The Smart Burst Buffer: An example of an ephemeral service and its connection to the long-term storage through the Hestia API

Philippe Couvée, Eviden

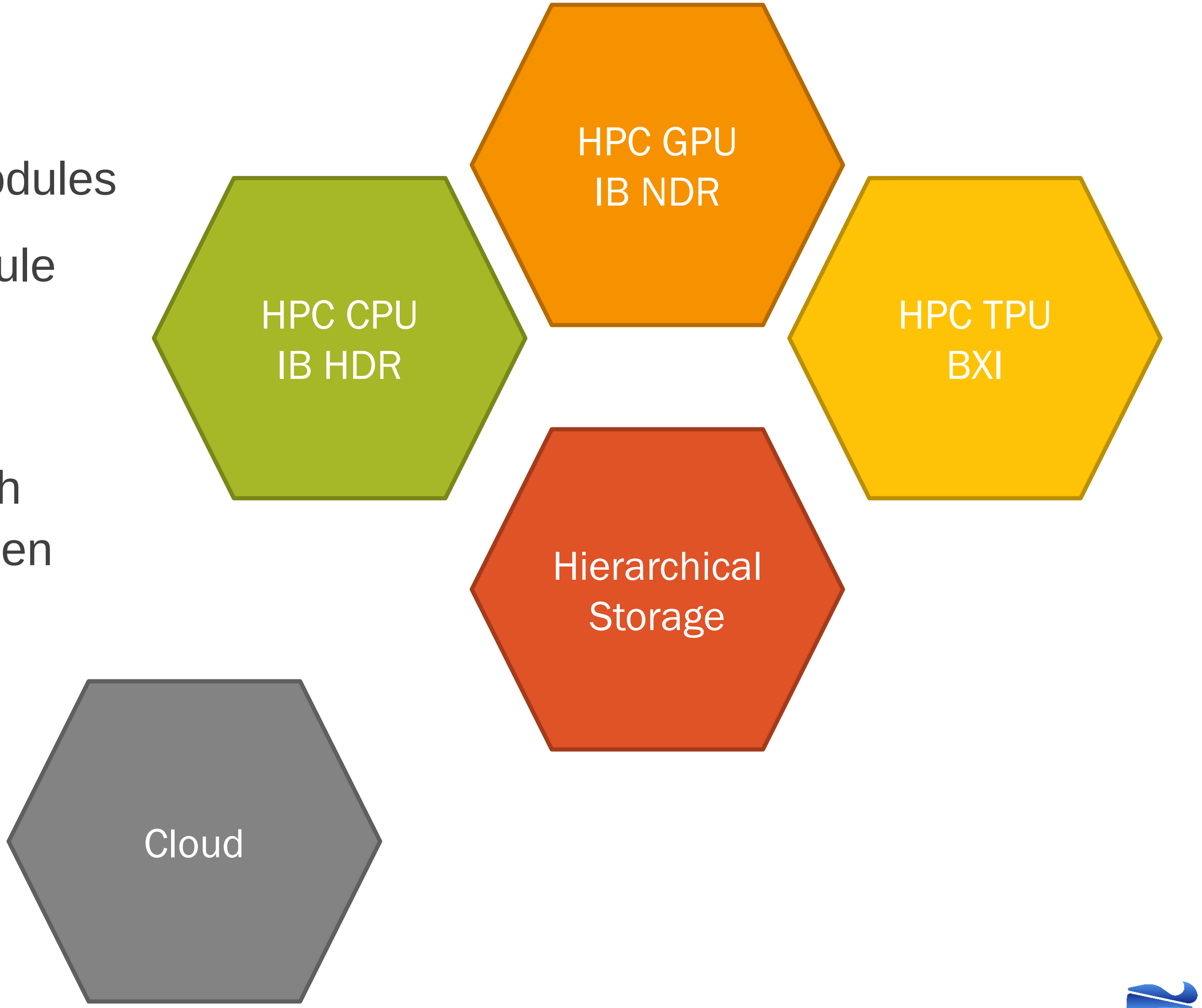
James Grogan, ICHEC

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 architecture

IO Challenges in the Modular SuperComputer Architecture

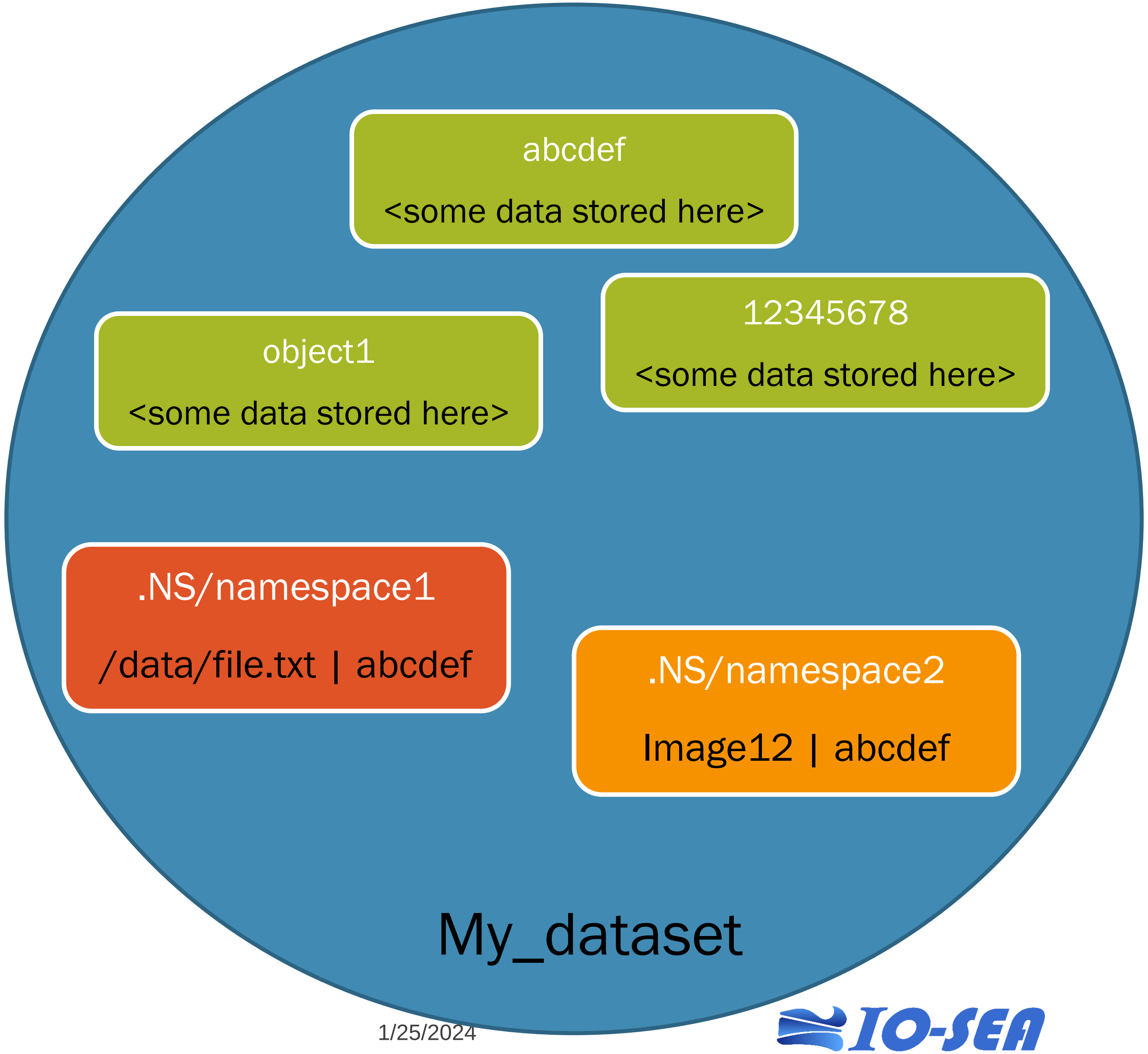
- Heterogeneous compute modules
- « Long Term » Storage module with multiple technologies
  - Hierarchical storage
- Low/medium bandwidth, high latencies connections between modules



