KaraboHDF5
Storing Scientific Data at the European XFEL

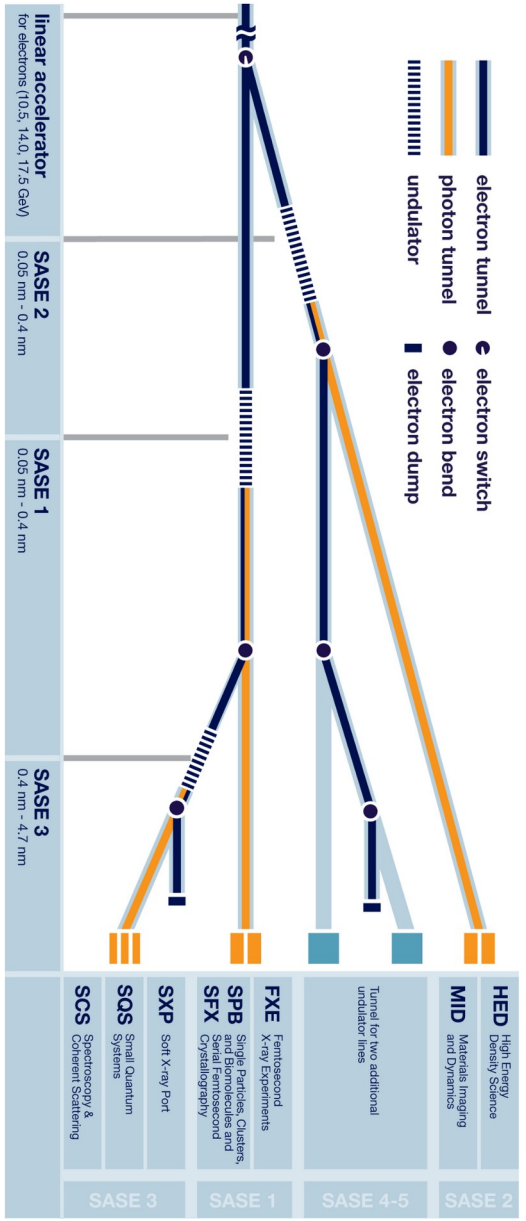
Jose Luis Vázquez-García
Data Department
Controls Software Engineer

Hamburg, May 26, 2025

The European XFEL

X-ray Free Electron Laser

- 3 SASE[1] lines, 7 instruments
- Characteristics
  - High volume of generated data (~PB/day)
  - High speed of data generation (~15 GB/s)



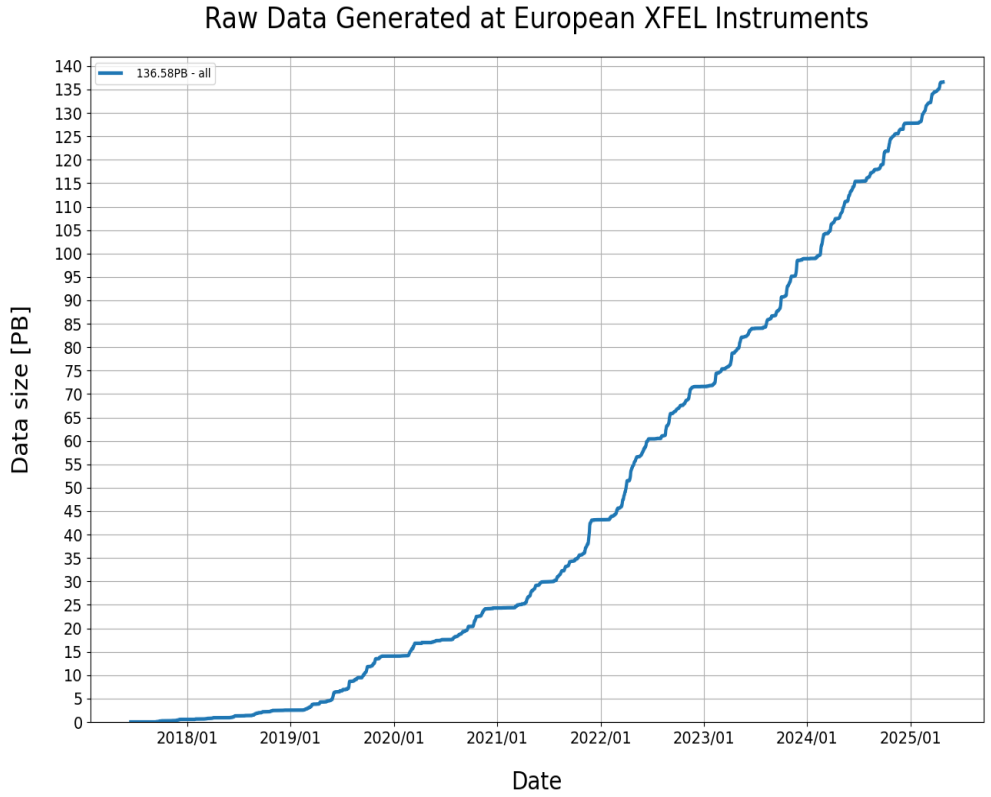
[1] *Self-amplified Spontaneous Emission*

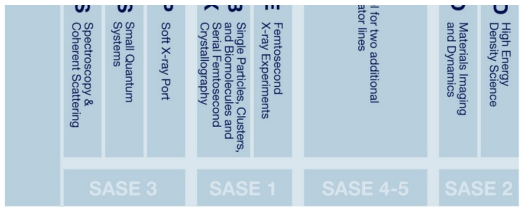
The European XFEL

X-ray Free Electron Laser

- 3 SASE[1] lines, 7 instruments
- Characteristics
  - High volume of generated data (~PB/day)
  - High speed of data generation (~15 GB/s)
- Problems:
  - Storage hardware cannot be expanded forever.
  - Both hardware and software must be able to handle high speed of data generation

