A Comparison of Threaded CPU and GPU implementations of interferometry operations

Bill Cotton, NRAO

- DFT based model calculation
- Griding visibilities
- Example using Obit

DFT based model calculation

- Direct transform of sky model to UV

$$V_{i,\nu} = \sum_{k=0}^{n} a_k \; e^{-2i\pi \frac{\nu}{\nu_0}(u_{j0}x_k + v_{j0}y_k + w_{j0}z_k)}$$

- Easily adapted to direction dependent effects
- Very compute intensive
- Mostly evaluation of sine and cosine

ftp://ftp.cv.nrao.edu/NRAO-staff/bcotton/Obit/GPUDFTv2.pdf

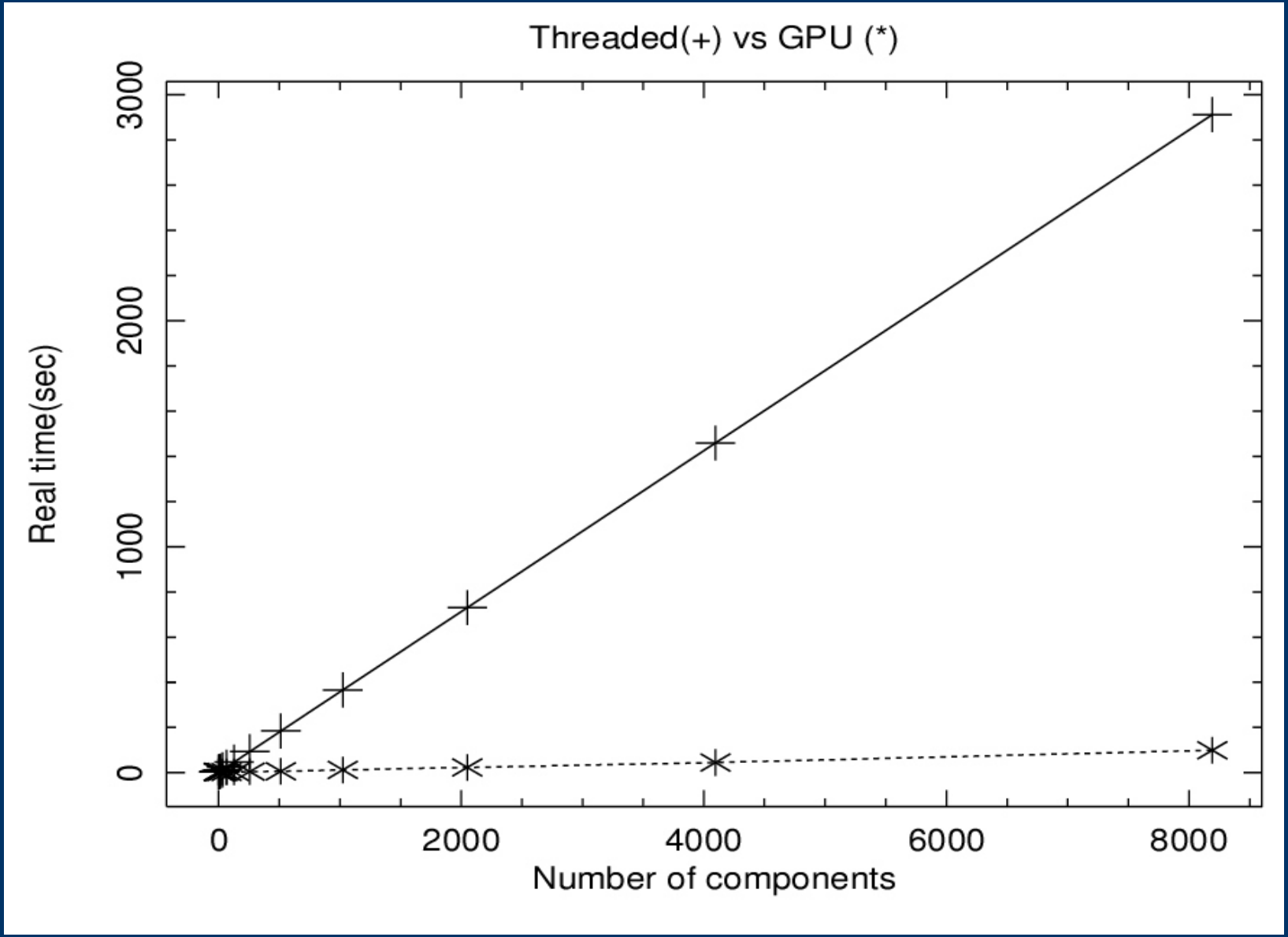
GPU DFT implementation

- Used nVidia GPUs programmed in CUDA
  - GeForce GTX 780, 2304 cores
  - Tesla K20c, 2496 cores
  - GTX 780 (cheaper) faster and used in subsequent tests
- Used CUDA intrinsic __sincosf for sine/cosine
- Test program used:
  - 100 sets of 10,000 random visibilities, 512 channels
  - range of number of point components in sky model
  - multiple streams to overlap data transfer, computation
- Measured wall clock execution times

