Application of GPUs for Online Monitoring in Tomography

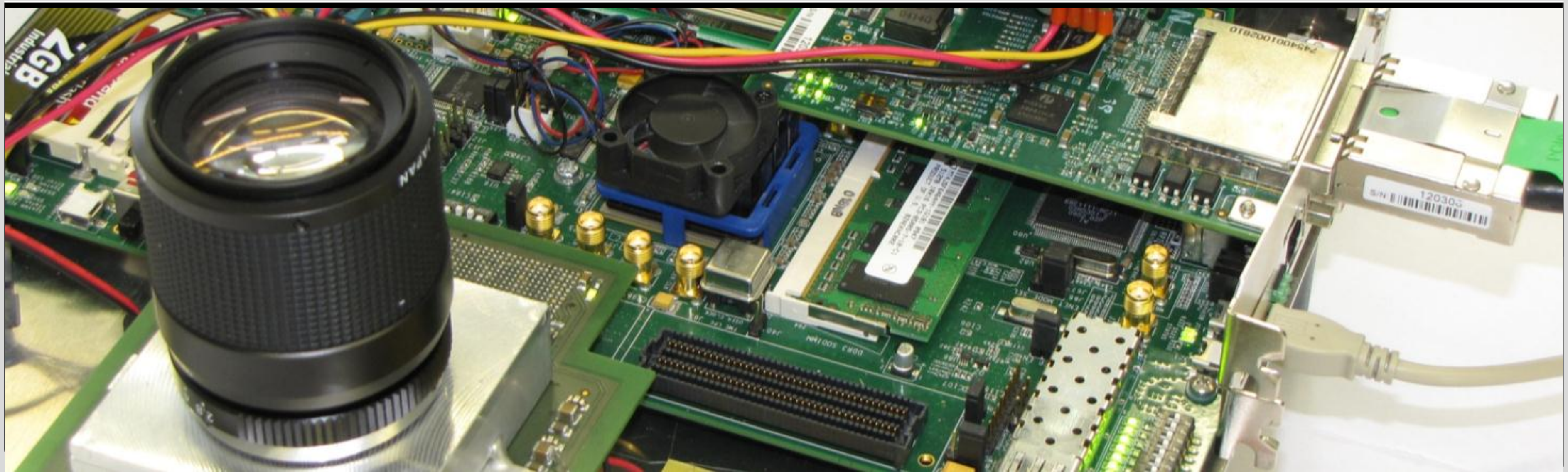
Andreas Kopmann
Karlsruhe Institute of Technology

ANKA Synchrotron Radiation Facility and Institute for Data Processing and Electronics

Application of GPUs for Online Monitoring in Tomography

Andreas Kopmann
Karlsruhe Institute of Technology

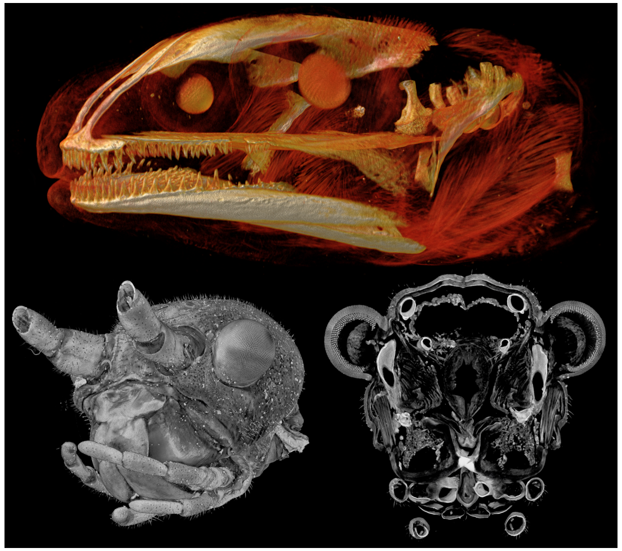
ANKA Synchrotron Radiation Facility and Institute for Data Processing and Electronics

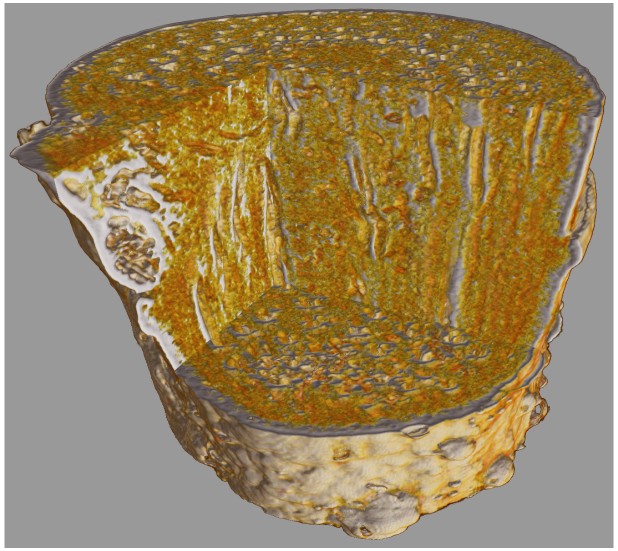


Motivation

- One main application field is Biology
- We want to analyze small animals
- Recording motion is desired (more information)

- Started UFO project: *Ultra-Fast X-ray Imaging of Scientific Processes with On-line Assessment and Data-driven Process Control*
- Result will be a new beamline at ANKA
- Commissioning will start beginning of next year

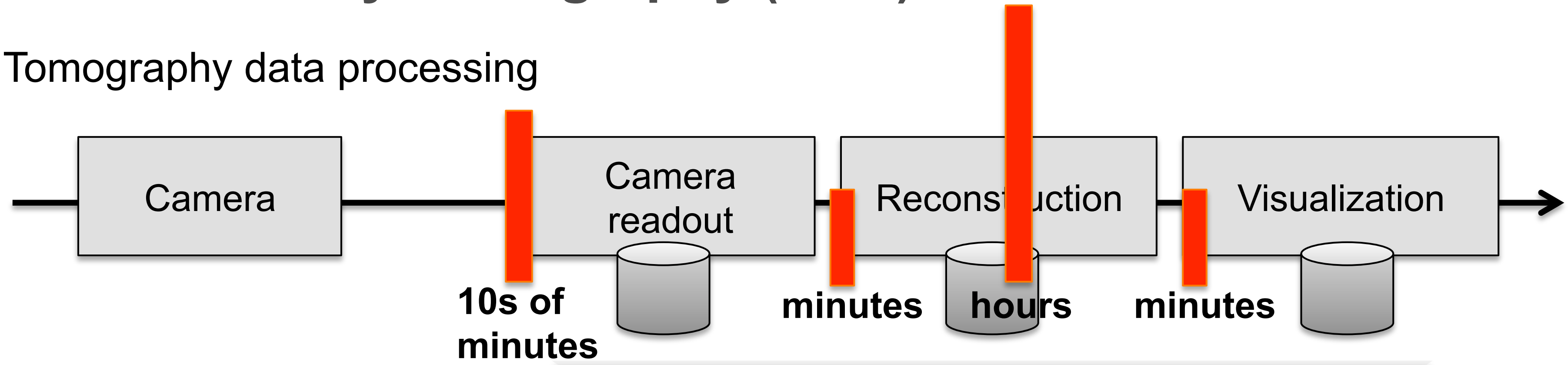




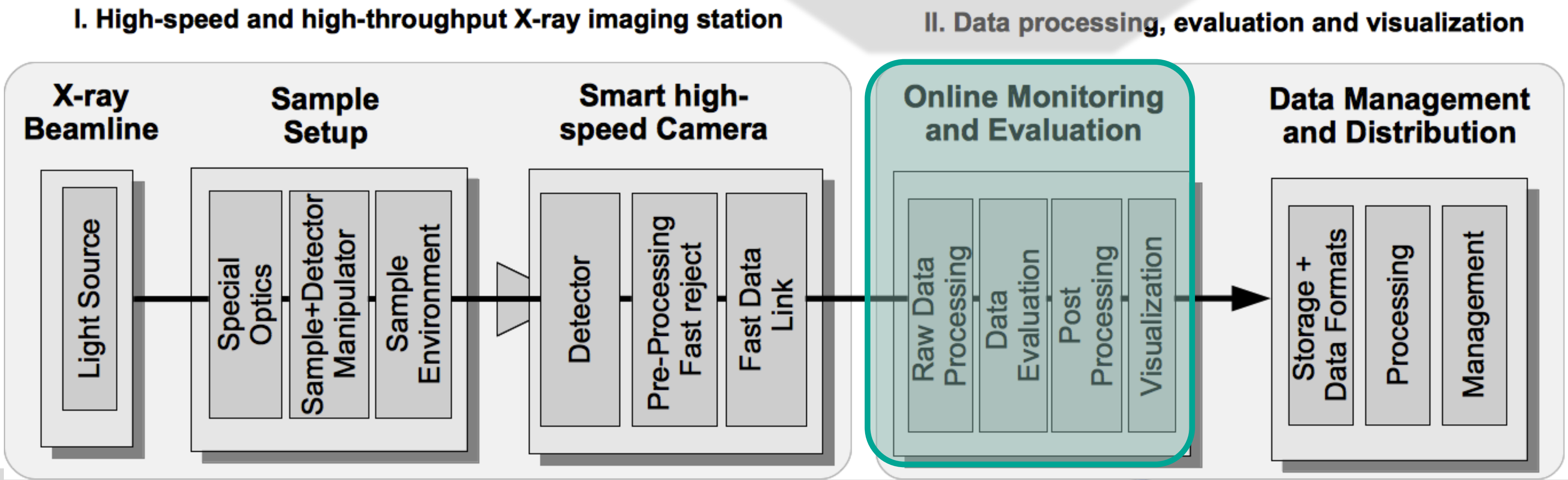
From top to bottom: newt larva (*Euproctus platycephalus*), stick insect (*Peruphasma schultei*), bamboo segment

Ultrafast X-ray Tomography (UFO)

Tomography data processing

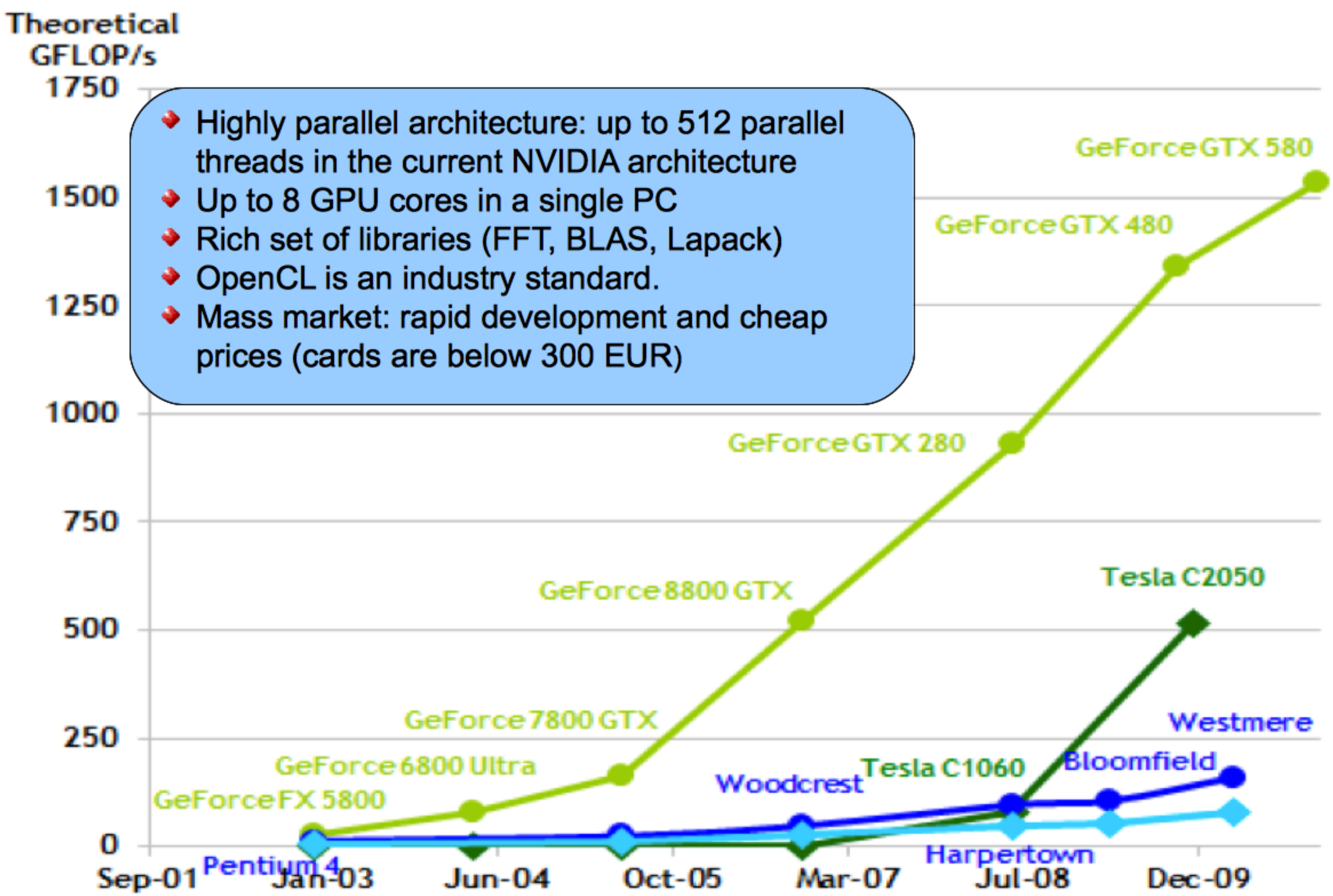


New: streaming setup



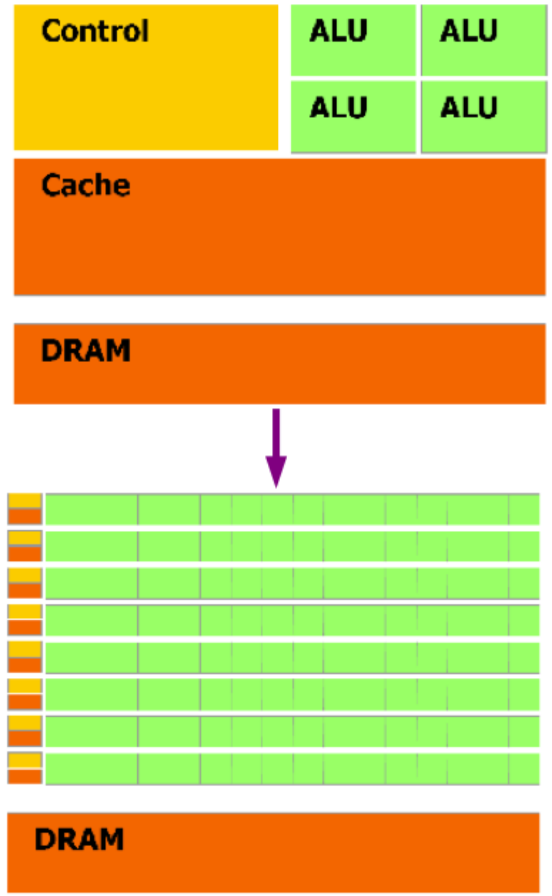
GPU Computing

- Highly parallel architecture: up to 512 parallel threads in the current NVIDIA architecture
- Up to 8 GPU cores in a single PC
- Rich set of libraries (FFT, BLAS, Lapack)
- OpenCL is an industry standard.
- Mass market: rapid development and cheap prices (cards are below 300 EUR)