Datasets & Namespaces

- **Datasets** are **data containers** hosting objects
  - No data organization, just collections of objects…
  - Could be seen as ***private*** file systems or object stores

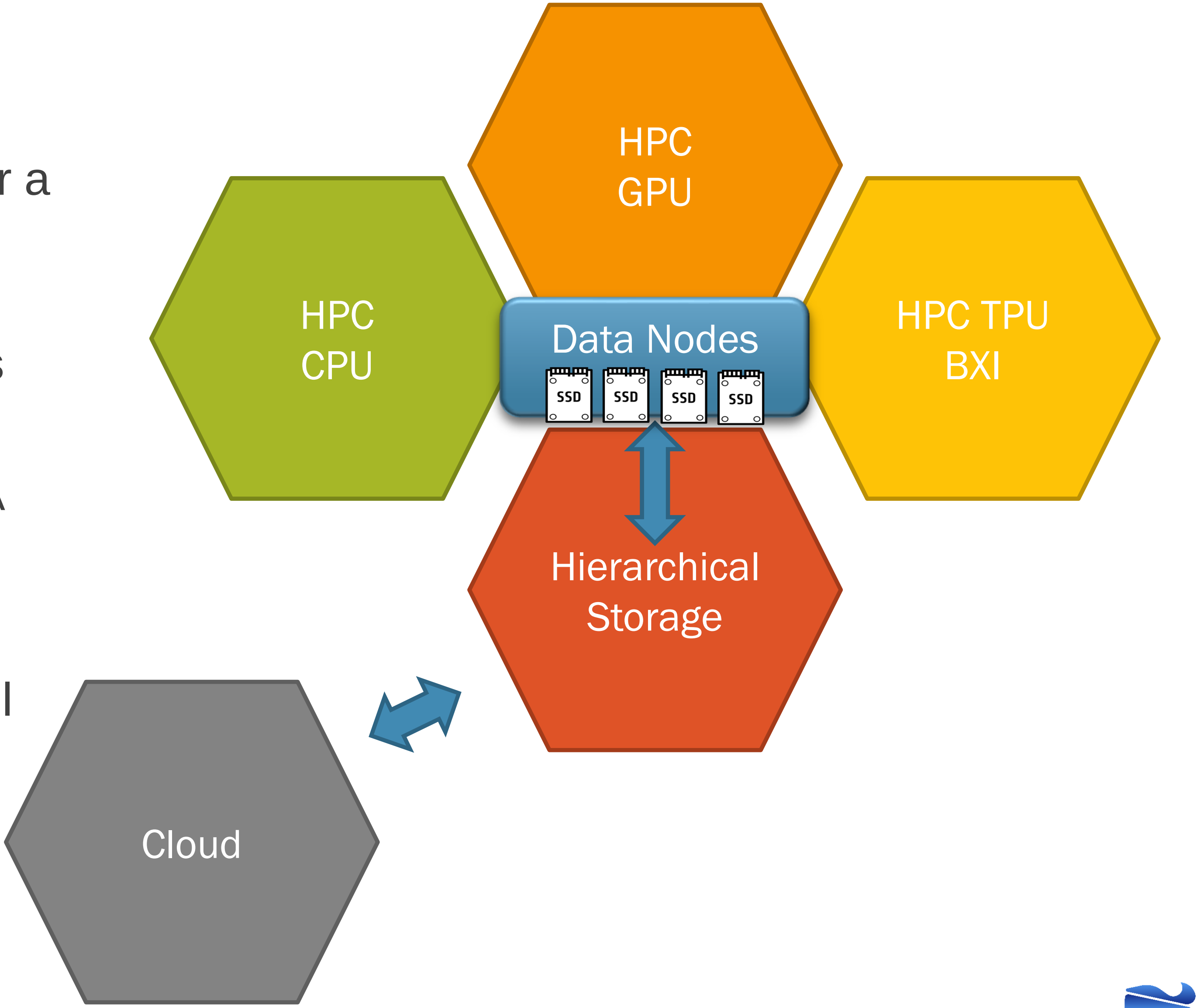
- Objects in Datasets are organized with **Namespaces**
  - Different Namespaces can be created in each Dataset
  - Namespaces can expose the same objects in the dataset through different protocols
    - POSIX, S3, …



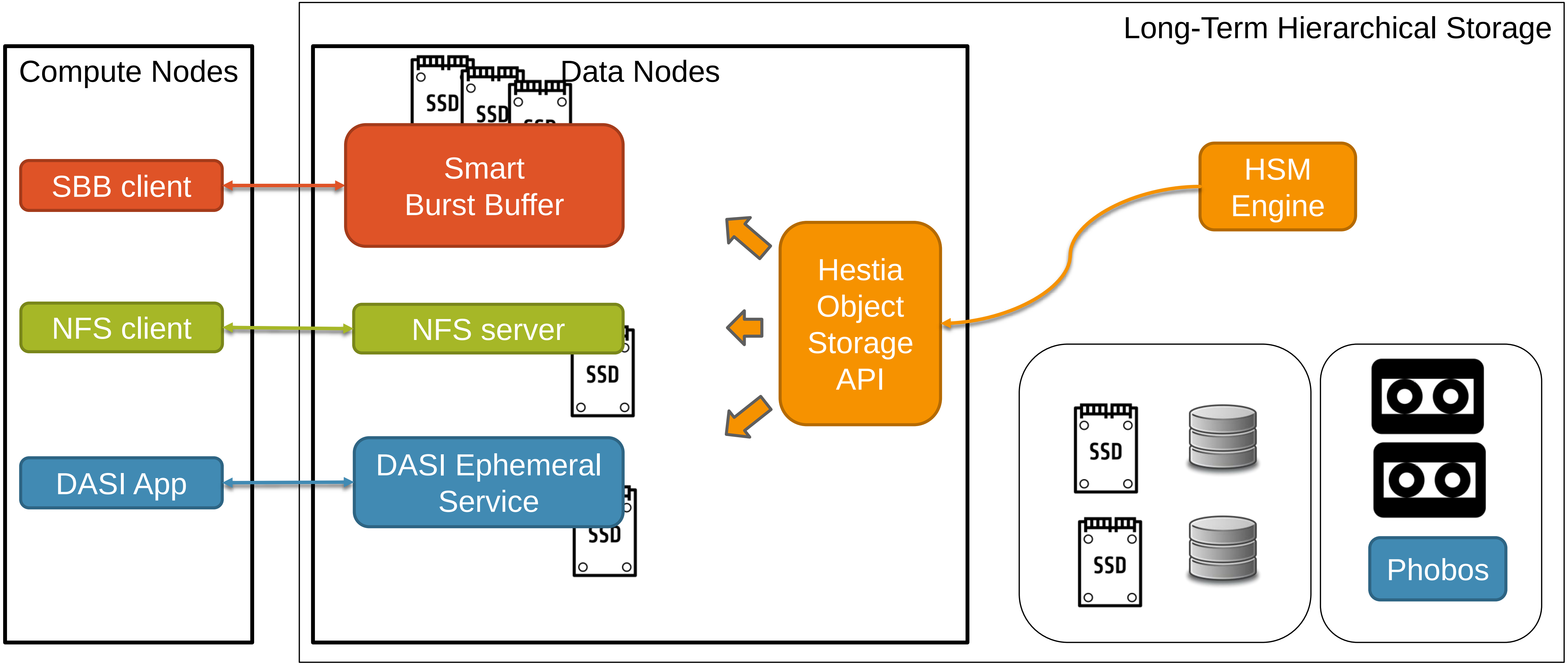
Data nodes for High Speed Access from HPC workflows

