Practical Experience with GPUs for high throughput computing

03/01/12

Agenda

GPU Computing
High-Throughput Tomography
Software & Hardware
Handling I/O bottleneck
Going to millisecond resolution in 4D

Status: Full throughput of CameraLink Interface Is reached (850 MB/s)

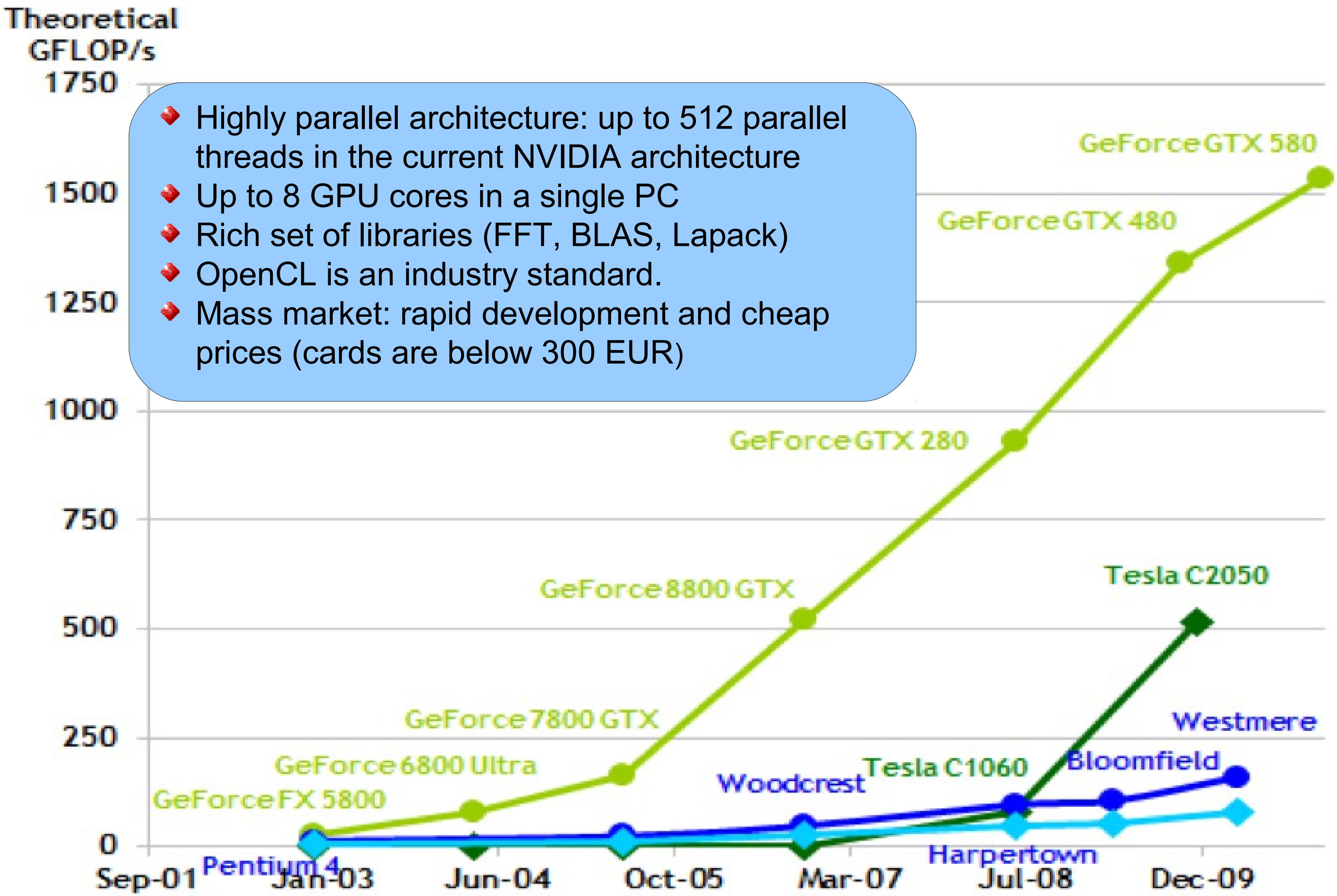
In collaboration with ESRF
European Synchrotron Radiation Facility
Grenoble, France

S. Chilingaryan, M. Vogelgesang, A. Mirone, A. Kopmann

Karlsruhe Institute of Technology
Forschungszentrum Karlsruhe in der Helmholtz-Gemeinschaft
Universität Karlsruhe (TH) Research University • founded 1825

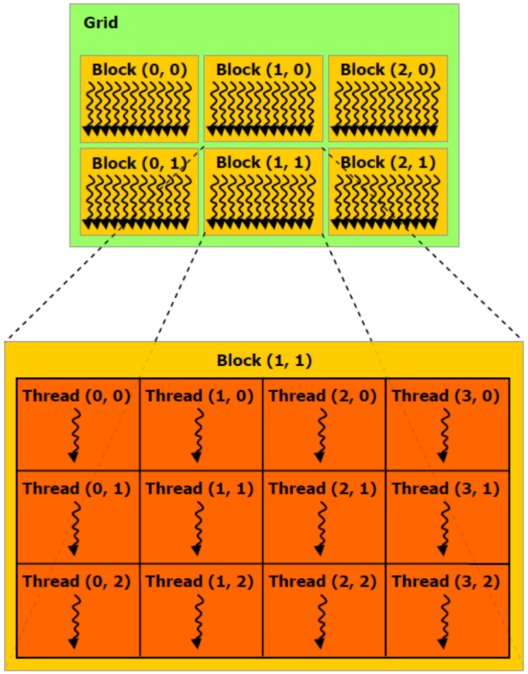
GPU Computing

- 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)



CUDA General Concepts

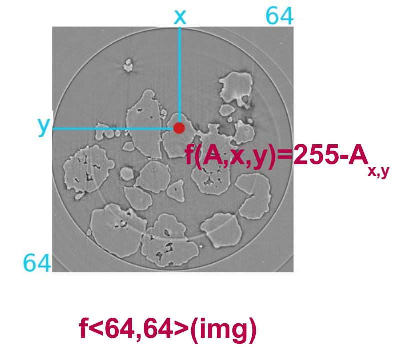
Karlsruhe Institute of Technology



a) Define a Grid of parallel tasks
- Each thread associated with 1D, 2D, or 3D number/coordinate
- Threads within block are executed in parallel and share fast memory

b) Define a kernel function
- The thread coordinate available in function context

c) Specify dimensions and execute kernels
- Not limited by number of processors
- Number of threads in block <= 512
- Number blocks in grid <= 65535*65535



Task Grid Definition

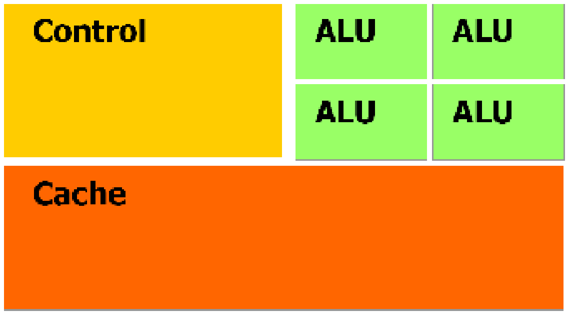
```
Dim3 blocks(16,16);
Dim3 grid(M/16, N/16);
gpuInverse<<<grid,blocks>>>(res,img);
```

