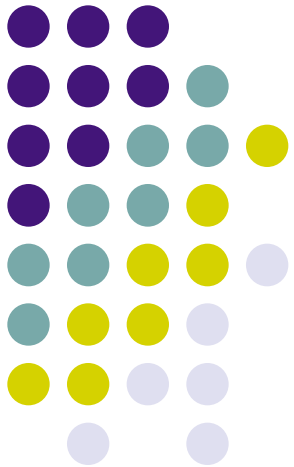
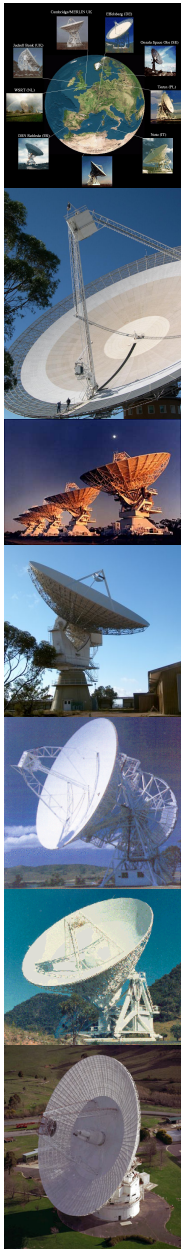
DiFX Overview

Adam Deller
ASTRON

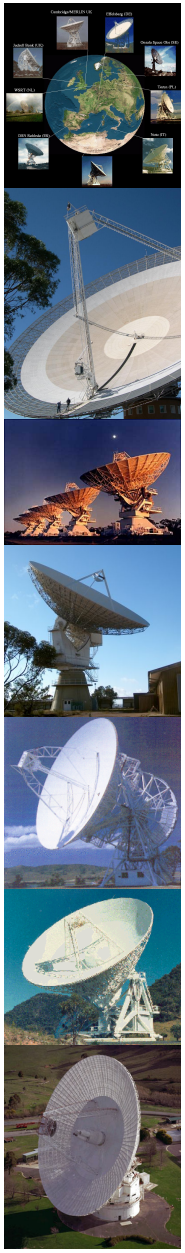


5th DiFX workshop, Haystack Observatory



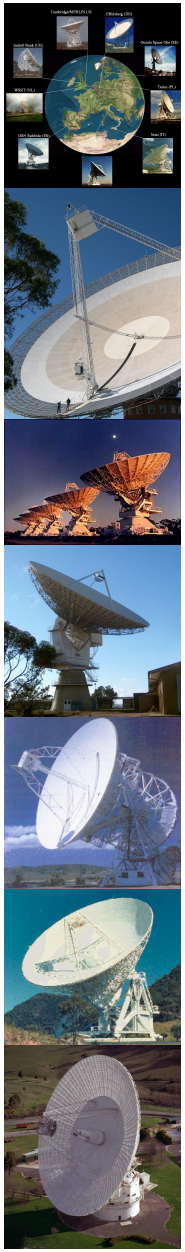
Outline

- History
- Requirements
- SVN layout
- High level overview of DiFX components
- The guts of DiFX (mpifxcorr) in detail
  - Data management and flow
  - The “Core” of the mpifxcorr
  - Scaling and writing of visibilities
- Performance guidelines



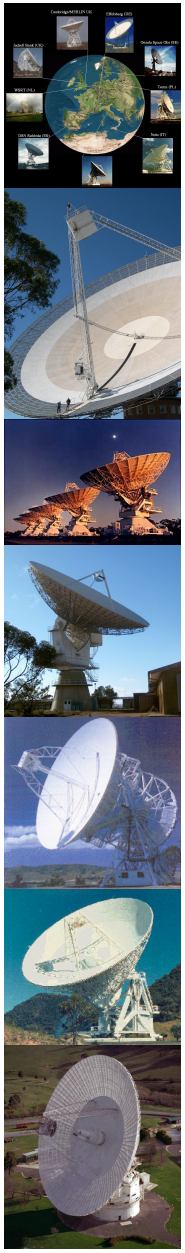
DiFX history

- Developed as a “stop-gap” measure to allow Australian Long Baseline Array to migrate to disk-based recording (2006)
- Development was focused accordingly
- Adopted by NRAO for VLBA for evaluation in 2008, full time in 2009
- Now used at at least 5 or 6 institutes, there are multiple people devoting development time, and over 120 people on the difx-users mailing list



