OM: Online Data Analysis and Feedback for Serial X-ray Imaging

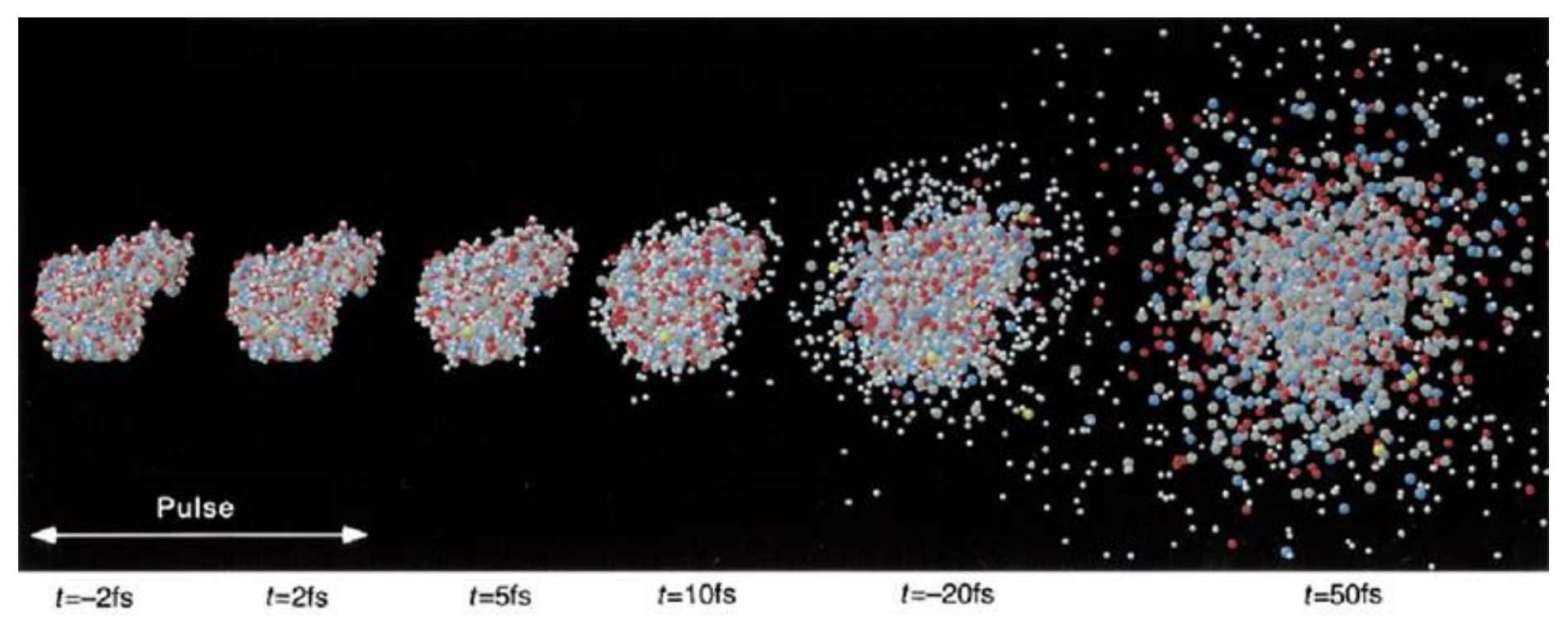
SLAC NATIONAL ACCELERATOR LABORATORY

Abdullah Al Maruf*, Valerio Mariani**, Amedeo Perazzo

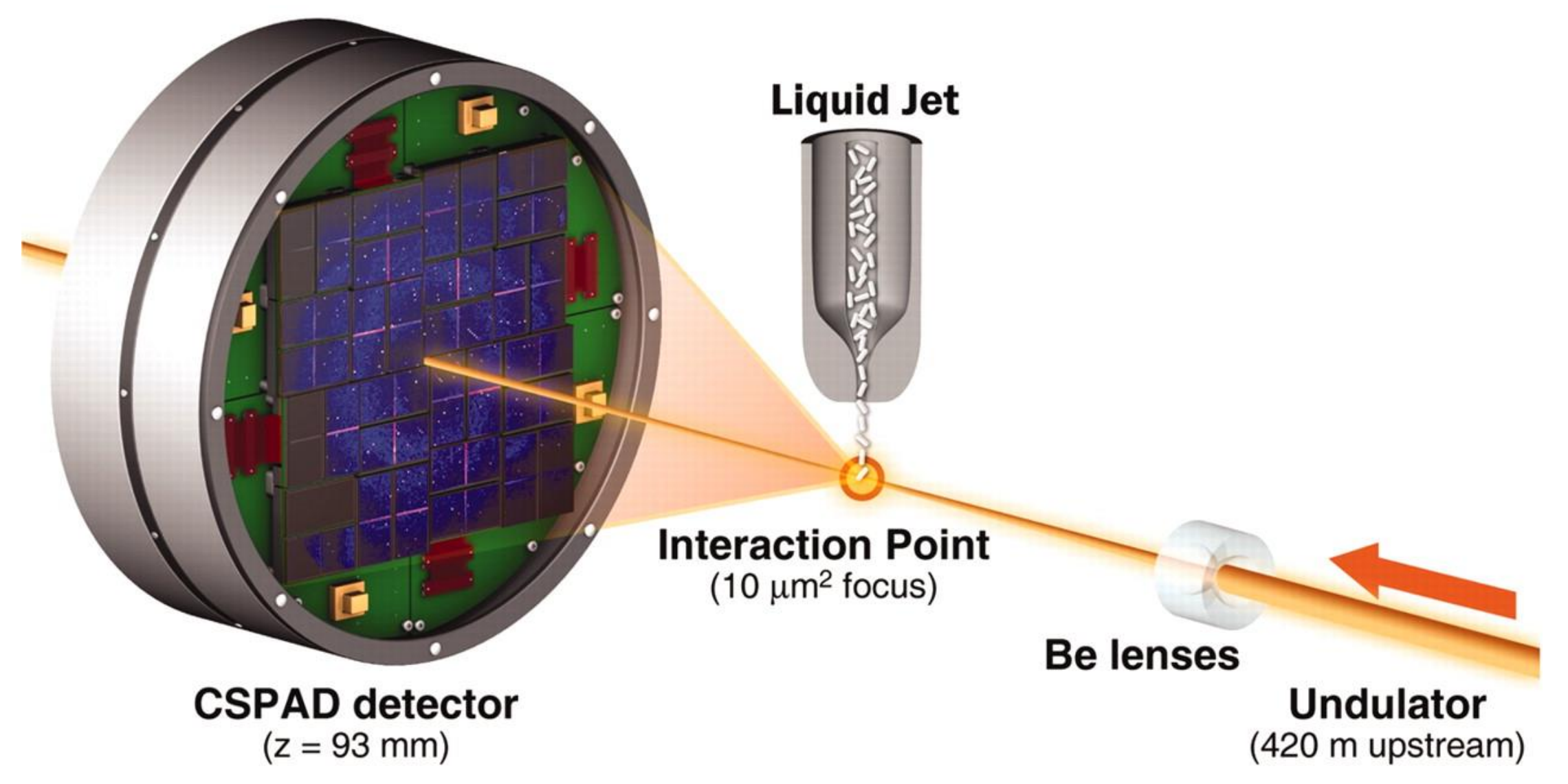
*amaruf@slac.stanford.edu (student)
**valmar@slac.stanford.edu (mentor)

Motivation

- ❖ The advent of X-ray free-electron lasers (**XFELs**) has opened new possibilities for structural biology.
- ❖ The ability of XFELs to deliver a very high radiation dose to the sample within femtosecond time scales has been exploited in several newly developed techniques, mostly based on the so-called '**diffraction-before-destruction**' method.



- ❖ In the Serial Femtosecond Crystallography (**SFX**) techniques, samples are constantly flowed in a liquid jet across a pulsed X-ray source which has a repetition rate of up to 120 Hz.



