DASI: a user centric data access and storage interface

Jenny Wong, Metin Cakircali, Olivier Iffrig, Simon Smart, James Hawkes and Tiago Quintino

Forecasts & Services Department, ECMWF, Reading

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.

About ECMWF

Established in 1975, Intergovernmental Organisation

- 23 Member States | 12 Cooperating States
- 450+ staff

24/7 operational service

- Operational NWP – 4x HRES+ENS forecasts / day
- Supporting NWS (coupled models) and businesses

Research institution

- Experiments to continuously improve our models
- Reforecasts and Climate Reanalysis

Operate 2 EU Copernicus Services

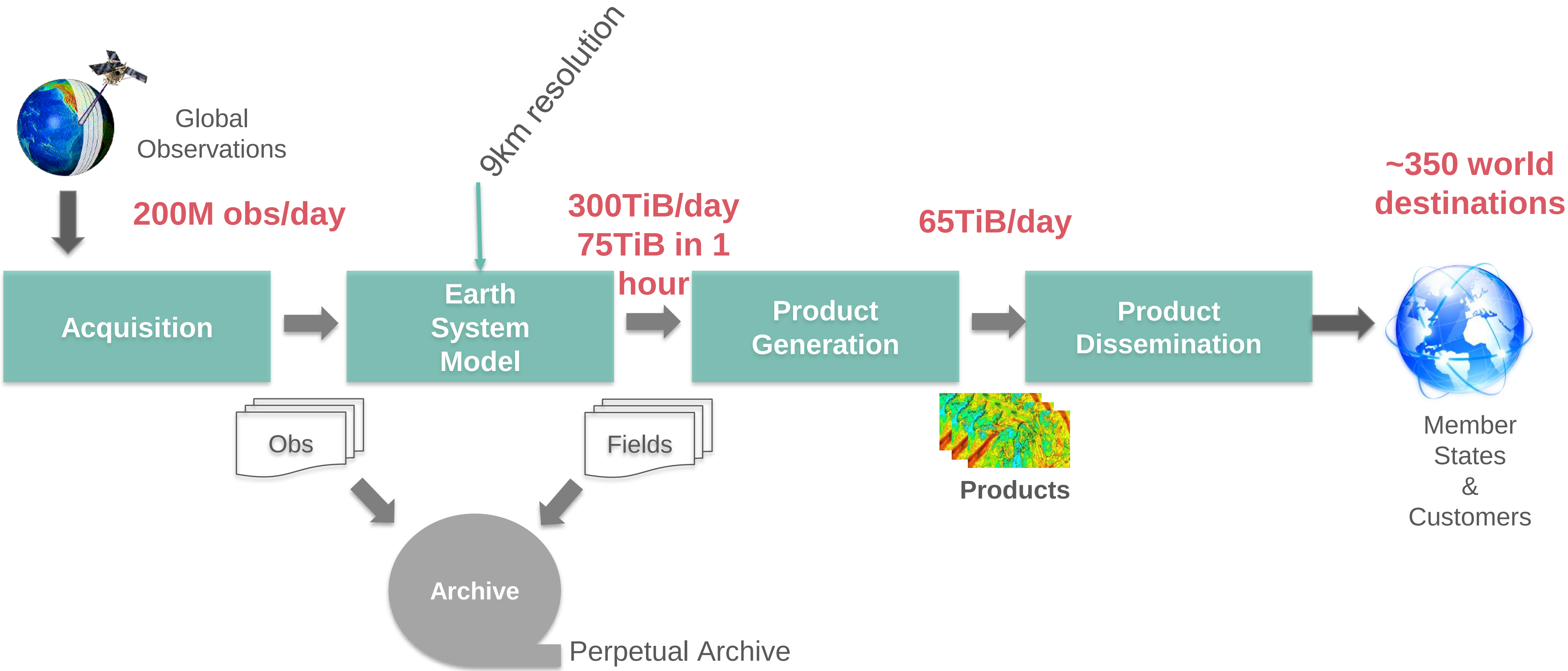
- Climate Change Service (C3S)
- Atmosphere Monitoring Service (CAMS)
- Support Copernicus Emergency Management Service CEMS

Destination Earth

- Operates two Digital Twins
- Operates the DestinE Digital Twin Engine (DTE)

Reading, GB
Reading
Bonn
Bonn, DE
Bologna
Bologna, IT

ECMWF’s Production Workflow



Global Observations

200M obs/day

9km resolution

300TiB/day
75TiB in 1 hour

65TiB/day

~350 world destinations

Acquisition → Earth System Model → Product Generation → Product Dissemination → Member States & Customers