[1] *Self-amplified Spontaneous Emission*

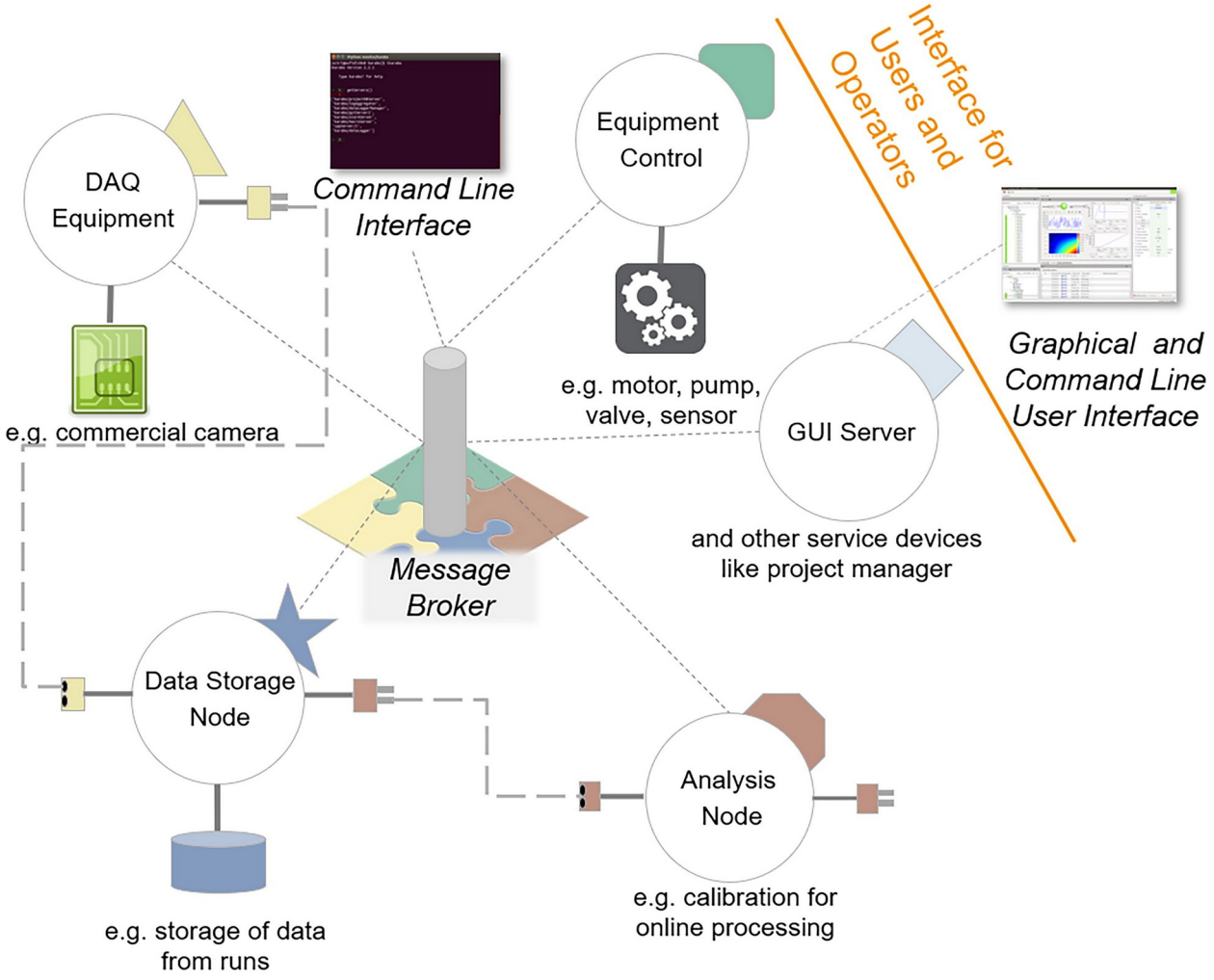




The control system: *Karabo (i)*

SCADA system, written from scratch in C++ and Python.

- Event-driven, fault-tolerant, highly distributed.
- Modular:
  - ***Karabo devices*** make up the system.
  - ***Karabo Hash***: flexible data structure to store and communicate data between devices.



The control system: *Karabo (ii)*

The **Hash**:

- De facto karabo standard data structure to store and transmit data.
- **Flexible hierarchical** container.
- ***Keyword – value*** container.
- Values can be almost any type, including other hashes.
- **Attributes** (keyword-value pairs) can be attached to values.
- Provides a **tree-like** interface to data.

```
In [1]: from karabo.middlelayer import Hash

In [2]: h = Hash('detector', 'AGIPD',
   ...:          'geometry', [2, 128, 256])

In [3]: print(h)
<detector{}: 'AGIPD', geometry{}: [2, 128, 256]>
```

The control system: *Karabo (ii)*

The **Hash**:

- De facto karabo standard data structure to store and transmit data.
- **Flexible hierarchical** container.
- ***Keyword – value*** container.
- Values can be almost any type, including other hashes.
- **Attributes** (keyword-value pairs) can be attached to values.
- Provides a **tree-like** interface to data.

```
In [1]: from karabo.middlelayer import Hash

In [2]: h = Hash('detector', 'AGIPD',
   ...:          'geometry', [2, 128, 256])

In [3]: print(h)
<detector{}: 'AGIPD', geometry{}: [2, 128, 256]>

In [4]: attributes = {'timeStamp': 12345,
   ...:               'version': '1.0'}

In [5]: value = Hash('language', 'Python',
   ...:              'version', '3.12')
```

The control system: *Karabo (ii)*

The **Hash**:

- De facto karabo standard data structure to store and transmit data.
- **Flexible hierarchical** container.
- ***Keyword – value*** container.
- Values can be almost any type, including other hashes.
- **Attributes** (keyword-value pairs) can be attached to values.
- Provides a **tree-like** interface to data.

```
In [1]: from karabo.middlelayer import Hash

In [2]: h = Hash('detector', 'AGIPD',
   ...:          'geometry', [2, 128, 256])

In [3]: print(h)
<detector{}: 'AGIPD', geometry{}: [2, 128, 256]>

In [4]: attributes = {'timeStamp': 12345,
   ...:               'version': '1.0'}

In [5]: value = Hash('language', 'Python',
   ...:              'version', '3.12')

In [6]: h.setElement(key = 'api',
   ...:              value = value,
   ...:              attrs = attributes)

In [7]: print(h)
<detector{}: 'AGIPD', geometry{}: [2, 128, 256], api{'timeStamp': 12345, 'version': '1.0'}: <
language{}: 'Python' => String
version{}: '3.12' => String
>>
```

