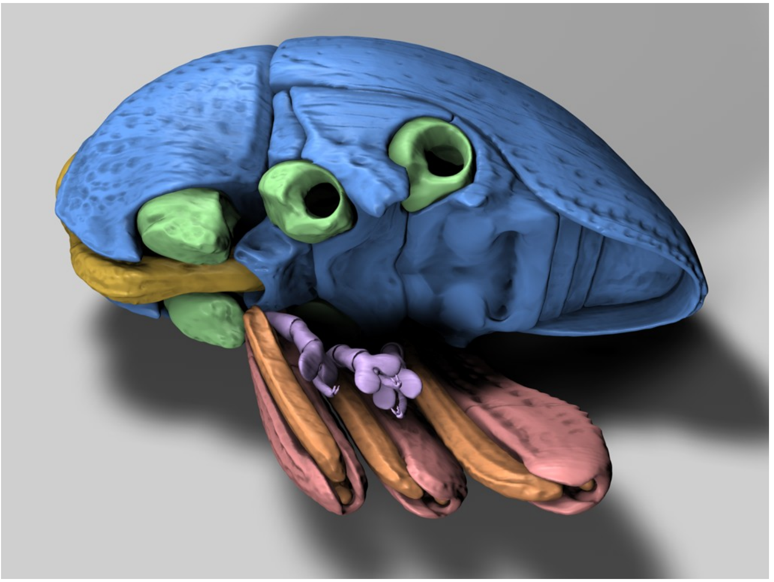
A High Performance Platform for Real-Time X-ray Imaging

Agenda

Synchrotron Tomography at KIT
Hardware & Software Platform
Optimizing Tomography Speed



Example for 3D X-Ray imaging. The functional groups of a flightless weevil are colored

Architectures

NVIDIA GT200
NVIDIA Fermi
NVIDIA Kepler

AMD VLIW5
AMD GCN

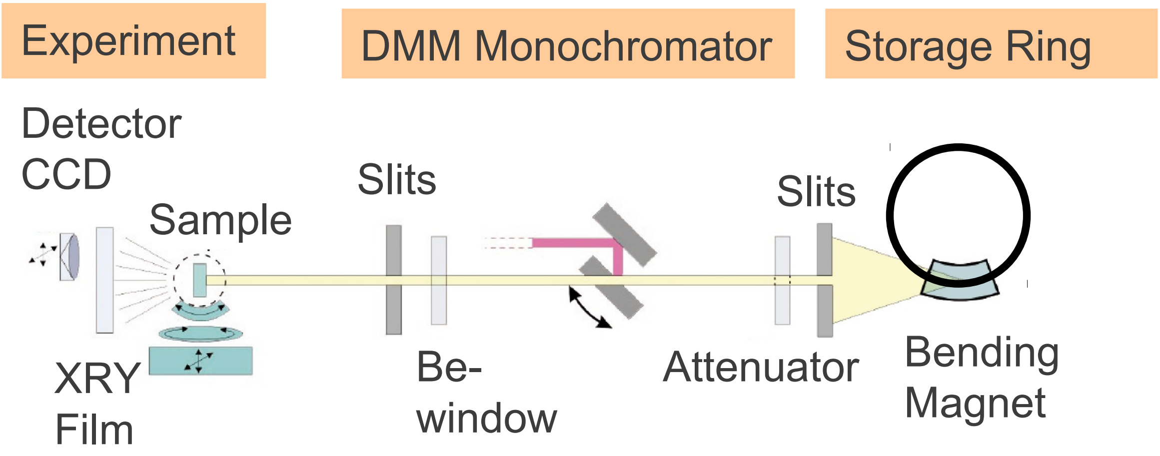
Authors

Suren A. Chilingaryan, KIT
Michele Caselle, KIT
Thomas van de Kamp, KIT
Andreas Kopmann, KIT
Alessandro Mirone, ESRF
Uros Stevanovic, KIT
Tomy dos Santos Rolo, KIT
Matthias Vogelgesang, KIT

In collaboration with ESRF:
European Synchrotron Radiation Facility

ANKA Synchrotron at KIT

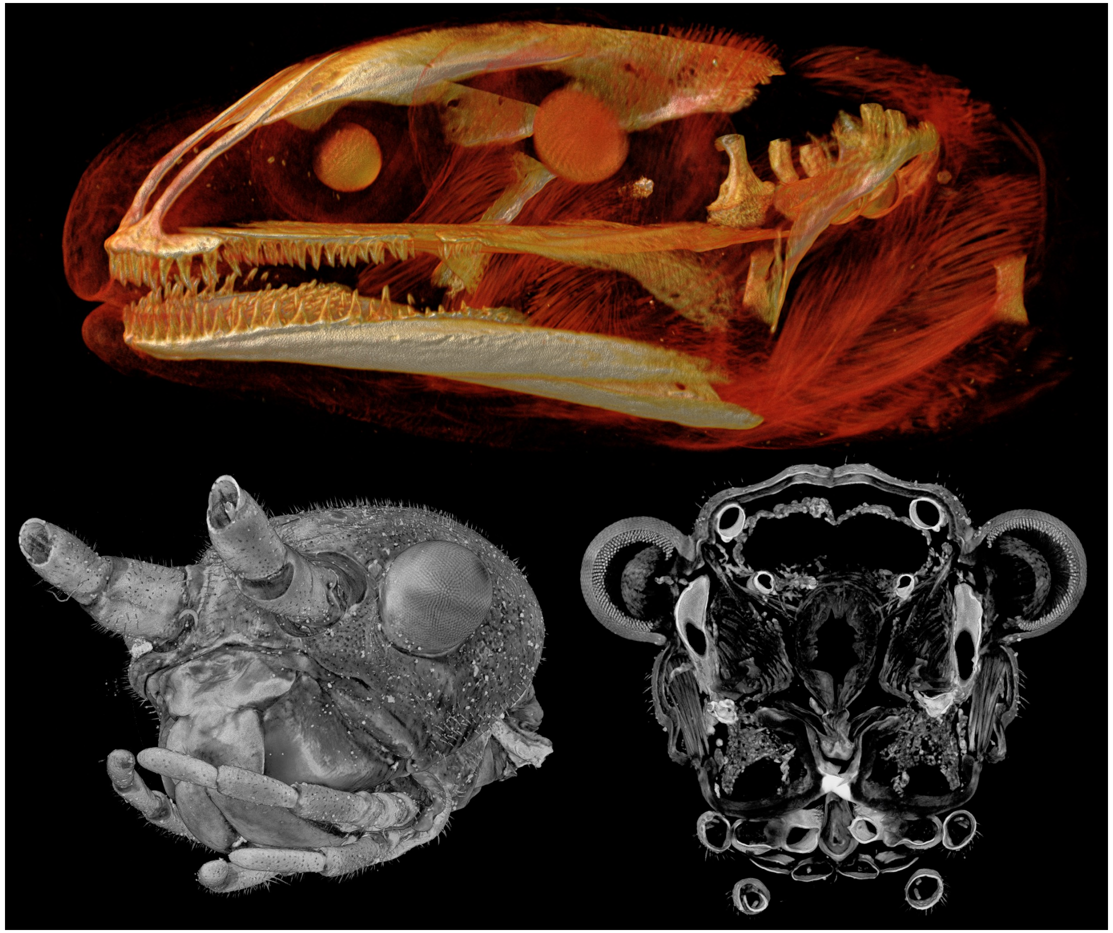
Karlsruhe Institute of Technology



ANKA synchrotron (left) and schematics of TOPO-TOMO beamline (right).

The rotating sample in front of a pixel detector is penetrated by X-rays produced in the synchrotron. Absorption at different angles is registered by camera and 3D map of sample denisity is reconstructed.

Examples



Heads of a newt larva showing bone formation and muscle insertions (top) and a stick insect (bottom), acquisition time 2s.

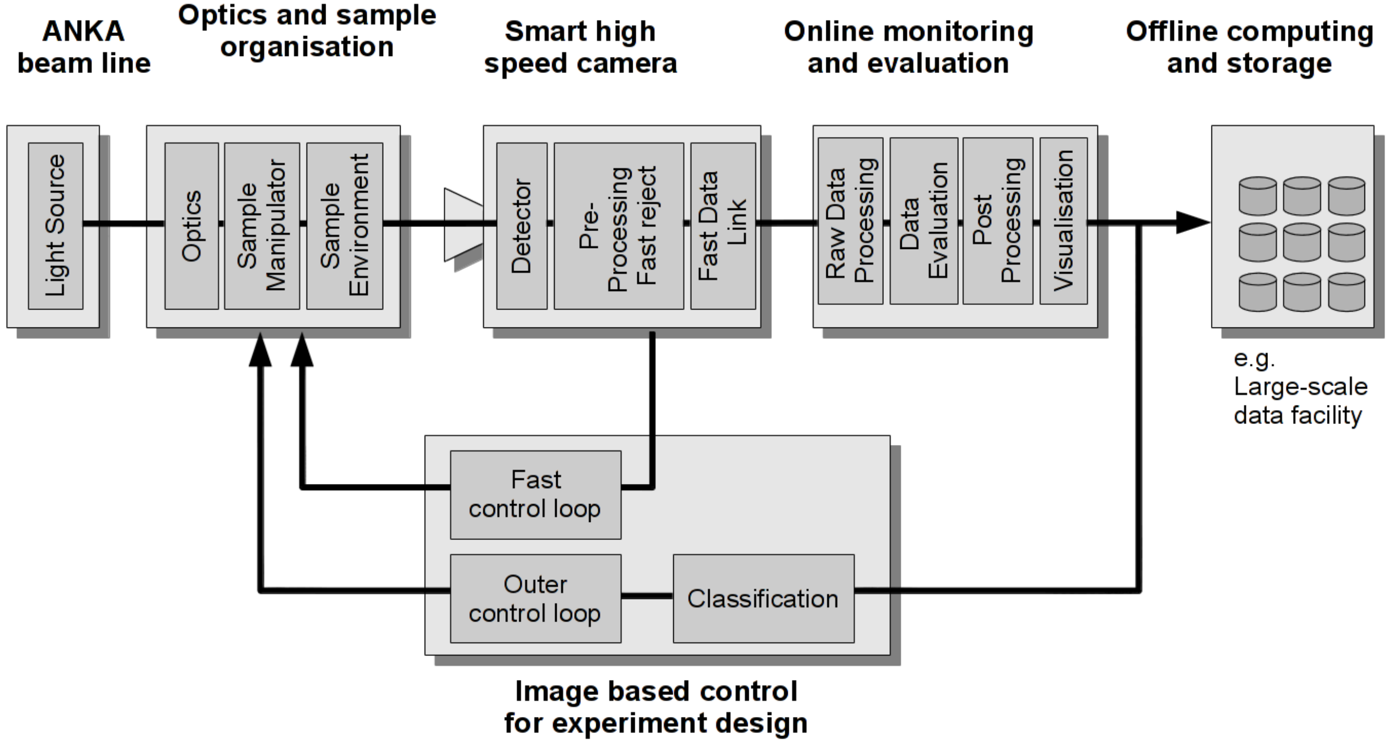
4D tomogram of wheat weevil

UFO Project

Ultra Fast X-ray Imaging of Scientific Processes with On-Line Assessment and Data-Driven Process Control

