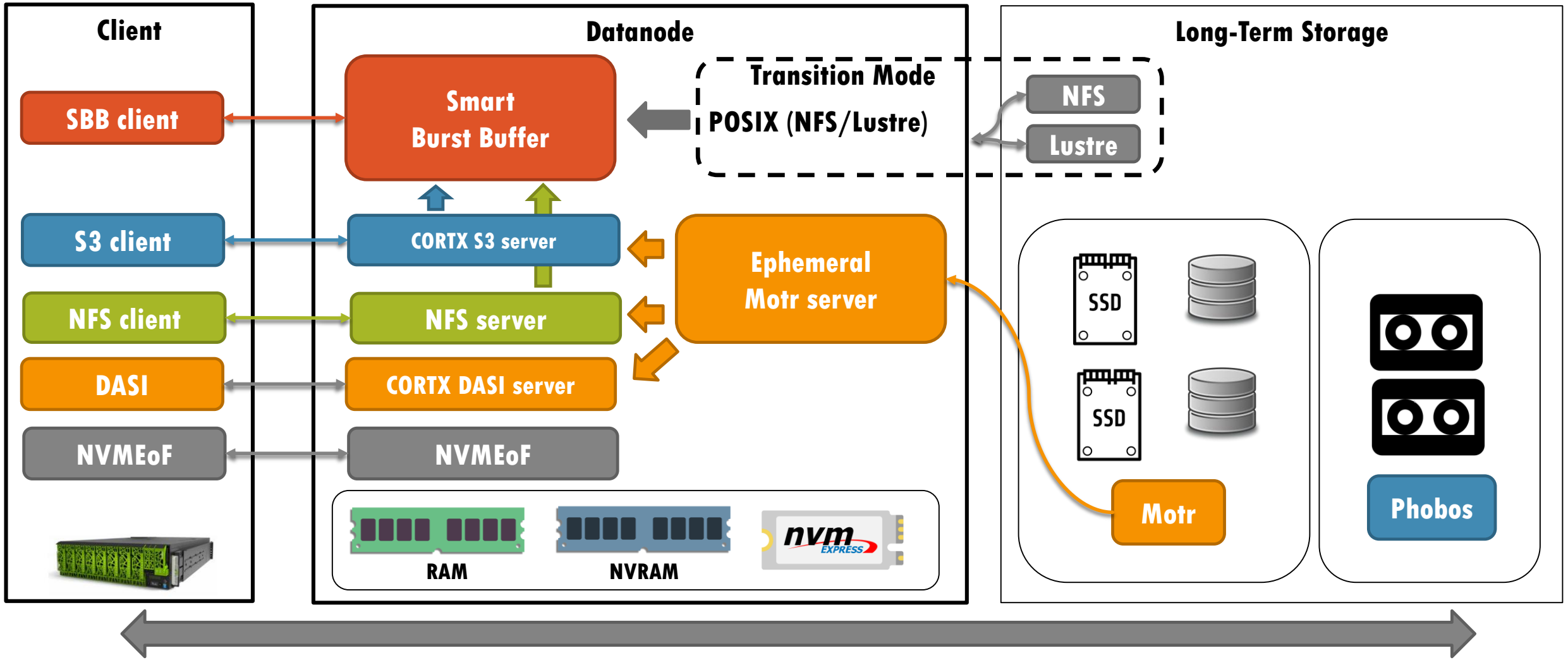
- DASI is not a client-server design
  - DASI is a wrapper around lower-level datastores
  - DASI depends on specific clients (e.g. S3 client for an S3 datastore) behind the scenes
- DASI will not store policies or user authorization data in user-space, this must be handled by the datastore backends
- Motr-Cortx does not really have a client-server design
  - A Motr-Cortx server will be developed and deployed on datanodes, for DASI to talk to

Ephemeral I/O Services



Ephemeral I/O Services

Questions on DASI concept and design?



DASI API and Examples

prototype

DASI API (Python)

```
# Open a DASI session to the service provider and access a dataset

DASI(service=<URI>, collection=<string>) ➔ session


# Insert a bytestream fully identified by metadata

session.put(metadata={}, data=<bytestream>)


# Get/List all data matching a query (e.g. subset of metadata)

session.get(query={}) ➔ [bytestream]

session.list(query={}) ➔ [metadata]


# Set policies on all data matching a query

session.set_policy(query={}, policies={})
```

(and more...)

