IMAS data processing

Daan van Vugt

IGNITIONCOMPUTING

iter
china eu india japan korea russia usa

Paulo Abreu

ignitioncomputing.com
dvanvugt@ignitioncomputing.com

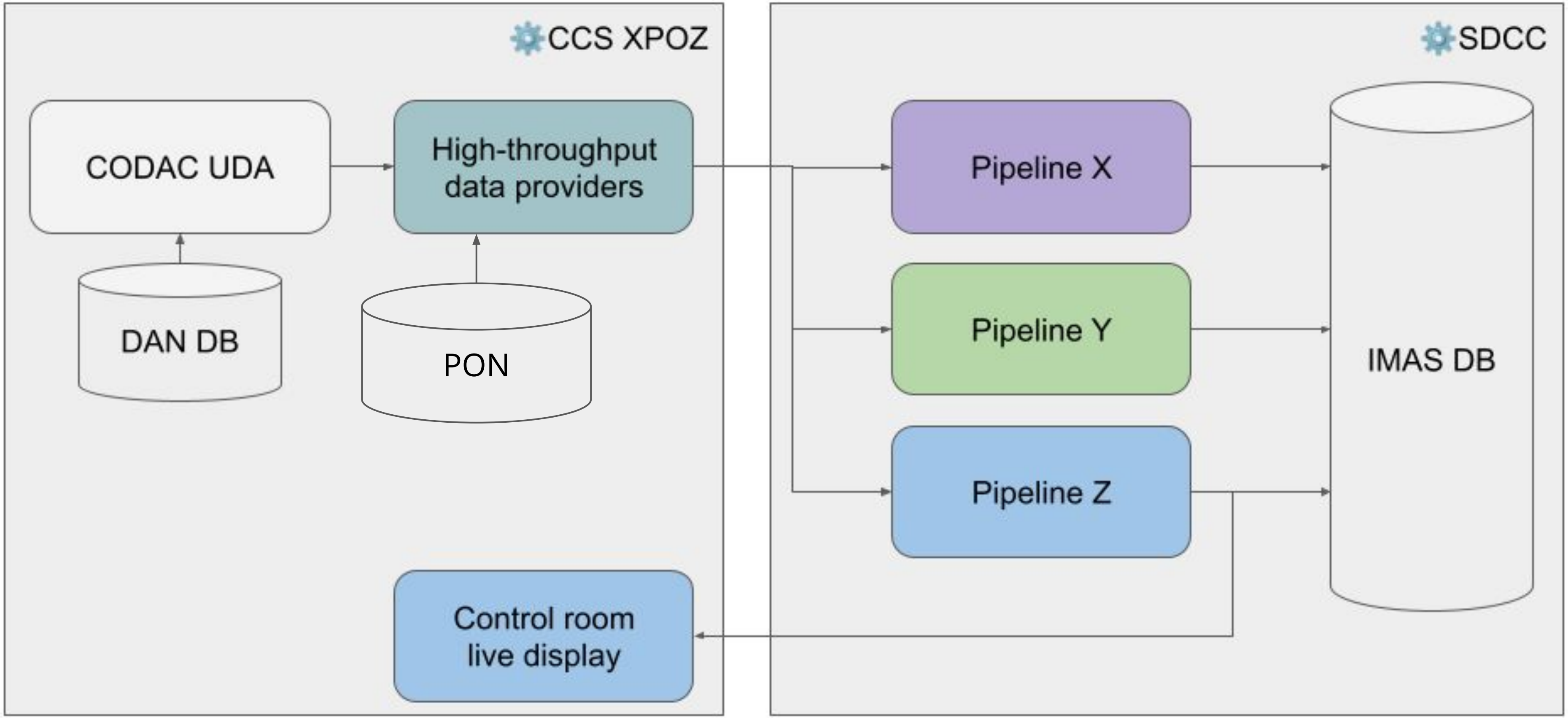
Agenda

- Data processing context and goals
- Architecture
- Mapping data
- Streaming data
- NetCDF IMAS format
- Interfaces

Goals for this project

- Prepare for live and full processing of pulse data
- Standardize processing pipeline description, codes
  - EFIT++ for example
- Store processing results in DB
- Render some processing output in control room
  - Resilience against slow codes, run on latest data
  - Bonus: run custom pipelines live