CUDA Kernel

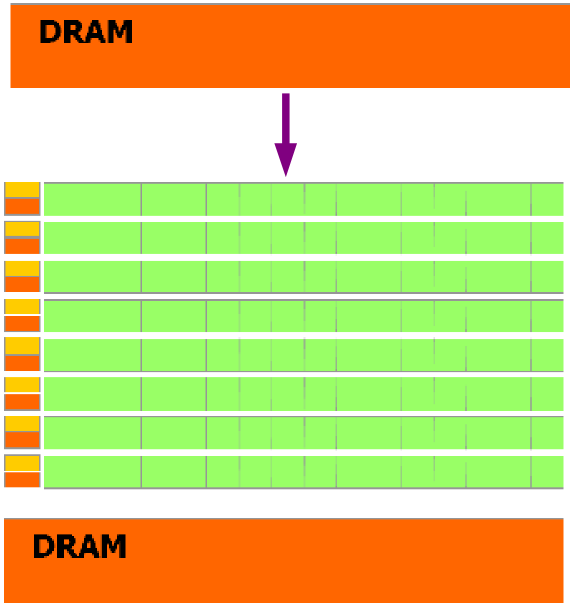
```
__global__ void gpuInverse(float **res, float **img) {
        int i = threadIdx.x, j = threadIdx.y;
        res[i][j] = 255 - img[i][j];
}
```

Challenges in GPU Computing

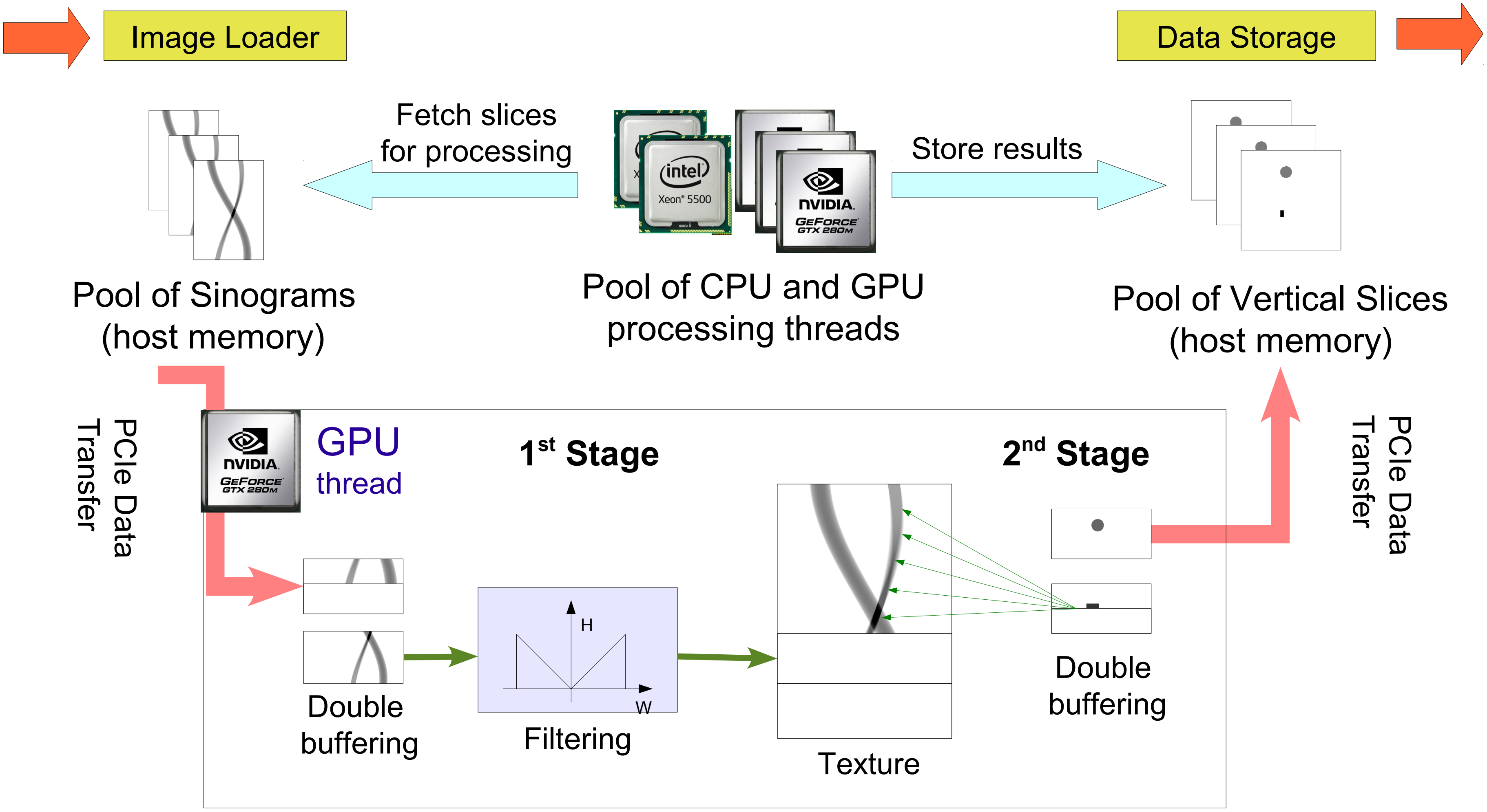
- Highly parallel architecture: up to 512 parallel threads in the current NVIDIA architecture
- Up to 16 GPU cores in a single PC
- Rich set of libraries (FFT, BLAS, Lapack)
- OpenCL is an industry standard.
- Mass market: rapid development and cheap prices (cards are below 300 EUR)



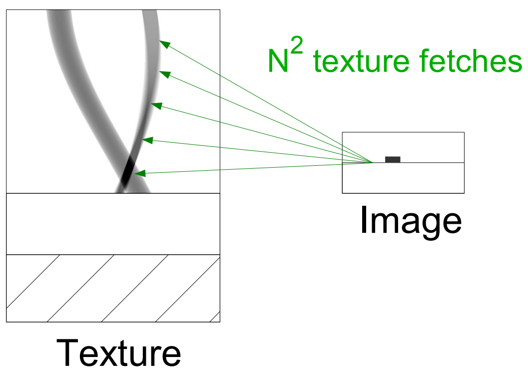
- **Data Transfer (only ~ 6 GB/s between GPU and memory)**
  - Reduce amount of data transfers
  - Use pinned memory for transfer if possible
  - Interleave data transfers with computations
- **Memory (multiple hierarchies, limited cache)**
  - Memory allocation & cleanup are expensive
  - Data alignment 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



