GPUs for Calibration Processing at XFEL.EU

GPU Round Table – DESY, 6.11.2018

Steffen Hauf
(steffen.hauf@xfel.eu)

Overview

- Detectors @ European XFEL
- Detector Calibration
  - Tools we use
  - Online Calibration
  - GPU-based Corrections

- With contributions from AGIPD, DSSC and LPD consortia, first users and instruments, DET, CAS, ITDM, CFEL computing….

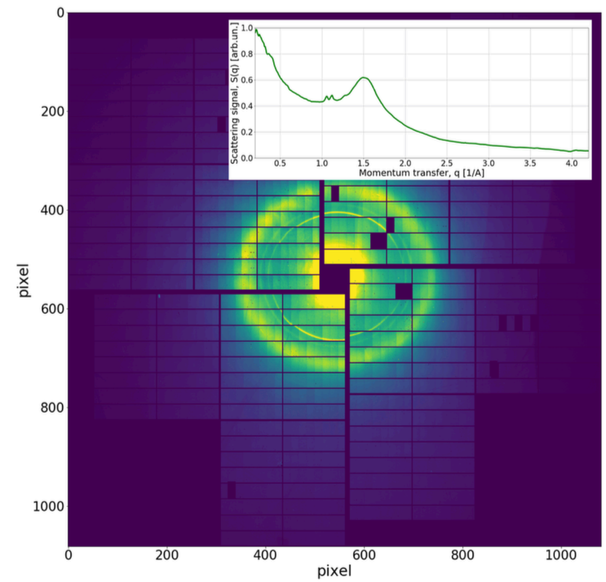
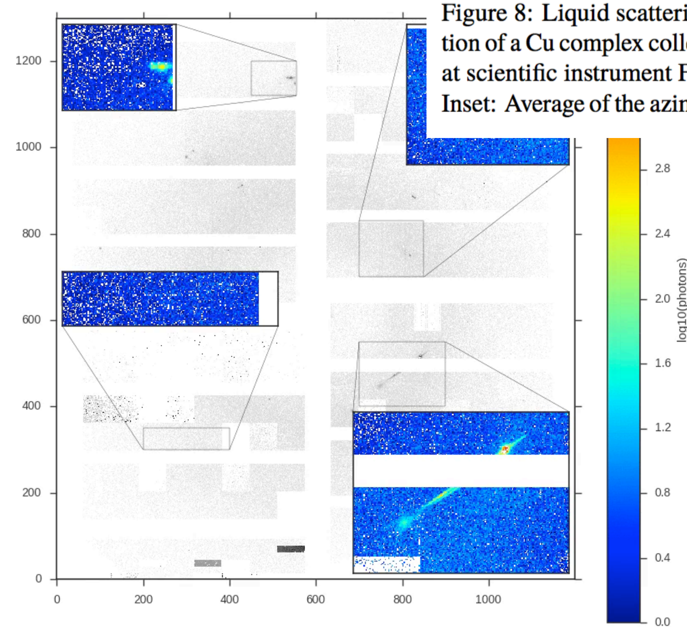


Figure 8: Liquid scattering pattern of tetrahydrofuran solution of a Cu complex collected with the LPD detector [23,24] at scientific instrument FXE [25](corrected for dark offset). Inset: Average of the azimuthally integrated set of 150 image.

LPD 1M

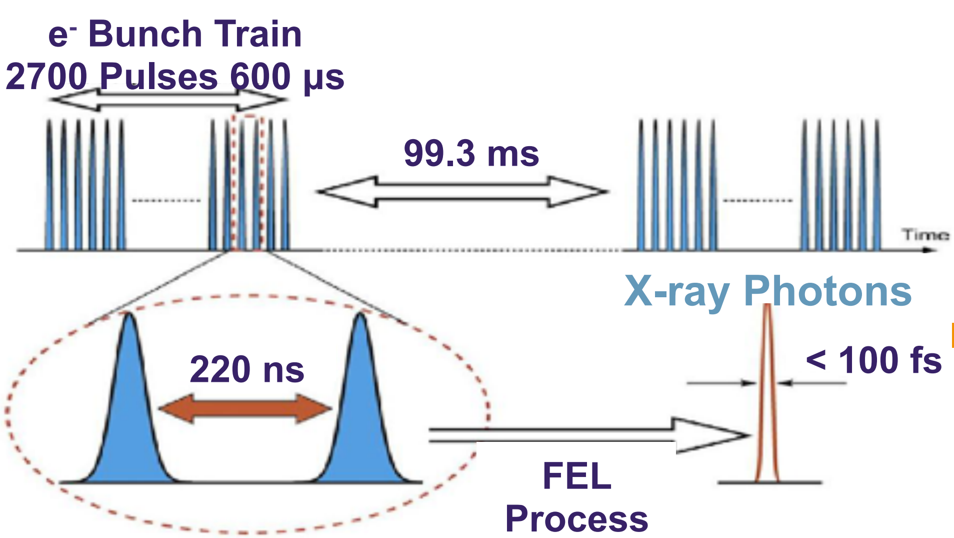
AGIPD 1M



Detectors: from First Ideas to User Operation – 2006 til 2017

- The European XFEL pulse structure poses strict constraints on detectors (e.g. intensity and time structure)
- No commercial imaging detectors available
- Call for expression of interest launched in 2006
- 3 project proposals were selected with the goal to finally have at least one fast 2D imaging detector

e$^-$ Bunch and X-ray Pulse Structure




European XFEL Project Team
c/o Deutsches Elektronen-Synchrotron DESY
in der Helmholtz-Gemeinschaft,
Notkestraße 85,
D-22607 Hamburg, Germany

XFEL
X-Ray Free-Electron Laser

Call by the:

European Project Team for the X-ray Free-Electron Laser

for:

Expressions of Interest

to:

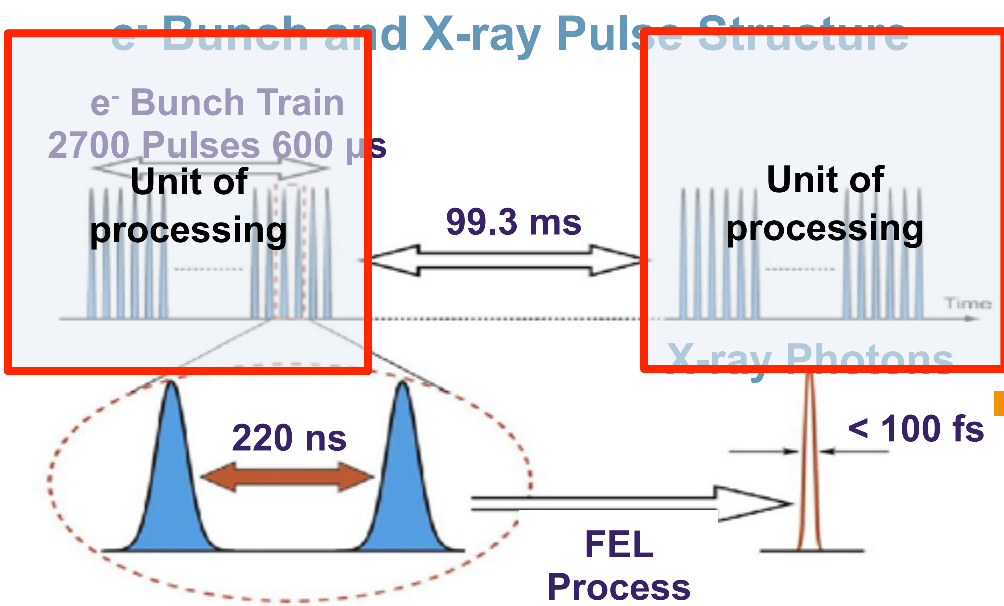
Develop and Deliver Large Area Pixellated X-ray Detectors.