Challenges in GPU Computing

- Highly parallel architecture: up to 512 parallel threads in the current NVIDIA architecture
- Up to 16 GPU cores in a single PC
- Rich set of libraries (FFT, BLAS, Lapack)
- OpenCL is an industry standard.
- Mass market: rapid development and cheap prices (cards are below 300 EUR)



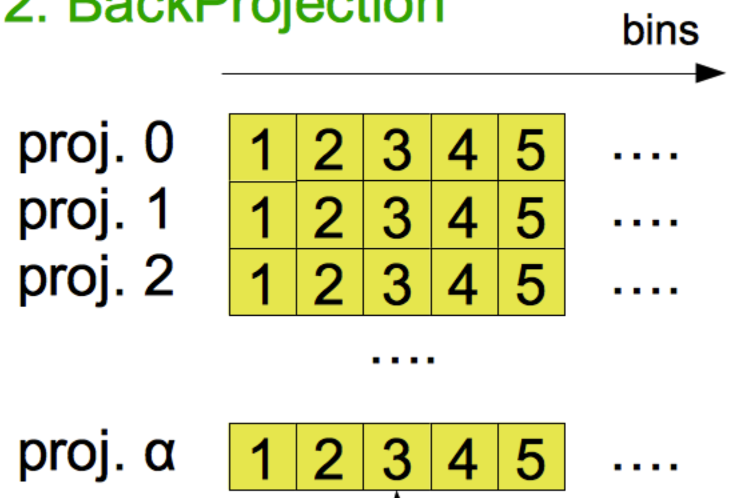
- **Data Transfer (only ~ 6 GB/s between GPU and memory)**
  - Reduce amount of data transfers
  - Use pinned memory for transfer if possible
  - Interleave data transfers with computations
- **Memory (multiple hierarchies, limited cache)**
  - Memory allocation & cleanup are expensive
  - Data alignment is crucial, data padding may help
  - Use shared memory and follow memory access patterns
- **Arithmetics (only a few specific instructions are fast)**
  - Avoid doubles & integers
  - MADD – two operation at cost of one
  - Use texture engine for interpolation and caching

Reconstruction with Filtered Backprojection

1. Filtering

Multiplication with the configured filter in the Fourier space

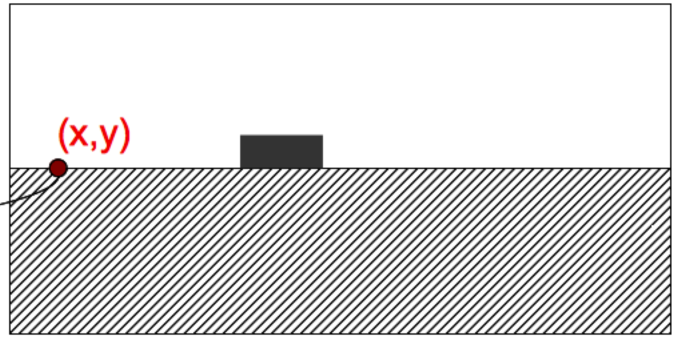
2. BackProjection



1. For each position we compute:
   $x \bullet \cos(\alpha) - y \bullet \sin(\alpha)$
2. Interpolate between neighboring bins
3. Sum over all projection
4. The sum is the value of (x,y)

$x \bullet \cos(\alpha) - y \bullet \sin(\alpha)$

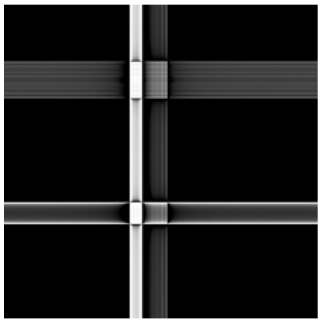
For each texel of output volume and for each projection we perform a single linear interpolation



Reconstruction with Filtered Backprojection

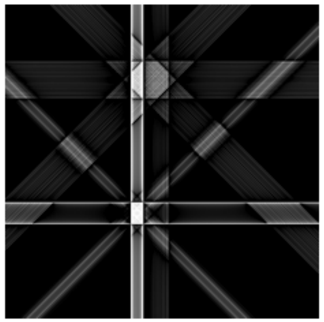
Filtered back-projection is used to produce 3D images from a manifold of two dimensional projections**.** Vertical slices are processed independently. For each slice all projections are smeared back onto the cross section along the direction of incidence yielding an integrated image.

n = 1

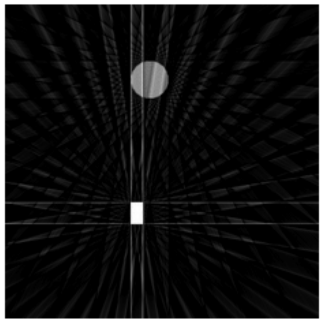


n = 2

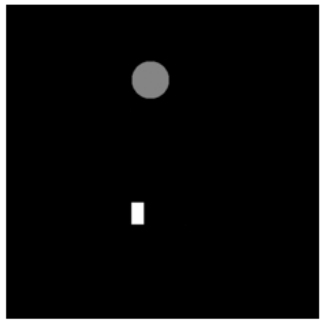
n = 3



n = 10

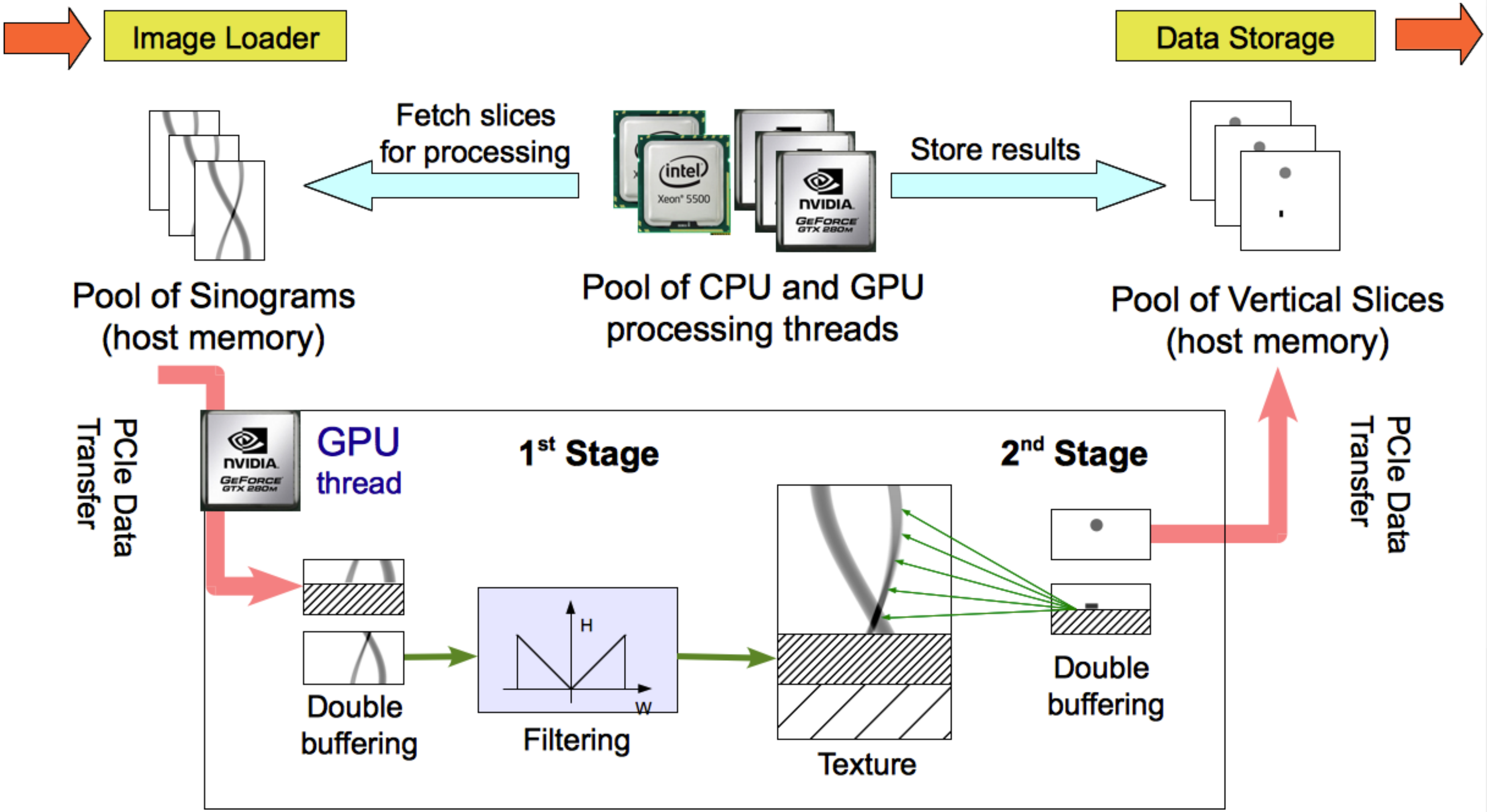


n = 50



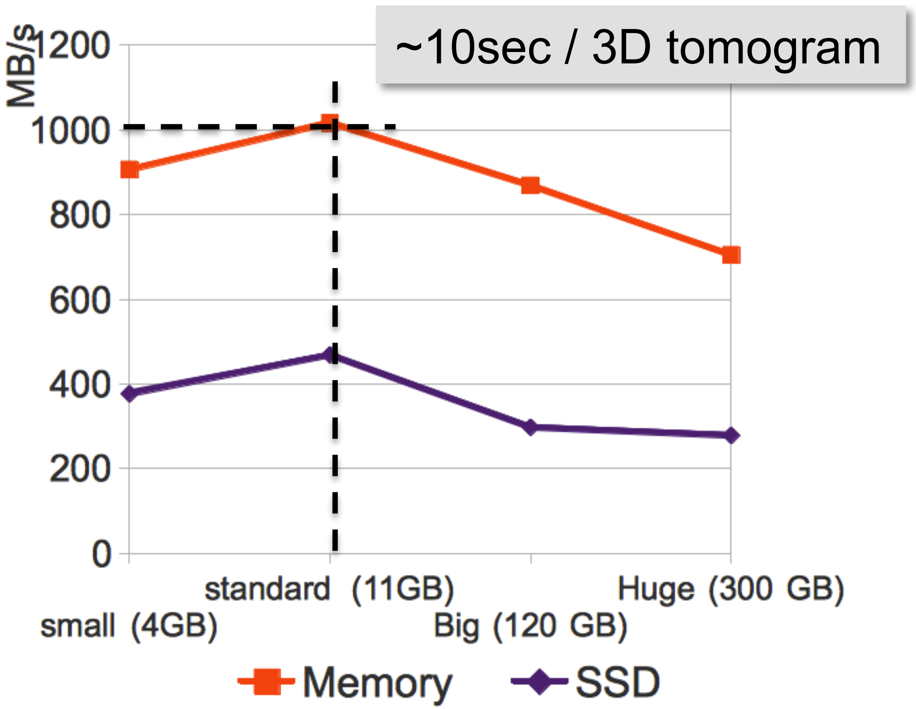
n = 1000

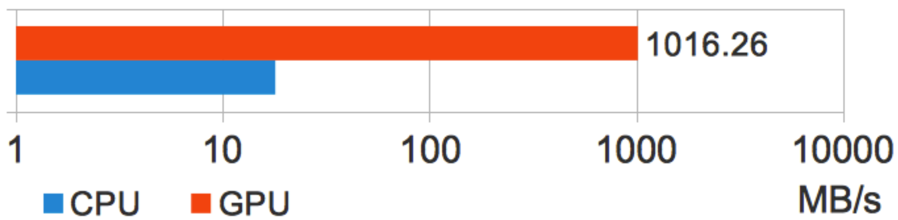
Parallel-beam Filtered Back Projection (FBP)