Collection Schema

```
model: string
date: datetime
experiment: int
epoch: int
variable: string
```

Metadata

```
model: covid-spread
date: 20210112
experiment: 42
epoch: 123
variable: R0
```

Query

```
experiment: 42
epoch: [123,124,125]
```


IO-SEA

prototype

DASI Workflow Example

```
# Open two DASI sessions, one for input and one for output data
input = DASI(service="file:///IOSEA-Mero.yaml", collection="ceitec-raw-images")
output = DASI(service="file:///IOSEA-Mero.yaml", collection="ceitec-processed-images")

# Make a query for input data
query = { "project": "IOSEA", "owner": "CEITEC", "experiment": 12389}  # "sample" not specified, will get all samples
raw_samples = input.get(query) as RawSamplesArray

# Tell the IO service that this data is not likely to be used again
input.set_policy(query, {"hotness":"cold"})

# Process each sample and put to DASI one-by-one, specifying full metadata
for sample in raw_samples:
    processed_sample = do_something(sample)
    metadata = { "project": "IOSEA", "owner": "CEITEC", "experiment": 12389, "sample": sample.name }
    output.put(metadata, processed_sample)
```

schema

project: string
owner: string
experiment: int
sample: string

IO-SEA

prototype

DASI Ephemeral Workflow Definition

```
services:
  - name: nfs_propagators
    cortx_nfs:
      namespace: my-run-2021-12-1-propagators-${ID}
      mountpoint: /mnt/USER/${ID}/propagators
  - name: raw-samples
    dasi:
      service: file:///IOSEA-Mero.yaml
      collection: ceitec-raw-images
      query:
        project: IOSEA
        owner: CEITEC
        experiment: 12389
                sample: *
```

Behind the scenes, a dasi.set_policy will be called on this query to trigger data movement before job execution.

prototype

POSIX Interoperability

- Goal: allow legacy applications to use DASI without modifying source code

- A low-level interface to DASI will convert DASI query to a list of raw datastore object IDs
  - With Motr-Cortx object store, this will provide mapping between DASI objects and Motr-Cortx objects
  - Will accept POSIX-like expression of DASI query

```
/ceitec-raw-images/IOSEA/ceitec/12389/sample-name
```

- NFS Ganesha can use this interface to expose DASI collections as NFS hierarchy

Questions on DASI API, workflow definitions and POSIX interoperability?

