ASKAP - a big data challenge

The Australian Square Kilometre Array Pathfinder

Results from the 2017 Hackathon

achieved large speed-up by optimizing and
accelerating our most time-consuming computations

Team: Steve Ord, Daniel Mitchell, George Bekiaris and Bärbel Koribalski

Mentors: Markus Wetzstein, Claudio Gheller

Our Focus

speed-up most time-consuming processes in ASKAPsoft: these are **gridding** and **CLEANing**

Monday's initial Profile: speed-up vs single core CPU

Convolution	Single core performance	GPU (CUDA)
gridding	492	14 000 (x 28)
degridding	469	38 000 (x 81)
CLEANing	245	926 (x 3.8)

Our Focus

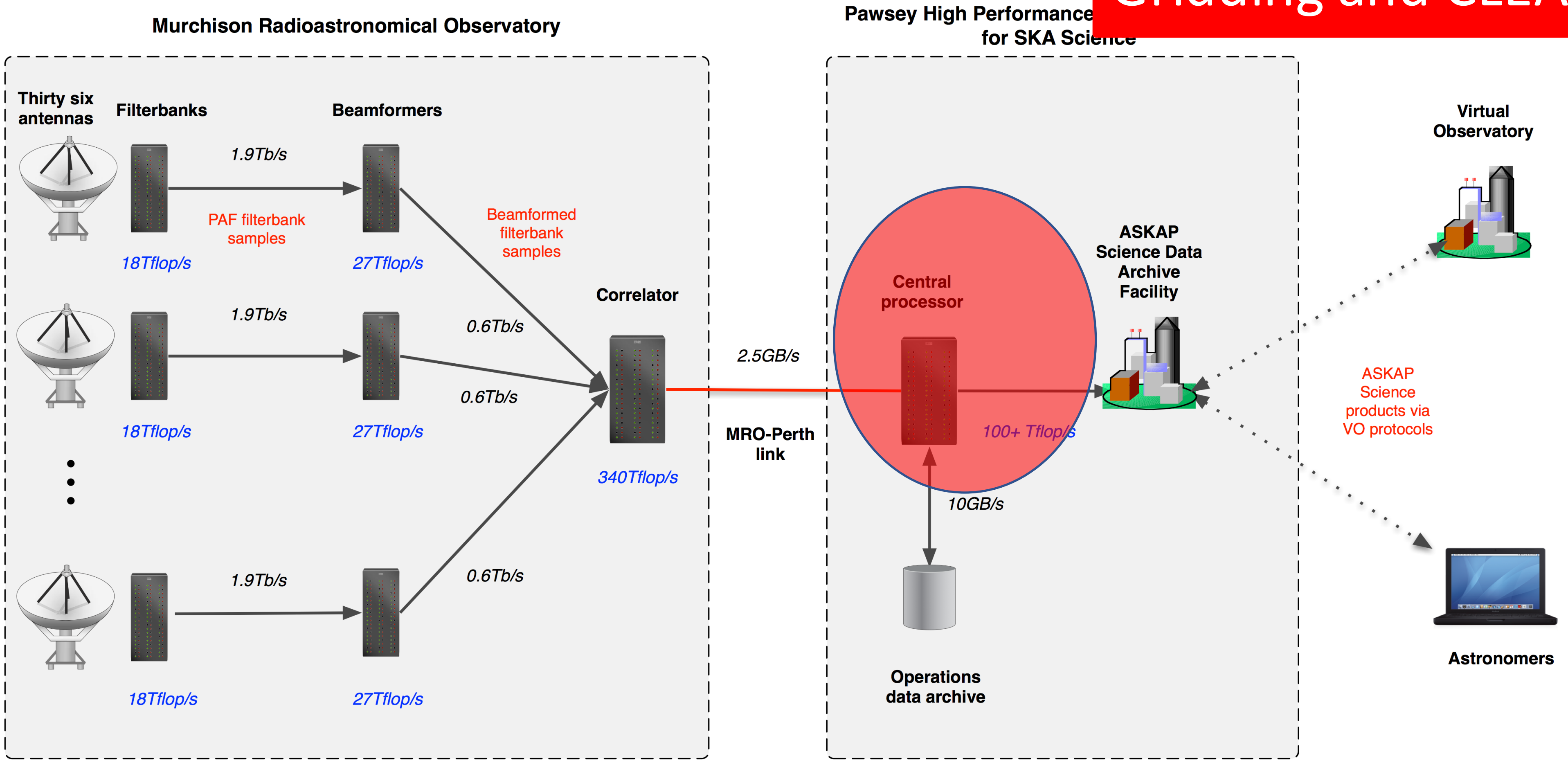
speed-up most time-consuming processes in ASKAPsoft:
these are **gridding** and **CLEANing**

