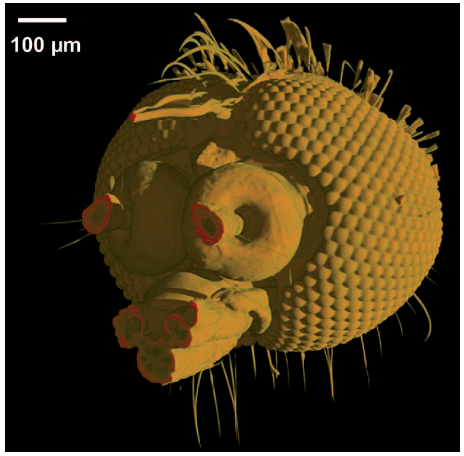
A GPU-based Architecture for Real-Time Data Assessment at Synchrotron Experiments

Karlsruhe Institute of Technology

Agenda: Problem Description,
Performance Evaluation,
What do we do to run faster?
Hardware Evaluation

Goal: Process up to 30GB of image Data in a minute (*1 Tflop/s required*)



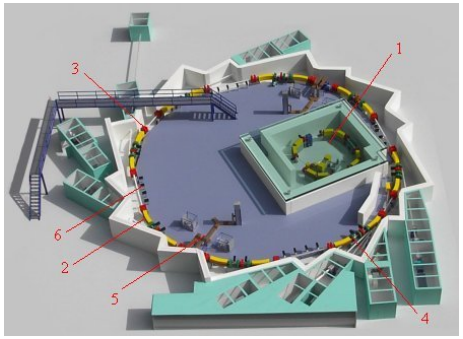
Mosquito head

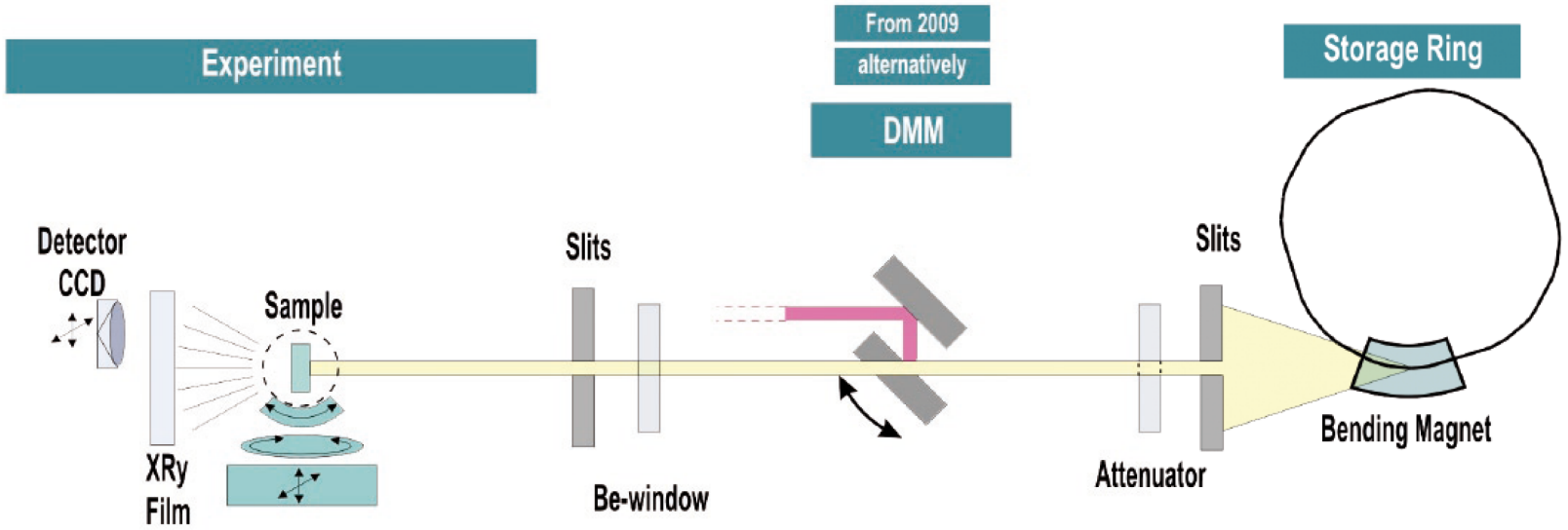
In collaboration with ESRF:
European Synchrotron Radiation Facility
Polygone Scientifique Louis Néel, 6 rue Jules Horowitz, 38000 GRENOBLE

S. Chilingaryan and A. Kopmann, IPE

Synchrotron Tomography

The sample in front of a pixel detector is penetrated by X-rays produced in the synchrotron.

