

# ASAPO: A High-performance streaming framework for real-time data analysis

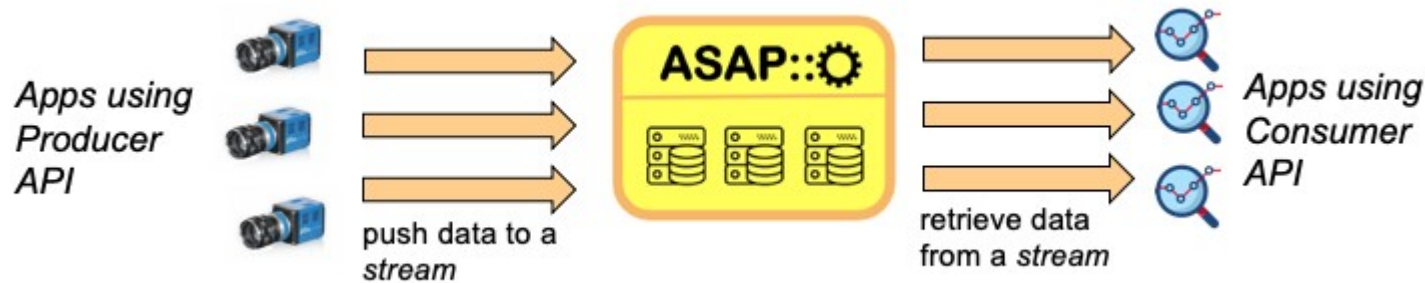
Mikhail Karnevskiy

DESY IT

SciComp Workshop 2025

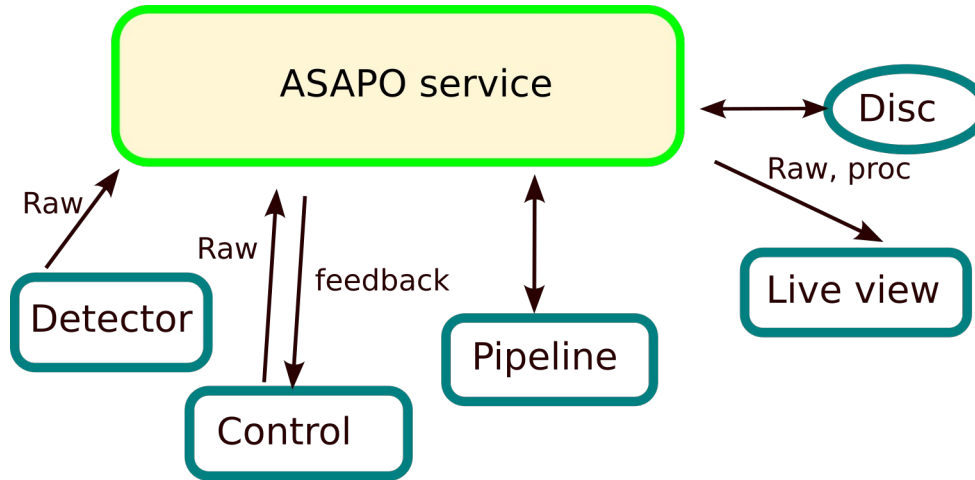
03 July, 2025

# Asapo introduction



- High-bandwidth communication between state-of-the-art detectors, control system, the storage system, and independent analysis processes across DESY facility
- In-memory data transfer with optional caching on disc
  - Large, scalable in-memory cache
  - Saving data to disc as service
  - Deliver data from disc
- Easy to use C++/Python interfaces:
  - Producer: sends data to Asapo
  - Consumer: get data from Asapo
- Stays for: Online data-processing. Data reduction. Feedback.