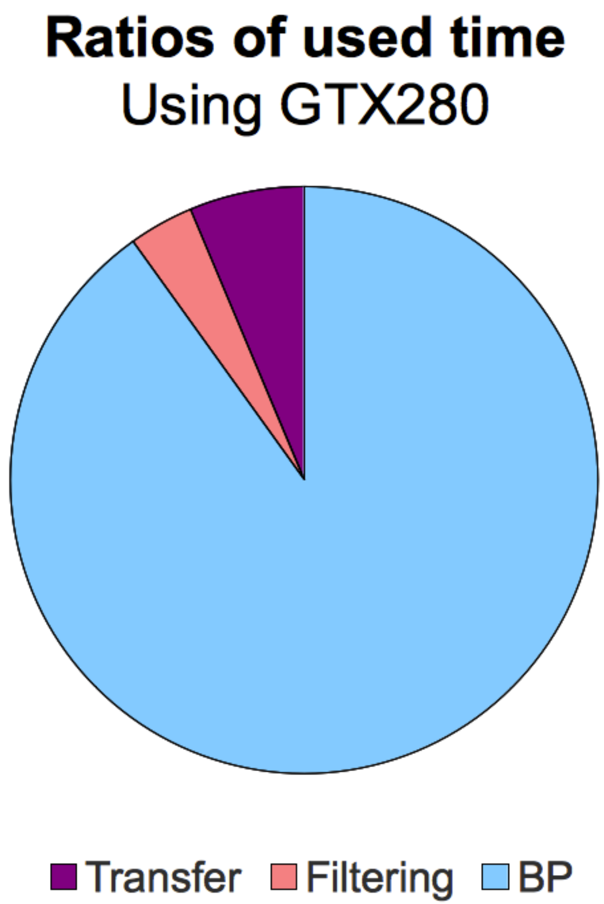


Performance GPU Computing

Example: tomographic reconstruction

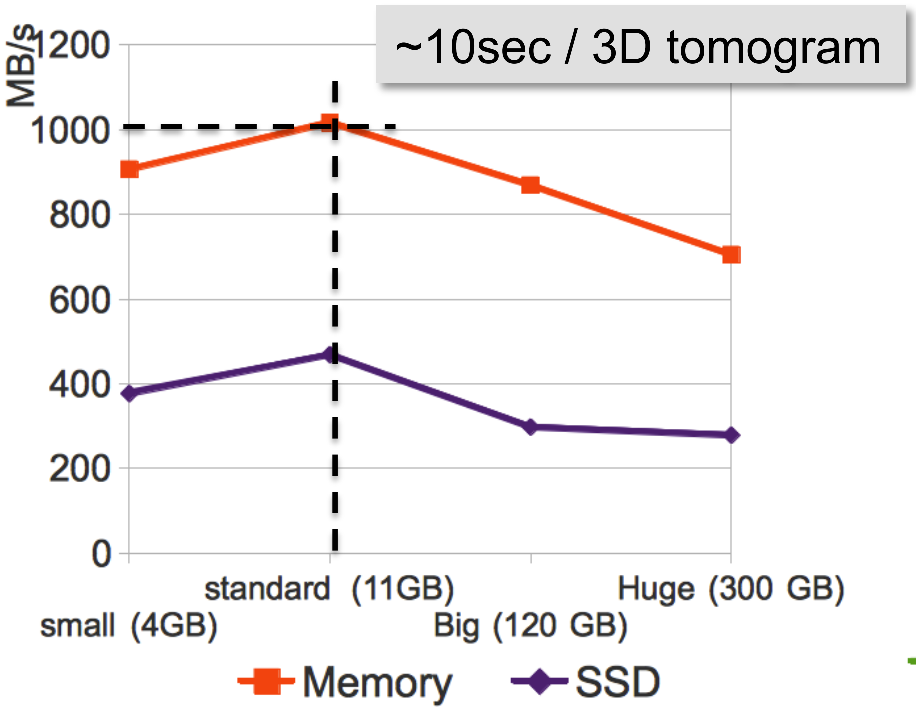


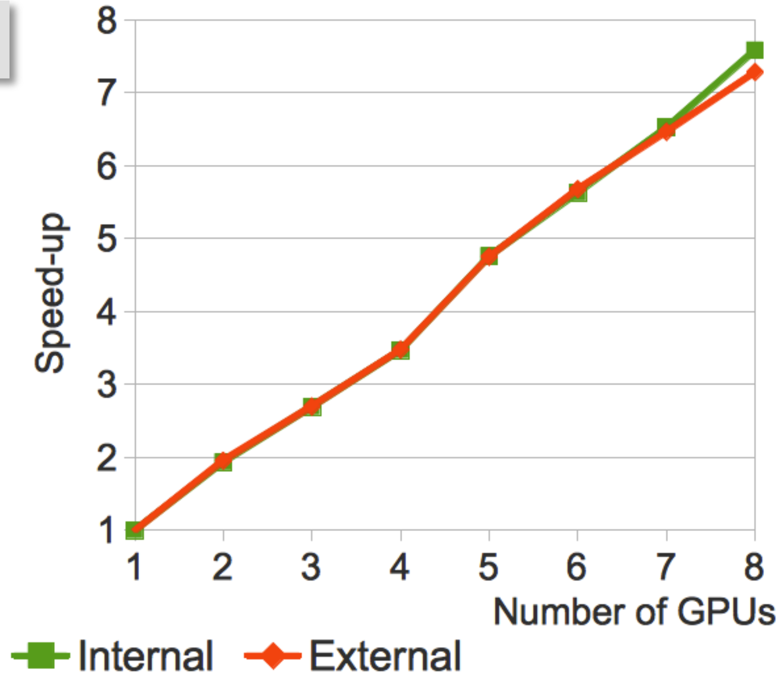


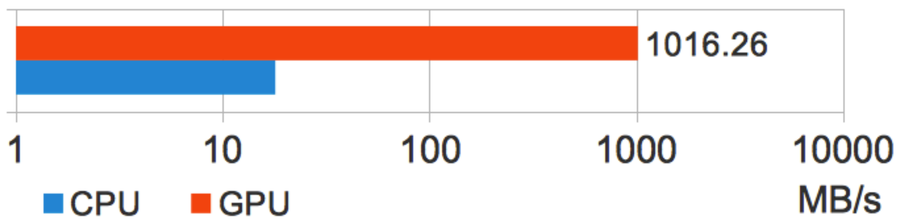


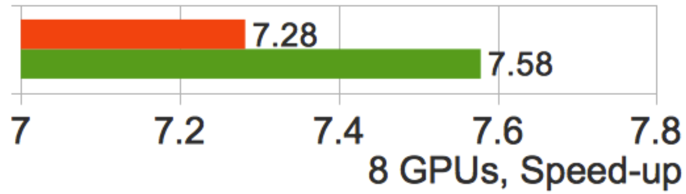
Performance GPU Computing

Example: tomographic reconstruction









- GPUs are useful for tomography
  → Bottleneck removed

Fermi Optimization

Computational Units

GT280	GTX580	Speedup
240 x 1.3 GHz	512 x 1.55 GHz	**2.5 x**
48 GT/s	49.4 GT/s	**1.0 x**

Texture Engine

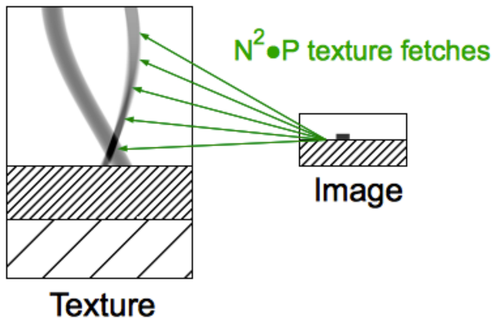
16 px × 16 px Thread block

$$v = x \bullet \cos(\alpha) - y \bullet \sin(\alpha)$$

$$\max_{x,y<N}(v) - \min_{x,y<N}(v) < N\sqrt{2}$$

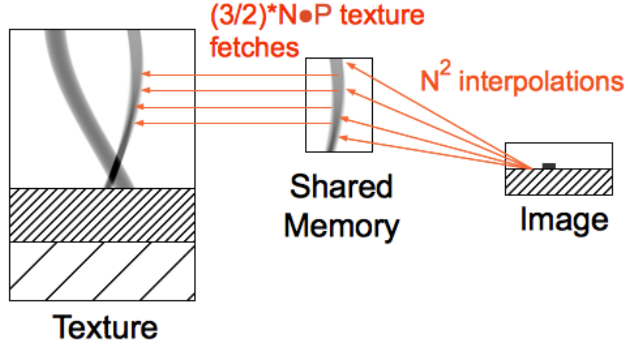
$$N\sqrt{2} < 1.5\,N$$

Each block of threads accesses actually only 3 • N / 2 bins per projection



Standard Version

Texture engine is heavily loaded



Fermi-optimized Version

Both texture & computations engines are used

Tuning for current GPU Architectures

Karlsruhe Institute of Technology

NVIDIA

GT200

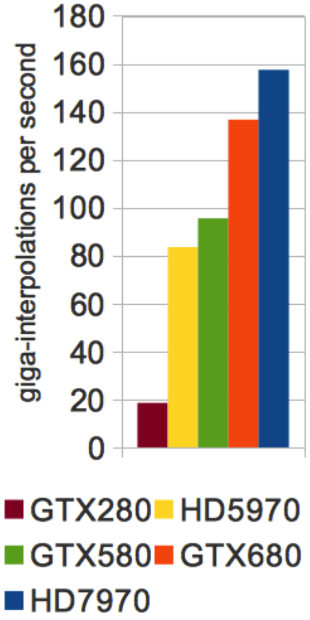
Base version
Uses texture engine

Fermi +100% GTX580

High computation power, but low speed of texture unit
Reduce load on texture engine: use shared memory to cache the fetched data and, then, perform linear interpolation using computation units.

Kepler +75% GTX680

Low bandwidth of integer instructions, but high register count
Uses texture engine, but processes 16 projections at once and 16 points per thread to enhance cache hit rate



AMD

VLIW +530% HD5970

Executes 5 independent operations per thread
Computes 16 points per thread in order to provide sufficient flow of independent instructions to VLIW engine

GCN +95% HD7970

High performance of texture engine and computation nodes
Balance usage of texture engine and computation nodes to get highest performance

- Optimization increases the benefit
  → How to support development?

UFO – Parallel Computing Framework: Requirements and Constraints

- Fast image processing of streams of n-dimensional data
- Open platform for shared algorithm development

- **Easily usable** by beamline scientists
- **Easily extendable** by developers

- Open source / Linux
- Must process 32-bit SP floating point
- Integration of legacy software
- Interface for automatic scheduling

- You are invited to use and contribute!

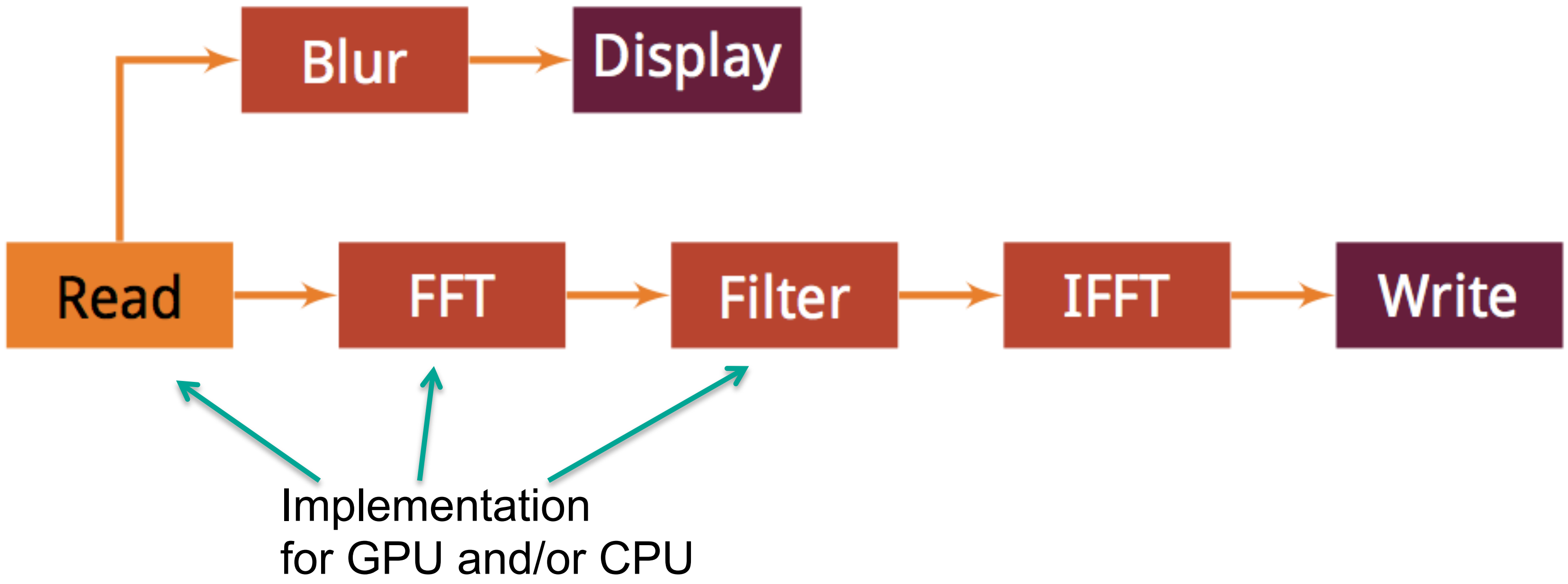
Yet Another Image Processing Framework?

- Other toolkits, frameworks and libraries lacked:
  - Available and modifiable source code
  - Continued development
  - GPU support
  - Streamed processing
  - Support for single-channel, floating point images

- What is the UFO toolkit?
  - A library written in C99 using GObject and **OpenCL**
  - A toolbox for scientists accessible through variety of managed languages
  - A platform for developers
  - Split into a core part and plugins

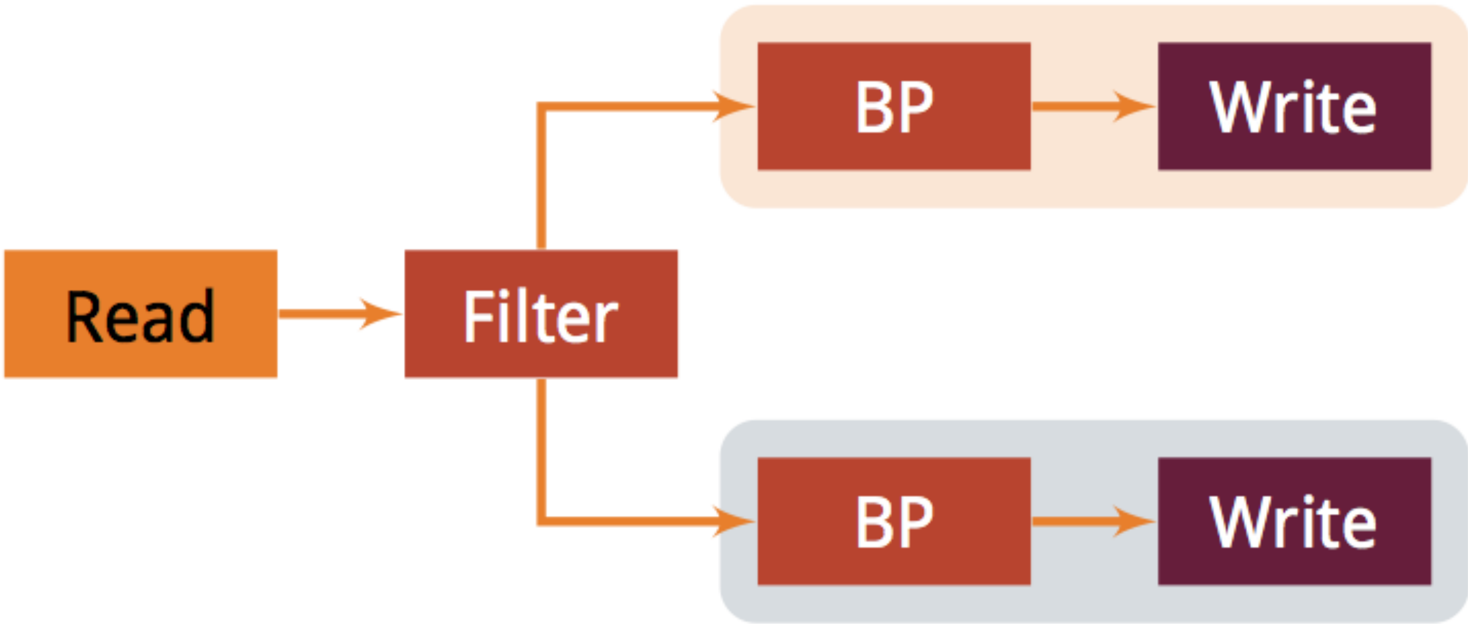
Architecture

- Compose processing algorithms as graphs of nodes
- Nodes process input data and generate output data
- Edges describe the flow from one output port to another input port