Deadline: 30 September 2006
http://xfel.desy.de/xfelhomepage

- Selected proposals
  Adaptive Gain Integrating Pixel Detector
  Large Pixel Detector
  DEPFET Sensor with Signal Compression

Detectors: from First Ideas to User Operation – 2006 til 2017

- The European XFEL pulse structure poses strict constraints on detectors (e.g. intensity and time structure)
- No commercial imaging detectors available
- Call for expression of interest launched in 2006
- 3 project proposals were selected with the goal to finally have at least one fast 2D imaging detector




European XFEL Project Team
c/o Deutsches Elektronen-Synchrotron DESY
in der Helmholtz-Gemeinschaft,
Notkestraße 85,
D-22607 Hamburg, Germany

XFEL
X-Ray Free-Electron Laser

Call by the:

European Project Team for the X-ray Free-Electron Laser

for:

Expressions of Interest

to:

Develop and Deliver Large Area Pixellated X-ray Detectors.

Deadline: 30 September 2006
http://xfel.desy.de/xfelhomepage

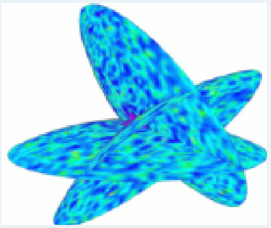
- Selected proposals
  - Adaptive Gain Integrating Pixel Detector
  - Large Pixel Detector
  - DEPFET Sensor with Signal Compression

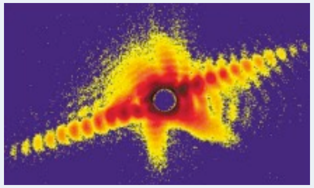
XFEL Scientific Instruments

Hard X-Rays

SPB — **Single Particles, Clusters and Biomolecules and Serial Femtosecond Crystallography**

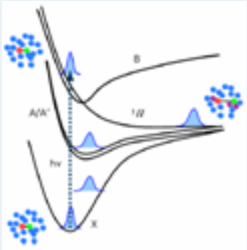
Will determine the structure of single particles, such as atomic clusters, viruses and biomolecules





MID — **Materials Imaging & Dynamics**

Will be able to image and analyse nano-sized devices and materials used in engineering

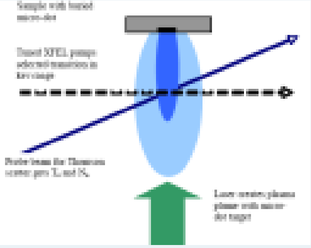


FXE — **Femtosecond X-Ray Experiments**

Will investigate chemical reactions at the atomic scale in short time scales molecular movies

HED — **High Energy Density Matter**

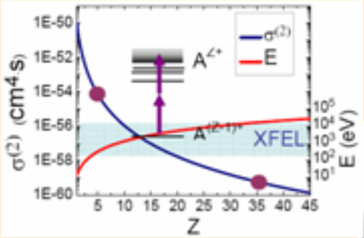
Will look into some of the most extreme states of matter in the universe, such as the conditions at the center of planets



Soft X-Rays

SQS — **Small Quantum Systems**

Will examine the quantum mechanical properties of atoms and molecules.