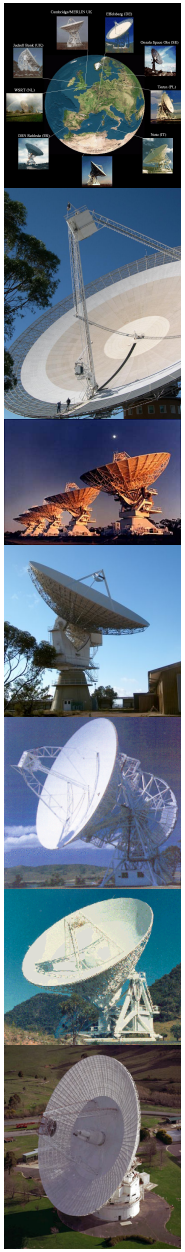
DiFX requirements

- Software:
  - An MPI installation (includes C++ compiler)
  - The Intel Performance Primitives library (free for evaluation, cheap for research use)
- Hardware:
  - At least one x86 style CPU (really, that's it)
  - Ideally: multiple, multi-core Intel boxes connected via GbE, 10GbE or Infiniband
  - Mk5 units if required for Mk5 playback
- Legal:
  - Sign/abide by license agreement (basically, don't use it to make money w/out consent)



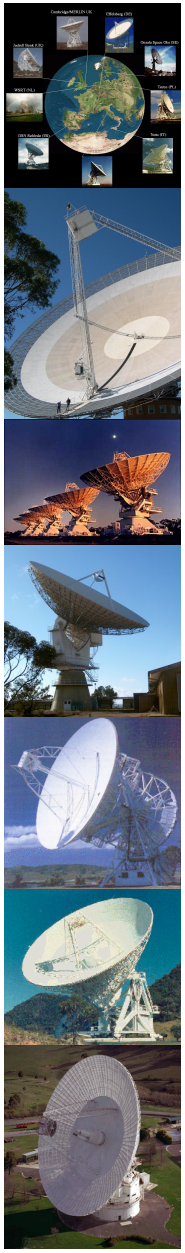
SVN layout

- Two ways to get DiFX from SVN
  - Tagged versions: under “master_tags” at the top level. Current version is DiFX-2.0.1 (DiFX-1.5.4 also available)
  - Active development: trunk [branches/ difx-1.5 no longer developed]
- Tagged versions: Recommended, frozen. Get self-contained units with “svn co difx/master_tags/DiFX-X.x.x/”
- Active: Can change! Grouped [libraries, mpifxcorr, applications, utilities]



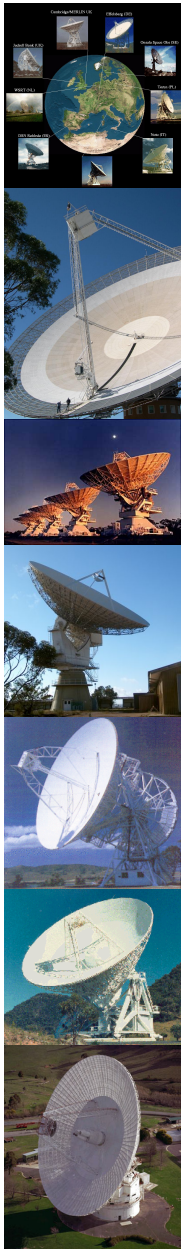
What DiFX "does"

- DiFX the correlation of sampled, quantized **baseband data** from multiple antennas to form **visibilities**, packaged neatly with appropriate metadata in a **useful format**
- Due to its modular layout, it has also proven possible to develop a number of accompanying tools that let you inspect baseband data, output visibilities etc



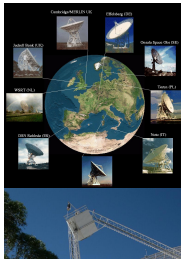
DiFX inputs

- Baseband data:
  - Can be formatted in LBADR, Mk4, VLBA, Mark5B, or VDIF format
  - Can be provided on a linux filesystem, on a Mark5 disk pack, or over ethernet (eVLBI)
- Experiment description and log data
  - Must provide the vex file produced by the experiment scheduling program (usually sched, sometimes sked)
  - Other metadata (clock offsets, Earth Orientation Parameters etc) provided by the user - some scripts to help get started here

