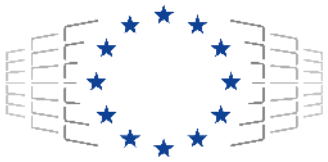
IO-SEA Hackathon

Hands-On Training with Data Access & Storage Interface (DASI)



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.

Outline

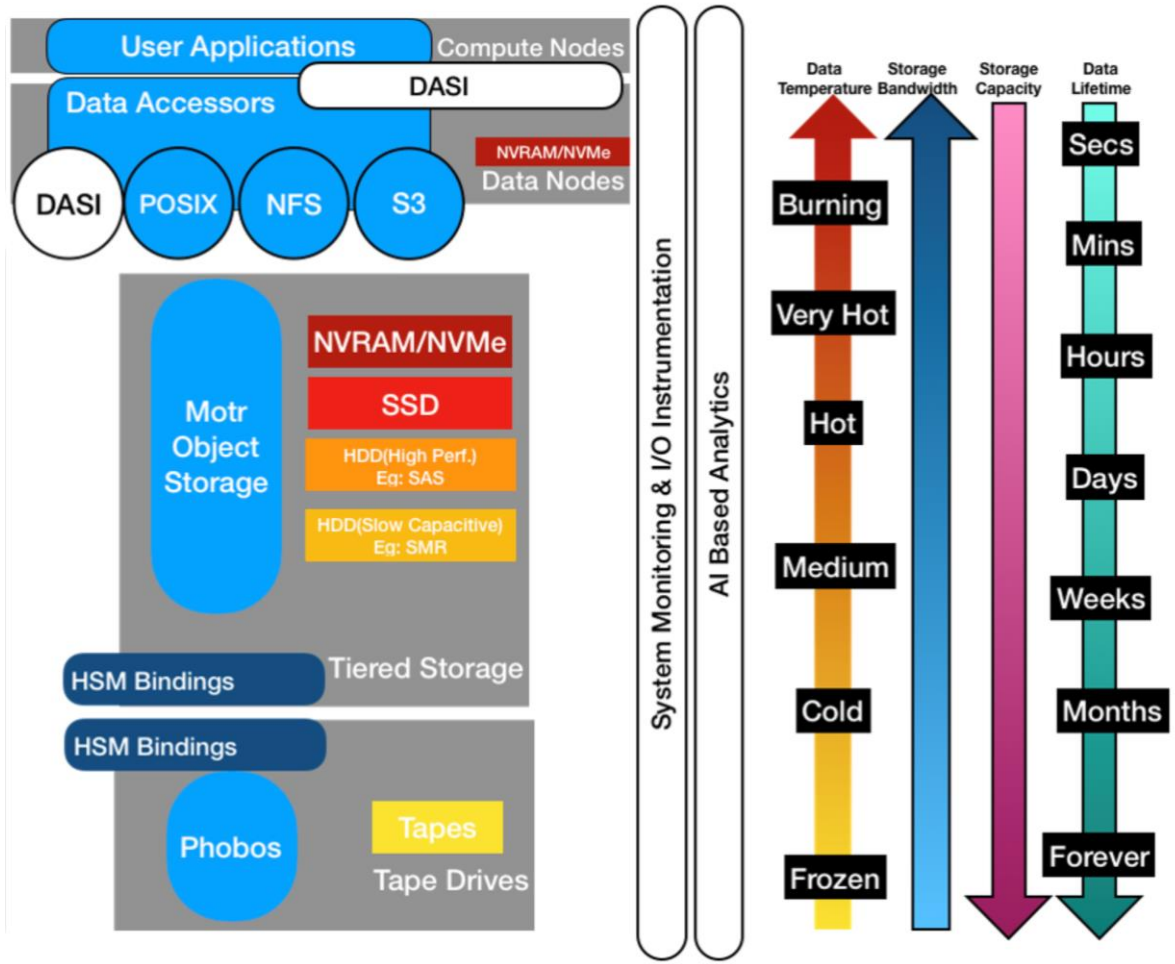
- IO-SEA Project
- DASI Concept and Design
- API

IO-SEA Project

- **Aim:** to implement solutions for scaling I/O intensive applications to exascale HPC systems

- **Components:**
  - Object stores
  - Hierarchical storage management (HSM)
  - Ephemeral services

