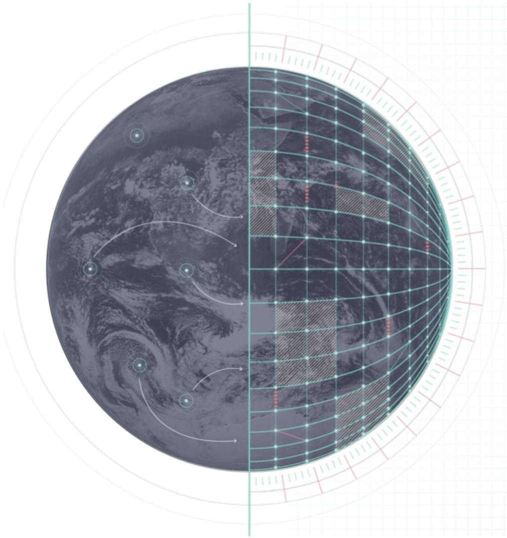
MultIO: A framework for message-driven data routing in high-resolution weather and climate modelling

20th ECMWF Workshop on HPC in Meteorology



Domokos Sármány, Mirco Valentini, Razvan Aguridan, Philipp Geier, James Hawkes, Simon Smart, Tiago Quintino

domokos.sarmany@ecmwf.int

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955648. The JU receives support from the European Union's 2020 Horizon 2020 research and innovation programme and Italy, France, Check Republic, United Kingdom, Greece, Netherlands, Germany, Norway. This project has also received funding from the European Union's Horizon 2020 research and innovation programme through grant agreement 801101.

ECMWF’s Forecasting System

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

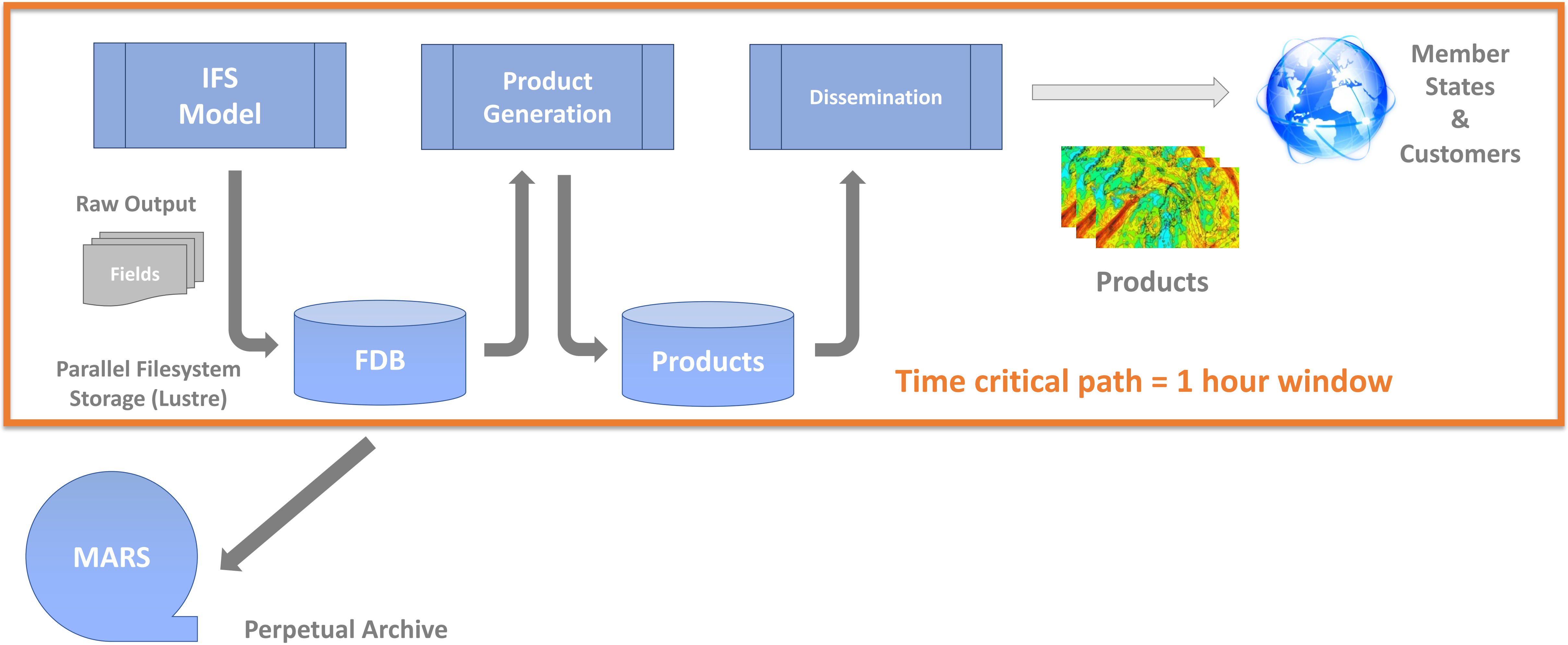
Reading
Bonn
Bologna

Reading, GB

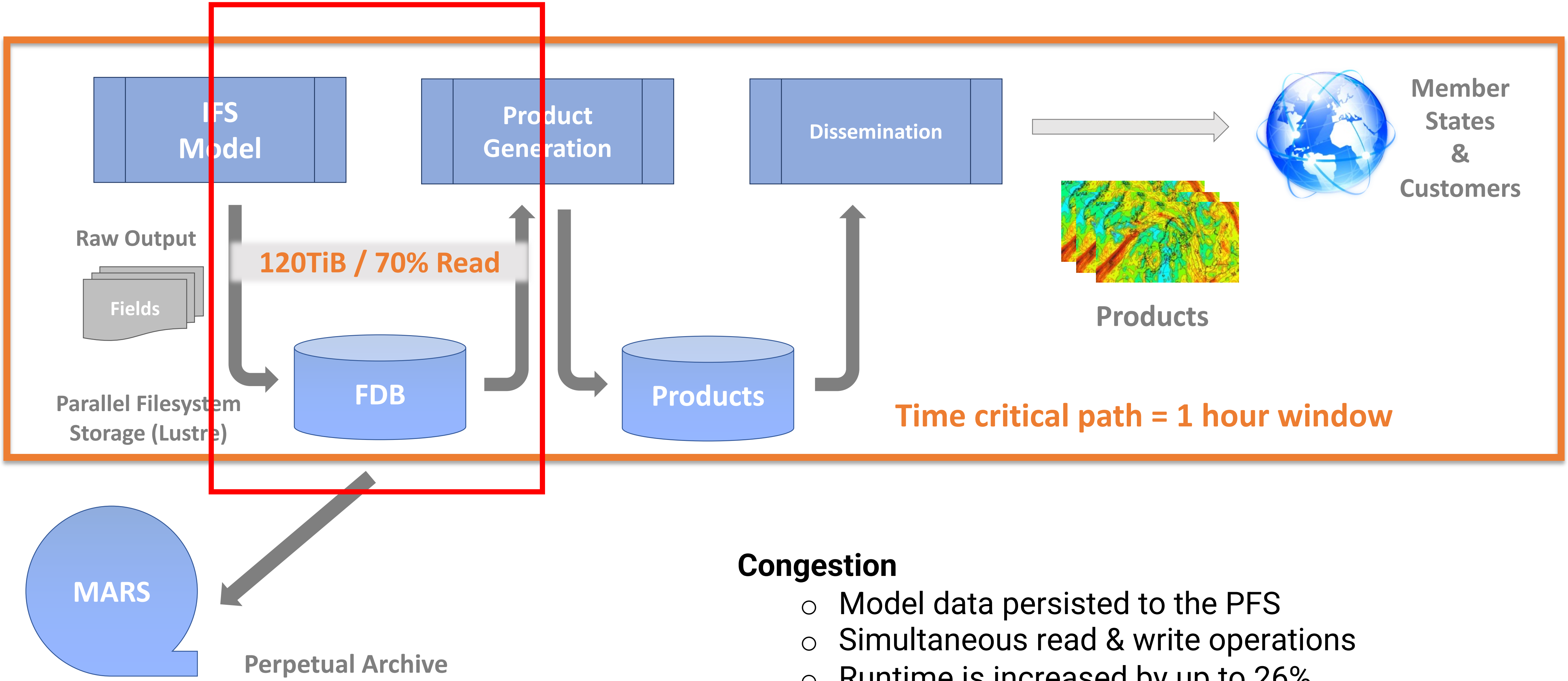
Bonn, DE

Bologna, IT

Use case 1 – ECMWF's production workflow



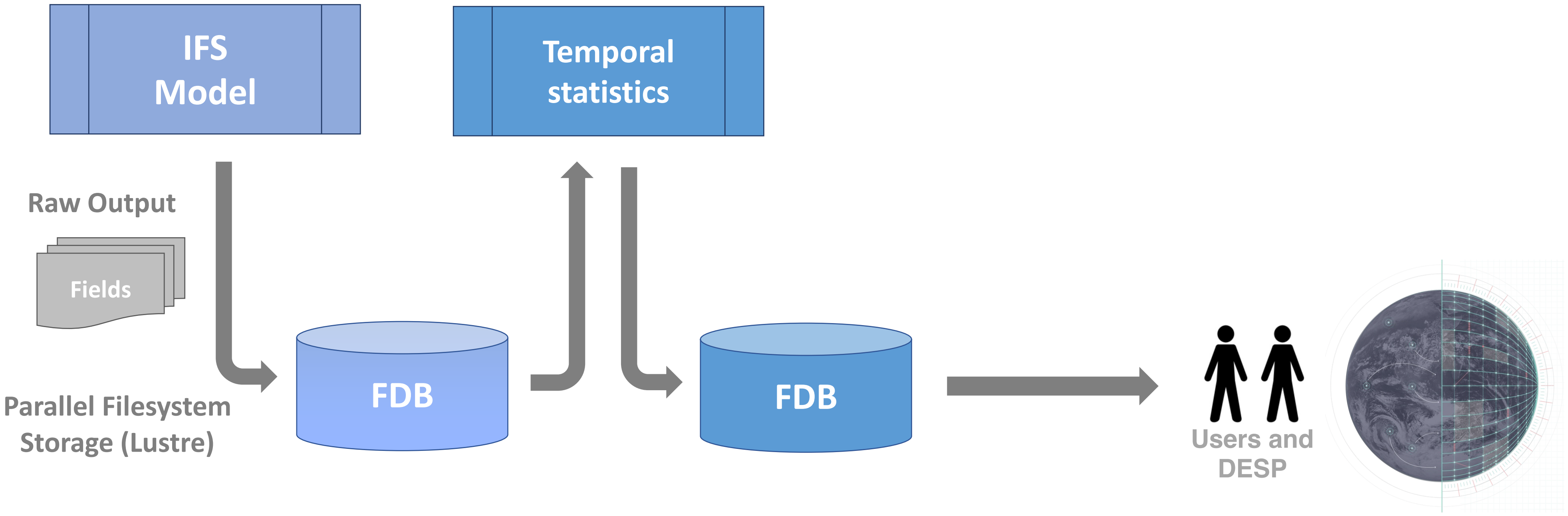
Use case 1 – ECMWF’s production workflow