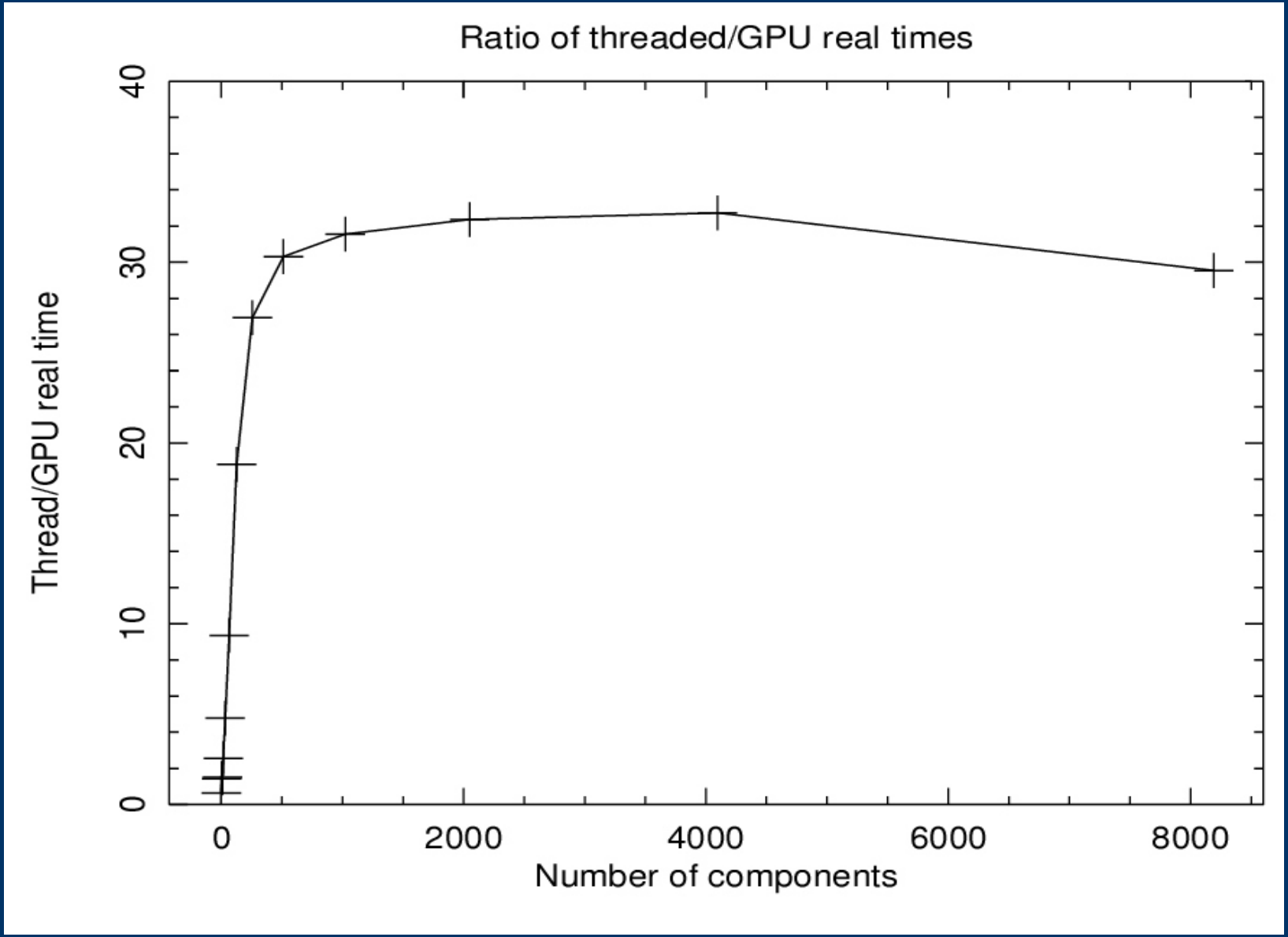
DFT multi-threaded implementation

- Used AVX based table lookup + Taylor series fast sine/cosine routine.
- Used gthreads thread pools for threading
- Same test case and computer as for GPUs
- Machine had 6 cores hyperthreaded to 12