Goals

- Increase sample throughput
- High speed tomography
- Tomography of temporal processes
- Allow interactive quality assessment
- Enable data driven control
  - Auto-tunning optical system
  - Tracking dynamic processes
  - Finding area of interest



S. Chilingaryan et. all
Institute for Data Processing and Electronics
Karlsruhe Institute of Technology

Online Monitoring Configuration

Camera

PCO.edge
PCO.dimax
PCO.4000

CameraLink
850MB/s

Ethernet (10 Gb/s)

Storage

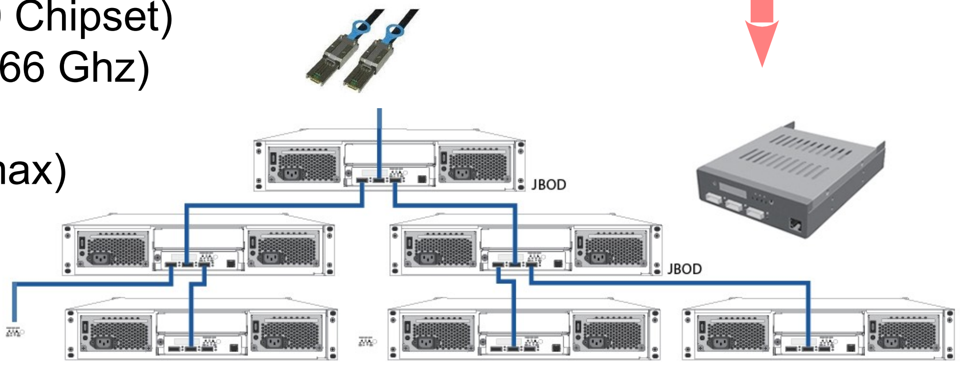
LSDF
Large Scale Data Facility

External PCIe x16 (8 GB/s)

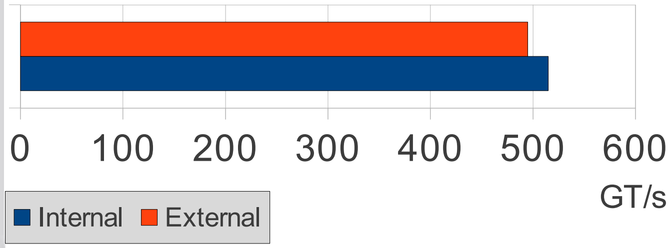
SFF8088 (2.4 GB/s)

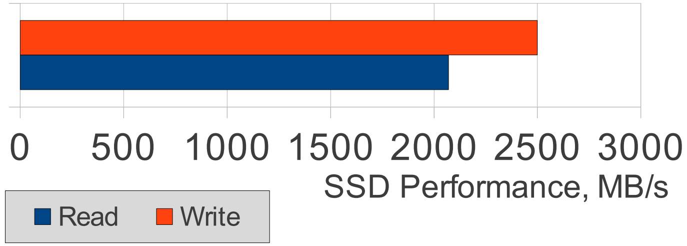
External GPU Box

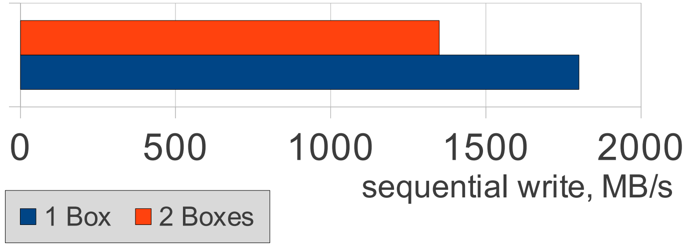
SuperMicro 7046GT-TRF (Dual Intel 5520 Chipset)
CPU: 2 x Xeon X5650 (total 12 cores at 2.66 Ghz)
GPUs: 4 x GTX590 External
Memory: 96 GB / 12 DDR3 slots (192GB max)
Network: Intel 82598EB (10 Gb/s)
Camera Link Frame Grabber (850 MB/s)
Storage: Areca ARC-1880-ix-12 SAS Raid
16 x Hitachi A7K200 (Raid6)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



SAS Attached Storage





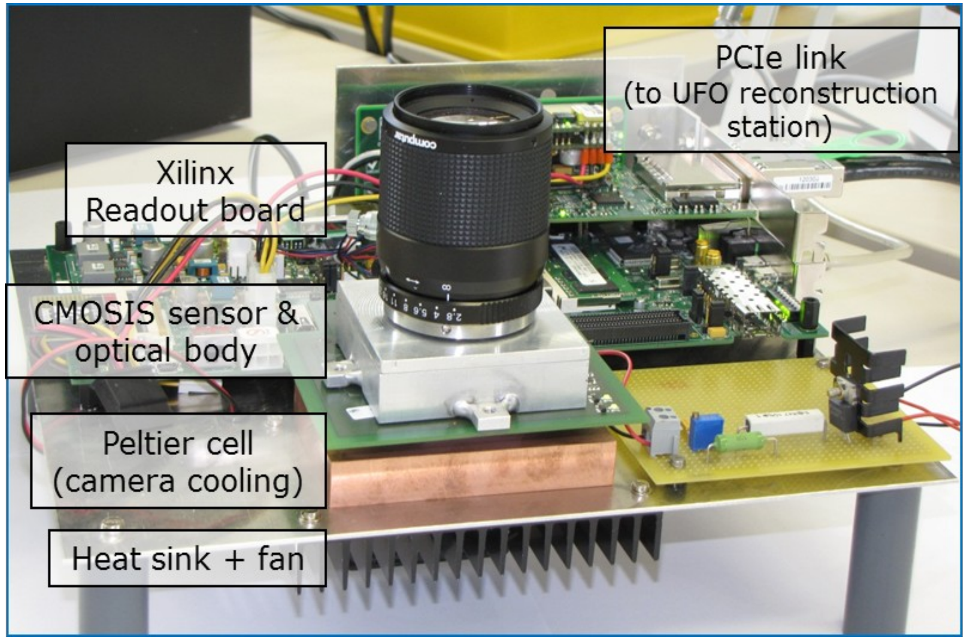


S. Chilingaryan et. all

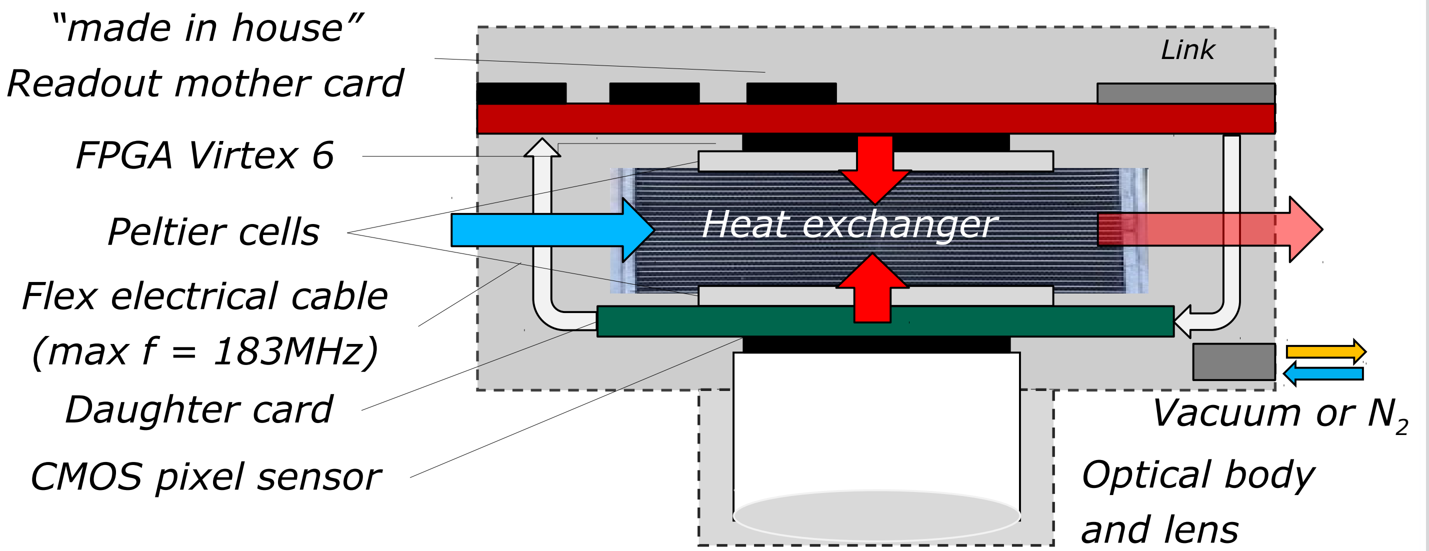
Institute for Data Processing and Electronics
Karlsruhe Institute of Technology

Next Generation: High-speed Programmable Camera

Karlsruhe Institute of Technology



First Camera Prototype

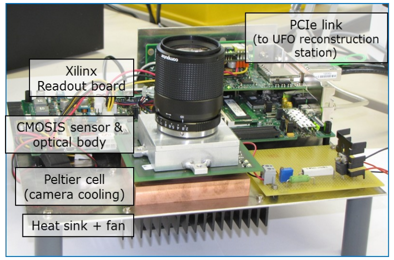


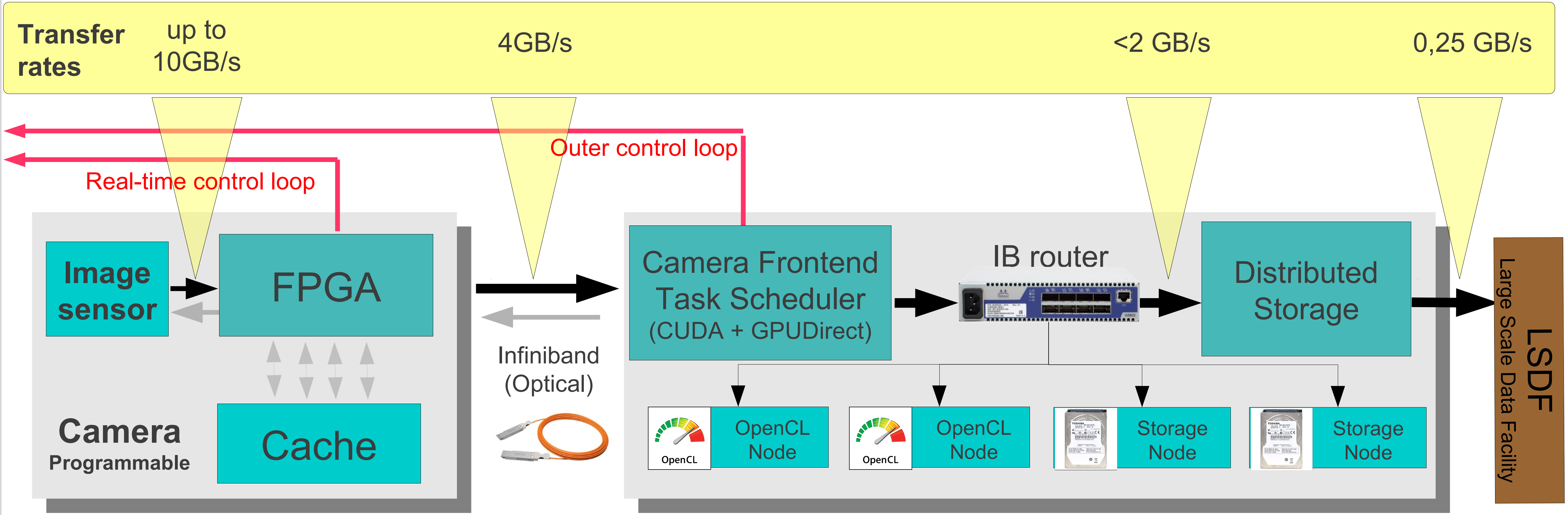
- ***High speed CMOS sensor***
- ***1Mpix, 5000 fps, 10 bits***
- ***Self-trigger & Data compression***
- ***On-line elaborations and control***
- ***Full Programmability***
- ***Direct connection to Infiniband-cluster***