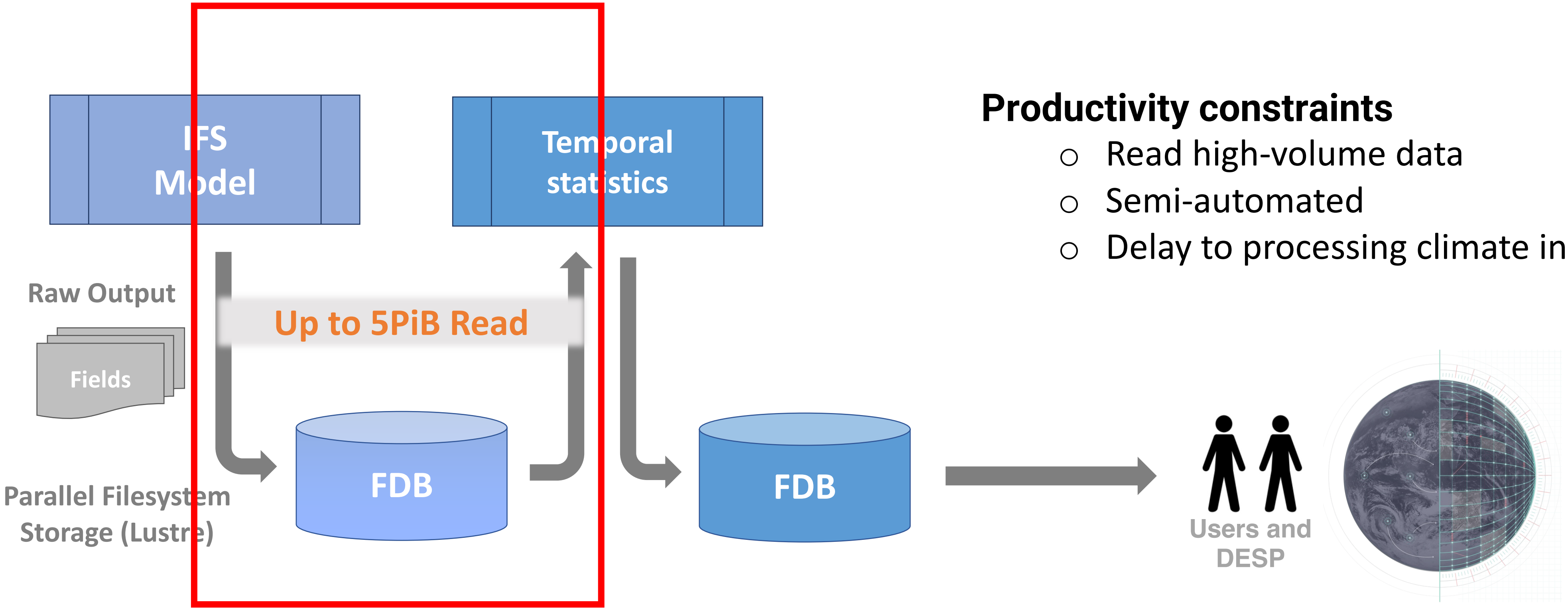
Congestion

- Model data persisted to the PFS
- Simultaneous read & write operations
- Runtime is increased by up to 26%

Use case 2 – High-resolution climate simulation



Use case 2 – High-resolution climate simulation

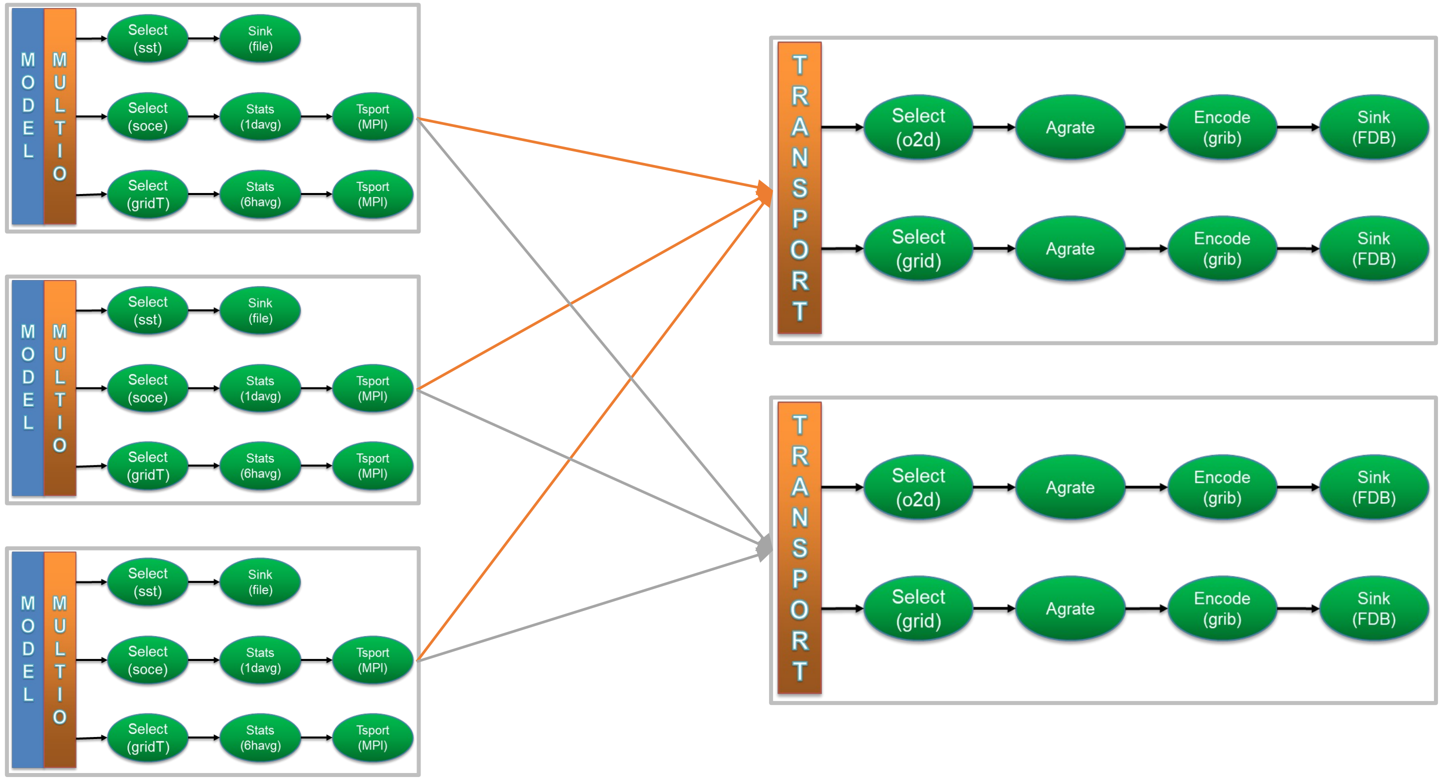


Productivity constraints

- Read high-volume data
- Semi-automated
- Delay to processing climate information

MultIO – high-level view

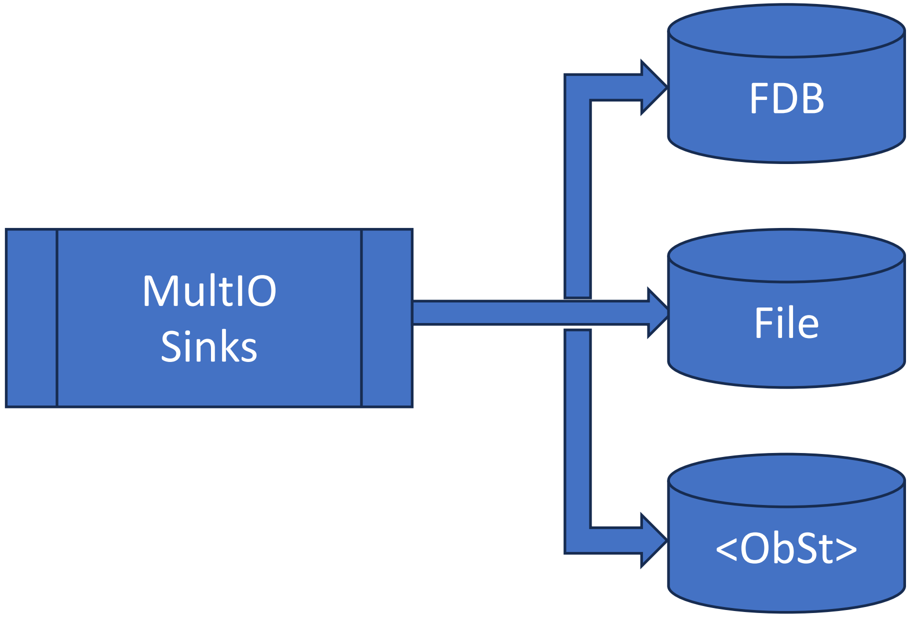
- **I/O-server functionality**
  - Asynchronous
  - Aggregate distributed fields
  - C/Fortran API
- **Processing pipelines**
  - In memory
  - User-programmable
  - Both partial/aggregated fields
  - User/pre-defined actions

