

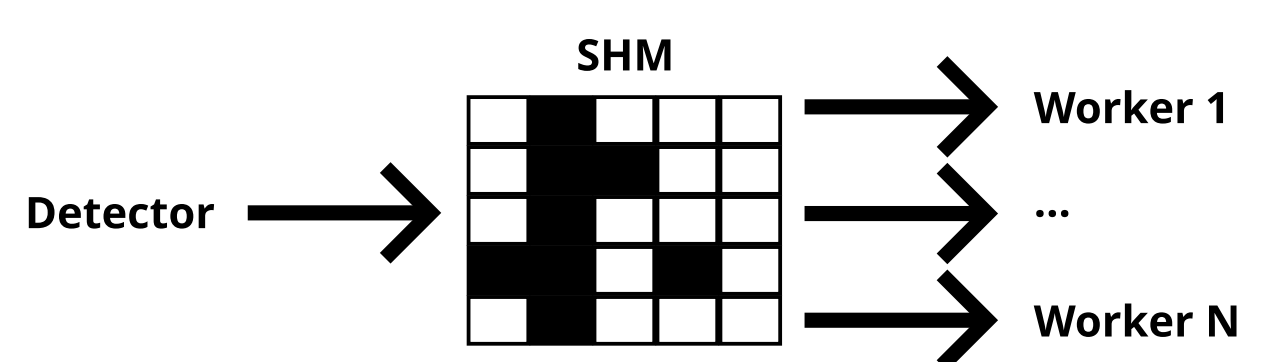
LiberTEM-live: Real-Time Data Processing for Electron Microscopy

What is LiberTEM-live

- A real-time streaming data processing engine, developed for electron microscopy, written in **Python** and **Rust**
- Main application: 4D STEM
- Supported detectors: Quantum Detectors MerlinEM, Amsterdam Scientific Instruments CheeTah T3, DECTRIS-based detectors (ARINA, QUATRO, ...)
- Experimental support: Gatan K2IS, ASI frame-based detectors
- Supports custom algorithms (user-defined functions; UDFs), for example for custom contrast methods, portable between live and offline processing
- Throughput: tested with real detectors up to 2GiB/s. Performance depends on your custom computation and your hardware
- Scalable: runs on CPU and multi-GPU systems
- Can be integrated into custom workflows and user interfaces
- Current UI: live plotting in jupyter notebooks. In development: GUI based on CEOS PantaRhei

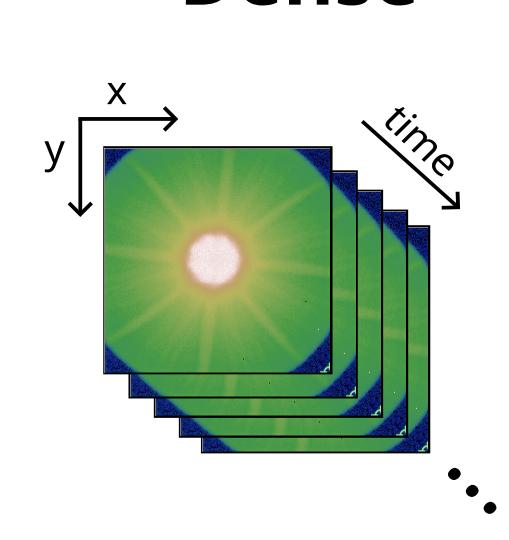
Real time?