S. Chilingaryan et. all

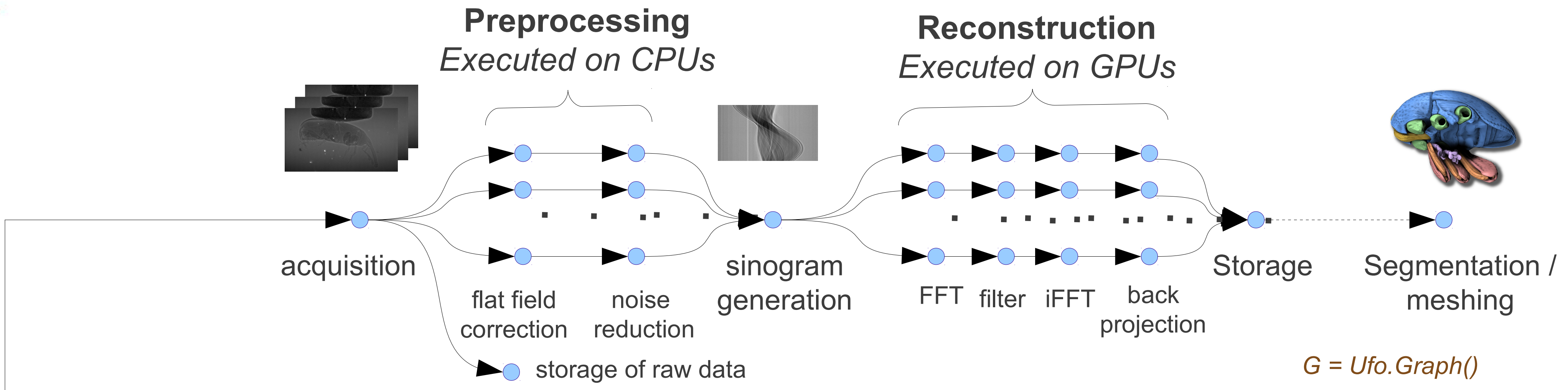
Institute for Data Processing and Electronics
Karlsruhe Institute of Technology

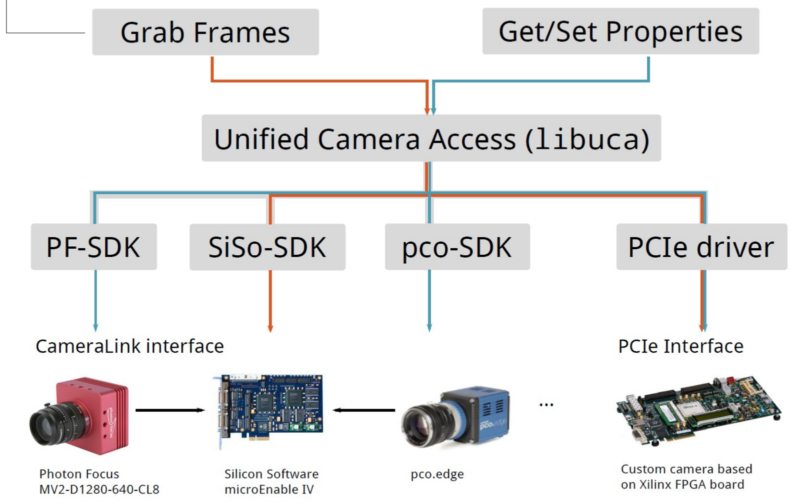
Next Generation: Processing Cluster





UFO Framework





OpenSource

http://ufo.kit.edu/framework

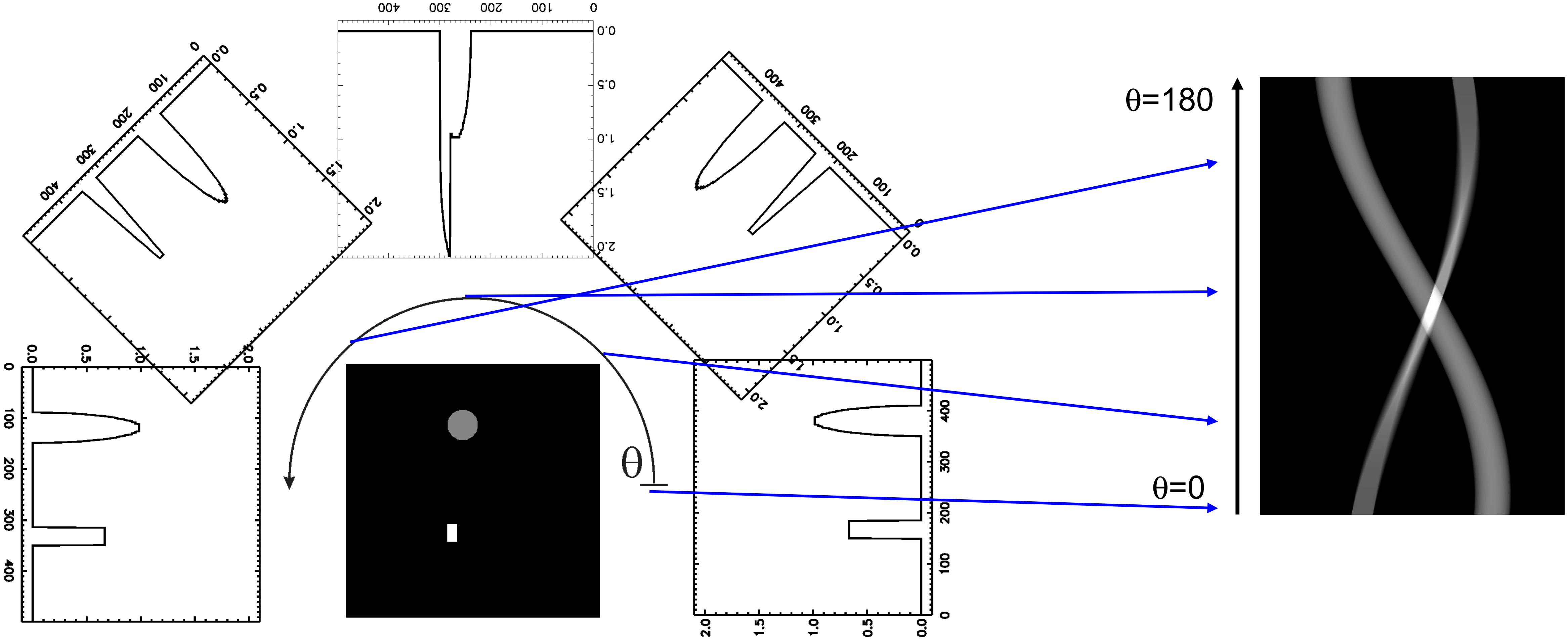
Features

- Easy Algorithm Exchange
- Camera Abstraction
- Pipelined Processing
- Glib/GObject, scripting language support with introspection
- OpenCL + automated management of OpenCL buffers

```
G = Ufo.Graph()

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

angle = numpy.pi / num_projections
fbp.set_properties(axis=axis, angle_step=angle)
raw_wr.set_properties(path=raw_path, prefix='raw')
wr.set_properties(path=output_path, prefix='slice')

rd.connect_to(cp)
cp.connect_by_name("output1", fbp, "input")
cp.connect_by_name("output2", raw_wr, "input")
fbp.connect_to(wr)

g.run()
```


S. Chilingaryan et. all

Synchrotron Tomography



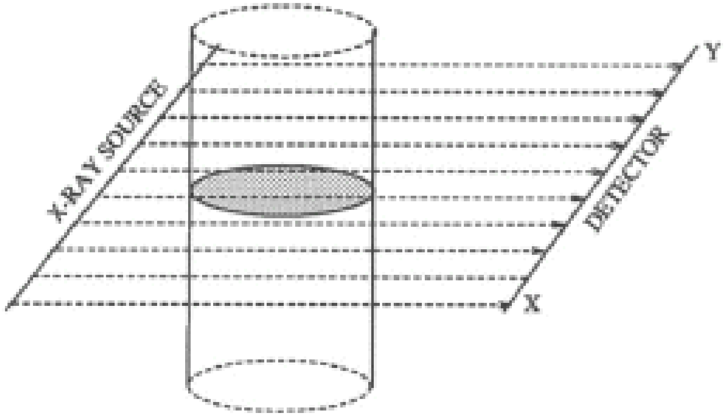
The sample is evenly rotated and the pixel detector registers series of parallel 2D projections of the sample density at different angles.

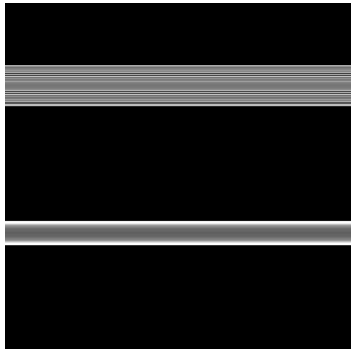
S. Chilingaryan et. all

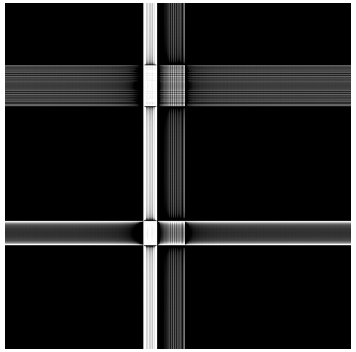
Institute for Data Processing and Electronics
Karlsruhe Institute of Technology

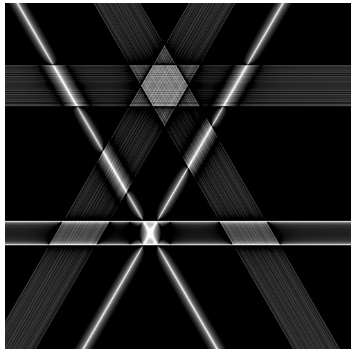
Tomographic Reconstruction

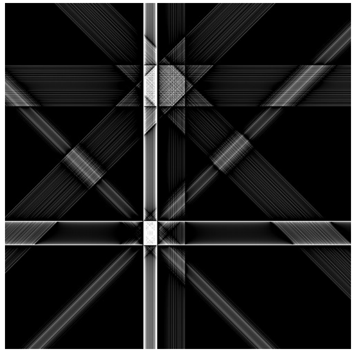
Filtered back-projection is used to produce 3D images from a manifold of two dimensional projections**.** Vertical slices are processed independently. For each slice all projections are smeared back onto the cross section along the direction of incidence yielding an integrated image.