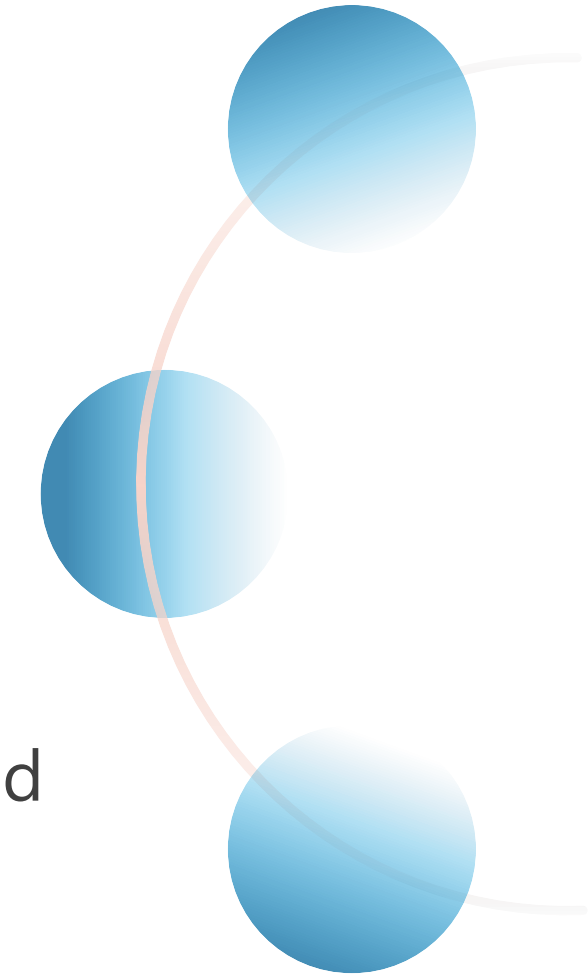
Instrumentation and monitoring

Objectives

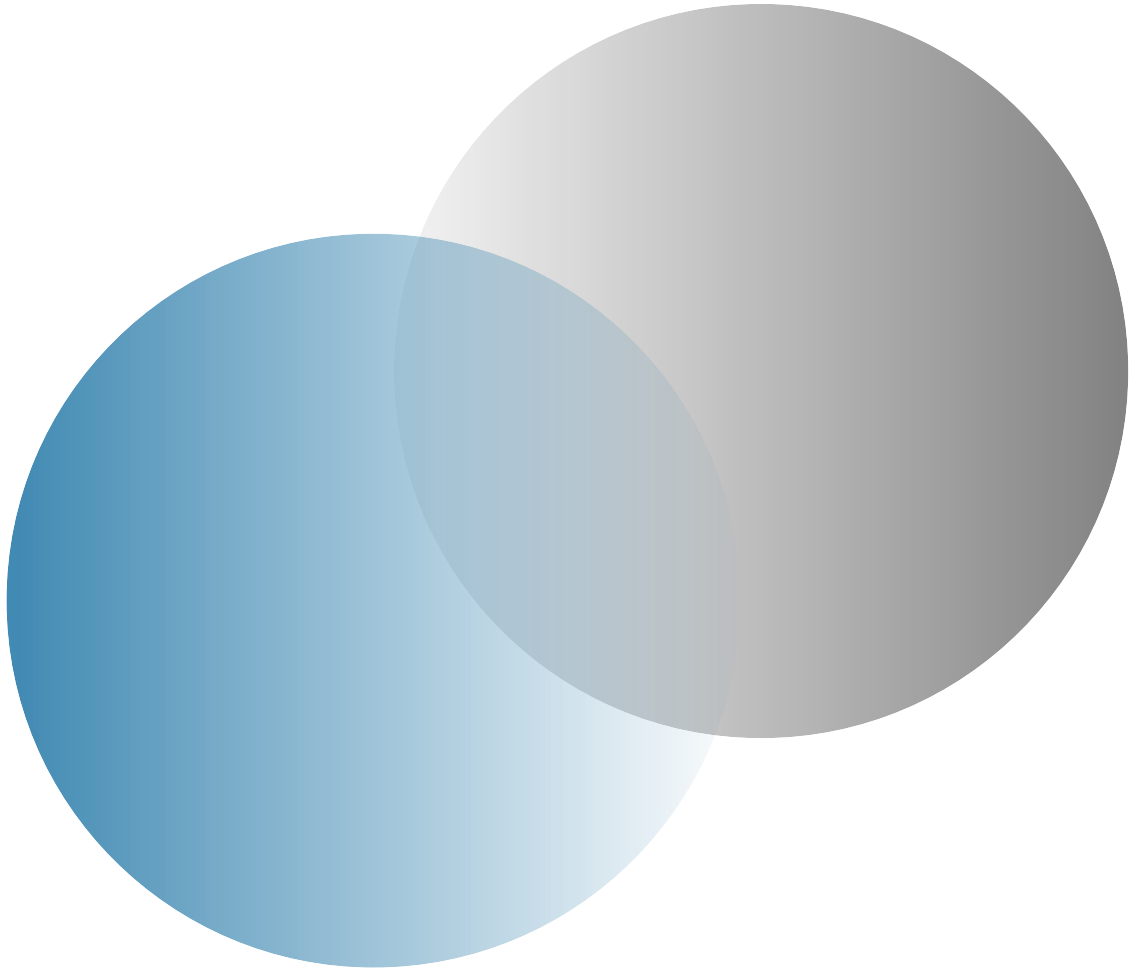
Collect and analyse applications and workflows I/O behaviour (IO Instrumentation)

Collect and analyse infrastructure health (HealthChecker)

Use collected data to recommend dimensioning and data placement decisions (AI based analytics)