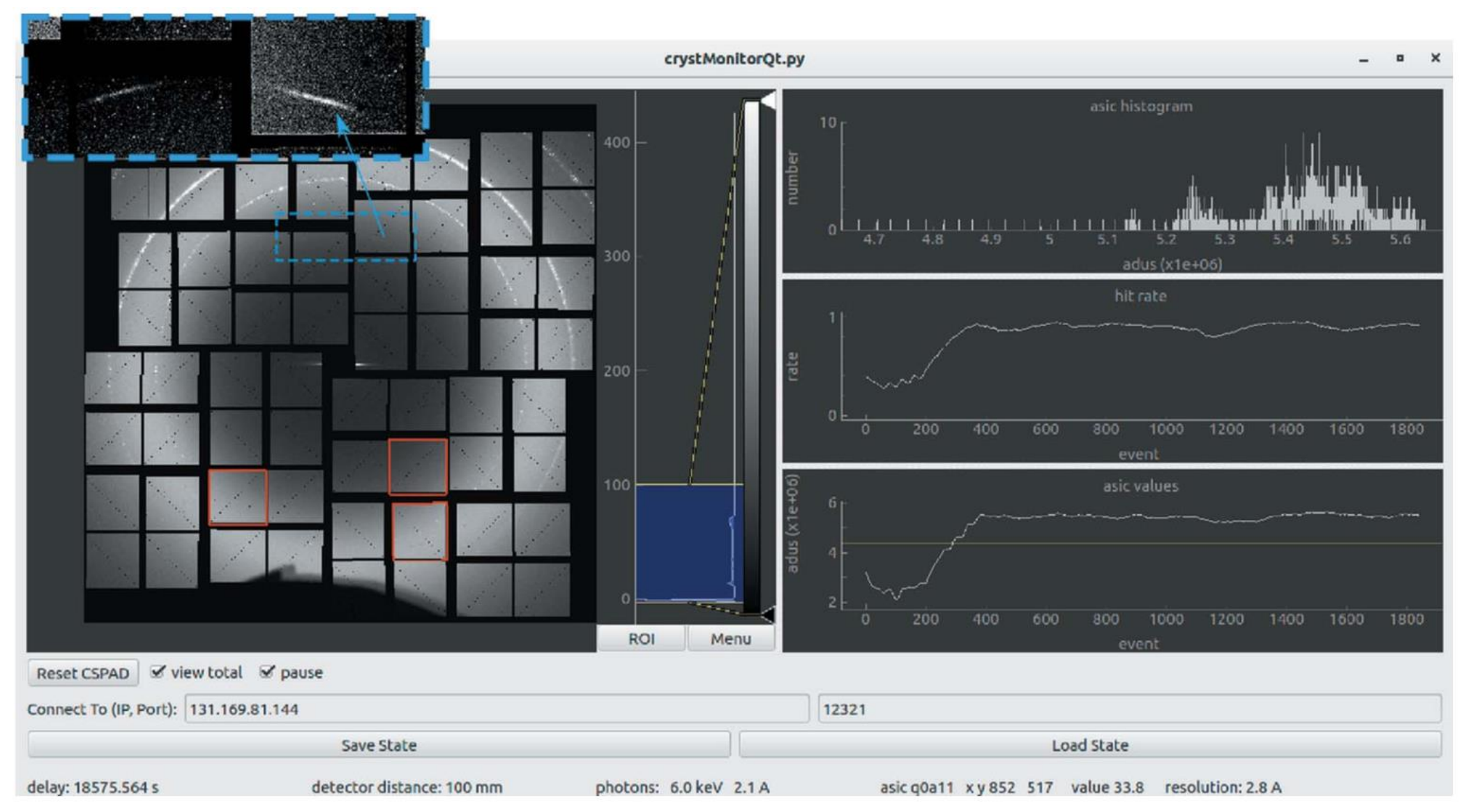
- ❖ By monitoring experimental conditions in close to real time, we can maintain the system in an optimal condition.
- ❖ Hence, we need to develop a universal software platform (for almost any detector) to **visualize** and **analyze** the diffraction data in **real time**, enabling the rapid course-correction of the experiment to save highly expensive sample resources.

  - ➢ Accuracy is not a strong requirement, low latency is!
  - ➢ Low latency is more important than completeness