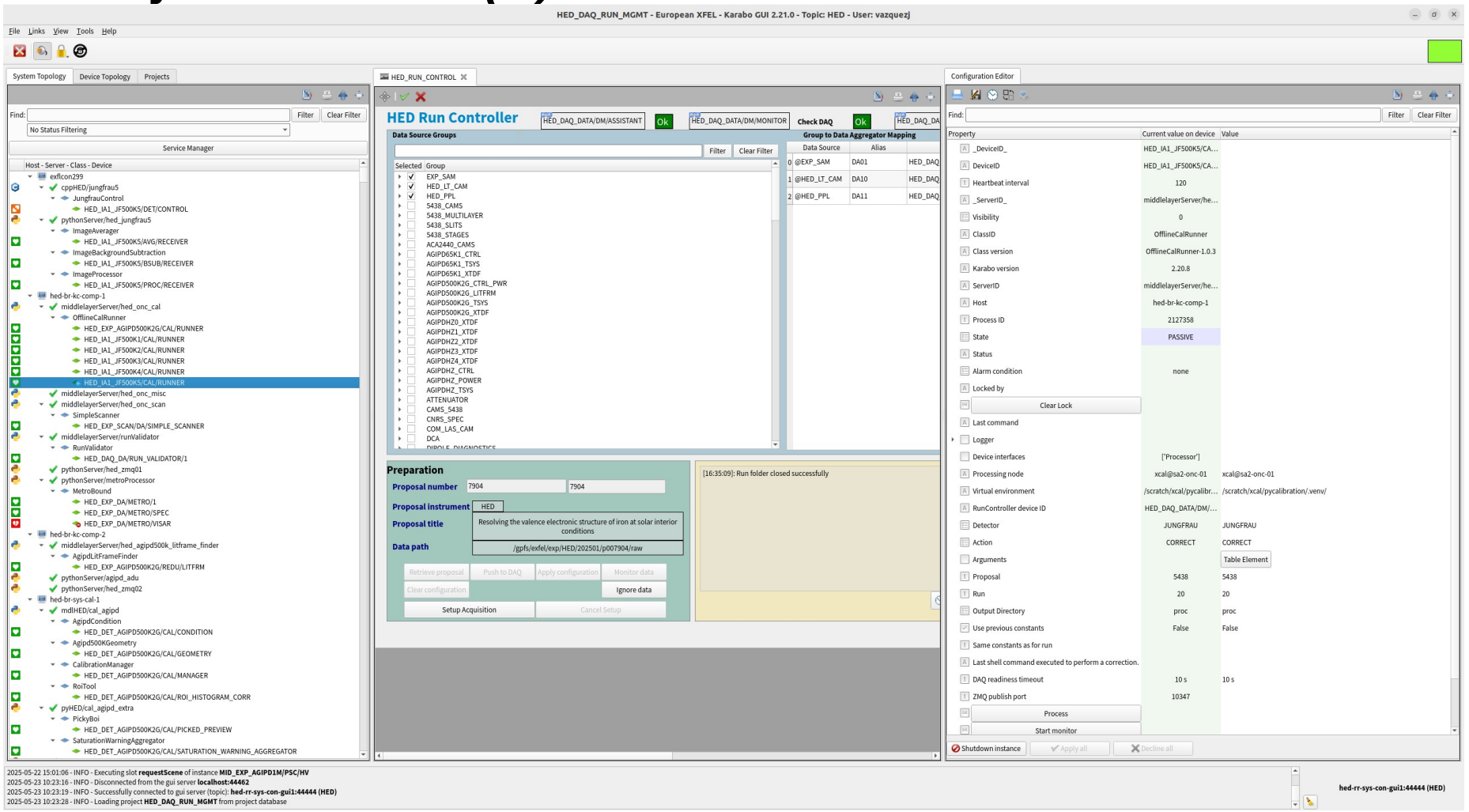
The control system: *Karabo (ii)*

The **Hash**:

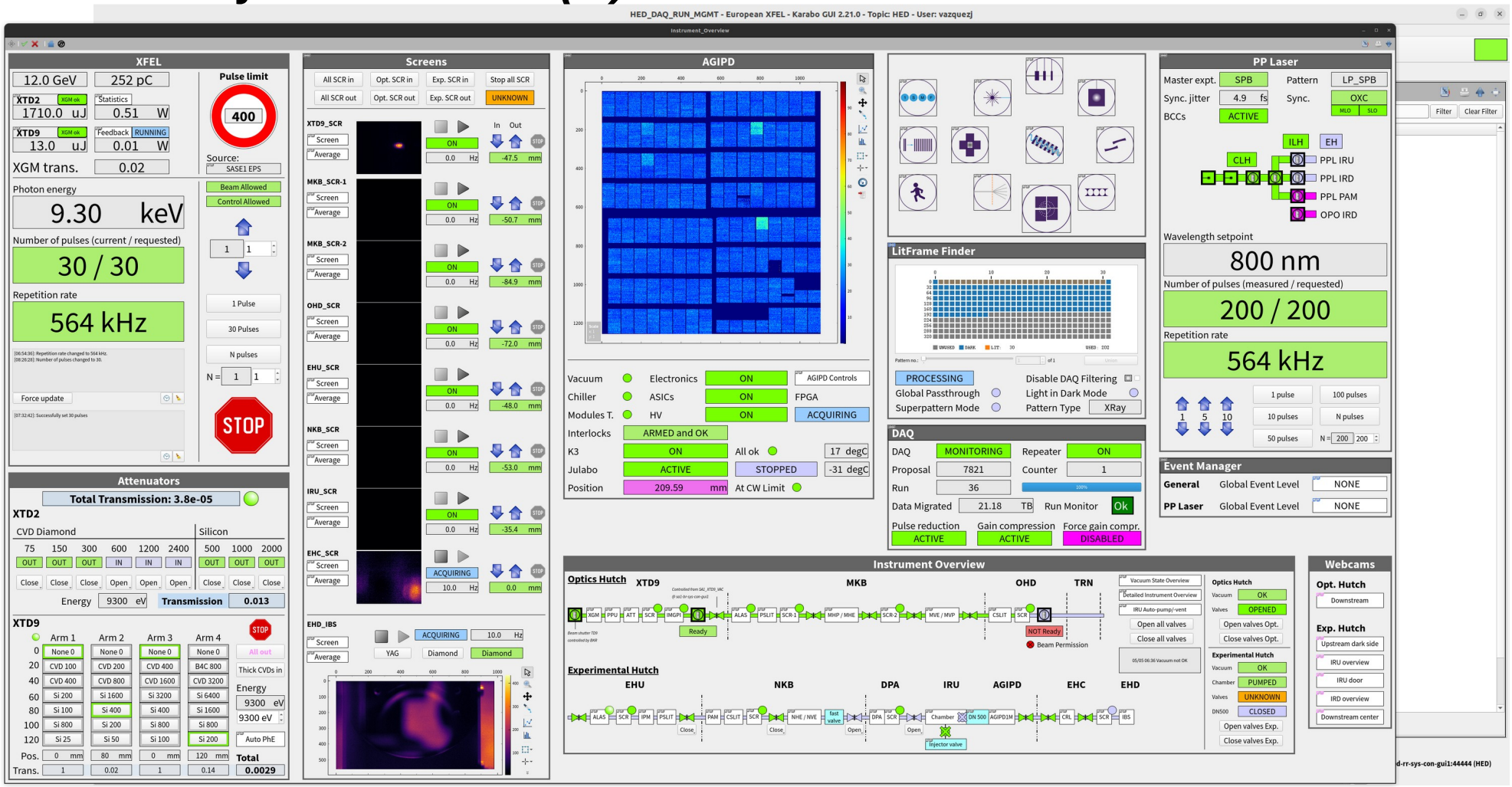
- De facto karabo standard data structure to store and transmit data.
- **Flexible hierarchical** container.
- ***Keyword – value*** container.
- Values can be almost any type, including other hashes.
- **Attributes** (keyword-value pairs) can be attached to values.
- Provides a **tree-like** interface to data.

```
In [1]: from karabo.middlelayer import Hash

In [2]: h = Hash('detector', 'AGIPD',
   ...:          'geometry', [2, 128, 256])

In [3]: print(h)
<detector{}: 'AGIPD', geometry{}: [2, 128, 256]>

In [4]: attributes = {'timeStamp': 12345,
   ...:               'version': '1.0'}

In [5]: value = Hash('language', 'Python',
   ...:              'version', '3.12')

In [6]: h.setElement(key = 'api',
   ...:              value = value,
   ...:              attrs = attributes)

In [7]: print(h)
<detector{}: 'AGIPD', geometry{}: [2, 128, 256], api{'timeStamp': 12345, 'version': '1.0'}: <
language{}: 'Python' => String
version{}: '3.12' => String
>>

In [8]: print(h['detector'])
AGIPD

In [9]: print(h['api'])
<language{}: 'Python', version{}: '3.12'>

In [10]: print(h['api.language'])
Python
```