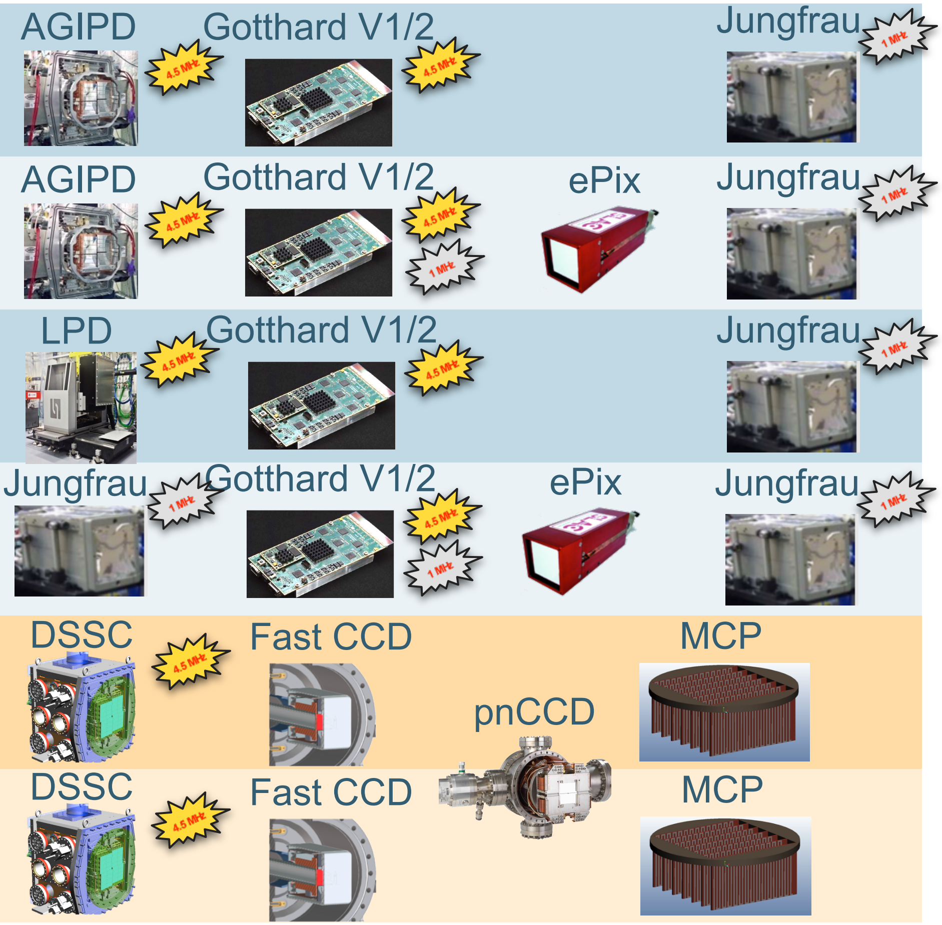


SCS — **Soft X-Ray Coherent Scattering/Spectroscopy**

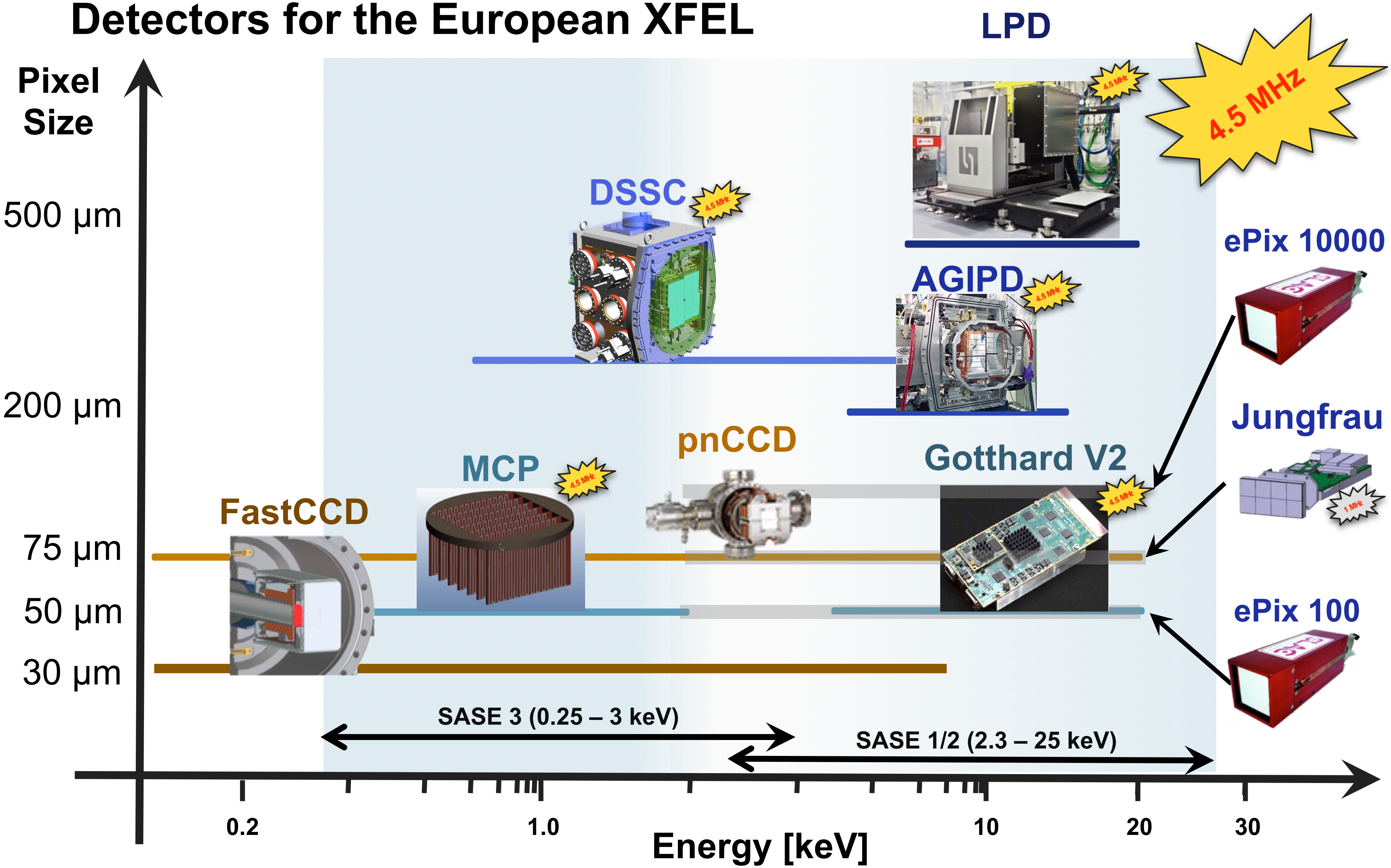
Will determine the structure and properties of large, complex molecules and nano-sized structures.

Detectors for the Scientific Instruments

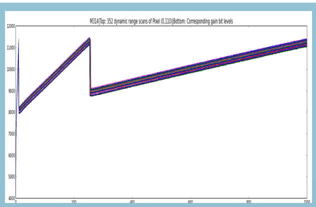
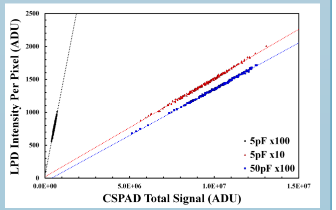
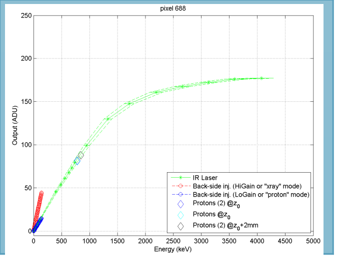
Source	Energy	Instrument	Detectors
SASE I	High E	Single Particles, Clusters and Biomolecules (SPB)	AGIPD, Gotthard V1/2, Jungfrau
SASE I	High E	Materials Imaging & Dynamics (MID)	AGIPD, Gotthard V1/2, ePix, Jungfrau
SASE II	High E	Femtosecond X-ray Experiments (FXE)	LPD, Gotthard V1/2, Jungfrau
SASE II	High E	High Energy Density Matter (HED)	Jungfrau, Gotthard V1/2, ePix, Jungfrau
SASE III	Low E	Small Quantum Systems (SQS)	DSSC, Fast CCD, pnCCD, MCP
SASE III	Low E	Spectroscopy and Coherent Scattering (SCS)	DSSC, Fast CCD, pnCCD, MCP



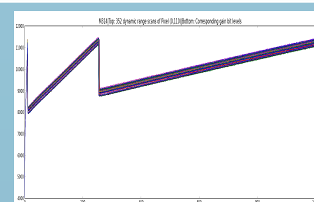
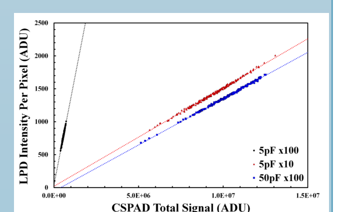
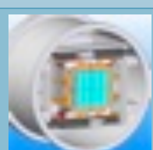
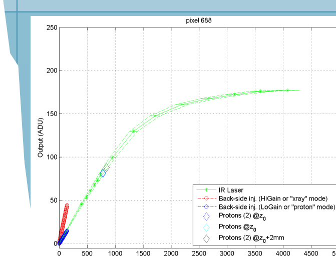
Detectors for the European XFEL



MHz Rate, High Dynamic Range Detectors – Challenges for Calibration

Detector	Specs	Modularity	Gain Switching	Gain Curve
AGIPD 	1 Mpixel, 4.5 MHz 352 memory cells 200µm sq. pixels 1-10^4 ph@ 12 keV 3 – 13 keV	16 modules in 2 cols x 8 rows on 4 quadrants	3 gain stages with automatic switching	
LPD 	1 Mpixel, 4.5 MHz 508 memory cells 500µm sq. pixels 1-10^5 ph@ 12 keV 5 – 25 keV	16 modules per supermodule (2x8) 16 SM on 4 quadrants	3 gain stages with on front-end selection	
DSSC 	1 Mpixel, 4.5 MHz 800memory cells 204µm hex. pixels 1-10^4 ph@ >1keV 0.5 – 6 keV	16 modules in 2 cols x 8 rows on 4 quadrants	Non-linear gain in ASIC (miniSDD), in sensor (DePFET)	