- Ideal: system can run indefinitely, and can keep up with the data rate of the detector
- By principle, the performance depends on the actual **user code** that is run on the data stream, and on the **hardware**, so we cannot make absolute guarantees
- Our case: **soft** or **firm** real-time, depending on the use-case. For a preview, we can afford to drop frames, but if we are recording data, it can lose value if frames are dropped.
- There is still a nontrivial amount of **latency** in the whole pipeline, from receiving data to showing results, and also **jitter**, which is for example introduced by garbage collection of Python
- If the receive buffer is approx. as large as one acquisition: can **relax** real-time requirements by waiting between acquisitions until the receive buffer is completely drained
- In the relaxed scenario: main limiting resource: size of **shared memory receive buffer**

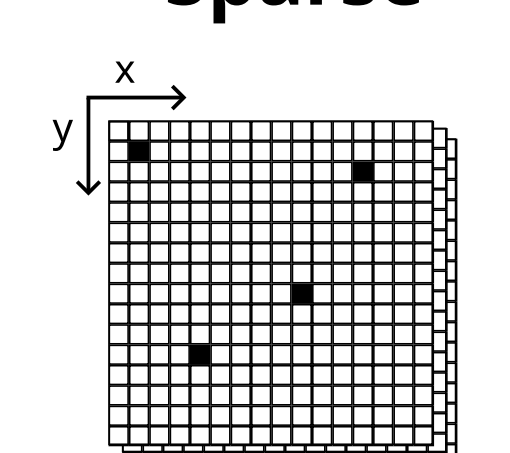


What does the data look like?