Obs

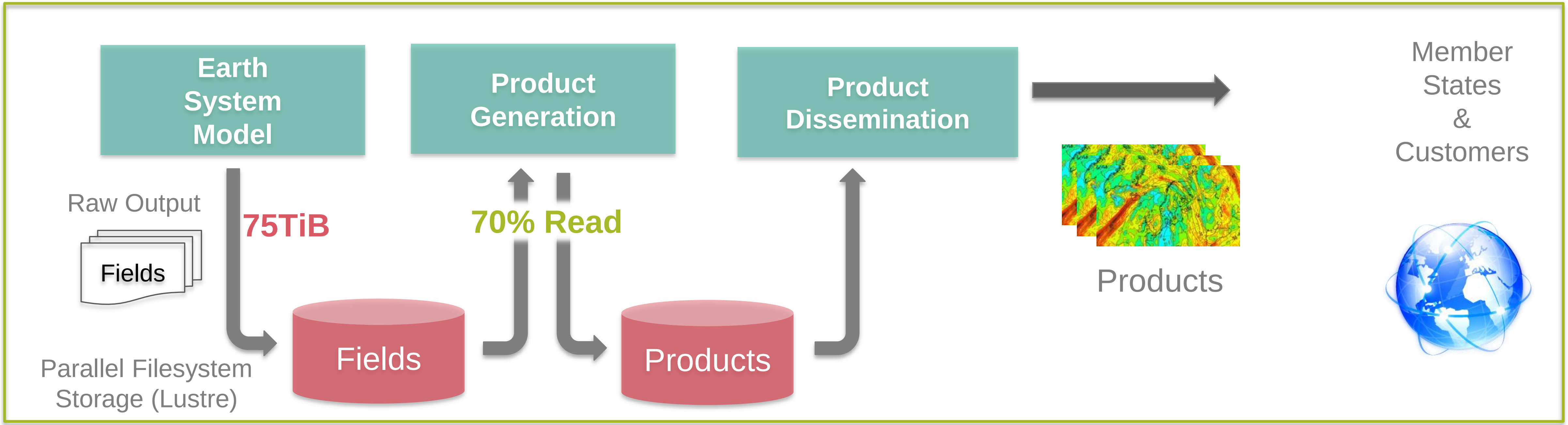
Fields

Products

Archive

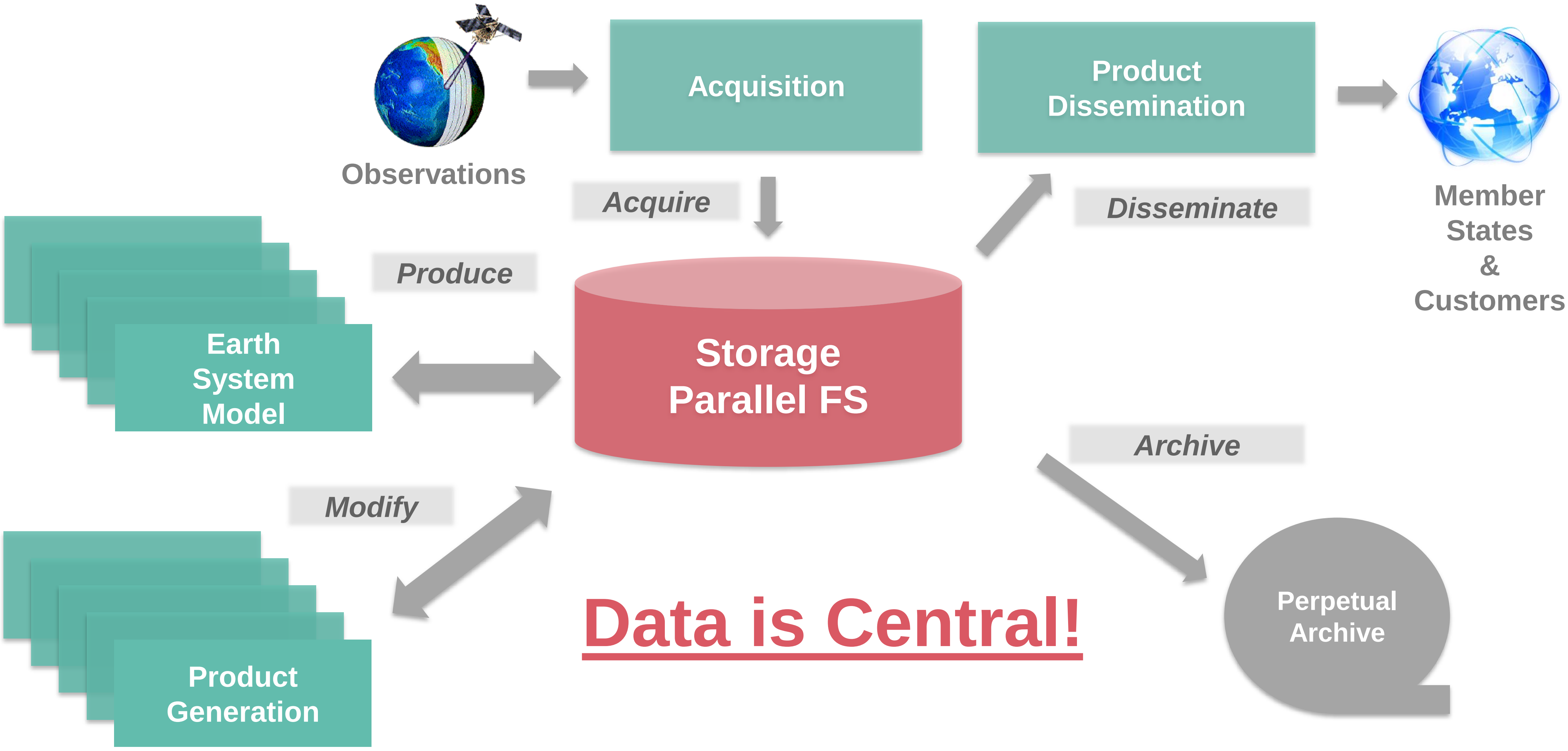
Perpetual Archive

ECMWF's Production Workflow



Time critical path = 1 hour window

ECMWF’s Production Workflow



Observations

Acquisition

Product Dissemination

Member States & Customers

Acquire

Disseminate

Produce

Earth System Model

Storage Parallel FS

Archive

Modify

Product Generation

Perpetual Archive

Data is Central!

IO-SEA

Semantic Data Management