Parallel-beam Filtered Back Projection



FBP on Fermi



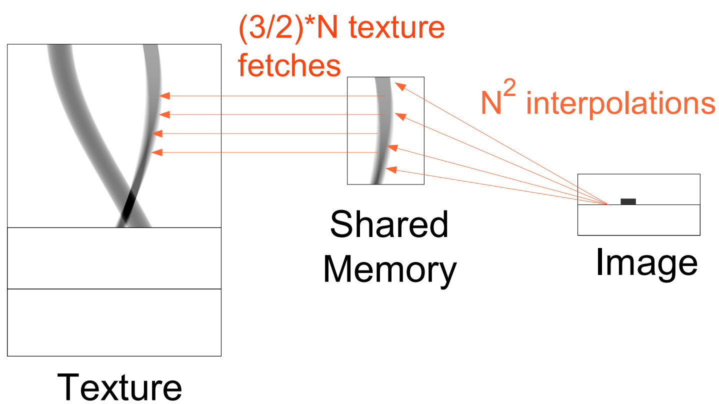
Standard Version

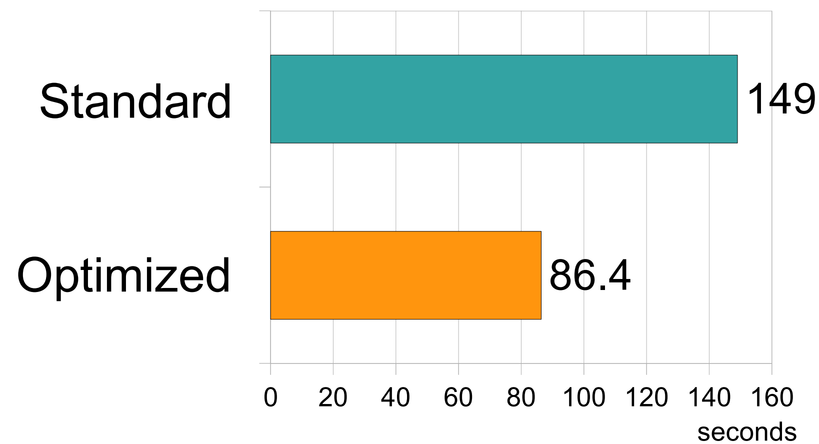
Texture engine is heavily loaded

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



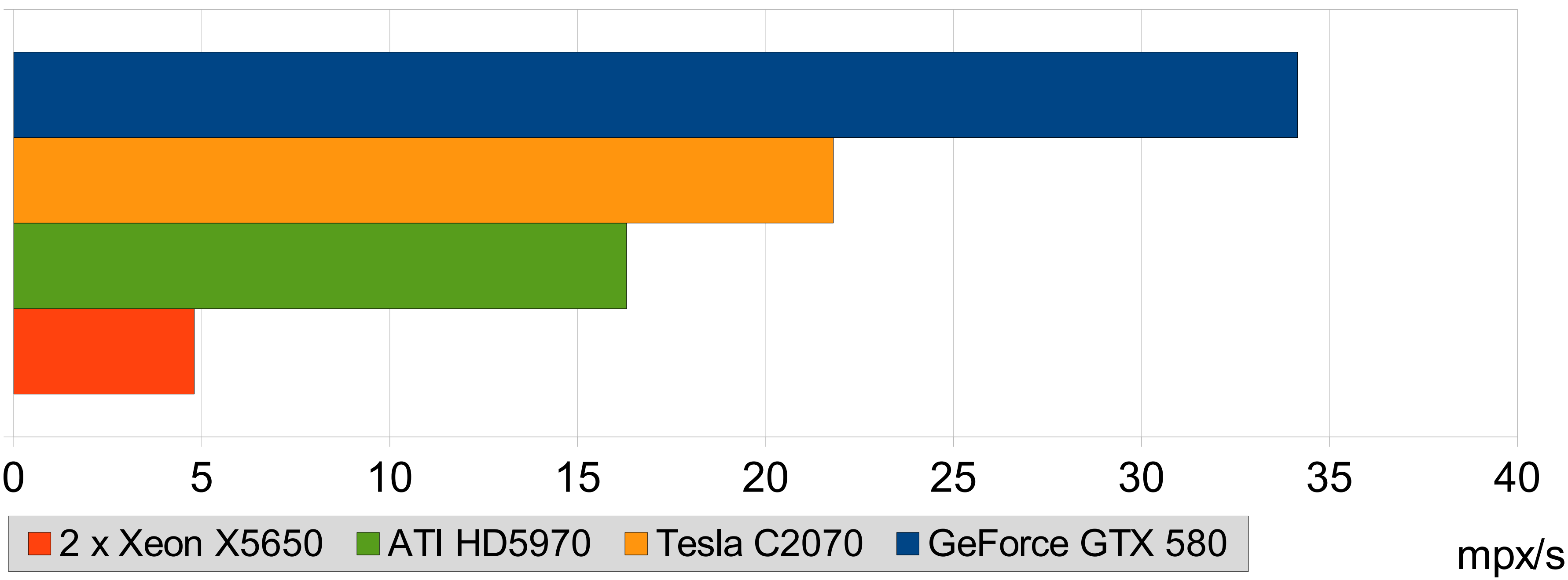


Fermi-optimized Version

Both texture & computations engines are used

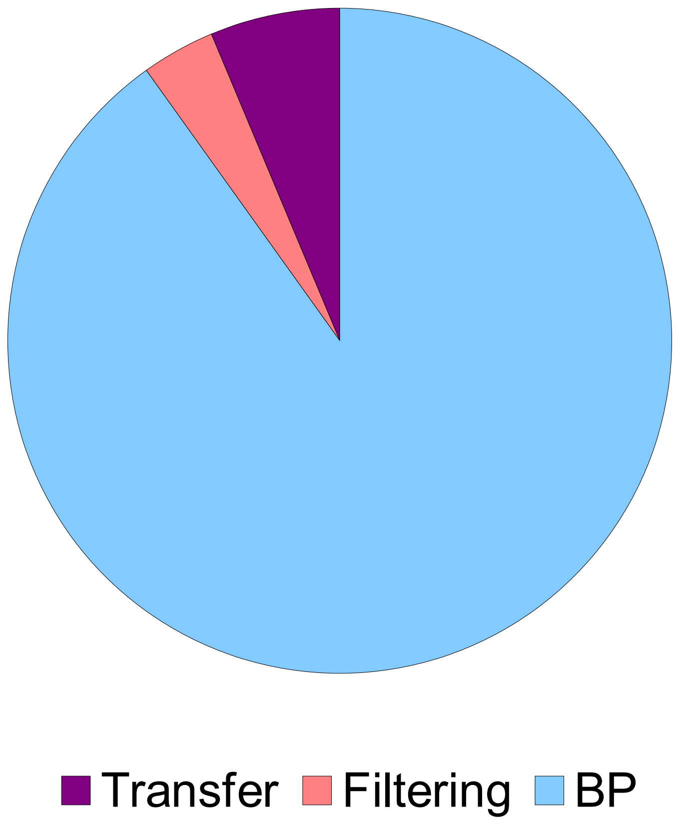
Performance: GeForce vs. Tesla vs. ATI

	GTX580	Tesla C2070	ATI HD5970	Xeon X5650
Threads	512 @ 1.54 GHz	448 @ 1.15GHz	3200 @ 0.725 GHz	6 @ 2.66 GHz
SP/DP	1.58 / 0.20 TF	1.03 / 0.51 TF	4.64 / 0.93 TF	0.13 / 0.06 TF
Memory	1.5 @ 192 GB/s	6 @ 144 GB/s	1 @ 128 GB/s	96 @ 32 GB/s
Texture	1 x 49.4 GT/s	1 x 42 GT/s	2 x 58 GT/s	