Speed-up vs single CPU core (using pgi compiler)

convolution	Single CPU core	CUDA on GPU	OpenACC		
			GPU	GPU with unified memory (relative to Monday)	12-core CPU with 24 threads (relative to Monday)
gridding	1			23 (202)	4 (38)
degridding	1			40 (210)	14 (72)
CLEANing	1	48.6*	45*	33 (33)	14 (14)

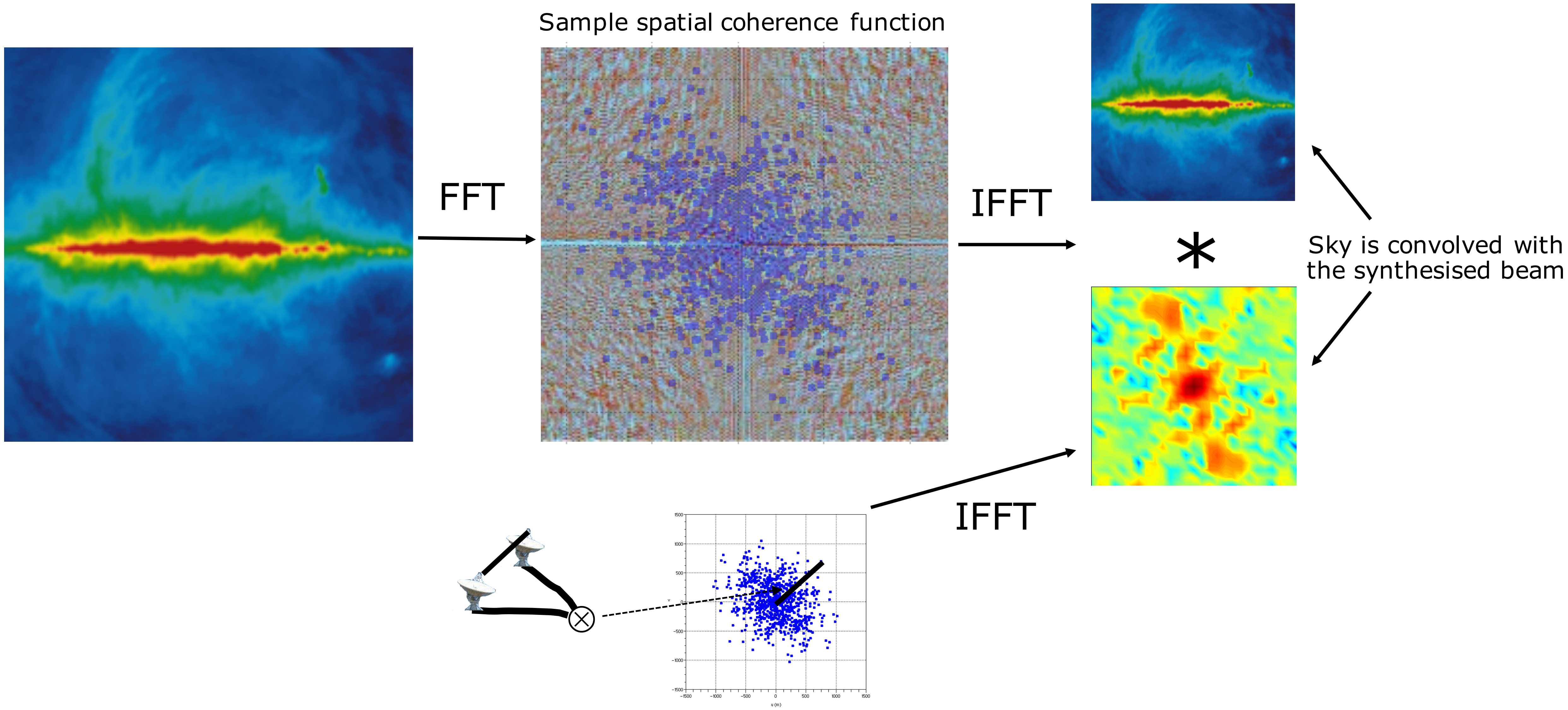
ASKAP data flow