- Data is indexed by its scientific metadata, according to a hierarchical schema
- The key used to index data carries scientific meaning
  - Not just a UUID
  - Not just storing metadata with data
  - The metadata is **used to index and uniquely identify the data**
- ECMWF archive from 1984-2023 (>600PiB) is all addressed with the same data language

Non-semantic key

~~8s09sno5tdyjopj92asy23~~

Semantic key

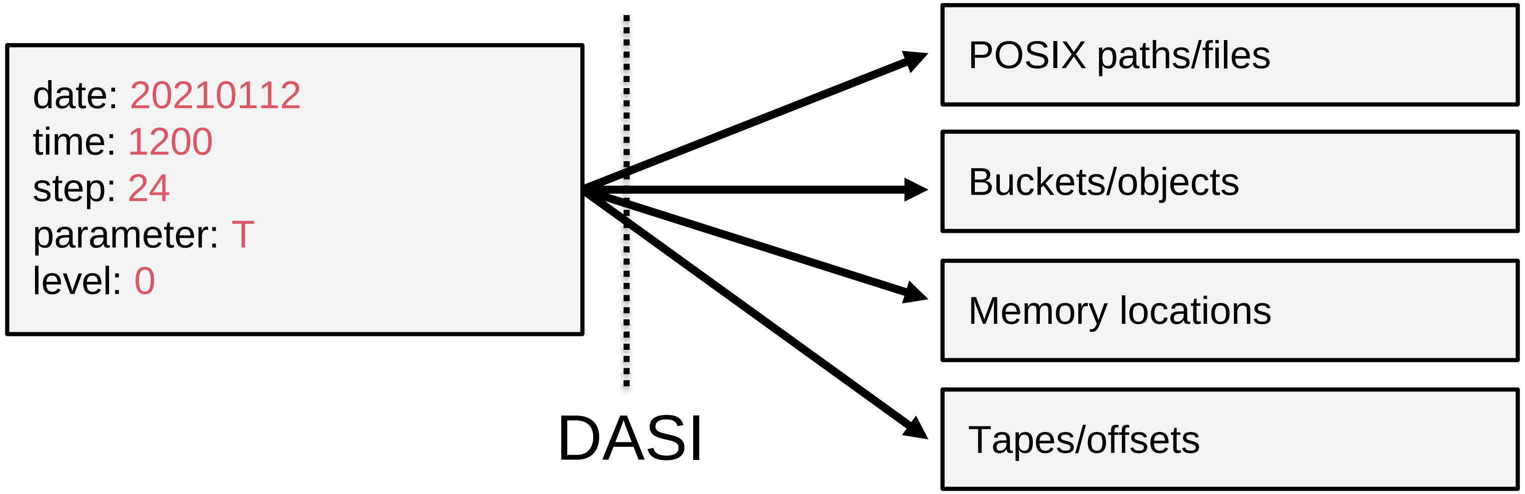
date: 20210112
time: 1200
step: 24
parameter: T
level: 0