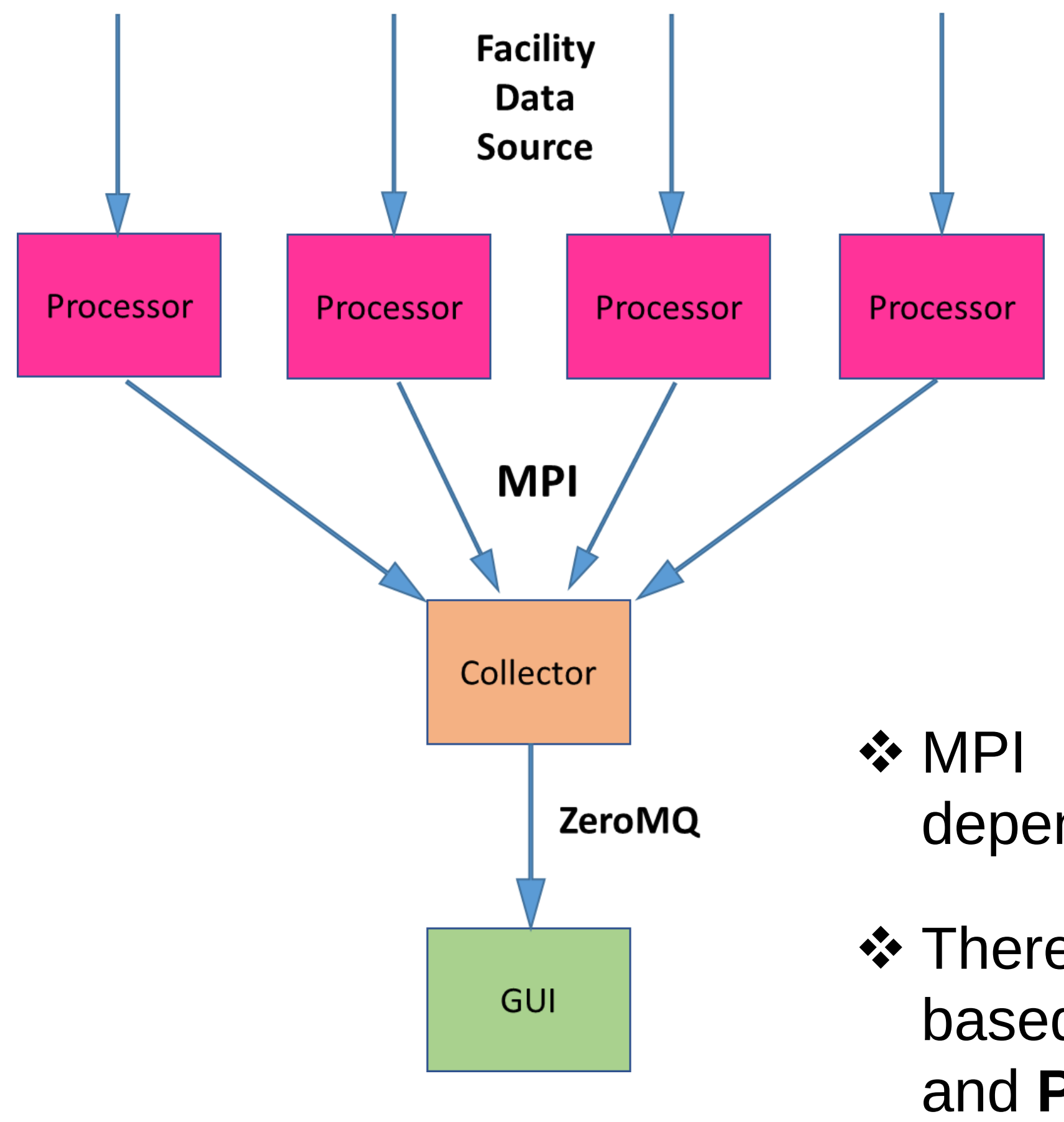
Boutet et al., Science (2012)
Neutze et al., Nature (2000)

What is OM?

- ❖ **OM** (**O**nDA **M**onitor) is a framework for the development of programs that can **monitor** x-ray imaging **experiments** in **real-time**. It is also a set of ready-to-use monitoring programs.
- ❖ OM can process imaging data in the **broadest sense**: multidimensional pixel-based data, for example, diffraction patterns, photoemission spectrums, images from cameras, microscopes or x-ray detectors.