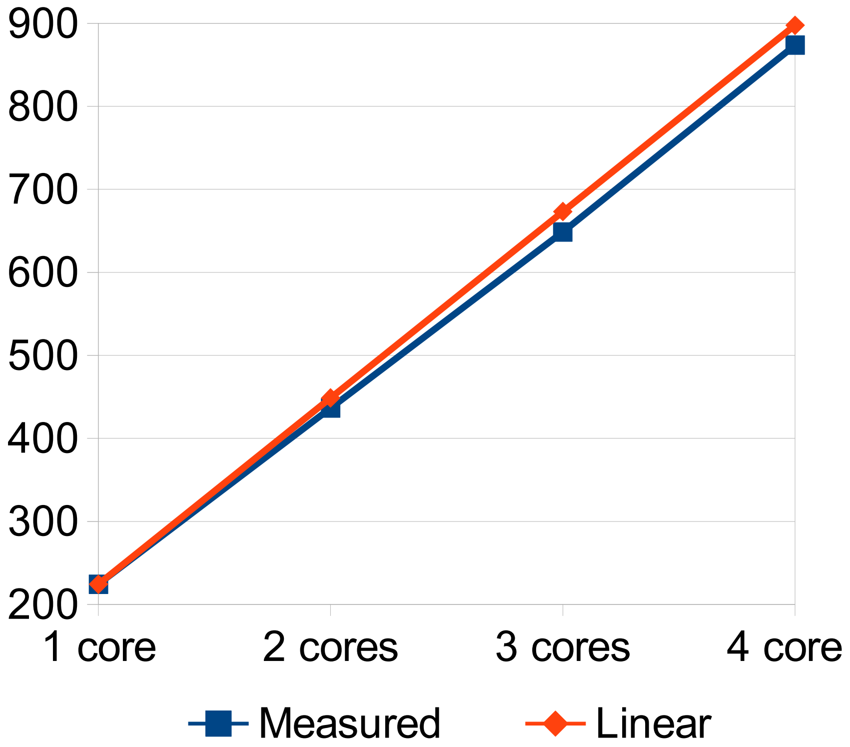


Analysis

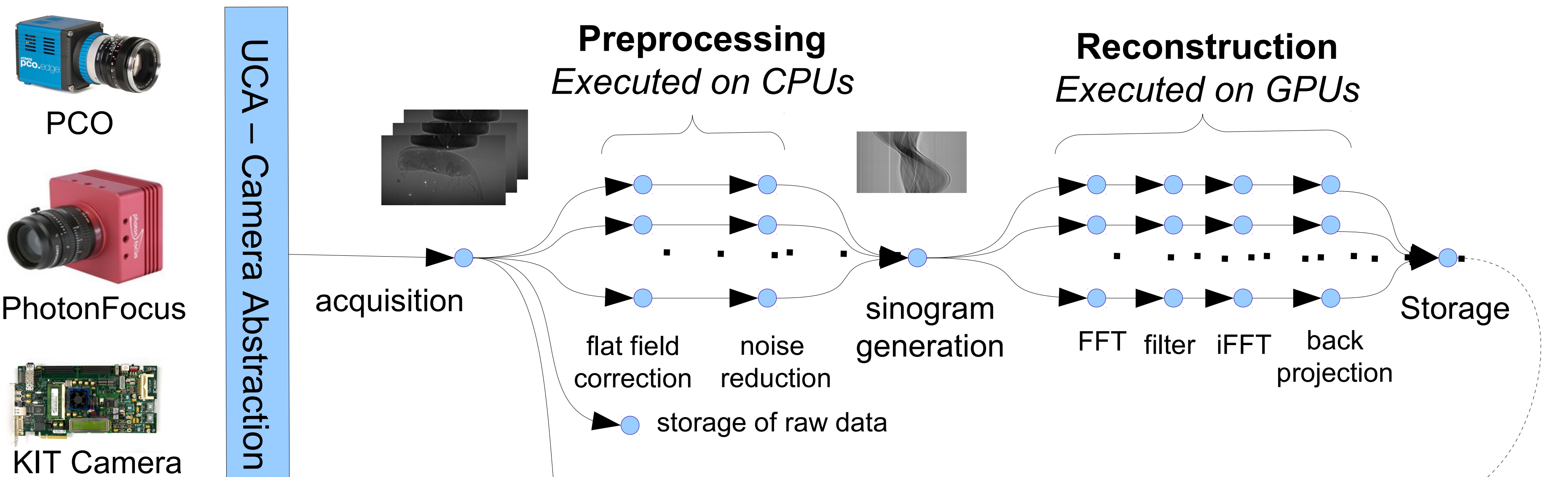
Ratios of used time
Using GTX280



Scalability
Using Tesla S1070

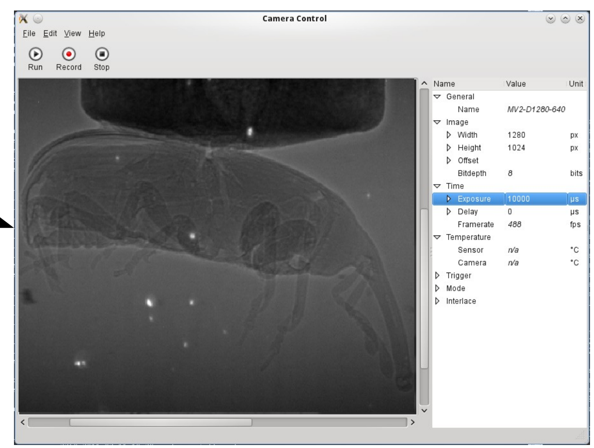


Software: UFO Framework

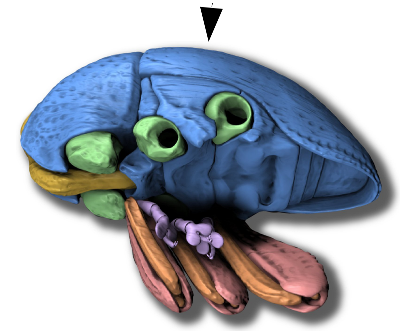


Sample code

```
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

Hardware: Reconstruction Station

<u>SuperMicro 7046GT-TRF</u> (Dual Intel 5520 Chipset)
CPU: 2 x Xeon X5650 (total 12 cores at 2.66 Ghz)
GPUs: 2 x GTX 580 + 4 x GTX580 External
Memory: 96 GB / 12 DDR3 slots (192GB max)
Storage: Areca ARC-1880x SAS Raid
16 x Hitachi A7K200 (Raid6)
4 x Crucial RealSSD C300 (Raid0)
Network: Intel 82598EB (10 Gb/s)
Camera Link Frame Grabber (850 MB/s)