Semantic Data Management

- The most basic semantic data access can be done with files…

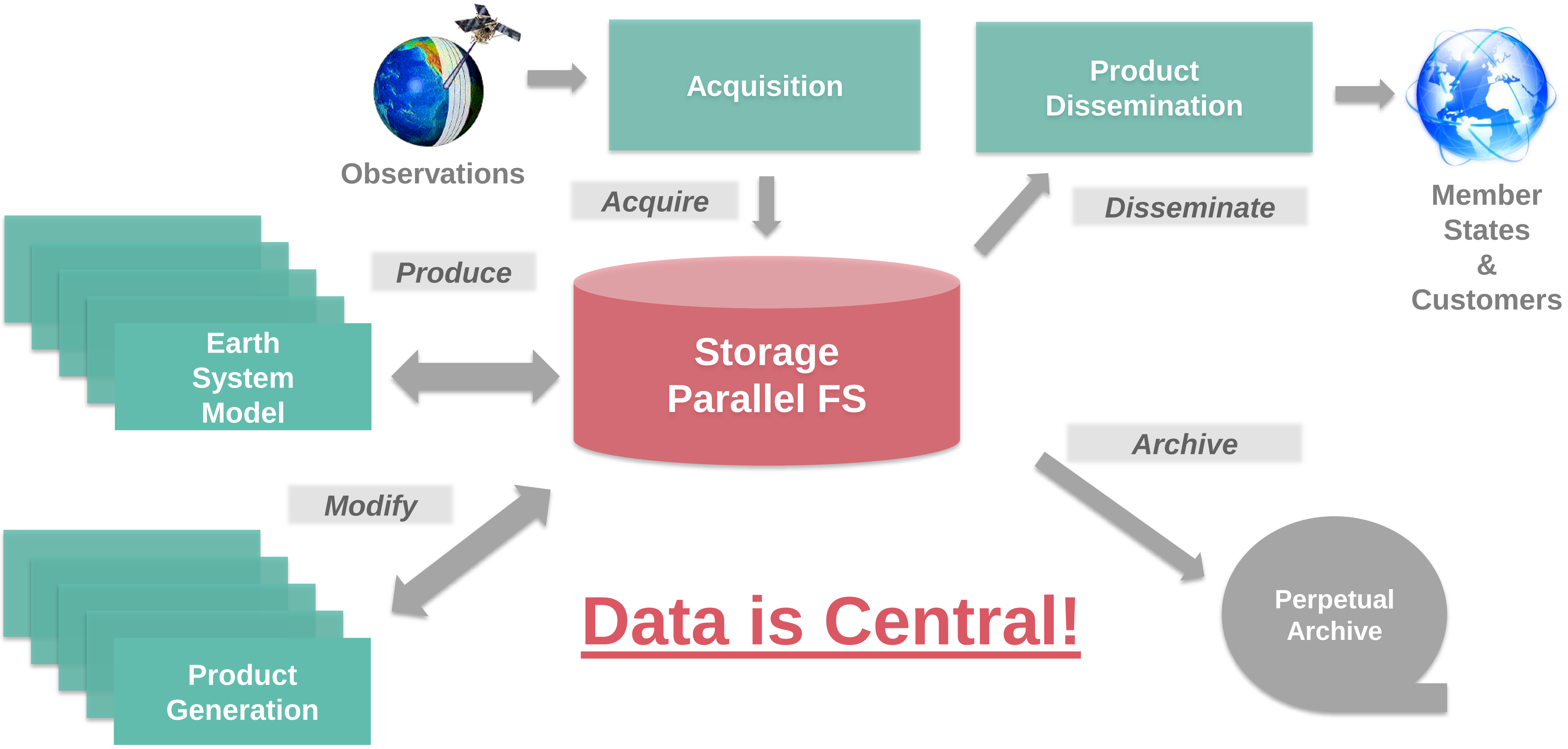
```
/../20210112/1200/24/0/T/...
```

- … but a proper implementation decouples the scientific identification from the storage resource