DFT GPU v. multi-threading



DFT GPU v. multi-threading, cont'd



DFT GPU v threading, cont'd

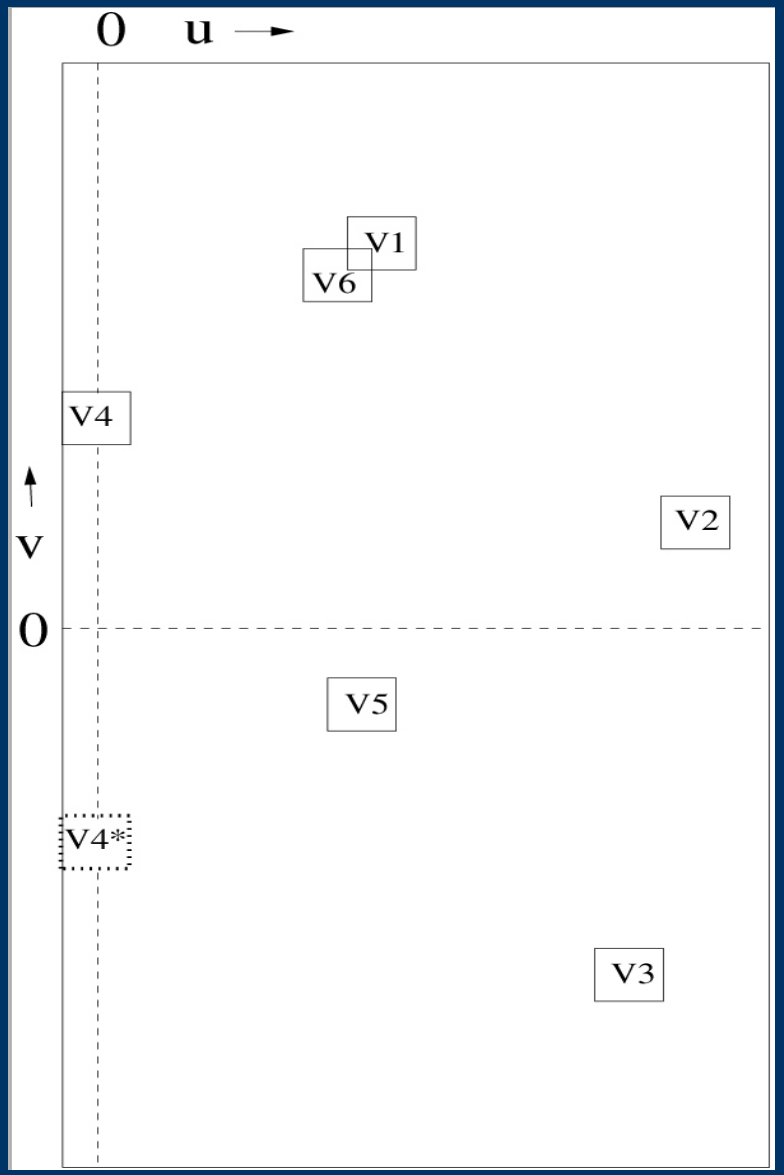
- GPU implementation 30 X faster than 6 core threading.
- Very large amount of computing per input data point.
- Data access very good match to GPU and CPU.
- No dependencies, each calculation independent

Griding visibilities

- Visibilities randomly sampled on u-v plane
- Use convolutional griding to allow use of FFT
- Faceted imaging for “w” problem is very parallel
  - same data used for each facet
  - compact, separable convolution kernal (7 x 7)
- Dependencies in overlapping u-v kernal.

ftp://ftp.cv.nrao.edu/NRAO-staff/bcotton/Obit/GPUGrid.pdf

Griding visibilities, cont'd



u-v half plane showing overlapping contributions of visibilities.