Dense

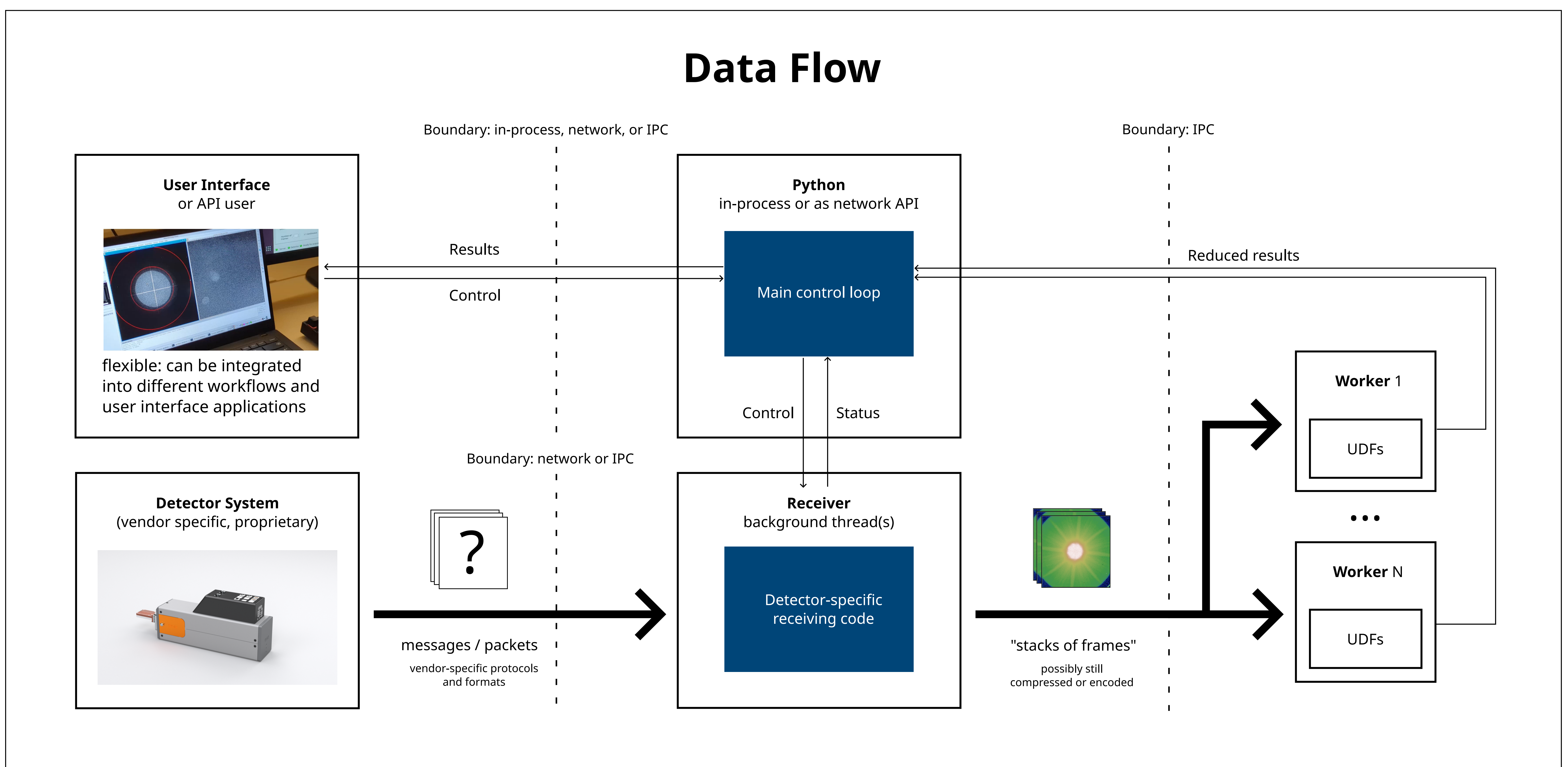


Sparse



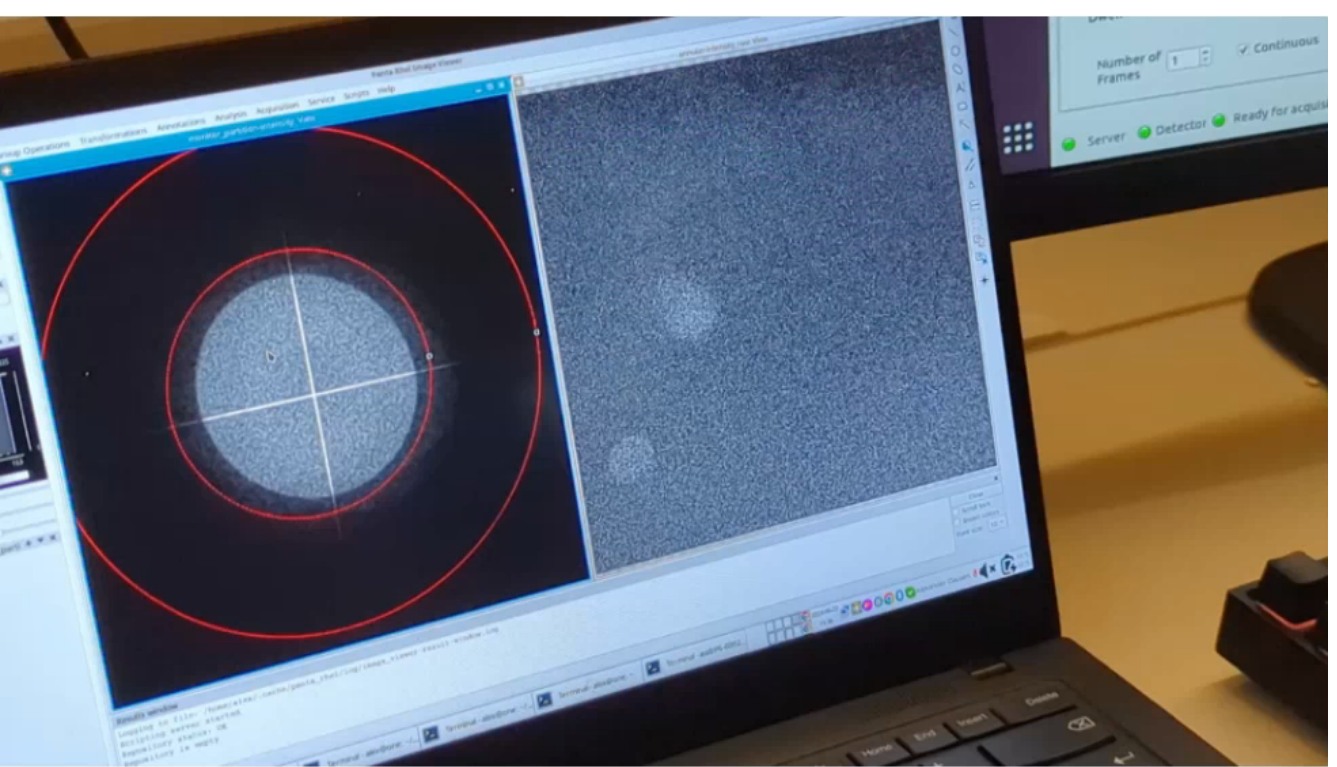
- Detectors: like high-speed greyscale cameras
- Input data is a stream of 2D images, with **headers** interspersed, in a **vendor-specific** encoding/protocol, possibly compressed, ...
- Typical detector sizes: 128x128, 192x192, 256x256, 512x512, 1860x2048
- Typical frame rates: 1kHz ~ 120kHz
- Frame rates and sizes can vary at runtime: they are influenced by settings like binning, only recording a subset region of the detector, or bit depth