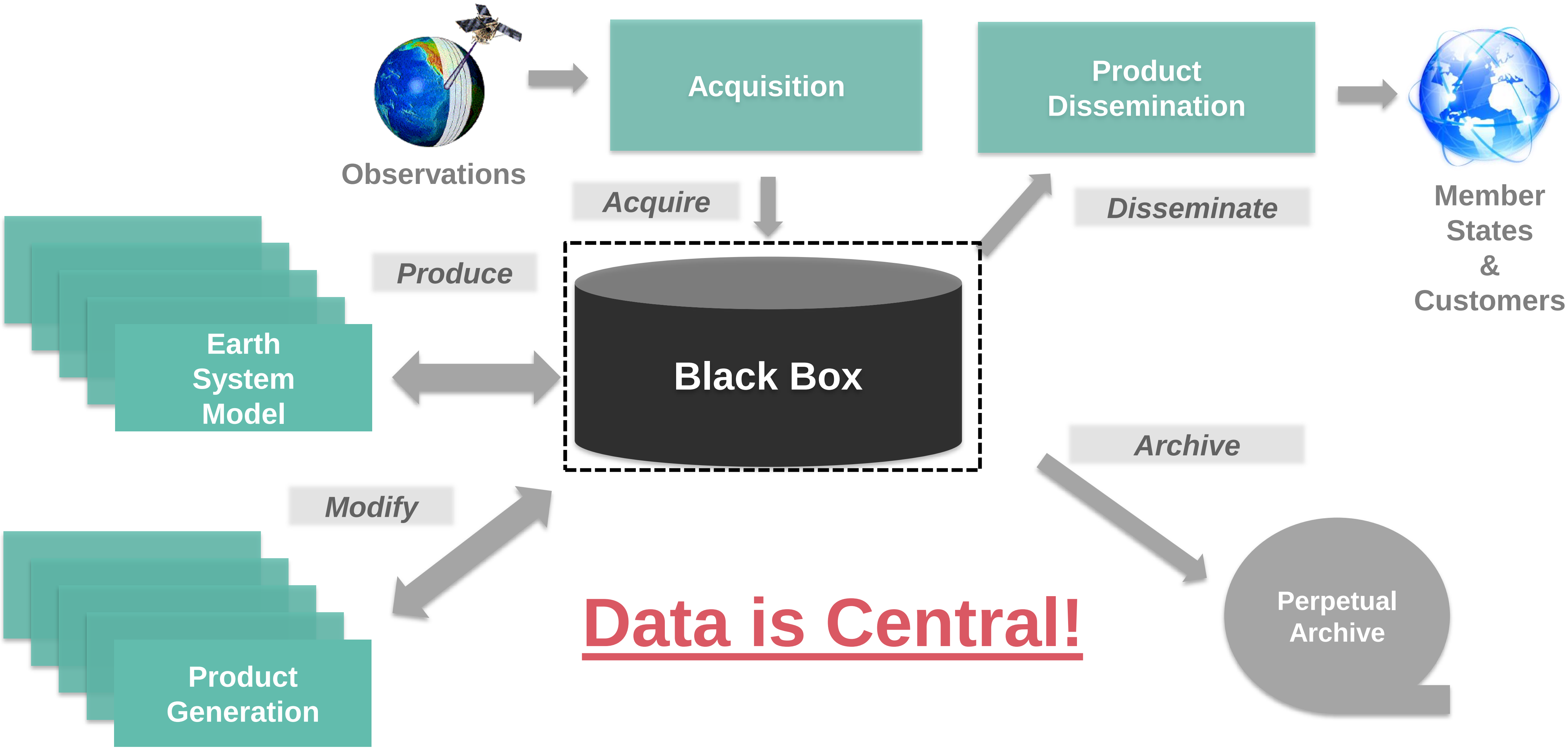
- … and the applications don’t need to care how the objects are stored.

ECMWF’s Production Workflow



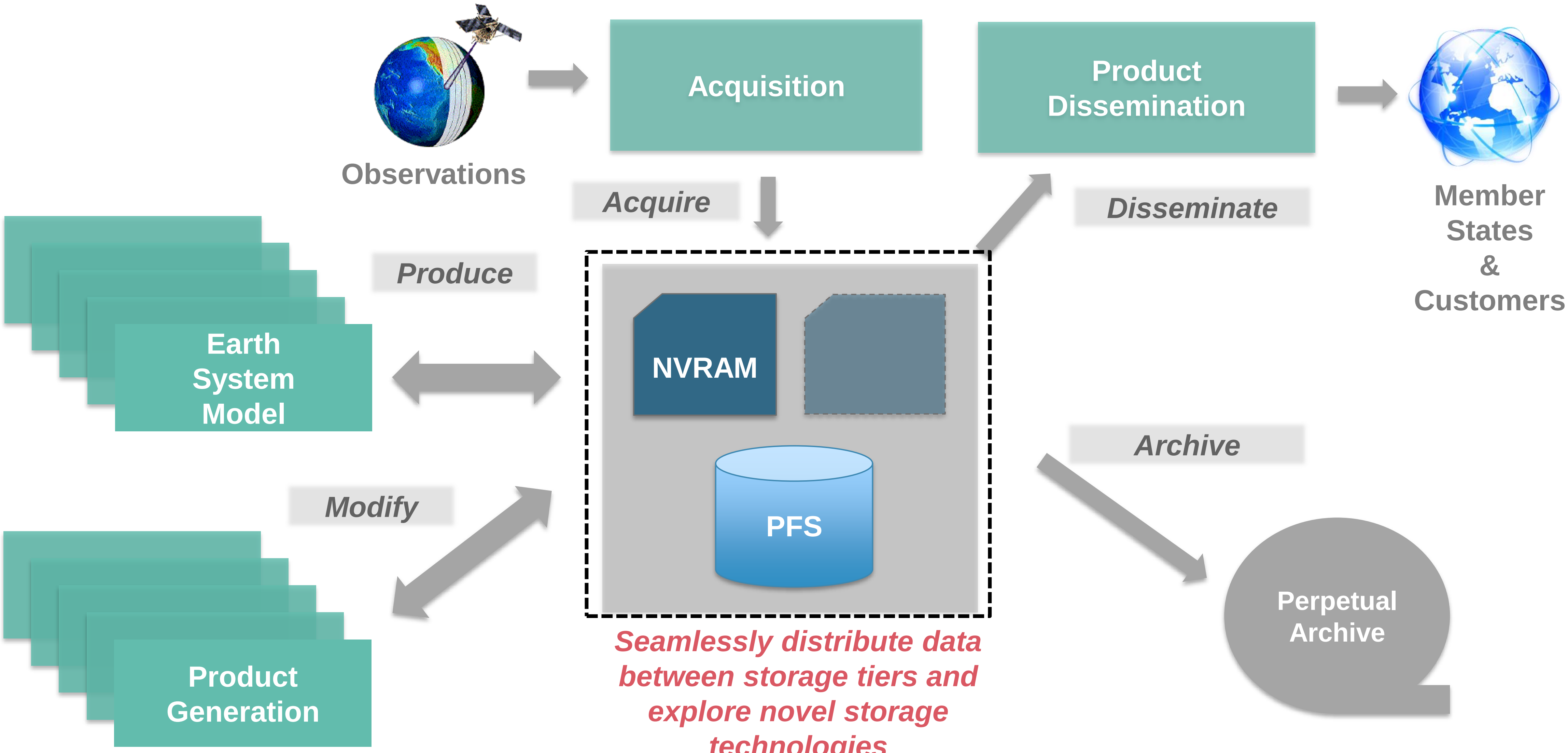
Data is Central!