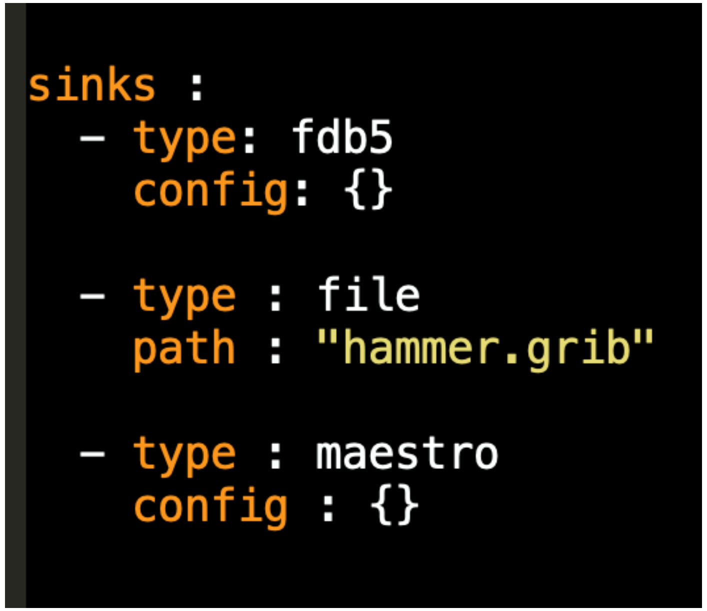


MultIO – a bit of history

🔀 **MultIO – Multiplexing I/O**

- Simultaneous output to multiple storage
- Ideal way to test novel storage technologies



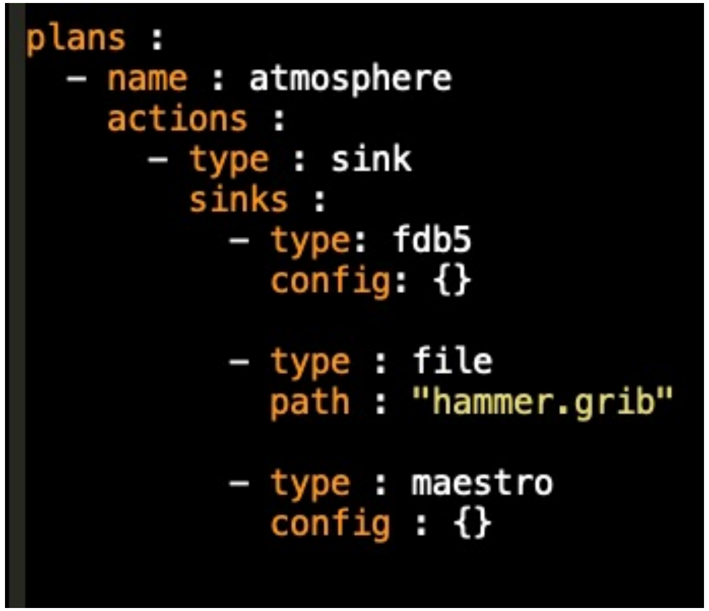


```
sinks :
  - type: fdb5
    config: {}

  - type : file
    path : "hammer.grib"

  - type : maestro
    config : {}
```

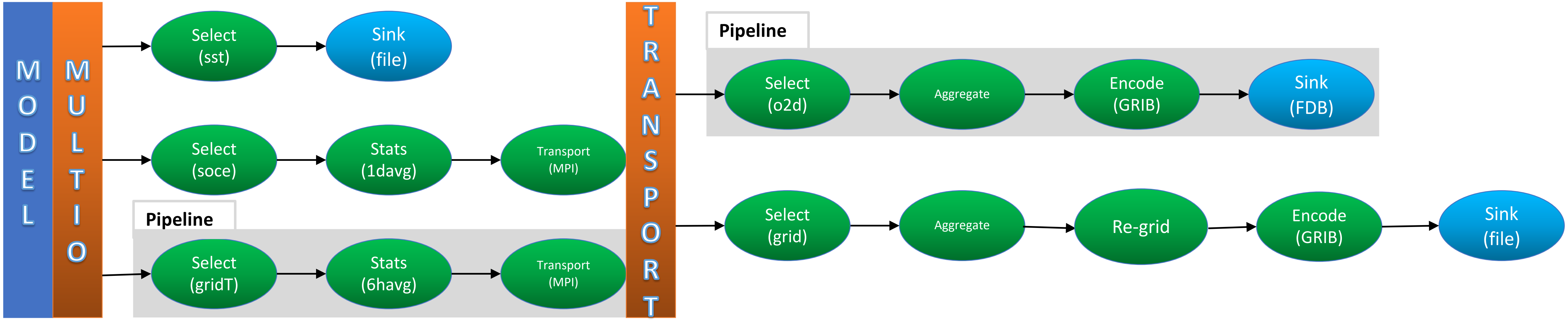
MultIO – a bit of history

🔀 MultIO – Multiplexing I/O

- Simultaneous output to multiple storage
- Ideal way to test novel storage technologies
- Now a single action in the pipelines
- More than just I/O – on-the-fly post-processing with multiple pipelines

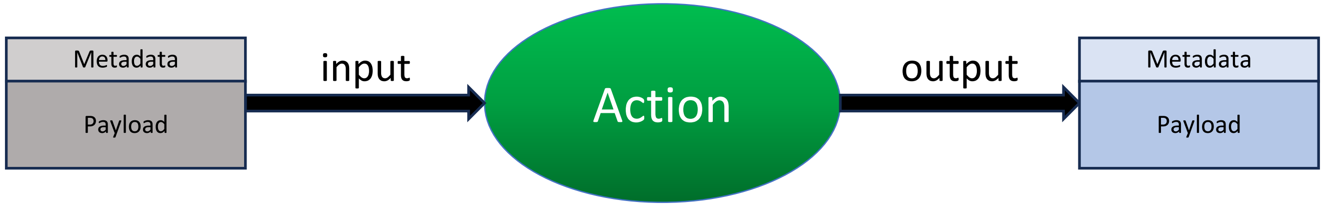


```
plans :
  - name : atmosphere
    actions :
      - type : sink
        sinks :
          - type: fdb5
            config: {}

          - type : file
            path : "hammer.grib"

          - type : maestro
            config : {}
```



Message-driven routing decisions

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

Data payload
 - Field data (e.g. array of doubles)
 - GRIB2
 - Grid partition
```



Message

- Metadata, a unique description
- Payload

Contract

- Between message and action
- Decisions are based on
  - input message’s metadata
  - action’s default behaviour
  - action’s configuration
- Output message can be
  - same message
  - same payload, new metadata
  - new message
  - no message

Message-driven routing decisions

👱 **User-defined actions**

- The user in full control by defining the contract
- The user populates the metadata through the multio interface
- The user defines the action’s behaviour and implements it (conforming to multio’s action interface)

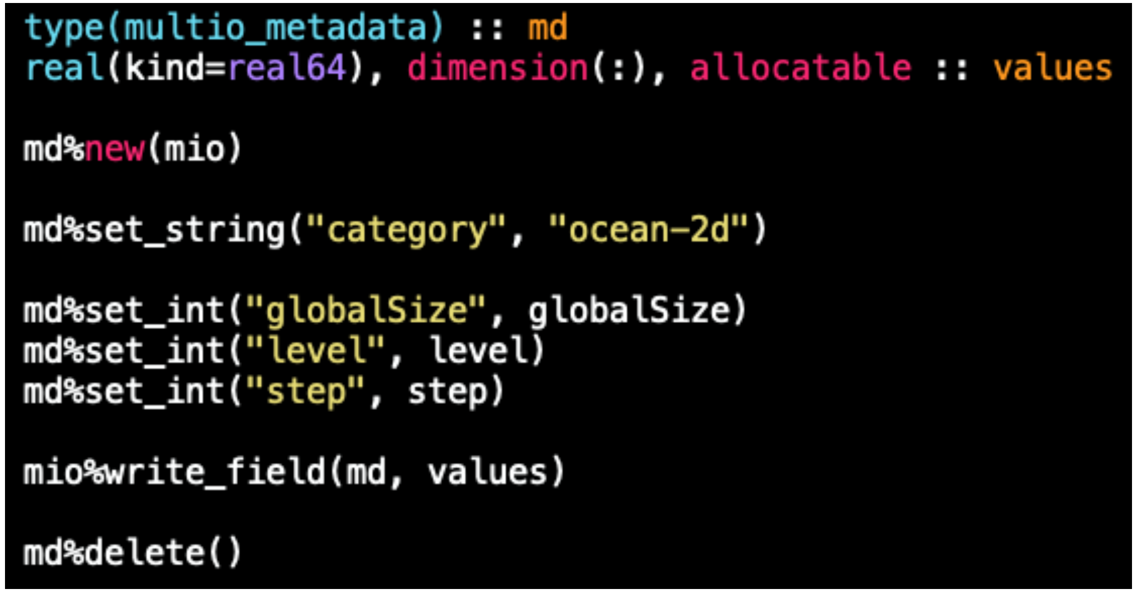
💻 **Pre-defined actions (shipped with multio)**

- The user need to be aware of the requirements on the metadata
- The user still populates the metadata through the multio interface

👨‍💻 **Both user-defined and pre-defined actions**

- The user need to be aware of metadata injected into the message by previous actions

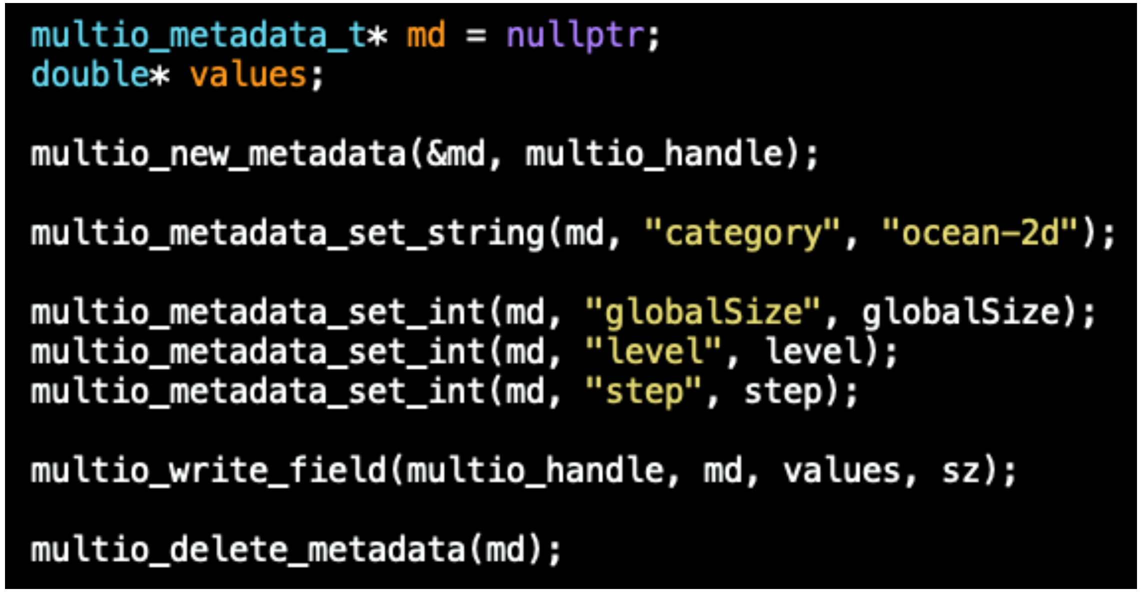
Pipeline interface



