

CASA and ALMA and some perspectives in relation to the SKA

Bojan Nikolic

Cavendish Laboratory/Kavli Institute for Cosmology
University of Cambridge

22 July 2010

Disclaimer

These views are entirely my own, and people from the CASA or other projects have not had a chance to comment on them

The big picture

1. The ALMA off-line data reduction package is a formal deliverable of the North American partners to the ALMA project – this package is CASA
 - 1.1 Scientific commissioning tries as a goal to do substantially all tasks in CASA
 - 1.2 In early science it is expected users will use CASA for reducing observations
 - 1.3 The ALMA data reduction pipeline is based on CASA (but this *all* I know about it!)
 - 1.4 Other packages can/could (perhaps with small modification) reduce ALMA data (in fact, demonstrated already)
2. The “official” data reduction package for EVLA is also CASA
 - 2.1 But AIPS is just about still being developed and some commissioning is being done in AIPS
3. Significant (high-quality) man-power over many years has used/committed by the CASA project. Some people working in Europe (Garching/Edinburgh/Bonn)

Data reduction for a ALMA: basics

In interactive mode, the types of tasks are **very** similar to current telescopes:

1. Viewing/flagging of visibility-based data
 2. Calibration:
 - ▶ Bandpass
 - ▶ Time-variable antenna gain
 - ▶ Absolute flux calibration
 3. Gridding/weighting in UV-plane
 4. Deconvolution (mostly Clark clean?)
 - ▶ ALMA will have well-behaved dirty beams in most configurations!
 5. Viewing the results
-
6. Ad-hoc tasks required for commissioning

Data reduction for a ALMA: complications

1. Mosaicing – at present it looks like a lot of science will require this:
 - ▶ Conventional algorithms should suffice
2. Data volumes: ~ 1 Tb / day
 - ▶ High, but can be handled by current hardware
 - ▶ Inconvenient only if you have to keep re-doing the data reduction
3. Model fitting in UV plane:
 - ▶ Good sampling of UV plane reduces the need for this
 - ▶ Not much used in present packages/telescopes anyway
4. Data reduction pipeline
 - ▶ Deliver publication-quality data-cubes to the users

Why this should be a simple job (for mm-wave telescopes)

The entire data process is very **data-centric** with the data models being almost **obvious** and generally well established – in other words almost everybody thinks about the following in the **same** way:

1. Visibility data
2. Flag tables
3. Calibration tables
4. Sky-frequency data cubes (i.e., the results of analysis)

As a result each stage of processing can be neatly separated from the other stages

Some notes about CASA implementation

1. CASA (then AIPS++) was started in 1990 (1?) – it is now ~20 years old
2. It is implemented in C++
 - 2.1 But it essentially does not use C++ libraries in the processing part (there is some STL usage)
 - 2.2 Some FORTRAN/LAPACK routines are used
 - 2.3 QT is used for some viewing applications
3. Some of the functionality implemented in C++ is available in Python
 - 3.1 Bindings are generated from XML descriptions using what is now essentially a custom technique
 - 3.2 A small amount functionality is implemented in XML+Python (the “task” interface)

CASA/casacore split

casacore

- ▶ Basic data structures (arrays, etc)
- ▶ More specialised data structures
 - ▶ Tables
 - ▶ Images
 - ▶ Measurement Set
- ▶ Mathematical operations/algorithms
- ▶ Some deconvolution
- ▶ Coordinate systems/physical units

CASA

- ▶ Even more specialised data structures, e.g., CalibrationTables
- ▶ Calibration
- ▶ Map making / full deconvolution implementation
- ▶ All gui applications (viewers/flaggers)
- ▶ casapy Python binding

Experiences in ALMA commissioning: scientific data calibration/imaging

1. Generally all necessary tools are there and images and data-cubes with ALMA have been made already
2. Flagging/plotting/viewer applications being actively worked on but generally useful already
3. Some not un-expected problems with calibration and imaging very sparse UV plane coverage that were obtained with only 3 or 4 telescopes
4. Performance a bit of a worry (but is being worked on)
5. Astronomers seem to *not* find CASA easy to use – but not sure how much of that is due to lack of understanding of Python and/or synthesis imaging

Experiences in ALMA commissioning: other tasks

1. Nobody in Chile ever seems to have even looked at the C++ code – changing / adding to it seems very unlikely
2. Many tasks consist of:
 - ▶ Extract data from MS into numpy structures
 - ▶ Write Python code to analyse
3. Most people do not really understand Python
4. Many tasks are in-fact done outside of CASA:
 - ▶ Telescope pointing analysis ('TPoint')
 - ▶ Holography ('clic')
 - ▶ Some calibration routines, e.g., delay, focus, pointing offset, baseline vectors ('TelCAL')