Multi-GPU Execution

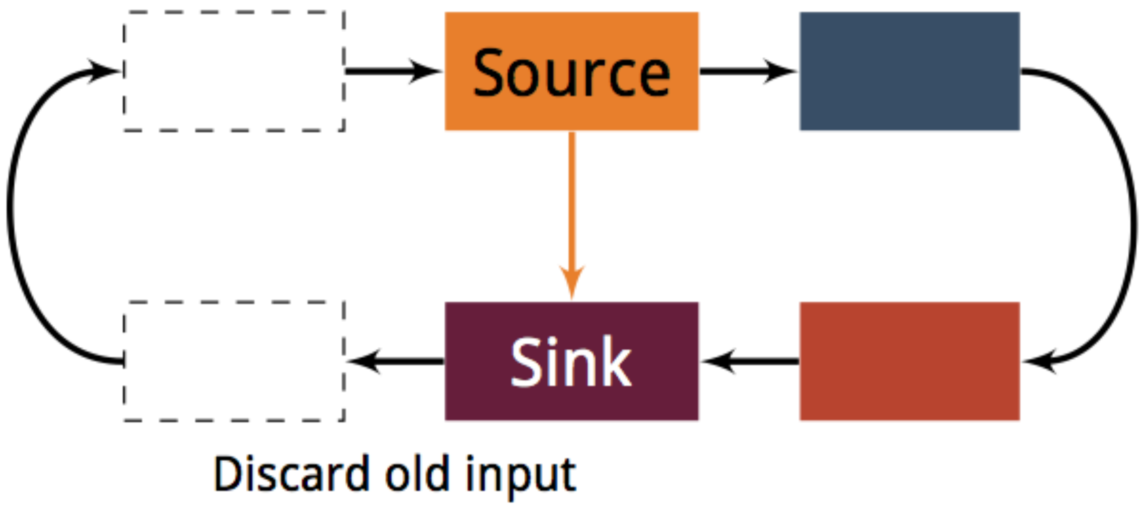
- Execute each node in its own thread
- Assign the same GPU device on identical branches
- Assign CPU if node has low computational demands or does not supply GPU implementation



Data Flow

- Allocate a fixed amount of buffers and push them into asynchronous queues to be written into and read from
- Re-use the same buffer object
- Initiate transparent transfer between host and device when a GPU node requests data that was produced by a CPU node and vice versa

- Result: → simpler synchronization and less memory copies



Implementation of Nodes

- Most nodes are stateful, thus they must be initialized:
  - ```
    void node_init(Node *self) {

            /* initialize necessary member variables */

    }
    ```
- Execution is performed in kernel style:
  - ```
    void node_process_gpu(Node *self,
            Buffer *input[], Buffer *output[],
            cl_command_queue cmd_queue)
    {
            cl_mem *in_mem = get_device_array(input[0]);
            cl_mem *out_mem = get_device_array(output[1]);

            /* Do something with the input
               and write results into output ... */

    }
    ```

Usage

- In Python a simple FFT round trip would be

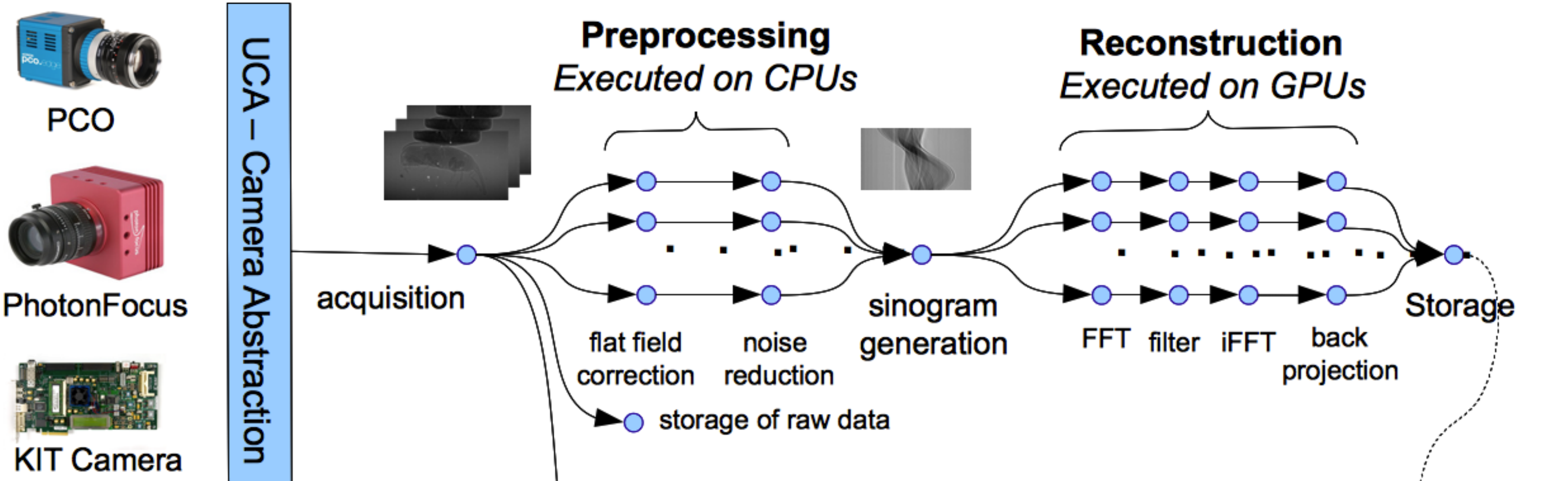
```
from gi.repository import Ufo

g = Ufo.Graph()
reader = g.get_plugin('reader')
writer = g.get_plugin('writer')
fft = g.get_plugin('fft')
ifft = g.get_plugin('ifft')

reader.connect_to(fft) # connect two nodes
fft.connect_to(ifft)
ifft.connect_to(writer)

g.run() # execute multi-threaded and GPU accelerated
```

UFO-Framework



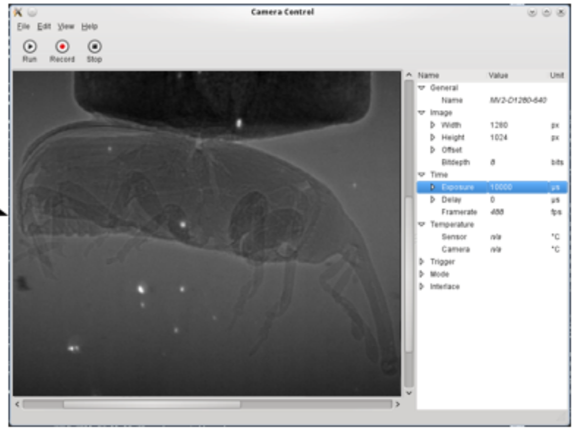
Sample code

```
G = Ufo.Graph()

cp = g.get_filter('copy')
rd = g.get_filter('uca-reader')
fbp = g.get_filter('fbp)
wr = g.get_filter('writer')

fbp.set_properties(axis, angle)

rd.connect_to(fbp)
fbp.connect_to(wr)
g.run()
```

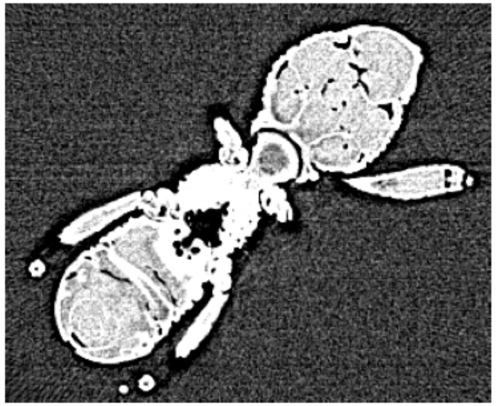


Camera control

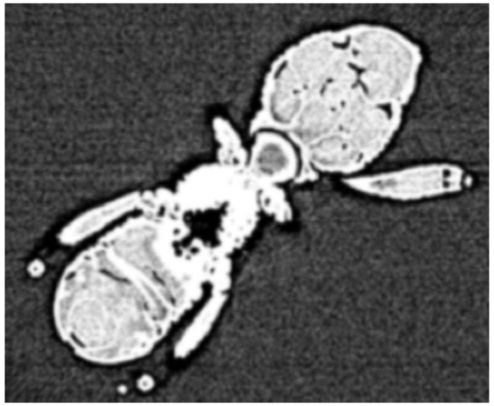
Segmentation / meshing

Example: Non-local Means Filtering

- Used to pre-process radiographs taken under low-light (= high-speed) conditions
- Denoised pixel is the weighted sum of all patch areas in a search window around that pixel
- We used p=3, w=21and σ=10
- Reference CPU code is parallelized with OpenMP