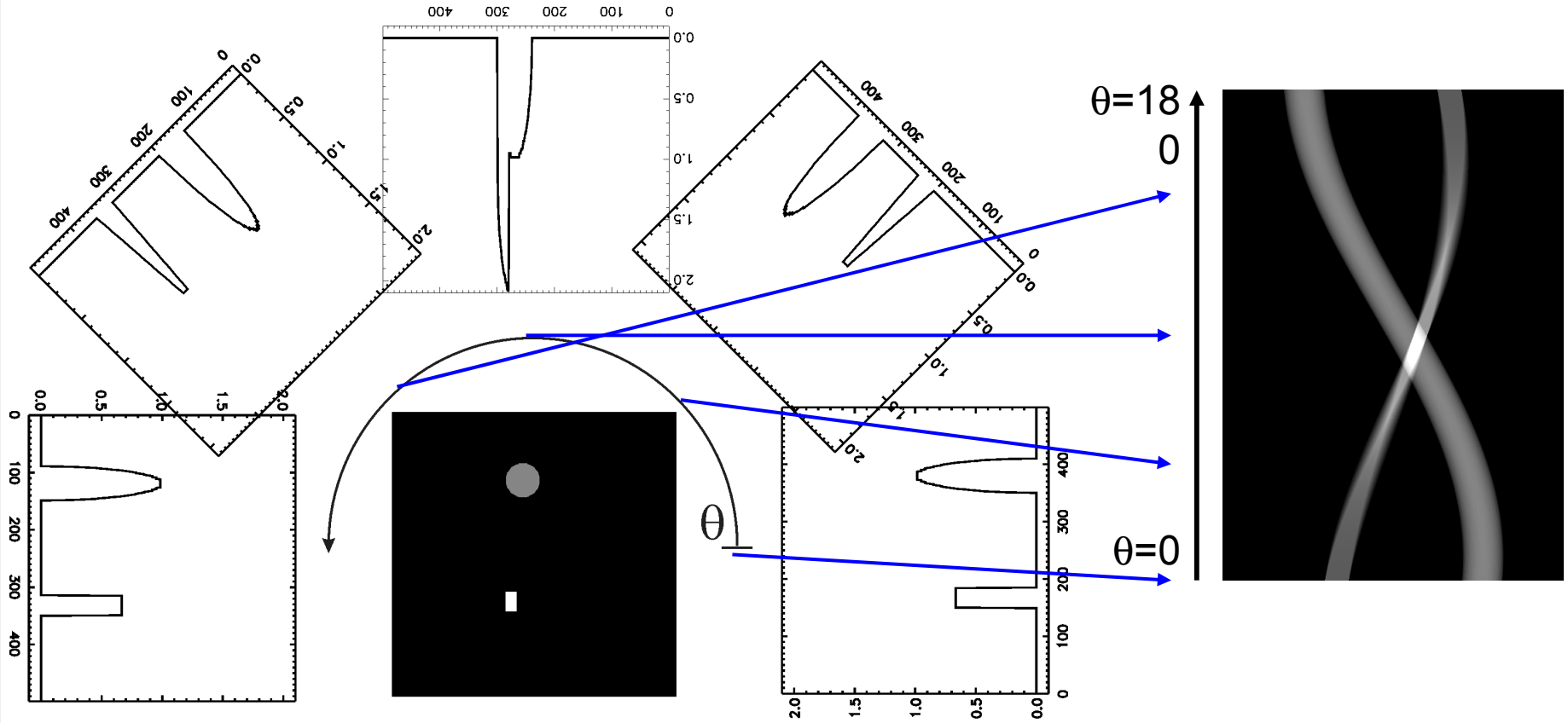




TOPO-TOMO beamline at ANKA

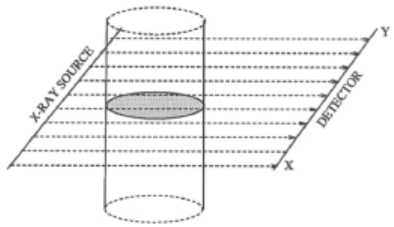
Imaging

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



Reconstruction

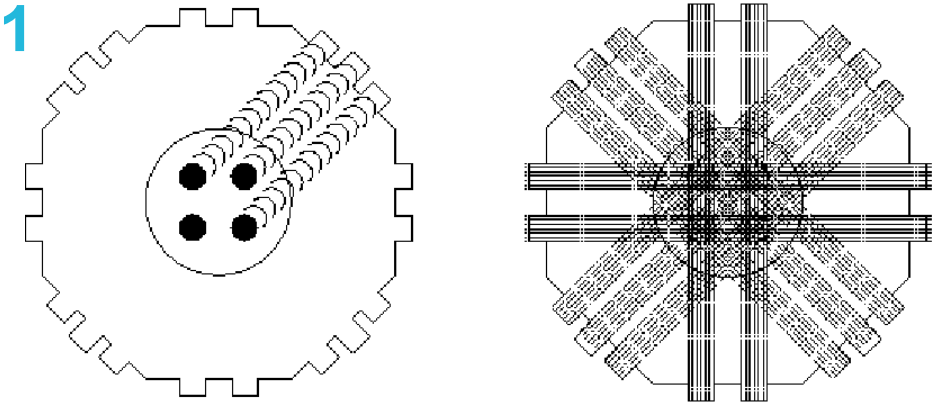
Filtered back-projection is used to produce 3D images from a manifold of two dimensional projections.



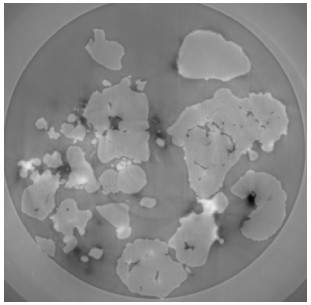
Reconstruction, slice by slice:

1. **Back-projection** - the projections are replicated along the projective angle and integrated.
2. **Filtering** - ramp filter is used to sharpen details

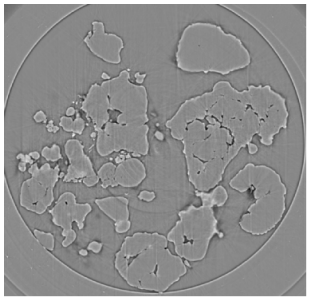
1



2



unfiltered



filtered

Number of operations: *~10 * number_of pixels * number_of_projections*
This makes *10 Tflop for image of 1000^3 pixels and 1000 projections*

Sample Data-Set

Porose polyethylene grains in a conical plastic holder

Karlsruhe Institute of Technology

Sample Data Set:

Number of Projections:	2000
Resolution of Projections:	1776x1707
Total Size:	24GB

Resulting 3D Image:

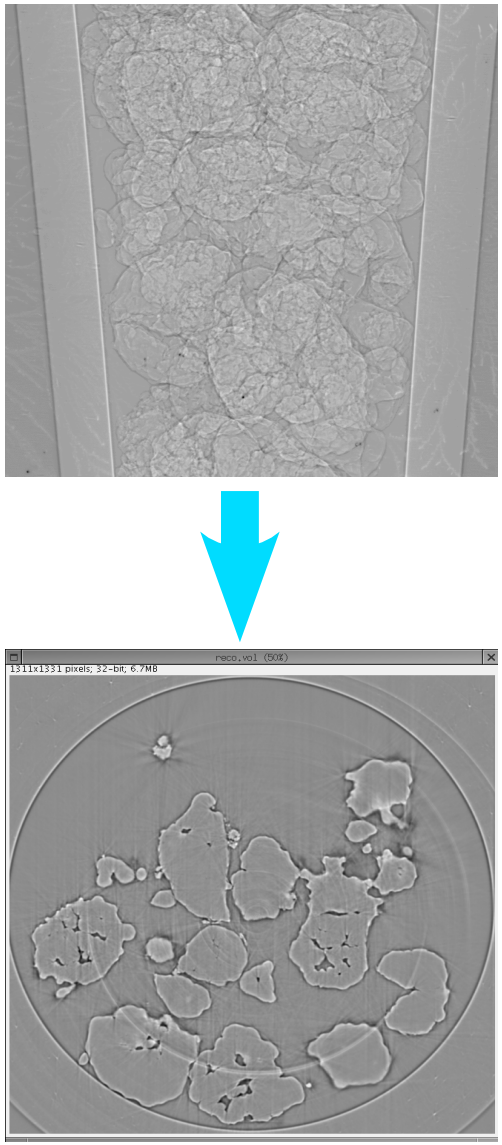
Resolution:	1691x1331x1311
Total Size:	11GB

Required Operations:

Data I/O:	35GB
Filtering:	~ 0.6 TFlop
Back Projection:	~ 53.0 TFlop

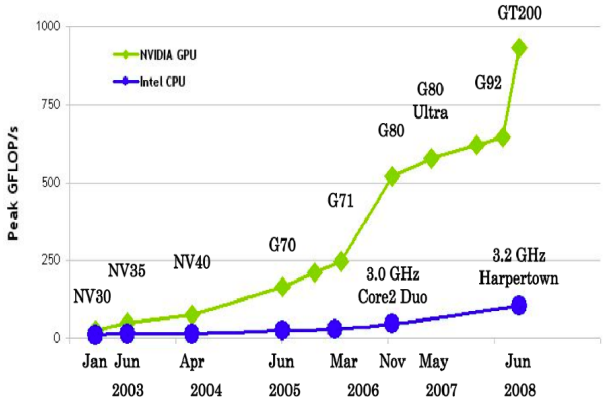
Maximum Performance Estimation:

Opteron 6176 (~ 110Gflops)	~ 10 minutes
HDD I/O (~ 50 MB/s)	~ 15 minutes



Solution: GPU Computing

- Highly parallel architecture: up to 480 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)



AMD Opeteron 6176 (1500$)

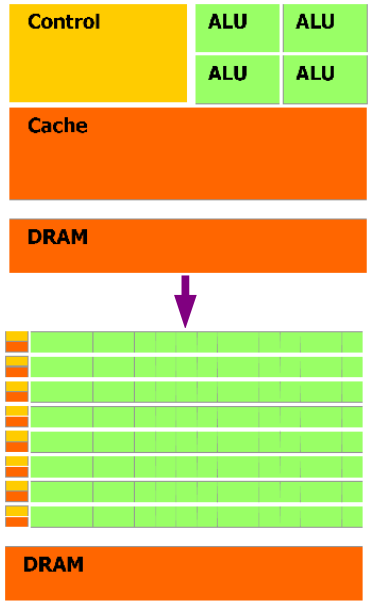
Number of cores:	12
Frequency of cores:	2.3 Ghz
Peak performance:	110.4/55.2 GFlop/s

NVIDIA GeForce GTX480 (500$)