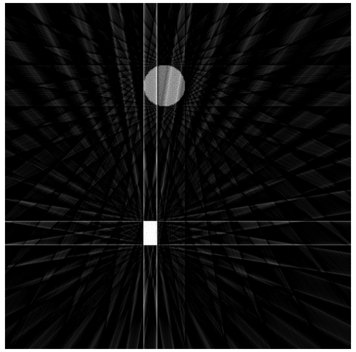


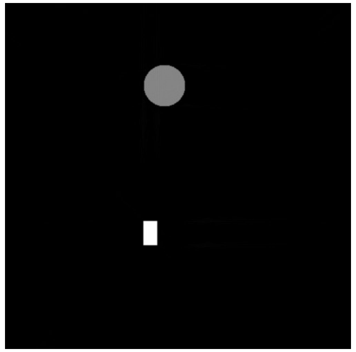

n = 1


n = 2


n = 3


n = 10


n = 50

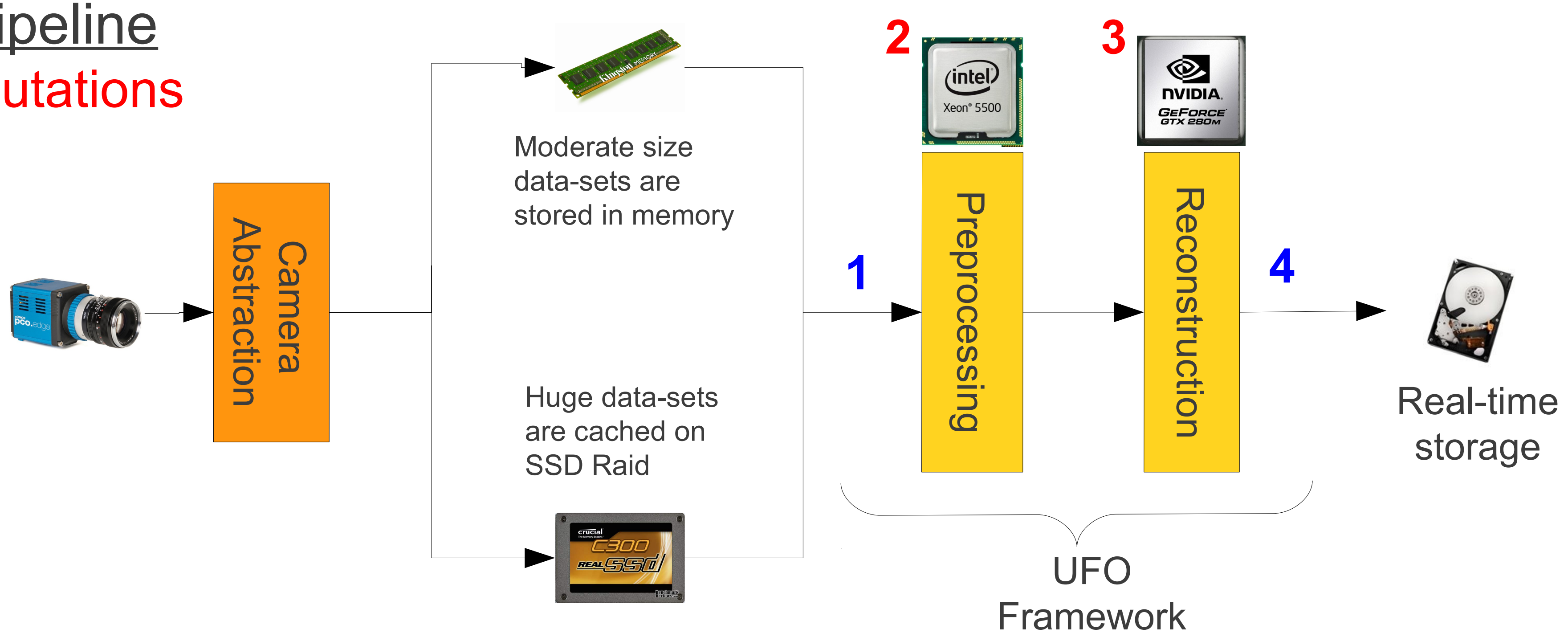

n = 1000

S. Chilingaryan et. all

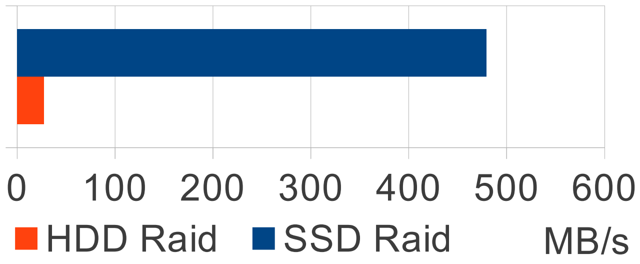
Processing Chain: pipelined Architectures

<u>4 stage pipeline</u>

I/O + Computations



1. Reading data from fast SSD Raid-0 (random reads are effective)
2. Scheduling and preprocessing using SIMD instructions of x86 CPUs
3. Reconstructing on GPUs
4. Storing to Raid on magnetic disks (sequential writes are effective)