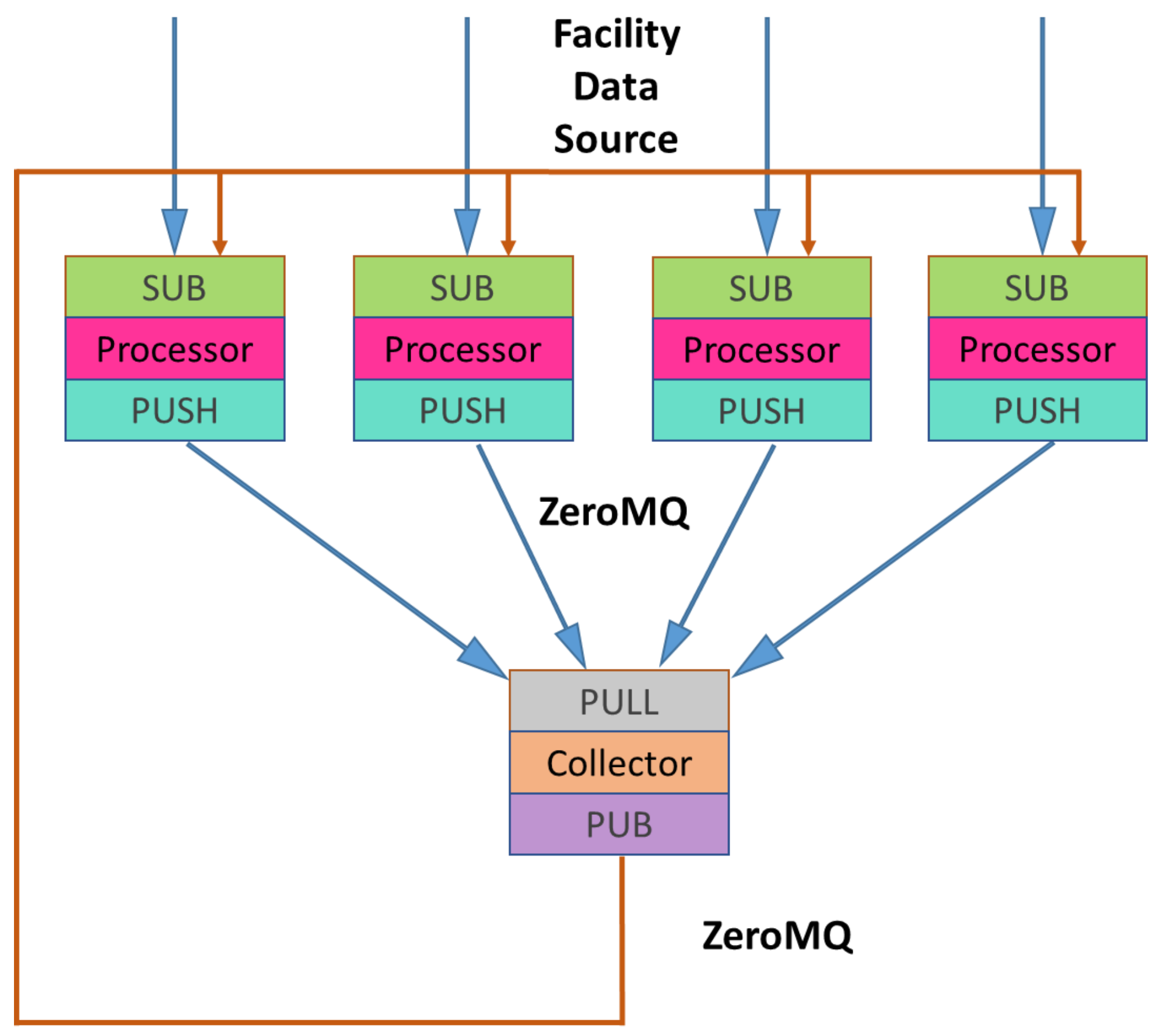
Parallelization



- ❖ Initially OM utilized MPI parallelization, to communicate between processing and collecting nodes.
- ❖ MPI, while fast, has a static approach to process data and to allocate hardware resources, with unused cores during the experiment.
- ❖ In case of an error, or to change the number of nodes, one needs to restart OM, interrupting the monitoring of the experiment and loosing the accumulated statistics.
- ❖ MPI also requires beamline-facilities to install several dependencies before running OM successfully.
- ❖ Therefore, to overcome this issue, we utilized a new socket-based approach, replacing MPI, to achieve higher **Scalability** and **Portability** of OM.

- ❖ Here, we implemented the **ZeroMQ** (**ØMQ**) sockets in the parallelization layer of OM for rapid data distribution and internal communication – to monitor the experimental data in the GUI to course-correct in real time!

  - ➢ **PUSH** + **PULL**: Each processor subscribes to the collector node and pushes the processed data to it.
  - ➢ **PUB** + **SUB**: The collector node publishes feedback to one or more processors that subscribe to the stream



Mariani et al., J. App. Crystallography (2016)

Conclusions

In this project, we implemented a ZeroMQ socket-based parallelization layer in the current OM software to achieve higher efficiency in the hardware resource management during the experiment and to reduce external dependencies, that were necessary in the MPI parallelization, making it highly scalable and portable for future application and development.

This work makes the scalability of OM possible, and it will allow the development of software to monitor resource usage and determine when more or less resources are needed.

Accessibility

OM is written in Python, licensed under the permissive GNU 3.0 license. OM is completely free and opensource. Individuals are free to modify, extend and adapt OM for research or commercial use, provided that they share the modification.

OM is available on CondaForge and PyPI for Linux. The current release and source code can be downloaded and reviewed at the OM GitHub repository. More information and documentation is available at the OM website

GitHub Repo

Website

Acknowledgments

This work was supported by the DOE's Exascale Computing Project (ECP) Summer Fellowship Program.

I would also like to thank Dr. Valerio Mariani and Dr. Amedeo Parazzo for their mentorship and support, respectively, throughout this project.

U.S. DEPARTMENT OF ENERGY
ECP EXASCALE COMPUTING PROJECT