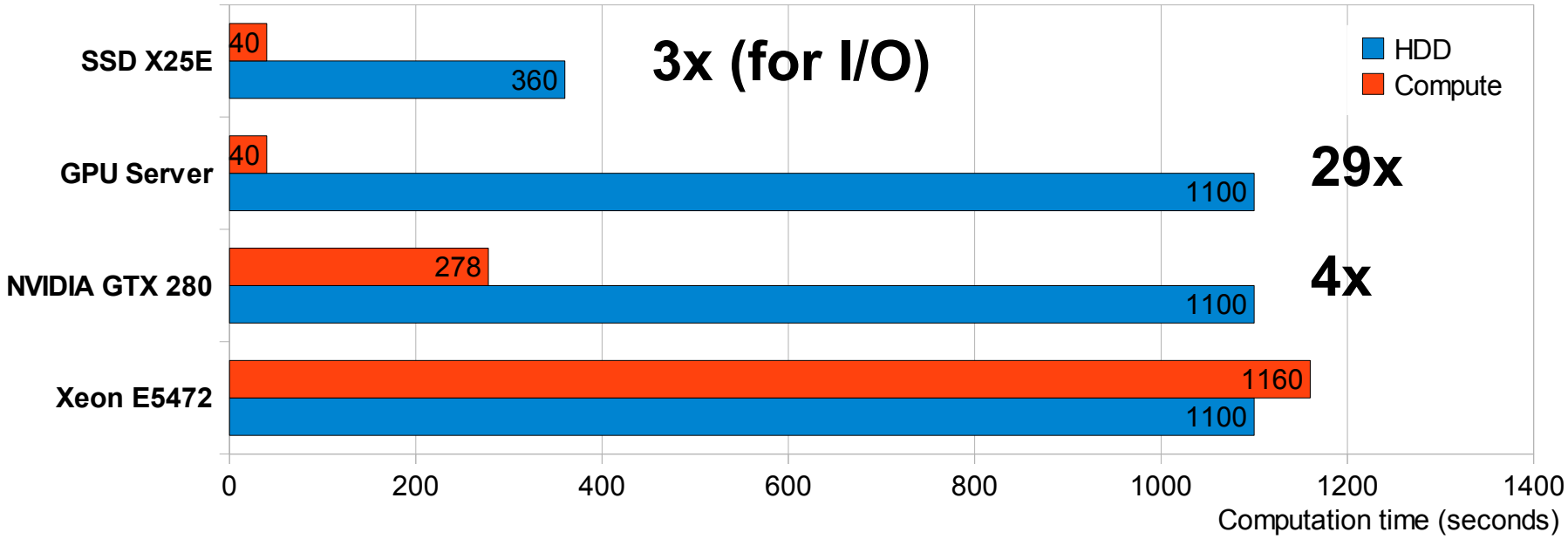
Parallel processors:	480
Frequency of cores:	1.3 GHz
Peak performance:	1345 / 168(672) GFlops

ATI Radeon HD 5970 (~700$)

Parallel processors:	2 x 1600
Frequency of cores:	725 MHz
Peak performance:	1640 / 928 GFlops



Test Setup and Results



	2 x Xeon E5472	GPU / CUDA	GPU Server / CUDA	Intel SSD X25E
Type of Computation	CPU / Xeon E5472 8 core, 3 GHz	**NVIDIA GTX 280 1 core**	**GTX295, GTX480 6 cores**	GPU / CUDA 240 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	WDC5000AACS	**Intel X25-E**
Price	5500$ (2000$ CPU)	1000$ (400$ GPU)	8000$ (2000$ GPU)	9000$ (800$ SSD)

Compiler Benchmark

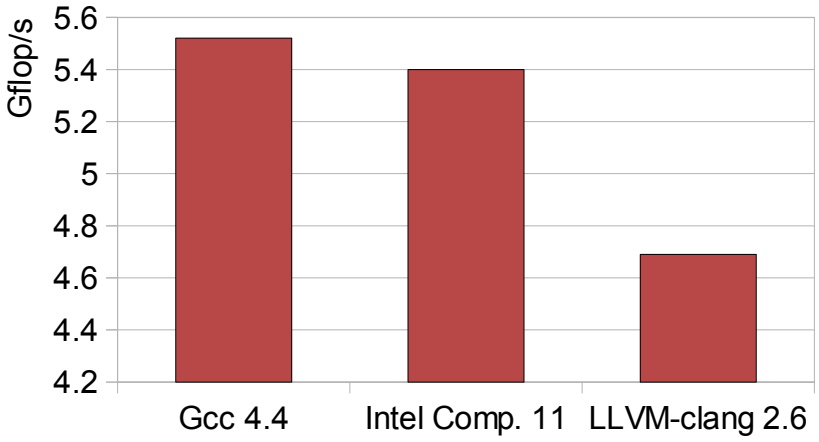
Karlsruhe Institute of Technology

Compiler flags

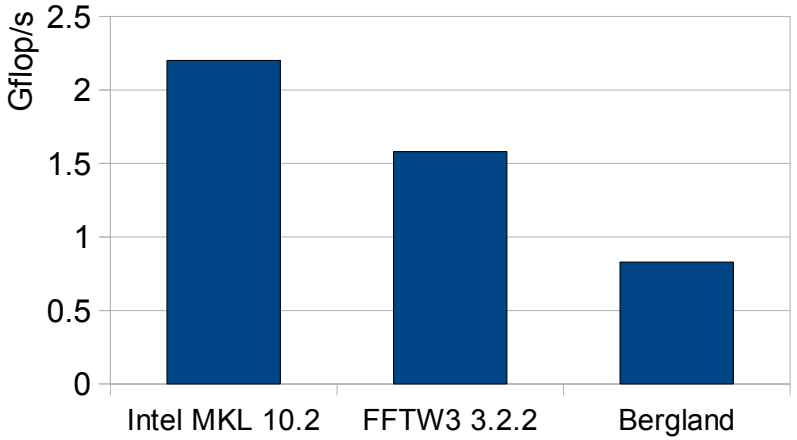
Intel: -O3 -xS (with SSE vectorization)

Gcc/LLVM: -O3 -march=nocona (with SSE vectorization)

Back Projection
(Compiler Benchmark)