High-level Data Flow / Components per Compute Zone

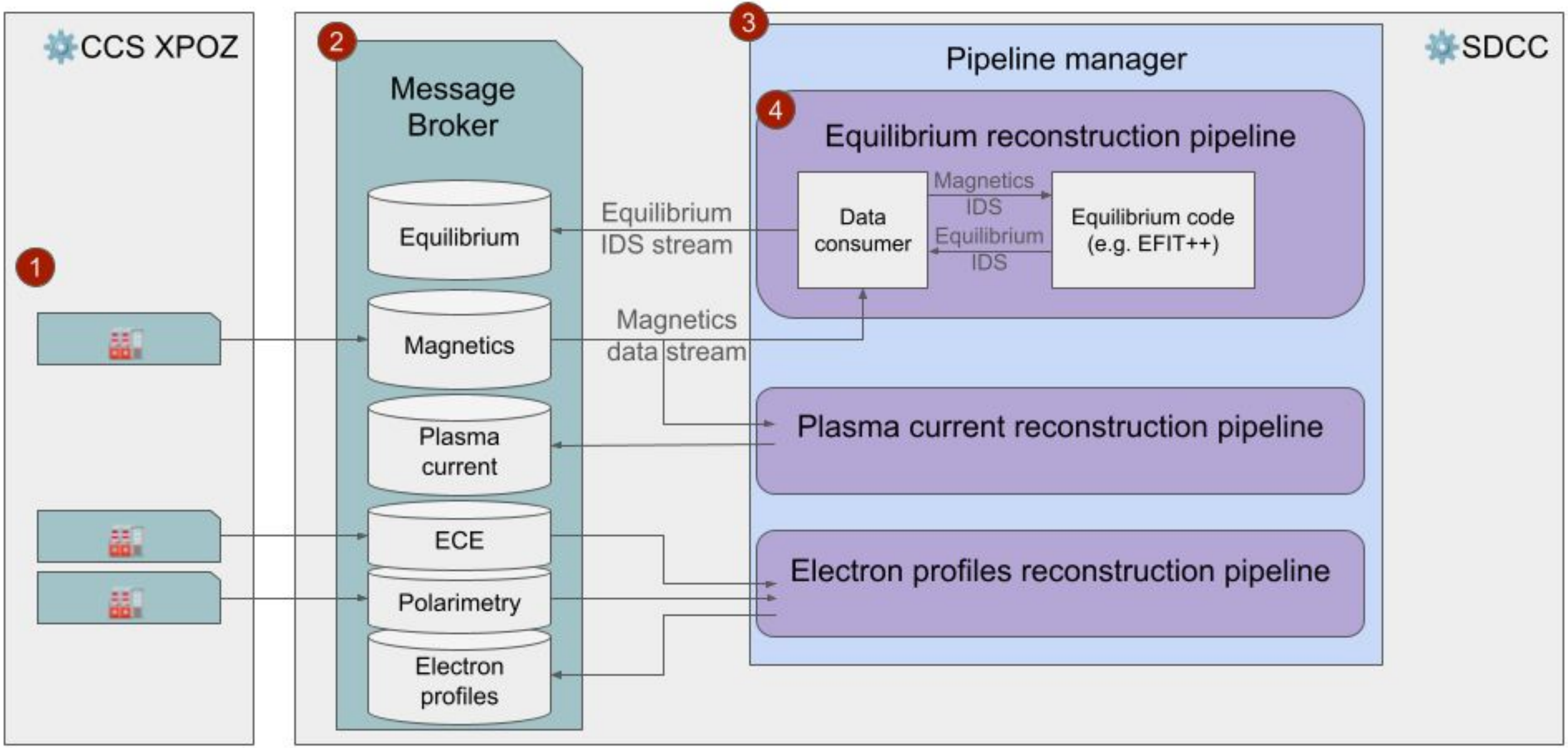


Data collection challenges

- Data arriving at different timebases
- High volume
- CODAC expects to need all FS bandwidth to store DAN experimental data
  - May not be allowed to read the files during a pulse
    - Also low fsync freq, leading to unnecessary latency
- Alternative would be to stream data from the CODAC writers
  - Also need to implement reader from disk
    - On-disk format with custom compression
  - CODAC may develop ZeroMQ approach
  - Consistency with disk reading replays