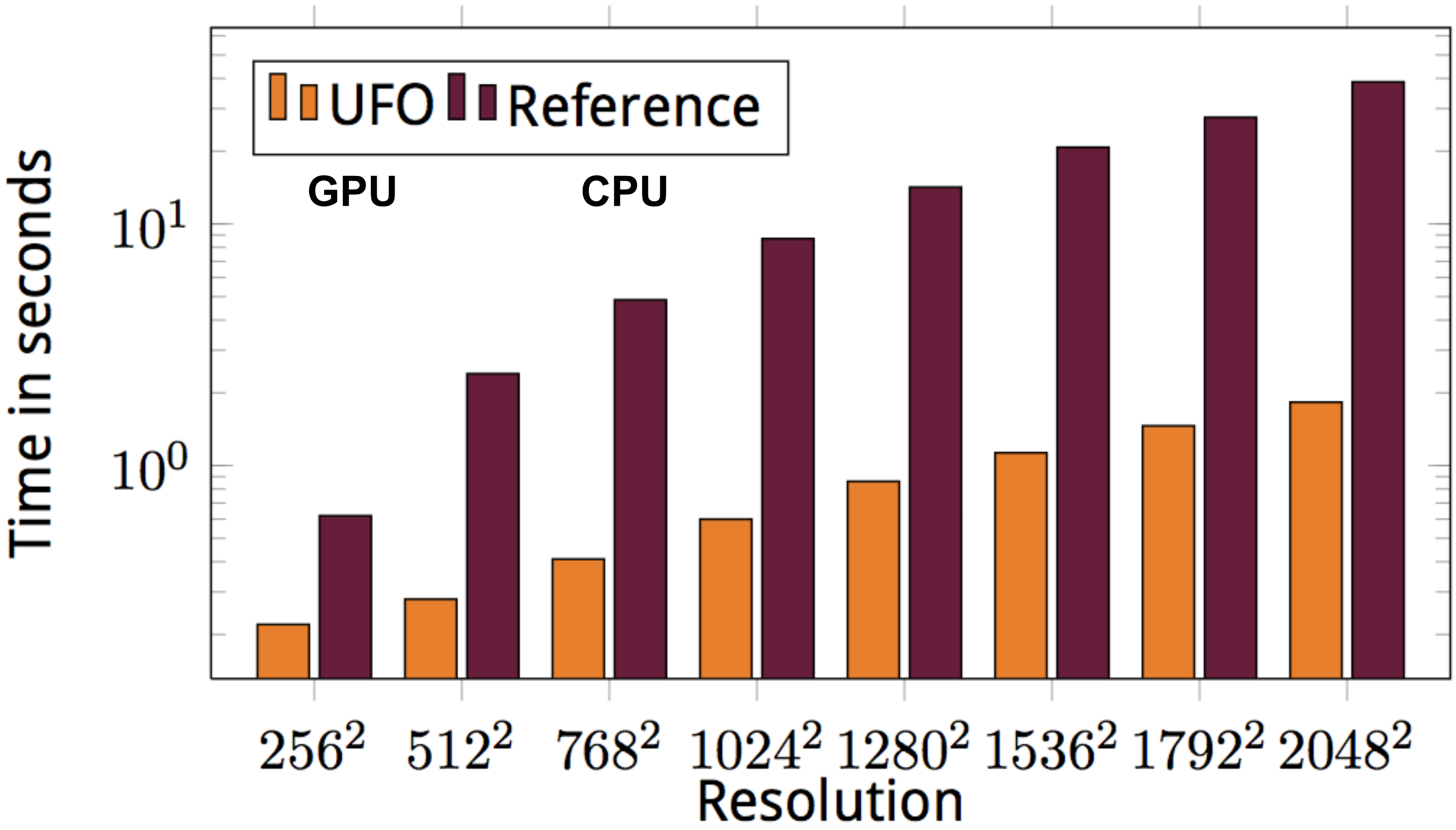
Acquired Image



Gaussian Filter

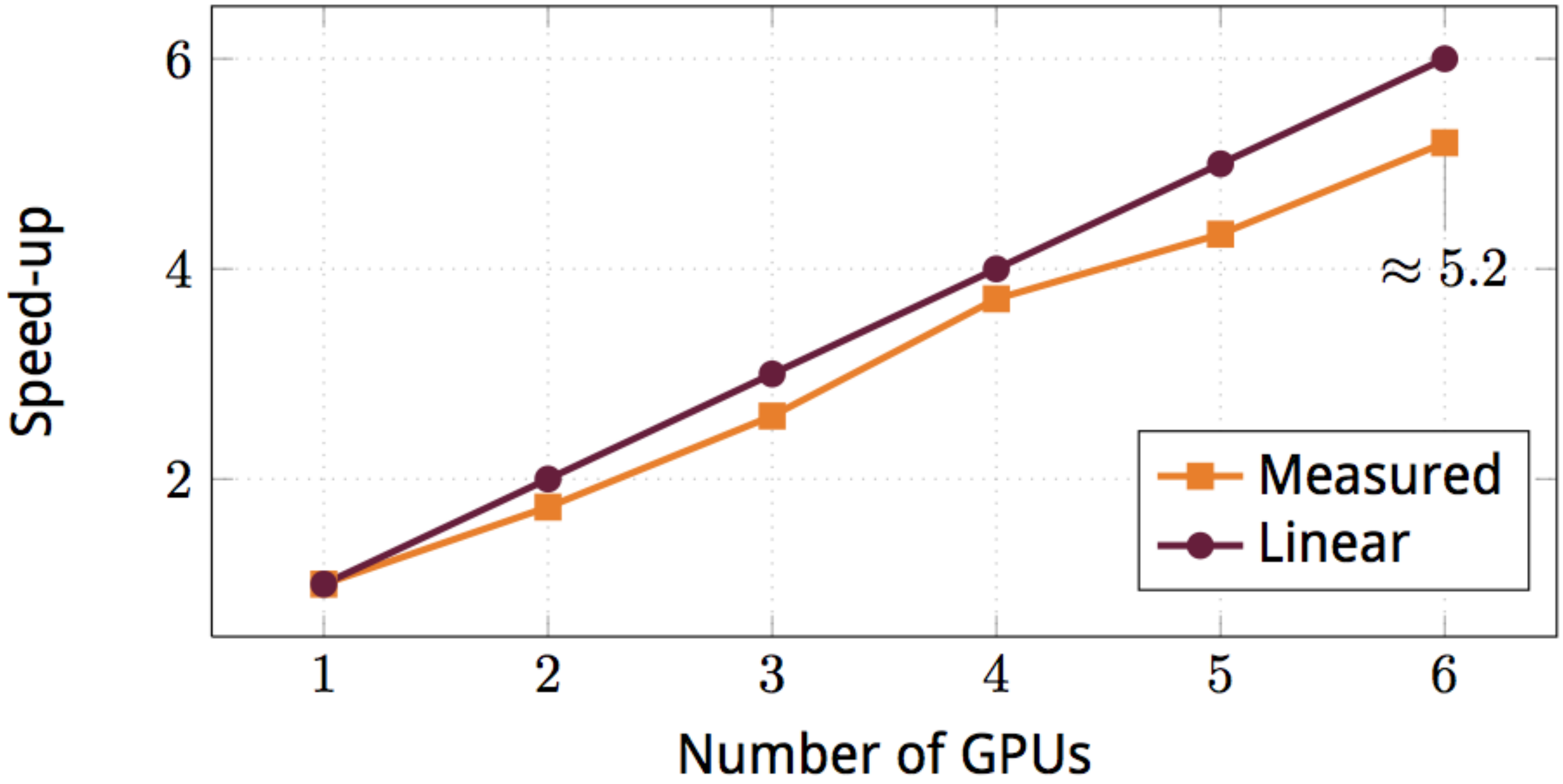
NLM Filter

Processing Time



- Single GPU is 10x faster than parallel COPU code

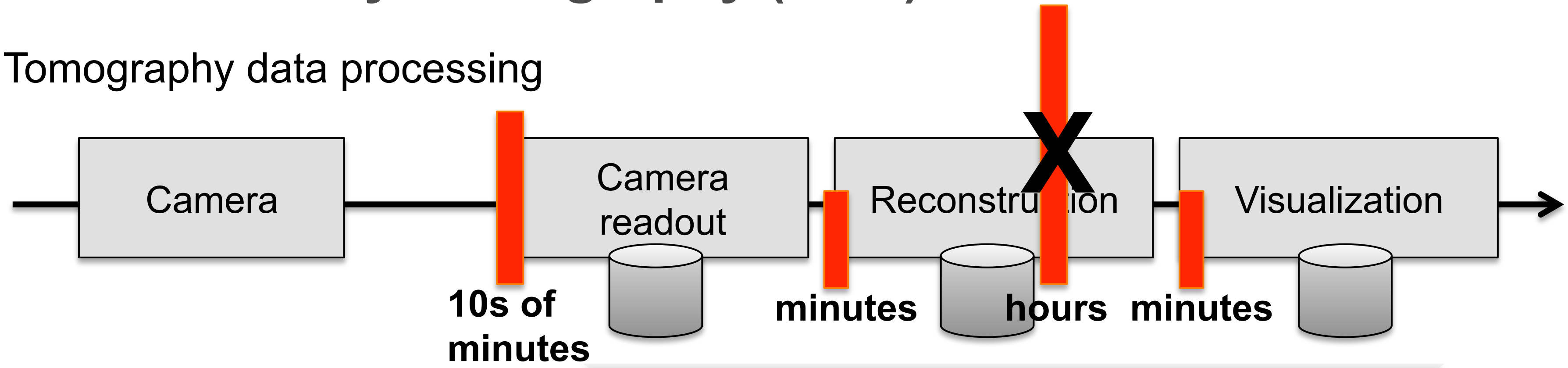
Performance: Multi-GPU Speed-up



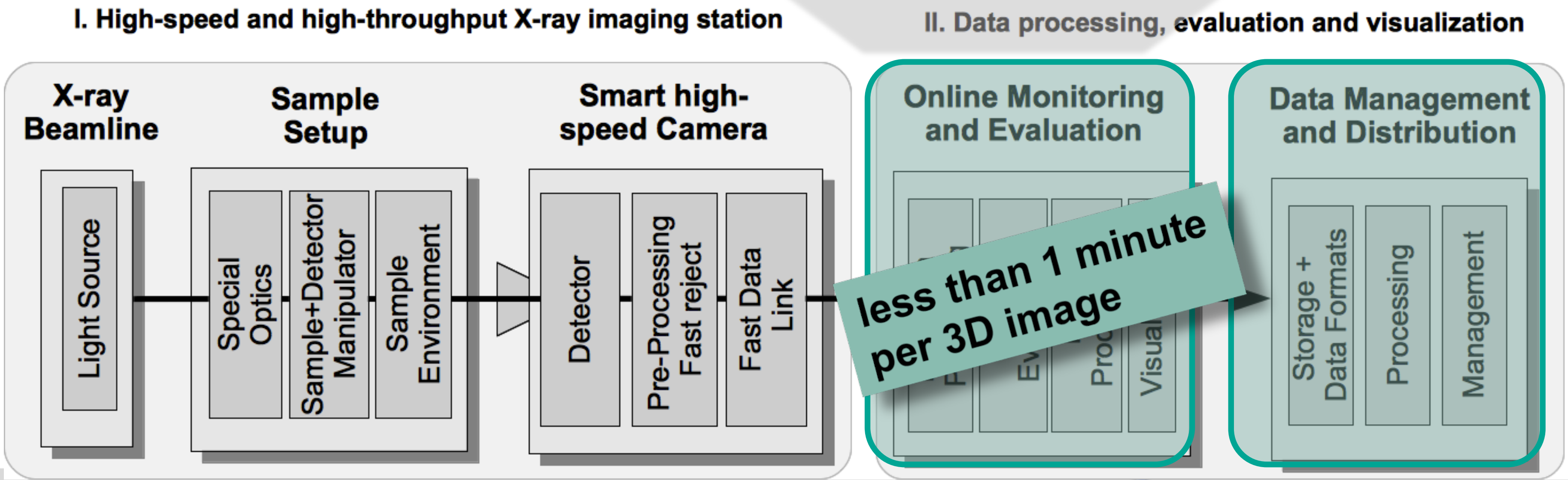
- GPU code scales nearly linear → much better than CPU
- OpenCL is useful now
- UFO-Framework: http://ufo.kit.edu

Ultrafast X-ray Tomography (UFO)

Tomography data processing



New: streaming setup



Hardware Platform for Online Monitoring

Karlsruhe Institute of Technology

Camera

PCO.edge
PCO.dimax
PCO.4000

CameraLink
850MB/s

Ethernet
10 Gb/s

Storage

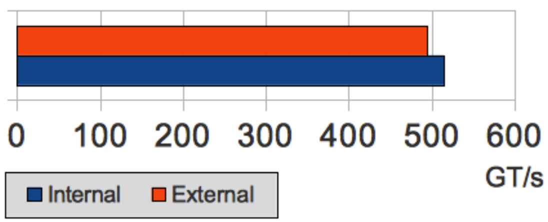
LSDF
Large Scale Data Facility

External PCIe x16 (8 GB/s)

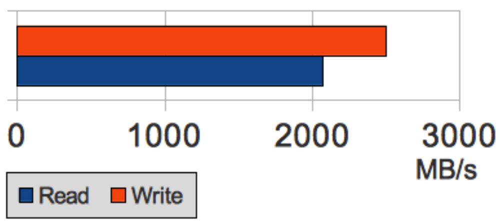
SFF8088 (2.4 GB/s)

SuperMicro 7046GT-TRF (Dual Intel 5520 Chipset)
CPU: 2 x Xeon X5650 (total 12 cores at 2.66 Ghz)
GPUs: 4 x GTX590 External
Memory: 96 GB / 12 DDR3 slots (192GB max)
Network: Intel 82598EB (10 Gb/s)
Camera Link Frame Grabber (850 MB/s)
Storage: Areca ARC-1880-ix-12 SAS Raid
16 x Hitachi A7K200 (Raid6)
8 x Intel SSD 510 (Raid0)