MHz Rate, High Dynamic Range Detectors – Challenges for Calibration

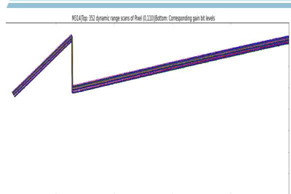
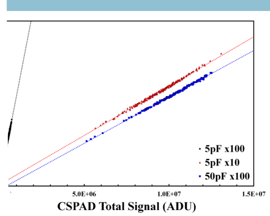
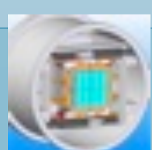
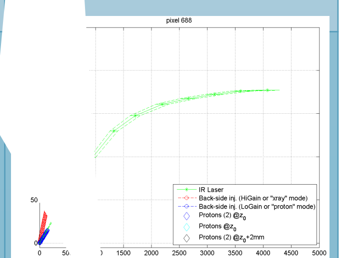
Detector	Specs	Gain Curve
AGIPD 	1 Mpixel, 4.5 M 352 memory ce 200µm sq. pixel 1-10⁴ ph@ 12 k 3 – 13 keV	
LPD 	1 Mpixel, 4.5 M 508 memory ce 500µm sq. pixe 1-10⁵ ph@ 12 k 5 – 25 keV	
DSSC 	1 Mpixel, 4.5 MH 800memory cells 204µm hex. pixe 1-10⁴ ph@ >1ke 0.5 – 6 keV	

- Three gains per pixels
- Offsets depend on signal
- Analogue gain evaluation
- ….

- Six gains per pixels
- Offset errors multiply
- Air scattering in FF
- ….

- Non-linear gain
- One ADU per photon
- ….

MHz Rate, High Dynamic Range Detectors – Challenges for Calibration

Detector	Gain Curve
AGIPD 	
LPD 	
DSSC 	

AGIPD:
- Three gains per pixels
- Offsets depend on signal
- Analogue gain evaluation
- ….

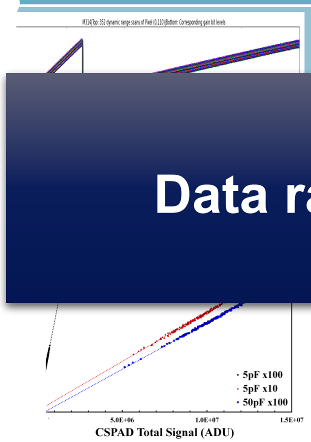
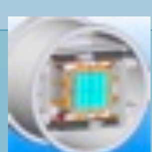
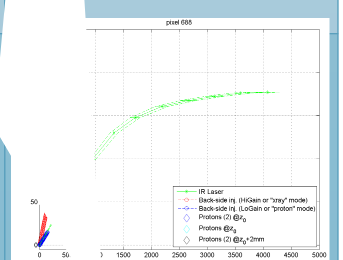
LPD:
- Six gains per pixels
- Offset errors multiply
- Air scattering in FF
- ….

DSSC:
- Non-linear gain
- One ADU per photon
- ….

Calibration is non-trivial!

Facility-side calibration recommended

MHz Rate, High Dynamic Range Detectors – Challenges for Calibration

Detector	Gain Curve
AGIPD 	
LPD 	
DSSC 	

- Three gains per pixels
- n signal
- aluation

Data rates: ~10-13 GB/s

- Six gains per pixels
- Offset errors multiply
- Air scattering in FF
- ….

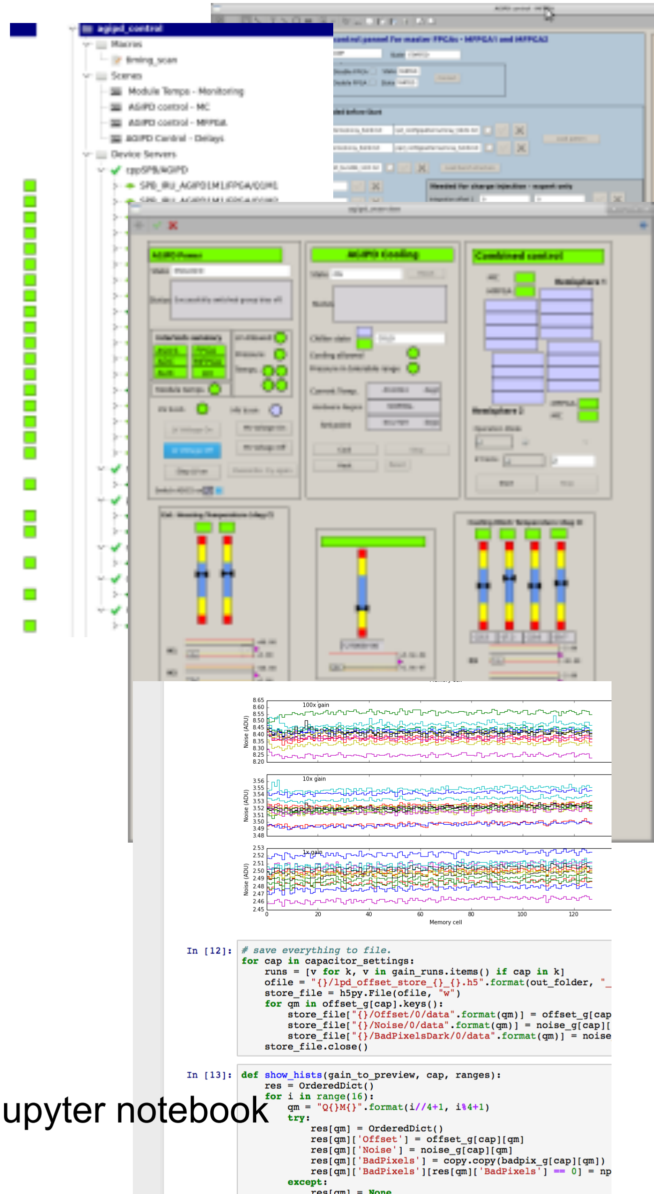
- Non-linear gain
- One ADU per photon
- ….

Calibration is non-trivial!