The control system: *Karabo (iii)*



The control system: *Karabo (iii)*

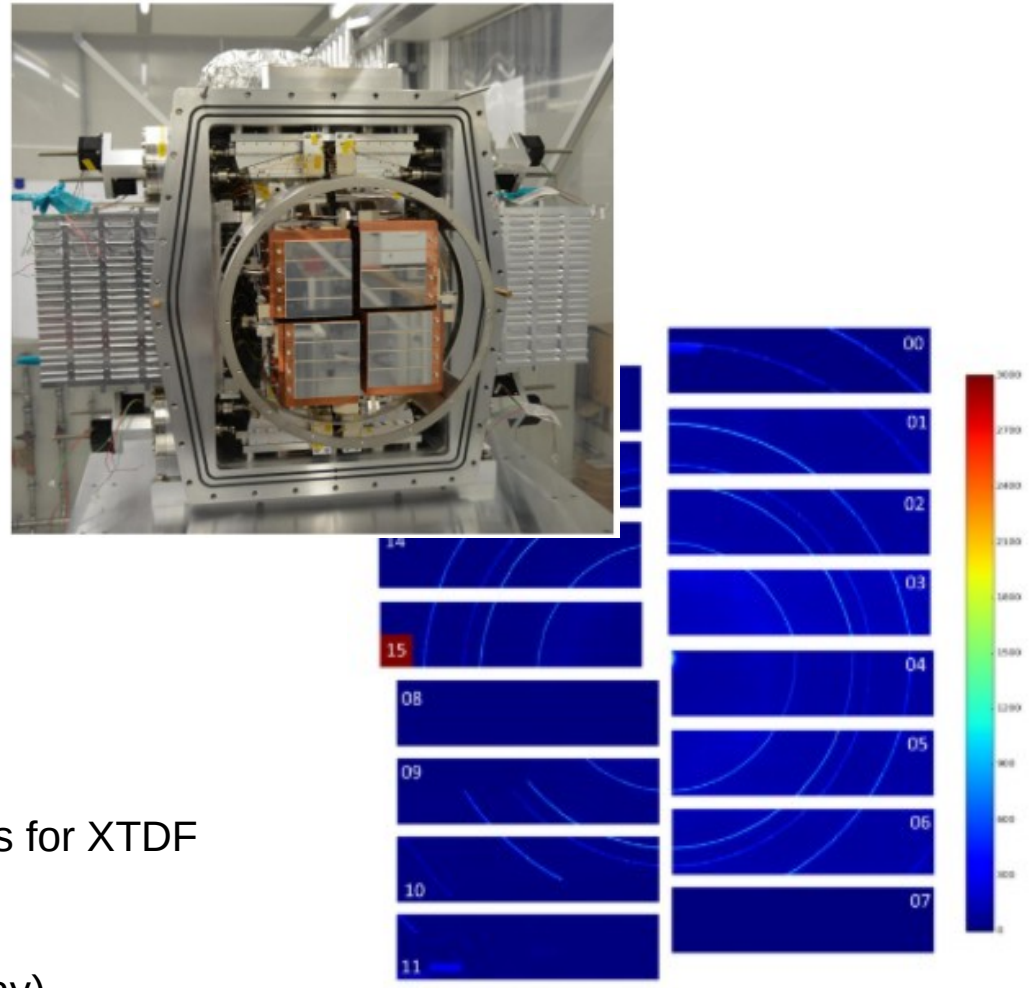


Data Generation (i)

Data sources

- Fast 2D sources:
  - **AGIPD**
  - **LPD**
  - **MSSC**
- Other imagers:
  - **Jungfrau**
  - **Gotthard**
  - etc...

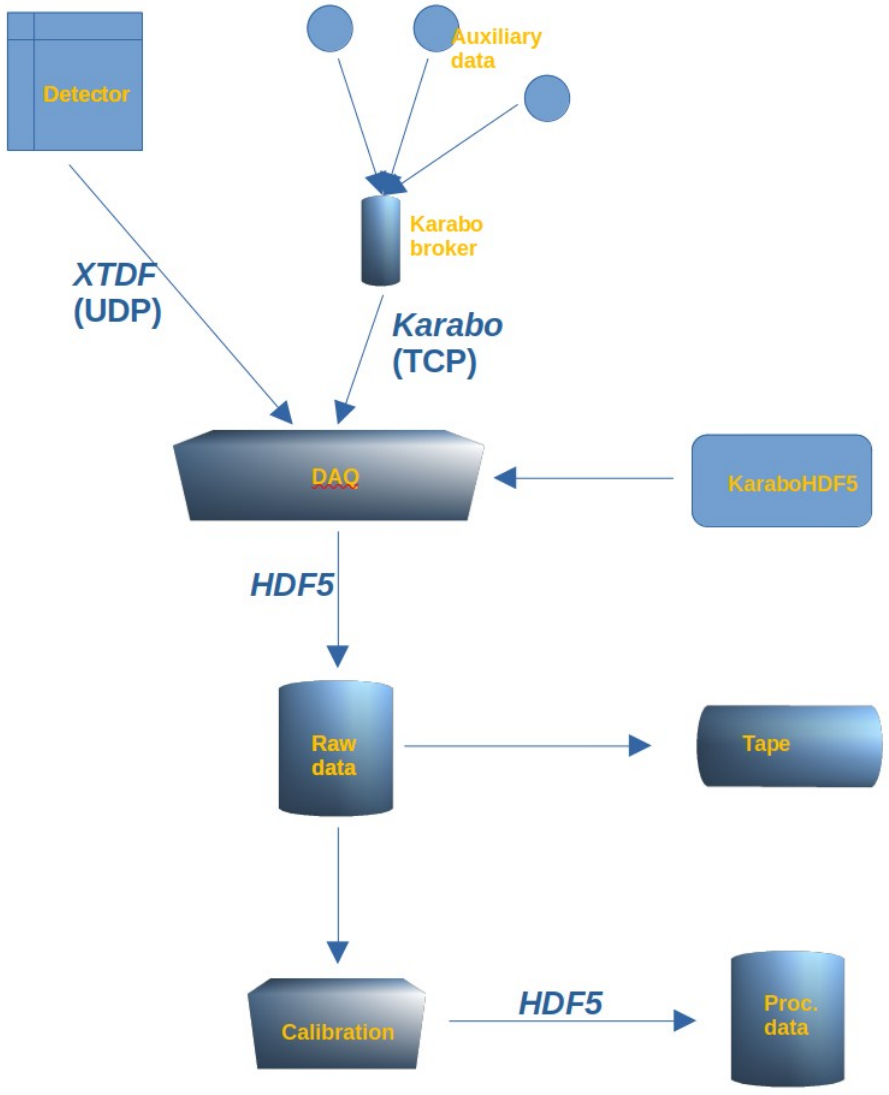
- Auxiliary data:
  - Digitizers
  - Motors
  - etc…
- Data characteristics:
  - **Fast** (up to 15 Gbps for XTDF imagers)
  - **Size** (several PB/day)