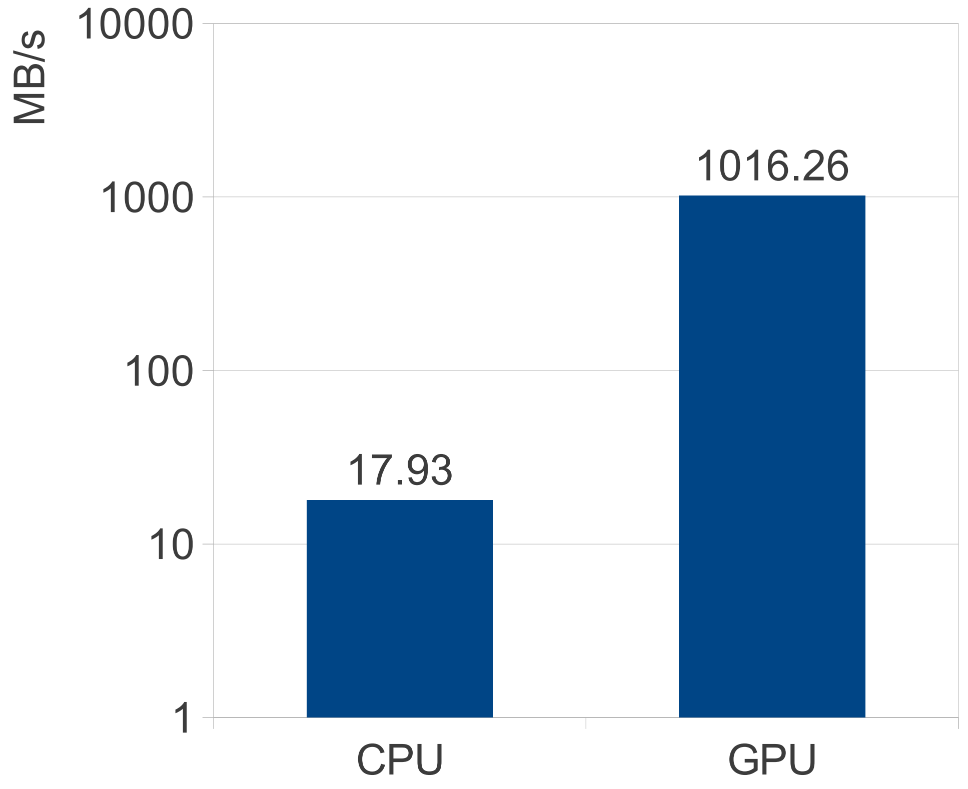
S. Chilingaryan et. all

Institute for Data Processing and Electronics
Karlsruhe Institute of Technology

Performance of Tomographic Reconstruction

Scaling with dataset size

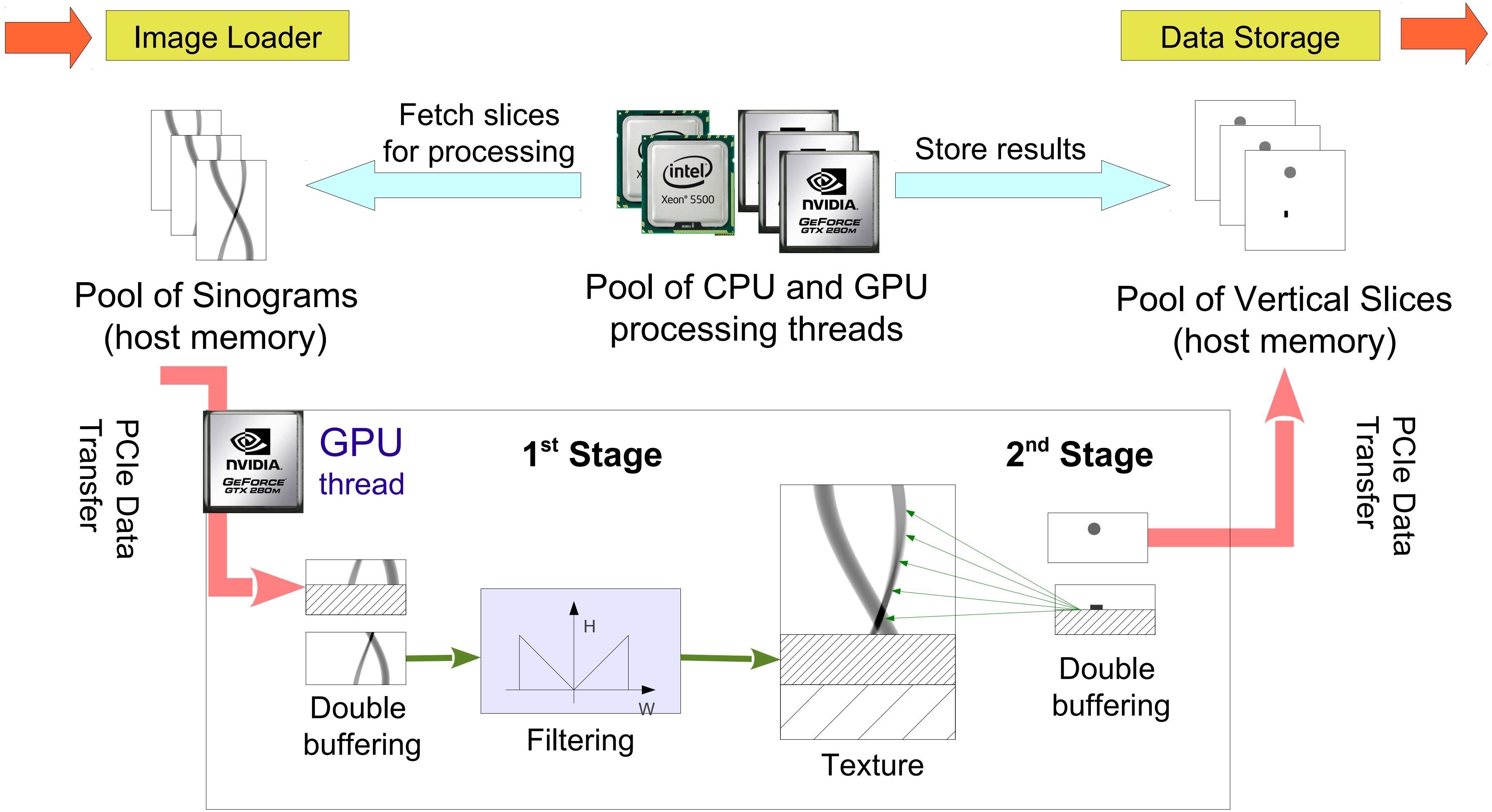
CPU vs. GPU



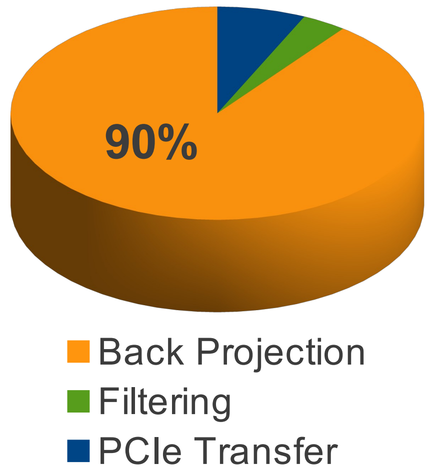
11 GB standard data set

S. Chilingaryan et. all

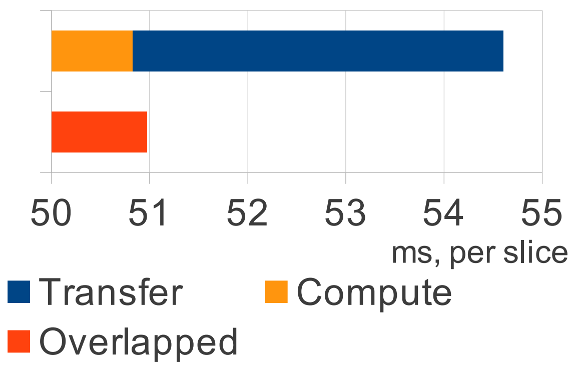
Basic Implementation



Ratio of operations



Overlapping Efficiency



S. Chilingaryan et. all

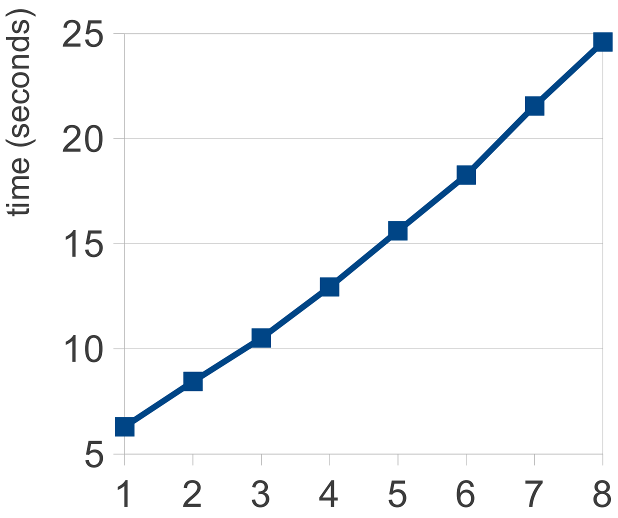
Institute for Data Processing and Electronics
Karlsruhe Institute of Technology

Extension Box

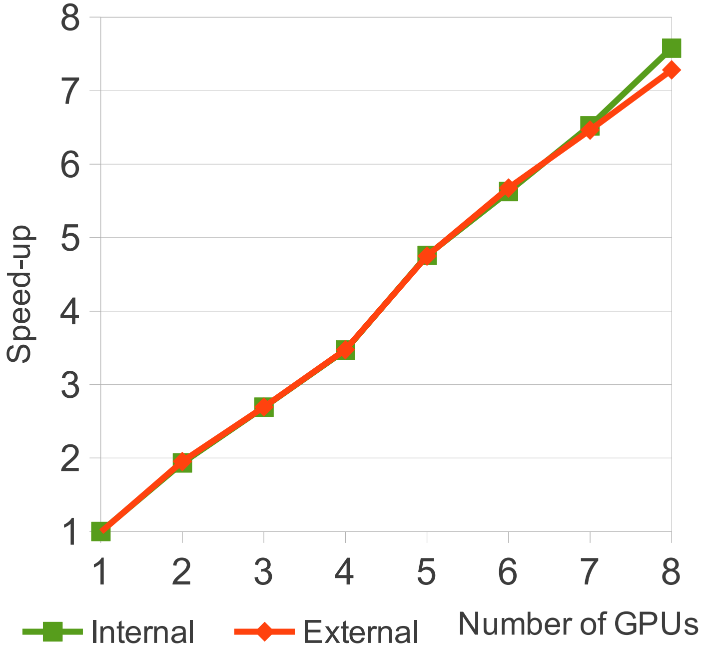
1 x PCIe x16 2.0
4 x GTX590
8 GPU cores

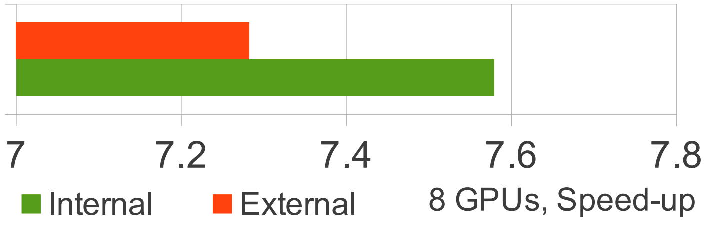
External GPU Enclosure
by One Stop Systems

Initialization

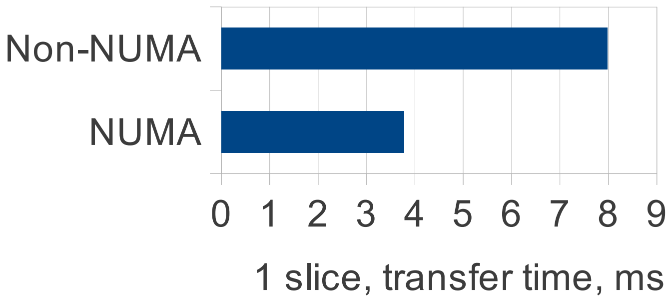


Scalability

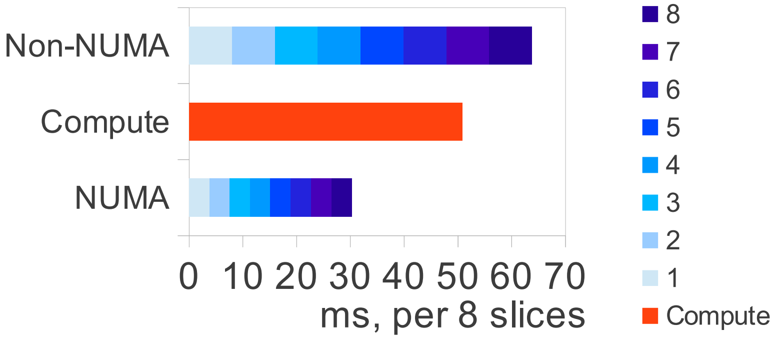


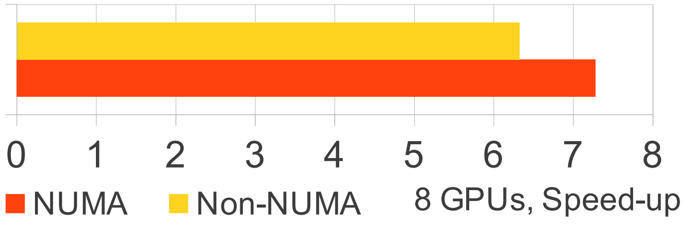


NUMA Effects



With external box configuration

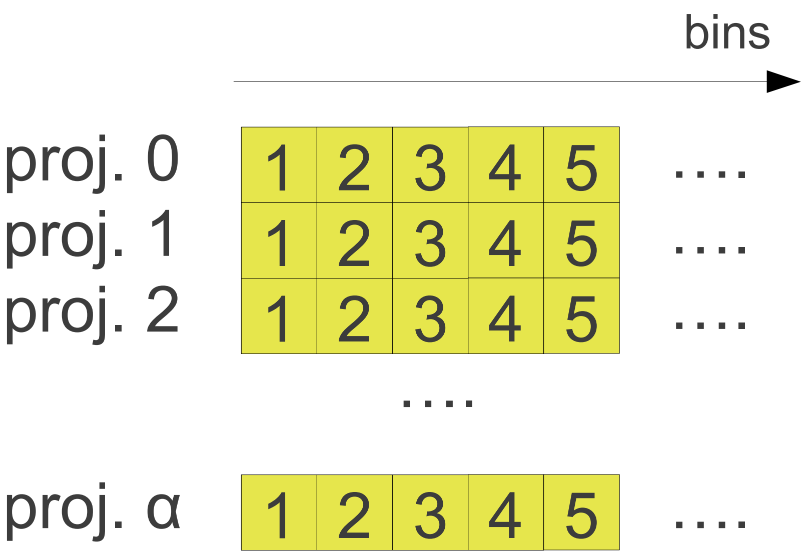




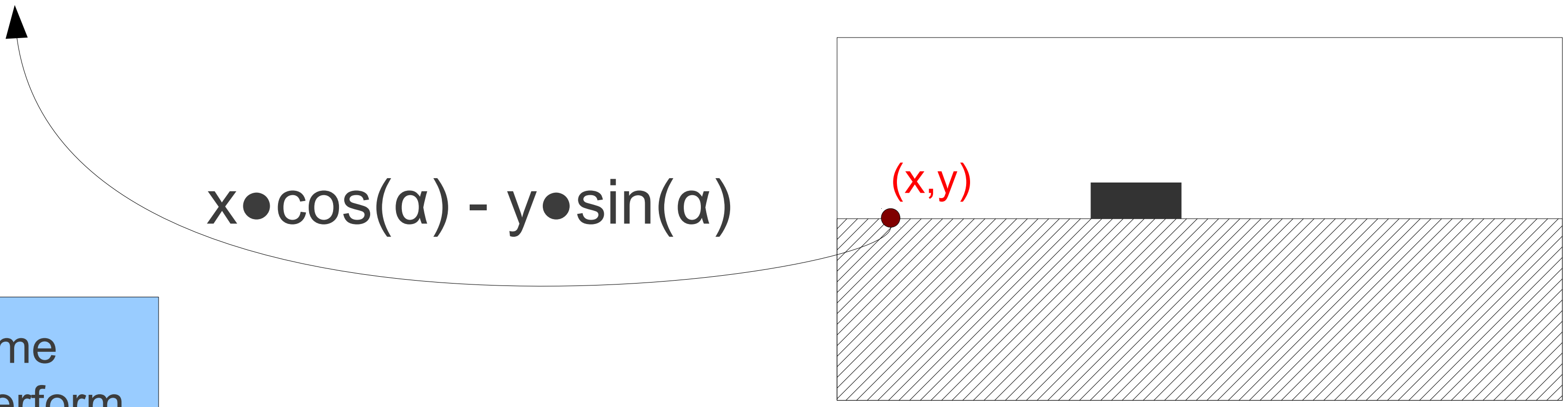
S. Chilingaryan et. all

Institute for Data Processing and Electronics
Karlsruhe Institute of Technology

Back Projection Explained



1. For each position we compute: x●cos(α) - y●sin(α)
2. Compute linear interpolation between 2 neighboring bins
3. Sum over all projection
4. The sum is the value of (x,y)