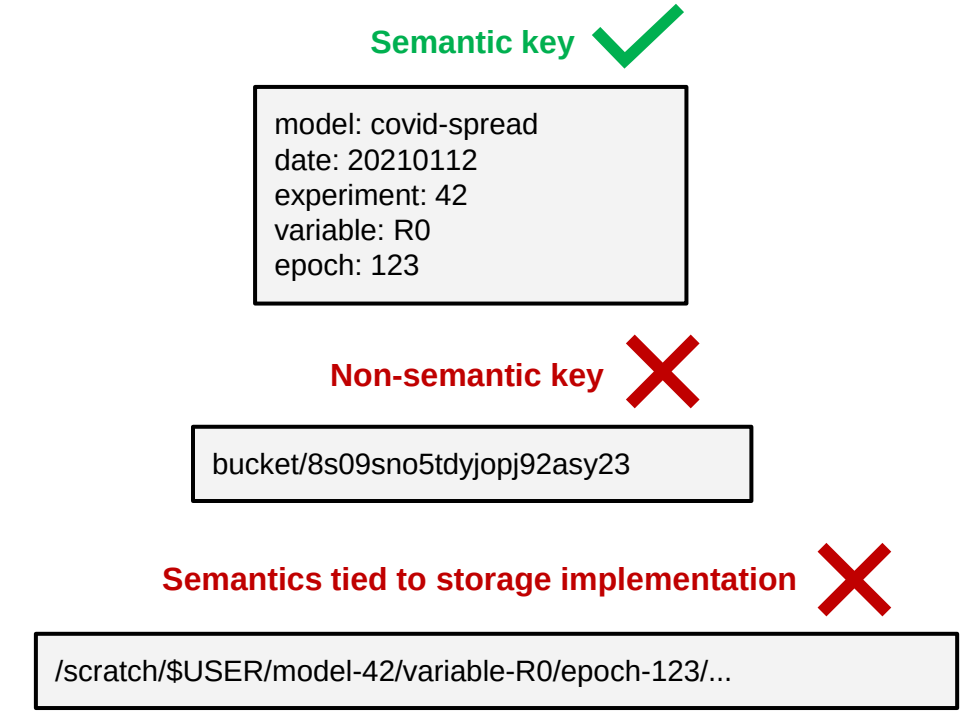
- **DASI** sits between the user applications and HSM as an application interface for abstracting the complex storage layer from users



DASI Concept and Design

DASI Concept

- **Data Access and Storage Interface (DASI)** provides a scientifically meaningful way to manage data, and abstracts the underlying storage technologies
- **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