AGIPD Imager (front) and captured image shown in Karabo (back)

Data Generation (iii)

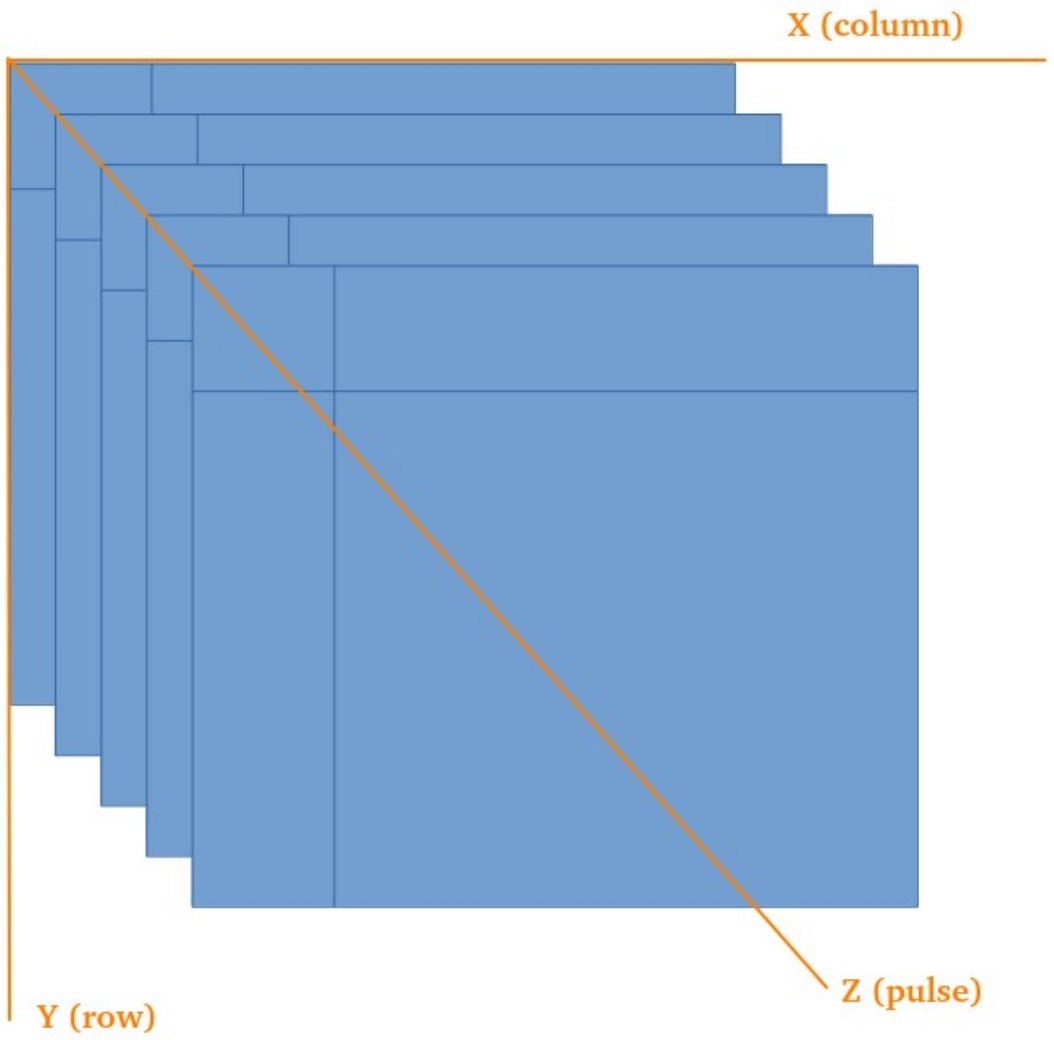
Many data formats must be handled



Data Reduction: *when* storing the data

Data compression

- Preliminary tests with data compressed with zlib + HDF5 did not provide enough storage speed.
- **HDF5 compression + chunking** fits the requirements:
  - On acquisition start, create a file template with chunks along the 3rd axis (pulses).
  - Chunks with no data will not be written.

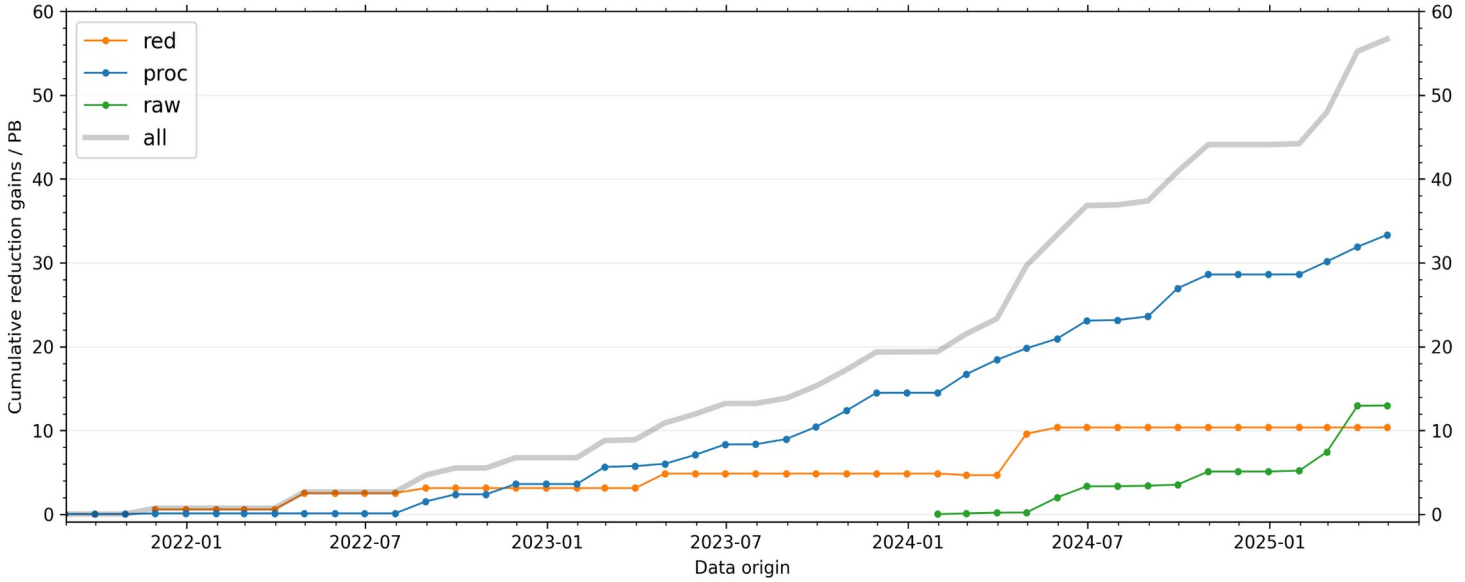


Imager data in XTDF format

Data Reduction: *when* storing the data

Data compression

- Preliminary tests with data compressed with zlib + HDF5 did not provide enough storage speed.
- **HDF5 compression + chunking** fits the requirements:
  - On acquisition start, create a file template with chunks along the 3rd axis (pulses).
  - Chunks with no data will not be written.

