

Real-time computer

Bob Sault

Compute functions

- Fringe rotation / “delay tracking”
- Real time calibration and imaging
- Real time science analysis

Hardware architecture

- Cluster computing environment decomposed by frequency
 - “Natural” decomposition
 - Fits well with the output of the correlator
 - Algorithms chosen for small intercommunication
 - Minimises moving the data around
- “Generic” hardware architecture approach
- Multi-threading by source and other “parallelisations” will be exploited as needed.
- Pipelining allows trade-off between load balancing and memory

Computational requirement

- Computational requirement $\sim 0.8\text{-}2$ Tflop/s
- Peak computing requirement ~ 5 Tflop/s
- 256 nodes, 1024 cores
- 256 GigE connections to correlator

Hardware acceleration

- Fringe tracking / “delay tracking” step in FPGA??
- GPUs

IIC testbed

- IIC cluster: 54 nodes (4 cores per node; 2 or 16 Gbytes per node)
- GPU research should be concluded within ~2 months

GPU Scientific Computing

Why graphics processors (GPUs)?

- Compute power (floating point operations s^{-1} ; FLOPS)
- Cost (FLOPS/\$)
- Power efficiency (FLOPS/watt)
- Accessibility via high-level programming interfaces

GPU Application to MWA

Why GPUs for the MWA RTS?

- *data-parallel* application w/ *high arithmetic intensity*

GPU numbers:

- CPU→GPU speedup for initial benchmarks:
 - 2D FFT (1024^2): **12x**
 - Gridding (1024^2 , 8x8 kernel): **114x**
- NVIDIA Quadro FX 5600:
 - 346 GFLOPS (max, theoretical); 4 GB s⁻¹ host-device bandwidth (max)

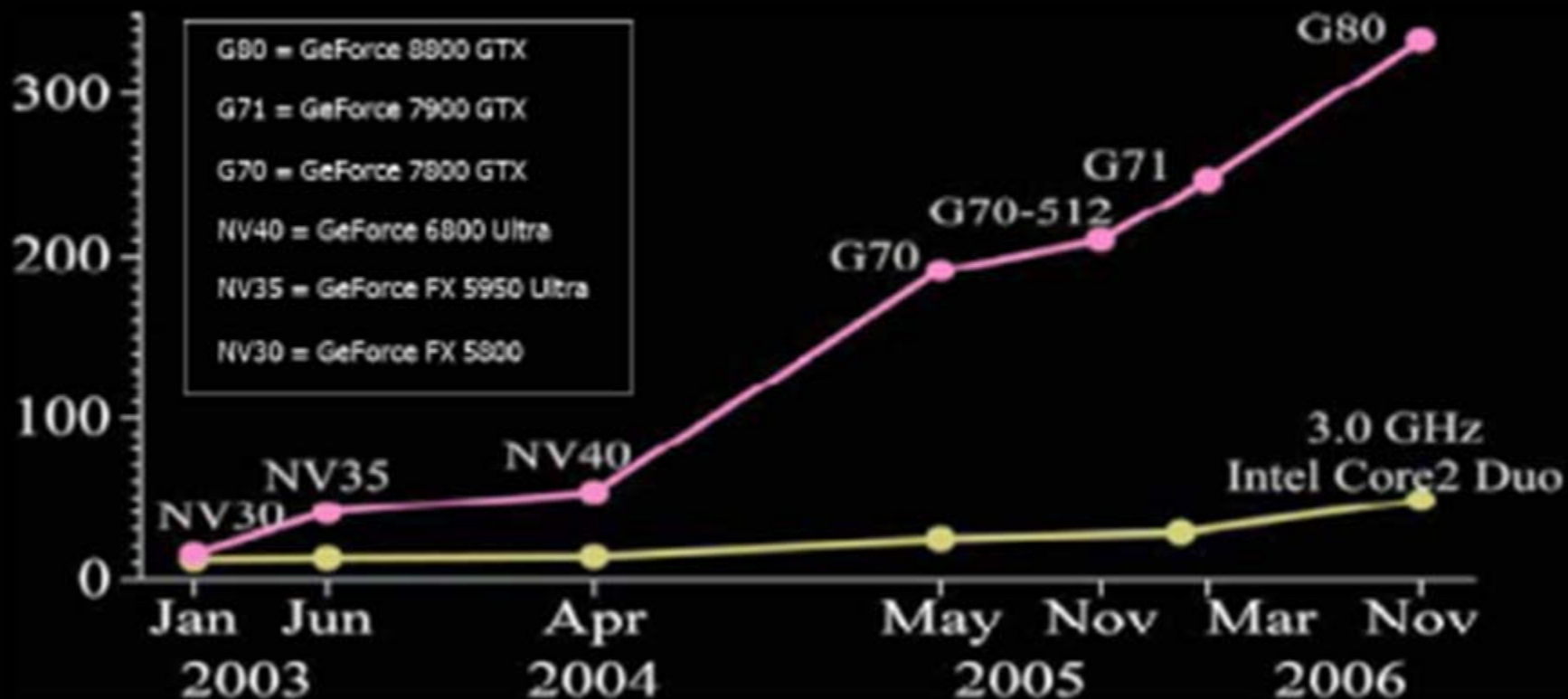
RTS Numbers:

- 1.7 GB/s core bandwidth from visibility integrator
- O(100) or fewer pipeline threads anticipated
- a few TFLOPS total (conservative estimate)

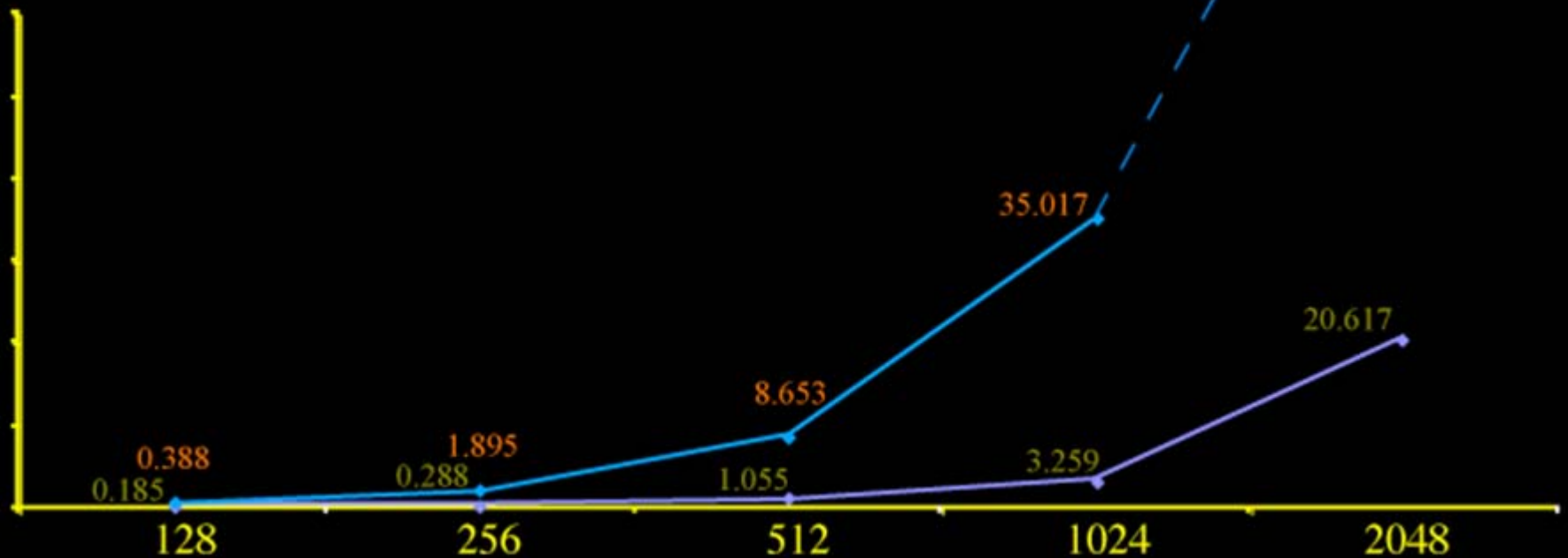
Can a GPU service an entire thread, end-to-end?

Impact of I/O blocking?

GFLOPS



2D FFT: Execution time vs. image size (powers-of-2) (Non power-of-2 transforms will be somewhat slower)



Initial Benchmarks: Platform

- Hardware
 - 2.4 GHz dual-core AMD Opteron, 4GB RAM
 - NVIDIA Quadro FX 5600
- Software
 - AMD Core Math Library (ACML), OpenMP version
 - NVIDIA CUDA
 - OpenGL