ECMWF's Production Workflow



Data is Central!

Semantic Data Access > Flexible Data Storage



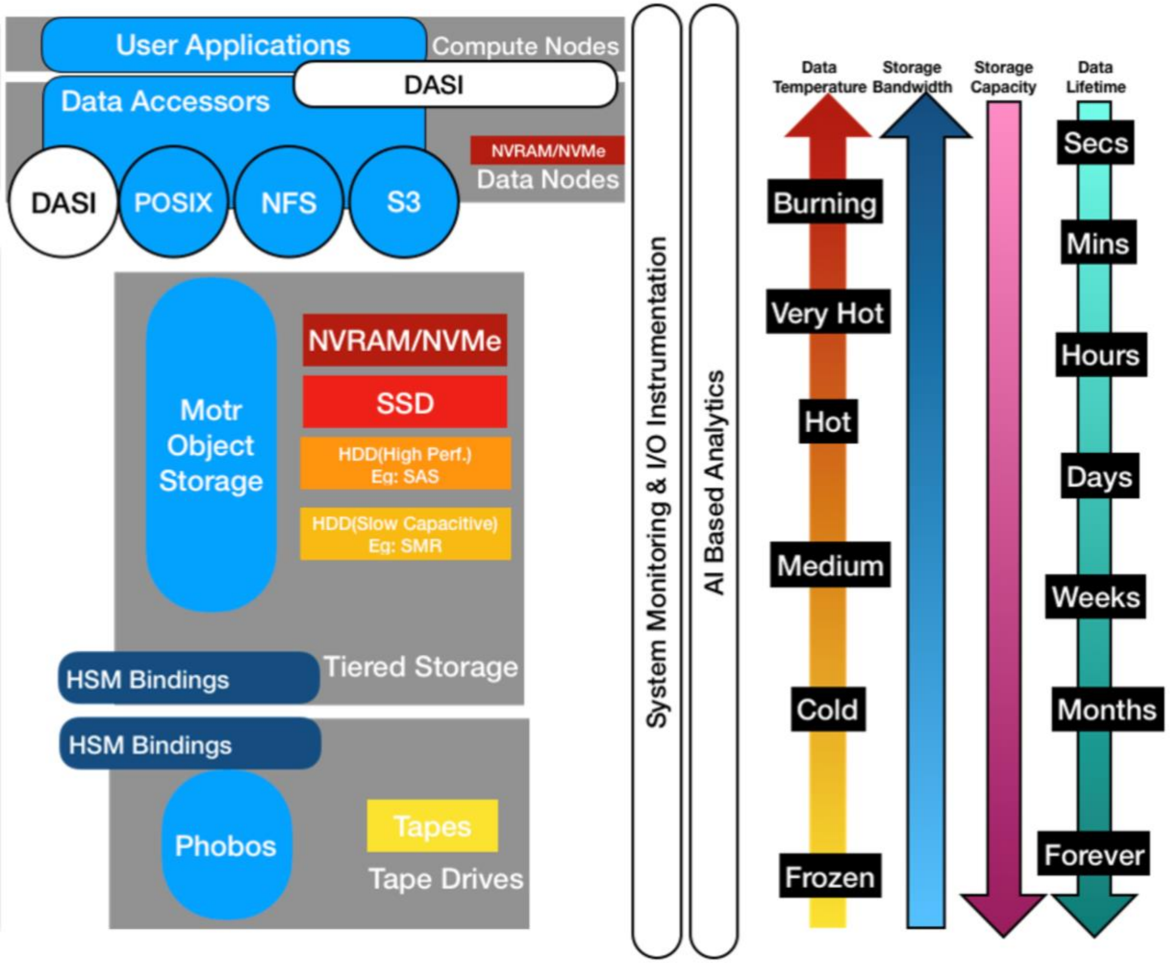
Seamlessly distribute data between storage tiers and explore novel storage technologies

