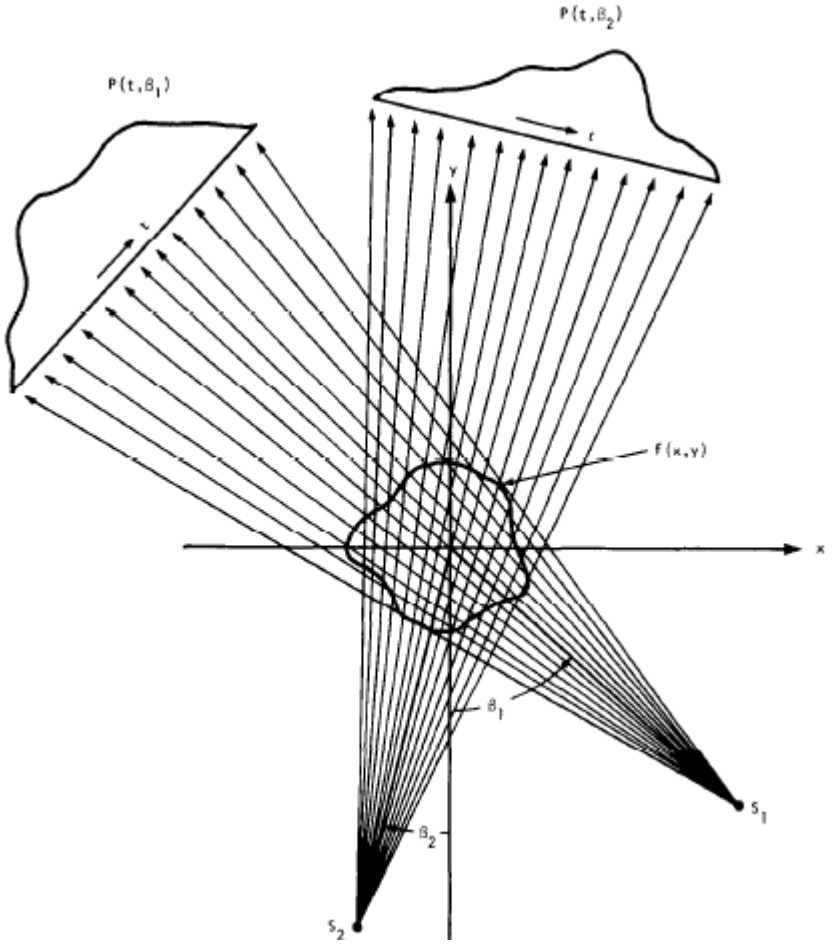
Accelerated image reconstruction on a cluster of two AMD GPUs in CBCT with non-uniform detector geometry