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

S. Chilingaryan et. all

Institute for Data Processing and Electronics
Karlsruhe Institute of Technology

Fermi: Balancing texture fetches with computations

Karlsruhe Institute of Technology

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

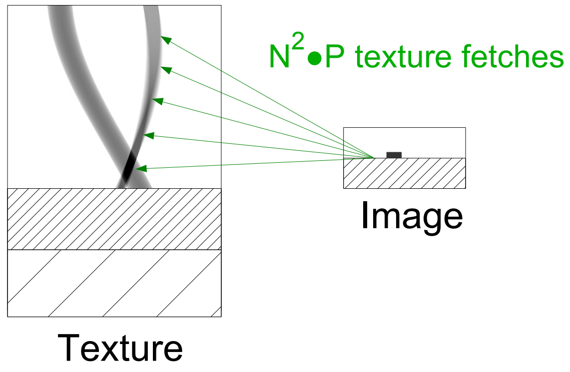
Thread block

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

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

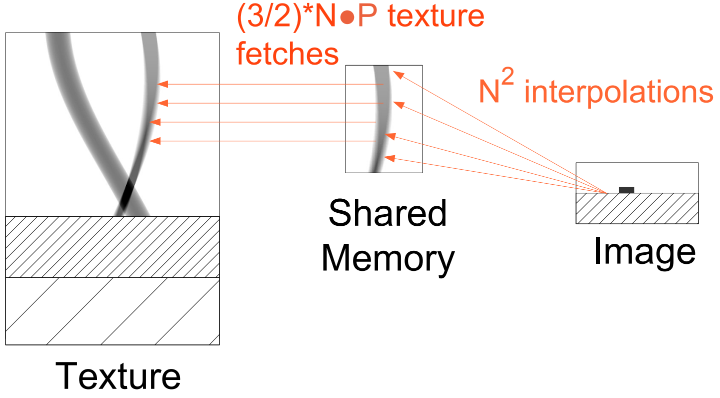
$$N\sqrt{2} < 1.5\,N$$

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



Standard Version

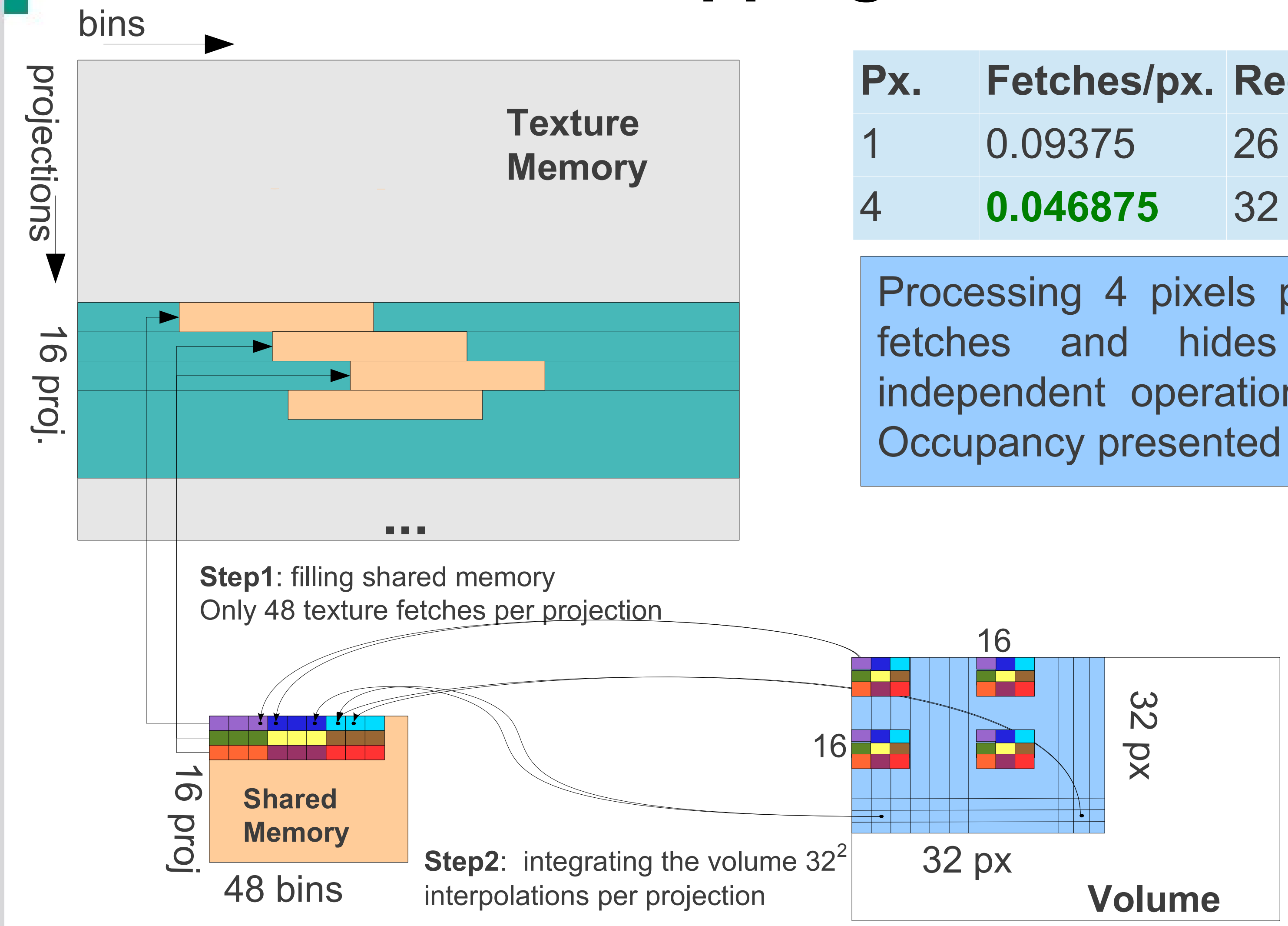
Texture engine is heavily loaded



Fermi-optimized Version

Both texture & computations engines are used

Pixel to thread mapping



Px.	Fetches/px.	Regs	ShMem	Occup.	Ind. Instr
1	0.09375	26	1536	66%	1
4	**0.046875**	32	3072	66%	**4**

Processing 4 pixels per thread reducing amount of texture fetches and hides operation latencies with multiple independent operations (see Better Performance at Lower Occupancy presented by V. Volkov at GTC2010)

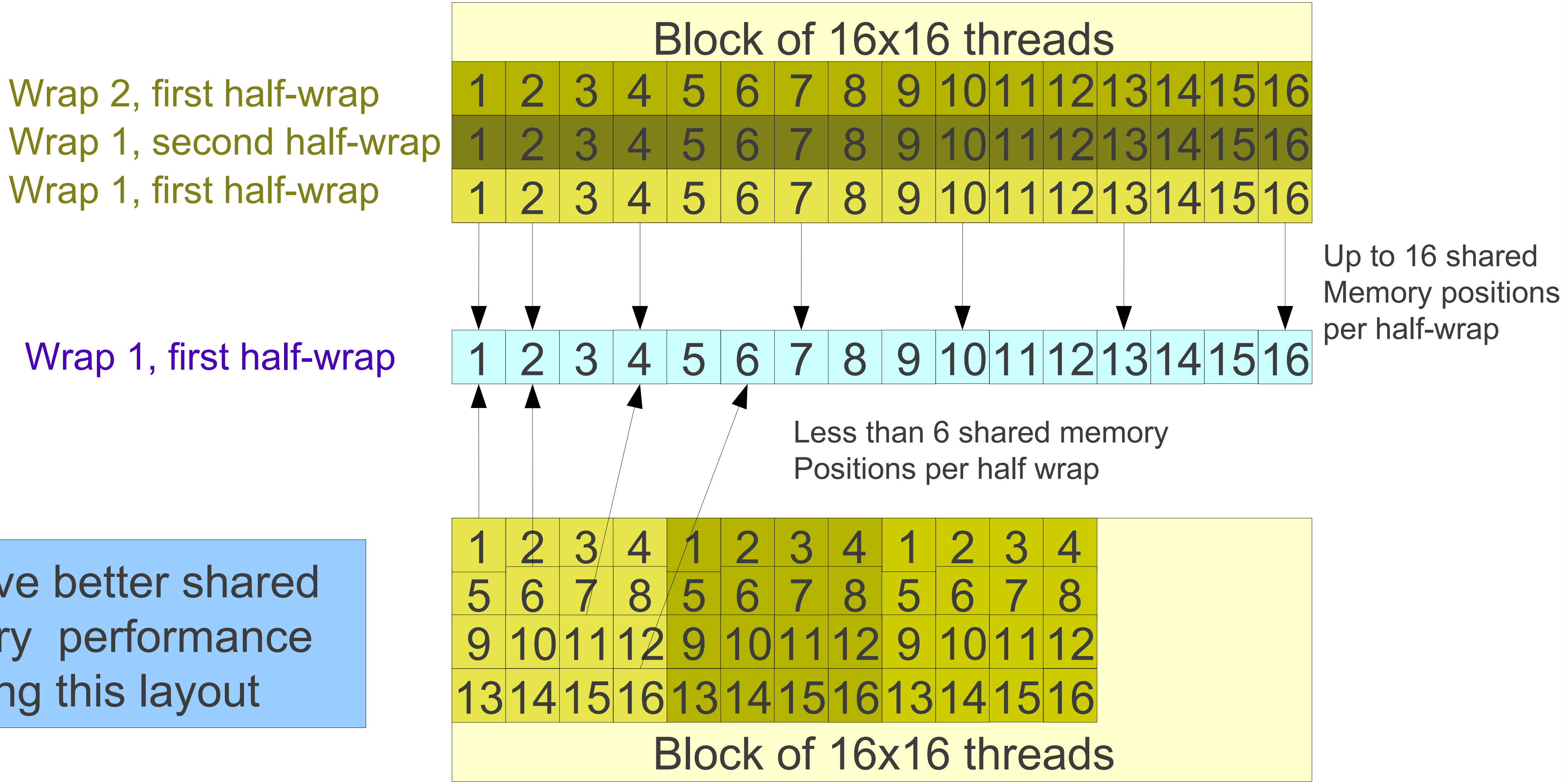
Processing in multiple passes, 16 projections each