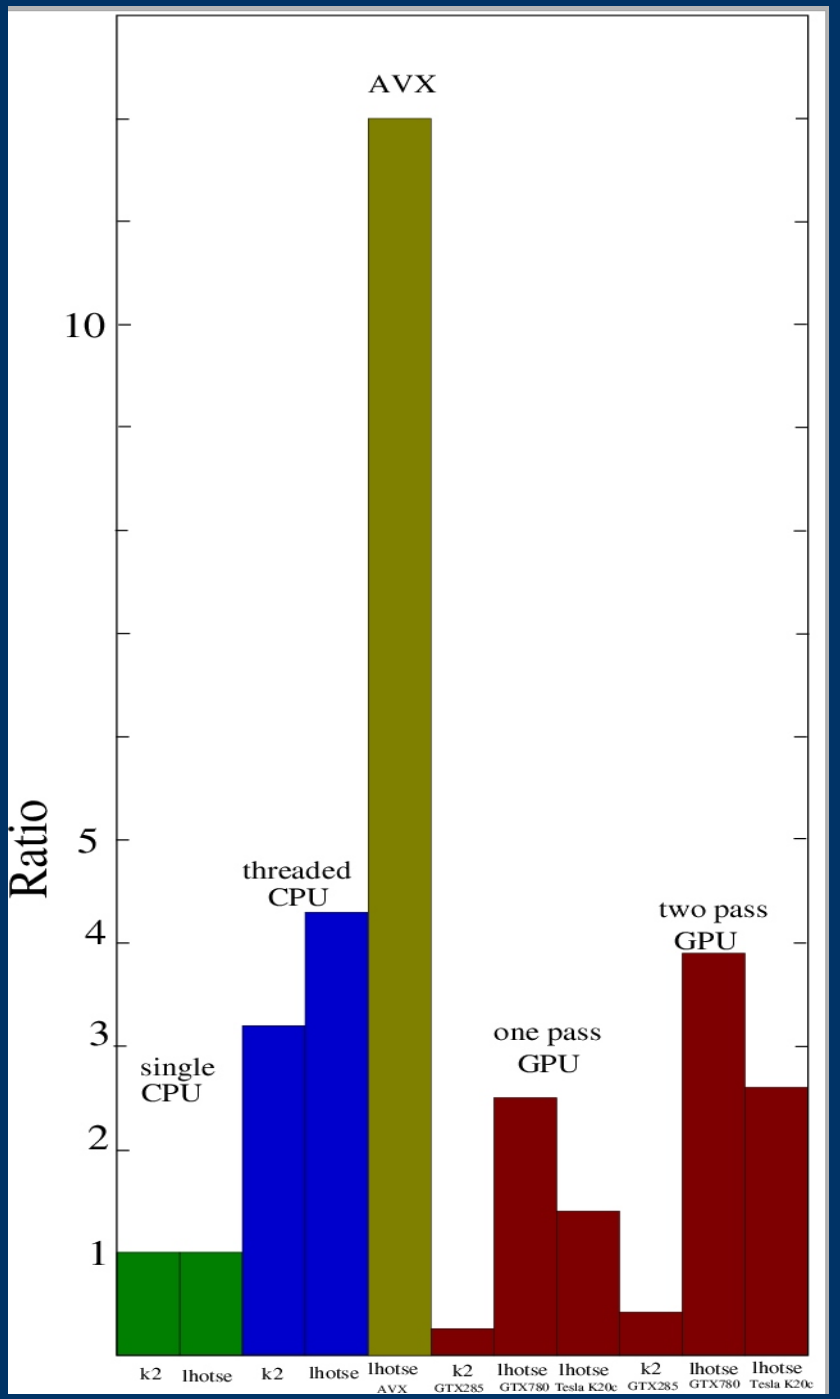
Griding, GPU implementation

- Many cores, must use atomic adds to update grid.
- Two passes through each set of visibilities.
  1) rotate data for facet, calculate grid locations
  2) convolve data, accumulate in grids
- Accumulate used atomic adds
- Tested on GTX 285, GTX 780 and Tesla 20c
- Test data
  - used EVLA u-v coverage
  - 100 x 10,240 visibilities, 1024 channels
  - 7 2048x2048 facets

Griding, threaded implementation

- One grid per thread per facet
- Two pass implementation as for GPU
- Tested with and without AVX vector enhancement

Griding comparison



Griding comparison

- GTX 285 much slower than single CPU core
- Multi-threading using AVX much faster than GPU
- Without AVX, multi-threading similar to GPU
- Reasons for poor GPU performance:
  - quasi-random grid access
  - atomic adds are expensive