Facility-side calibration recommended

Tools we use

Karabo GUIs



Jupyter notebook

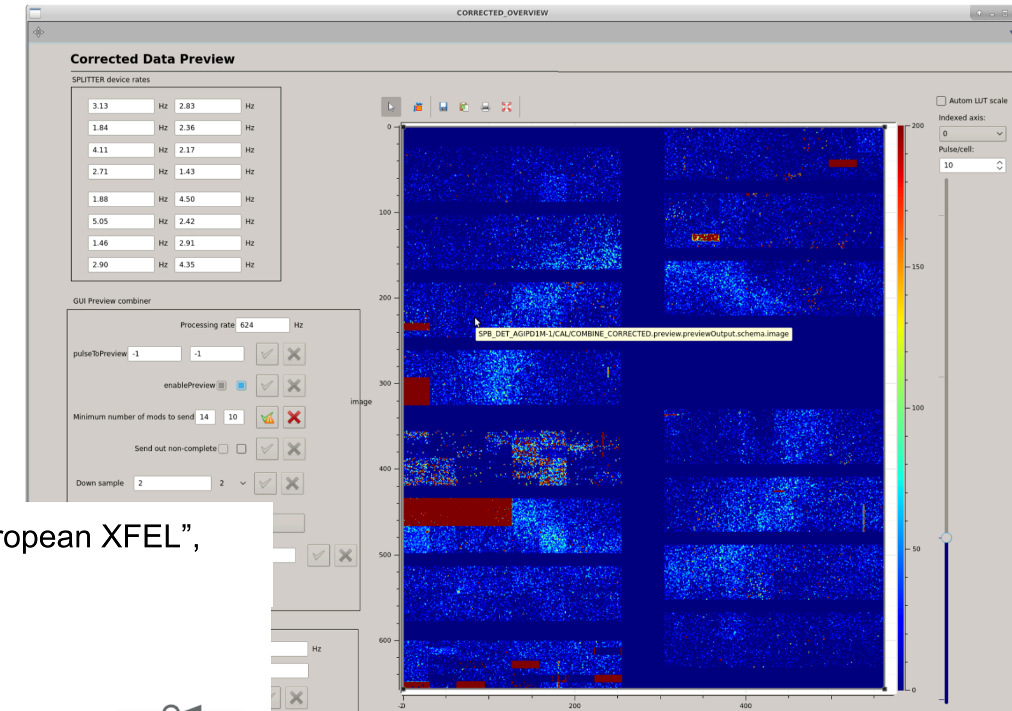
- Karabo – the European XFEL Control and Analysis Framework
  - DAQ, online corrections, multi-module combination, and user exposure are all implemented in this framework. See e.g. *Kuster et al.: Detectors and Calibration Concept for the European XFEL; Fangohr et al.: Data Analysis Support at European XFEL*
  - Detector Control is implemented in Karabo, working on top of consortia-provided libraries
  - Karabo now follows a 2-4 week release schedule, with bug fixes made quickly available
    - ► Many improvements in responsiveness during last 6 months as mandated with ever growing installations
    - ► DAQ and calibration and detector control follow relevant updates in these release cycles, leading to stability and performance improvements
    - ► Beam-time like conditions are a must for proper evaluation
- IPython and project Jupyter
  - For offline analysis and characterization
  - For transparency of what we do
- ZMQ
  - For exposing data to users via a well-established path

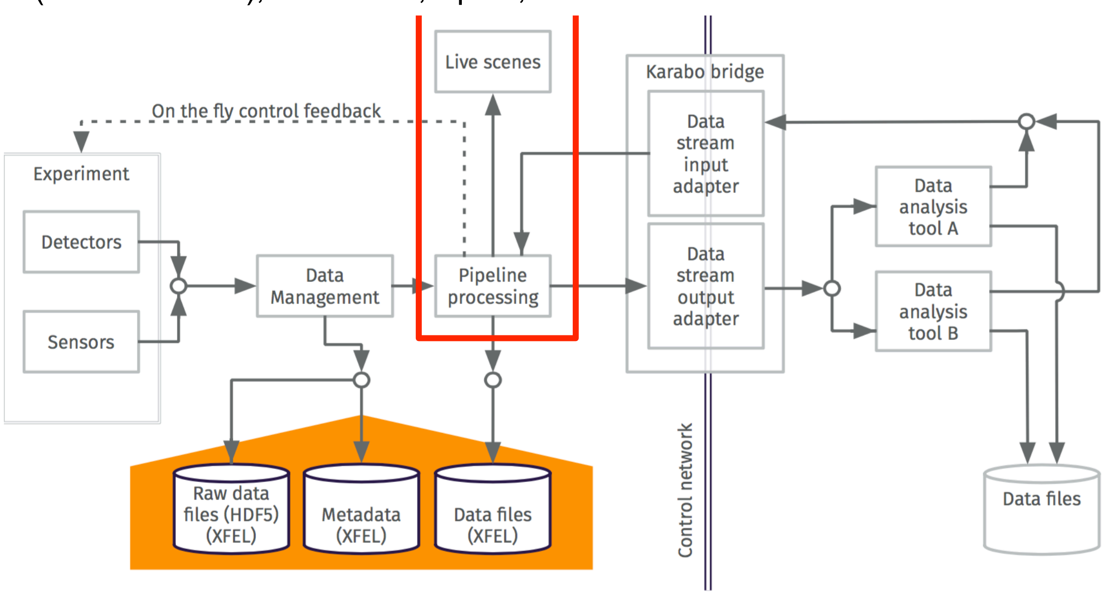
Online Calibration

- Has seen extensive use at both instruments for rapid feedback
- Feeds user-provided online tools via a Karabo-Bridge*
- GPU algorithms available if data rates require

*H. Fangohr et al., "Data Analysis Support in Karabo at European XFEL", (ICALEPCS'17), Barcelona, Spain, Oct. 2017

Corrected online preview for AGIPD

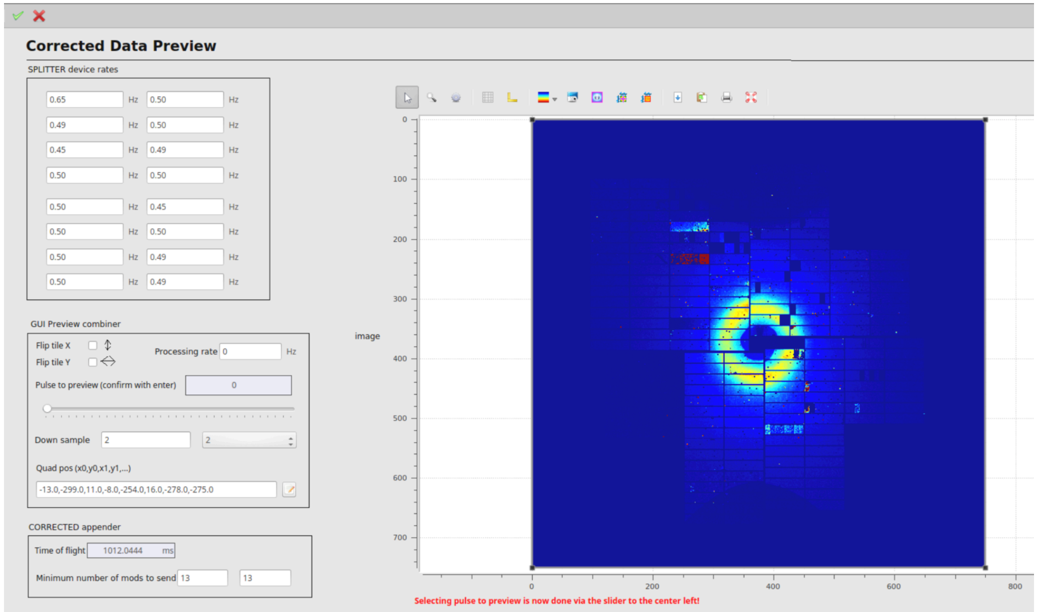




	AGIPD	LPD
Gain evaluation	X	X
Offset correction	X	X
Relative gain correction	X	X
Bad pixels	G/O/N	G/O/N
KRB devices for MPIX	103	69