CameraLink 850MB/s

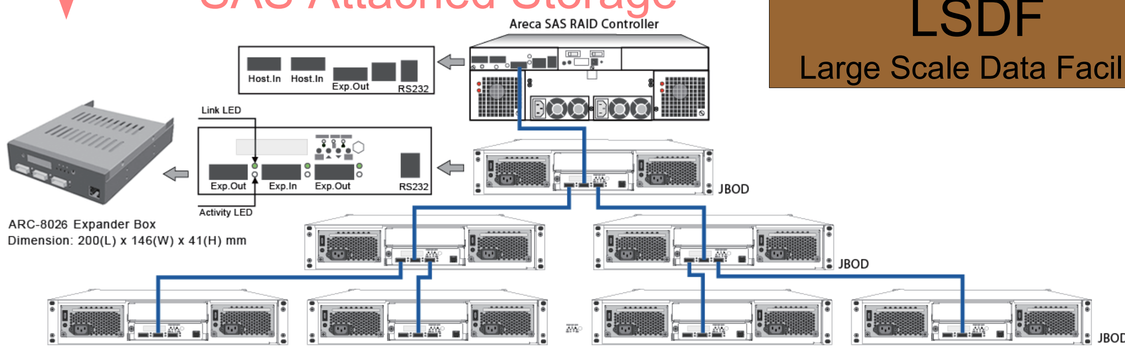
External PCIe x16 (8 GB/s)

Ethernet (10 Gb/s)

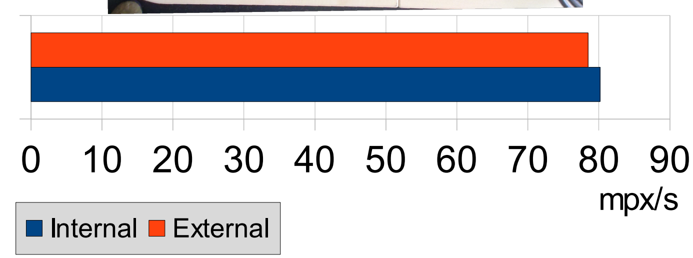
SFF8088 (2.4 GB/s)

External GPU Box

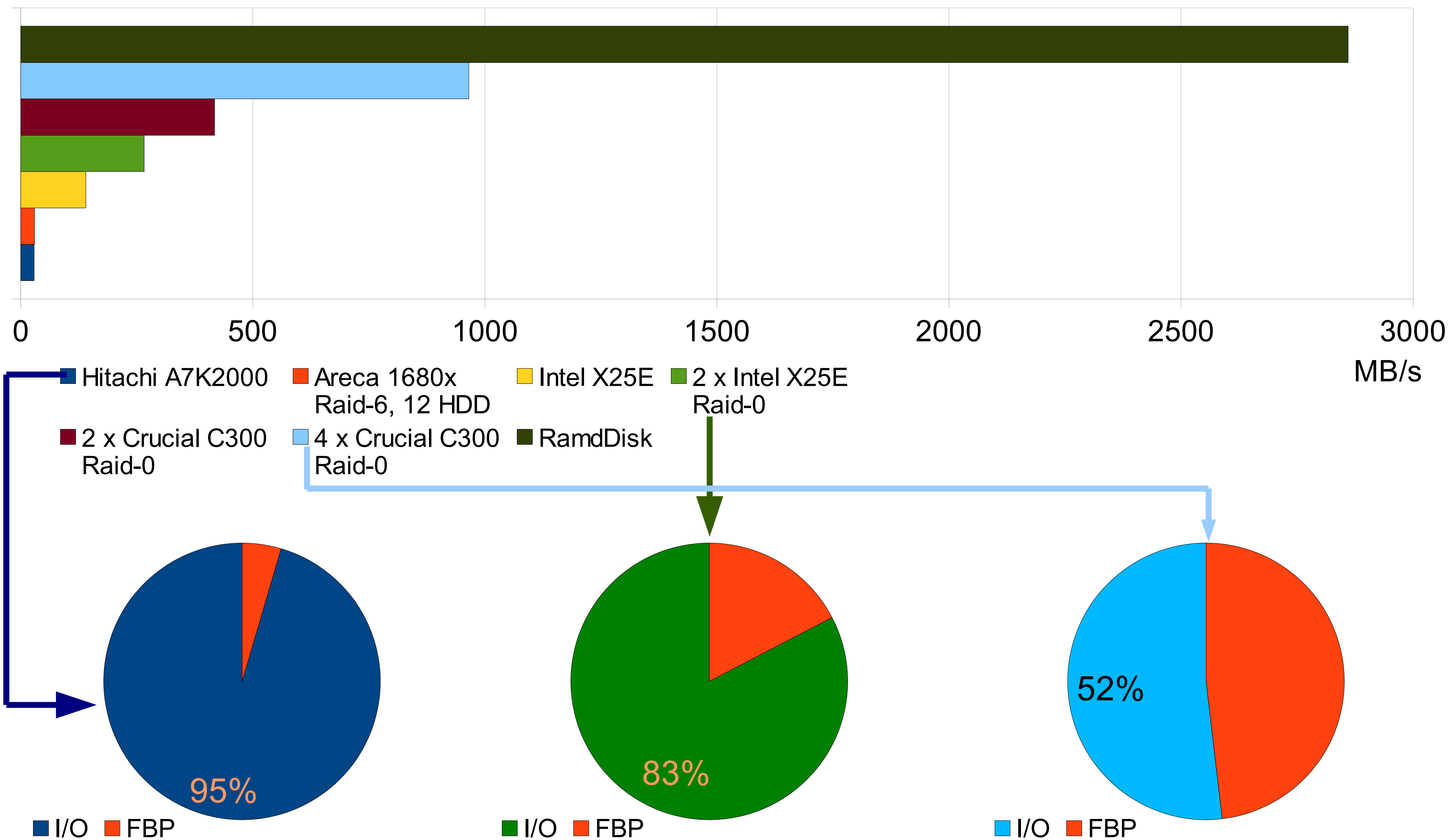
SAS Attached Storage



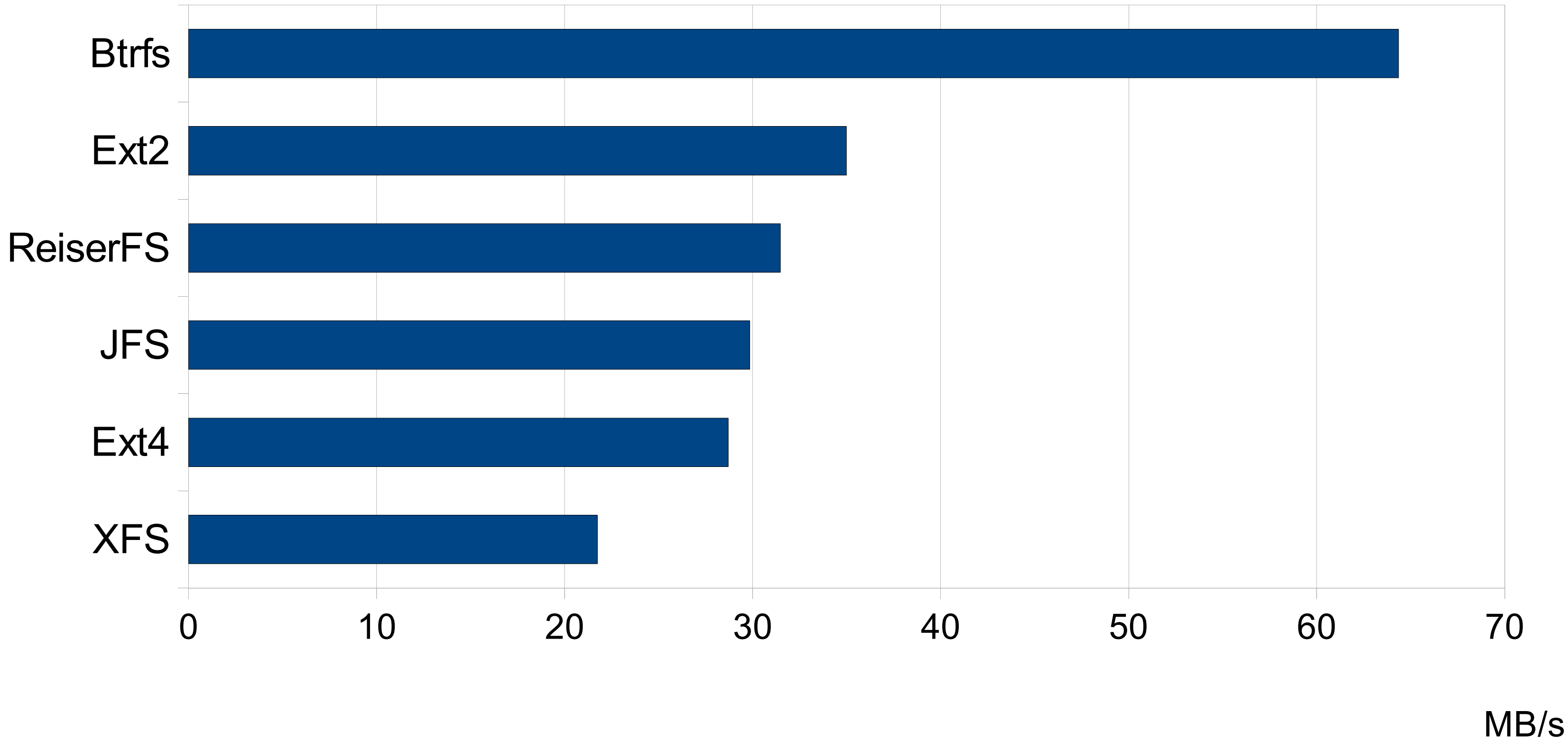
LSDF
Large Scale Data Facility



I/O: Reading EDF Files



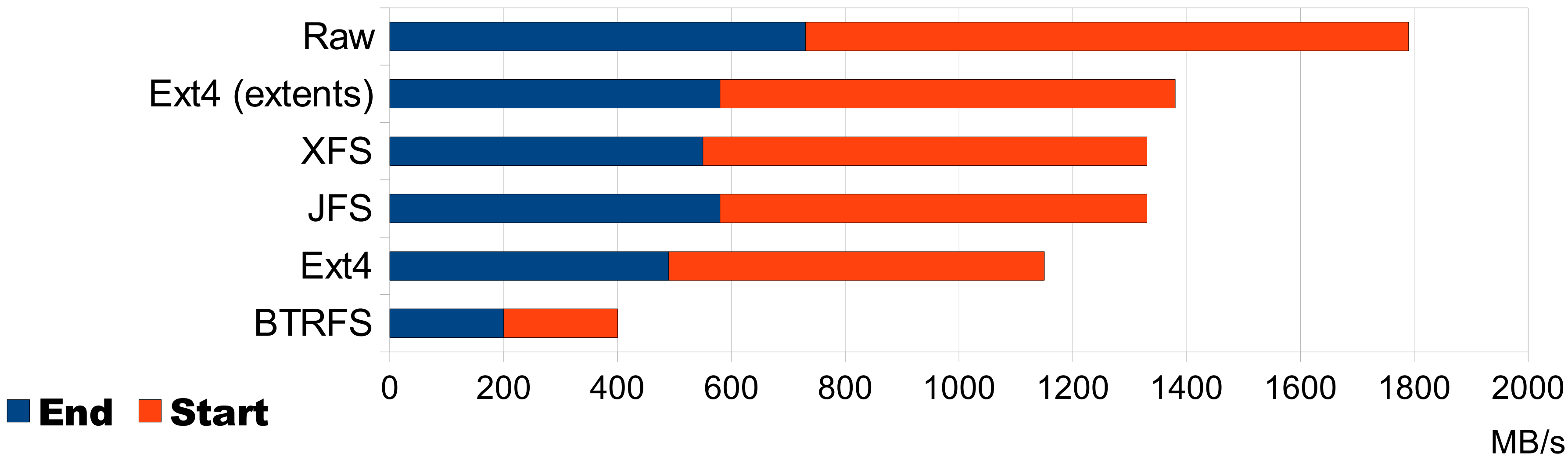
I/O: Influence of File System



* Performance measured with Hitachi A7K2000 hard drive

I/O: Streaming the Acquired Images

16 Hitachi A7K200 in Raid0, OpenSuSe 11.3 Kernel 2.6.34



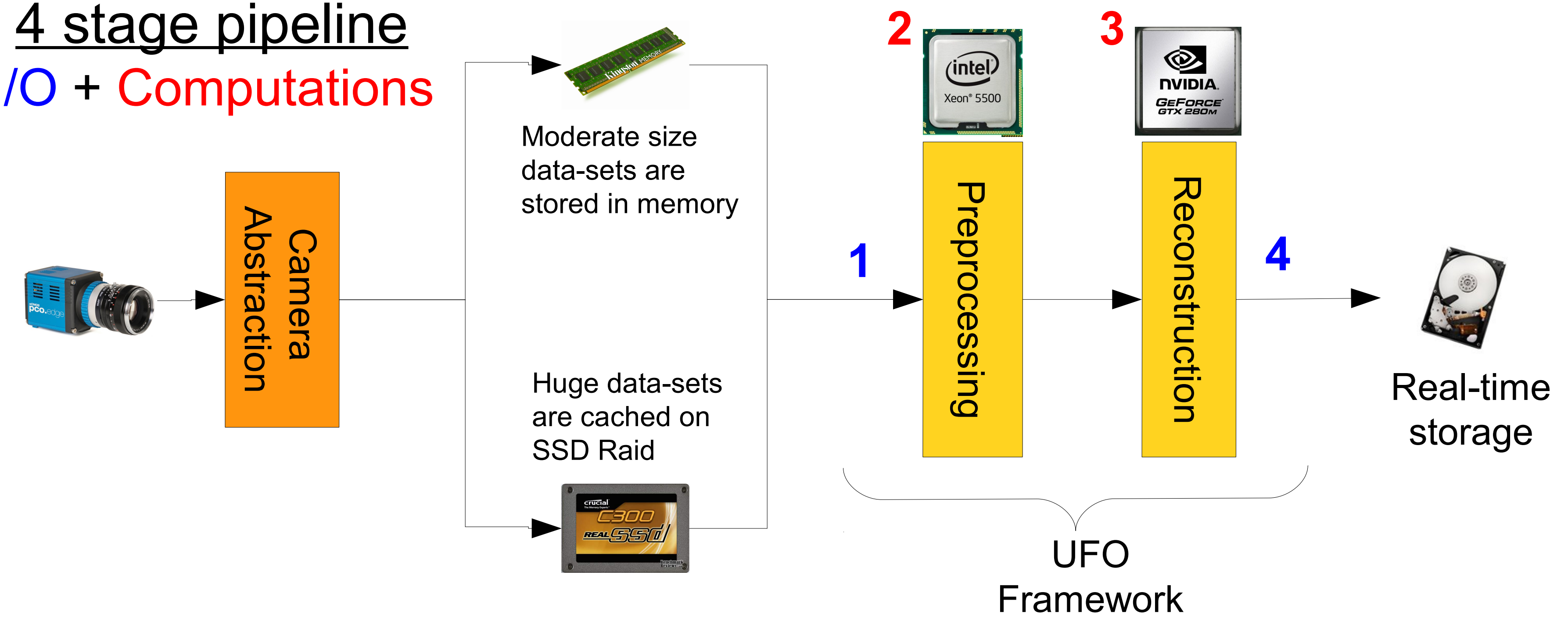
- We lost 30% with fastest FS available and with growth of speed FS penalty grows in percent