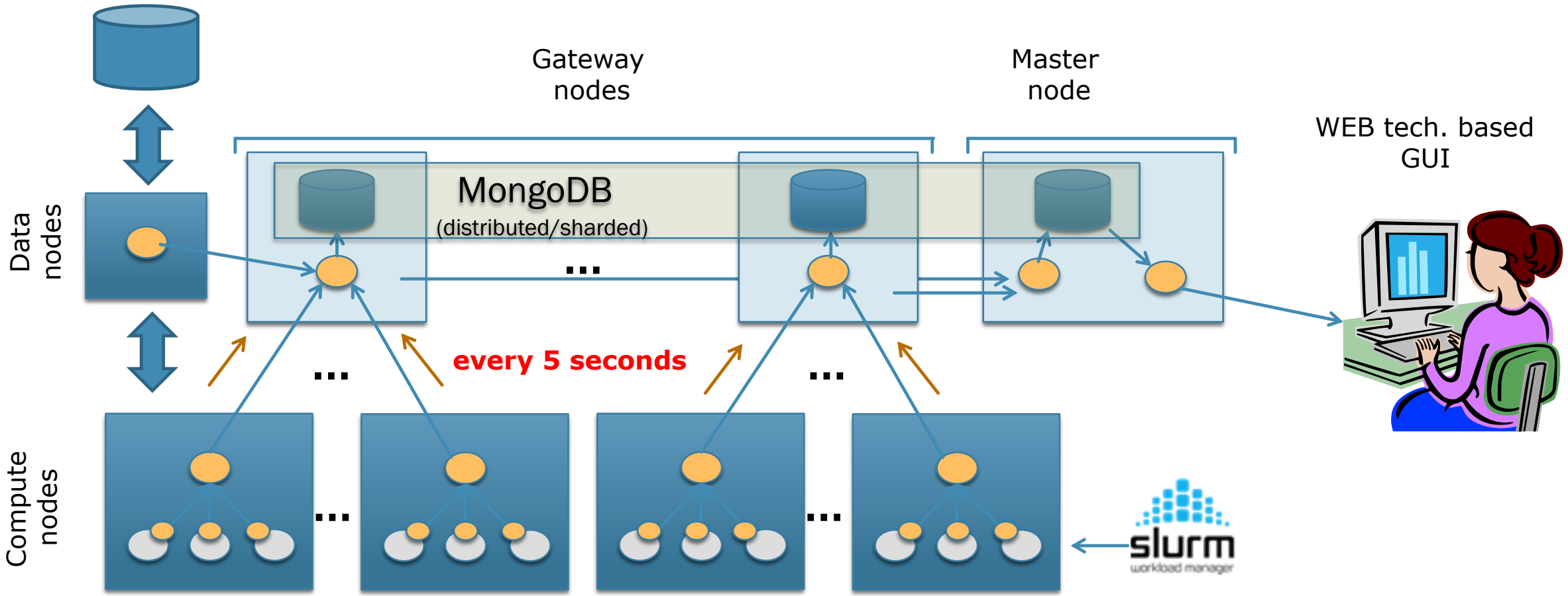


IO Instrumentation



IO Instrumentation Architecture & Components

- Extended in IO-SEA to collect metrics on data nodes



--ioinstrumentation=yes

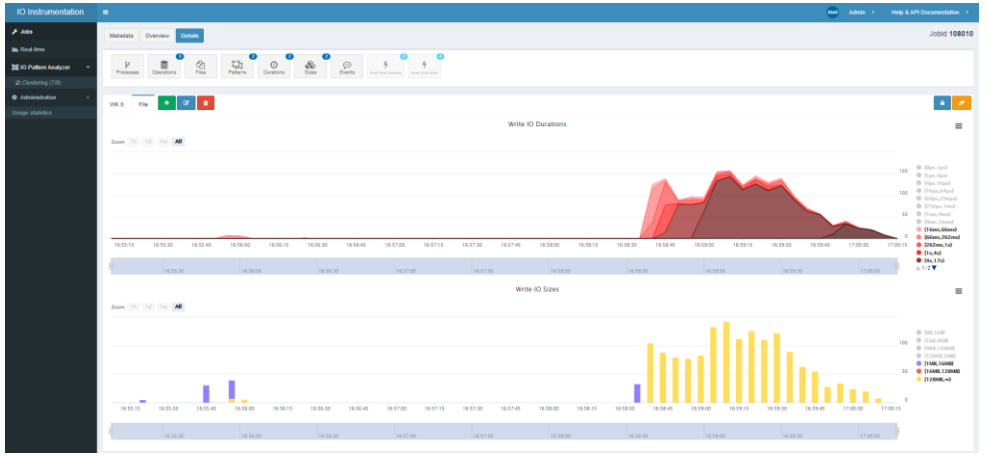
IO Instrumentation:
Collect & Display IO related data