Our work here in Cambridge: `wvrgcal`

Overview:

1. Access the recorded WVR data from the Measurement Set (it is stored together with the astronomical data)
2. Compute the coefficients to translate these data to path errors as function of time
3. Translate these to antenna complex gains as function of time
4. Write out a CASA “T Jones” gain calibration table
5. The user can then inspect/apply this calibration table using all of the standard tools in CASA

wvrgcal: implementation

Implementation is as two C++ components: `libAIR` (atmospheric modelling/computation of gains, calibration tables, i.e., everything specific to this problem) and `bnmin1` (minimisation/inference)

External libraries/programs used

1. Boost
2. SWIG
3. GSL

Parts of CASA(-core) used

1. Reading Measurement Set (casacore)
2. Writing gain calibration table (CASA)

wvrgcal: User interaction

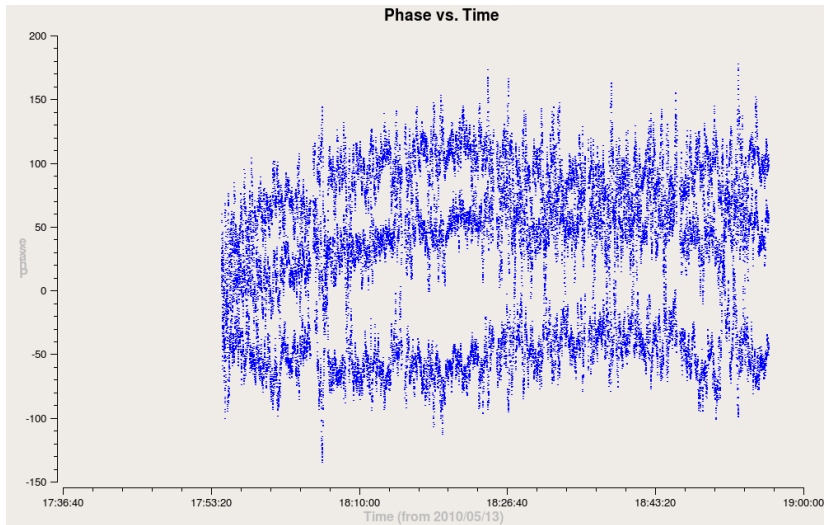
For general users

- ▶ A command line executable
- ▶ Easily callable from CASA using simple wrapper, runs as a **separate process**

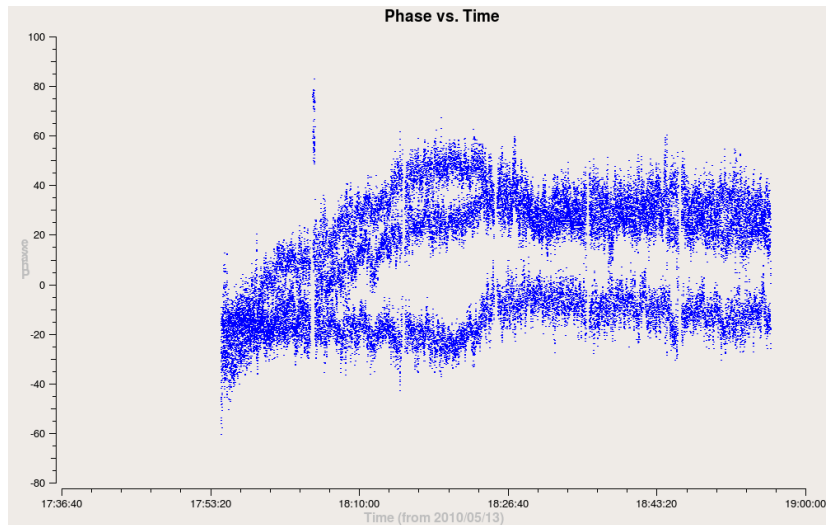
For development

- ▶ Python modules generated with SWIG
- ▶ + some higher level Python code
- ▶ + specialised plotting

Example WVR phase correction



Example WVR phase correction



casata

<http://www.mrao.cam.ac.uk/~bn204/alma/casata.html>

More Python-ic and programmable set of routines building on top of CASA.

- ▶ No use of global variables
- ▶ Automatically instantiate new tools to allow procedures
- ▶ More intelligent data extraction from Measurement Sets (i.e., a task for data extraction)
- ▶ Steps toward more pipeline-like data reduction... simple things like:
 - ▶ Automatically decide some parameters like calibration table names
- ▶ Fully publicly discussed and developed (but only by me so far!)