We focus on two tasks within the processor: Gridding and CLEANing

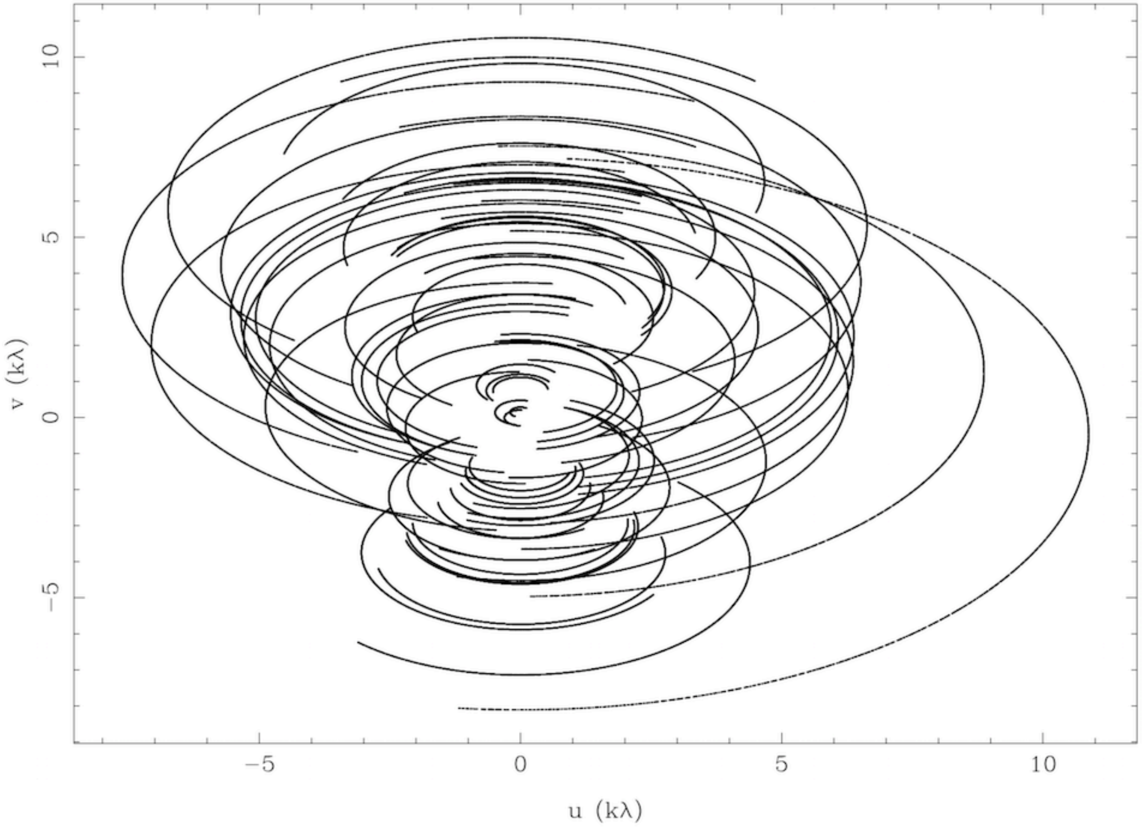


T. Cornwell, February 22 2010

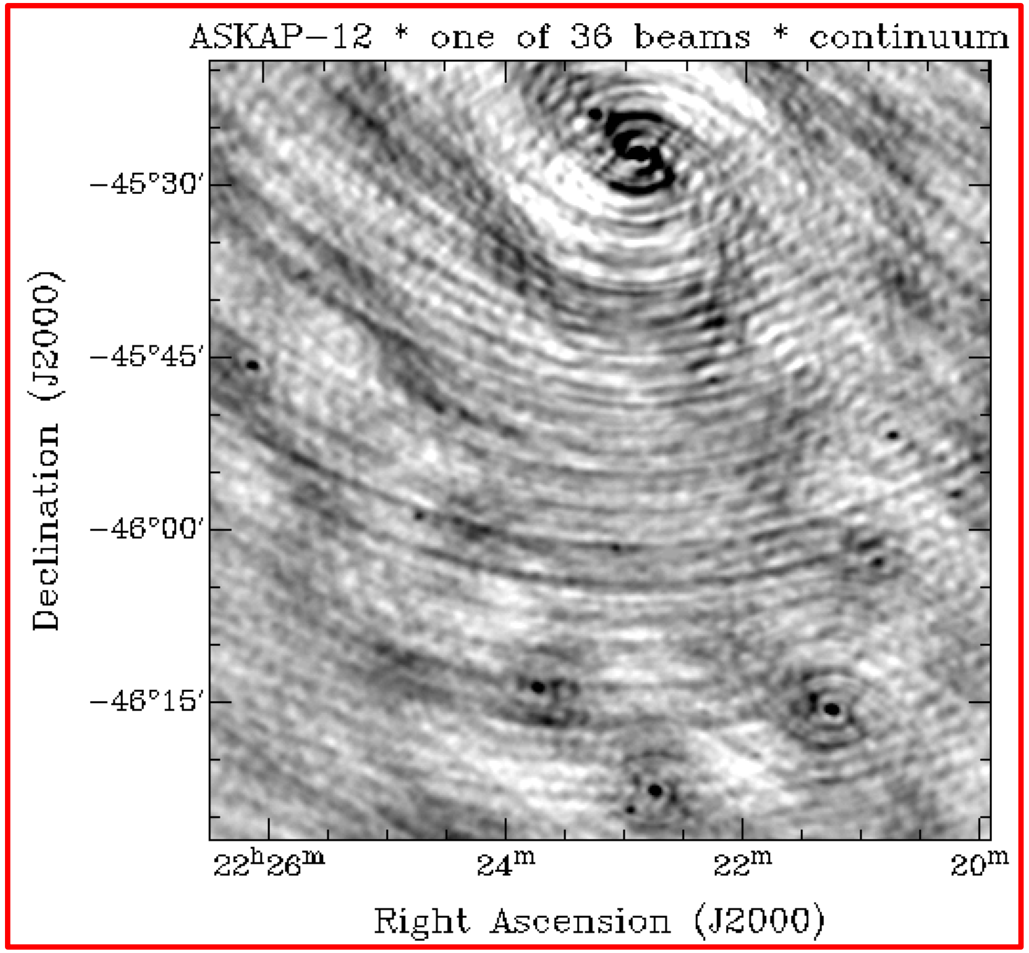
ASKAP data flow