Online Calibration

Corrected online preview for LPD



- Has seen extensive use at both instruments for rapid feedback
- Feeds user-provided online tools via a Karabo-Bridge
- GPU algorithms available if data rates require
- Online processing is module parallel, chunks size always one train
  - "Splitter" devices as entry points to pipelines from → pass every nth train, safe guard DAQ by dropping on slowness
  - "Combiner" device as exit points:
    - ► Combine modules into stack: [pulse, module, x y]
    - ► Combine modules into image: [pulse, x', y']

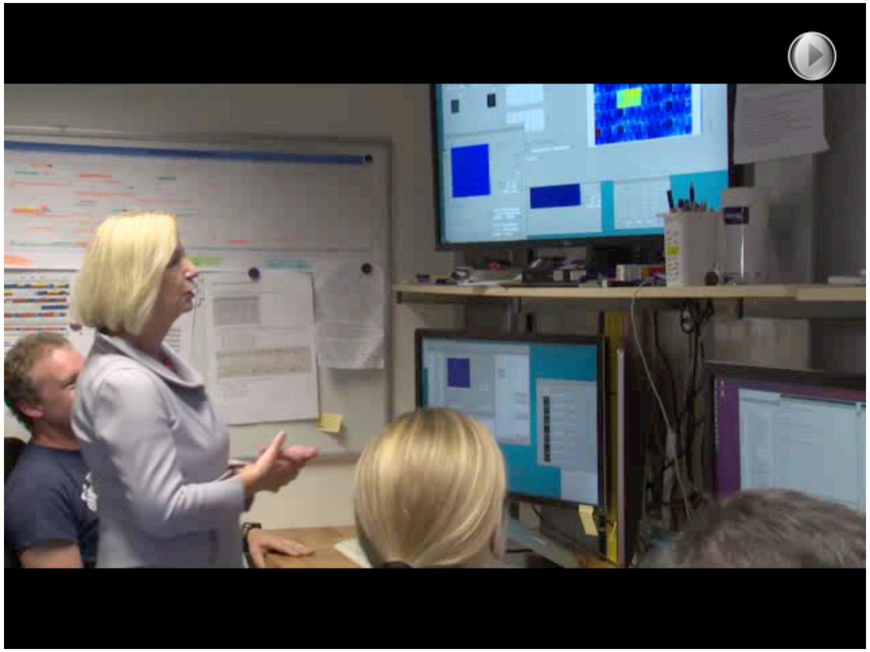
→

- Needs to combine 16 data streams @1.6 Gb/s tot.
- Can optionally mask BP
- Configurable number of modules to "wait" for

Online Calibration

- Has seen extensive use at both instruments for rapid feedback
- Feeds user-provided online tools via a Karabo-Bridge
- GPU algorithms available if data rates require
- Online processing is module parallel, chunks size always one train
  - “Splitter” devices as entry points to pipelines from → pass every nth train, safe guard DAQ by dropping on slowness
  - “Combiner” device as exit points:
    - ▶ Combine modules into stack: [pulse, module, x y]
    - ▶ Combine modules into image: [pulse, x’, y’]

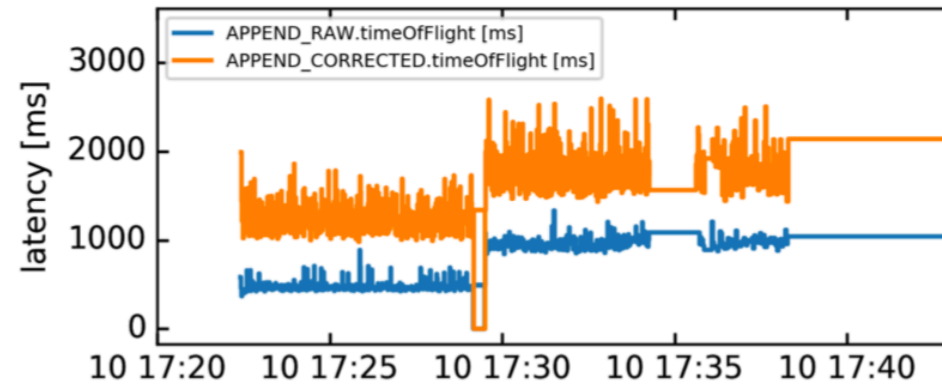
Corrected online preview for AGIPD

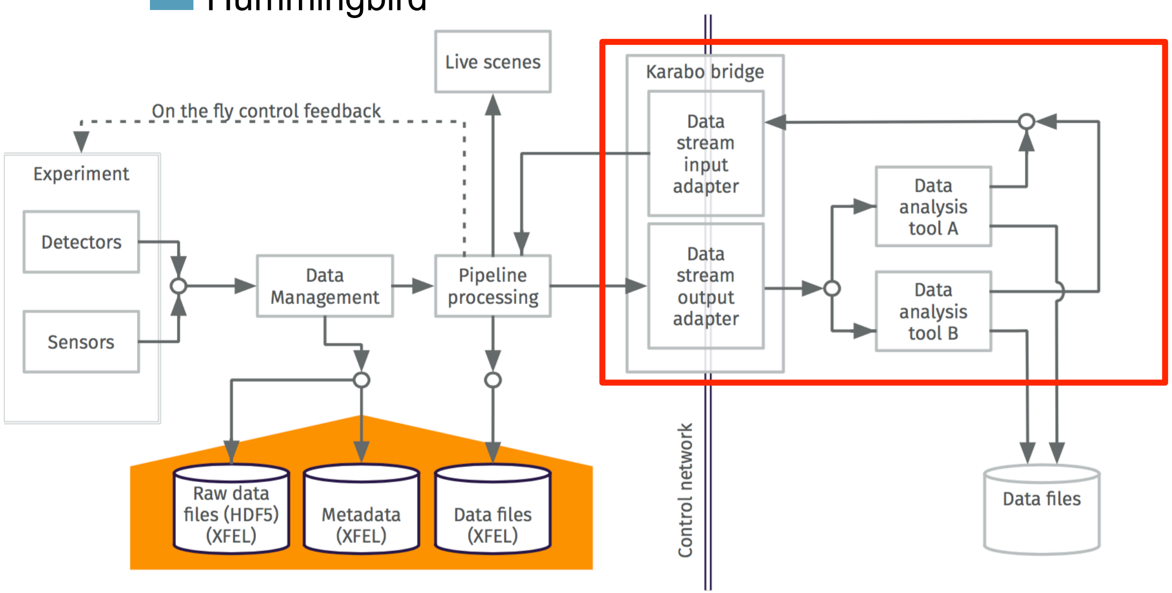


- Needs to combine 16 data streams @2 Gb/s tot.
- Can optionally mask BP
- Configurable number of modules to “wait” for

Online Calibration

- Feeds user-provided online tools via a Karabo-Bridge device (T. Michelat)
  - Usually combiner in appender mode
  - Connects to CAS-provided ZMQ bridge
  - 3-5 Hz rate at 128 cells, 256 cells at 1Hz
  - 2s latency with 256 memory cells
- Applications so far:
  - ONDA
  - CASS
  - Hummingbird





- Latency includes:
  - Data acquisition
  - Data formatting on DAQ
  - Data forwarding to pipelines
  - Data selection at pipeline entry points:
    - ► Every nth train
    - ► Cells containing FEL pules
  - Combining of 16 streams from modules
  - Data advertising on ZMQ

Python-based Calibration and Data Analysis Suite – Using GPUs via pyCUDA*

*Klöckner, Andreas, et al. "PyCUDA and PyOpenCL: A scripting-based approach to GPU run time code generation." *Parallel Computing* 38.3 (2012): 157-174.