IO-SEA

IO-SEA Project

DASI, a Data Access and Storage Interface, sits between the user applications and HSM as an application interface for abstracting the complex storage layer from users

- Enables data management using domain specific and scientifically meaningful metadata keys
- Separates data management from the underlying backend storage technology



IO-SEA

DASI Design

DASI API

- Frontend abstraction

DASI Core

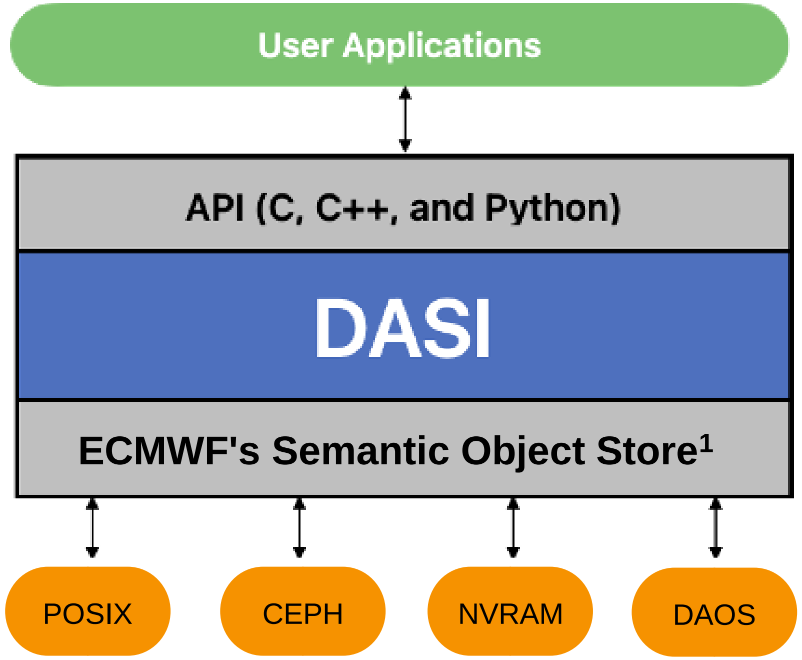
- Converts requests into indexable identifiers
- Expands query requests (ranges, wildcards, etc.)

DASI Index Abstraction

- Mapping between keys and object locations in datastore

DASI Datastore Abstraction

- Object-store-like API for raw storage objects



[1] Fields Database (FDB), https://github.com/ecmwf/fdb

Configuration

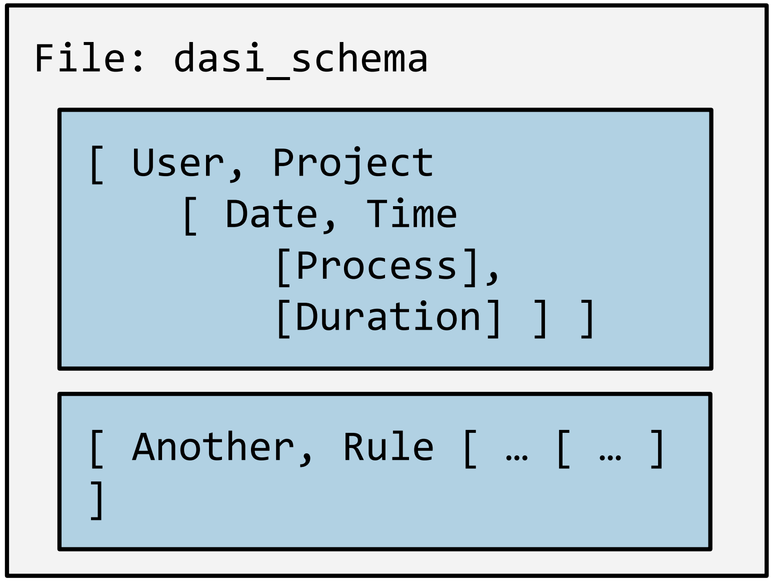
Rules

- Rules specify a set of metadata keys and their hierarchy
- Each rule is a tree with three levels
- Multiple keys can be specified in each level
- Each level can contain multiple branches

Schema

- Collection of rules
- Contains all metadata keys

Example: Rule in a Schema

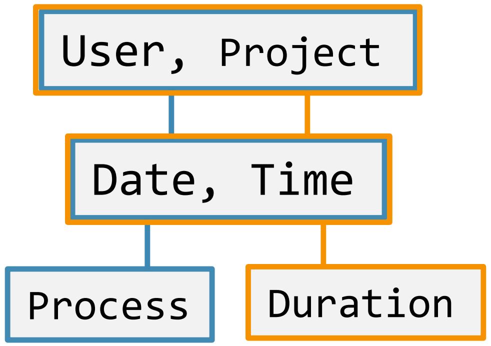


```
File: dasi_schema

[ User, Project
    [ Date, Time
        [Process],
        [Duration] ] ]

[ Another, Rule [ … [ … ]
]
```

Configuration

Data Addresses

- Dictionary of key/value pairs
- Specify all keys along a path from the root node to a leaf node in a rule



Roots

- Storage paths
- Multiple can be specified, with different backend storage technologies
- Roots can be configured differently

Example: DASI config file

```
---
schema: /path/to/schema/file
spaces:
   - roots:
        - path: /path1/to/output/data
          writable: true
        - path: /path2/to/output/data
```

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 keys need?