```fortran
type(multio_metadata) :: md
real(kind=real64), dimension(:), allocatable :: values

md%new(mio)

md%set_string("category", "ocean-2d")

md%set_int("globalSize", globalSize)
md%set_int("level", level)
md%set_int("step", step)

mio%write_field(md, values)

md%delete()
```

🧑‍💻 **Fortran/C API**

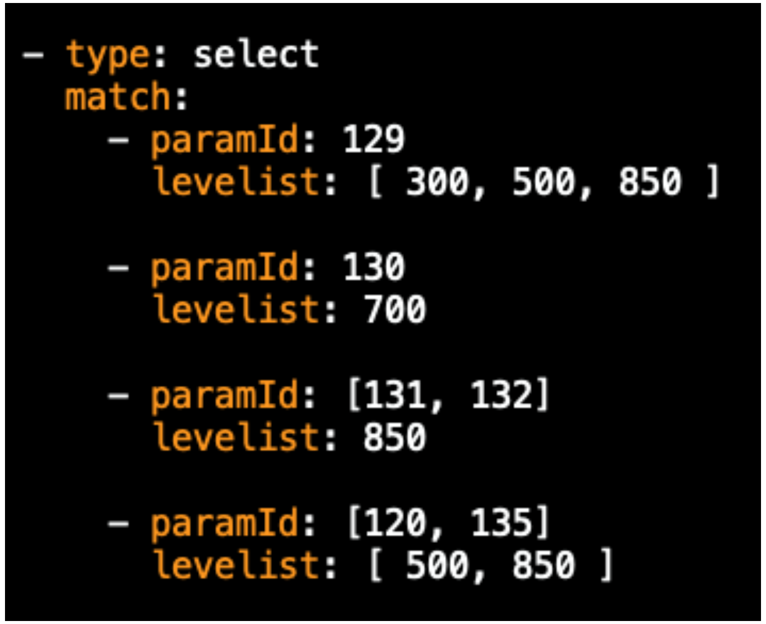
- Metadata is a key-value dictionary
- Create metadata handle
- Populate metadata
- Pass metadata + data
- Delete metadata handle



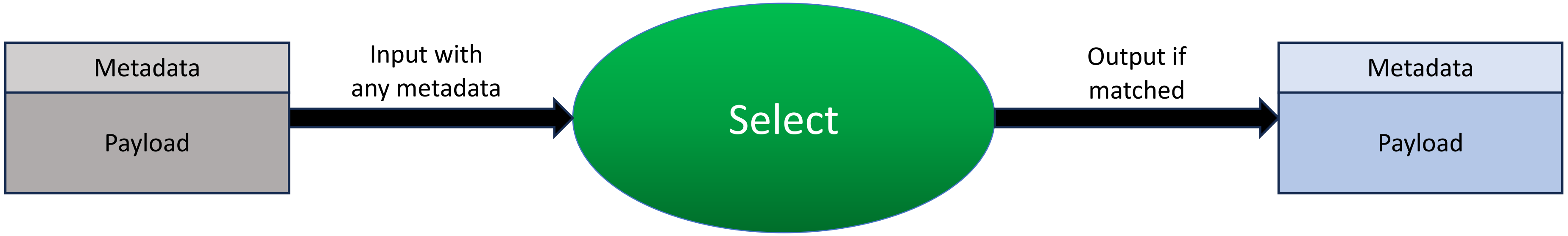
```c
multio_metadata_t* md = nullptr;
double* values;

multio_new_metadata(&md, multio_handle);

multio_metadata_set_string(md, "category", "ocean-2d");

multio_metadata_set_int(md, "globalSize", globalSize);
multio_metadata_set_int(md, "level", level);
multio_metadata_set_int(md, "step", step);

multio_write_field(multio_handle, md, values, sz);

multio_delete_metadata(md);
```

Pre-defined actions: Select

- Filter on metadata
- Never change the message
- Filter on any keys
- Both ‘and’ and ‘or’ supported



```
- type: select
  match:
    - paramId: 129
      levelist: [ 300, 500, 850 ]

    - paramId: 130
      levelist: 700

    - paramId: [131, 132]
      levelist: 850

    - paramId: [120, 135]
      levelist: [ 500, 850 ]
```



Metadata

Payload

Input with any metadata

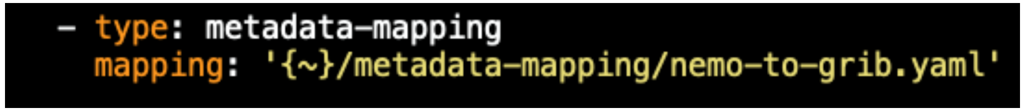
Select

Output if matched

Metadata

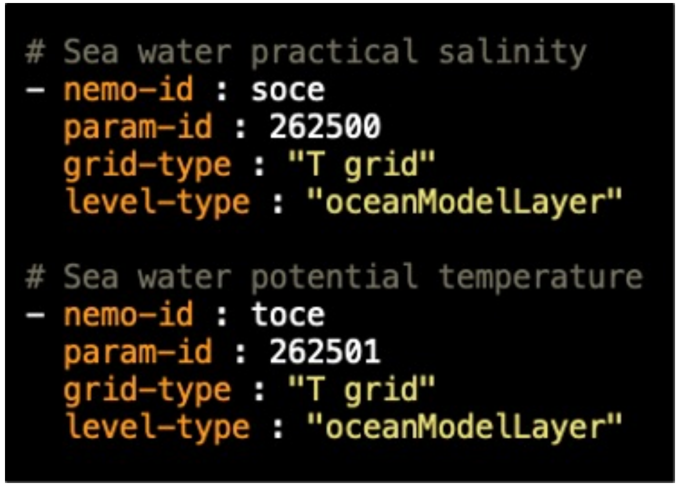
Payload

Pre-defined actions: MetadataMapping

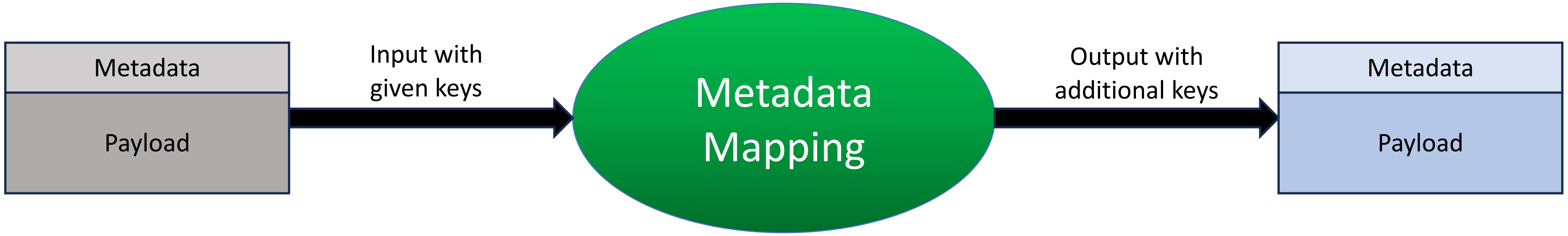


```
- type: metadata-mapping
  mapping: '{~}/metadata-mapping/nemo-to-grib.yaml'
```

- Inject additional metadata
- Never change the payload
- Always apply
- Require mapped-from metadata



```
# Sea water practical salinity
- nemo-id : soce
  param-id : 262500
  grid-type : "T grid"
  level-type : "oceanModelLayer"