GPU error correction

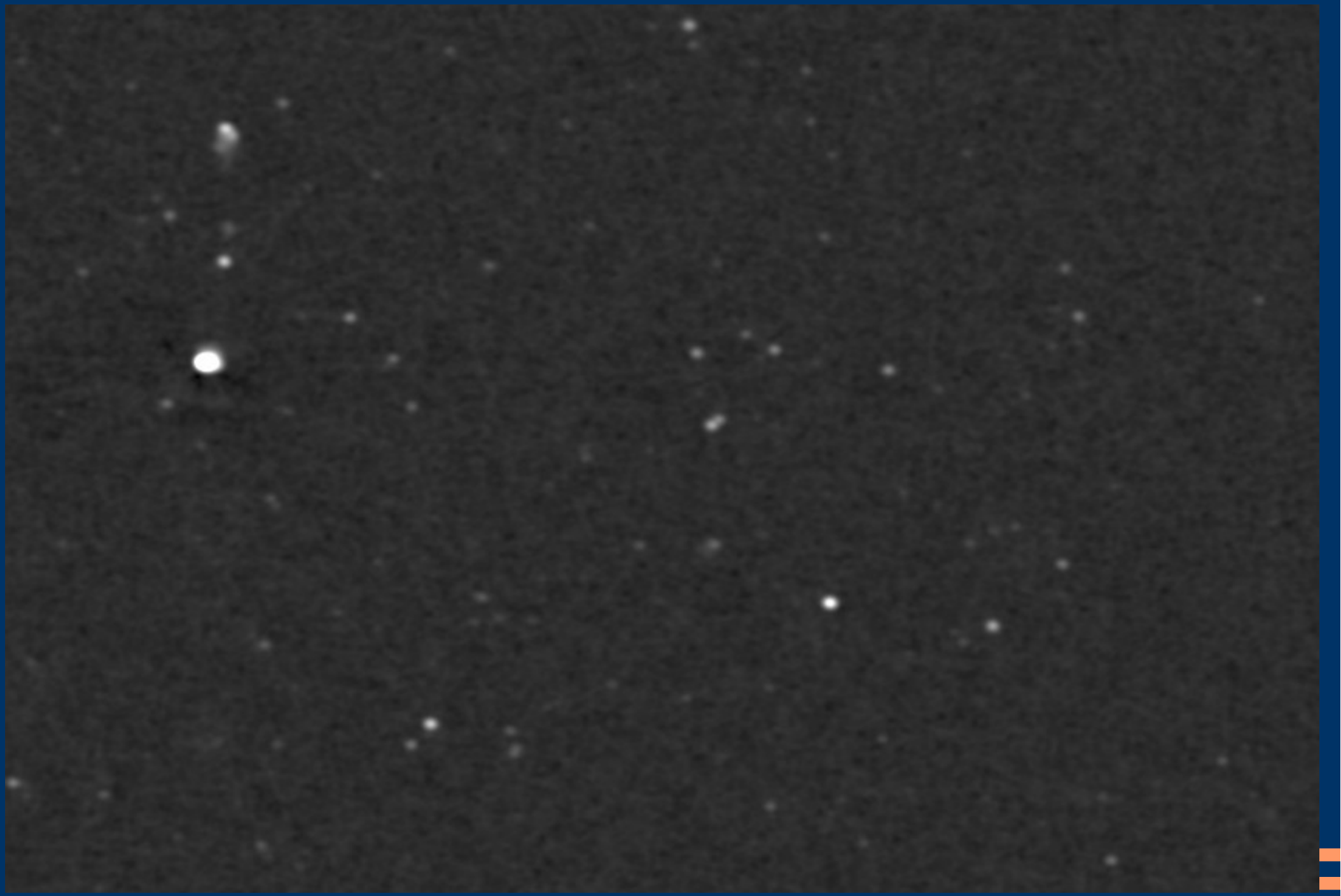
- Memory error correction in GPUs makes them expensive and slow.
- Used griding test on the GTX 285 (no ECC) to determine rate of serious errors
- Facets all the same and compared.
  - 634 executions over 342 hours detected no errors.
- ECC may not be good value for money

Example implementation in Obit

- Implemented AVX version in Obit wideband imager
- Ran on 16 core computer with RAID disks
- Deep integration, EVLA 2-4 GHz,
  - 62 hrs, C, BnA
  - 832 GByte data (before baseline dependent averaging)
- <17 hours for imaging plus self-cal
- 1.1 μJy/bm RMS <3” resolution

Widefield Deep integration

Close up Deep integration



Thanks to Scott Ransom for use of his GPUs, CPUs and expertise