DAN2IMAS, PON2IMAS internal repos

Workflows / Processing Pipelines

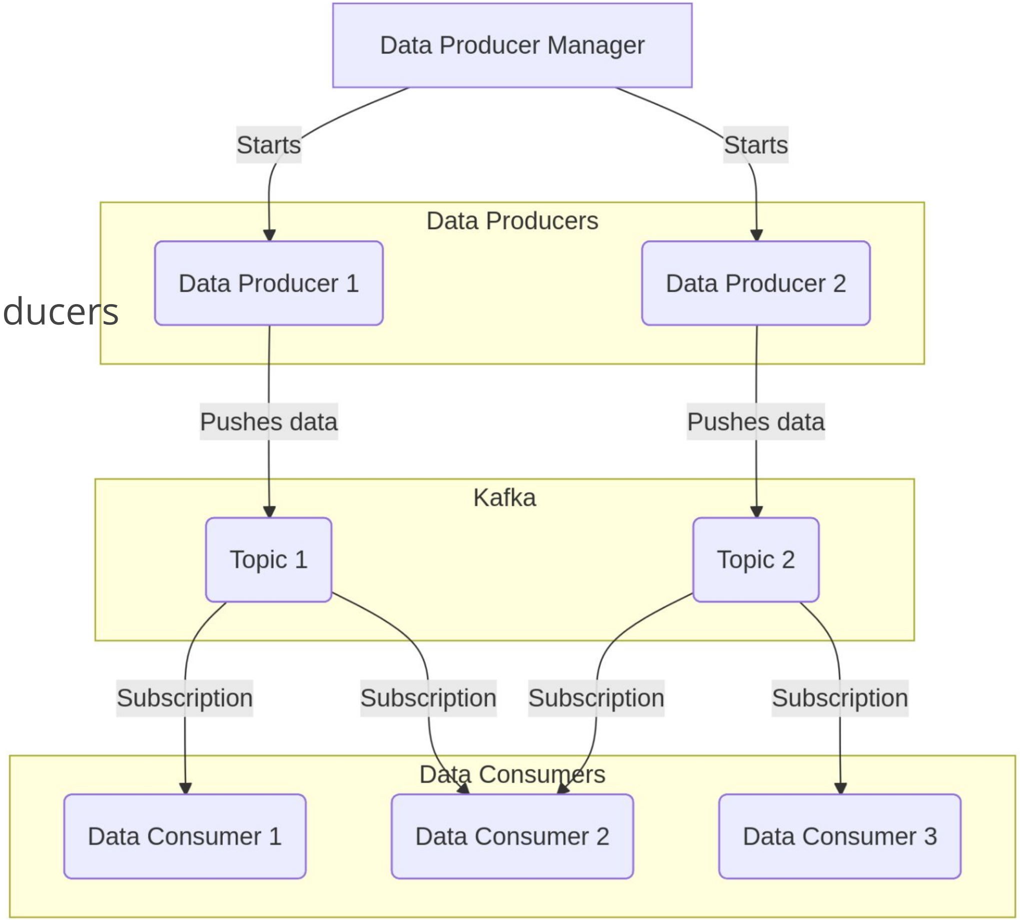


Architecture

- Kafka message broker
- Publish topics per IDS, start time & sampling frequency
  - Expect tool to request new streams and topics
  - Data publishers
    - DAN
    - PON
    - more?
- Data readers
  - Kafka
  - M3
- IDS transport, serialization