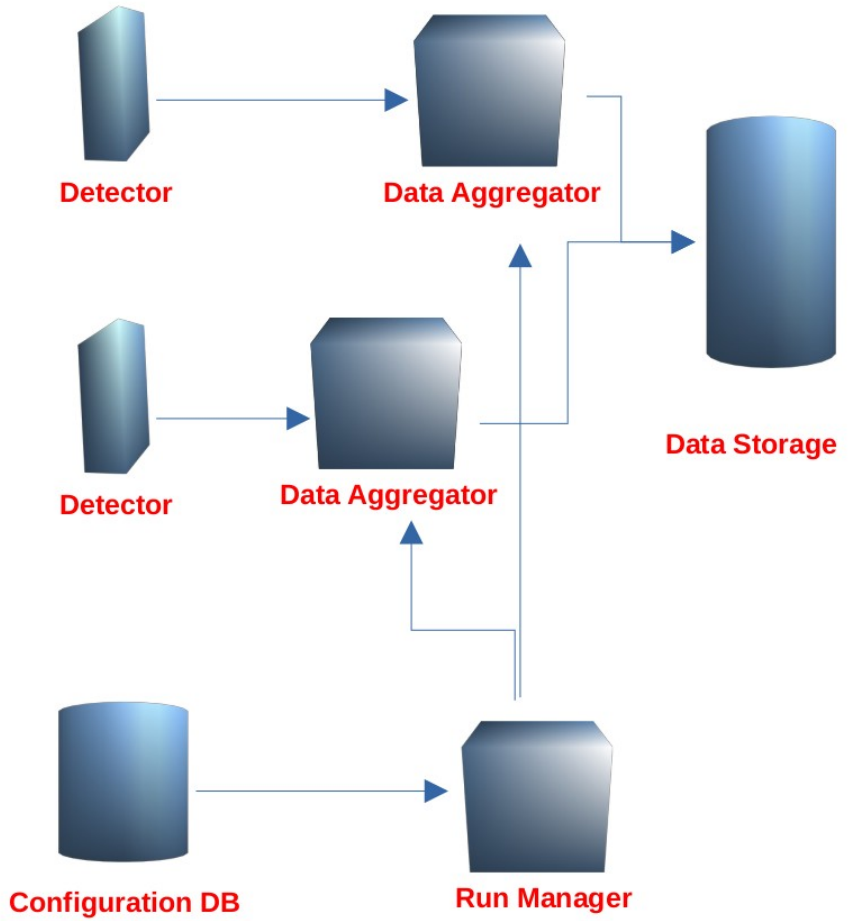


The DAQ (i)

Data Acquisition System: **Detectors** + **Hardware** + **Karabo devices**

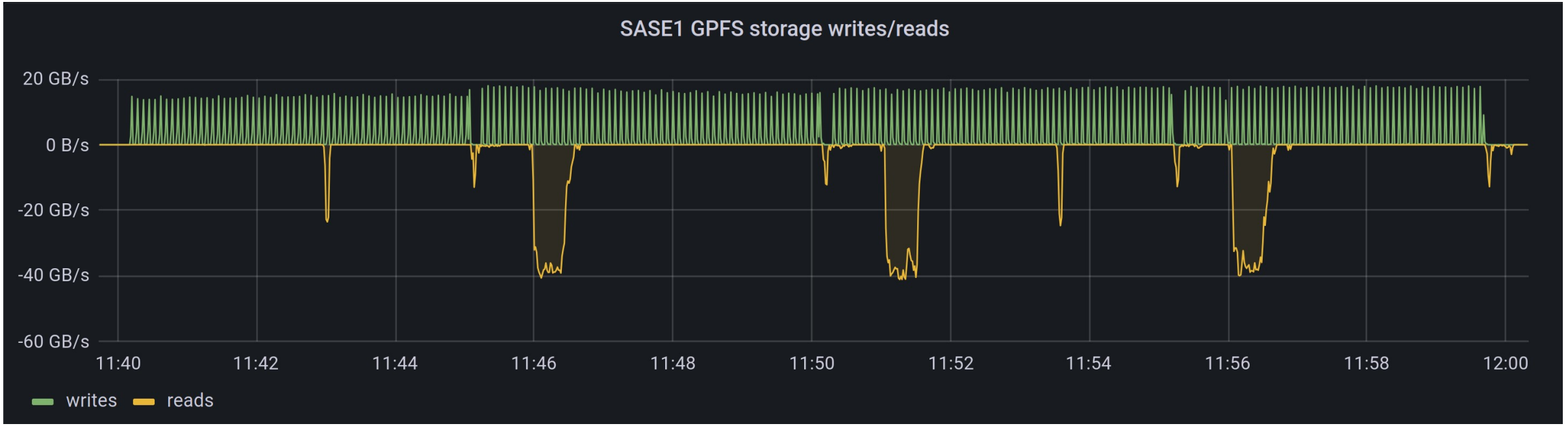
Data Aggregators:

- A Data Aggregator is a *Karabo device*
- Each aggregator gathers data from a given set of sources.
- Stores its data in one HDF5 file (one file per aggregator).
- **Real-time data recording** is possible, by storing data in several HDF5 files in parallel.
- The DAQ can be scalated, by adding more aggregators.