# Sea water potential temperature
- nemo-id : toce
  param-id : 262501
  grid-type : "T grid"
  level-type : "oceanModelLayer"
```



Metadata

Payload

Input with given keys

Metadata Mapping

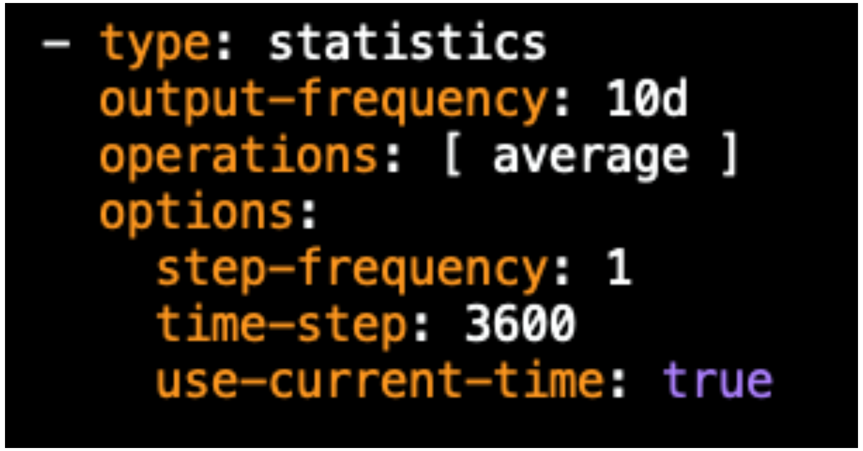
Output with additional keys

Metadata

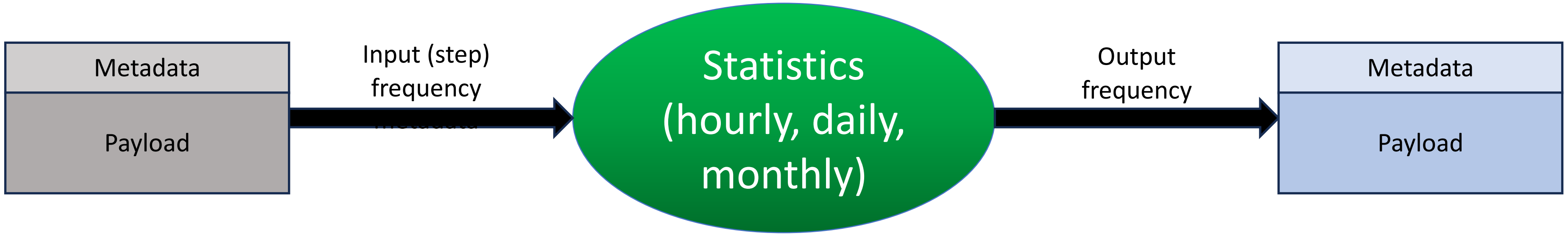
Payload

Pre-defined actions: Statistics

- Min/max
- Accumulate
- Average
- Flux average
- 🔨 Standard deviation
- 🔨 Synoptic means
- 🔨 Statistics restarts

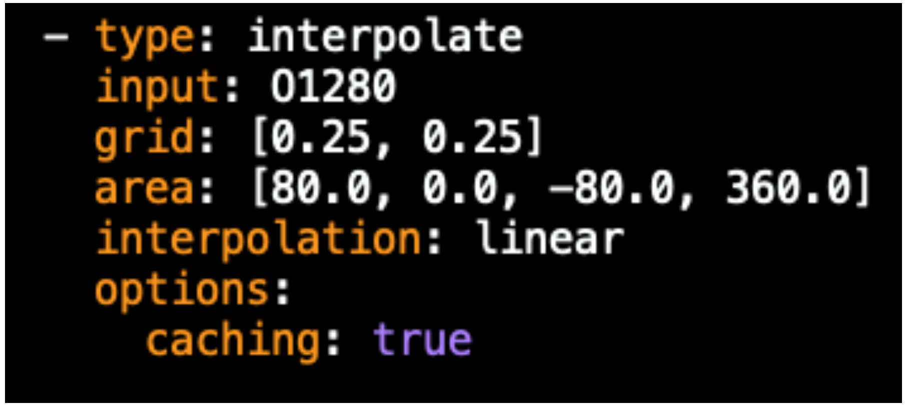


```
- type: statistics
  output-frequency: 10d
  operations: [ average ]
  options:
    step-frequency: 1
    time-step: 3600
    use-current-time: true
```

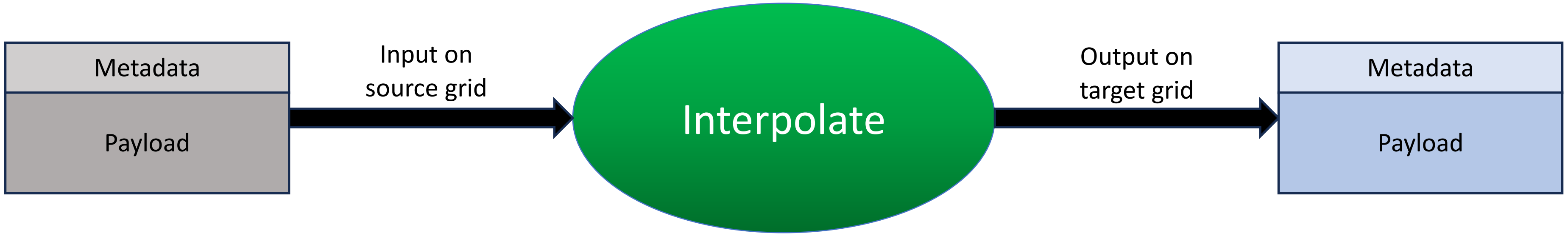


Pre-defined actions: Interpolate (re-grid)

- Uses the MIR library internally (github.com/ecmwf/mir)
- Interpolation between supported grids
- Cropping to rectangular domains
- Recent: support (e)ORCA ocean and HEALPix grids

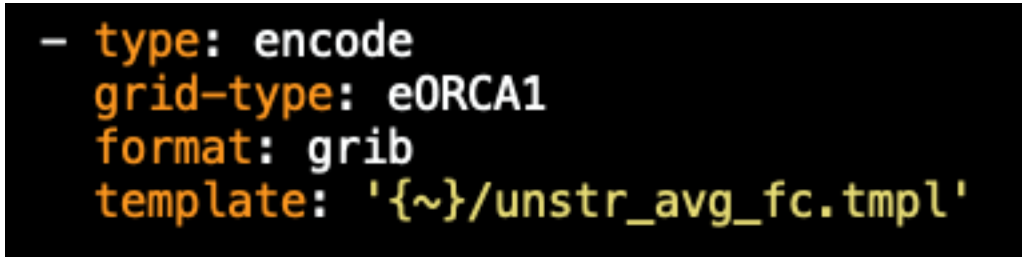


```yaml
- type: interpolate
  input: O1280
  grid: [0.25, 0.25]
  area: [80.0, 0.0, -80.0, 360.0]
  interpolation: linear
  options:
    caching: true
```

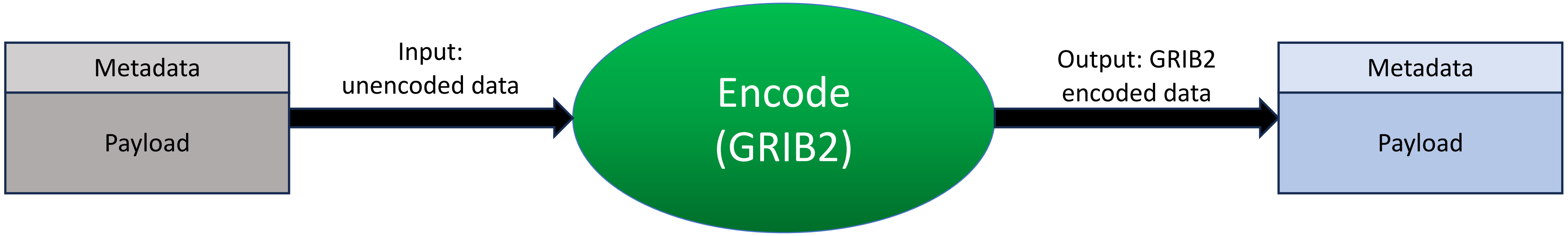


Pre-defined actions: Encode

- Uses eccodes internally (github.com/ecmwf/eccodes)
- Full GRIB2 support
- Some GRIB1 support (phasing out)



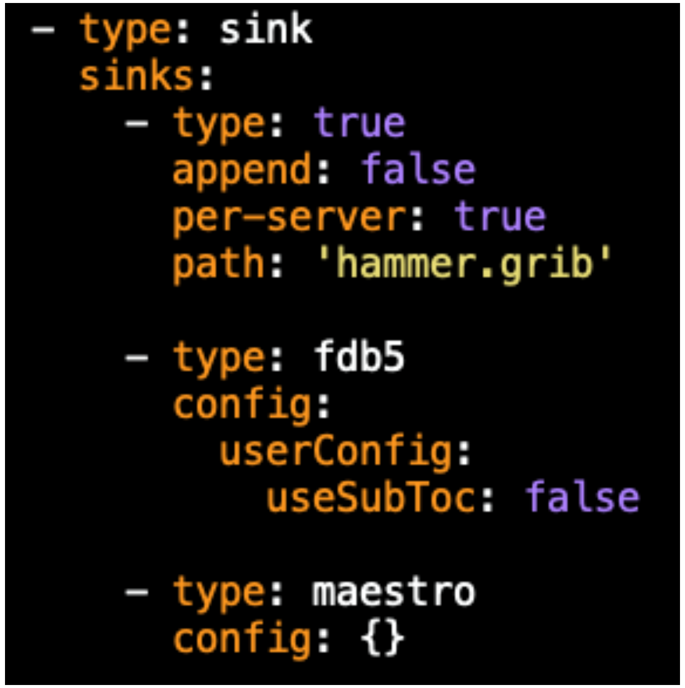
```
- type: encode
  grid-type: eORCA1
  format: grib
  template: '{~}/unstr_avg_fc.tmpl'
```



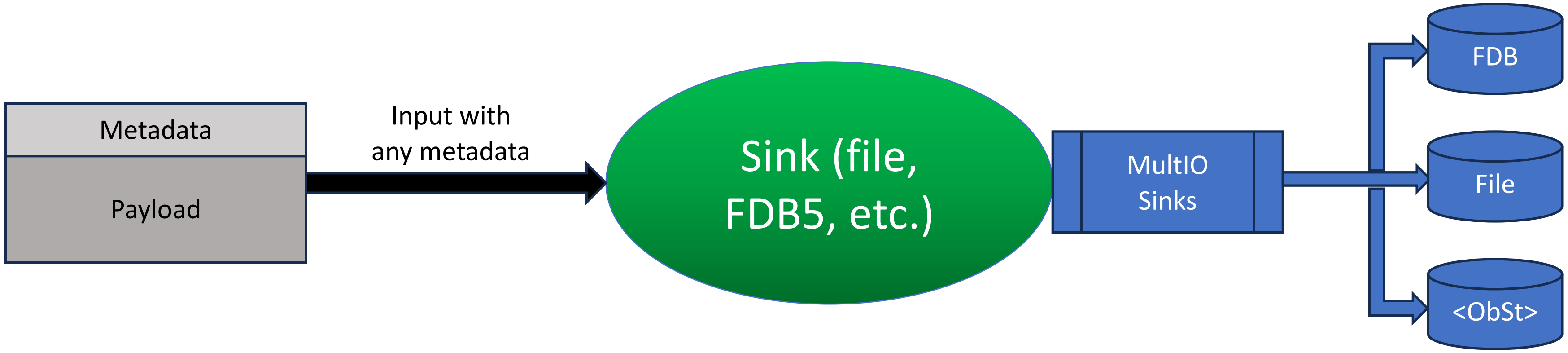
Pre-defined actions: Sink

MultIO sinks – Multiplexing I/O

- Simultaneous output to multiple storage
- Ideal way to test novel storage technologies
- Now a single action in the pipelines
- More than just I/O – on-the-fly post-processing with multiple pipelines

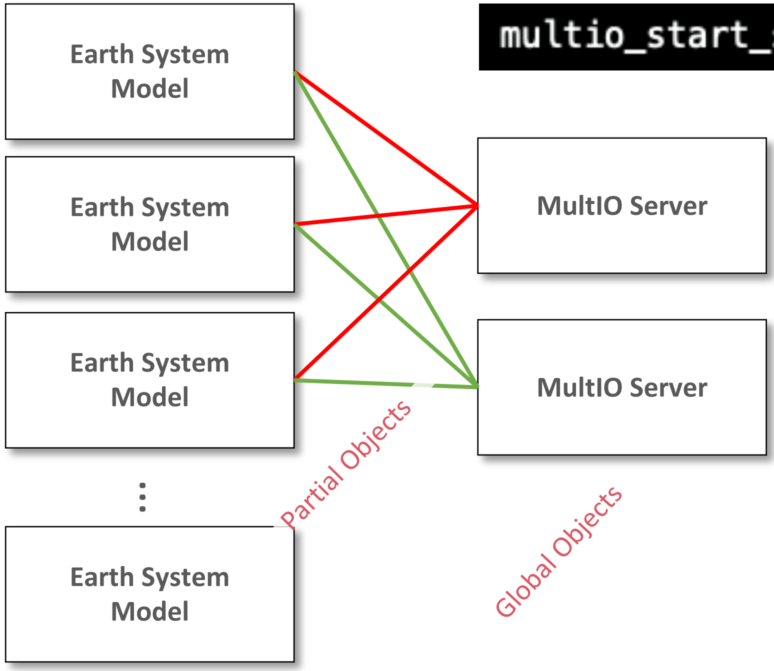