- Runs **Ephemeral Services** for a workflow **close to compute nodes**, enabling access to datasets through namespaces
  - Connected on HPC interconnects, enabling RDMA transfer speeds

- Multiple Data Nodes in parallel to reach target bandwidth
  - For some ephemeral services



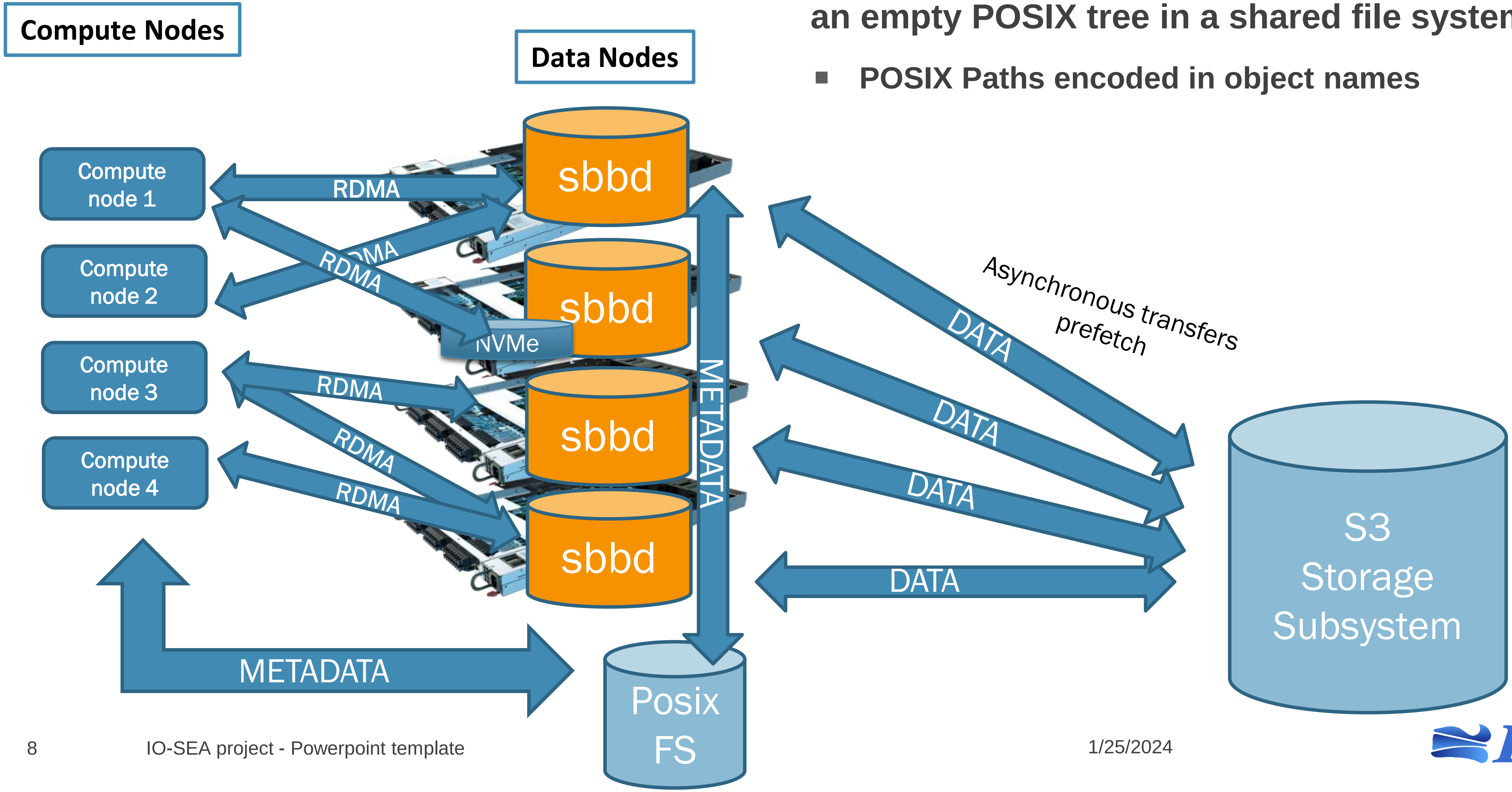
Ephemeral I/O Services, to expose a namespace to clients
Hestia API, as the main interface to Hierarchical Storage



The Smart Burst Buffer

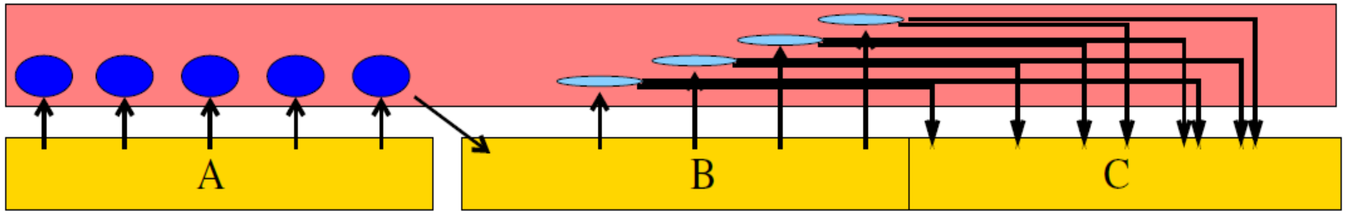
Smart Burst Buffer : Object Storage Backend support