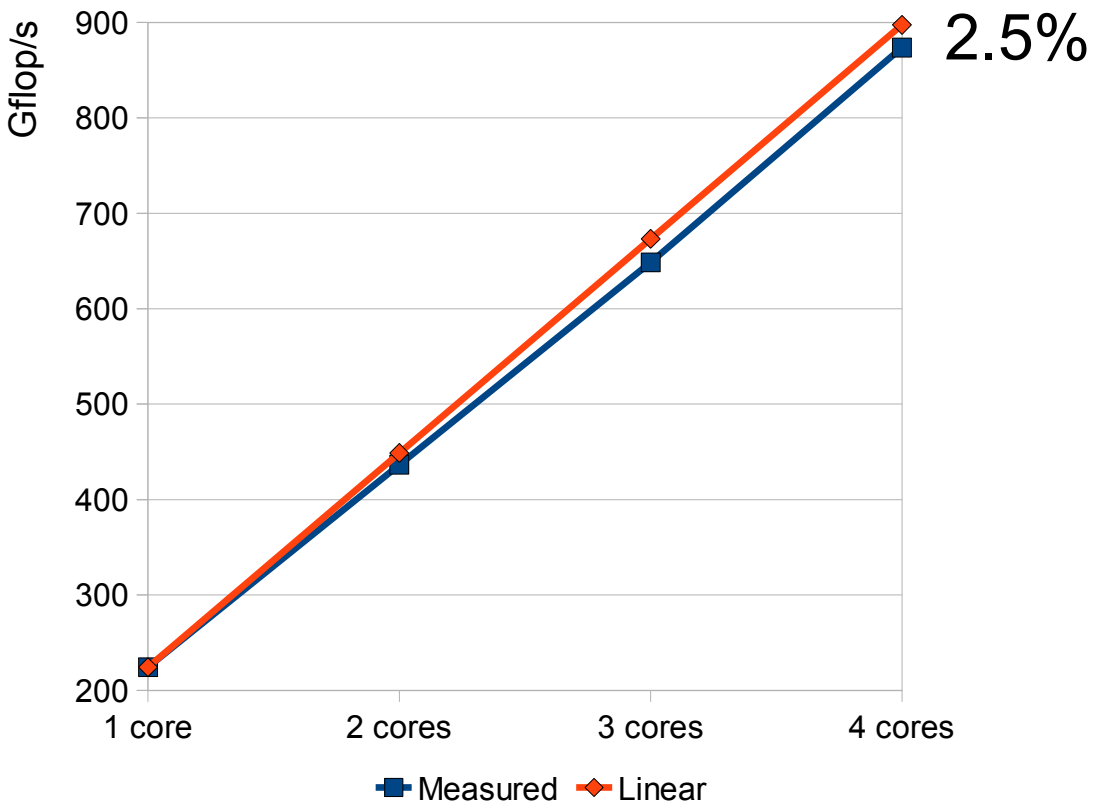


Filtering
(FFT Library Benchmark)



Decision: Using '**gcc**' and '**Intel MKL**' in all benchmarks

Scalability



Back Projection using Tesla S1070

Why NVIDIA and CUDA?

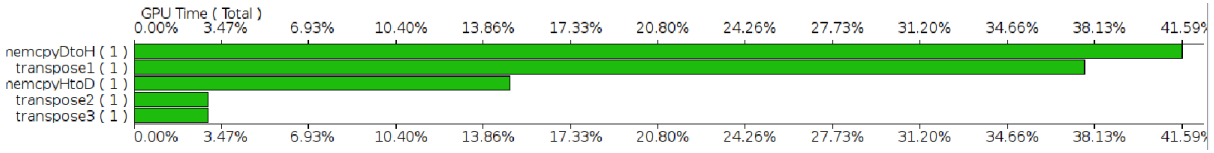
- Good Quality of Drivers
- Rich CUDA SDK:
  - cudpp (reduction), cuBLAS, cuFFT
  - cudaprof profiler
  - cuda-gdb debugger
- A lot of third party projects
  - OpenVIDA – Computer vision algorithms
  - Build Infrastructure for CMake: FindCUDA
  - Bindings for multiple languages: Python, Java, ...
- Easy migration to OpenCL
  - GPU code almost the same, initialization needs minor changes
  - OpenCL currently lacks library support

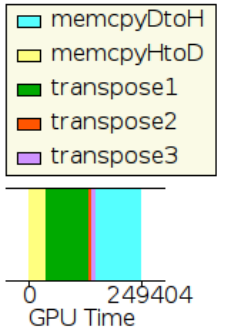
CUDA Debugger

```
Breakpoint 1, transpose3 () at /home/csa/cuda/1/test.cu:36
36          int dx = blockIdx.x * blockDim.x;
(cuda-gdb) thread
[Current Thread 2 (Thread 140710610523920 (LWP 16659))]
[Current CUDA Thread <<<(120,0),(0,0,0)>>>]
(cuda-gdb) p data[0]
$1 = 0
(cuda-gdb) next
[Current CUDA Thread <<<(120,0),(0,0,0)>>>]
transpose3 () at /home/csa/cuda/1/test.cu:37
37          int dy = blockIdx.y * blockDim.y;
(cuda-gdb) print dx
$2 = 1920
(cuda-gdb)
```

CUDA Profiler

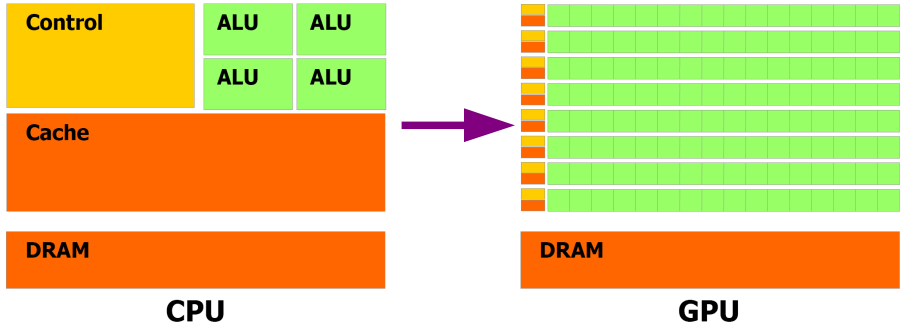
	Method	GPU Time	CPU Time	Occupancy	warp serialize	cta launched	gld 64b	gst 32b	gst 64b
1	memcpyHtoD	36433.9	37004						
2	transpose1	92264.5	93350	1	0	6554	104864	1677824	5160604
3	transpose2	7130.69	8103	1	597499	6554	104864	0	1343814
4	transpose3	7080.64	8207	1	0	6554	104864	0	1319868
5	memcpyDtoH	101772	104837						