```
- type: sink
  sinks:
    - type: true
      append: false
      per-server: true
      path: 'hammer.grib'

    - type: fdb5
      config:
        userConfig:
          useSubToc: false

    - type: maestro
      config: {}
```



I/O-server additional interface

```
multio_start_server(mio_conf)
```



```
mio%open_connections()

mio%write_domain(md, domain_data)

mio%write_mask(md, zmask)

mio%close_connections()
```

🧑‍💻 **Additional API for distributed data**

- ○ Single API call for server
- ○ Transport-layer abstraction
- ○ Book-keeping for topology
- ○ Local-to-global index mapping
- ○ Land-sea mask information

Usage 1: Ocean re-analysis

⚡ **Release!**

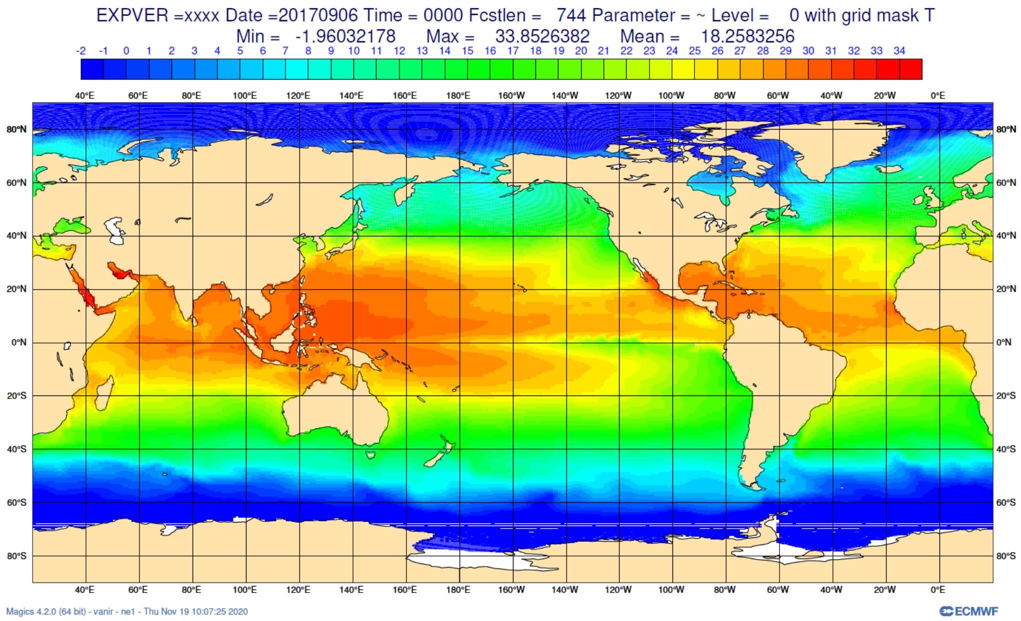
- ○ Version **2.0.1** for ORAS6 for production with NEMOv4 to begin in October/November

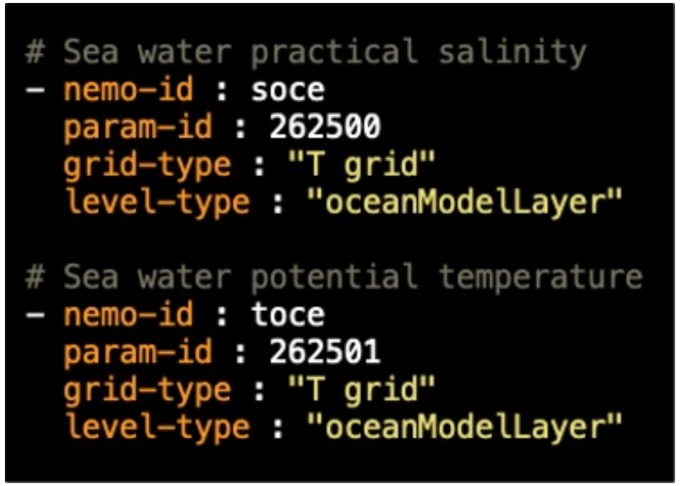
🌊 **GRIB2 ocean data in MARS**

- ○ Many new definitions for ocean
- ○ Support for (e)ORCA grids, curated and stored in atlas-io format (github.com/ecmwf/atlas-orca)

🌐 **NEMOv4 I/O-server & pipelines**

- ○ Compute hourly/daily/monthly statistics
- ○ Aggregation rules for (e)ORCA grids
- ○ Fully integrated in IFS operational toolchain


EXPVER =xxxx Date =20170906 Time = 0000 Fcstlen = 744 Parameter = ~ Level = 0 with grid mask T
Min = -1.96032178 Max = 33.8526382 Mean = 18.2583256
-2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
40°E 60°E 80°E 100°E 120°E 140°E 160°E 180°E 160°W 140°W 120°W 100°W 80°W 60°W 40°W 20°W 0°E
80°N 60°N 40°N 20°N 0°N 20°S 40°S 60°S 80°S
Magics 4.2.0 (64 bit) - vanir - ne1 - Thu Nov 19 10:07:25 2020
ECMWF

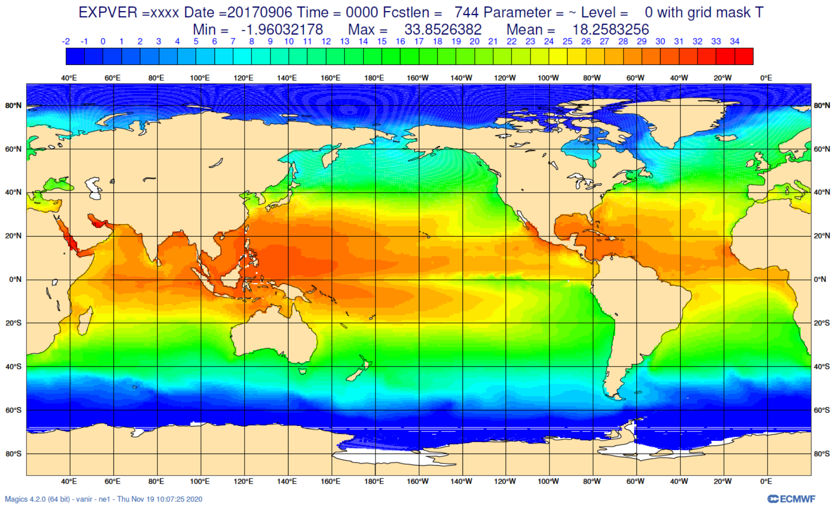


```
# Sea water practical salinity
- nemo-id : soce
  param-id : 262500
  grid-type : "T grid"
  level-type : "oceanModelLayer"