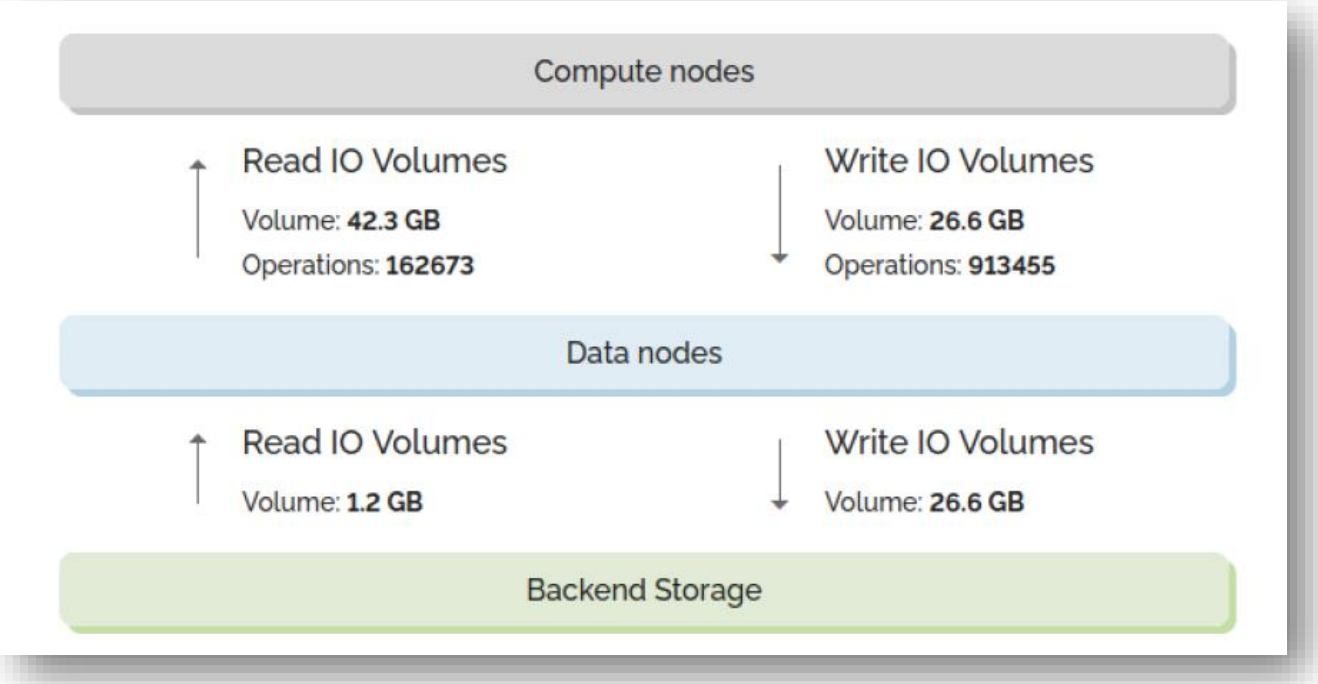
- **To expose an S3 Dataset through POSIX, SBB creates an empty POSIX tree in a shared file system**
  - **POSIX Paths encoded in object names**