Alain Bonissent
Centre de Physique des Particules de Marseille

X-ray computed tomography

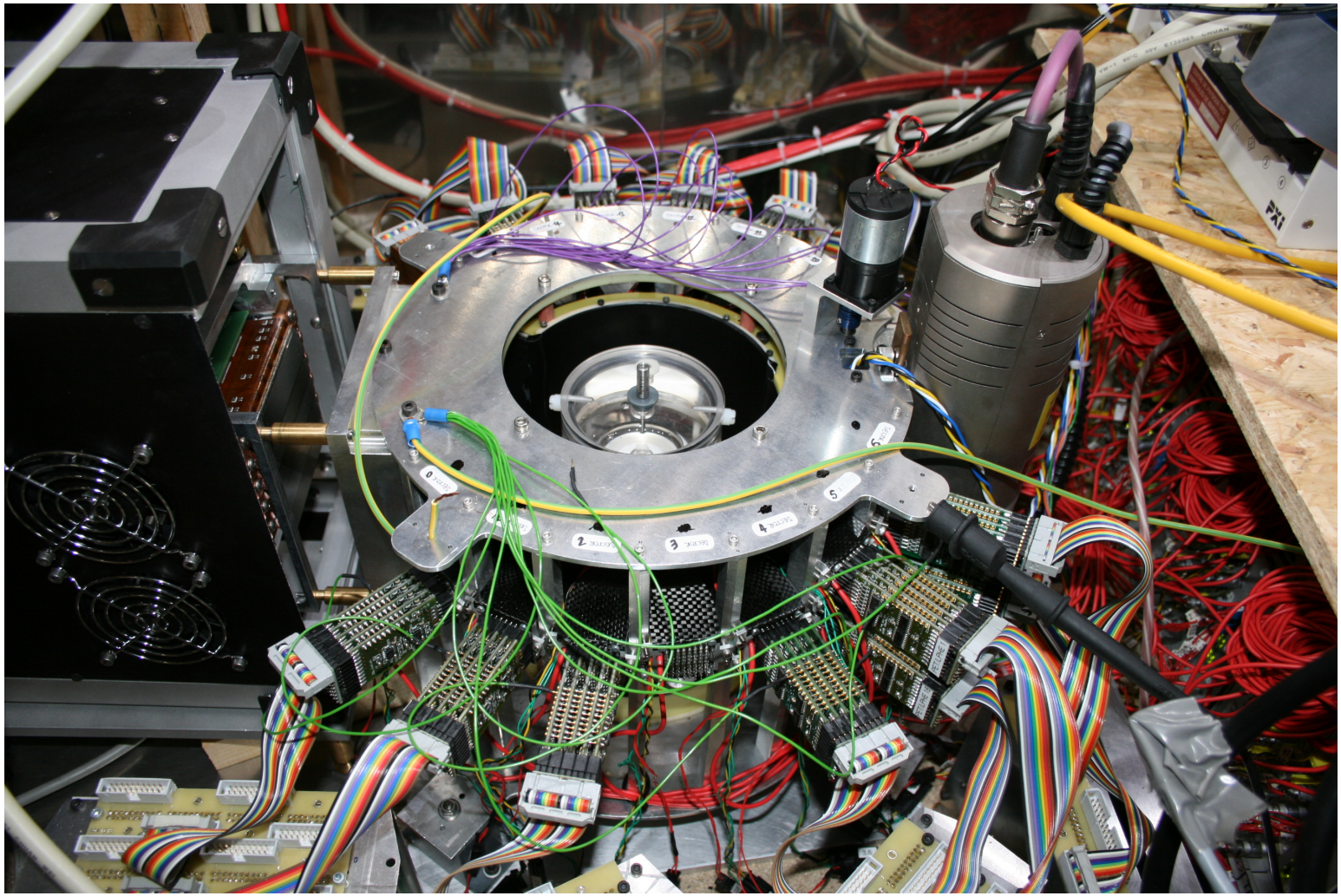


Same object seen under many different angles provides different projections.