# Asapo Data flow

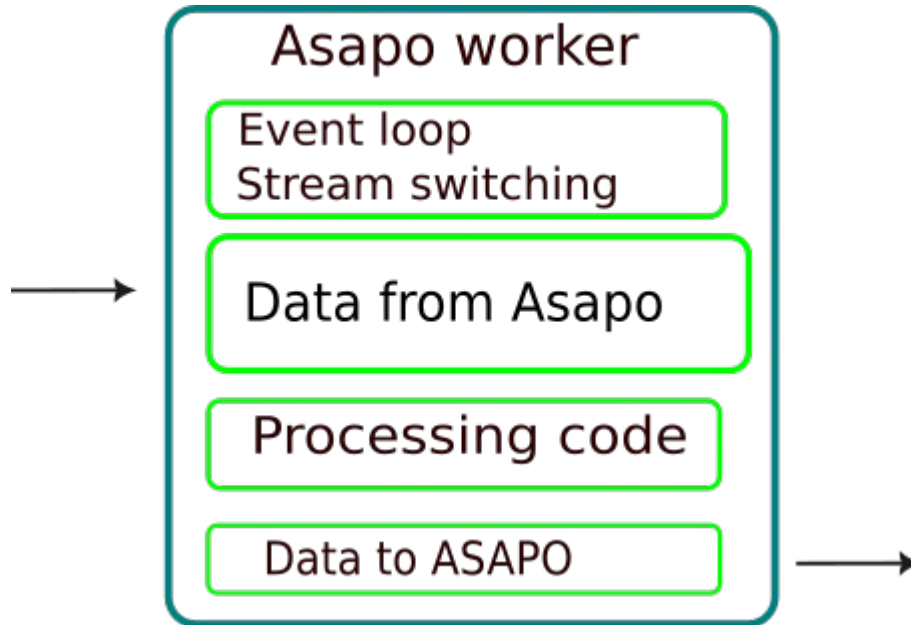


- Communication between different components across DESY facility
- Data-buffet with optional caching on disc
- Data writing goes in parallel with processing
- Multiple components are provided as a service for scientists

# Challenges

- Data-flow at kHz rate for 9M (highly compressed) images.
- Different detectors streams the data in different formats and being controlled in different ways
- Control data may come from different sources
- Data stream may not know, when it finishes
- Reprocessing may be requested at any time
- Minimum action from beamline scientists: data-processing as service
- Very little time for commissioning

# Asapo worker

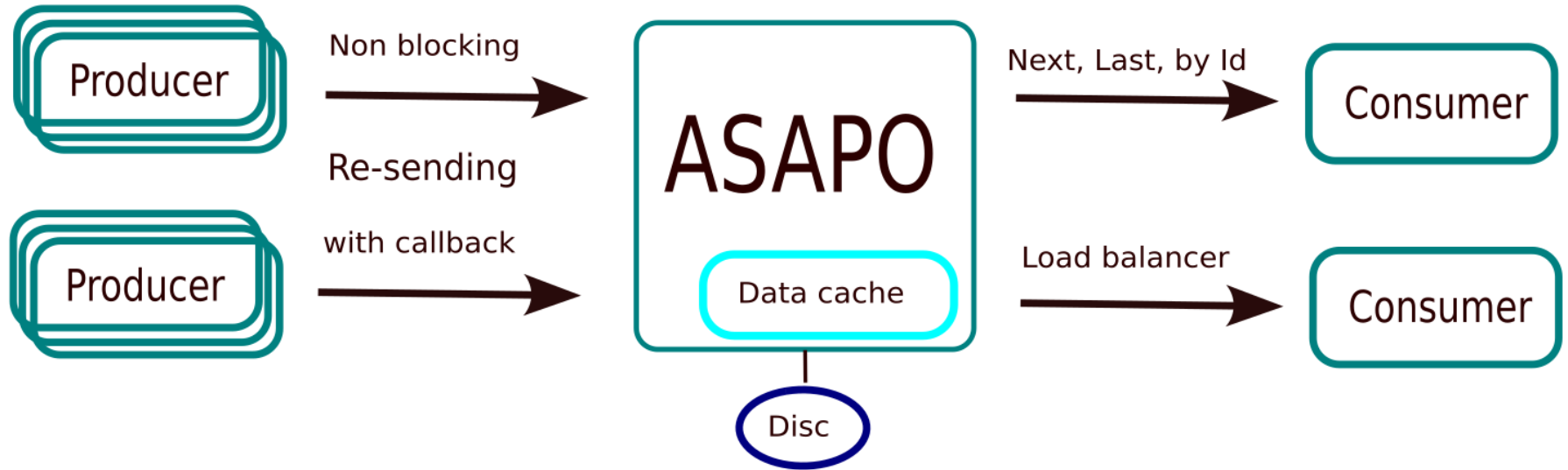


Examples:

- Saving data to file
- Radial integration of diffraction rings

- Uses Python or Cpp Asapo clients
- Data processing with a chain of workers
- Communication via Asapo service
  - Workers does not know each other, but knows data-source to retrieve.
- Worker runs constantly and solves a problem of stream switching.