Data reuse within Workflows: LQCD example

- 3 steps workflow
  - Steps B and C combined in a single sbatch





Compute nodes

Read IO Volumes

Volume: **42.3 GB**

Operations: **162673**

Write IO Volumes

Volume: **26.6 GB**

Operations: **913455**

Data nodes

Read IO Volumes

Volume: **1.2 GB**

Write IO Volumes

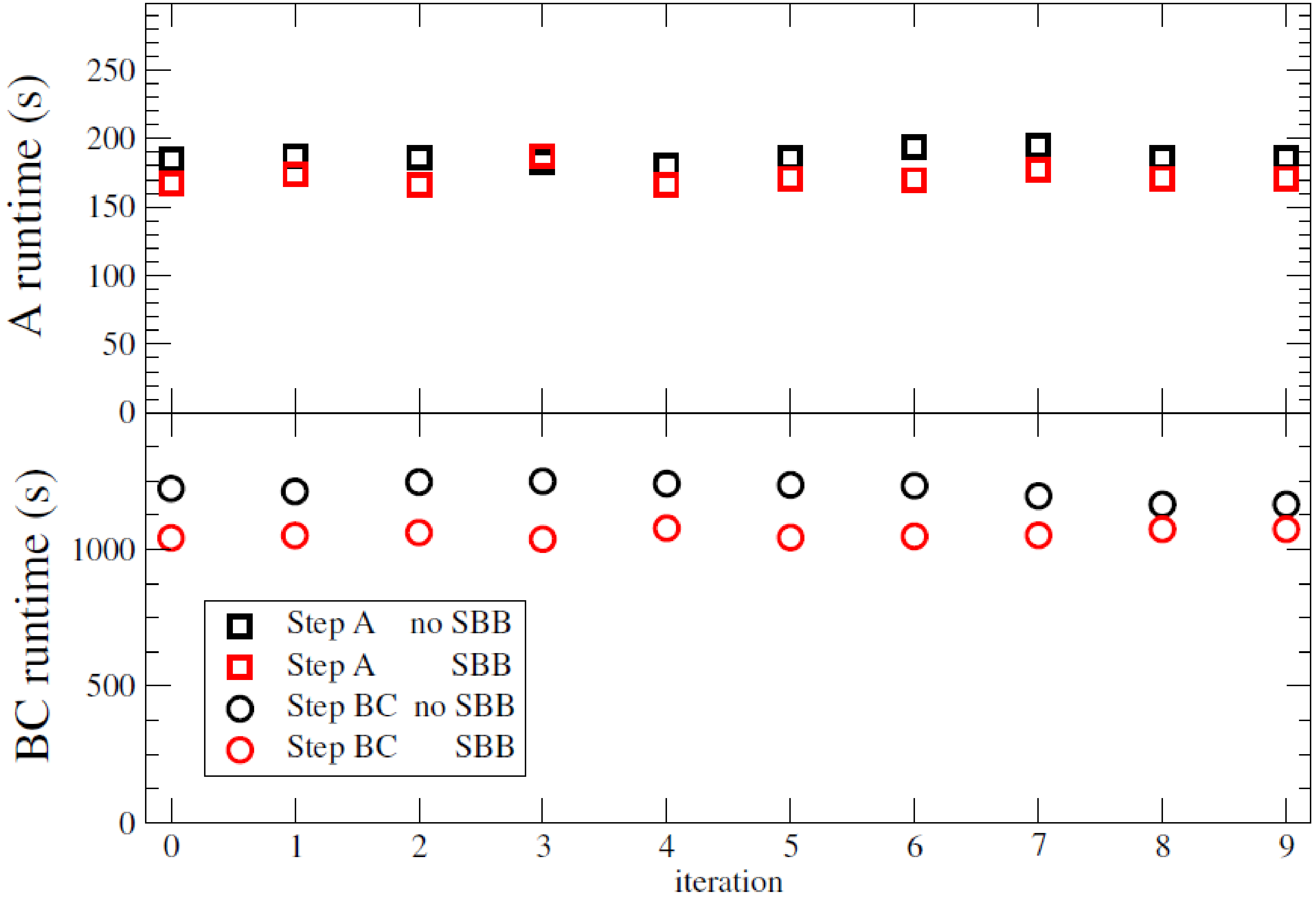
Volume: **26.6 GB**

Backend Storage

Performance improvment vs /p/project on DEEP

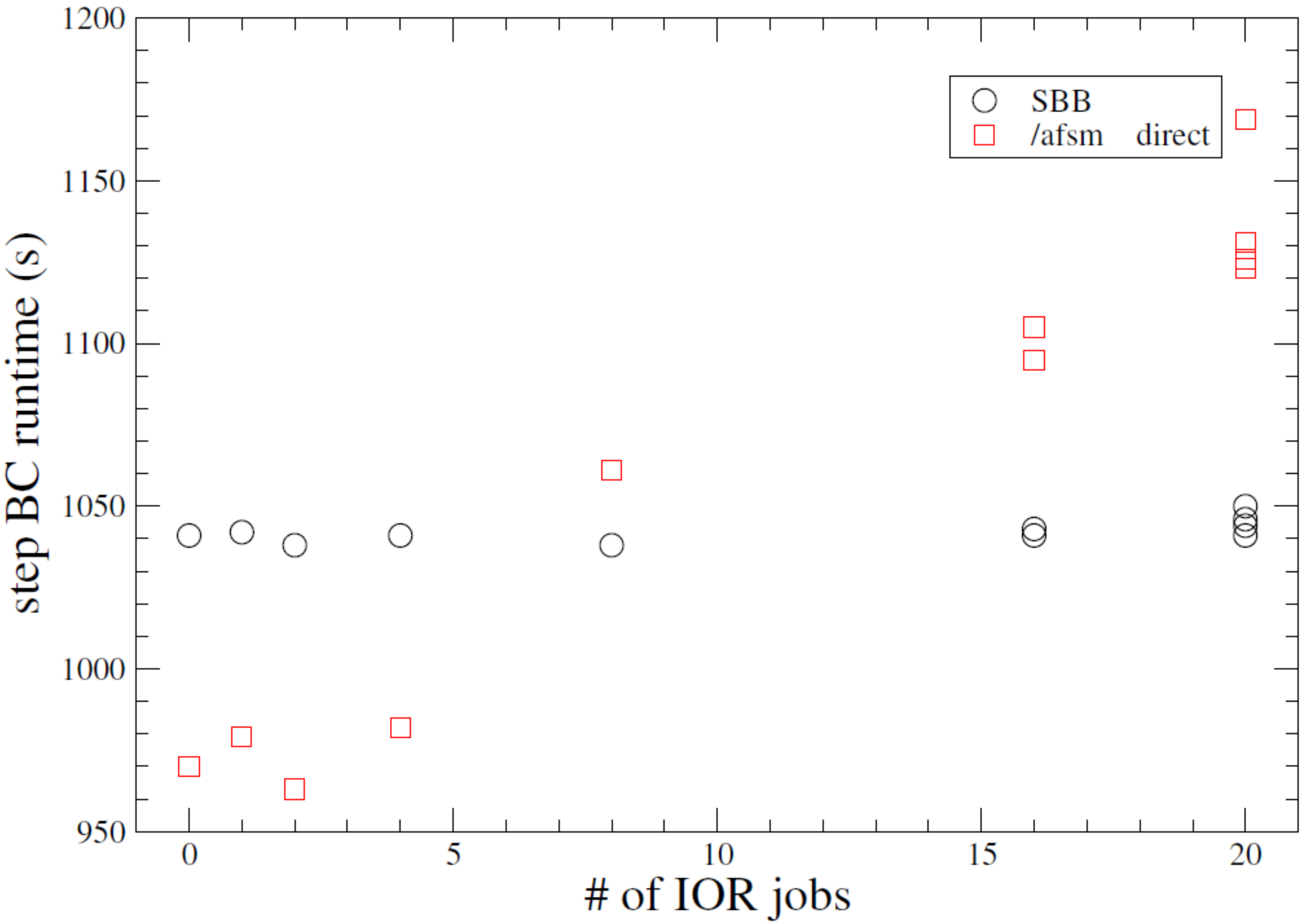
- /p/project on DEEP is an example of a « slow » access from a compute module
  - Big GPFS exported through NFS

- Even on sub-optimal SAGE2 datanodes, each workflow step is faster



Performance improvment vs /afsm on DEEP

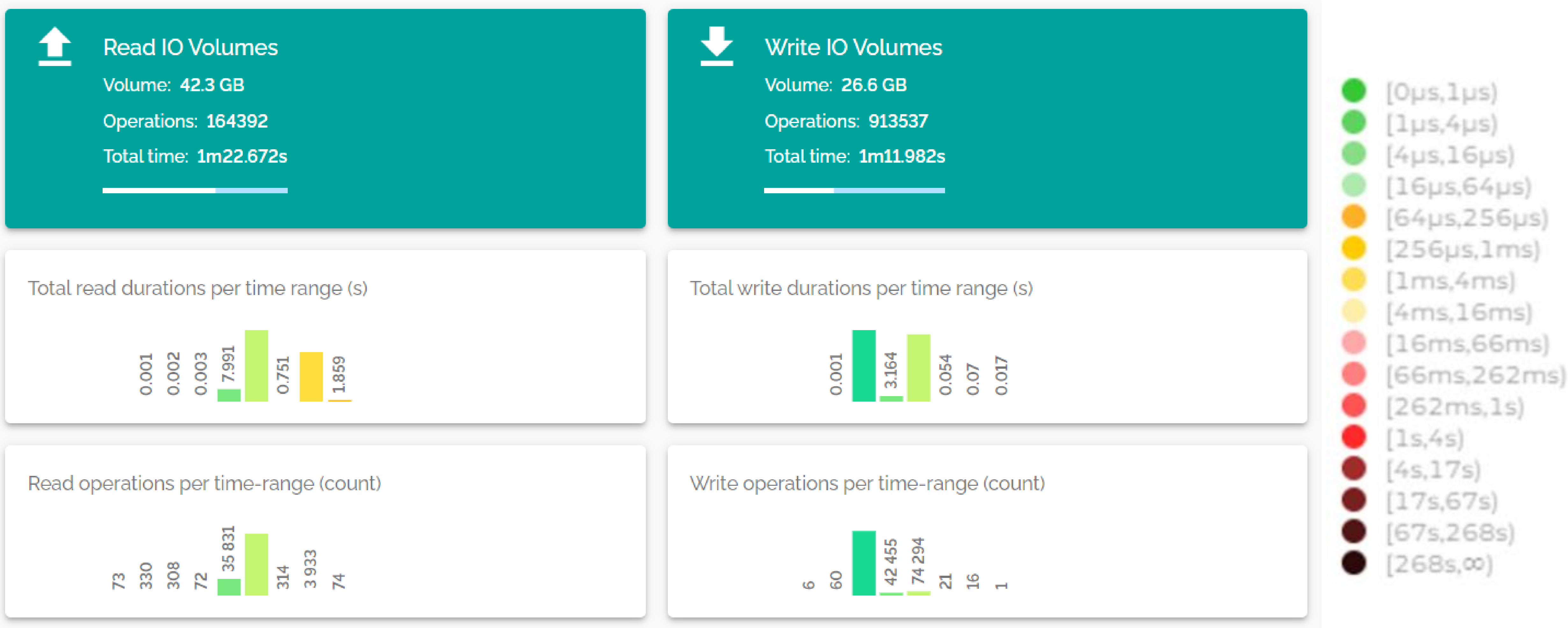
- /afsm on DEEP being local, all flash BEEGFS, it is FAST !
  - 30-40 GB/s sustained
- Ephemeral Services isolate the workflow from perturbations on the shared file systems caused by other application running in parallel



IO Instrumentation : LQCD Workflow

with and without Ephemeral Services

Direct Access to a shared file system



Read IO Volumes

Volume: **42.3 GB**

Operations: **164392**

Total time: **1m22.672s**

Write IO Volumes

Volume: **26.6 GB**

Operations: **913537**

Total time: **1m11.982s**

Total read durations per time range (s)

Total write durations per time range (s)

Read operations per time-range (count)

Write operations per time-range (count)

● [0µs,1µs)
● [1µs,4µs)
● [4µs,16µs)
● [16µs,64µs)
● [64µs,256µs)
● [256µs,1ms)
● [1ms,4ms)
● [4ms,16ms)
● [16ms,66ms)
● [66ms,262ms)
● [262ms,1s)
● [1s,4s)
● [4s,17s)
● [17s,67s)
● [67s,268s)
● [268s,∞)

End User Interfaces & APIs

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: SBB
    attributes:
      namespace: My_dataset.My_namespace
      targets: /mnt/USER/My_Workflow
      flavor: Medium
      datanodes: 4