Combining all the projections => 3D object

From Kak-Slaney :
Principles of Computerized Tomographic Imaging

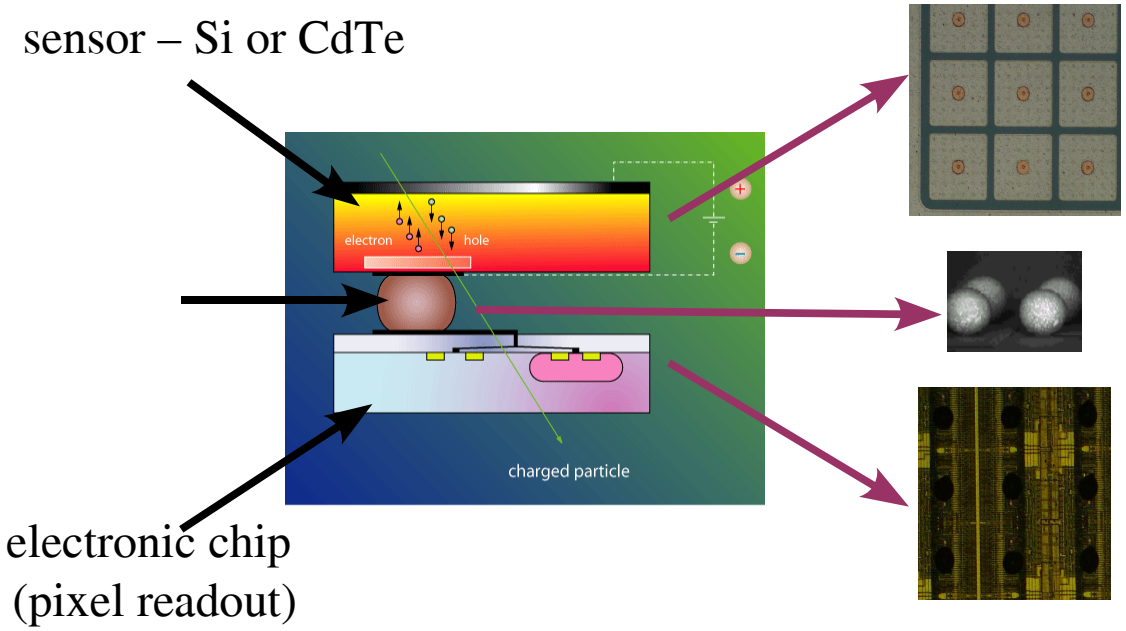
The small animal CT scanner



Xpad3

Xpad3 is the latest X ray detector developed at CPPM.

Hybrid : each pixel has its own analog and digital electronics



Pixel size : 130 x 130 µm
Total : 560x960 pixels

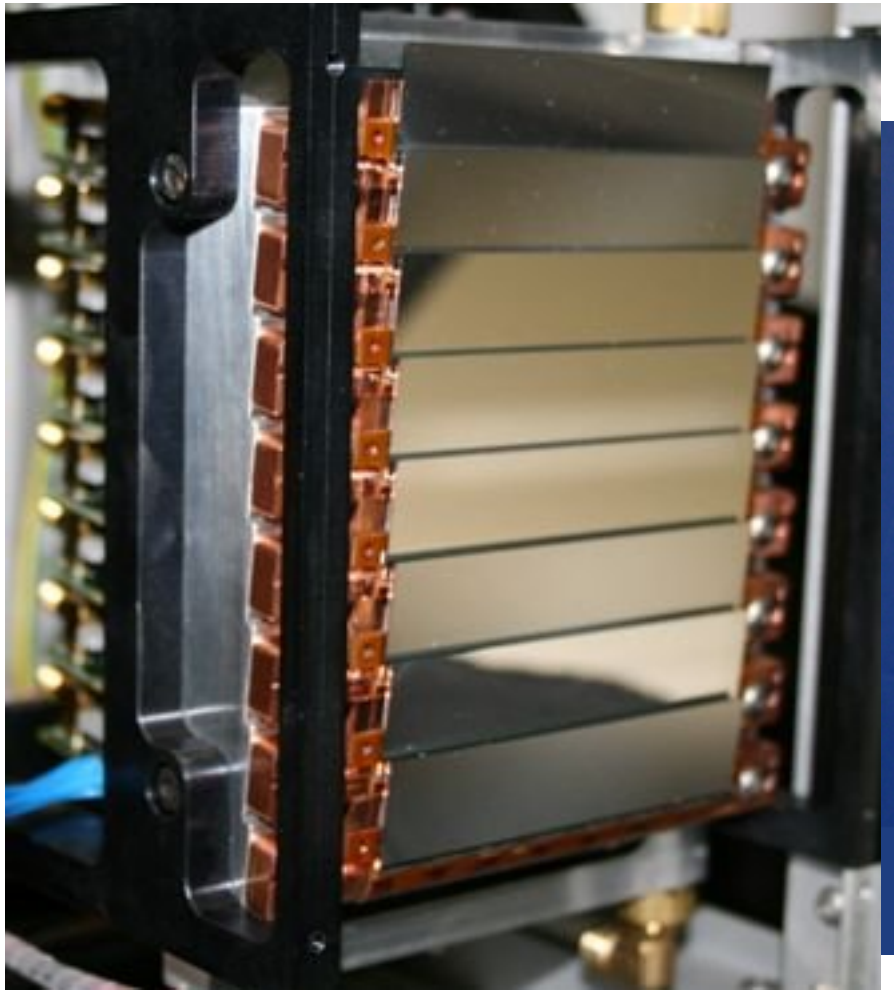
Fast readout and data transfer up to 1000 frames/s
(optical link and PciExpress)