The DAQ (ii)

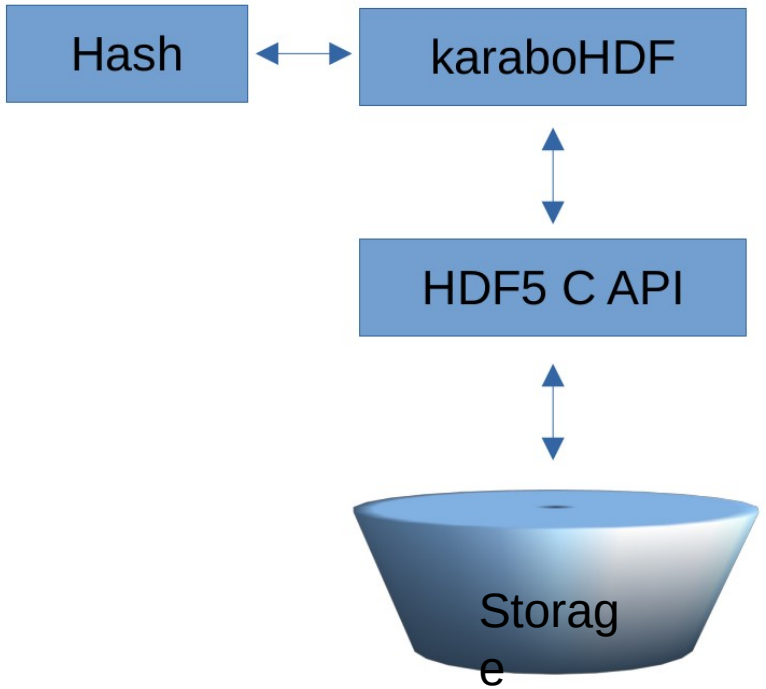
Sustained writing rates to **gpfs** storage over 15 GB/s



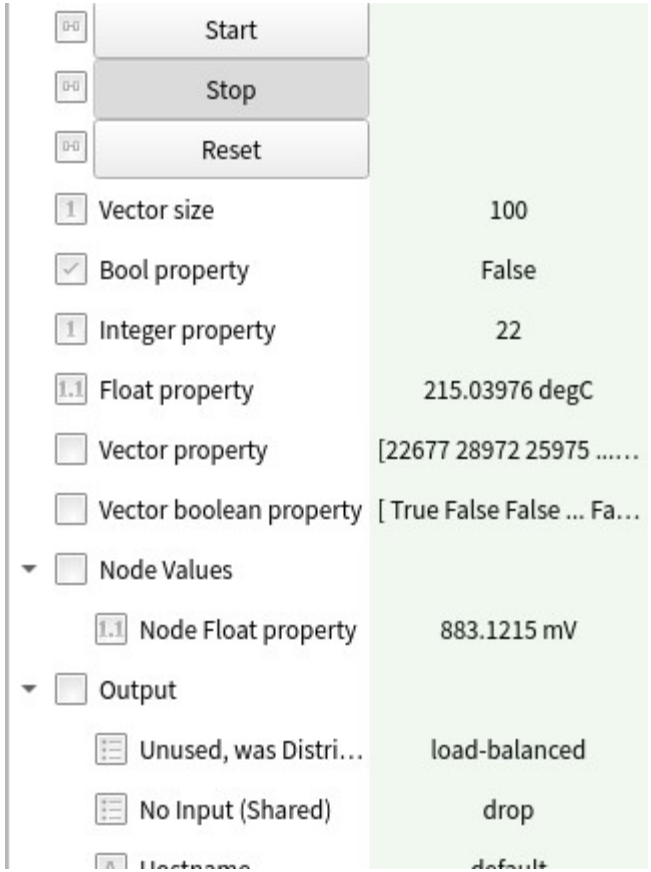
KaraboHDF5 (i)

C++ library written on top of the HDF5 C API.

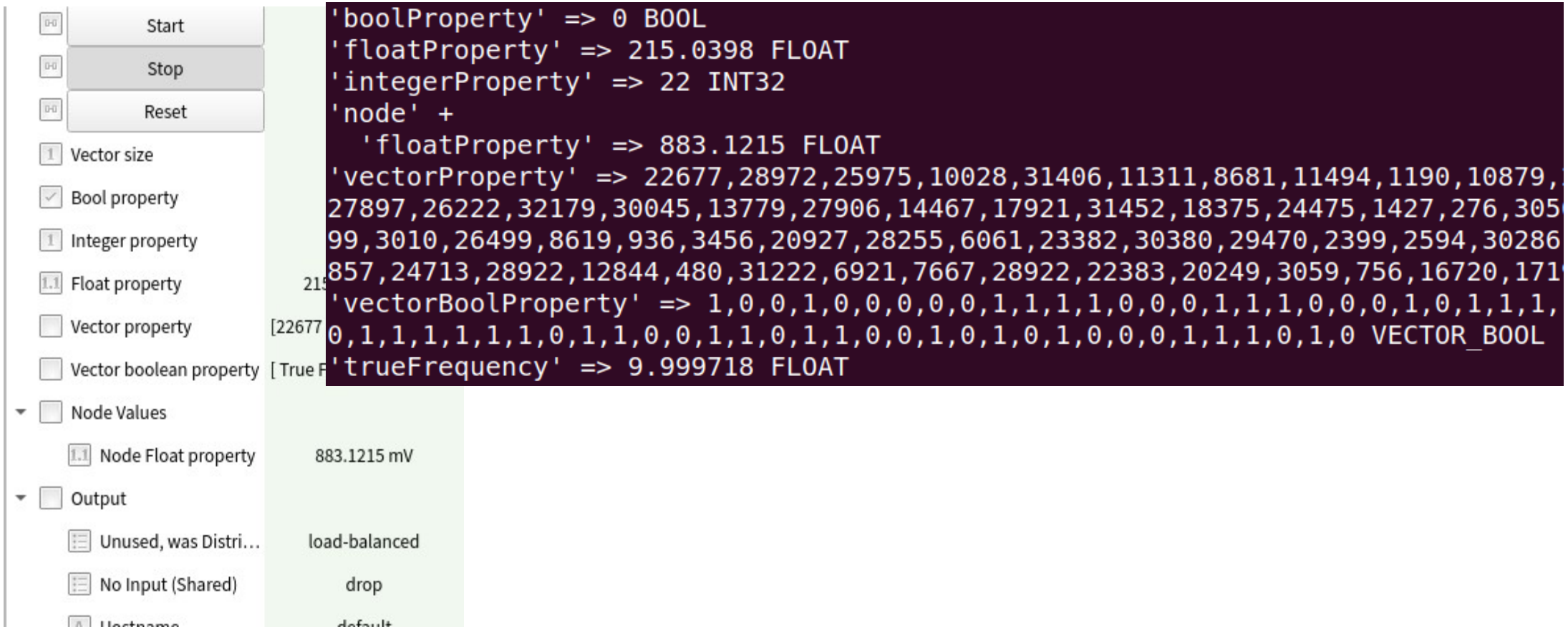
- Maps *Hash* data structures to HDF5 data objects.
  - A *Hash* can be mapped to a *Group*.
  - A *value* can be mapped to a *Dataset*.
  - *Attributes* of a *Hash entry* map to HDF5 *attributes* of a *Dataset.*
- *karaboHDF5* uses **compression** and **data chunking** to optimise storage.