# Sea water potential temperature
- nemo-id : toce
  param-id : 262501
  grid-type : "T grid"
  level-type : "oceanModelLayer"
```

Usage 1: Ocean re-analysis

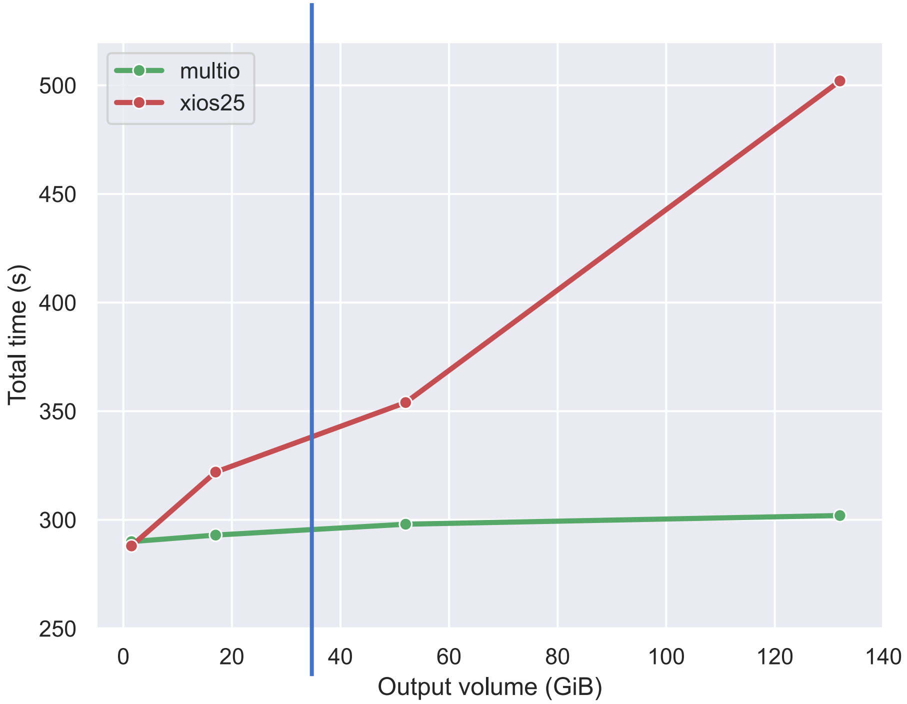
🎬 **Ocean6 re-analysis current production plans**

- ○ 6424 five-day assimilation loops
- ○ 11 ensemble members
- ○ 300 compute tasks per member
- ○ 20 I/O tasks per member

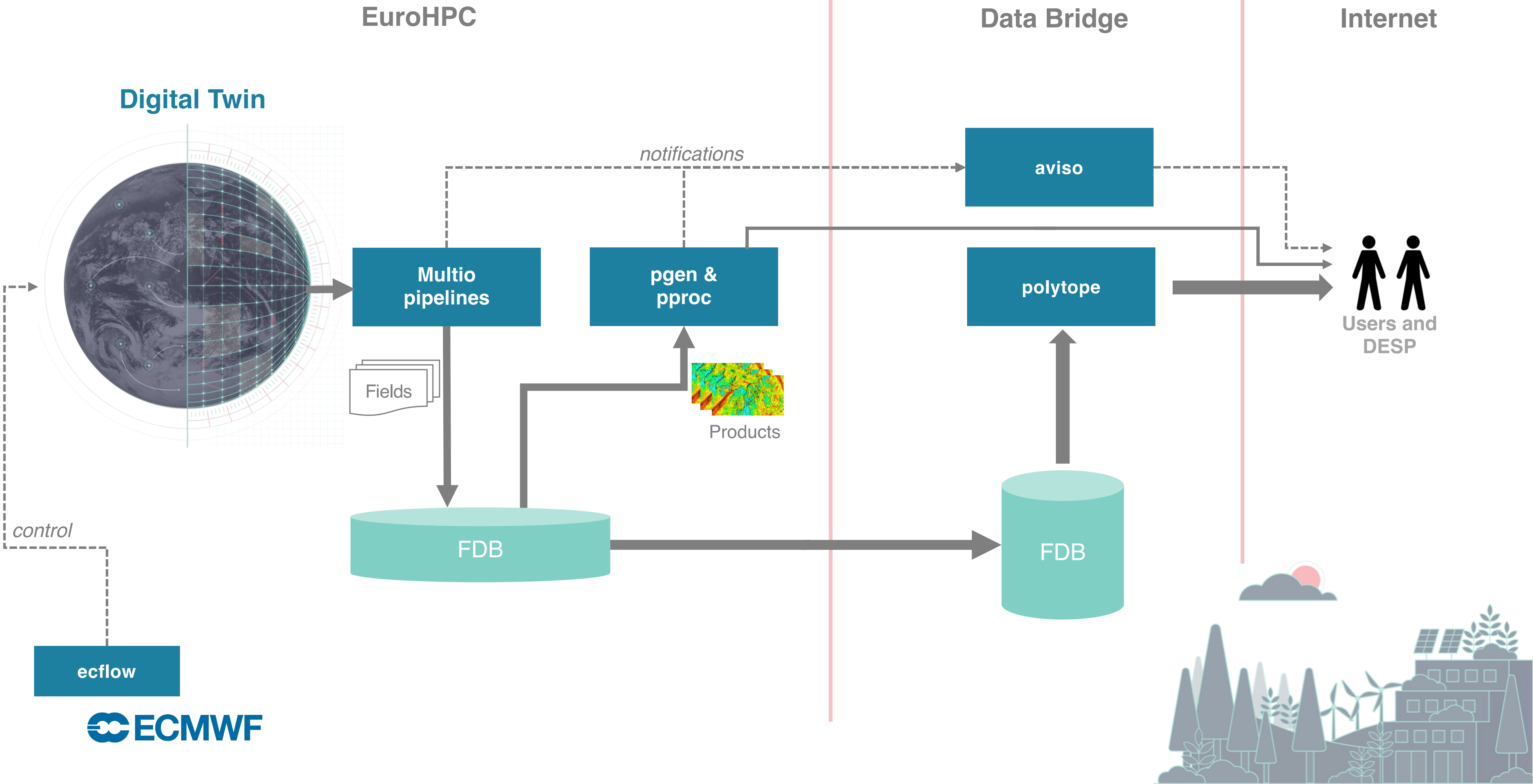


EXPVER =xxxx Date =20170906 Time = 0000 Fcstlen = 744 Parameter = ~ Level = 0 with grid mask T
Min = -1.96032178 Max = 33.8526382 Mean = 18.2583256
-2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
40°E 60°E 80°E 100°E 120°E 140°E 160°E 180°E 160°W 140°W 120°W 100°W 80°W 60°W 40°W 20°W 0°E
80°N 60°N 40°N 20°N 0°N 20°S 40°S 60°S 80°S
Magics 4.2.0 (64 bit) - vanir - ne1 - Thu Nov 19 10:07:25 2020
ECMWF

Comparison to xios 2.5 (best effort)

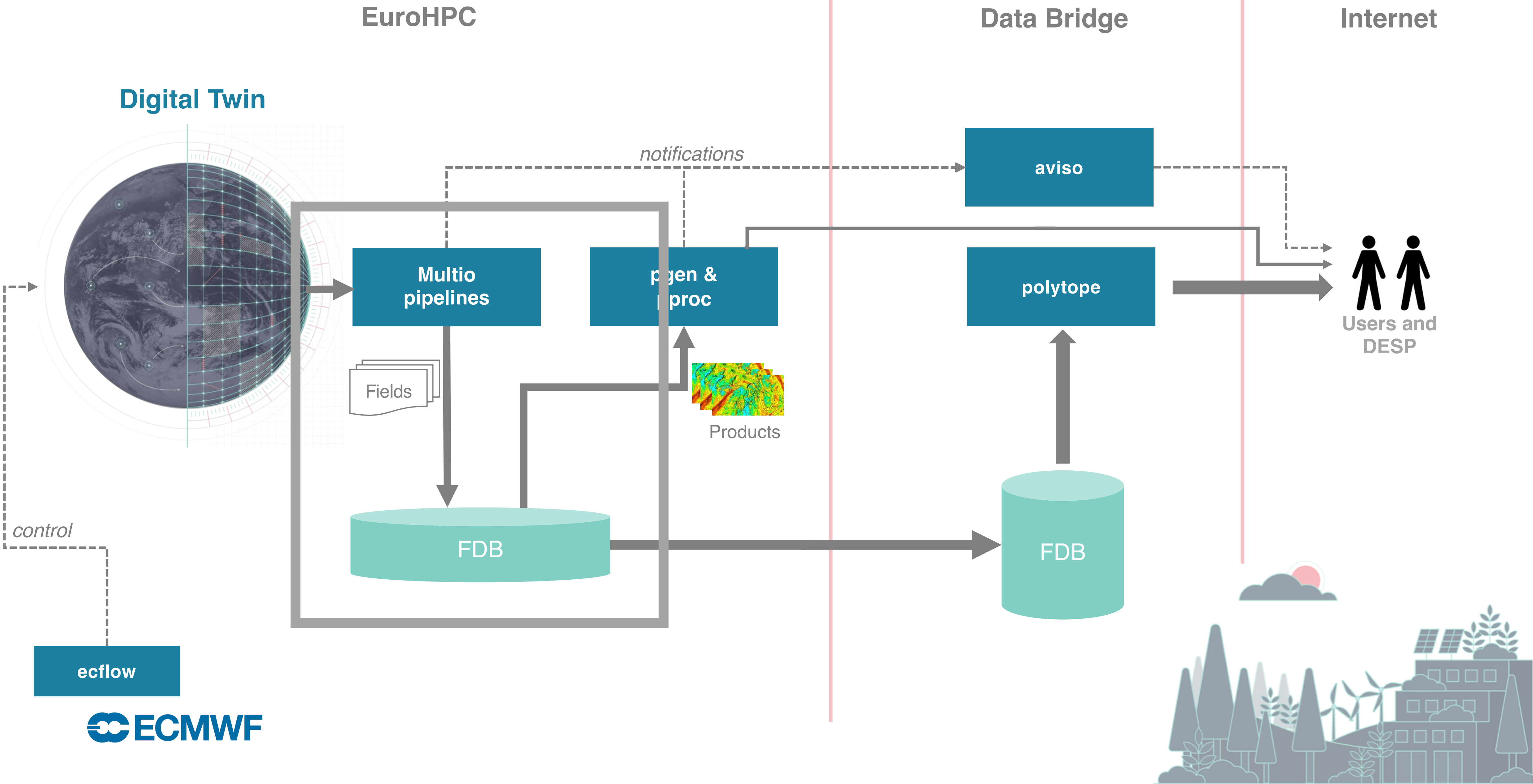


Usage 2: Destination Earth Phase 1 / NextGEMS



ECMWF

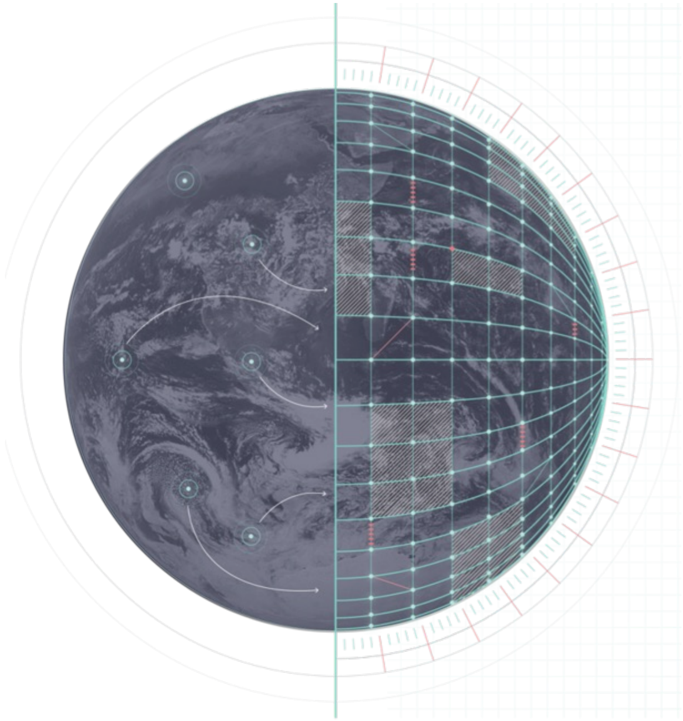
Usage 2: Destination Earth Phase 1 / NextGEMS



ECMWF

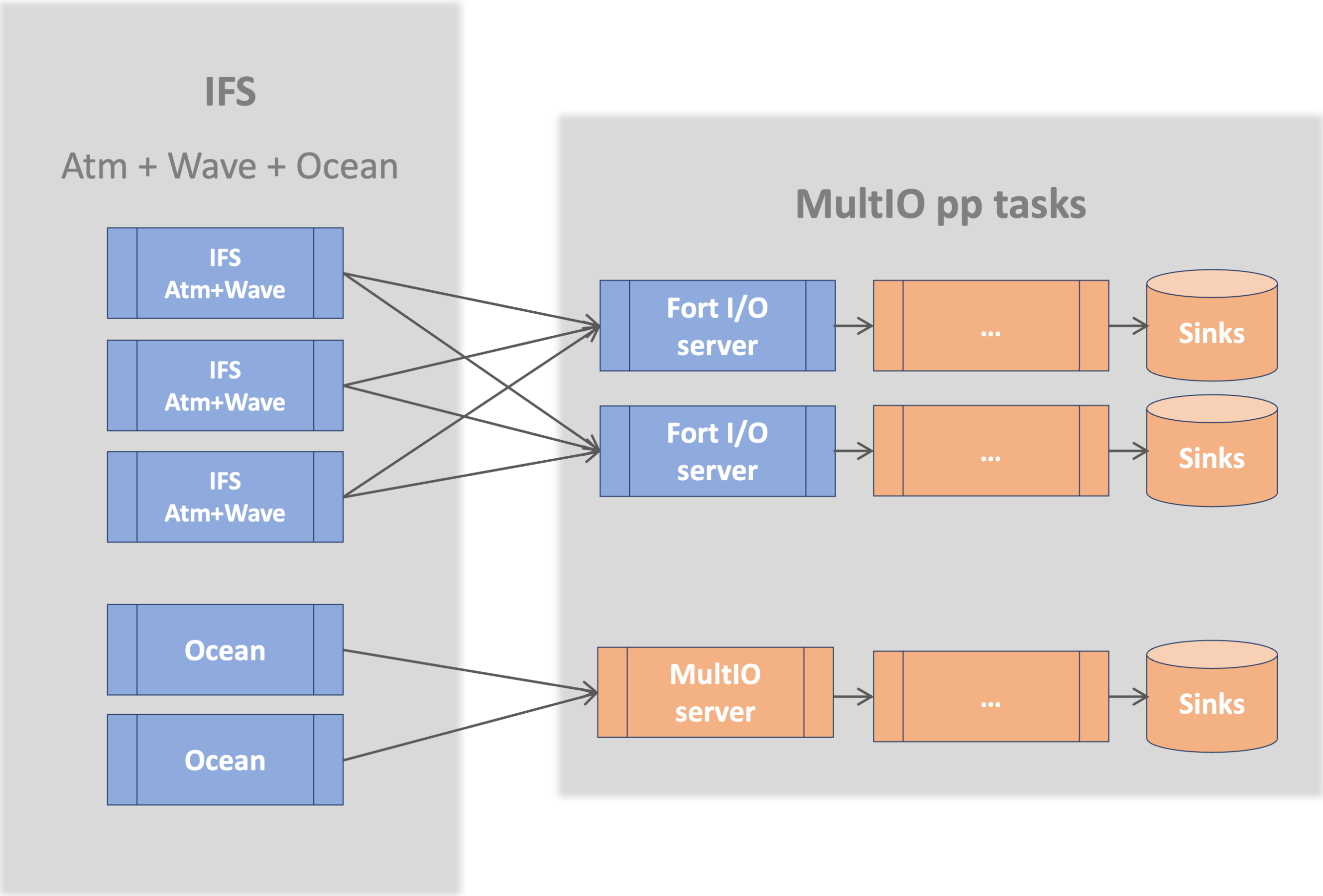
Usage 2: Destination Earth Phase 1 / NextGEMS

- **NextGEMS multi-year runs**
  - Coupled to NEMOv3 with no output
  - Post-processing pipeline for IFS
  - Statistics (monthly means)
  - Interpolation (re-gridding)
- **High-resolution DestinE climate runs**
  - Two coupled ocean-atmosphere models
    - IFS/NEMOv4
    - IFS/FESOMv2
  - I/O-server for both NEMOv4 and FESOMv2