- Principle: only record and transmit non-zero values
- Result: individual time-stamped events (**x, y, t**) of when and where an electron hit the detector
- For 4D STEM, the events are then put into bins for each scan position
- We support a streaming variant of the **standard CSR format**, by splitting the CSR matrix for the whole stream into chunks for processing



Example applications

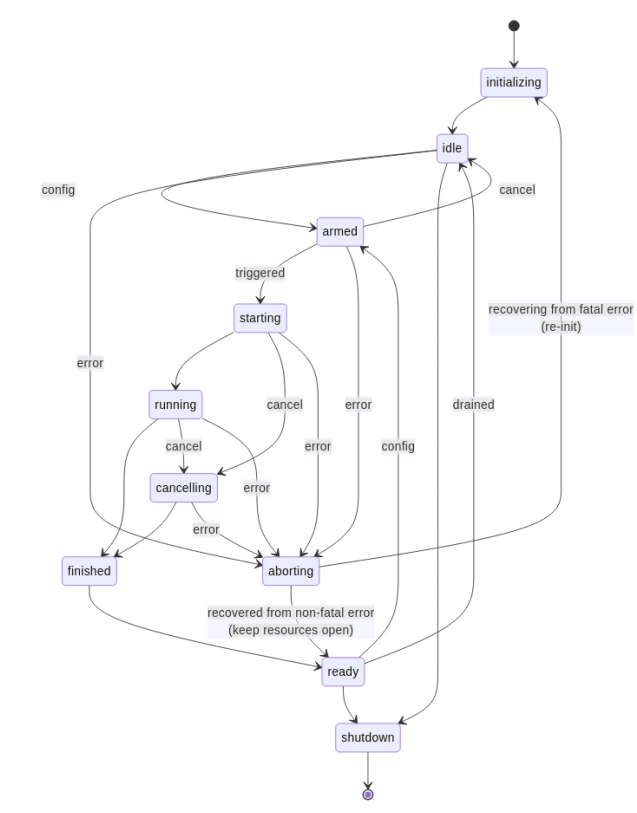
- Common 4D STEM functionality included/available: CoM, virtual detectors, single-sideband ptychography, strain mapping, ...
- LiberTEM-live, integrated into the CEOS PantaRhei GUI, as a 4D STEM interface (WIP)



- Not limited to electron microscopy - see "Live Iterative Ptychography" by D. Weber for an example use-case in X-ray ptychography: <https://doi.org/10.1093/mam/ozae004>
- Your use-case?

Future plans

- Support for new detectors, including Timepix4
- Work on user interfaces to make LiberTEM-live more accessible to microscopists
- Extract low-level components into re-usable libraries, wherever there is interest - generic array streams?
- Make the camera interface more generic - require less Python glue code and make it easier to integrate support for new detectors
- Use a common state machine for predictable behavior



LiberTEM-live

- Python / Rust
- Open source: GPLv3 / MIT

pip install libertem-live

https://libertem.github.io