XPAD specific complications

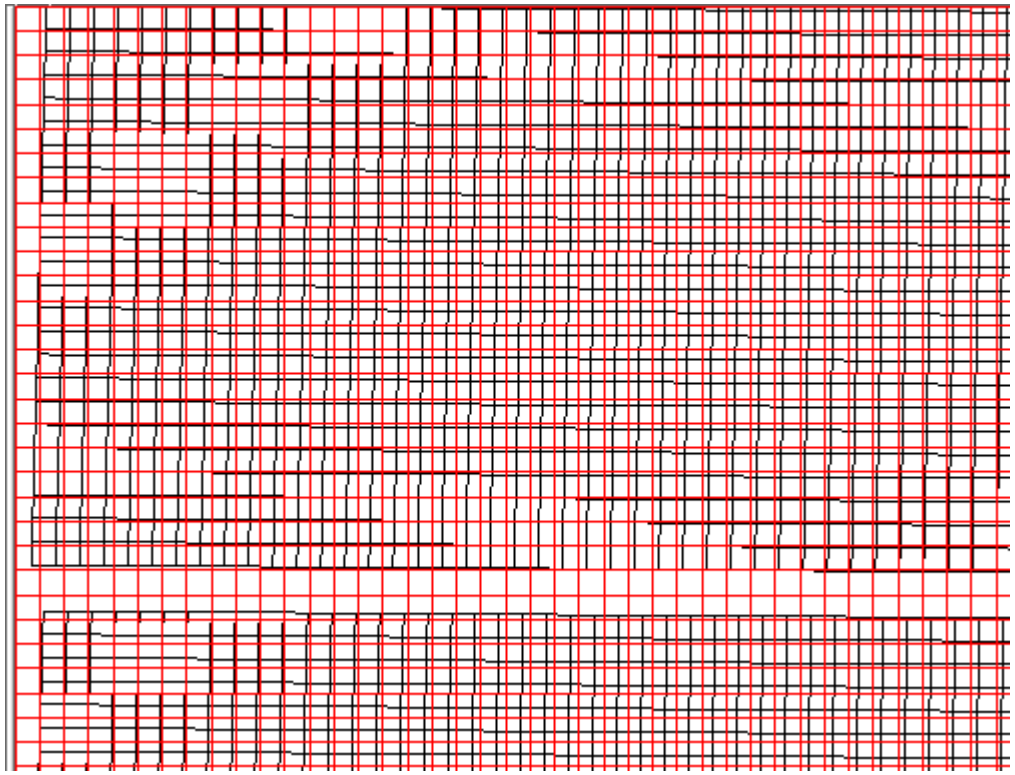
Made of chips 1x1cm
assembled in barettes,
barettes assembled in tiles

Problems :

effective pixels are not square;
shape and size vary with position;
dead regions between barettes;



XPAD tiles



Upper left corner Xpad 2
Ideal grid

Effective grid :

pixels are projected on mean plane

FDK

Feldkamp, Davis, Kress :
analytic tomographic reconstruction for cone-beam geometry

For each image :

- Cone beam correction, solid angle seen by pixels depends on their position, count => Log(count)
- Filtering : enhance high frequency components. Convolve with Fourier transform of ramp filter.

- Backproject :
  Fdivide the volume into small cubes (voxels);
  For each voxel, project to the detector plane, find the attenuation at this point, accumulate in the voxel
  This is the cpu intensive part : bilinear interpolation

Possible solutions to non-uniform geometry

- The easy and obvious :
  Rebin the images to a regular grid
  Easy to implement : bilinear interpolation once the location of each pixel is known;
  Standard FDK software can then be used.
  Fast if rebinning inside FDK loop, otherwise disk I/O
  Can affect the resolution because signal is not liear.