ASKAPsoft: uv-data to CLEANed image

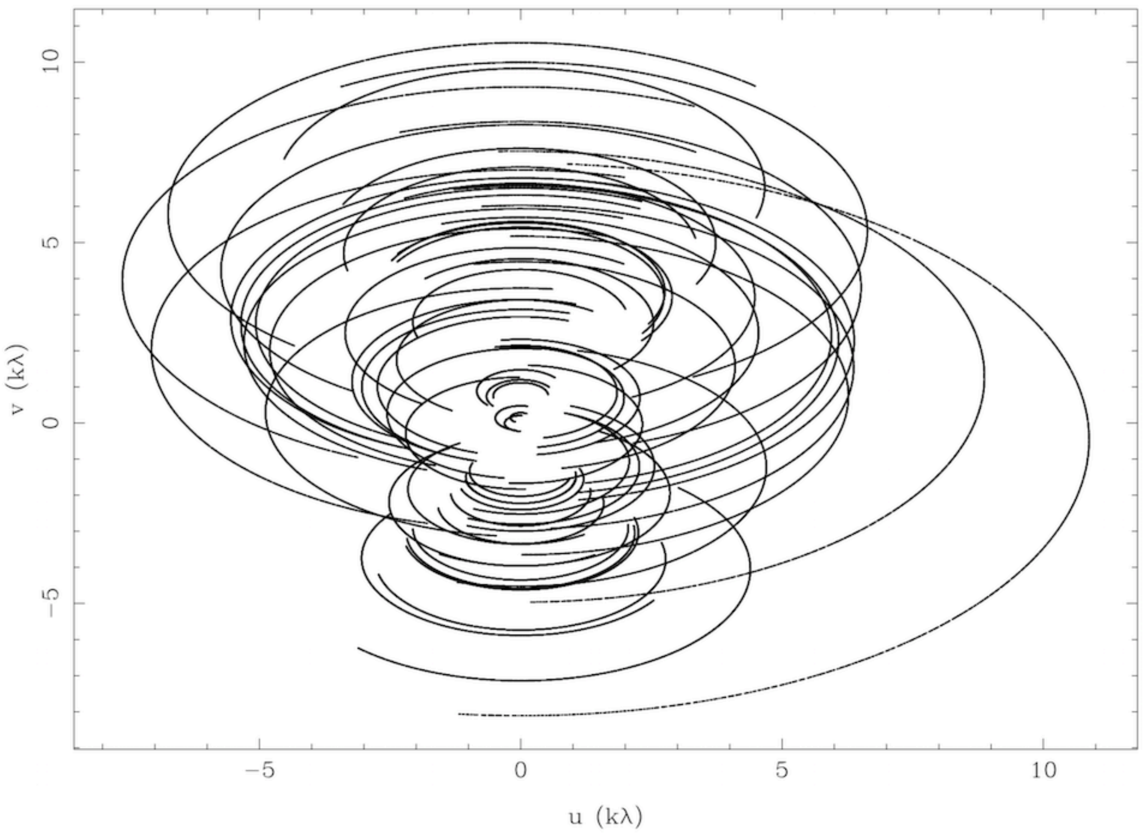


ASKAP-12 array * uv-coverage
(tracks of 66 antenna pairs over 12 hours)
Raw data: amplitude, phase, delay every 5 sec