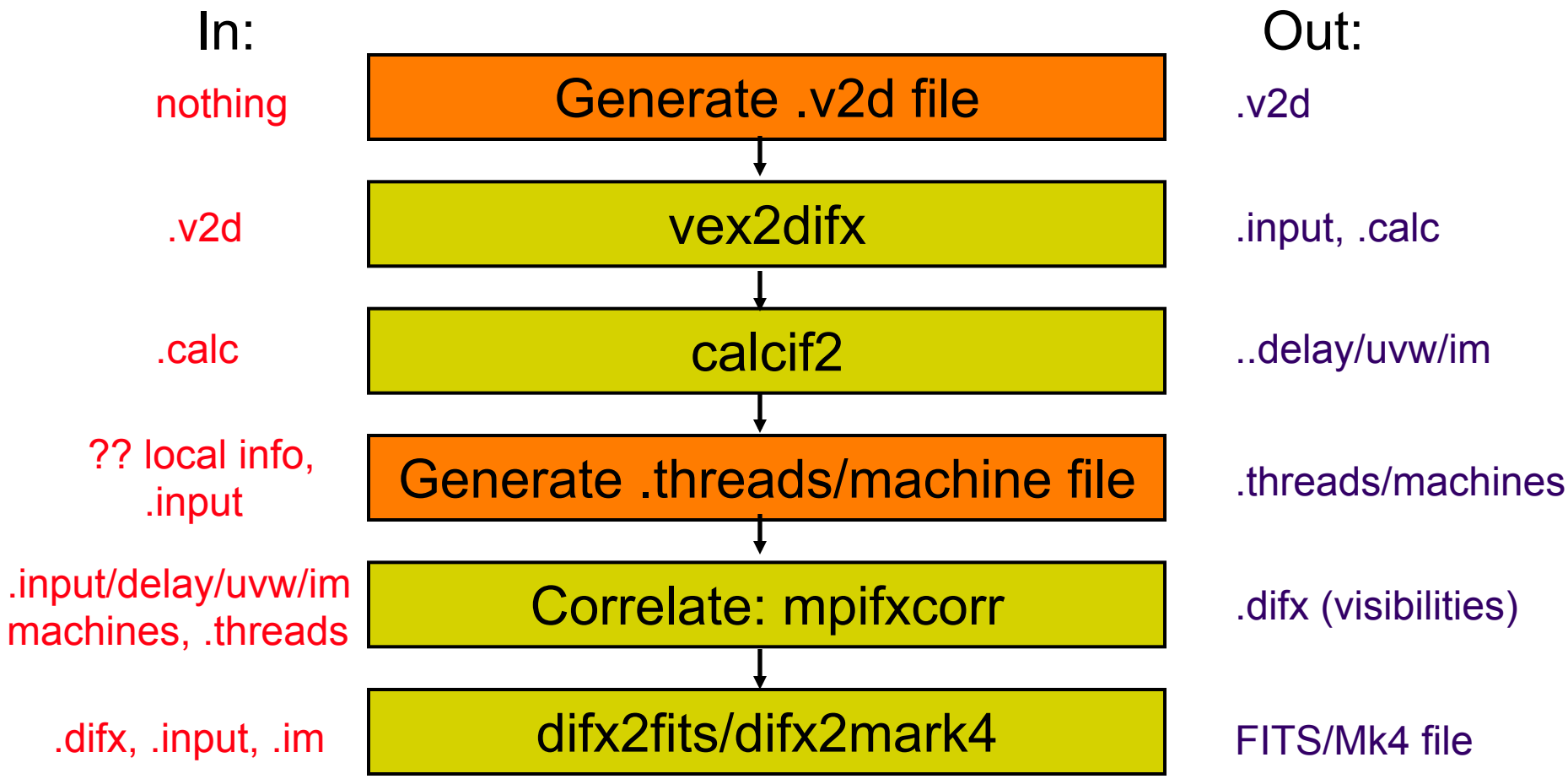


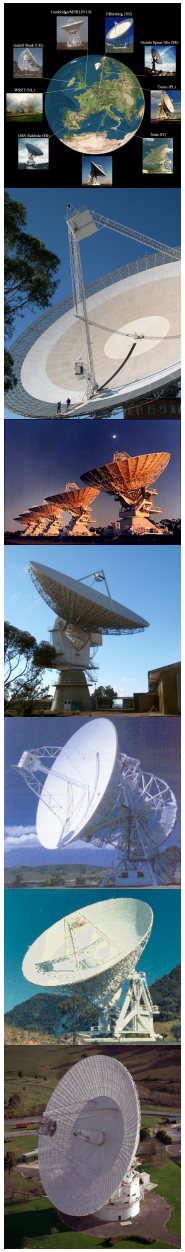
DiFX outputs

- FITS-IDI is the format provided for astronomical observations - DiFX-produced files for the VLBA are basically the same as those produced by the hardware correlator, with some additions
- The Mark4 format read by fourfit and other Haystack programs is a new option (more by others later)