Kafka

- Data producers
  - Manager that starts all producers
- Kafka
- Data consumers



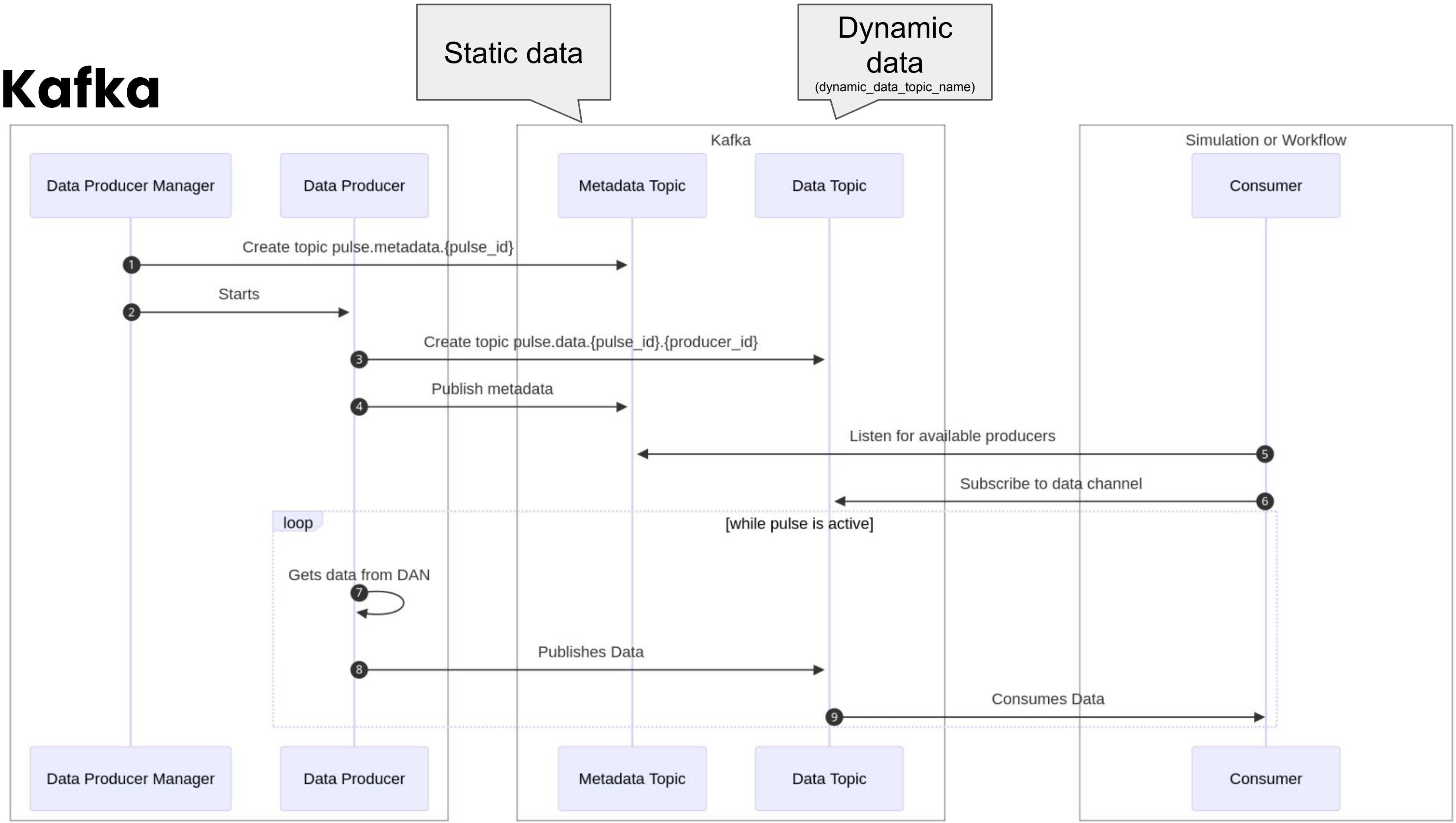
Kafka – Pulse topics

- Metadata topic
  - `live.metadata.{pulse_id}`
  - All data producers publish their metadata to this single topic
- Data topics
  - `live.data.{pulse_id}.{producer_id}`
  - Only a single producer exists for each data topic
- Replay
  - `replay.metadata.{pulse_id}.{replay_id}`
  - `replay.data.{pulse_id}.{replay_id}.{producer_id}`

`pulse_id` provided when starting producers (pulse number)

`producer_id` provided in producer configuration

Kafka


Static data
Dynamic data
(dynamic_data_topic_name)
Kafka
Simulation or Workflow
Data Producer Manager
Data Producer
Metadata Topic
Data Topic
Consumer
Create topic pulse.metadata.{pulse_id}
Starts
Create topic pulse.data.{pulse_id}.{producer_id}
Publish metadata
Listen for available producers
Subscribe to data channel
loop
[while pulse is active]
Gets data from DAN
Publishes Data
Consumes Data
Data Producer Manager
Data Producer
Metadata Topic
Data Topic
Consumer

https://github.com/iterorganization/IMAS-Streams

IMAS-Streams

- Much data in an IMAS time slice is static
- Serialize/deserialize unnecessarily repeats

- Split static data and dynamic data
- Only send data arrays for each time slice

- Limited to constant shapes
- Potential compatibility with video streaming (in future work)

Streaming IMAS data – metadata implementation

- JSON mapping
  - **metadata_version**: version string (“1.0”)
  - **dynamic_data_topic_name**
  - **sample_period_ns**: configured time period between two samples in nanoseconds
  - **pulse_id**
  - **signal_map**: a copy of the signal map that the data producer was configured with
  - **data_dictionary_version**
  - **ids_name**
  - **static_data**: serialized IDS (base64 encoded)
    - Relevant part of the machine description
  - **dynamic_data**: list of IDS paths to interpret dynamic data