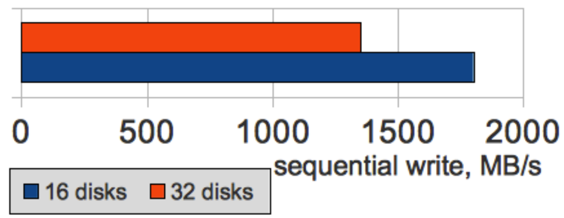
External GPU Box



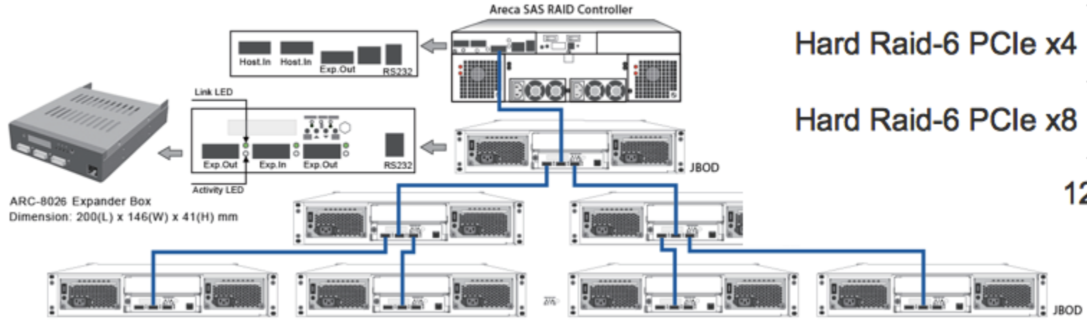
SSD Raid

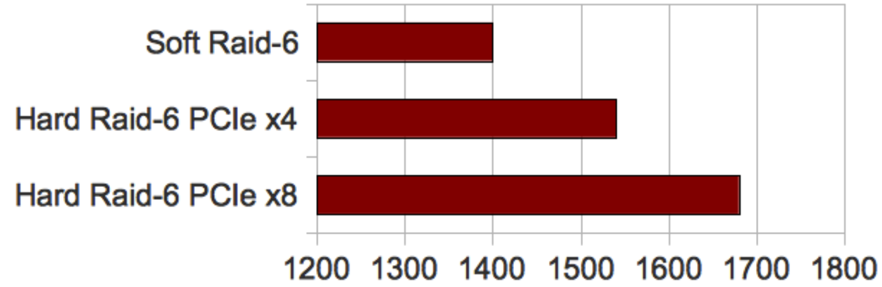


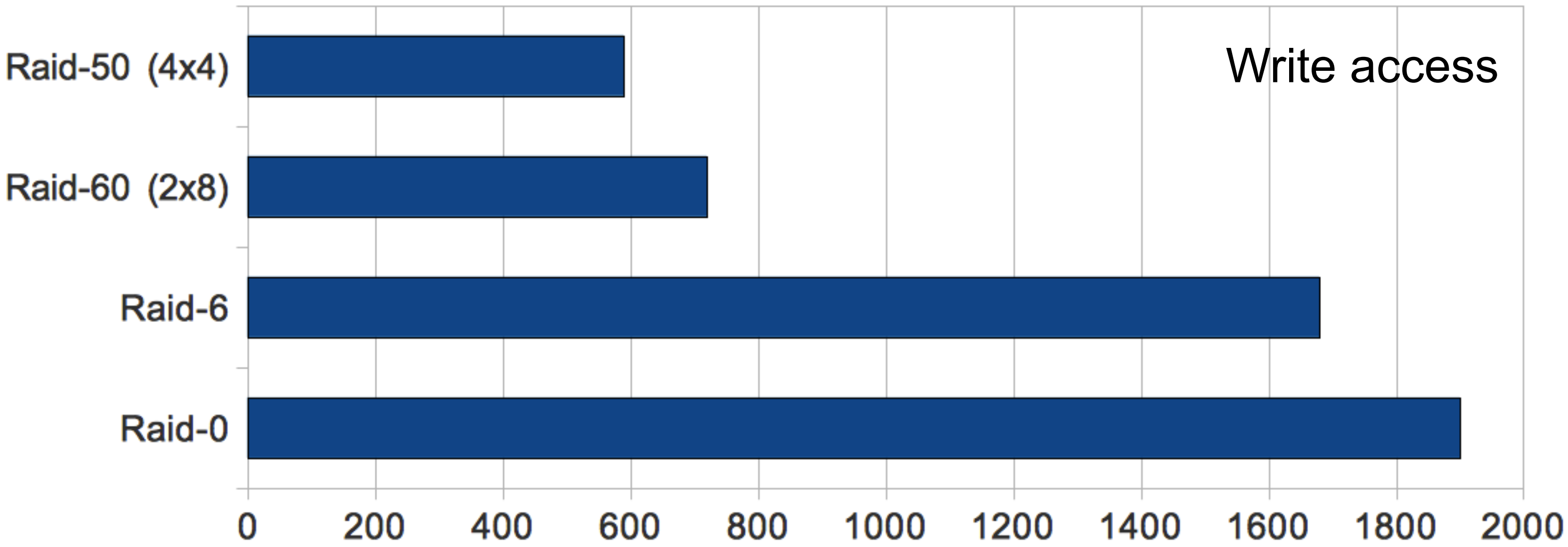
SAS Attached



Hardware Platform: Storage Box

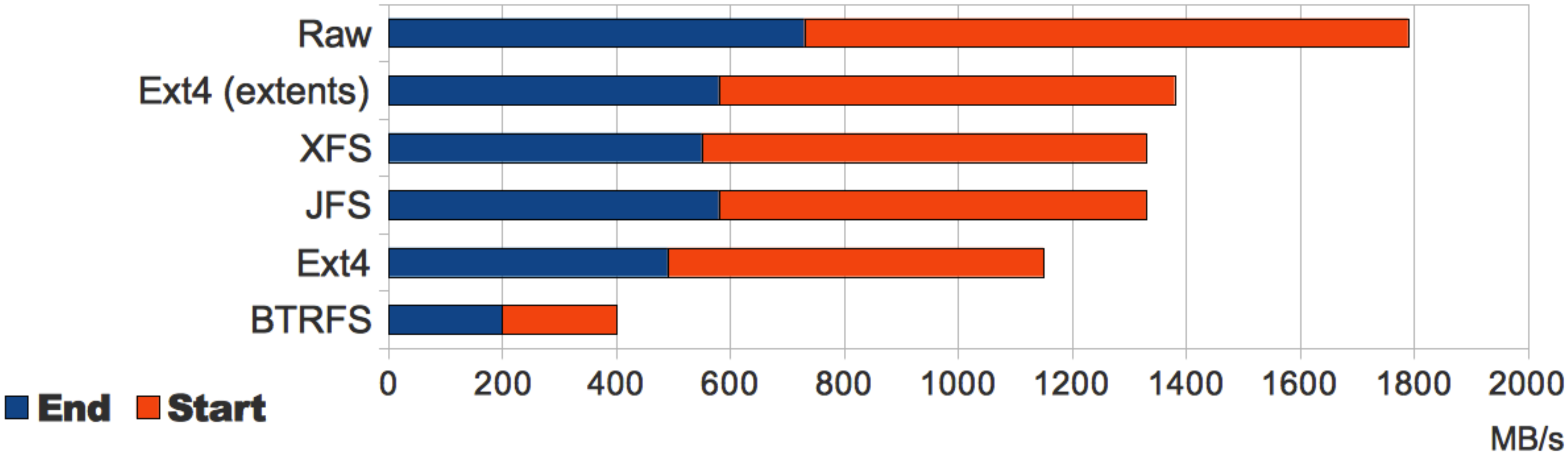






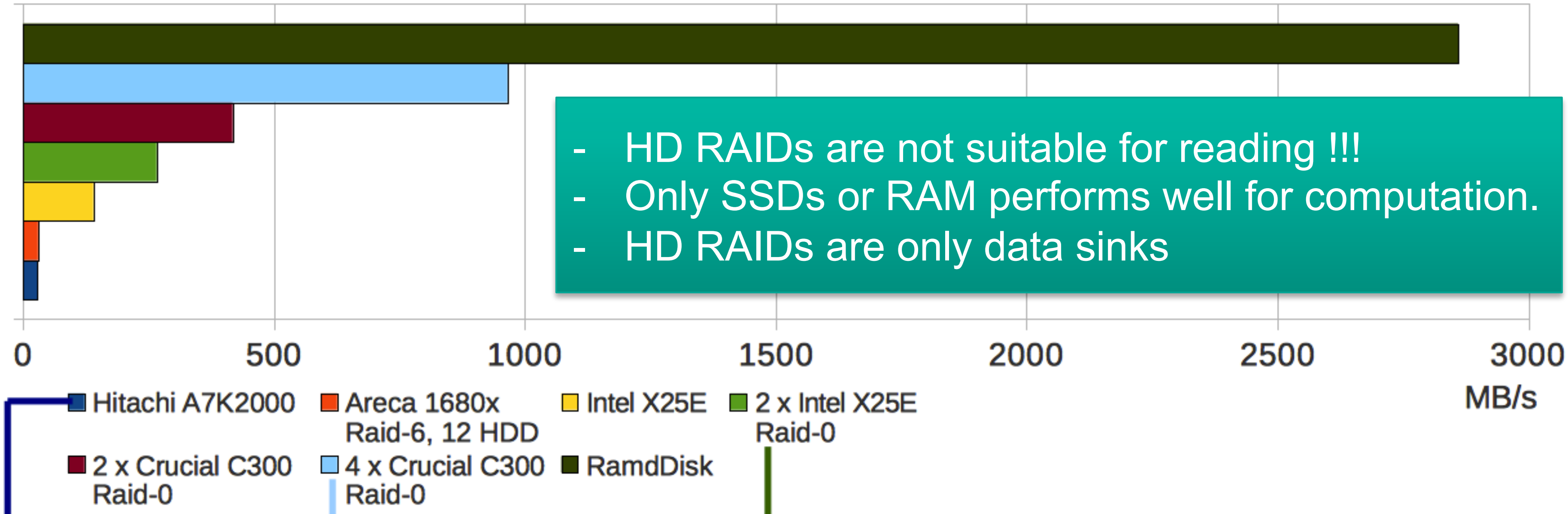
Hardware Platform: File systems (FS)

16 Hitachi A7K200 in Raid0, OpenSuSe 11.3 Kernel 2.6.34

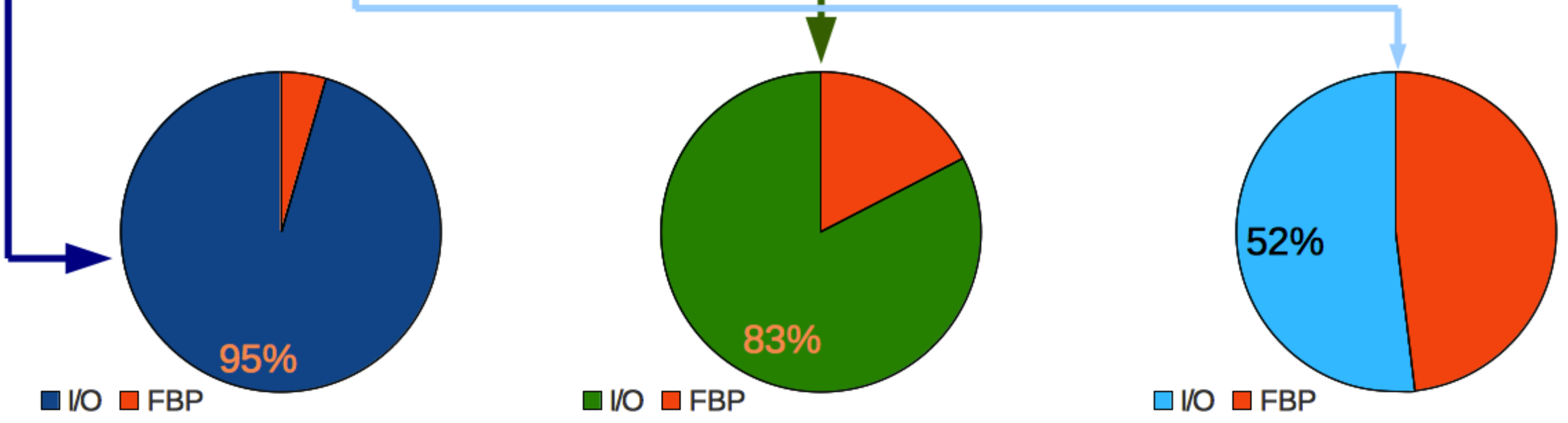


- We lost 30% with fastest FS available and with growth of speed FS penalty grows in percent
- Ext4 performance drops significantly if free space comes to the end. XFS on other hand have spurious reductions of speed on empty disk.
- Fragmentation reduces performance

Hardware Platform: Reading EDF Files



- HD RAIDs are not suitable for reading !!!
- Only SSDs or RAM performs well for computation.
- HD RAIDs are only data sinks

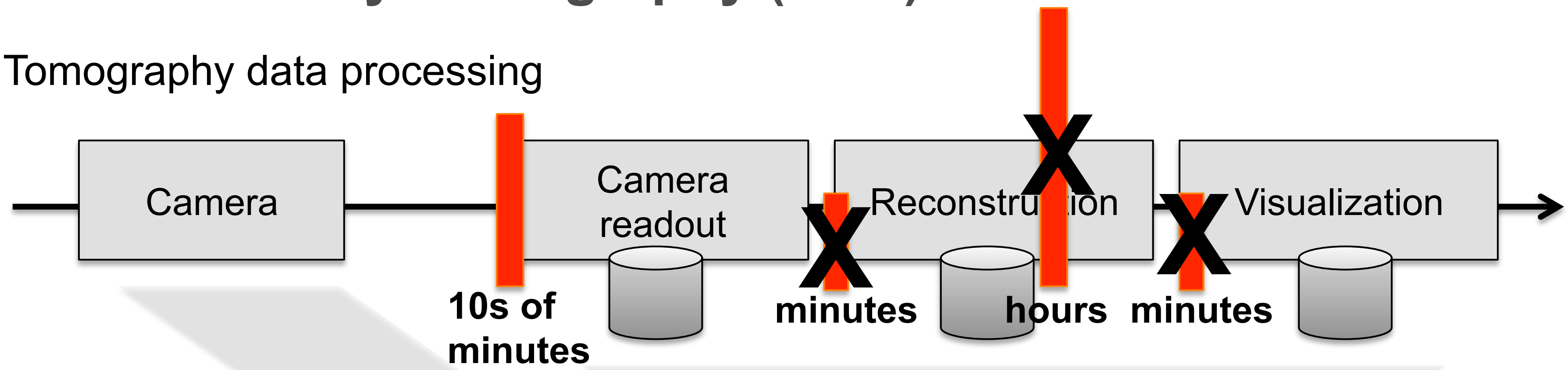


GPU Computing for Tomography?

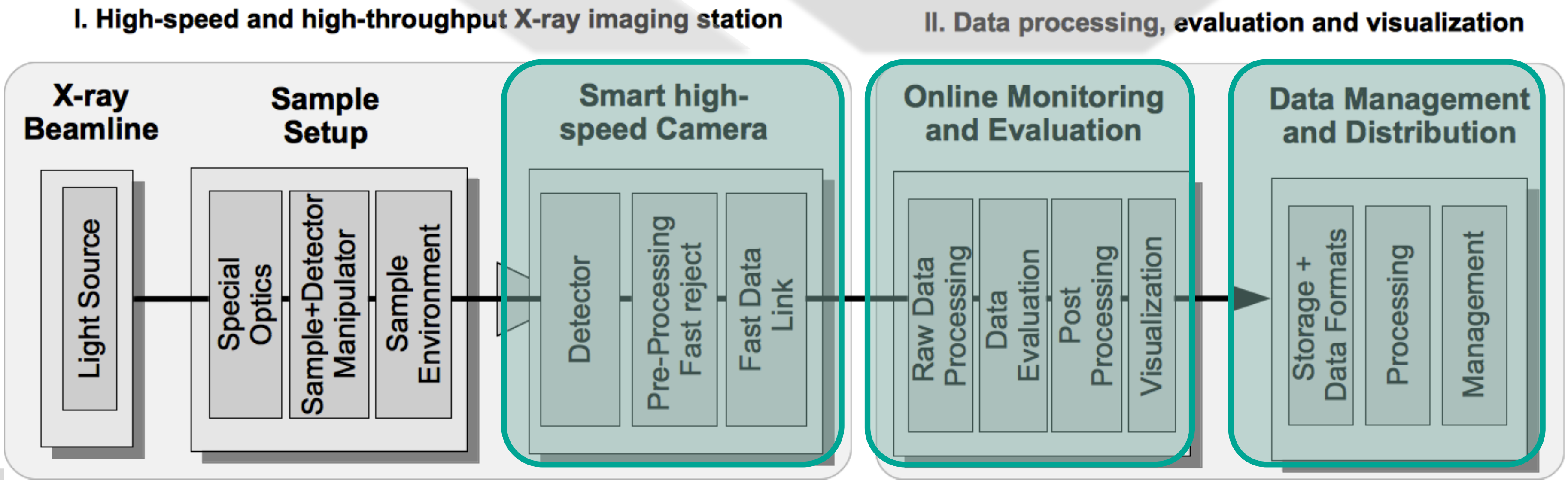
- GPU Architecture serves many needs of scientific community
  - All hardware resources may be used simultaneous
  - Speed-up of 50 – 100 times are possible
- Flexible and easy to use framework is developed
  - GPU+CPU processing with **OpenCL**
  - Efficient through pipelined architecture (hides I/O time)
  - Integration with scripting languages using Gobject-introspection
- Optimal performance requires care
  - Tuning for new architectures may be required
  - Hardware setup may be tuned as well
  - Handling I/O is an important task

Ultrafast X-ray Tomography (UFO)

Tomography data processing



New: streaming setup



Requirements fast X-ray Imaging

Goal: 3D and 4D X-ray imaging with a spatial-temporal resolution up to µm-µs

Sensors requirements:

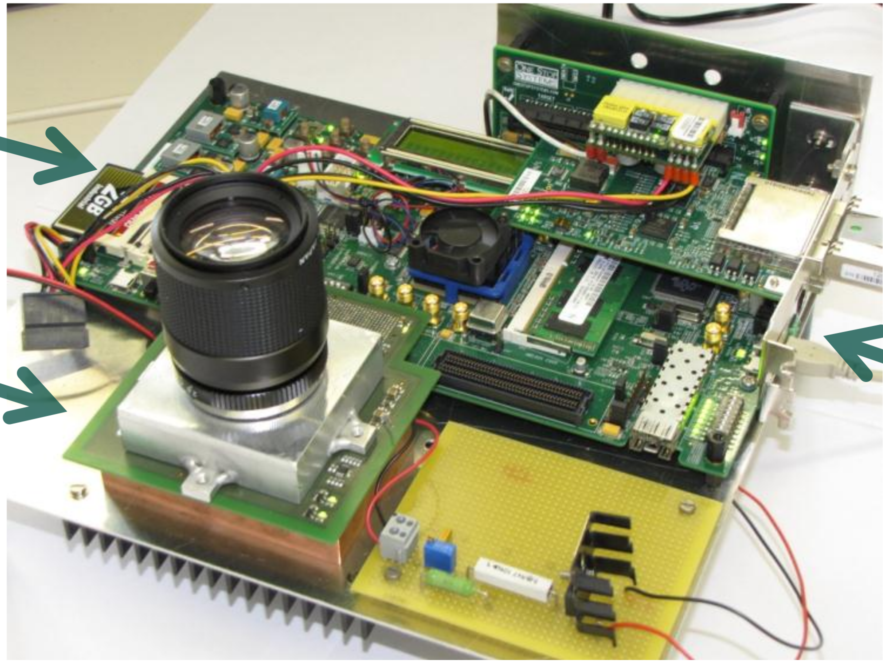
- High granularity monolithic silicon pixel detector, pixel pith few µm, several MPixels matrix.
- High dynamical range ≥ 60dB, ADC with 10 or more bit per pixel
- Low noise, fixed pattern noise (FPN), dark current < few tens of e^-
- High frame rate, range of kilo-frames/s
- Each event (or frame) with 100% of occupancy

Readout requirements:

- *Continuously data streaming* at full CMOS-sensor bandwidth without dead time
- *Intelligent self-event trigger* with a *Region-Of-Interest readout* to increase the original frame rate and same time to reduce the amount of data received

Example: IDT image-sensor → 1MPixels x 10bit/pixel x 5000 fps => 50Gbit/s

High-throughput Camera Platform



Commercial prototype Readout board

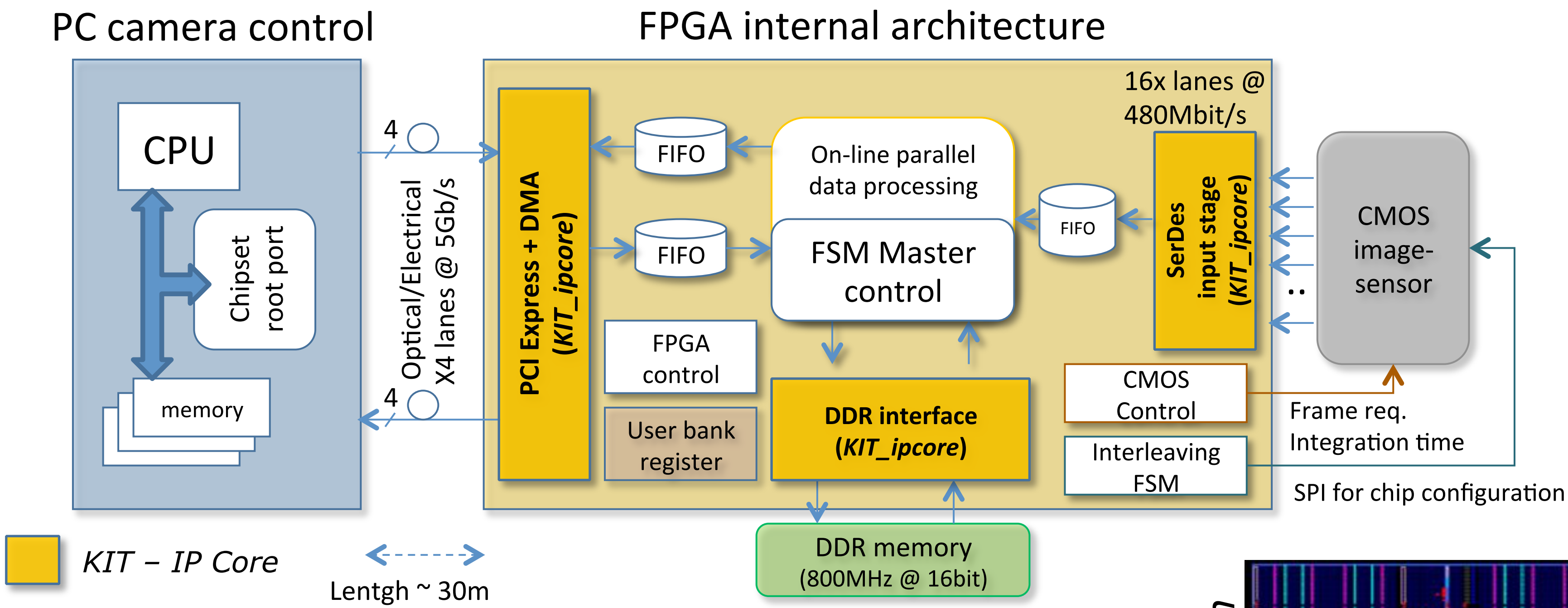
Daughter board with CMOS sensor

External PCI Express Link (to compute server)

Main features:

- Continuous data acquisition at full speed (ca. 2 GB/sec)
- Easily extendable to any available CMOS image sensor
- Fully programmable
  - Adjustable image exposure time and dynamic range, analog and digital pixel features as pixel threshold, mask, analog gain, on-line image processing etc.