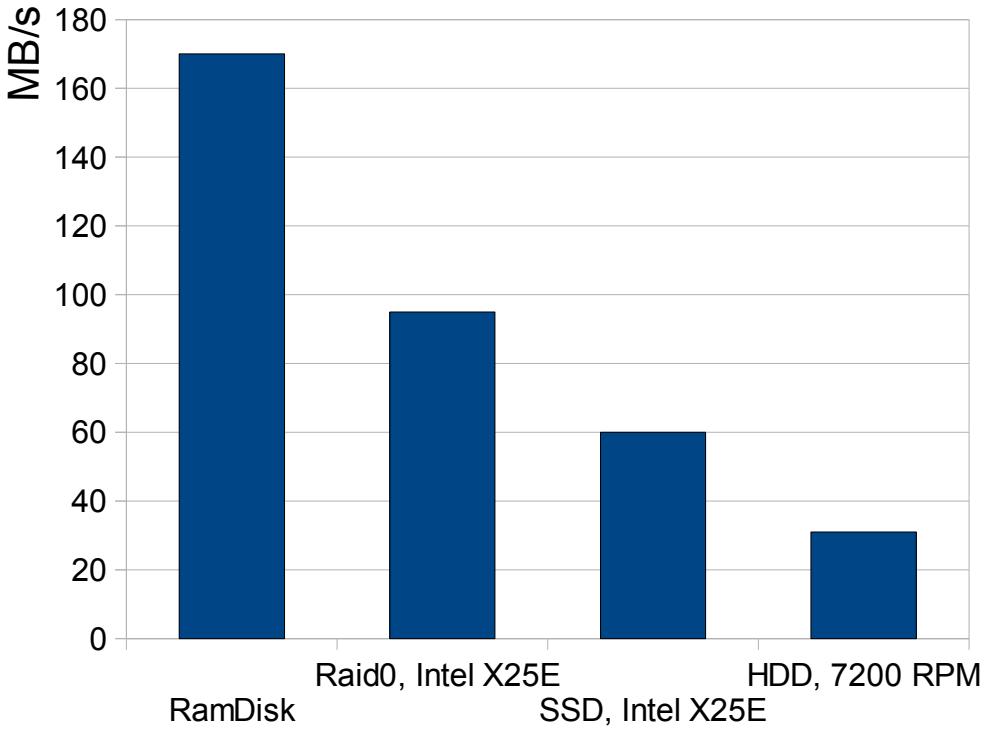
What do we do to run faster?

- **Memory (GPU lacks cache!)**
  - Allocate all device memory during initialization
  - Pad data to multiples of block dimensions
  - Optimally access memory (**increase of speed up to 10 times**)

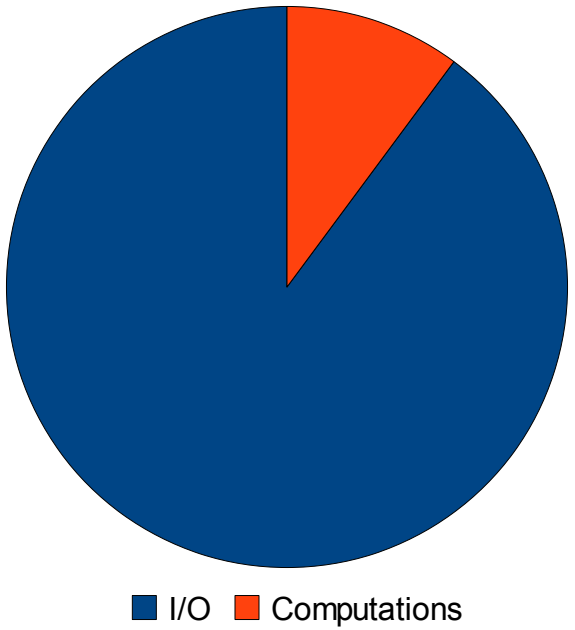


- **Data Transfer (Slow interface between GPU and memory ~ 2 GB/s)**
  - Reduce amount of data transfers between GPU and host (**extreme**)
  - Use pinned memory for transfers if possible (**up to 2 times**)
  - Interleave data transfers with computations
- **Back Projection (GPU is optimized only for a few specific instructions)**
  - Avoid integer arithmetics (it is only fast on Fermi)
  - MADD – two operation at cost of one
  - Use texture engine for interpolation
- **Filtering (cuFFT fast for 2^n cases and always do complex transforms)**
  - Pad data to a size equal to the closest power of 2
  - Use batched calls
  - Compute two real convolutions using a single complex