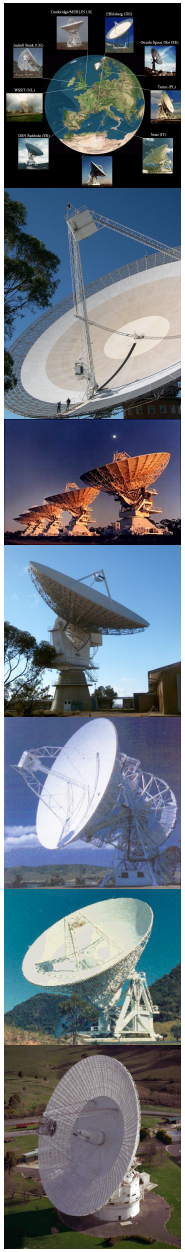
Correlation flow chart





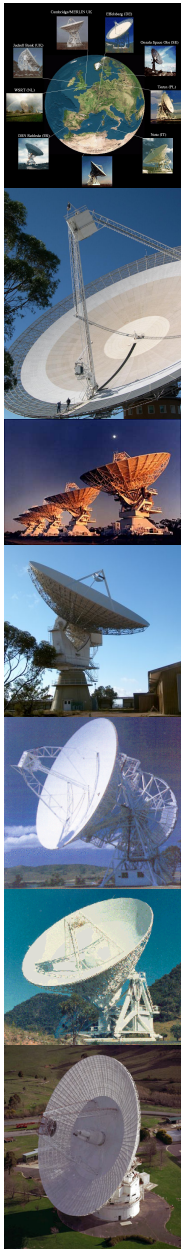
vex2difx: control file generator

- Takes the vex experiment description file and produces the .input file (which describes the setup of the correlator) and the .calc file (which describes the antenna positions, scan durations, source coords etc)
- Input .v2d file must provide path to vex file; can also override defaults for num channels, int time, clock values, data format, …



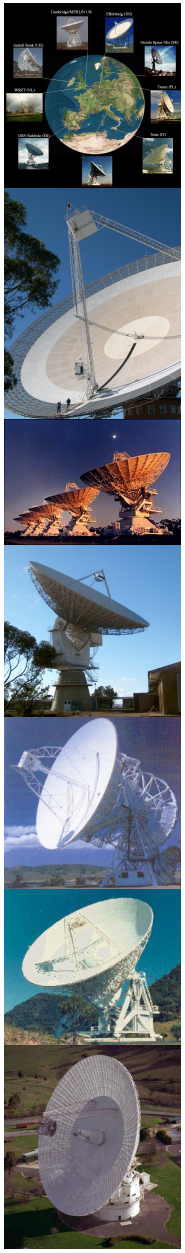
.input file

- Table based:
  - Common: other files, start/stop etc
  - Configuration: Number of channels, anything that might want to change from one scan to the next
  - Freq: IF frequencies, bandwidth, sideband
  - Datastream: Setup for each telescope
  - Baseline: What bands to correlate
  - Data/network: Where to load the data from



.calc file setup

- Table-like format, no explicit delimiters like .input file
  - Start/stop time (not necessarily same as .input file
  - Antenna info (xyz position, mount type, axis offset, …)
  - Source info (RA, dec, [parallax/pm…])
  - Scan info (start, duration, source)
  - Earth Orientation Parameters (EOPs)

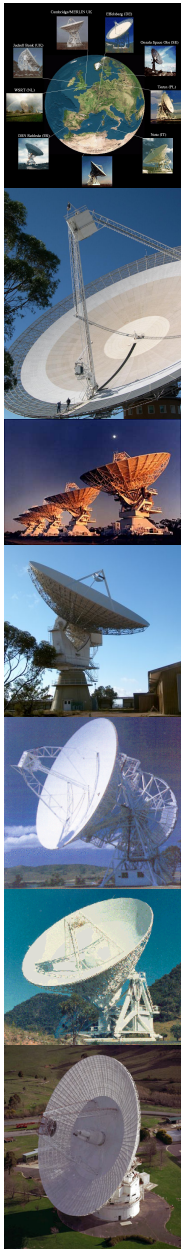


calcif2: geometric model gen.