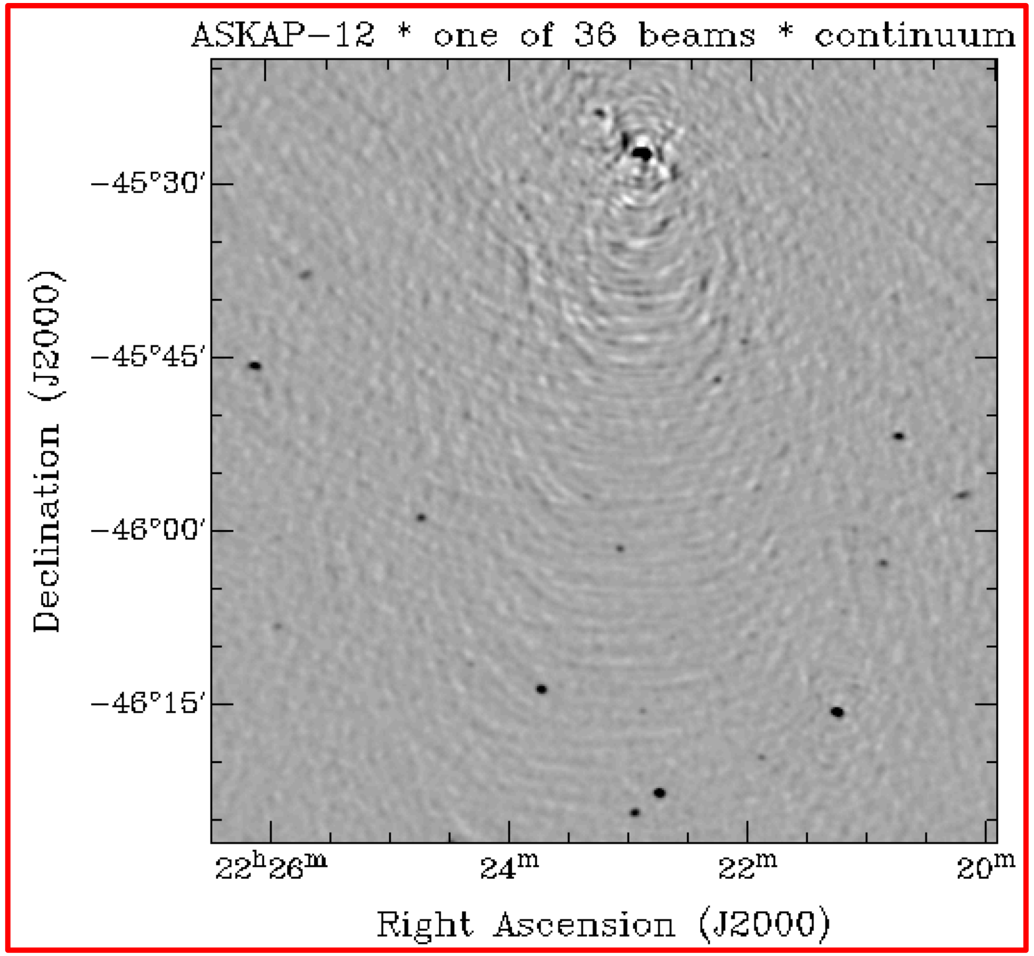


After Gridding and FFT

ASKAPsoft: uv-data to CLEANed image



ASKAP-12 array * uv-coverage
(tracks of 66 antenna pairs over 12 hours)
Raw data: amplitude, phase, delay every 5 sec