Handling I/O Problem



Ratio between
I/O and Computations

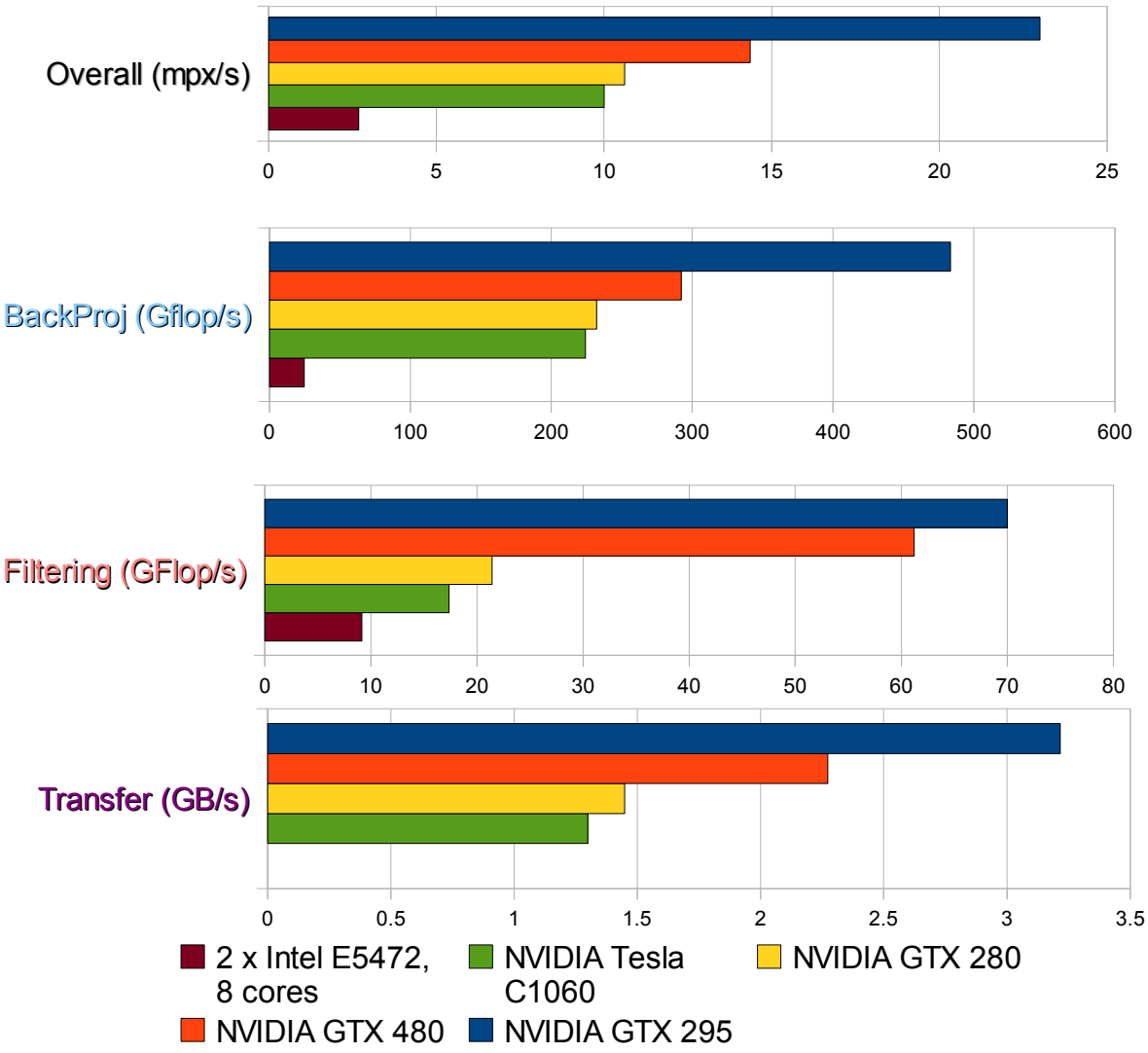


Next steps:

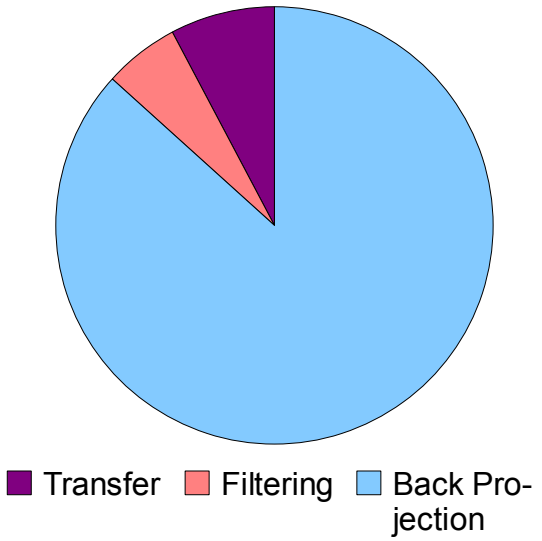
- PCIe x16 card with SSD raid (up 1TB/s)
- Read from frame-grabber directly into the memory and process
- FPGA based frame reject and compression of image data