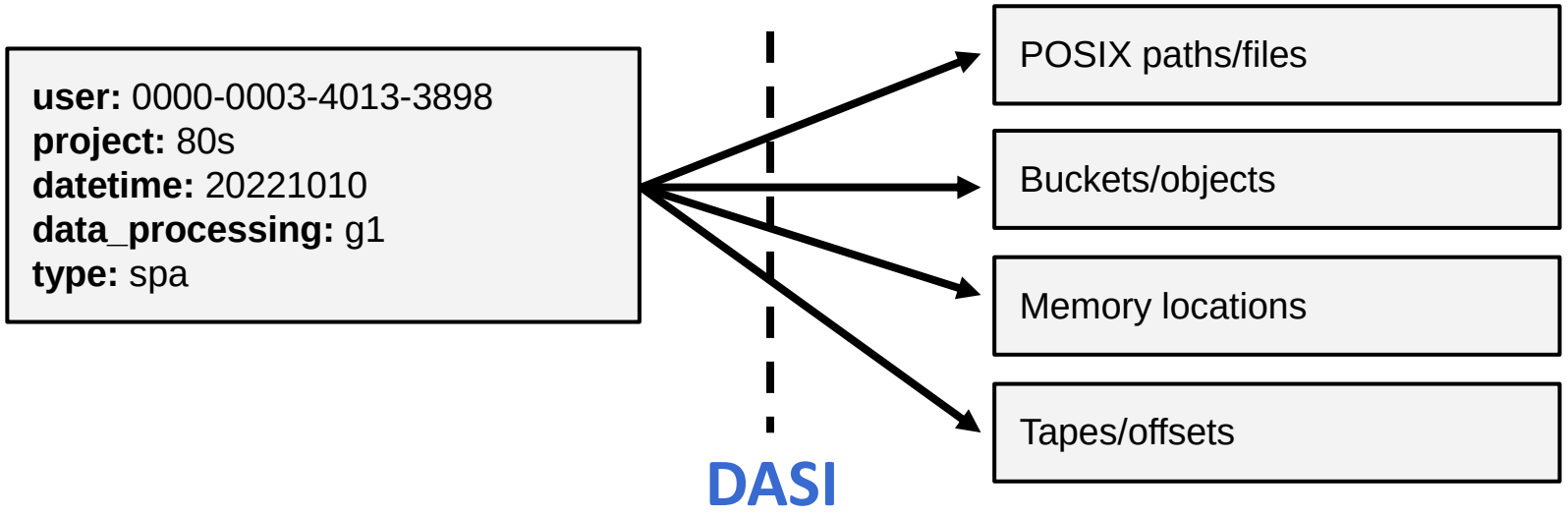
IO-SEA

Semantic Data Access as an Abstraction

- You have probably already done basic semantic data access with files and folders…

```
/../0000-0003-4013-3898/80s/20221010/g1/spa/...
```

… but a better implementation decouples the scientific identification from the storage resource



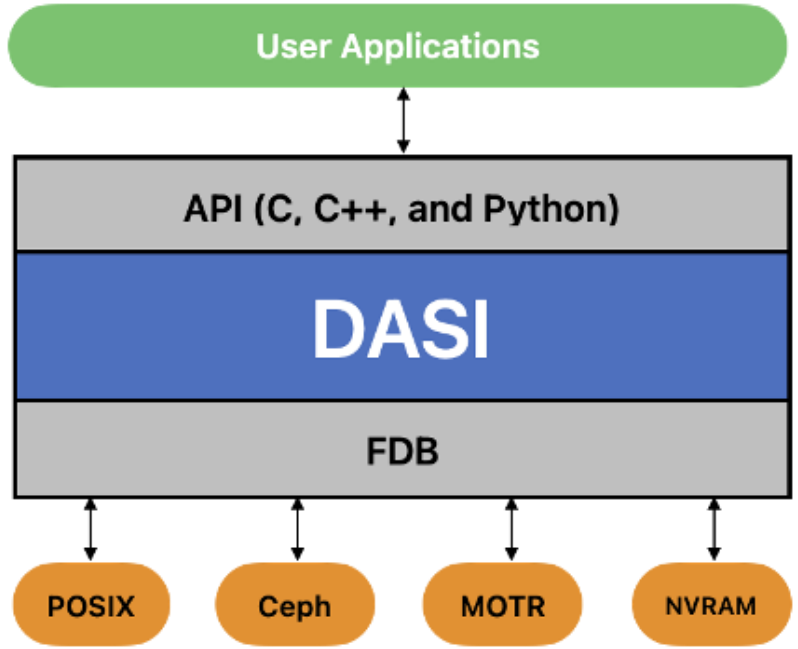
… and the applications don't need to care how the objects are stored

- Allows optimal usage of storage backend without leaking details to user-space

IO-SEA

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.)
- **DASI Index Abstraction**
  - Mapping between keys and object locations in datastore
- **DASI Datastore Abstraction**
  - Object-store-like API for raw storage objects



IO-SEA

DASI API and Usage

Implementation of DASI

- **DASI APIs Available**
  - C
  - C++
  - Python
  - Command Line Interface (CLI)
- **Configured through Collection Schemas**
  - Specify the semantic keys
  - Hierarchy of metadata keys
  - Each collection schema is a tree with three levels
- Data is addressed by specifying all the keys along a path from the root to a leaf

Example Schema

```
[ User, Project,
  [ Date, Time
    [Processing],
  ],
]
```

Example Address

```
User: jw
Project: IOSEA
Date: 20231101
Time: 12
Processing: training
```

IO-SEA

Building a Schema

- **Identify Data Collection**
  - Which data do you want to store together?
  - What are your data objects?
- **Define Metadata keys**
  - How do you uniquely identify an object?
  - If needed, what are the different sets of keywords need?
- **Determine Hierarchy**
  - Choose relevant order for the keywords
  - Schema supports branching

```
Example: Cryo Electron
Microscropy
```

```
Keywords: User, Project,
Date, Time, Processing, Type
```

```
Hierarchy:
1. User, Project
2. Date, Time
3. Processing, Type
```