- Ext4 performance drops significantly if free space comes to the end. XFS on other hand have spurious reductions of speed on empty disk.
- Fragmentation reduces performance

Hiding I/O with Pipelined Architectures

Karlsruhe Institute of Technology

4 stage pipeline

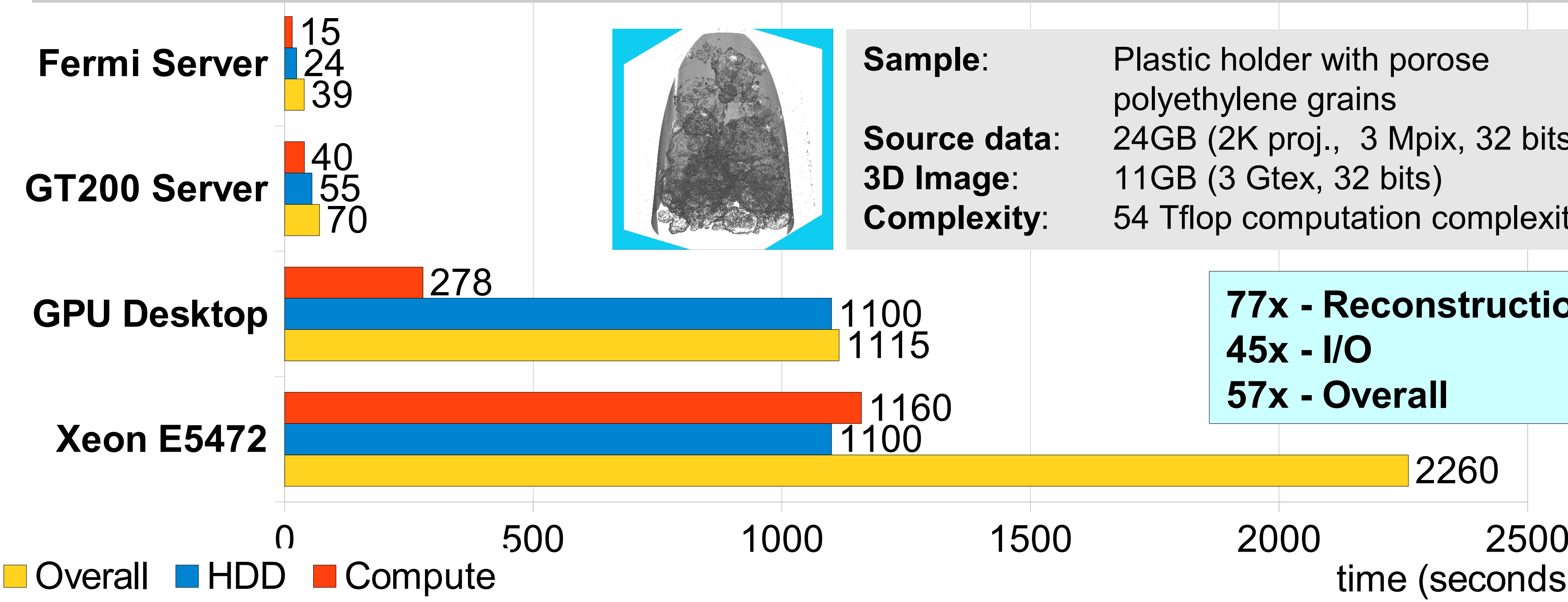
I/O + Computations



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

Performance

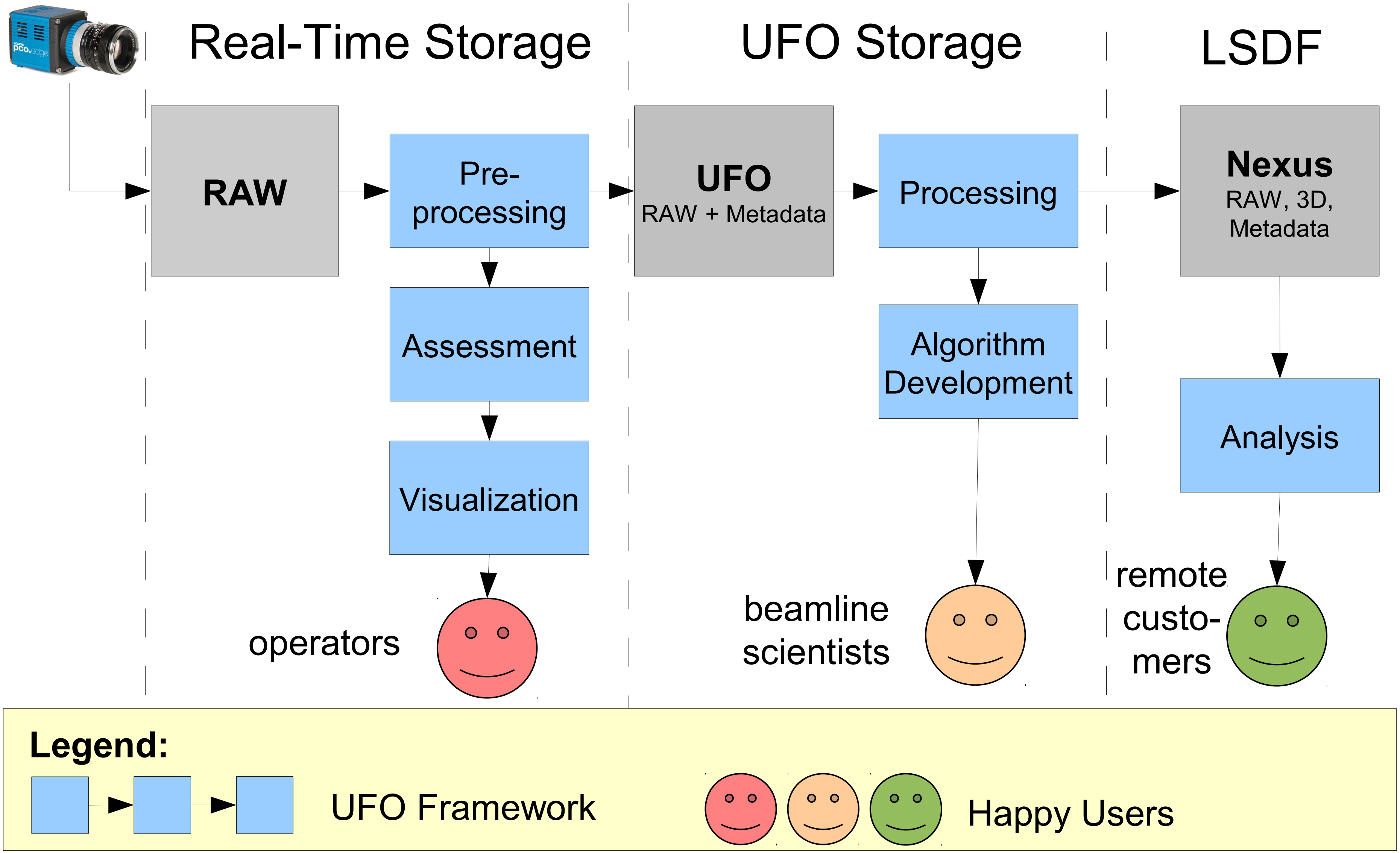
	Xeon Server	GPU Desktop	GT200 Server	Fermi Server
Computational Units	Xeon E5472 8 core, 3 GHz	**GTX 280** **1 core**	**3 x GTX295** **6 cores**	**6 x GTX580** **6 cores**
CPU	2 x Xeon E5472	Core2 E6300	2 x Xeon E5540	2 x Xeon E5540