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

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
```

Workflow Session Management

- **status** 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


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

Data Movers in the WDF

- 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

```
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: move
        elements:
          - "gauges/*.hdf5“
          - "input/*“
```

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

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)

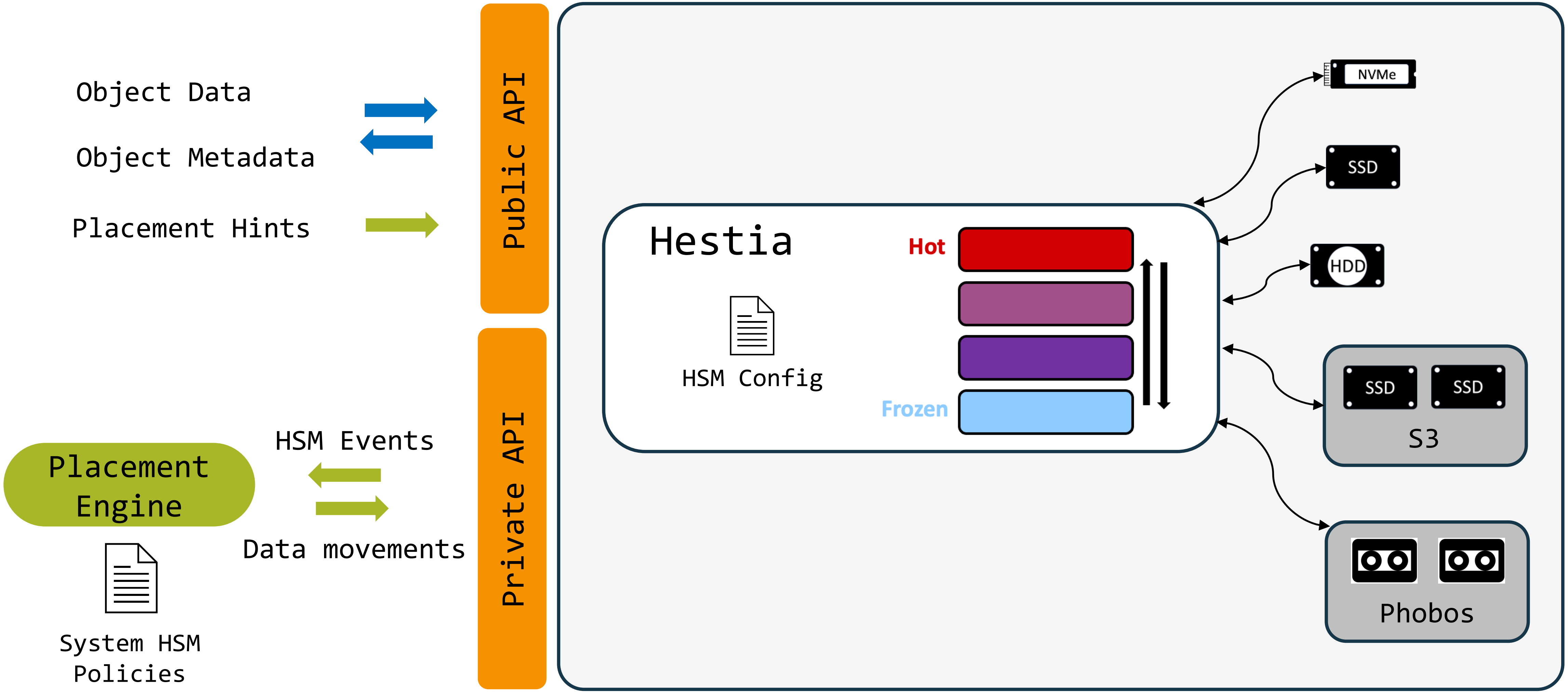
–**estimated_atime**: Estimated time the object will be used (in seconds)

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

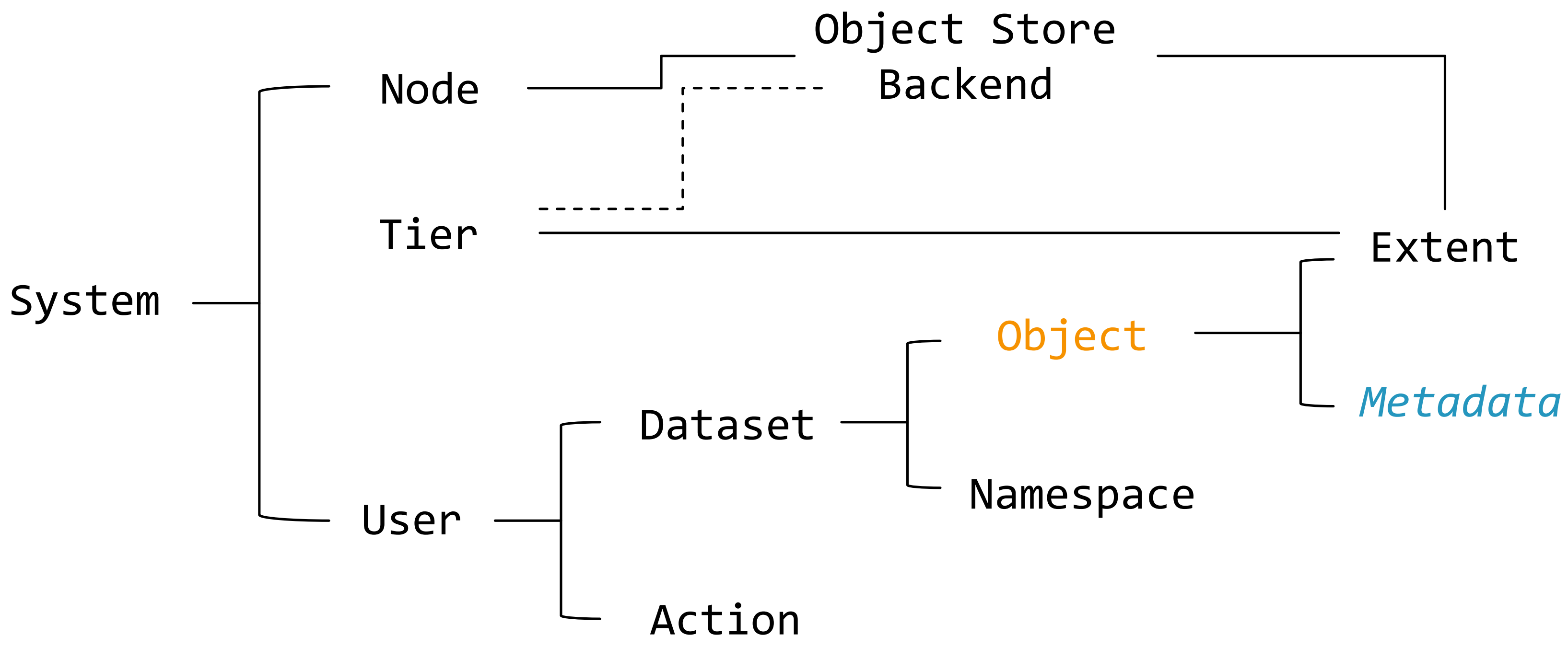
–**predefined_policy**: Name of a pre-configured policy