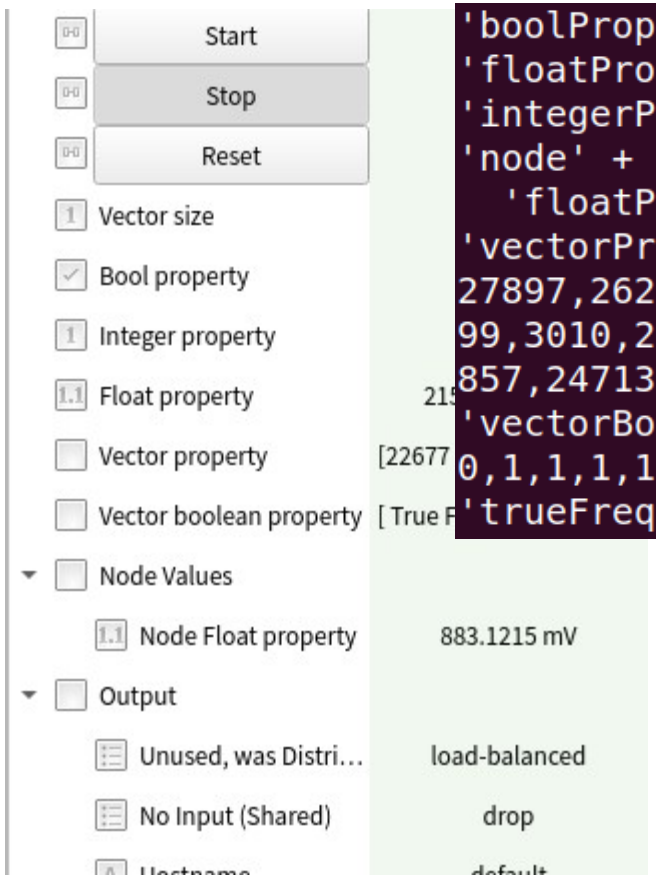
KaraboHDF5 (ii)



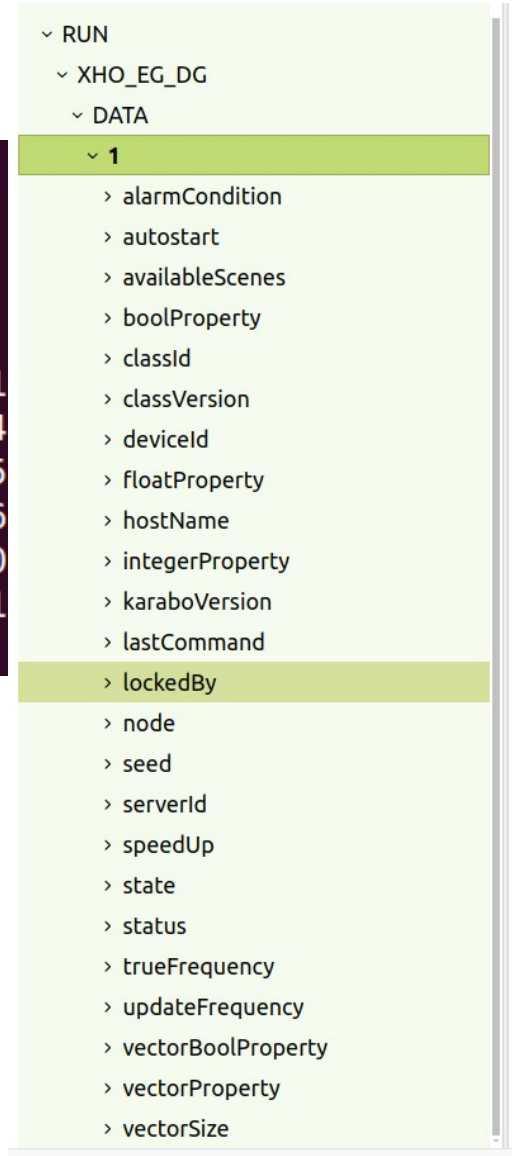
KaraboHDF5 (ii)



KaraboHDF5 (ii)




```
'boolProperty' => 0 BOOL
'floatProperty' => 215.0398 FLOAT
'integerProperty' => 22 INT32
'node' +
  'floatProperty' => 883.1215 FLOAT
'vectorProperty' => 22677,28972,25975,1
27897,26222,32179,30045,13779,27906,144
99,3010,26499,8619,936,3456,20927,28255
857,24713,28922,12844,480,31222,6921,76
'vectorBoolProperty' => 1,0,0,1,0,0,0,0
0,1,1,1,1,1,1,0,1,1,0,0,1,1,0,1,1,0,0,1
'trueFrequency' => 9.999718 FLOAT
```

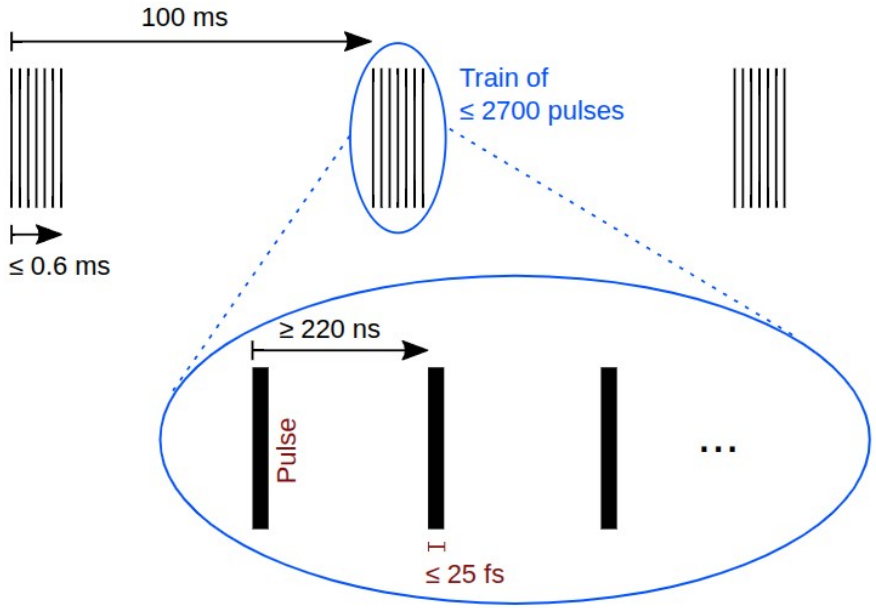



```
94,1190,10879,
5,1427,276,305
399,2594,30286
,756,16720,171
0,0,1,0,1,1,1,
0 VECTOR_BOOL
```

Data Reduction: *before* storing the data

Sheer amount of data is a problem: let's try to reduce it before saving the data.

- **Pulse-based reduction**.
  - Ignore images for pulses that have no information.
- **Train-based reduction**.
  - Ignore trains for pulses that have no information.



Data Reduction: *before* storing the data

Sheer amount of data is a problem: let's try to reduce it before saving the data

- **Pulse-based reduction**.
  - Ignore images for pulses that have no information.
- **Train-based reduction**.
  - Ignore trains for pulses that have no information.
- **Gain compression**, for AGIPD detectors.
  - Each pixel can have one of three gain stages, depending on incident number of photons.
  - In case of fixed gain, the gain array can be discarded.