Streaming IMAS data – dynamic data

- Array of doubles
  - IEEE-754 little-endian encoded

  - **dynamic_data** field in metadata explains how to interpret
    - First element is always “time”
    - For example:

```
[
    "time",
    "flux_loop[0]/flux/data",
    "flux_loop[0]/voltage/data",
    "flux_loop[1]/flux/data",
    "flux_loop[1]/voltage/data",
    ...
]
```

Streaming IMAS data – Producer & Reader

- Streaming data producer
  - Read DAN signal mapping
  - Read IMAS machine description
  - Filter machine description
    - E.g. keep mapped b_field_pol_probe[:], remove unmapped flux_loop[:]
  - Construct streaming IMAS metadata

- Streaming data reader
  - Read streaming IMAS metadata
  - Construct static data IDS object
  - Receive dynamic data
    - Apply and return (copy of) IDS object

> Default = return copy
> Allows users to do anything with the IDS
>
> Improved IMAS-Python performance to deepcopy IDSs (PR#65)

Performance test

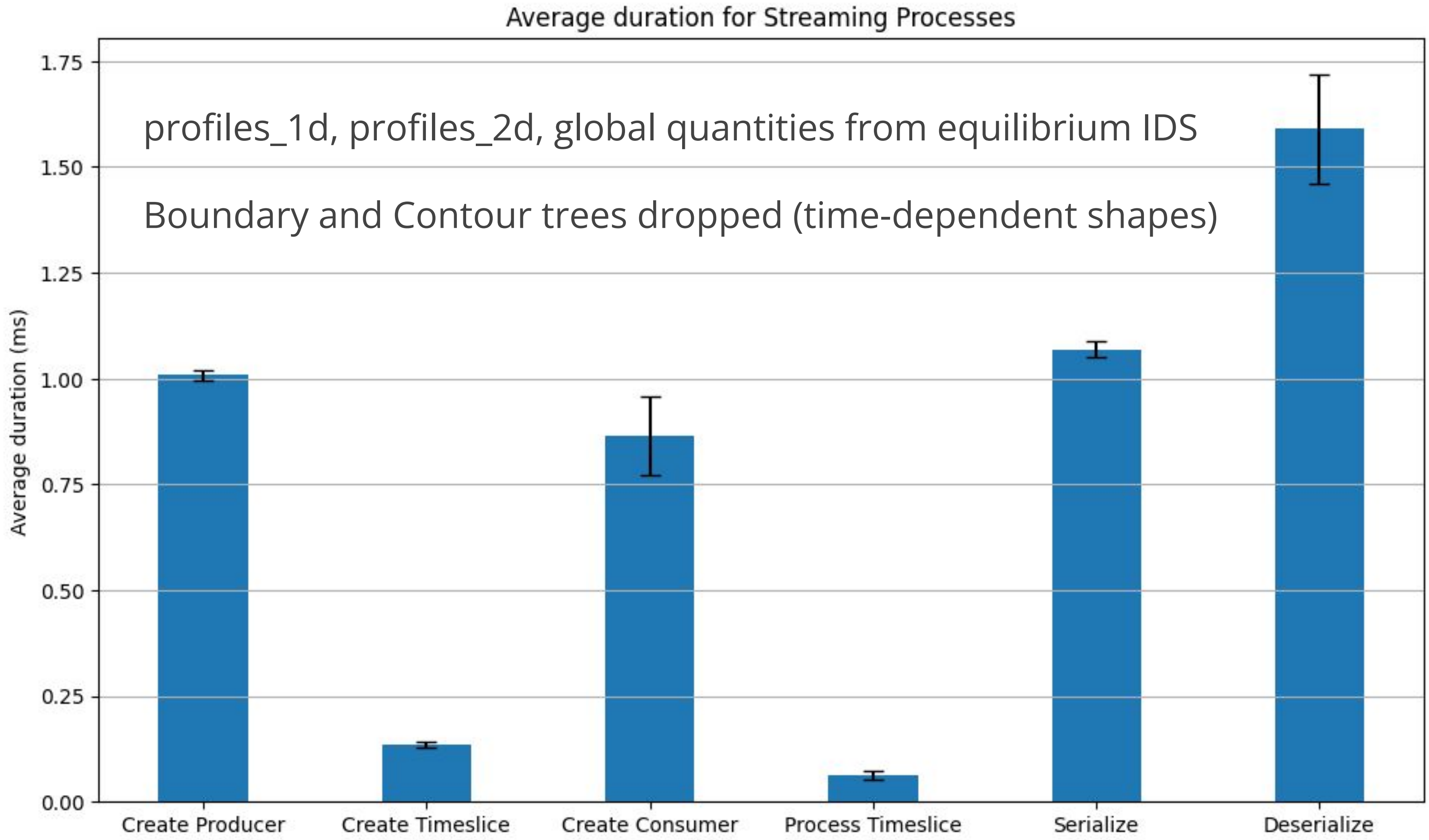
- ~ 350,000 messages, each with 1000 signals
- Store in AL HDF5 backend, IMAS_AL_DISABLE_VALIDATE=1
- ~Saturate network links