- Basic correction algorithms usually of form $y = mx + b$
  - b: pedestal/offset
  - m: gain factor

However, values are per-pixel, memory, cell and gain stage, so a single detector module requires GB of calibration constants. No strict ordering of memory cells in a chunk of data can be assumed.

- Threshold pixel values to define separate output array:

$$x < T_1 \rightarrow y = 0,$$
$$T_1 \leq x < T_2 \rightarrow y = 1,$$
$$x \geq T_1 \rightarrow y = 2$$

Used for gain bit evaluation of AGIPD

Python-based Calibration and Data Analysis Suite – Using GPUs via pyCUDA*

*Klöckner, Andreas, et al. "PyCUDA and PyOpenCL: A scripting-based approach to GPU run time code generation." *Parallel Computing* 38.3 (2012): 157-174.

Allows for instance to run-time optimize GPU-kernels, e.g. the sorting networks used for common-mode computation

```
"""
Commonmode approach based on a sorting network using Batcher's Odd-Even Merge Sort algorithm.
Can be tuned to parallelize on a warp with (32 threads).

"""
def oddeven_merge(lo, hi, r):
    step = r * 2
    if step < hi - lo:
        for i in oddeven_merge(lo, hi, step):
            yield i
        for i in oddeven_merge(lo + r, hi, step):
            yield i
        for i in [(i, i + r) for i in range(lo + r, hi - r, step)]:
            yield i
    else:
        yield (lo, lo + r)
```

Python-based Calibration and Data Analysis Suite – Using GPUs via pyCUDA*

*Klöckner, Andreas, et al. "PyCUDA and PyOpenCL: A scripting-based approach to GPU run time code generation." *Parallel Computing* 38.3 (2012): 157-174.

Allows for instance to run-time optimize GPU-kernels, e.g. the sorting networks used for common-mode computation

```
def generateNetwork(datalength, returnStats):
    pairs_to_compare = list(oddeven_merge_sort(datalength))
```

....

```
code = code.replace('{{ stats_header }}', statsHeader)
code = code.replace('{{ stats_code }}', statsCode)


code = code.replace('{{ blockSize }}', str(cons.shape[1])).replace('{{ c_len }}', str(cons.shape[0]))

return code, cons.astype(numpy.int32)
```

Python-based Calibration and Data Analysis Suite – Using GPUs via pyCUDA

- Performance-optimization for correction and calibration routines → near real-time performance seams possible

GPU-based for "near-realtime" corrections:

Offset, common-mode, split-event, statistics)

(scaled from Nvidia K2200 to K40, requires approx. 1 GPU per 10G line)

Corresponds to one 10G line (AGIPD, DSSC, LPD)

Detector	Data size (pixel × frames)	Processing time (ms)
AGIPD	$128 \times 512 \times 352$	76.6
DSSC	$128 \times 512 \times 512^a$	106.1
LPD	$256 \times 256 \times 512$	103.6

[a] Limited to 512 frames by the train–builder hardware, the detector head can store up to 800 frames.

Python-based Calibration and Data Analysis Suite – Using GPUs via pyCUDA

- Performance-optimization for correction and calibration routines → near real-time performance seams possible

Calibration constants on GPU amount to ~2-4GB in size per module. Memory thus can be limiting factor.

GPU-based for "near-realtime" corrections:

Offset, common-mode, split-event, statistics)

(scaled from Nvidia K2200 to K40, requires approx. 1 GPU per 10G line)

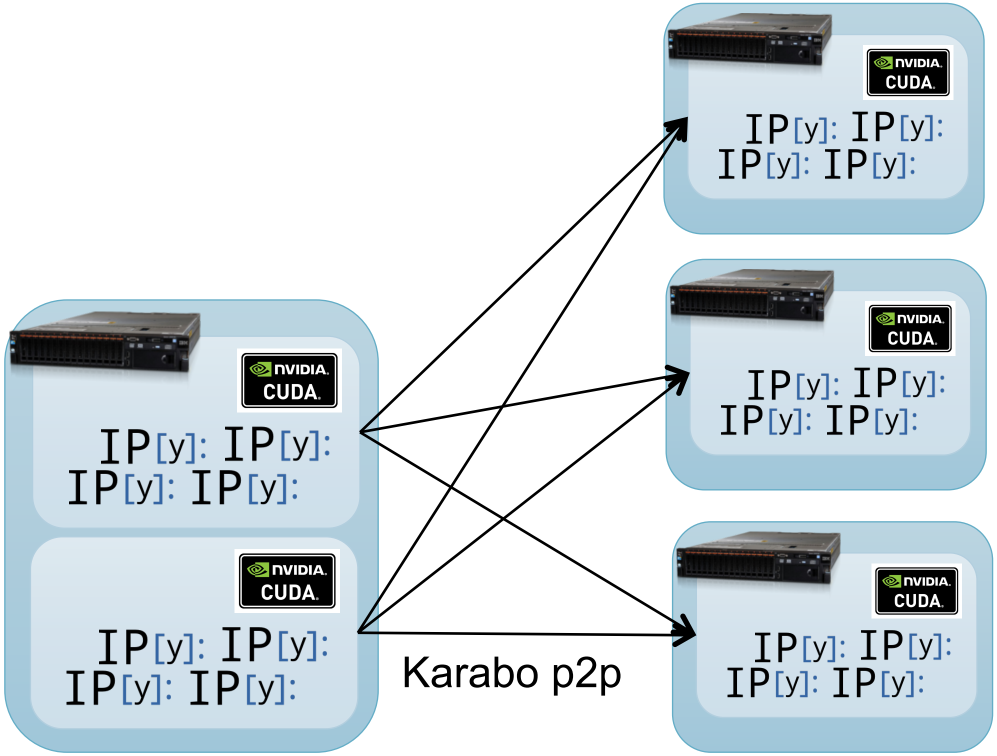
Corresponds to one 10G line (AGIPD, DSSC, LPD)

Detector	Data size (pixel × frames)	Processing time (ms)
AGIPD	$128 \times 512 \times 352$	76.6
DSSC	$128 \times 512 \times 512^a$	106.1
LPD	$256 \times 256 \times 512$	103.6

[a] Limited to 512 frames by the train–builder hardware, the detector head can store up to 800 frames.