Memory	16GB DDR3	4GB DDR2	96GB DDR3	96GB DDR3
HDD	WDC5000AACS	WDC5000AACS	**2 x Intel X25-E**	**4 x Crucial C300**
Software	OpenSuSe 11.4, CUDA 3.2, Intel MKL 10.2.1, gcc4.4 Compilation Flags: -O3 -march=nocona -mfpmath=sse			



Sample: Plastic holder with porose polyethylene grains
Source data: 24GB (2K proj., 3 Mpix, 32 bits)
3D Image: 11GB (3 Gtex, 32 bits)
Complexity: 54 Tflop computation complexity

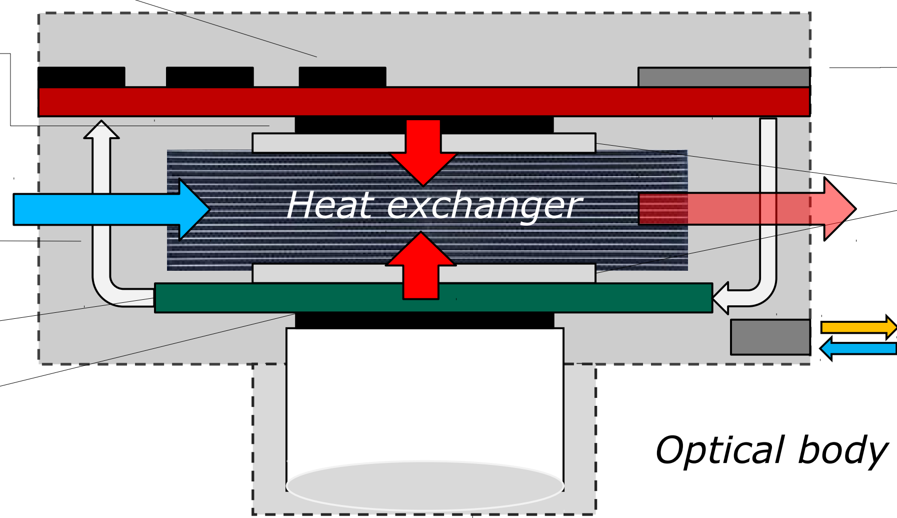
77x - Reconstruction
45x - I/O
57x - Overall

Data Flow

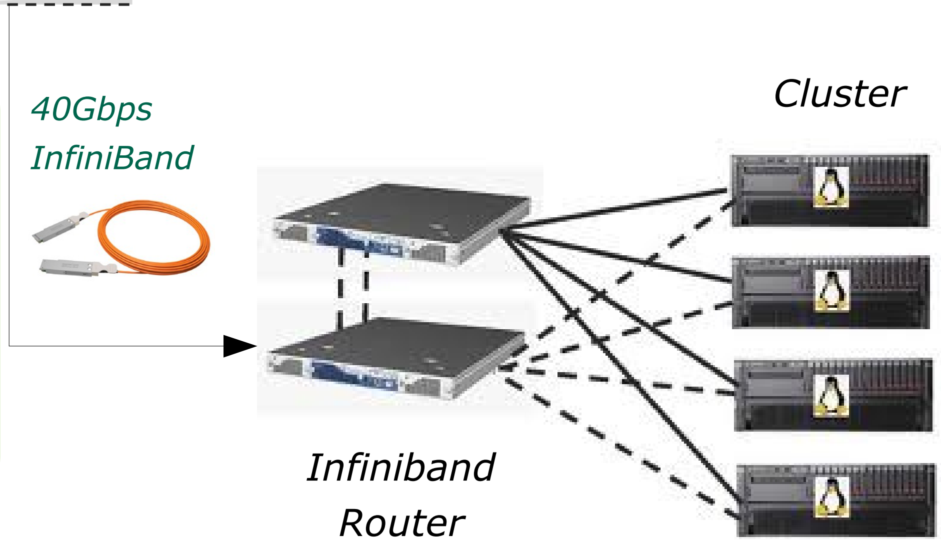


Real-Time Storage
UFO Storage
LSDF
RAW
Pre-processing
Assessment
Visualization
operators
UFO
RAW + Metadata
Processing
Algorithm Development
beamline scientists
Nexus
RAW, 3D, Metadata
Analysis
remote custo-mers
Legend:
UFO Framework
Happy Users

Breaking Limits: Camera & Cluster



- ***High speed CMOS sensor***
  1Mpix, 5000 fps, 10 bits
- ***Self-trigger (fast reject) & Data compression***
- ***On-line elaborations and control***
- ***Direct connection to Infiniband-cluster***
- ***Real-time data processing on the cluster***
- ***Image-based feedback control loop***



Summary

- GPU Architecture serves many needs of scientific community
  - All hardware resources may be used simultaneous
  - Speed-up of 50 – 100 times are possible
- 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
- Flexible and easy to use framework is developed
- Real-time 50,000 fps setup is under development