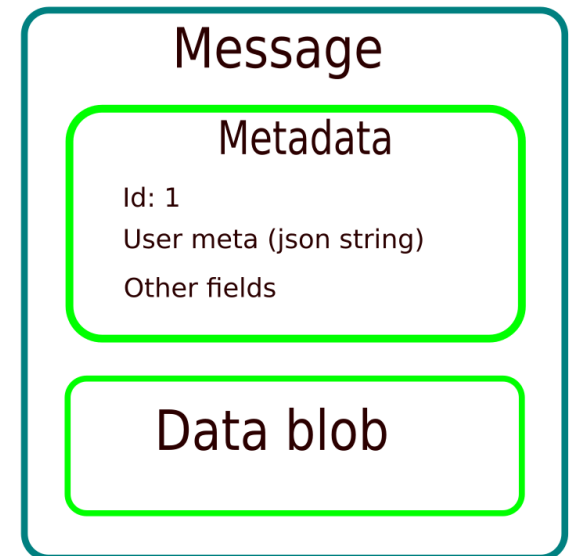
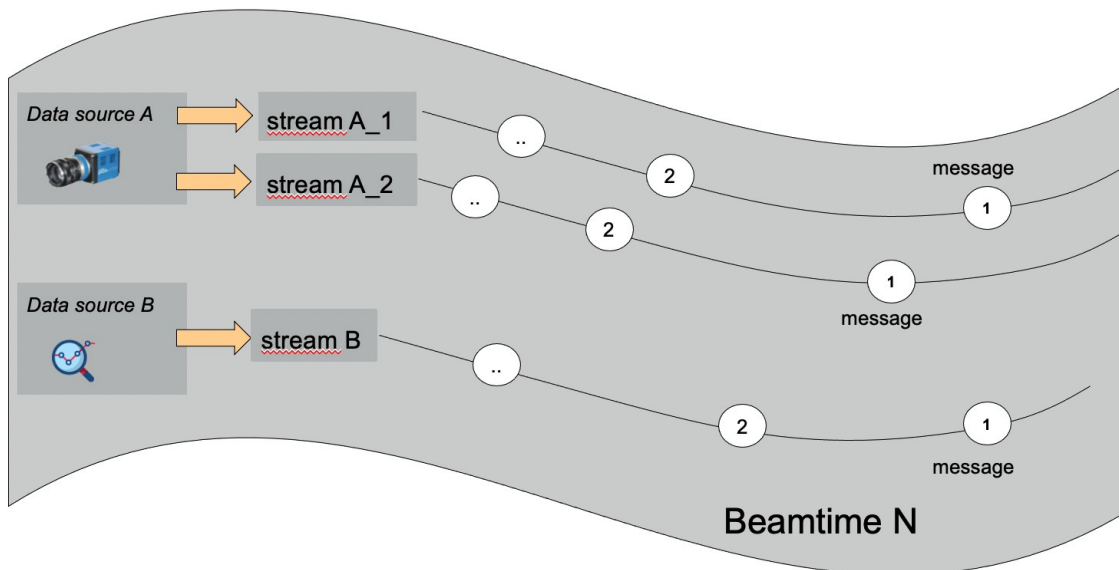
# Asapo API



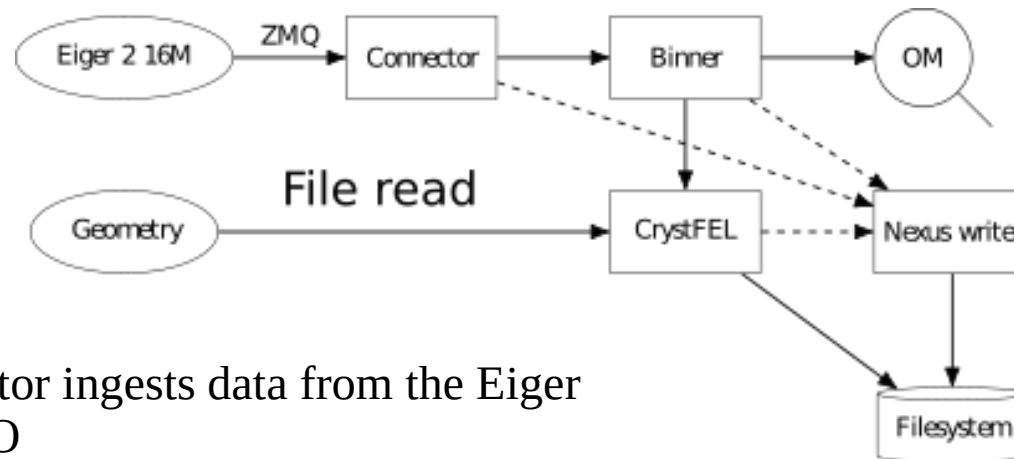
- Producers and consumer configured are fully independent
  - All components, including central service does not know, how many client are running
- Configuration is done using central endpoint, stream and data-source name

# Data in ASAPO

- Messages are indexes from 1 to N and form Stream.
- Streams are uniquely identified by its name, beamtime and data-source name
- Each message contains a binary data blob and a JSON metadata.
- Separate handling of data and metadata.
  - Data is stored in memory-cache and on disk, metadata is stored in database
  - This enables rich API and high throughput
- Synchronization of streams: several data-sources can be combined into a dataset