- calcif2 takes the info given in the .calc file and produces predicted delays and uvw values for each datastream
- This is stored in the .im file:
  - Set of sampled polynomials (usually every 2 minutes).
- DiFX-1.5 note: also generates .delay, .uvw files: densely packed series of samples (usually every second).
  - 2nd order interpolation used in correlator (errors at ~0.1fs level; detectable!)

difx2fits: FITS-IDI builder

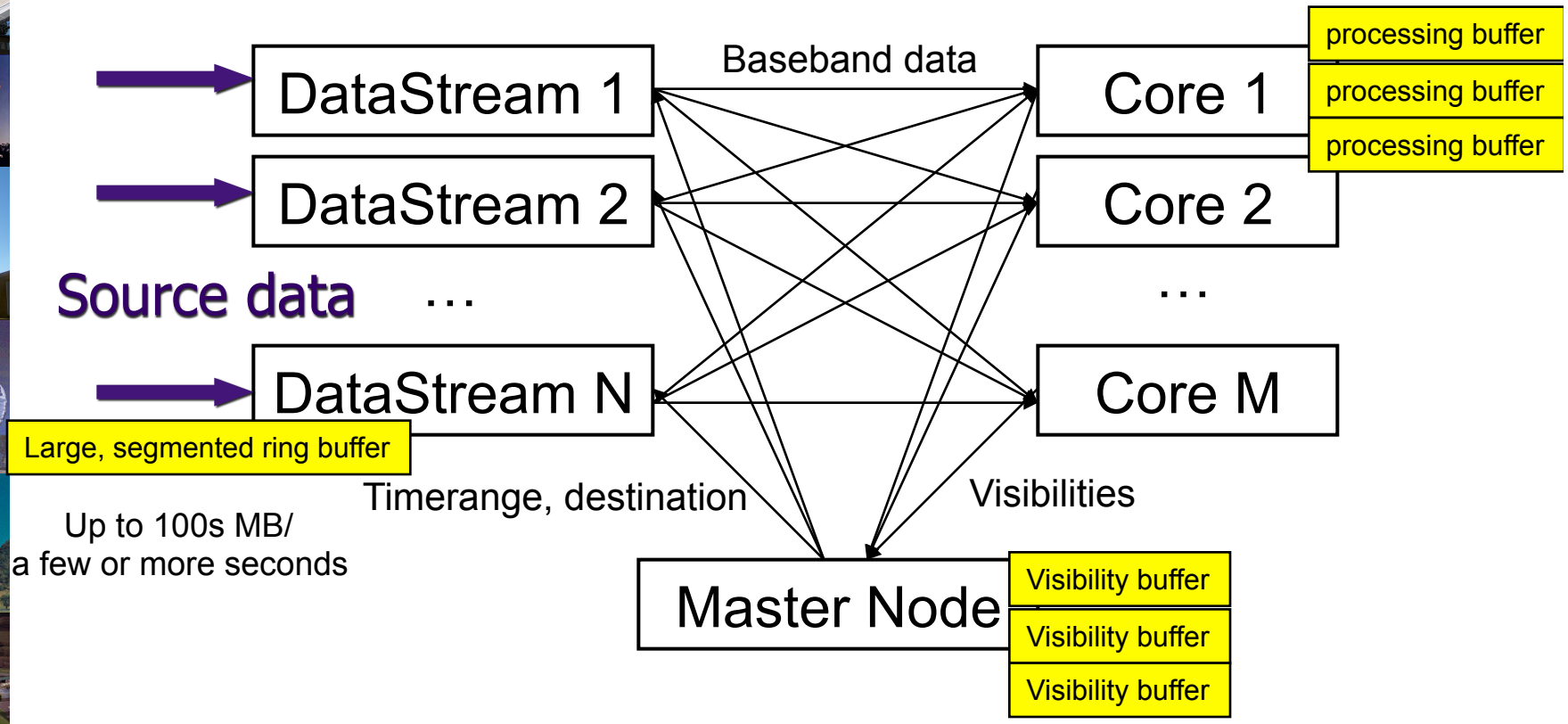
- The visibilities are written out of the correlator in a very simple format: ascii (DiFX1.5) or binary (DiFX2) header, followed by 32bit float binary data
- difx2fits takes these visibilities and the metadata from the control files, and builds a FITS-IDI file that can be loaded into eg AIPS
- More than one correlation can be combined into a single FITS file if compatible
- Further scaling of amplitudes is done