Case	Batch size	Duration	CPU time
Consume and discard	*N/A*	9.1 s	6.7 s
Consume and store	1	*DNF*	*DNF*
	100	125.8 s	121.5 s
	1,000	46.6 s	25.6 s
	10,000	46.8 s	16.0 s

Got to 18% in 33 min

Smaller IMAS HDF5 files for larger batches: more efficient chunking / compression

More time waiting for disk I/O


Average duration for Streaming Processes
profiles_1d, profiles_2d, global quantities from equilibrium IDS
Boundary and Contour trees dropped (time-dependent shapes)
Average duration (ms)
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Create Producer
Create Timeslice
Create Consumer
Process Timeslice
Serialize
Deserialize

https://github.com/iterorganization/IMAS-ITER-Mapping

IMAS-ITER-Mapping

- Basic signal mapping
  - Not processing!
- Special-case now
  - DAN/PON, scalars
- (strict)YAML file
- Complex stuff in later steps of pipeline
- Could be replaced later


```yaml
description: Example mapping file for the README
data_dictionary_version: 4.0.0
machine_description_uri: iter_md_magnetics_150100_5.nc
target_ids: magnetics
signals:
  flux_loop:
  - name: 55.AD.00-MSA-1001
    flux/data: AAA-BBBB-CCCC-DDD-EEEE:FF1001-XI [Wb]
    voltage/data: AAA-BBBB-CCCC-DDD-EEEE:FF1001-XP [V]
  - name: 55.AD.00-MSA-1002
    flux/data: AAA-BBBB-CCCC-DDD-EEEE:FF1101-XI [Wb]
    voltage/data: AAA-BBBB-CCCC-DDD-EEEE:FF1101-XP [V]
```

https://github.com/iterorganization/IMAS-ITER-Mapping/blob/main/docs/mapping.md

Mapping file features

- Map diagnostic signals to the IMAS Data Dictionary.
- Each mapping file applies to a single diagnostic IDS (Interface Data Structure).
- Static data, such as machine geometries, are provided in IDS format in a Machine Description IDS. This keeps the mapping files small and to-the-point.
- Indicate units of the source signals, to allow automatic conversion to the units specified in the IMAS Data Dictionary (only linear, no dBW - W)

CLI

- `imas-iter-mapping validate MAPPING_FILE`

```
(.venv) maarten@maarten:~/projects/iter-ipdp$ imas-iter-mapping validate 1000-signals-test-mapping.yml
Validating "1000-signals-test-mapping.yml" ...
Success: 1000-signals-test-mapping.yml is a valid IMAS ITER Mapping file
(.venv) maarten@maarten:~/projects/iter-ipdp$ imas-iter-mapping validate 1000-signals-test-mapping.error.yml
Validating "1000-signals-test-mapping.error.yml" ...
ValidationError: Unit [Wb] is incompatible with the IMAS Data Dictionary units [T]
  in "1000-signals-test-mapping.error.yml", line 12:
        field/data: CWS-SCSU-CC2A-WCC-WPU2:DP2151-XI0 [Wb]
```

- `imas-iter-mapping describe MAPPING_FILE`

```
(.venv) maarten@maarten:~/projects/iter-ipdp$ imas-iter-mapping describe 1000-signals-test-mapping.yml
IMAS-ITER-Mapping file "1000-signals-test-mapping.yml" maps 1000 signals to the magnetics IDS.

The Machine Description contains 4 channel types:
- 'flux_loop' has 261 elements, of which 0 (0%) have mapped signals.
- 'b_field_pol_probe' has 931 elements, of which 500 (54%) have mapped signals.
  1000 signals are mapped. That is, on average, 2 signals per element.
  0 signals (0%) have a unit that differs from the IMAS Data Dictionary (but can be transformed).
- 'b_field_phi_probe' has 45 elements, of which 0 (0%) have mapped signals.
- 'rogowski_coil' has 341 elements, of which 0 (0%) have mapped signals.
```