# Pipeline example

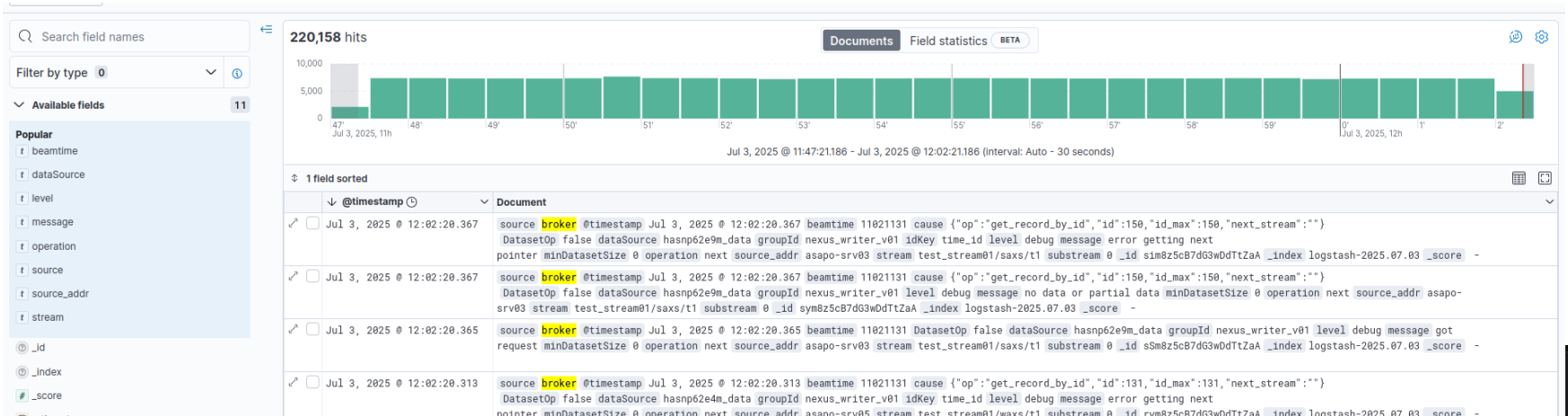


- ASAPO-Eiger-Connector ingests data from the Eiger ZMQ stream to ASAPO
- (Optional) Binner reduces images resolution to speed up later processing steps
- CrystFEL for peak search, indexing, and integration
- OM (OnDA Monitor) for live visualization
- Nexus writer can write raw, binned, or filtered (hits only) images and metadata to disk, depending on which data source it is connected to
- Currently, geometry/analysis results are read/written by CrystFEL from/to disk directly

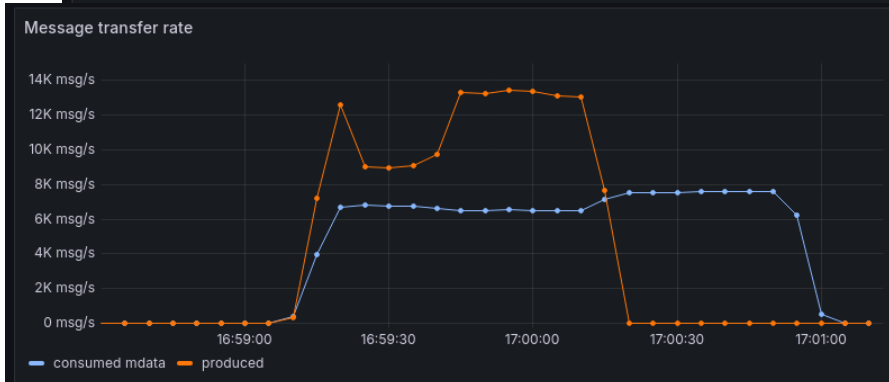


# Monitoring

- Monitoring is based on Grafana + InfluxDB
- Logging is based on Kibana and Elasticsearch



2025-02-06 09:41:26.770	data_source_c6e5cc16-663e-4761-8999-b833d5eda8d6_1	5001	unknown
2025-02-06 09:41:26.770	data_source_c6e5cc16-663e-4761-8999-b833d5eda8d6_0	5001	unknown



# Try ASAPO



- Git at DESY: <https://gitlab.desy.de/asapo>



- Pipy client packages. 

- Docs: <https://asapo.pages.desy.de/asapo/>

## **ASAPO standalone service:**

- Single docker with all asapo services
- Monitoring via Grafana  
- Limited functionality (not scalable)
- Fully functional API

# Summary

- Asapo is a streaming platform and service provided by DESY-IT
- Service is user at several beamlines at Petra III to establish data-flow and enable online data-processing
- Services are update few times per years to provide new features. Bugfixes-updates are possible during the user-run.
- Current development is focused to establish streaming-based data-flow.