The thorny scaling problem

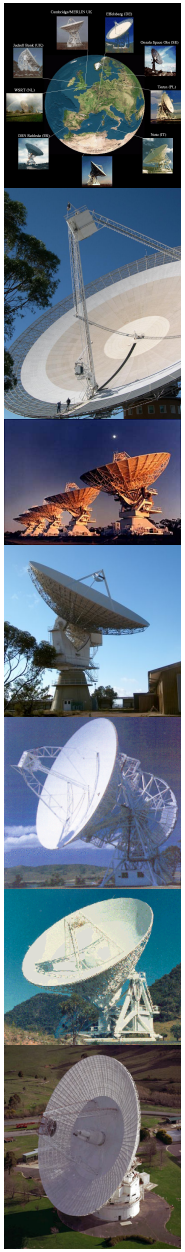
- At least 5 different amplitude effects need to be corrected (van Vleck, unpack vals, …)
- FITLD in AIPS is hard-coded to do some of these
- Therefore the combination of what is done at FxManager + difx2fits must match what FITLD expects
- Current status is documented on the wiki - rationalization at some point?

mpifxcorr architecture



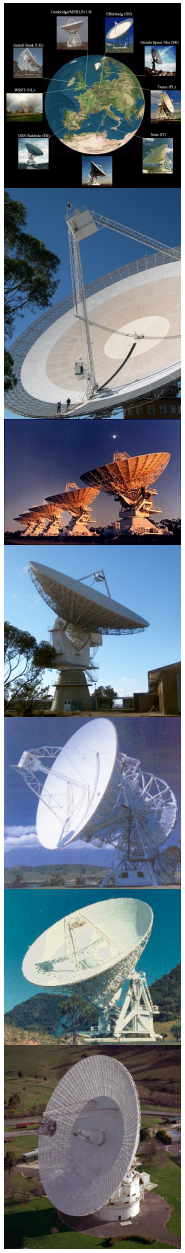
MPI is used for inter-process communications

Each data transfer is double buffered



FxManager correlation flow

- Start at requested time, step one block of FFTs at a time until end of correlation
- Step through scan by scan; if no Configuration matches scan, skip (contiguous scans not required)
  - DiFX1.5 note: Contiguous time, start to finish. Skip time if no active “Configuration”
- As visibilities are completed, release lock on visibility buffer slot (second thread writes out)