https://imas-python.readthedocs.io/en/stable/netcdf.html

IMAS NetCDF file format

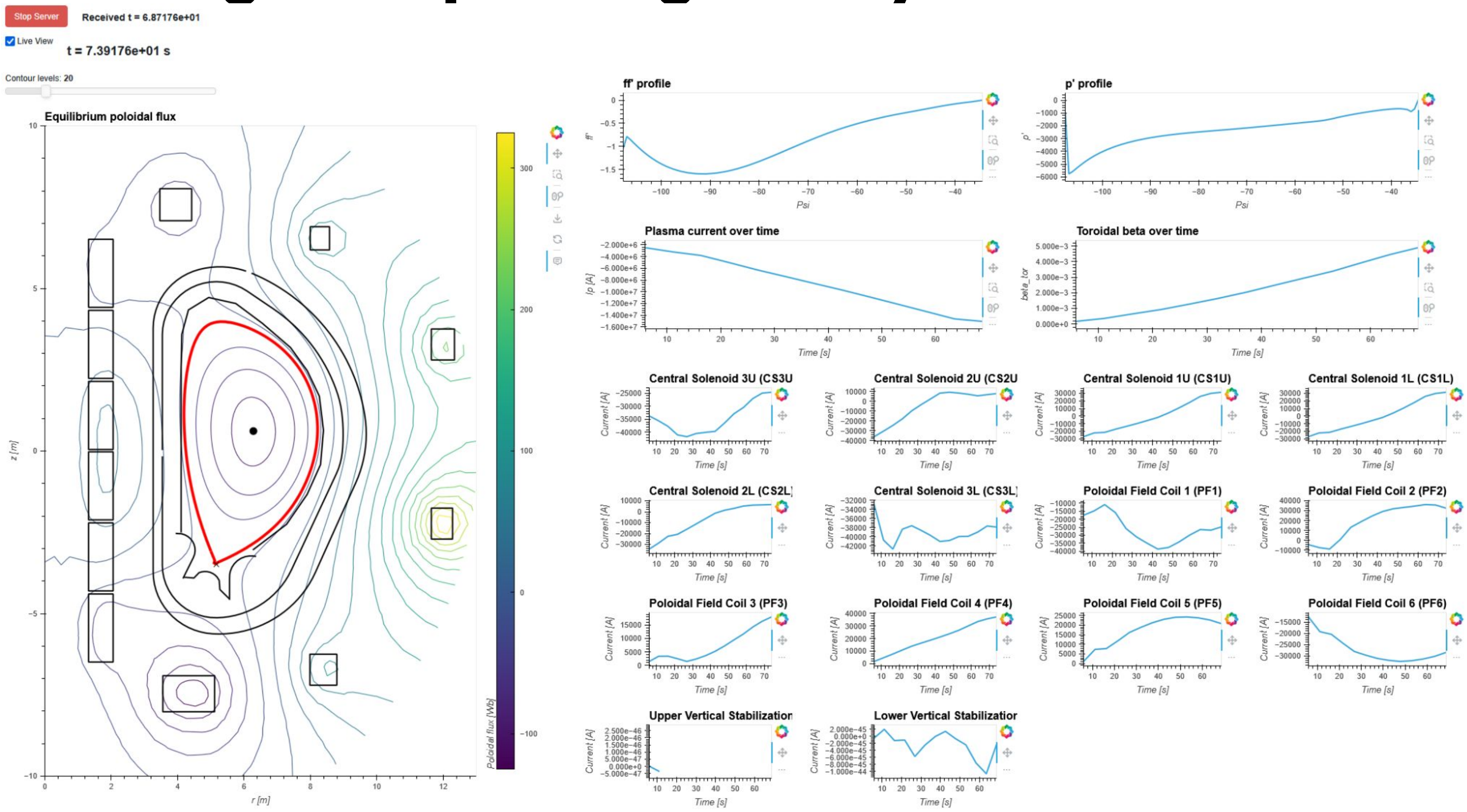
- Flattened, annotated, standardized ndarrays
  - S3-compatible
  - Conventions here: https://imas-python.readthedocs.io/en/stable/netcdf/conventions.html#imas-conventions-for-the-netcdf-data-format
  - Proposal on extending geometry descriptions: https://github.com/iterorganization/Fusion-Conventions/blob/main/docs/Geometry_Descriptions_for_Fusion_Conventions.md
- Write, read with IMAS-Python, xarray, fsspec
  - No IMAS-Core writing, slice writing support yet
- Let's talk about xarray vs awkwardarrays?