Where Karabo comes into play

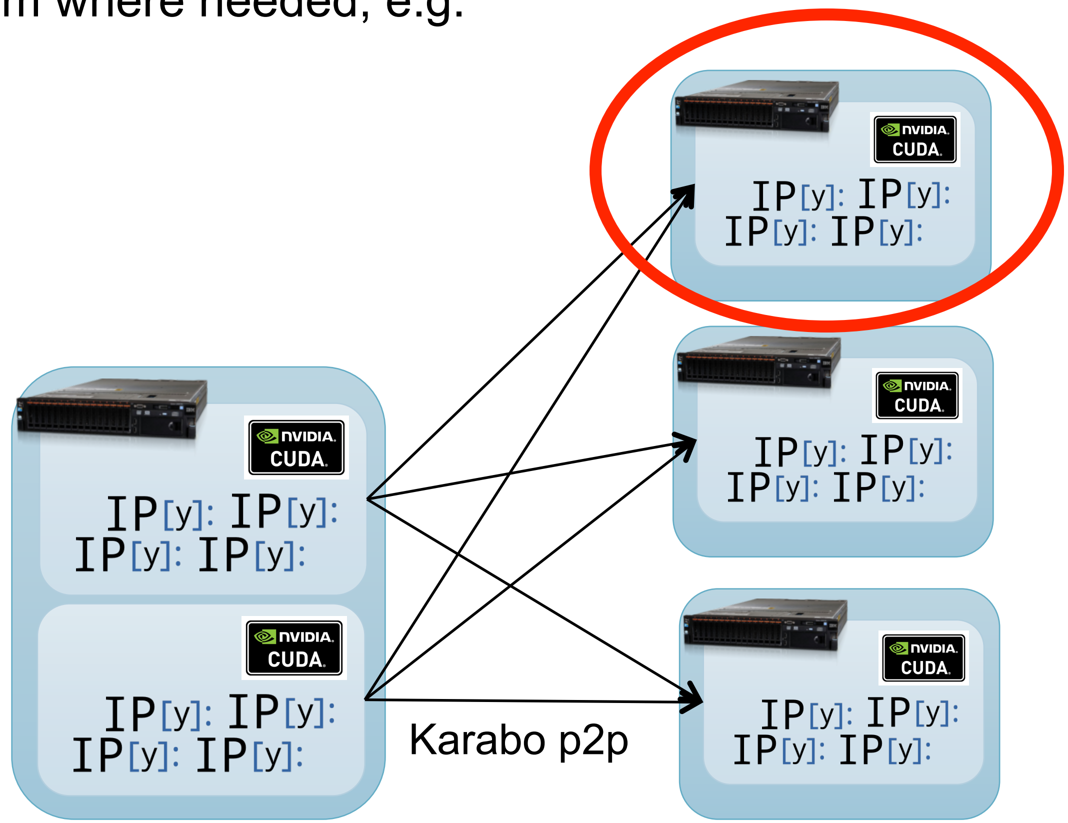
- Chose Karabo p2p for distributed computing
  - Allows for native integration of Karabo-data formats
  - Allows for easy integration into Karabo-GUI
  - Direct access to control-system where needed, e.g. detector operating conditions



Where Karabo comes into play

- Chose Karabo p2p for distributed computing
  - Allows for native integration of Karabo-data formats
  - Allows for easy integration into Karabo-GUI
  - Direct access to control-system where needed, e.g. detector operating conditions

Keep all GPU-related processing for one module on the same host and avoid copying data

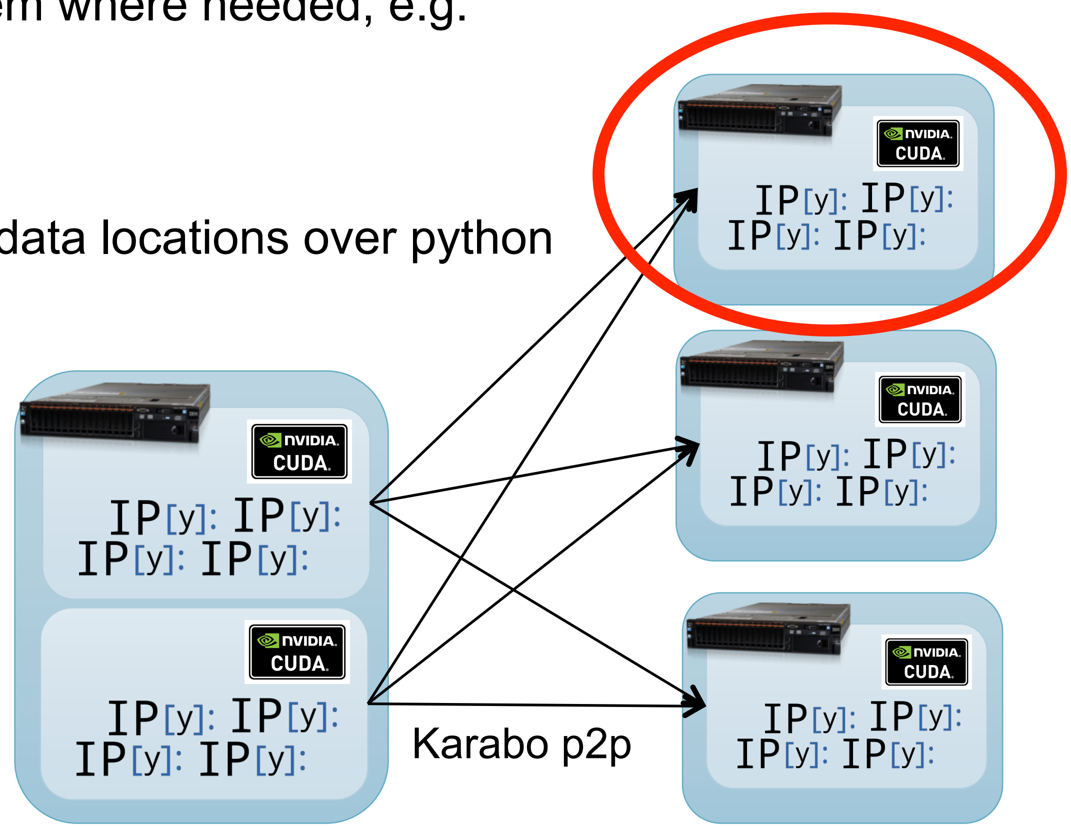


Where Karabo comes into play

- Chose Karabo p2p for distributed computing
  - Allows for native integration of Karabo-data formats
  - Allows for easy integration into Karabo-GUI
  - Direct access to control-system where needed, e.g. detector operating conditions

Keep all GPU-related processing for one module on the same host and avoid copying data

- Use IPC handles to pass GPU data locations over python process boundaries.
  - Works well, but:
    - ▶ Documentation could be better concerning how to deallocate properly.



Summary

- GPUs are in production use at European XFEL for online calibration of data from our MHz imaging rate detectors
  - Up to 256 Mpixel images/s achieved (250 MB/s on each GPU, 20% peak utilization)
  - Bottleneck not on GPU but in data combining, as a single host at some point has to handle 2GB/s or more in the future
  - Reasonable performance for now, but continuous optimization
  - Currently simple configurable pipeline of linear equations and thresholding, but more evolved corrections can be deployed as needed (common mode, split event correction).
  - PyCuda makes CUDA GPU integration easy within our Python framework (Karabo)

- Next steps:
  - More throughput (always on the wish list)
  - Add characterization routines: offset, noise, thresholds → implemented on GPU and tested offline, but interfaces in online environment need to be integrated
  - Smaller pixel detectors are now coming online (mostly 10Hz), which can profit from GPU-base split-event corrections

- Wish list
  - Better documentation and examples on CUDA IPC
  - Tutorials on RDMA with CUDA via Infiniband