Legend

- Processed by a single thread block (16x16)
- 48 bins of a projection required for current block
- 16 of the projections processed in a single pass

thr (1,1) thr (2,1) thr (3,1)
thr (1,2) thr (2,2) thr (3,2)
thr (1,3) thr (2,3) thr (3,3)

Optimizing shared memory reads

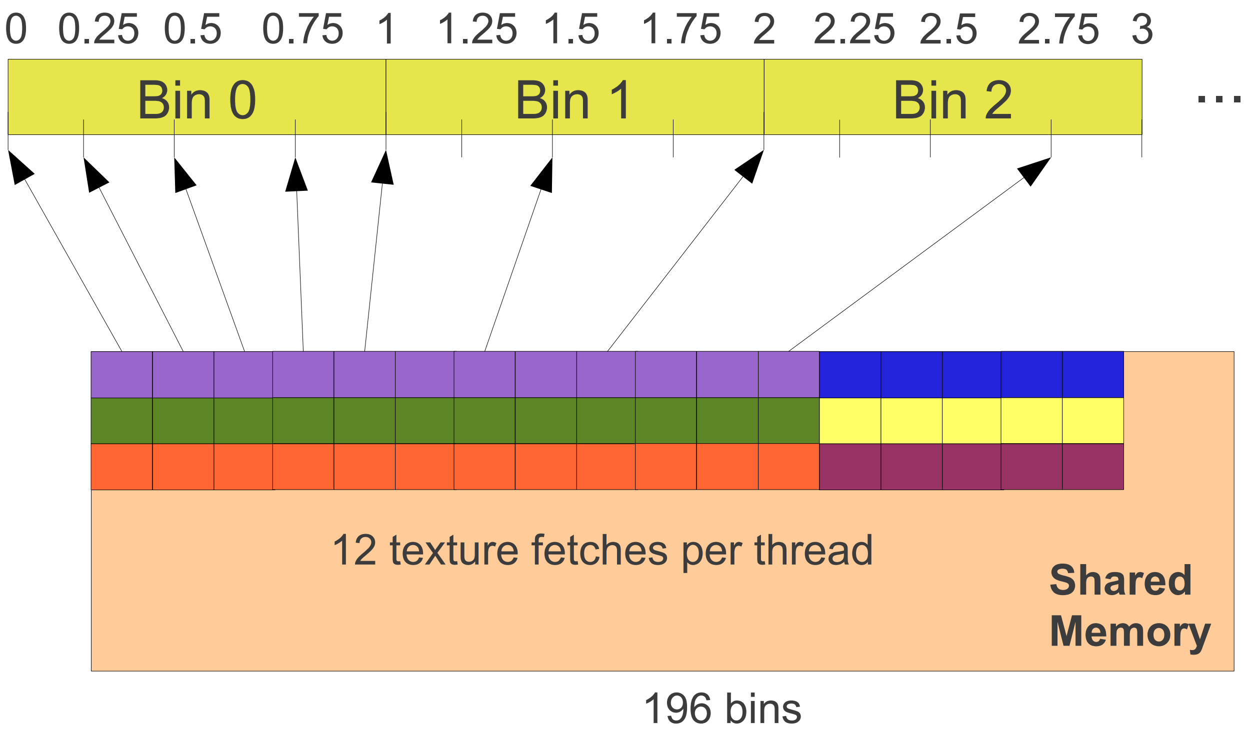


We have better shared memory performance using this layout

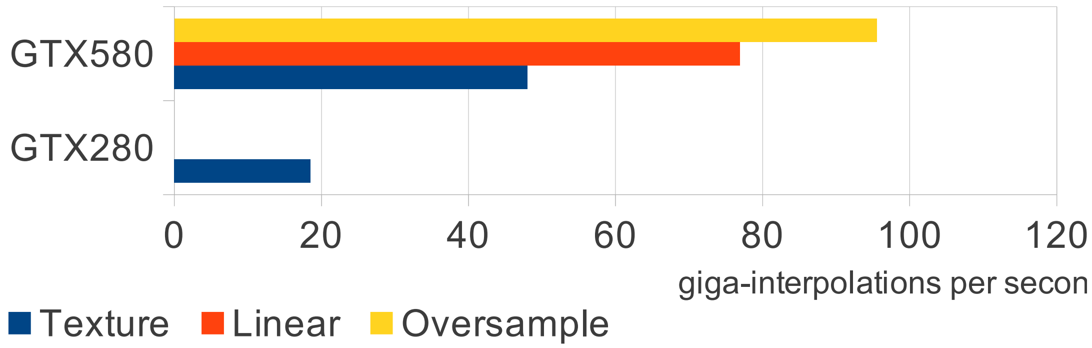
Reducing computation costs with oversampling

Linear interpolation is slow, and nearest neighbor is not precise enough

Method	Fetches/px.	Regs	ShMem	Occup.	Reads.	Flops.
Linear	0.046875	32	3072	66%	2	7
Oversample	**0.1875**	42	12288	**50%**	**1**	**4**



With oversampling the texture engine is used to interpolate 4 positions for each projection bin and near-neighbor interpolation is used then.

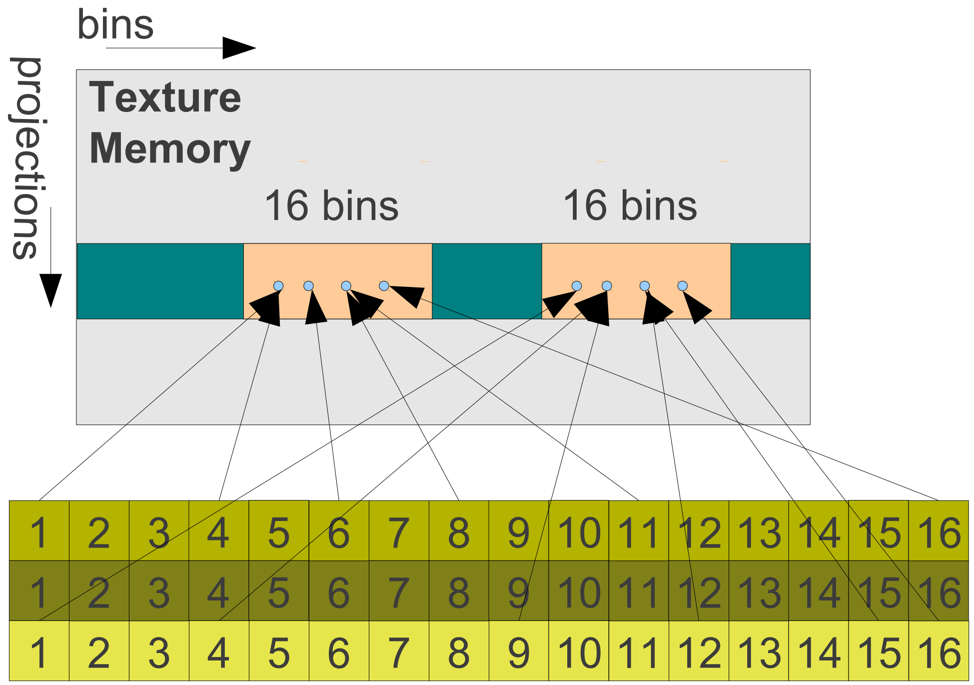


S. Chilingaryan et. all

Kepler: Fast Texture Engine is Back

Simple Texture Method

Texture Cache Hit Rate	89 %
Texture Throughput	79.3 GT/s
Theoretical Throughput	128.8 GT/s