  - Uniform output on HEALPix grids
  - Integrate output configuration

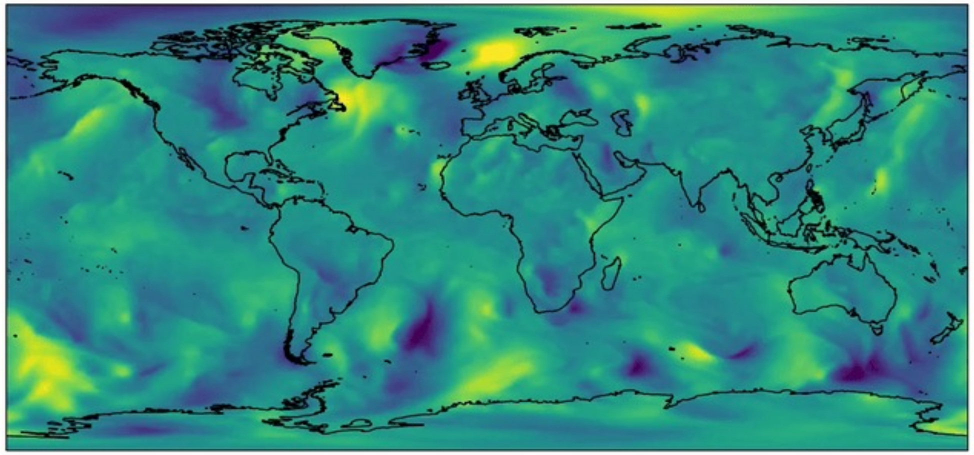


Usage 2: Destination Earth Phase 1 / NextGEMS

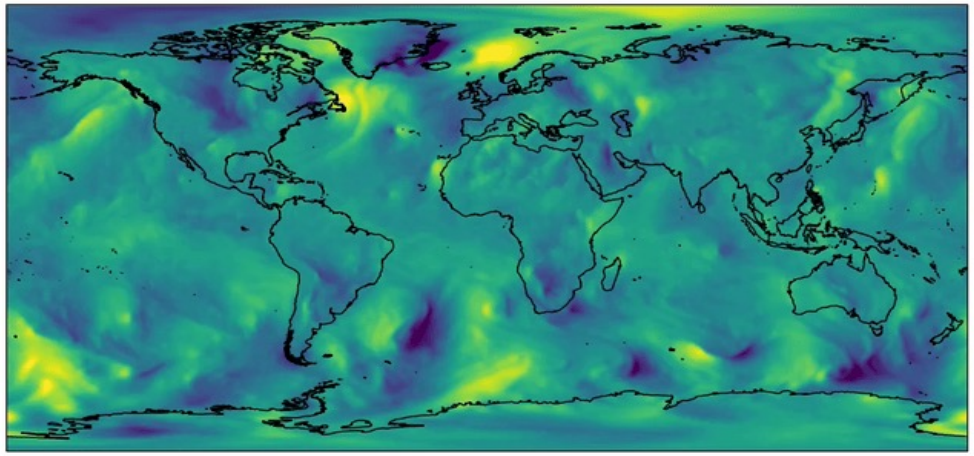
- **NextGEMS multi-year runs**
  - Coupled to NEMOv3 with no output
  - Post-processing pipeline for IFS
  - Statistics (monthly means)
  - Interpolation (re-gridding)
- **High-resolution DestinE climate runs**
  - Two coupled ocean-atmosphere models
    - IFS/NEMOv4
    - IFS/FESOMv2
  - I/O-server for both NEMOv4 and FESOMv2
  - Uniform output on HEALPix grids
  - Integrate output configuration



Current work and outlook



ML Model



The IFS

🤖 **Anticipation of AIFS**

- ○ Python interface for multio
- ○ Optimisations for more data to be processed

🌐 **Further developments for DestinE/operataions**

- ○ MultIO as I/O-server for IFS atmosphere and wave
- ○ Re-design GRIB2 encoding
- ○ Support of ERA6
  - Consolidate standard deviation
  - Consolidate synoptic means
  - Statistics checkpointing
- ○ More model-side post-processing (fullpos)

Messages to take home

*MultIO has programmable pipelines that allows **processing data closer to the model**, thus alleviating some of the burden on downstream users*

*MultIO provides an **asynchronous I/O-server** that will be first used in upcoming ECMWF's ocean re-analysis and Climate DT Phase 1 production*

*ECMWF is refactoring its IFS output and production stack to support **on-the-fly product generation** and MultIO as an I/O-server*