OCZ Z-Drive
Size: 1 TB, MLC
Read: 870MB/s
Write: 780MB/s

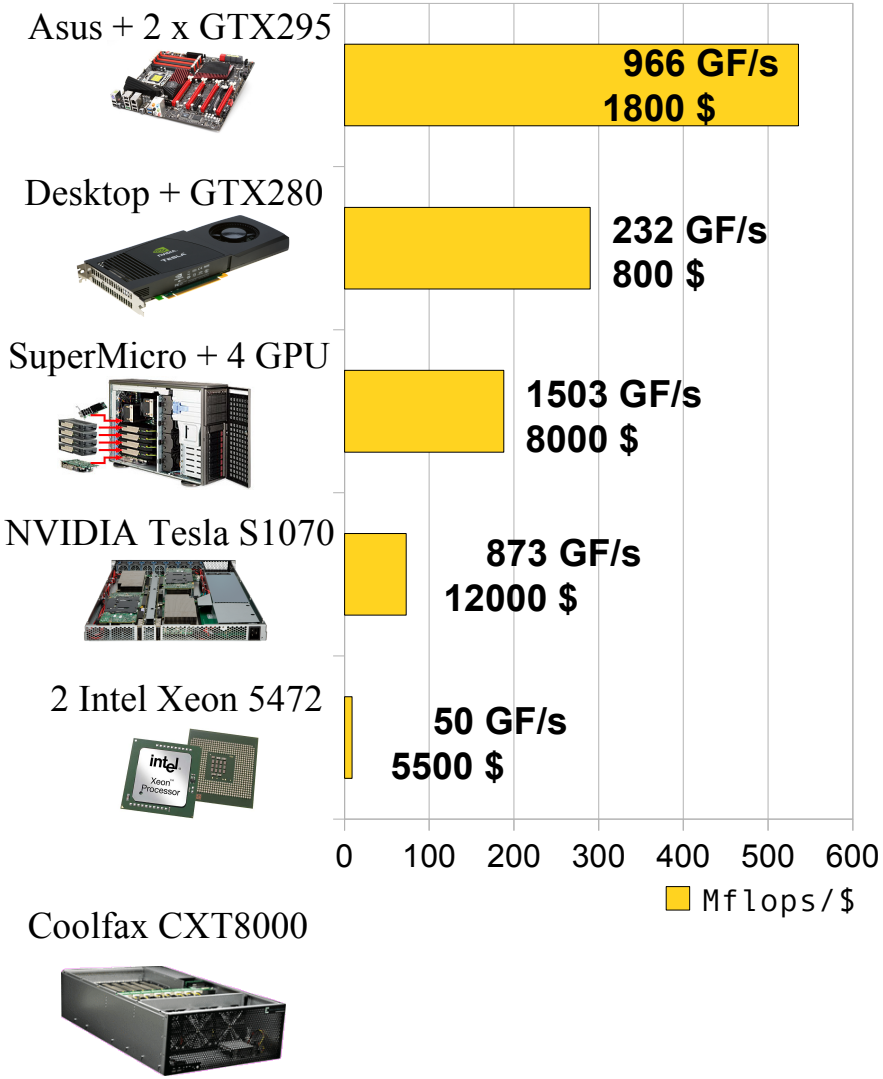
Benchmark Of NVIDIA Hardware



Ratio of Operations



GPU Servers



Asus Rampage III Extreme (1800$ = Core i5 + 2 x GTX295 + 8 GB)
Chipset: x58, 36 PCIe 2.0 lanes; 6 DDR3 slots (48 GB max)
PCIe 2.0 x16: 4 (x8 if all 4 are used)
Max Peak Performance (ATI): 18.56 Tflops / 3.7 Tflops
Max Peak Performance (NVidia): 7.15 Tflops / 595 GFlops

Standard Desktop (800$ = Core2 + GTX285 + 2 GB)

SuperMicro 7046GT-TRF (~8000$ = 2 Xeon + 4 GPU + 96 GB)
Chipset: Dual Intel 5520, 72 PCIe 2.0 lanes, 12 DDR3 slots (192GB max)
PCIe 2.0 x16: 4 (full speed), x4: 2 (in x16 slots); PCIe 1.0 x4: (in x8 slot)
Max Peak Performance (ATI): 18.56 TFlops / 3.7 Tflops
Max Peak Performance (NVidia): 7.15 TFlops / 2.5 Tflops

NVIDIA Tesla S1070 (~8000$ + 4000$ host)
System: Requires separate host server
GPU Devices: 4 x Tesla C1060 (960 parallel processors at 1.44 GHz)
Peak performance: 4.14 Tflops / 345 Gflops
GPU Memory Size: 16 GB

Dual Xeon 5472 Server (5500$ = 16 GB)
Max Peak Performance: 96 Gflops / 48 GFlops

Coolfax CXT8000 (36000$ = 2 Xeon + 8 Tesla C2050 + 144GB)
Chipset: Dual Intel 5520, 72 PCIe 2.0 lanes
PCIe Switch: PLX PEX8647; PCIe 2.0 x16: 8 (full speed)
18 DDR3 Memory Slots: 288 GB max
Max Peak Performance: 10 Tflops / 5 Tflops
Back Projection Performance: 2336 Gflops (Estimated)

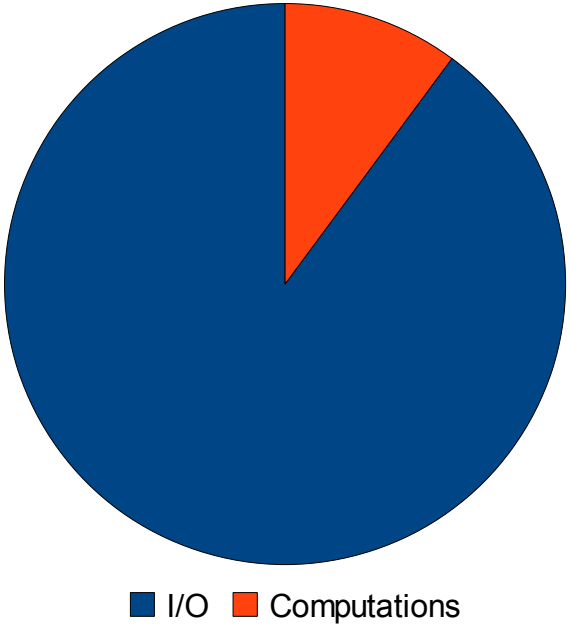
Conclusion

Goal: Process up to 30GB of image data in a minute

I/O

6 minutes

We will try disk-less in-memory processing to bring time under one minute.



Computations

40 seconds

Down from 30 minutes to less than a minute. The computation problem is solved.

Data Transfers is Dominating