Readout chain with FPGA IP-Cores

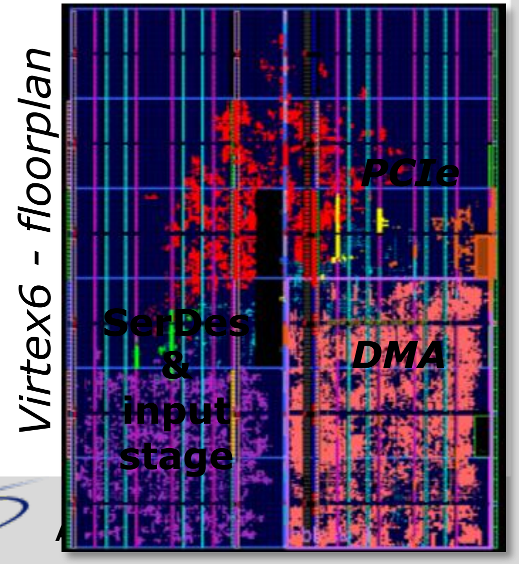


✓ **Firmware based on a Bus Master DMA (BMD) architecture up to 16Gb/s**

✓ **Three Intellectual Property (IP) logic blocks have been developed**:

- PCI Express 4x GEN2 + DMA
- DDR3 Interface
- Fast SerDes input stage

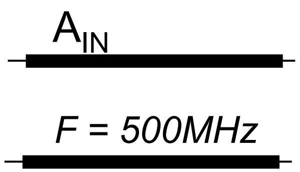
✓ **Linux-32/64 bit driver + GUI application**

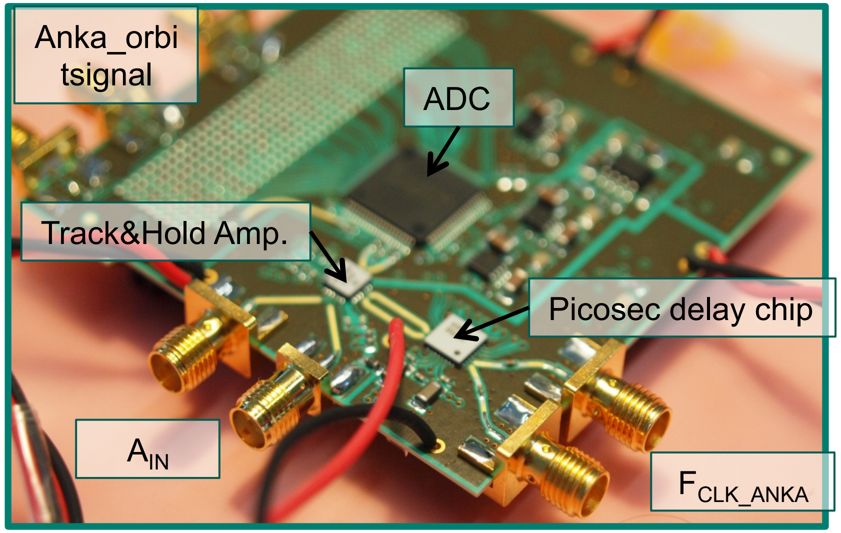


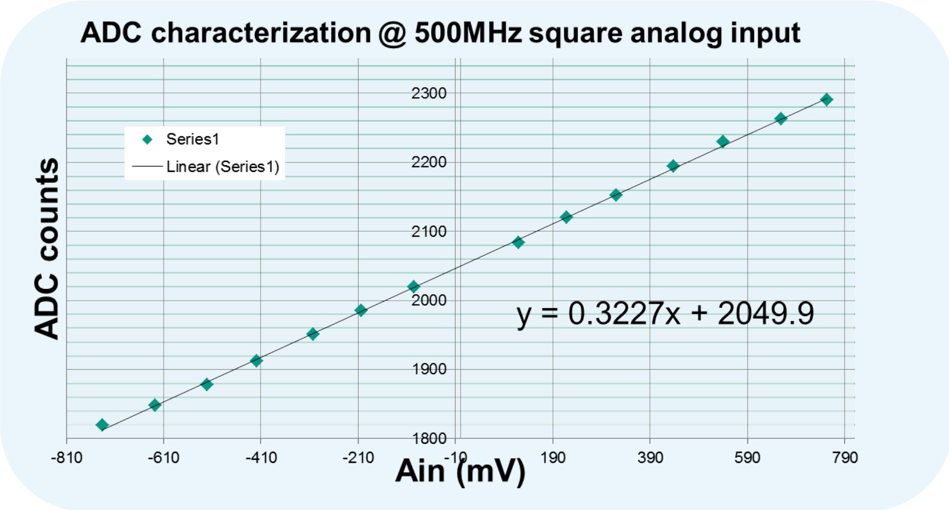
Hot Electron Bolometer (HEB) – Pilot board

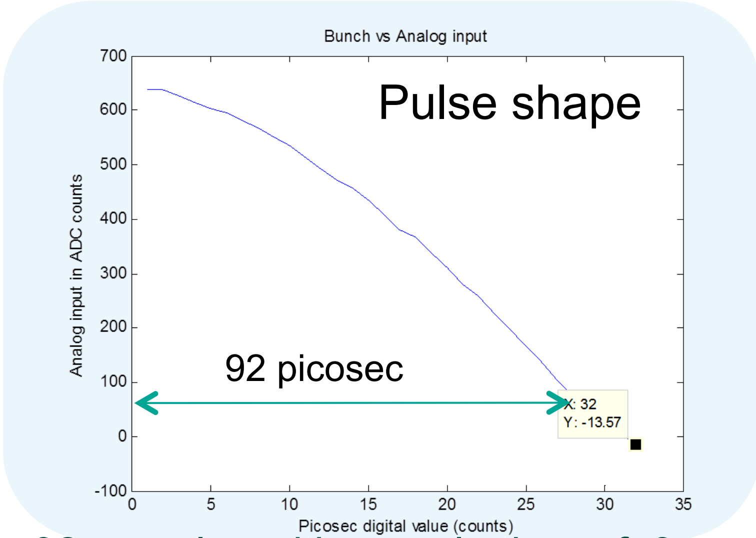
Concept:
Signal splitting + precise ps delay

Characterization with A_{in} @ 500MHz
with square and pulse shapes





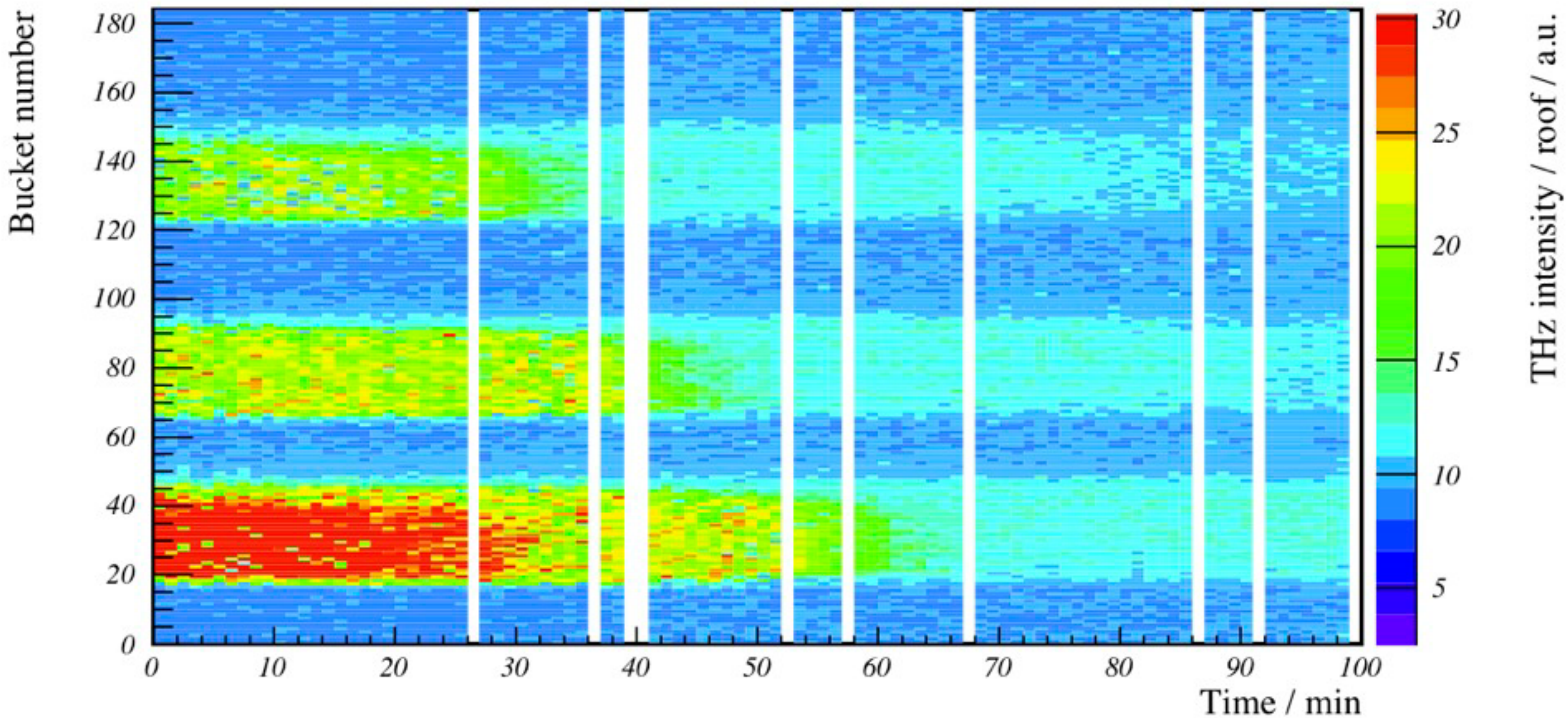




32 samples with sample time of 3psec

HEB: ANKA long-time Bunch Behavior

YBCO – terahertz detector



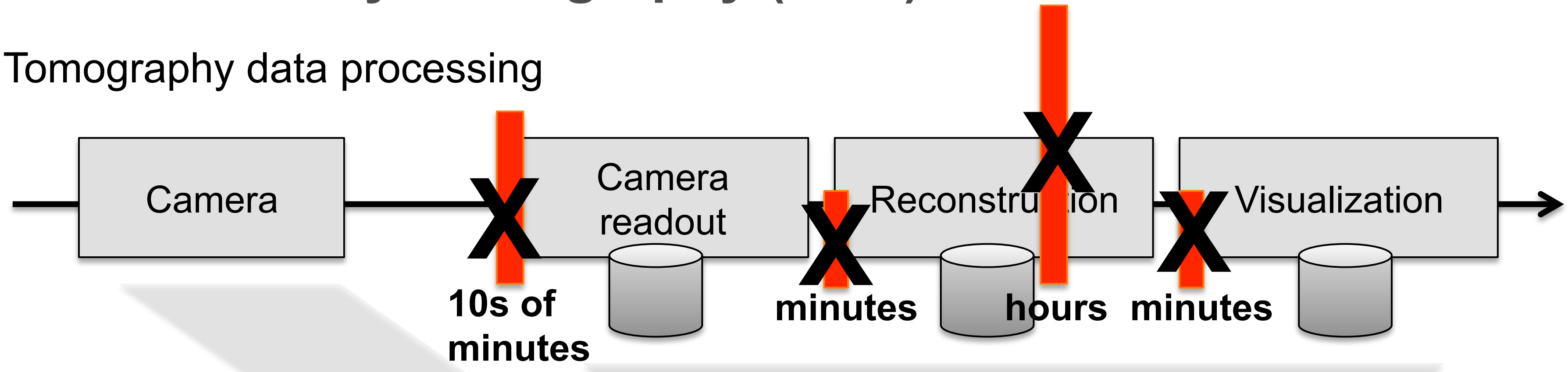
Next step:

- Development of 4 channel system

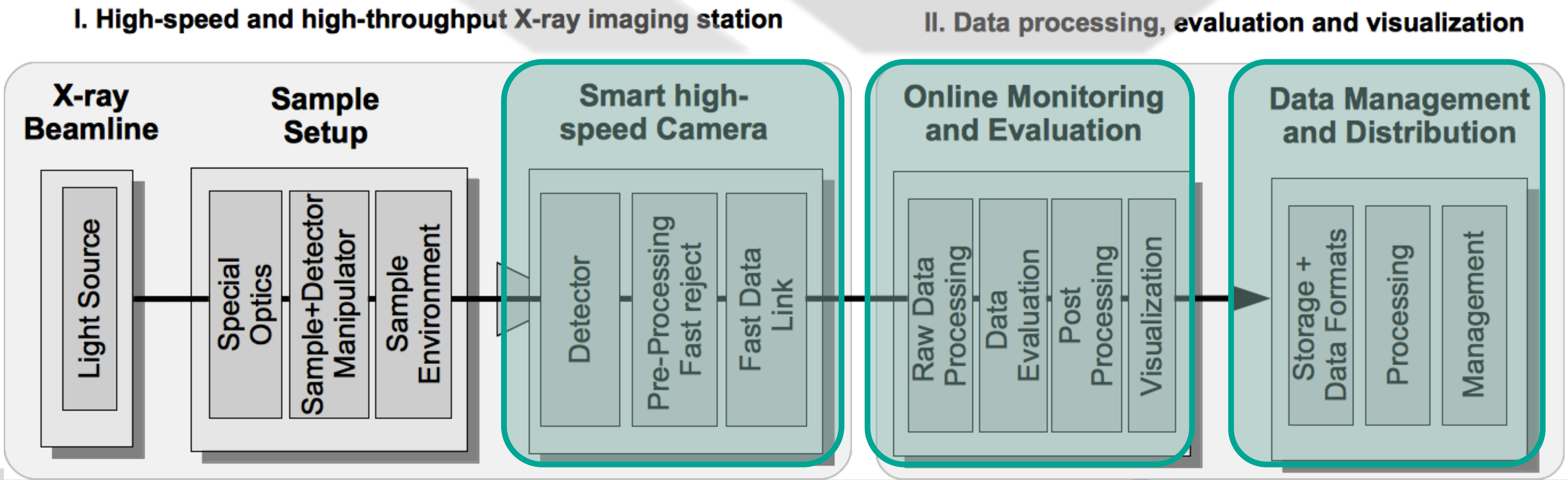
- Very fast and very synergetic
- Based on UFO-Tool chain

Ultrafast X-ray Tomography (UFO)

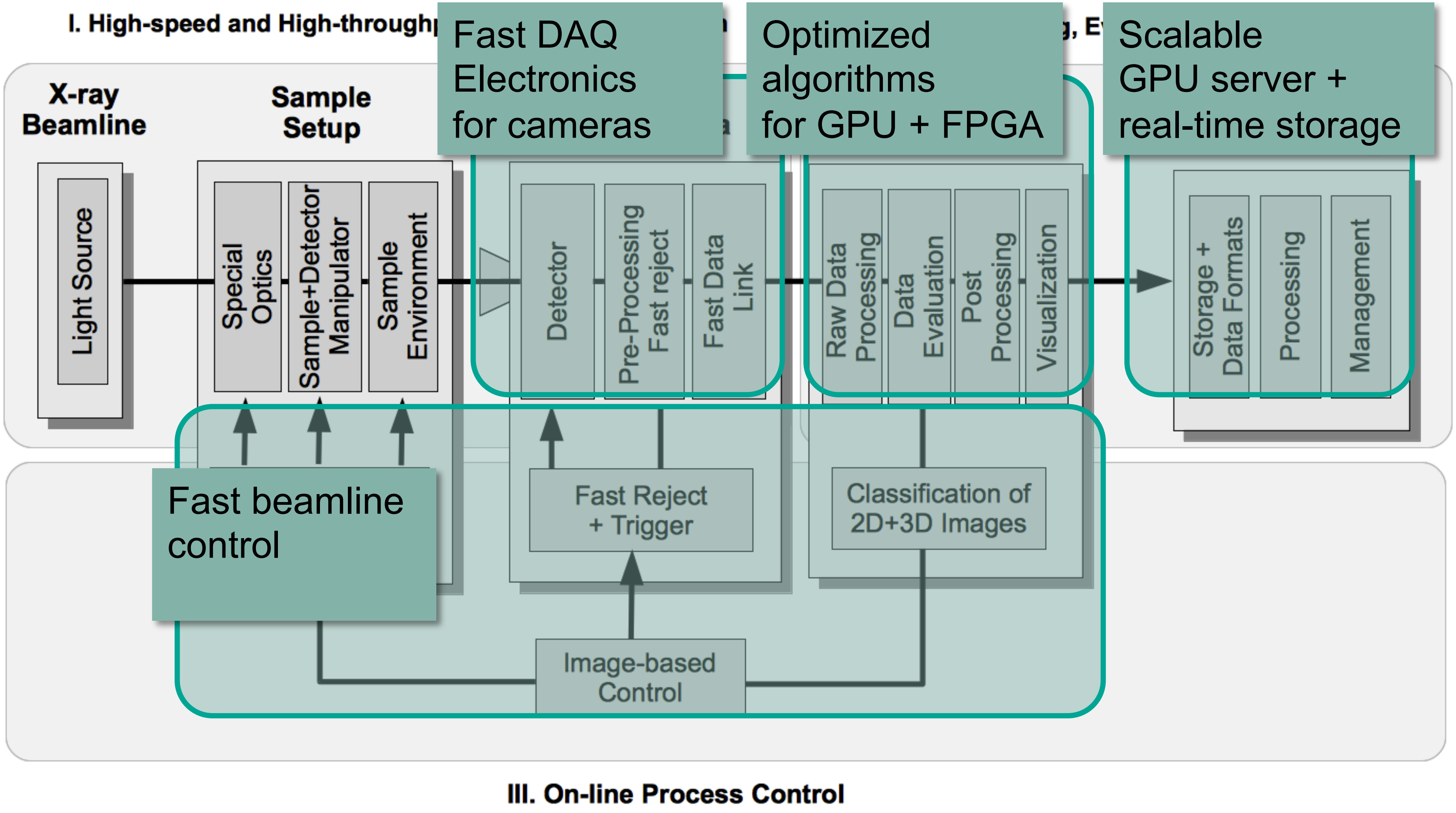
Tomography data processing



New: streaming setup



Ultrafast X-ray Tomography (UFO)



Conclusion

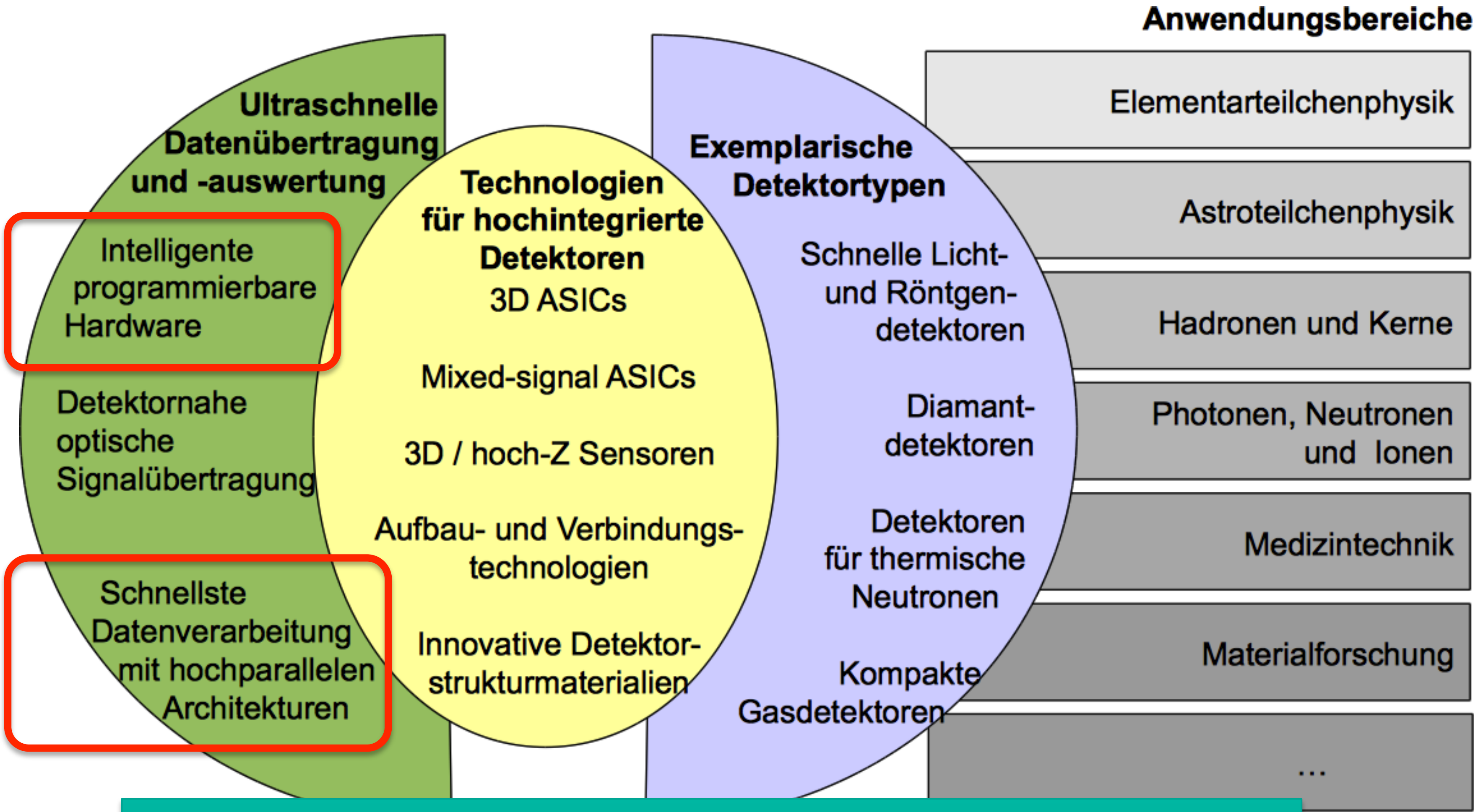
- Heterogeneous **FPGA+CPU+GPU high bandwidth infrastructure** for scientific application
  - General readout architecture with **2 GByte/s bandwidth**
  - Modular detector interface (by FMC connector)
  - PCI Express interface, 32/64bit Linux drivers

- Camera prototype with 2.3MPixel @ 330 fps
  - Fully programmable
  - Integrated in UFO-Parallel Computing Framework
  - Fast reject and trigger functions

Applications:
- Hot electron bolometer
- Smart phase contrast camera (Collab. w. HZG)

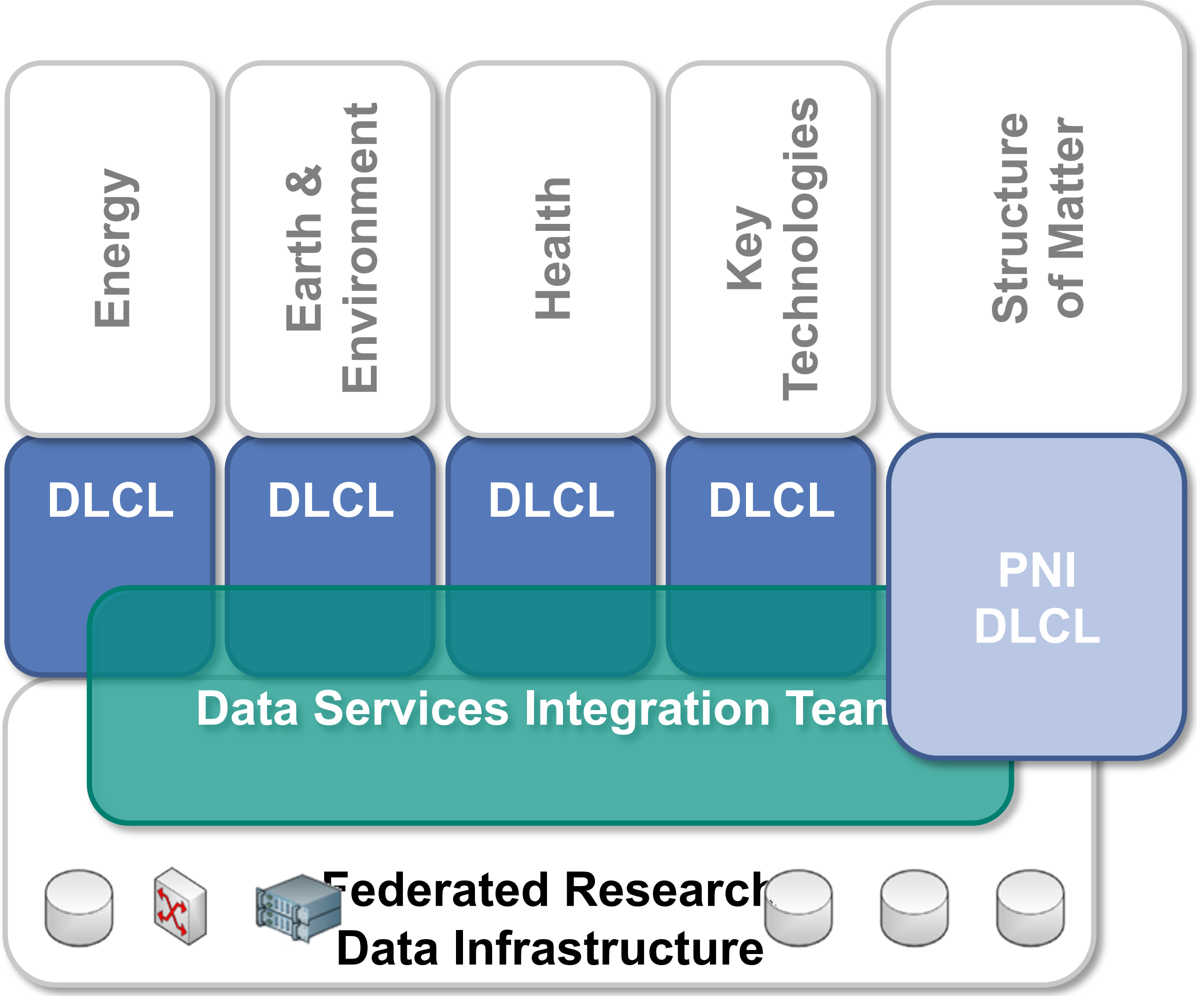
- Next: High-speed camera 1MPixel @ 5000 fps
  - Required bandwidth: 6 GBytes/s
  - Alternative computer interfaces (e.g. Infiniband)

Portfolio:
“Detector Technology and Systems Platform”



- The high-throughput DAQ-Platform is a key component to realize applications with new detectors

Portfolio: "Large Scale Data Management"



Tasks:
- Define and optimize data formats
- Simple use of infrastructure
- Standard data management techniques
- Preserve long-term data access
- …

"The future of HDRI"

UFO Team

Karlsruhe Institute of Technology

- Matthias Balzer
- Tilo Baumbach
- Felix Beckmann
- Jörg Burmester
- Suren A. Chilingaryan
- Michele Caselle
- Tomas Farago
- Thomas van de Kamp
- Andreas Kopmann
- Alessandro Mirone
- Anton Myagotin
- Tomy dos Santos Rolo
- Uros Stevanovic
- Matthias Vogelgesang
- Marc Weber

Shubnikov Crystallography Institute, Moscow

Saint Petersburg State University of Civil Aviation

Tomsk Polytechnic University

European Synchrotron Radiation Facility

Helmholtz-Zentrum Geesthacht