Alternate and better :
Distort the detector plane so that pixels positions are uniformly distributed.
The (small) distortion field is stored in a grid with 2x the resolution. Positions are linearly interpolated into pixel coordinates. Then pixel count is interpolated only once.

600x900 pixels and 360 projections, 600x600x900 voxels reconstruction would need several hours on conventional CPU

The AMD 9270 GPU

2GB on board memory DDR5
1.2 Tflops single precision
240 Gflops double precision

800 stream cores
PCIExpress x16

Programming the GPU

Coding for the AMD GPU : Brook+

- Kernels execute on the GPU.
- Brook+ is the programming language for kernels
  Evolution of Brook (Stanford University)
  C with a few extensions :
    transfer data between CPU and GPU
    access stream indices, (can be multidimensional)
- Stream : a set of data elements on which the kernel is applied independently : can run in parallel
  Pixels in an image
  Voxels
  Image calibration constants

brook+ is processed by pre-compiler -> C++

FDK in Brook

Most operations can run in parallel

- Image preprocessing : pixel level
- Convolution : Use a trivial algorithm with nested loops not fully efficient but fast anyway

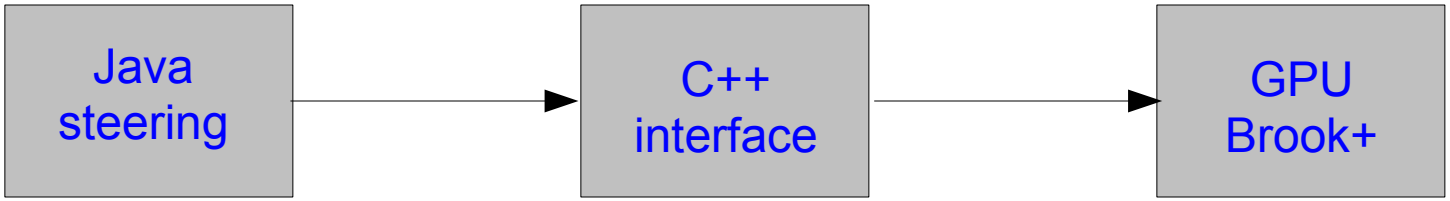
- Backpropagation : voxel driven ; reconstructed volume is 1 stream. Random access to image pixels (DDR5)

- Load images : disk->cpu->gpu

- volume in GPU during all processing, to disk at the end

- Problems : volume in double precision is more than GPU memory (2 GB) : Process in 2 halves; streams limited in size, need to divide volume in slices

Program architecture



Prepare lookup tables for geometry corrections and interpolation

neighbours and weights

Find dead pixels (flatfield and dark)

control execution

Initialize GPU device;
Load tables

Disk IO :
load images
write reconstructed volume

Control execution of kernels in GPU

Intensive operations

Image calibration; interpolation of bad pixels;