Determine Hierarchy

- Choose relevant order for the metadata keys
- Schema supports branching

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 ]
  ]
]
```

Example Addresses

```
# 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 ]
  ]
]
```

```
User: jw
Project: Training
Date: 2023/01/01
Time: 1200
Processing: Mean
```

```
Institute: ECMWF
Project: IOSEA
Date: 2023/01/01
Type: Presentation
```

Usage

DASI APIs Available

- C
- C++
- Python
- Command Line Interface (CLI)

Functionality

Archive

- Saves data in the data address provided, a dictionary of key/value pairs compatible with schema

List

- Query contents for subset of metadata keys
- Returns key/values pairs associated to data archived

Retrieve

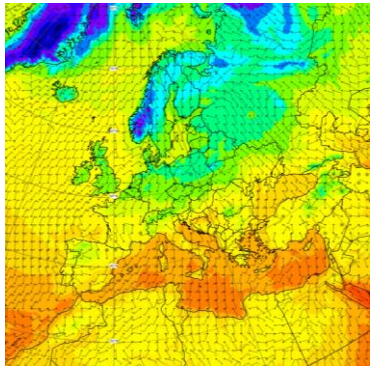
- Returns data associated to data address provided

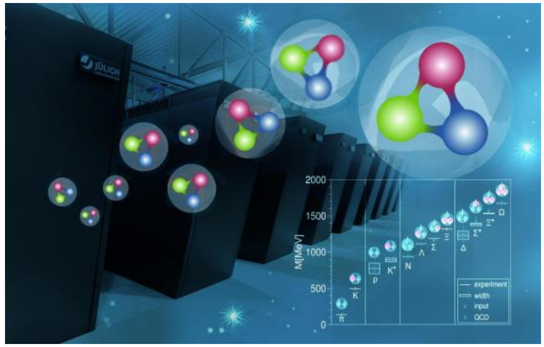
Python API Example

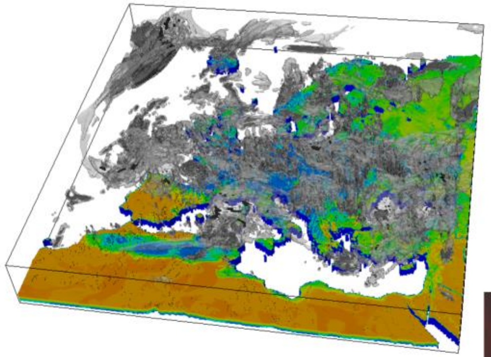
```
dasi = Dasi(“/path/to/config/file”)


# ARCHIVE
key = {“User”: “jw”, “Project”: “IOSEA”, “Date”: “20231101”, “Location”: “Reading”}
dasi.archive(key, data)


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


# LIST
dasi.list(({User:{jw}, Date: {20231101}}) ➔ [metadata]
```

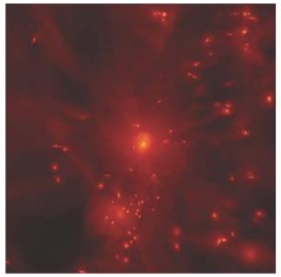
IO-SEA Use Cases

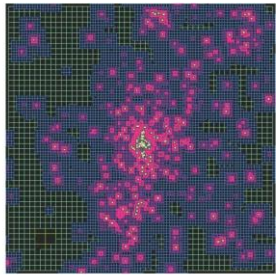
- **ECMWF** uses DASI for Integrated Forecast System weather forecasting workflow
- **Lattice Quantum Chromodynamics** uses DASI for markov-chain scientific checkpoint files
- **Terrestrial Systems Multiple-Physics** (TSMP) uses DASI for output from TSMP model components
- **RAMSES** code for modelling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

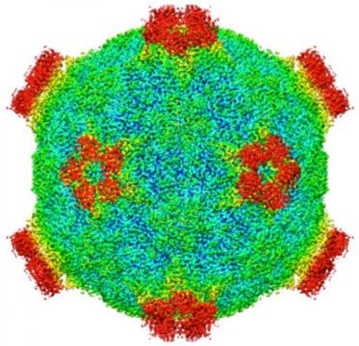












Where to find more about DASI ?

- **Documentation**
    - https://dasi.readthedocs.io/

- **Open-Source Code**
    - github.com/ecmwf-projects/dasi
    - Example: Histogram (Python API)
    - Example: Weather (C API)

- **Binary Packages**
    - https://github.com/ecmwf-projects/dasi/releases/tag/0.2.2