IO-SEA

Example Schema

```
# Rule 1
[ User, Project,       # Level 1: specifies top level directory
  [ Date, Time,        # Level 2: specifies filename
    [Processing],      # Level 3: indexes entries in file
  ],
]

# Rule 2
[ Institute, Project,
  [ Date, Location?,   # “?” used for optional key
    [Type],
  ]
]
```

IO-SEA

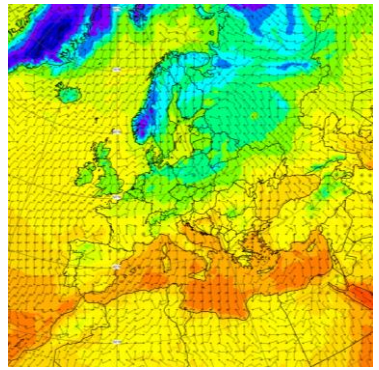
DASI Python API

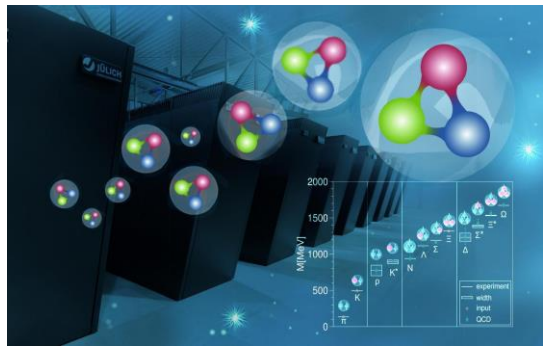
```
dasi = Dasi(“config.yaml”) # location of schema file defined in config.yaml

# Save some data (bytestream) using associated metadata (key)
key = {“User”: “jw”, “Project”: “IOSEA”, “Date”: “20231101”, “Location”: “Reading”}
dasi.archive(key, data)

# Retrieve saved data
data = dasi.retrieve({User:{jw}, Project:{IOSEA}, Date:{20231101}, Location:{Reading}) ➔ [bytestream]

# Use list to query data stored in Dasi
dasi.list(({User:{jw}, Date: {20231101}}) ➔ [metadata]
```

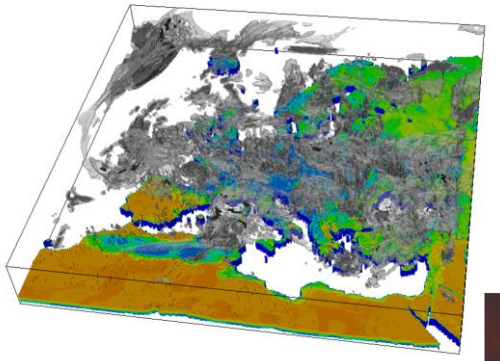
IO-SEA

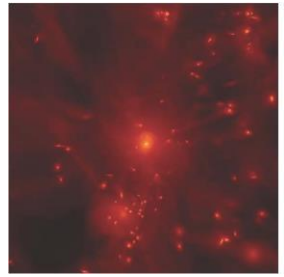
IO-SEA Use Cases

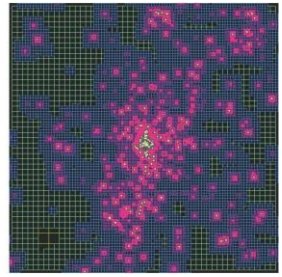
- **ECMWF** uses DASI for Integrated Forecast System weather forecasting workflow
- **Lattice Quantum Chromodynamics** uses DASI for markov-chain scientific checkpoint files
- **Terrestrial Systems Multiple-Physics** (TSMP) uses DASI for output from TSMP model components
- **RAMSES** code for modelling astrophysical phenomena uses DASI for post-processing
- **CEITEC** electron microscopy facility DASI for raw imagery and processed images

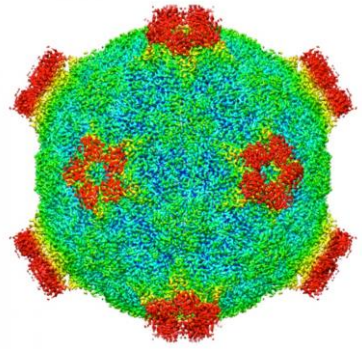












IO-SEA