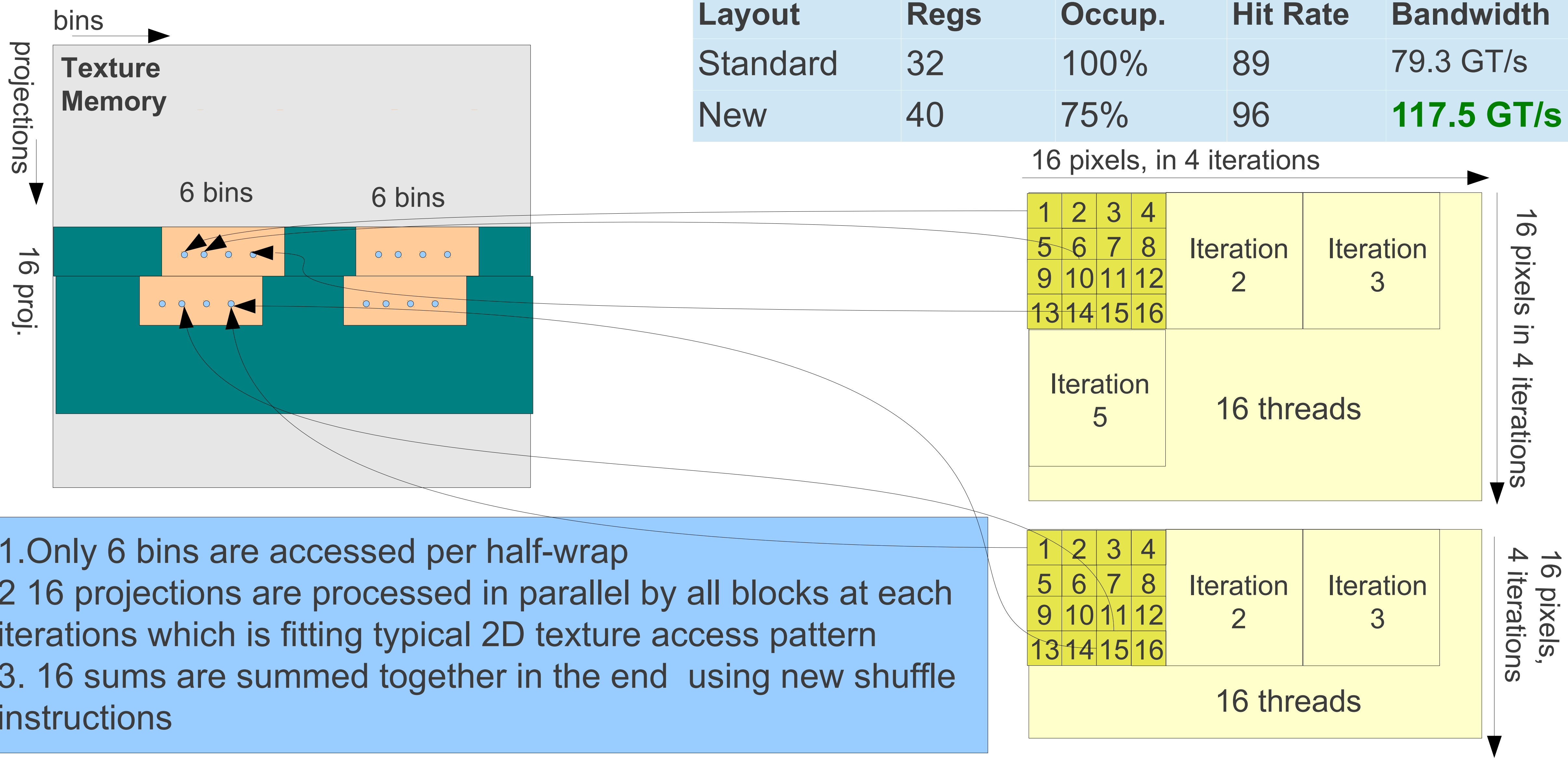
Block of 16x16 threads

	GT580	GTX680	Increase
Texture Engine	49.4 GT/s	128.8 GT/s	2.6 x
Computational Units	16 x 32 x 1.55 GHz	8 x 192 x 1.006 GHz	1.94 x
Int multipl., bit ops., type conv	16 x 16 x 1.55 GHz	8 x 32 x 1.006 GHz	0.65 x
Shared Memory	48 KB	48 KB	1
Blocks per SM	8	16	2
Registers	32K per SM, 63 per thr.	64K per SM, 63 per thr.	

1. Up to 16 bins are accessed per half-wrap,
2. All threads are accessing a single texture row

Institute for Data Processing and Electronics
Karlsruhe Institute of Technology

Optimizing cache efficiency

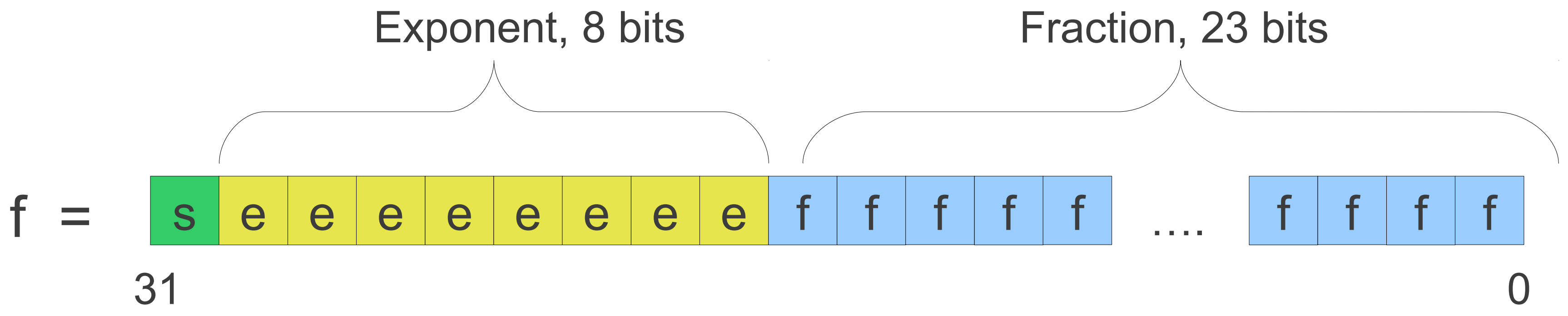


Layout	Regs	Occup.	Hit Rate	Bandwidth
Standard	32	100%	89	79.3 GT/s
New	40	75%	96	**117.5 GT/s**

1.Only 6 bins are accessed per half-wrap
2 16 projections are processed in parallel by all blocks at each iterations which is fitting typical 2D texture access pattern
3. 16 sums are summed together in the end using new shuffle instructions

S. Chilingaryan et. all
Institute for Data Processing and Electronics
Karlsruhe Institute of Technology

Faster rounding



IEEE 754 single-precision floating point number

$$f = -1^{s} \cdot 2^{e-127} \cdot (1 + \sum b_i \cdot 2^{i-23})$$

Only 23 significant positions, for small positive numbers:

$$F + 2^{23} = 2^{23} \cdot (1 + \sum b_i \cdot 2^{i-23})$$

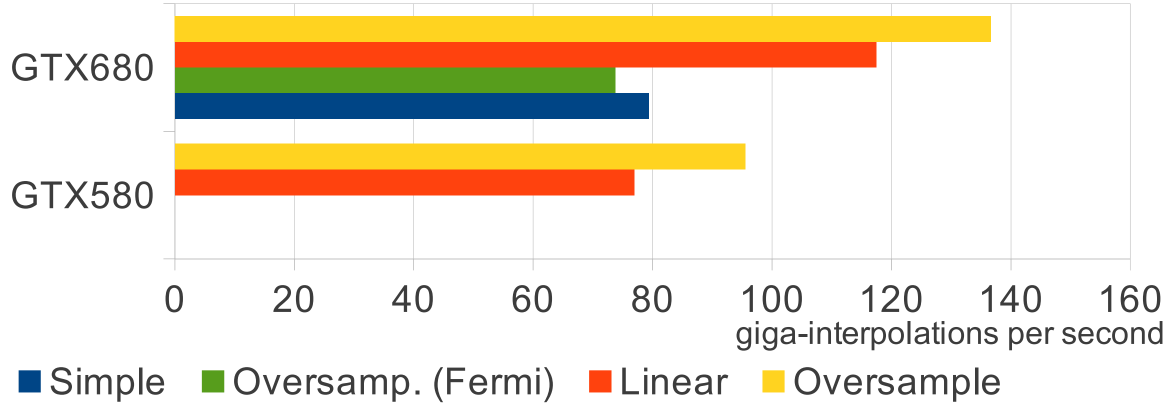
i.e. no fractional part

$$round(f) = f + 2^{23} - 2^{23}$$

$$(int)f = f + 2^{23} - 0x4B000000$$

Kepler Oversampling Algorithm

1. New stage pre-computing per-block offsets
2. Offsets are exchanged using shuffle instruction
3. Faster rounding is not used due overlap of rounding and floating point operations



S. Chilingaryan et. all

Institute for Data Processing and Electronics
Karlsruhe Institute of Technology

AMD Architectures

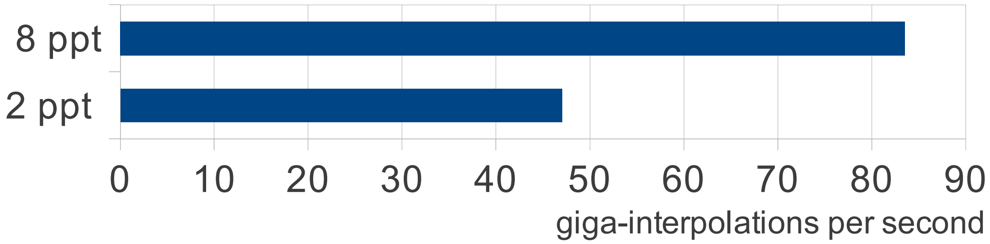
Radeon HD 5970

VLIW5

Requires quintuples of independent operations in command flow:

Block size: 16x16 => 8x8

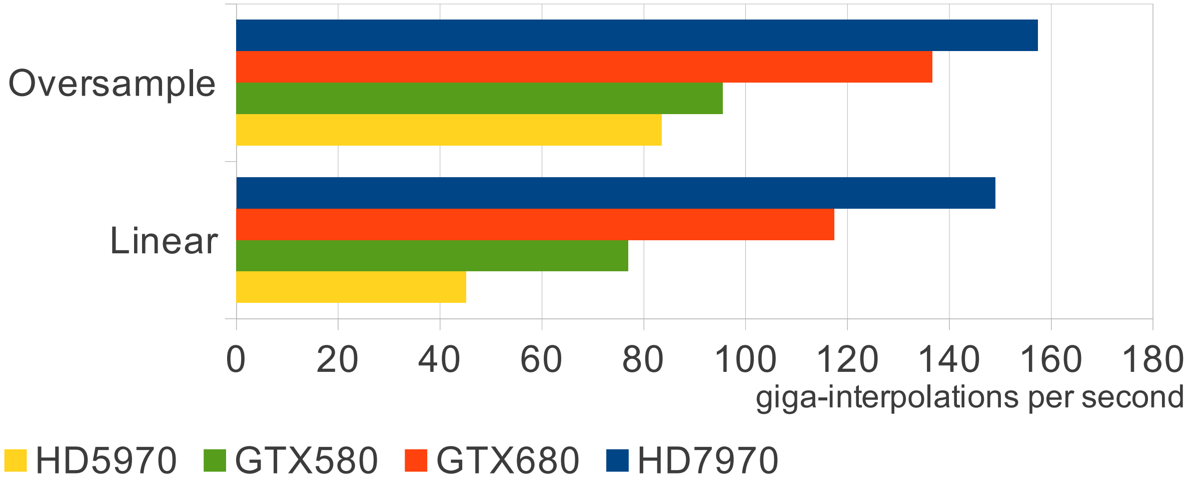
Points per thread: 2 => 8



Radeon HD 7970

GCN

No special tunning required



- Only a single chip running in dual chip configurations
- Memory/Computations overlapping in beta and have not worked in my setup
- Many functions are not optimal, for instance CopyRect family functions are slow
- Compiler Doesn't support local arrays, manual unrolling is required

Back Projection Kernels

GT200
Uses texture engine

Kepler
Uses texture engine, but processes 16 projections at once and 16 points per thread to enhance cache hit rate

Fermi/GCN
Caches textures in shared memory and performs interpolations using computation nodes

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

Fermi/GCN Fast
Oversampling uses texture engine to interpolate values at 4 predefined points and then uses near-neighbor interpolation to avoid costly computations

Cypress Fast
Oversampling algorithm with 16 points per thread.

Kepler Fast
Rounding optimization to get over performance limits of Kepler and usage of new shuffle instructions to exchange data between threads in wrap

Summary

Karlsruhe Institute of Technology

- GPU computing fits extremely well the needs of Synchrotron Imaging
- However, special care required to get to really high speeds
  - Pipelined architecture is efficient way to hide I/O time
  - The architecture-specific optimizations are often required
- We develop a platform for high speed time resoluted X-ray Imaging with possibility of real-time control
- Open-source image processing framework is designed
  - GPU/CPU processing with OpenCL
  - Integration with scripting languages using Gobject-introspection
  - Available from http://ufo.kit.edu/framework
- A programmable camera is currently under design to enable real-time control
  - Up to 1 Mpix at 5000 frames per second
  - Direct connection to Infiniband cluster
  - Programmable integrated logic for real-time control
- A chain of filters for parallel-beam tomography has been developed
  - Throughputs of up to 500 MB/s can be handled with a single PC
  - A clustered solution is under development
-