- per job & per workflow Metrics display
  - Job & workflow tables to find jobs to analyze
  - Job Overview
  - Job Metadata
  - Job Details
    - Workspace to dig into overtime metrics

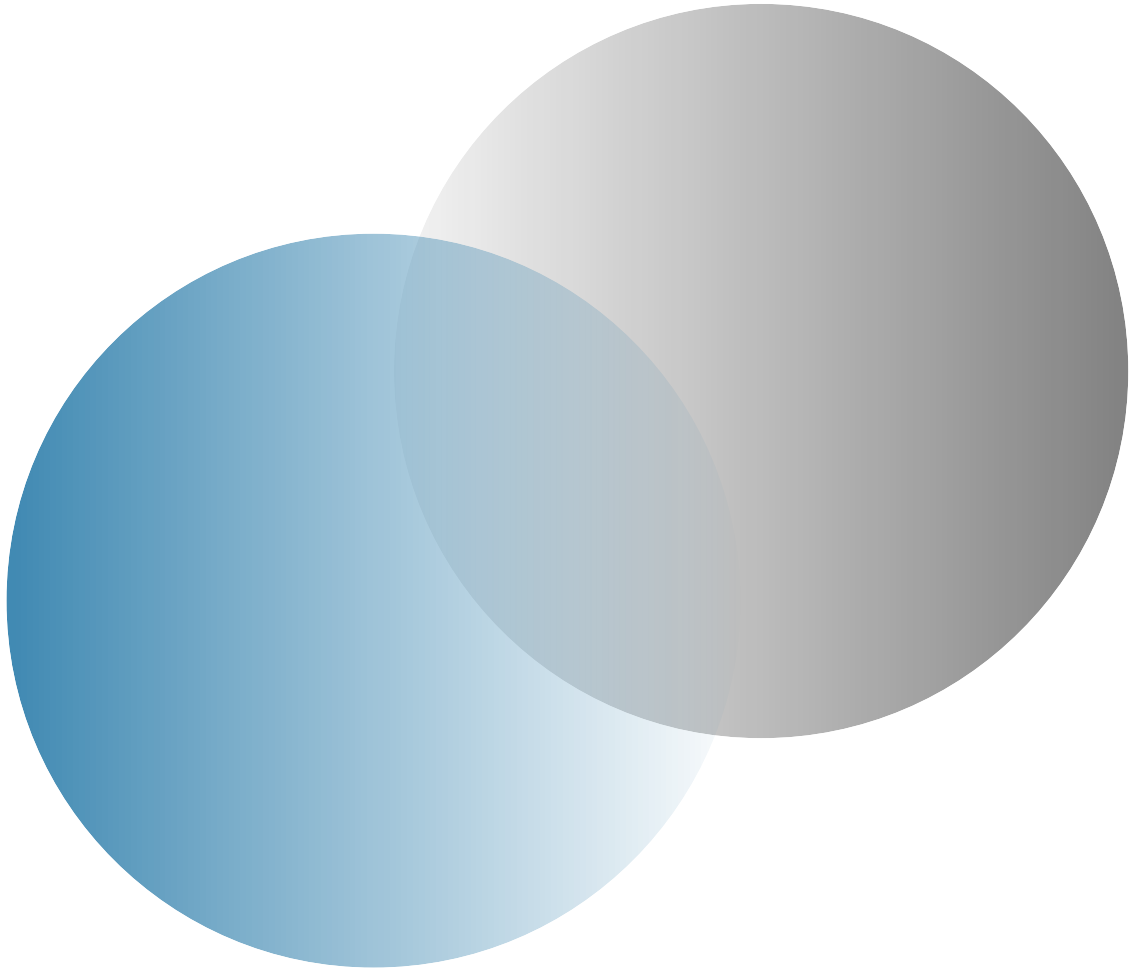
- 2 user profiles
  - Regular user : access only their jobs
  - Admin : see all jobs and access app settings

