When hardware and software work in *Concert*

M. Vogelgesang*, T. Faragó†, T. Rolo†, A. Kopmann* and T. Baumbach†
*Institute for Data Processing and Electronics, †Institute for Photon Science and Synchrotron Radiation
Karlsruhe Institute of Technology

Motivation

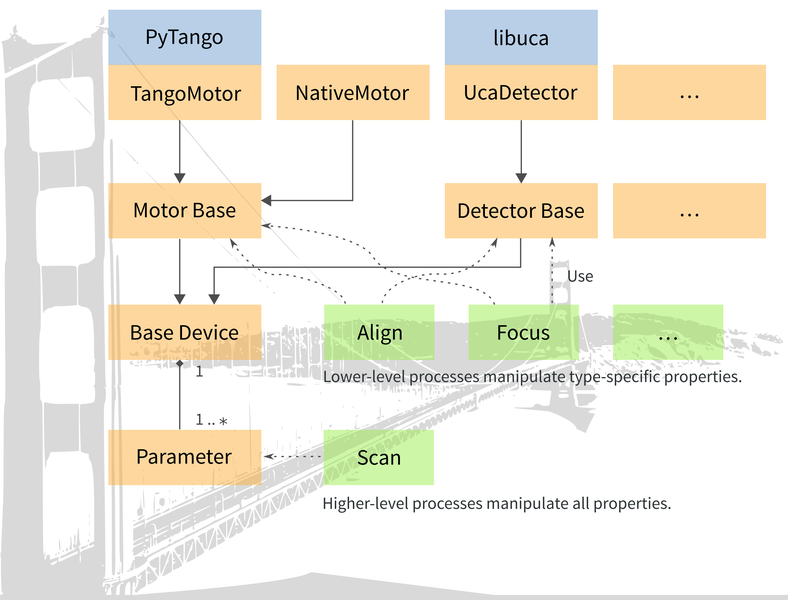
Current X-ray beamline management faces

- increasing amount of data
- lack of process and compute integration
- more concurrent execution
- time-consuming setup

Concert Control System

- Flexible control system for Python 2.7+
- Leverages existing control software, e.g. TANGO
- Easy installation: `pip install concert`
- IPython, NumPy and NeXpy integration

System architecture



Quantities

- *Type safety* through unit validation:

```
pump.flow_rate = 2 * q.liter / q.s
motor.position = 5 * q.hour
>>> Sorry, `position' can only receive meter unit
```

Asynchronous execution

- Map (a)synchronous operations to *Futures*

```
# Synchronous access
m.position = 2.1 * q.mm
print(m.position)
```

```
# Asynchronous access
f = m.set_position(2.1 * q.mm)
f.wait()
f = m.get_position()
print(f.result())
```

Sessions

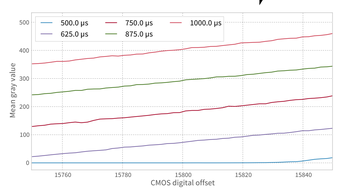
- Encapsulate experiments and define relationships
- Facilitates a simple command-line interface:

```
$ concert init <session> or concert fetch <url>
$ concert {start, edit, show, log, rm} <session>
```

Processes

Parameter scans

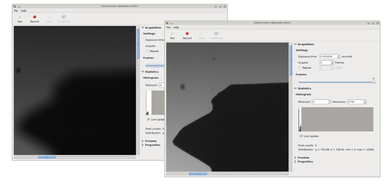
```
scanner = Scanner(detector['exposure-time'], lambda: offset_scan,
                  minimum=0.5*q.ms, maximum=10*q.ms)
x, y = scanner.run().result()
plot_double_scan(x, y)
```



Characterization of a CMOS-based detector: For five different exposure time settings, a digital ADC offset was changed and the mean grey value of the acquired data computed and plotted within Python.

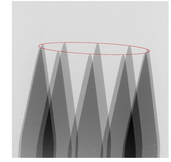
Calibration

```
maximizer = Maximizer(focus_motor['position'],
                      lambda: std(detector.grab), bfgs)
detector.start_recording()
f = maximizer.run()
f.add_done_callback(lambda: detector.stop_recording())
```

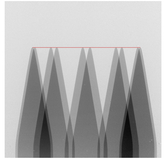


Result of focusing: The position of the `focus_motor` is optimized with respect to the standard deviation of the currently acquired image. The detector is stopped when the process is finished.

```
a = Aligner(Ellipse(), read_frames(), x_motor, rotation_motor)
a.run().wait()
```



Before alignment: The sample tip follows an elliptic path during the rotation. The aligner deduces the correct `x_motor` position.



After alignment: The sample moves perpendicular to the detector plane.

Fast GPU-based reconstruction

Tomographic reconstruction of a high-speed scan carried out by our integrated GPU-based data processing framework. *Left:* a flat-field-corrected projection. *Right:* a reconstructed and visualized tomographic volume.

Conclusion

- Open-source control system interface
- Rapid experiment prototyping
- Improved experiment throughput and data process integration