FDK backpropagation with double interpolation

Performances Fdk on Xpad3 data

Total for a volume 560 x 560 x 960, 360 projections

200 s end to end, wall clock time

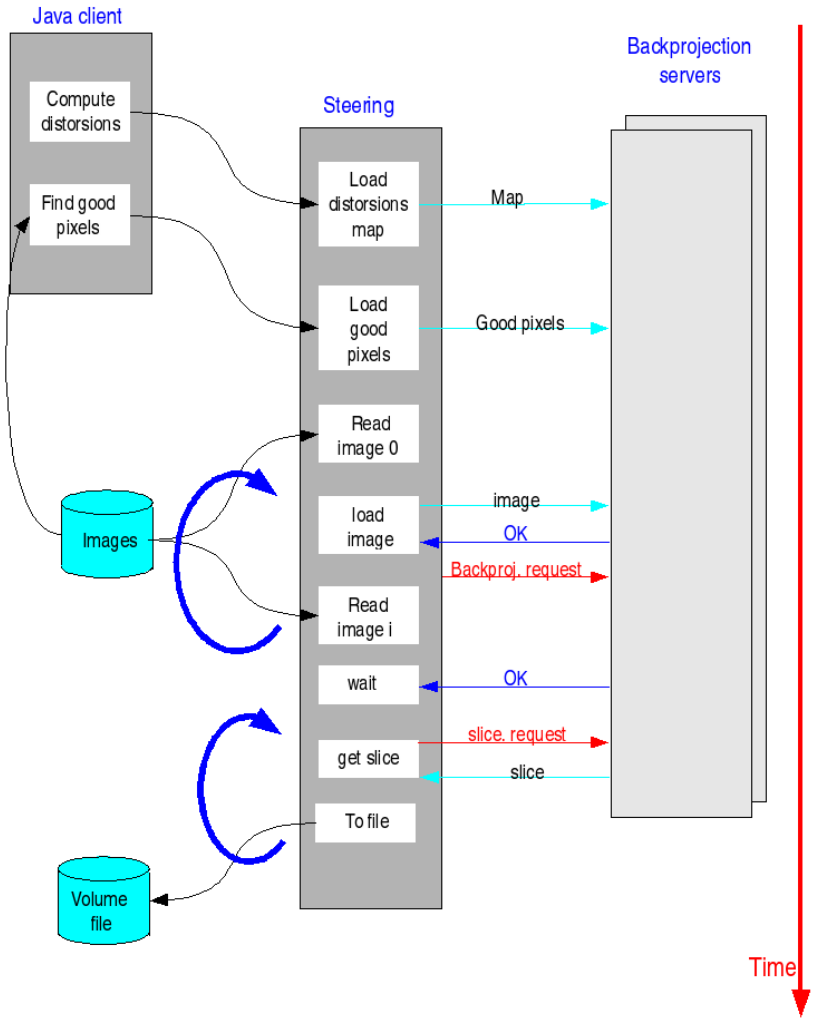
Includes processing and disk I/O both ways

Expected for CPU version : several hours.

Can do better :

run GPU processing and I/O in parallel;
use 2 GPUs : no need to read images twice

MultiGPU



Two PCI Express ports on motherboard
Can accommodate a second 9270

Client-server architecture
One server per GPU

Communication :

sockets for synchronization
shared memory for images
and volume slices

Brook+ has multiGpu capabilities,
but I was unable to make it work

multiprocess and sockets
not so elegant but robust

Performance 2 GPU

Total end to end 133 s, improved by x110 w.r. to full cpu

But Initialization 15s, transfer cpu->gpu 35s, writing volume on disk : 30s
total 80s CPU+I/O irreducible

Effective processing 50s.

Speedup 200 - 300 with 2 GPUs, 100 - 150 with a single GPU

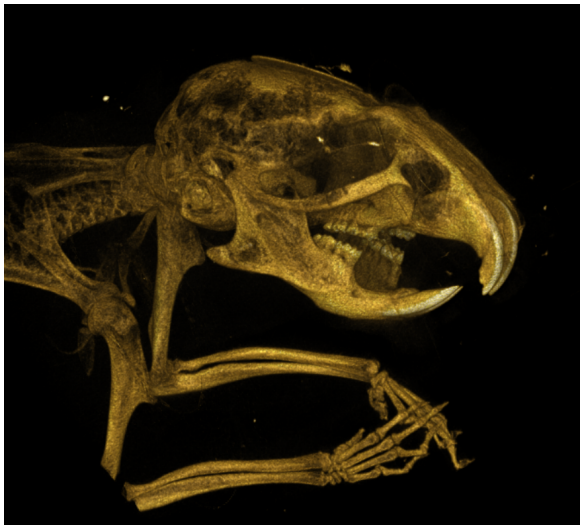
Conclusion

Huge gain in performance for limited effort
(parallel processing and fast random memory access)

FDK is simple, stream computing well suited
Coding is easy, debugging can be tricky

MultiGpu : added complexity, more problems

It does work !



CT scan at CPPM
S. Nicol, S. Karkar, C. Hemmer, D. Benoit