NetCDF native IMAS readers

- Goal: directly read NetCDF IMAS files from Fortran, C++, MATLAB
  - Hope: more performance, faster compile times
  - Stretch goal: demonstrate efficient reading over HTTP transport (UDA alternative?)
- Method:
  - Native ncread APIs with limited helpers for DD interpretation
  - No AL-Core, HLI
- Release Q2 2026

https://imas-muscle3.readthedocs.io/en/latest/

Streaming data plotting library



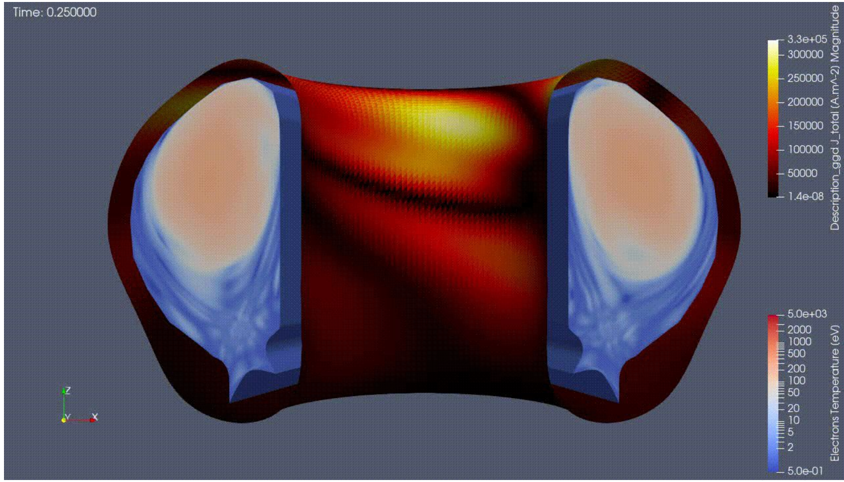
https://imas-paraview.readthedocs.io/en/latest/

IMAS-ParaView

- Tool for converting IMAS IDS to VTK

- ParaView plugins for visualizing various IMAS data
  - Generalized Grid Descriptions (GGDs)
  - 1D/2D Profiles
  - Many other Machine Description structures

- CLI tool available
  - GGD → VTK
  - VTK → GGD



Current in vacuum vessel wall, and electron temperature of a JOREK simulation visualized with IMAS-ParaView https://github.com/iterorganization/imas-standard-interfaces

IMAS interfaces

- Sometimes too many options to store data in IMAS
- Encode some necessary conditions for code/data interoperability
    - IDS variable requiredness, optionality
    - Use IMAS-validator for stricter checks
    - Whole interface far too complex. Use LLM to read interface code instead?
- Check incompatibility upfront
    - In M3 workflow documentation pages
    - Checked reads
- Data reduction before sending

https://imas-muscle3.readthedocs.io/en/latest/actor_olc.html

Operating Limits & Conditions checking

```python
@validator("pf_active")
def validate_force_limits_pf(ids):
    """Validate forces in poloidal field coils are within operational limits"""
    for coil in ids.coil:
        for key in pf_force_limits.keys():
            if key in coil.name:
                if coil.force_vertical.data.value:
                    assert coil.force_vertical.data.value < pf_force_limits[key][0]
                    assert coil.force_vertical.data.value > pf_force_limits[key][1]
```

- **halt_on_error**: (bool) Whether or not the simulation should be forcibly stopped when a validation test fails. Defaults to False.
- **extra_rule_dirs**: (str) The rule directories in which to look for IMAS-Validator rulesets. If inserting multiple, split them with a ';'. Defaults to ''.
- **rulesets**: (str) The names of rulesets to run in the found rule directories. If inserting multiple, split them with a ';'. Defaults to 'PDS-OLC'.
- **apply-generic**: (bool) Whether or not to apply the generic bundled validation tests. Defaults to True.

https://github.com/iterorganization/IMAS-Validator

Let’s talk about

- IMAS software stack
  - IMAS-Python
  - IMAS-Validator
  - IMAS-M3-visualisation
  - IMAS-ParaView
  - Performance
  - …
- Data processing pipelines, mapping libraries
- NetCDF, xarray, cloud data
- Interfaces
- …

Thanks to Paulo Abreu, Olivier Hoenen, Simon McIntosh, and more @ ITER

Work done by IC team

ℹ️ **Would you like to collaborate? Let's talk!**

ignitioncomputing.com
dvanvugt@ignitioncomputing.com