    e.g. predefined_policy=”temporary_data”

Long Term Storage and the Hestia API

Interaction with Storage Media



Hestia Data Model



Operations

CREATE

READ

UPDATE

DELETE

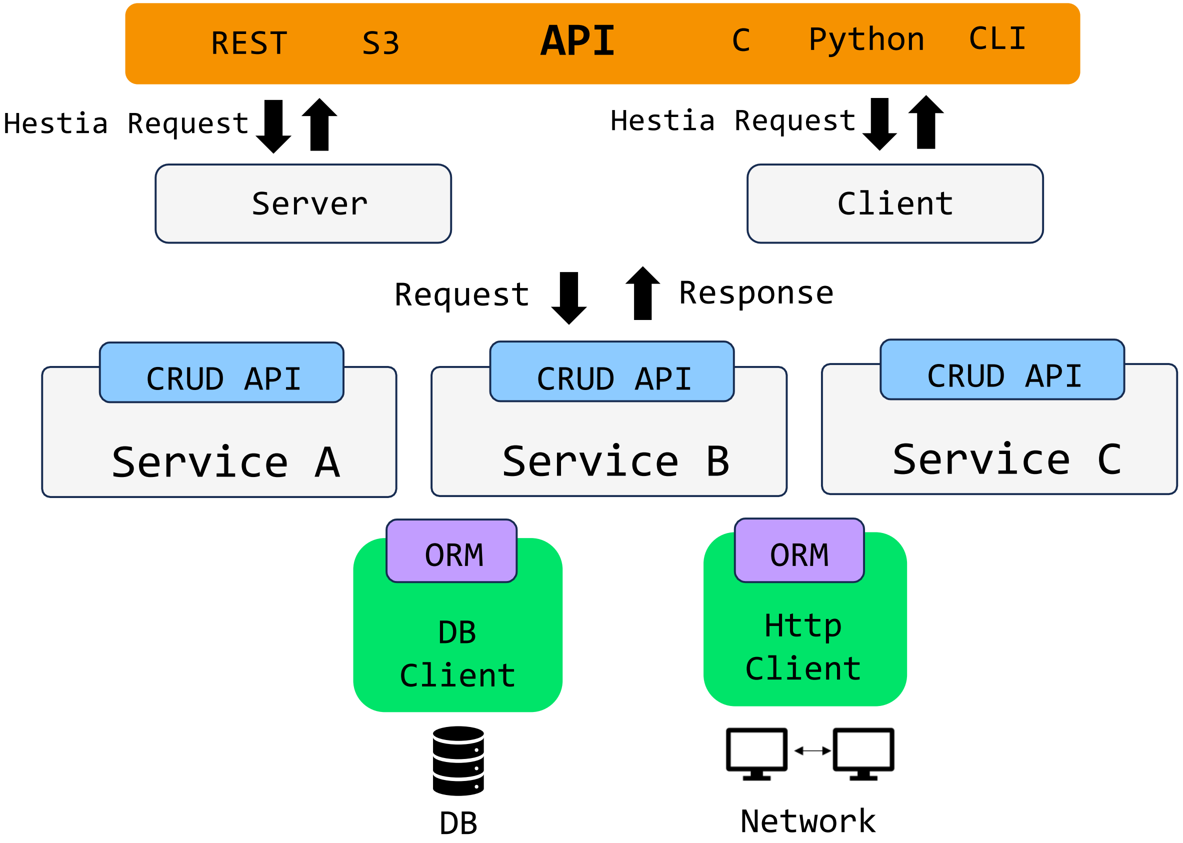
Relations

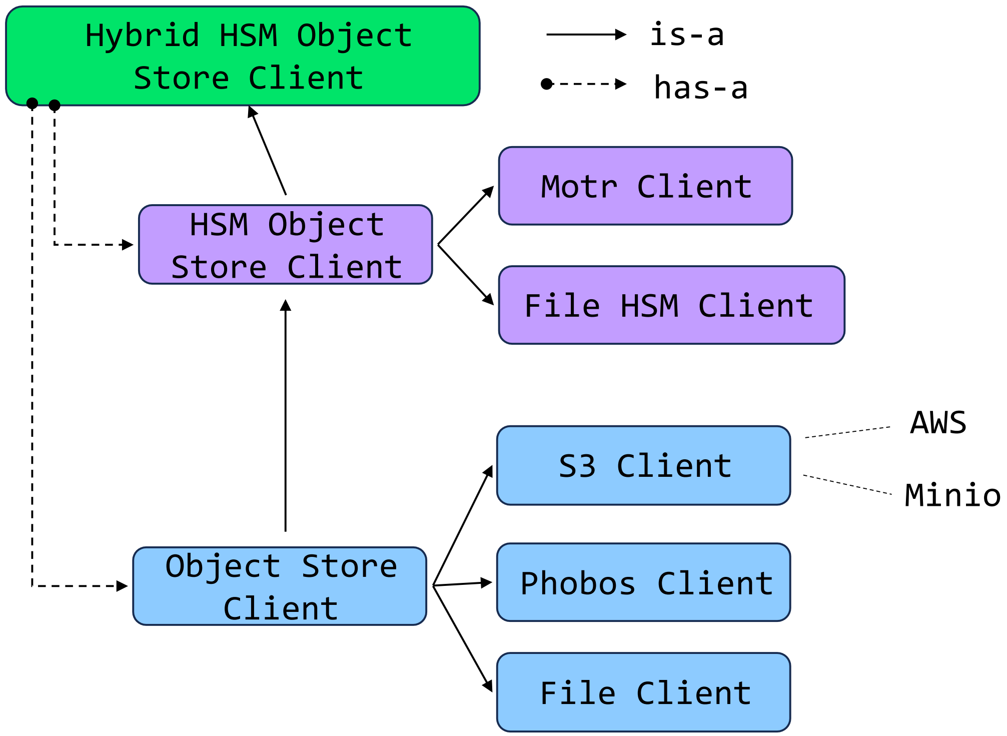
One To Many

One To One

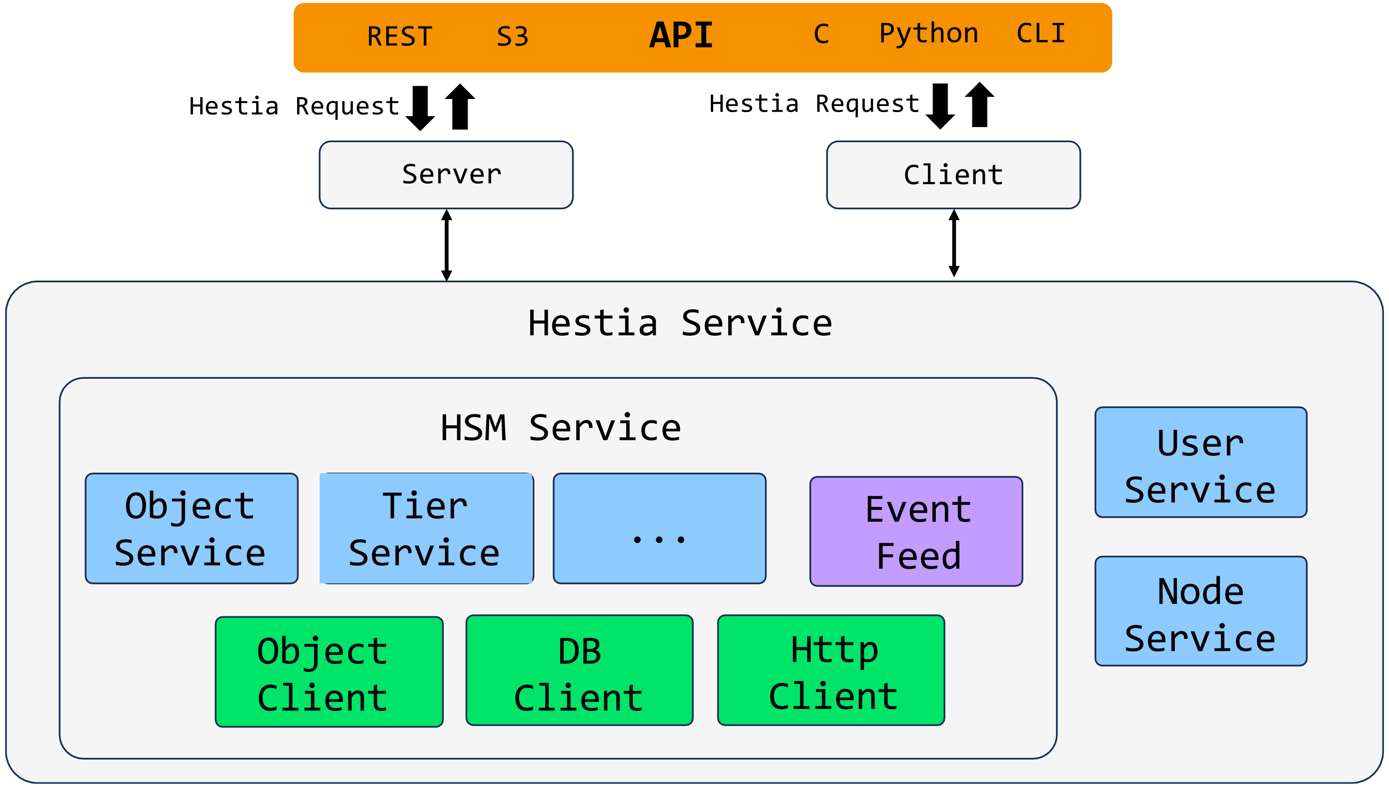
Many To Many

Hestia Architecture





Hestia System



REST S3 **API** C Python CLI

Hestia Request

Hestia Request

Server

Client

Hestia Service

HSM Service

Object Service

Tier Service

...

Event Feed

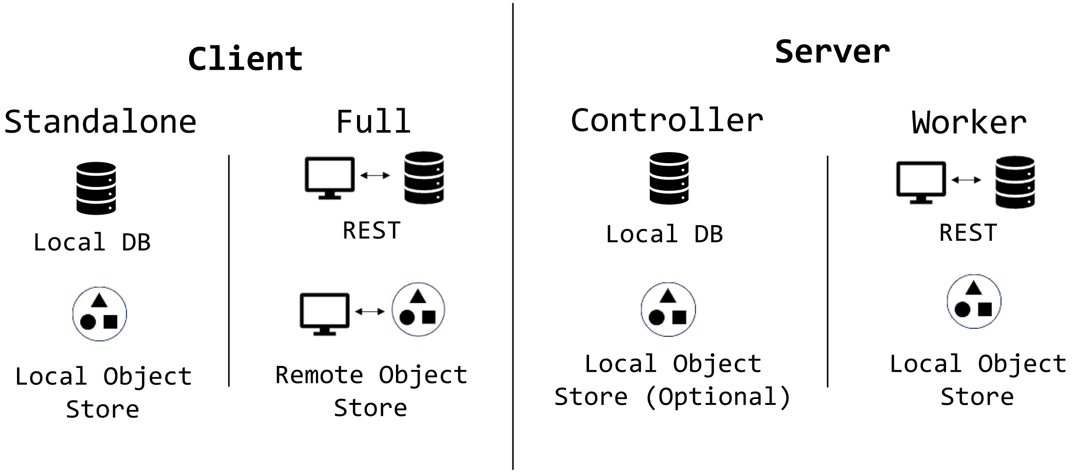
Object Client

DB Client

Http Client

User Service

Node Service



Client

Standalone

Local DB

Local Object Store

Full

REST

Remote Object Store

Server

Controller

Local DB

Local Object Store (Optional)

Worker

REST

Local Object Store

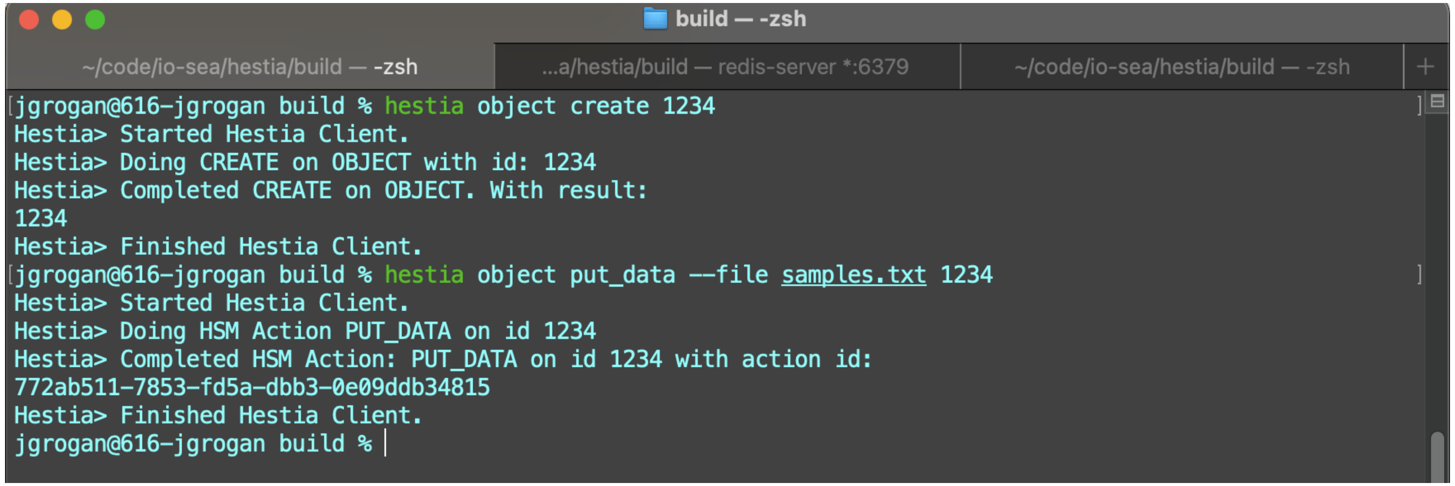
Trying it out…

- C++ with CMake – low build and runtime dependencies

- Open Source – MIT license

- CI/CD with RPM builds – can build and run on Mac and Linux

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


build — -zsh
~/code/io-sea/hestia/build — -zsh
...a/hestia/build — redis-server *:6379
~/code/io-sea/hestia/build — -zsh

```
jgrogan@616-jgrogan build % hestia object create 1234
Hestia> Started Hestia Client.
Hestia> Doing CREATE on OBJECT with id: 1234
Hestia> Completed CREATE on OBJECT. With result:
1234
Hestia> Finished Hestia Client.
jgrogan@616-jgrogan build % hestia object put_data --file samples.txt 1234
Hestia> Started Hestia Client.
Hestia> Doing HSM Action PUT_DATA on id 1234
Hestia> Completed HSM Action: PUT_DATA on id 1234 with action id:
772ab511-7853-fd5a-dbb3-0e09ddb34815
Hestia> Finished Hestia Client.
jgrogan@616-jgrogan build %
```