- Keycloack based user authentication

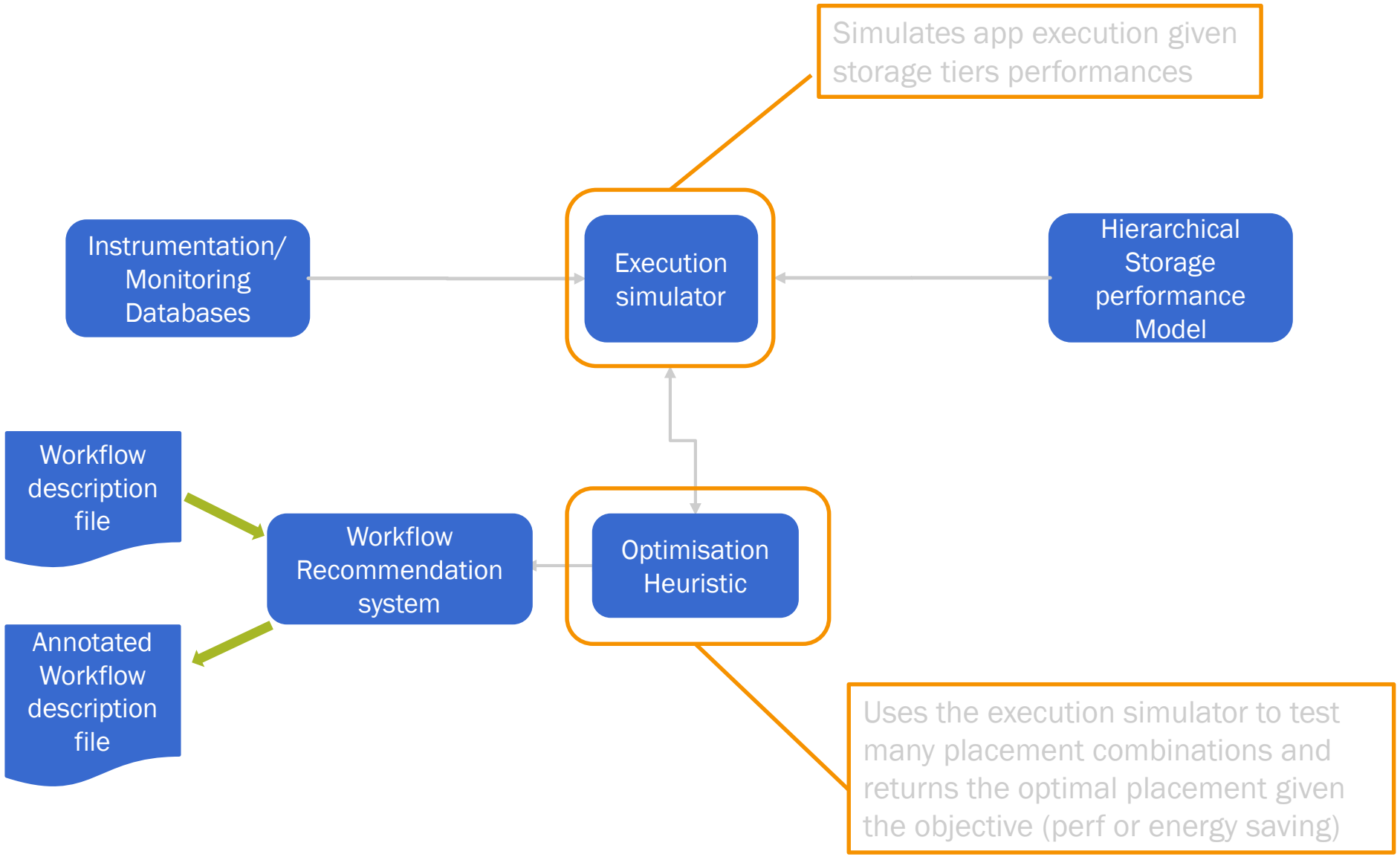
- Slurm integration (option **--ioinstrumentation=yes)**
  - Enabled upon user request



Recommendation system



IO Pattern analyzer : IO-SEA explorations



Questions?