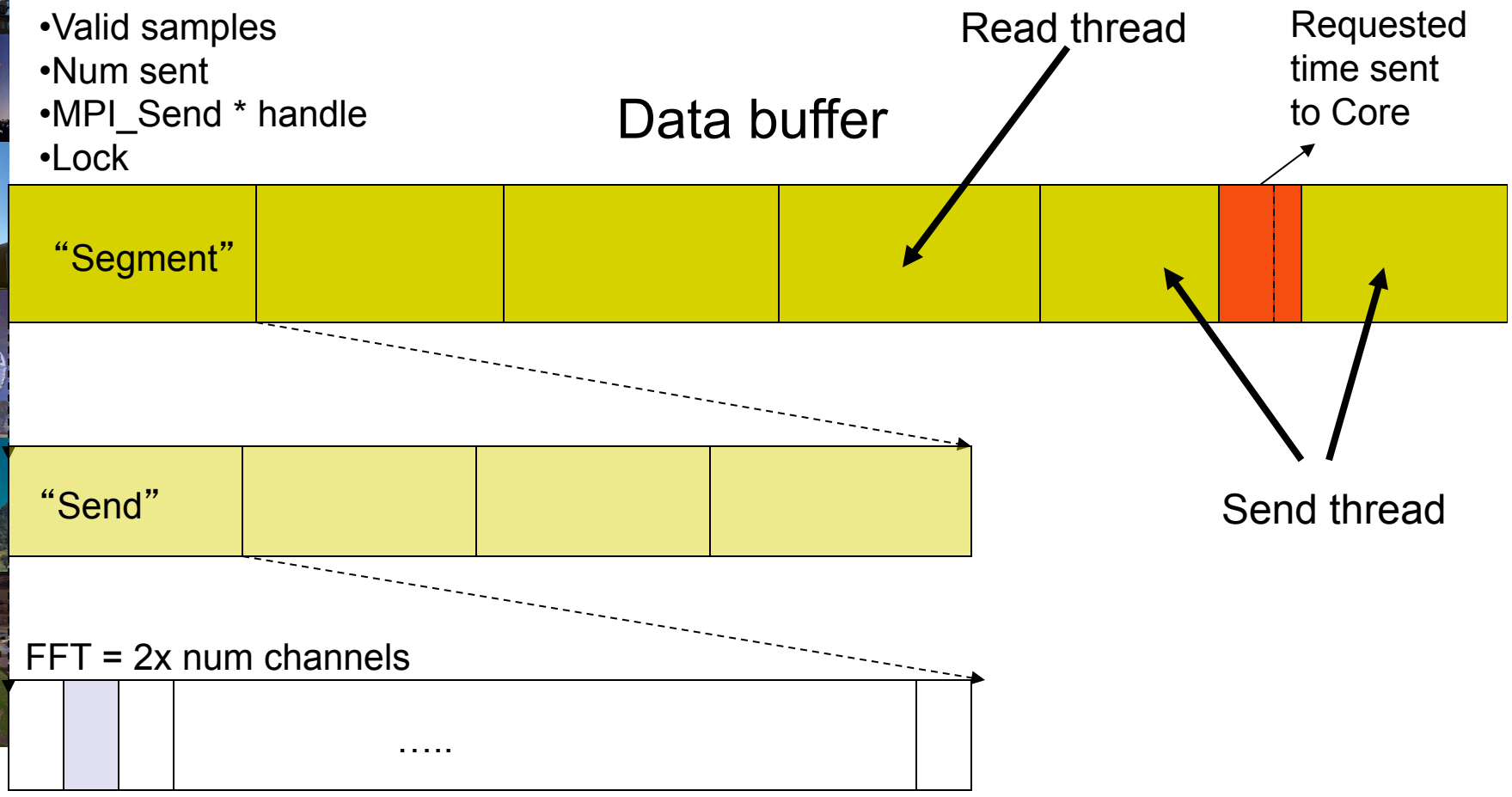


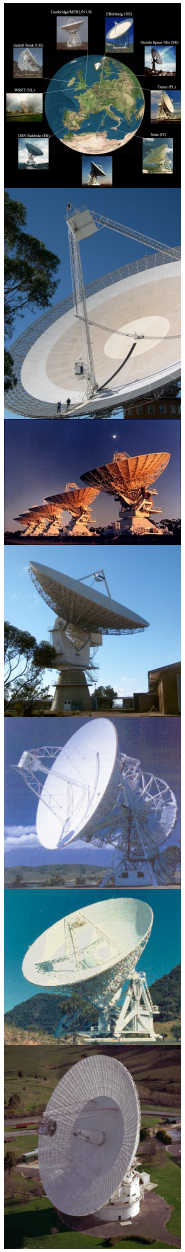
Datastream correlation flow

- Two threads: Main (receives requests, sends data) and read (fills the buffer)
- Each maintains a lock on at least one segment of the databuffer at all times
- While data remains, the read thread will keep populating the data buffer until told to stop
- Main thread just dumbly fulfils requests until told to stop by Manager
- Sends a short flag to Core if no valid data

Datastream correlation flow

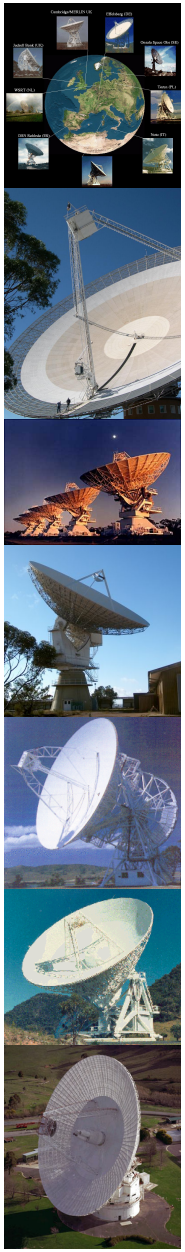
- Start time
- Valid samples
- Num sent
- MPI_Send * handle
- Lock





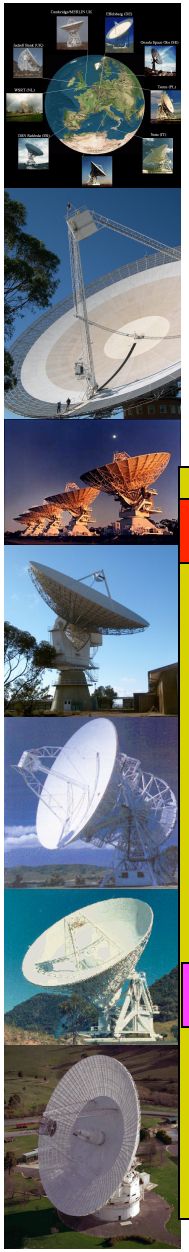
Core correlation flow

- N+1 threads: 1 for send/receive, the rest to do correlation (.threads file)
- One buffer slot is processed at a time - each process thread gets 1/Nth of FFTs
- More locking is required so the threads can aggregate their results, which are stored in one long array (for ease of sending back)
- Keeps looping until a terminate message is received from FxManager