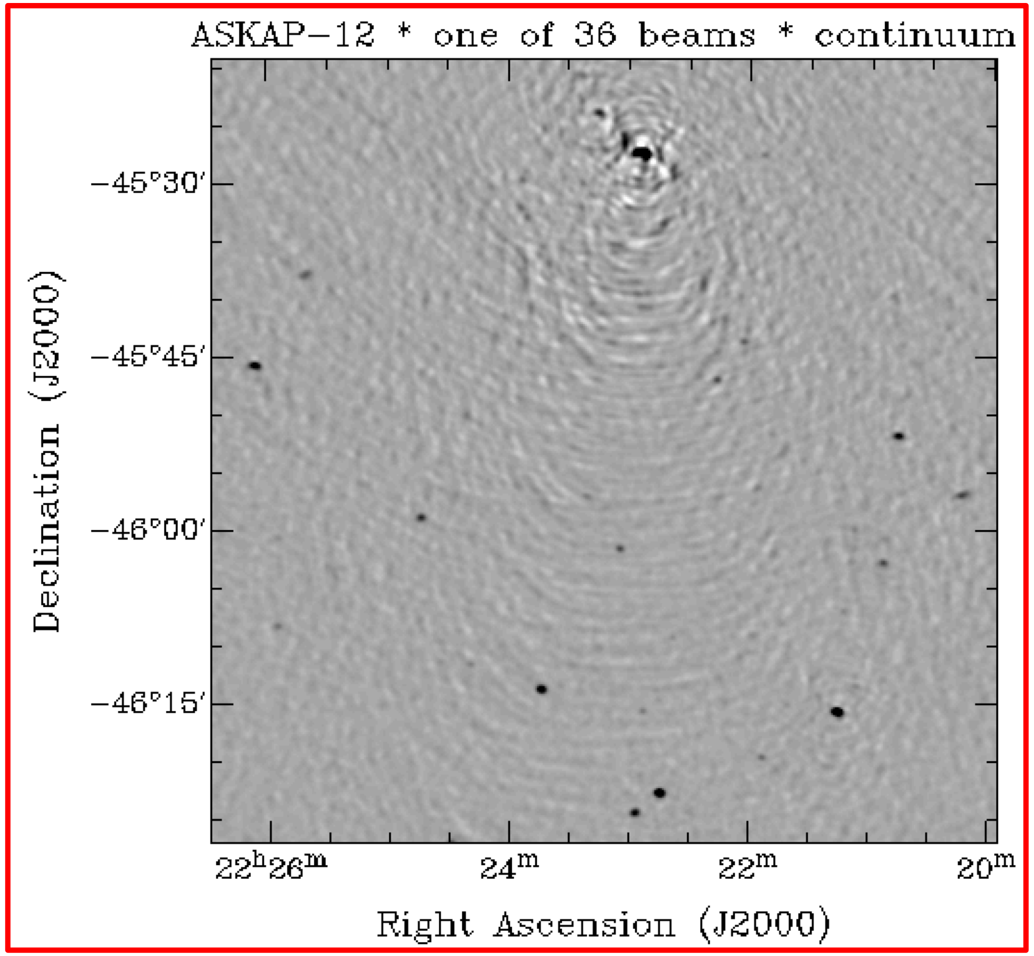
After CLEANing and restor

ASKAPsoft: uv-data to CLEANed image

Computational challenge:

Near real-time processing of
36 beams x 17000 channels
for growing data volumes

Expect: 2.000 radio continuum sources per beam in each field (70M sources in 1000 fields)



After CLEANing and restor

Evolution and Strategy

- Parallel loops over data being gridded (convolutions) or degridded (dot products).
- Atomic adds during gridding.
- Gang loops over data, vectorise convolution loops.
- Replace complex multiplications with floating point operations.

- Compile full codebase with ACC and implement changes
  - Multicore version: speedup on dedicated cray machine; improve memory management.
  - GPU version: tests for future hardware purchases; next generation of software.
- Will test on different platforms:
  - Pawsey Galaxy (500 x 20-core CPUs; 48 Kepler GPUs; gcc & cray compilers)
  - Pawsey Athena (brand new: Pascal GPU, pgi compiler)
  - CSIRO Bracewell (~ 500 Pascal GPU cluster)
  - CSCS Piz Daint

Problems we encountered

- Compiler issues
  - successful with pgi compiler (eg on Pawsey Athena)
  - gcc compiler (immature openacc implementation)
  - need to test the Cray compiler
  - full codebase with pgi compiler (or combination of compilers)

- CUDA does not support atomic min/max for floating points

- Multiple #pragma for GPUs not efficient for CPUs

Was it all worth it ?

- Yes ☺
- Great people, great organisation and great location !

- We will continue the development towards full ASKAP and SKA
  - optimising our code
  - accelerating our code
  - test/deploy best compilers
  - optimise CPU / GPU node usage
  - speed-up ingest (eg, via multi-disk usage ?)

Outlook

- fully implement and test in **ASKAPsoft** pipeline

  => use for ASKAP-12 Early Science (from next week)
- continue development towards full ASKAP (36 antenna array)
- continue development towards SKA Phase 1

in Australia

- 130,000 dipoles
- max. 40 km baseline
- 50 – 350 MHz

in South Africa

- 200 dishes
- max. 150 km baselines
- 0.35 – 1.8 GHz and 4.8 – 13.8 GHz