Under the hood in Core

- Each thread is identical, and has an array of “Mode” objects, which handle the station-based processing for each Datastream
- Mode knows how to unpack the different formats, and then handles fringe rotation, FFT and fractional sample correction
- After telling each Mode to do its thing, the thread grabs the appropriate results and XMACs, as described in input file



Core in pictures

Baseband data from each telescope

Read/send thread

Subint visibilities

Subint slot

Core object

Proc. thread

Mode objects for each datastream

XMAC

Thread visibilities

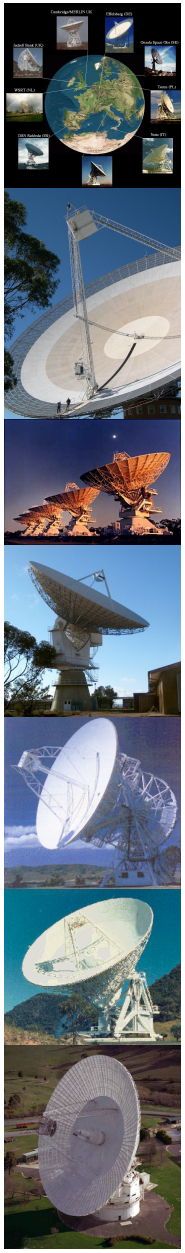
Baseband data pointer

unpacked data

Intermediate data

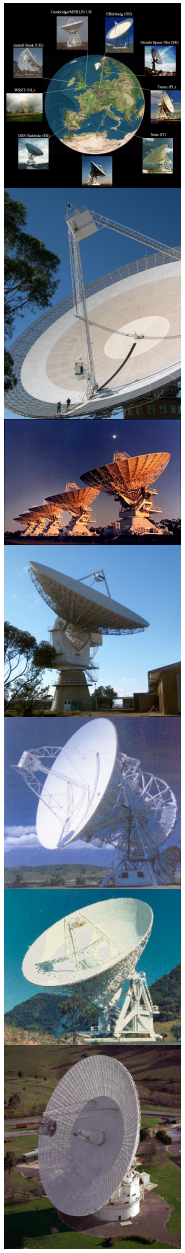
Final data for XMAC

Repeated for each subband



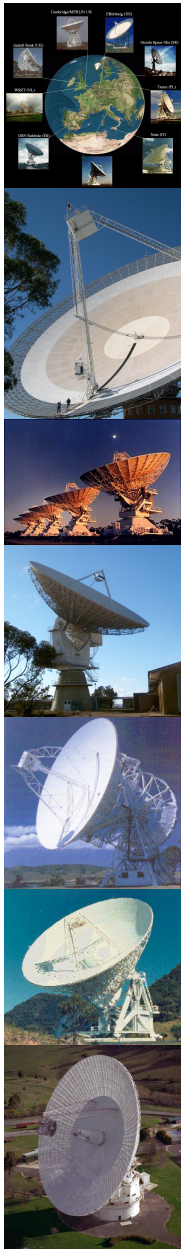
Scaling and writing visibilities

- Along with the sub-integrated visibilities, Core maintains a count of valid samples that were used
- This is sent to FxManager and divided by the expected number of samples
- Used to adjust the visibility amplitude up (and weight down) at the FxManager, before writing to disk
- At the same time, visibilities can be scaled by mean autocorrelations and predicted Tsys, but this is not recommended with difx2fits



Other data products

- Starting with DiFX2, phase cal tone extraction and Tsys extraction is also possible in the correlator
- Phase cal: vex2difx will configure automatically unless overridden.
  - Specifies tone frequency in MHz
  - And which tones go into FITS file
- Tsys: must be configured in .v2d file. Noise diode assumed to be locked to local 1PPS and and switched at integer frequency in Hz



Performance guidelines

- For < ~15 antennas, correlation load is linear with number of antennas, and is always linear with bandwidth
- The old 80 core VLBA cluster (2.5 GHz Xeon CPUs from 2008) can correlate 6+ Gbps of aggregate data rate (i.e. 600+ Mbps/antenna for the 10 station array)
- A 2010-era CPU core gives ~2x better performance that these machines - sandy bridge better again
- Nowadays: 1 CPU core ($100/200) can consume ~150 Mbps (~40 MHz b/w)