The Programmable Telescope

Tim Cornwell
ASKAP Computing Lead
New generation of radio telescopes

- Flourishing around the world
  - Atacama Large Millimetre Array (ALMA)
  - Expanded Very Large Array (EVLA)
- Square Kilometre Array has spurred multiple projects
  - Allen Telescope Array - many small antennas
  - LOFAR - low frequency pathfinder
  - ASKAP - phased array feed pathfinder
  - MeerKAT - mid frequency single pixel pathfinder
  - MWA, PAPER - low frequency EOR pathfinders
  - EVLA - high frequency pathfinder
- Many career options!
Square Kilometre Array (SKA)

- 100 times improvement in both sensitivity and field of view
- Baselines ranging from ~ 10m to 3000km
- Frequencies ranging from ~ 60MHz to 25GHz
- Excellent survey telescope
- $2B project run by international consortium
- Australia has proposed to host the telescope
- Design and construction throughout next decade
Use novel technologies to get very large field of view
All sky monitor at low frequencies (< 500MHz)
~ 30 square degree field of view to see neutral hydrogen (~1GHz)
Huge avalanche of data from telescope ~ Exabytes per day
Scientific questions the SKA will address

- Probing the Dark Ages
  - When & how were the first stars formed?
- Cosmology and Galaxy Evolution
  - Nature of Dark Energy and Dark Matter
- Strong-field tests of General Relativity
  - Was Einstein correct?
- Origin & Evolution of Cosmic Magnetism
  - Where does magnetism come from?
- Cradle of Life
  - What and where are the conditions for life?
Australian Strengths

• Single large country with intrinsic “radio-quiet” attributes

Murchison Radio Observatory (MRO): Australia’s SKA Candidate site
Murchison Radio Observatory site office
Murchison Radio Observatory staff
Low frequency projects at MRO

- **Murchison Widefield Array**
  - MIT, Haystack, RRI, Univ. Melbourne, ANU, CSIRO
  - EOR, Solar, transients

- **PAPER**
  - UCB/NRAO/UVA EOR project
  - Uses AIPY package for data processing

- **CoRE**
  - Chippendale/Subrahmanyan EOR experiment
Low frequency projects at MRO

• Murchison Widefield Array
  • MIT, Haystack, RRI, Univ. Melbourne, ANU, CSIRO
  • EOR, Solar, transients

• PAPER
  • UCB/NRAO/UVA EOR project
  • Uses AIPY package for data processing

• CoRE
  • Chippendale/Subrahmanyan EOR experiment
AIPY: Astronomical Interferometry in Python

http://setiathome.berkeley.edu/~aparsons/aipy

Aaron Parsons
Univ. of California
Berkeley
Australian SKA Pathfinder = 1% SKA

- **Wide field of view telescope (30 sq degrees)**
  - Sited at Boolardy, Western Australia
  - Observes between 0.7 and 1.8 GHz
  - 36 antennas, 12m diameter
  - 30 beam phased array feed on each antenna
  - Started construction July 2006
  - 6 antenna prototype (BETA) late 2010
  - Full system late 2012

- **Scientific capabilities**
  - Survey HI emission from 1.7 million galaxies up z ~ 0.3
  - Deep continuum survey of entire sky ~ 10uJy
  - Polarimetry over entire sky

- **Technical pathfinder**
  - Demonstration of WA as SKA site
  - Phased Array Feeds
  - Computing
The Programmable Telescope

- Telescope simulation, observations, data reduction, analysis, visualization, research all becoming scriptable

- Simulations
  - *e.g.* ASTRON MeqTrees package

- Observing
  - *e.g.* ASKAP observing
  - Observations will be scripted in Python
  - Most observing will use canned scripts
  - Special observations will require custom scripts

- Data reduction, analysis, visualization
  - *e.g.* casapy, aipy, parseltongue
  - Solutions are script-based rather than customized for one problem
  - More effective to provide than single purpose custom solution
  - Much more flexible
  - Can take advantage of large packages *e.g.* SciPy

- Requires knowledge of scripting languages
  - Python now dominating
Science archives

- Many (most) telescopes have archived data available on-line
- For ASKAP, primary interaction will be via the science archive
- Great opportunities for research
  - Can pick and choose specific observations
    - e.g. All “Standard calibration” fields from VLA
  - Use scripting to reduce huge volumes of data
- VO interfaces should make this simpler
  - Progressing....
Layers of user interfaces
## Layers of user interfaces

<table>
<thead>
<tr>
<th>User Interface</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Interface</td>
<td>Submit high level commands, queries</td>
</tr>
<tr>
<td>Graphical User Interface</td>
<td>Interactive execution of standard operations</td>
</tr>
<tr>
<td>Scripting</td>
<td>Controlled execution of repeated or complex operations</td>
</tr>
<tr>
<td>Command Line Interface</td>
<td>Interactive execution, debugging scripts</td>
</tr>
<tr>
<td>Compiled Language</td>
<td>Speed, robustness</td>
</tr>
</tbody>
</table>
# Layers of user interfaces

<table>
<thead>
<tr>
<th>Web Interface</th>
<th>Submit high level commands, queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphical User Interface</td>
<td>Interactive execution of standard operations</td>
</tr>
<tr>
<td>Scripting</td>
<td>Controlled execution of repeated or complex operations</td>
</tr>
<tr>
<td>Command Line Interface</td>
<td>Interactive execution, debugging scripts</td>
</tr>
<tr>
<td>Compiled Language</td>
<td>Speed, robustness</td>
</tr>
</tbody>
</table>
### Layers of user interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Interface</td>
<td>Submit high level commands, queries</td>
</tr>
<tr>
<td>Graphical User Interface</td>
<td>Interactive execution of standard operations</td>
</tr>
<tr>
<td>Scripting</td>
<td>Controlled execution of repeated or complex operations</td>
</tr>
<tr>
<td>Command Line Interface</td>
<td>Interactive execution, debugging scripts</td>
</tr>
<tr>
<td>Compiled Language</td>
<td>Speed, robustness</td>
</tr>
</tbody>
</table>

**Ease of learning**
<table>
<thead>
<tr>
<th>User Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Interface</td>
<td>Submit high level commands, queries</td>
</tr>
<tr>
<td>Graphical User Interface</td>
<td>Interactive execution of standard operations</td>
</tr>
<tr>
<td>Scripting</td>
<td>Controlled execution of repeated or complex operations</td>
</tr>
<tr>
<td>Command Line Interface</td>
<td>Interactive execution, debugging scripts</td>
</tr>
<tr>
<td>Compiled Language</td>
<td>Speed, robustness</td>
</tr>
</tbody>
</table>
# Layers of user interfaces

<table>
<thead>
<tr>
<th>Interface Type</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Interface</td>
<td>Submit high level commands, queries</td>
</tr>
<tr>
<td>Graphical User Interface</td>
<td>Interactive execution of standard operations</td>
</tr>
<tr>
<td>Scripting</td>
<td>Controlled execution of repeated or complex operations</td>
</tr>
<tr>
<td>Command Line Interface</td>
<td>Interactive execution, debugging scripts</td>
</tr>
<tr>
<td>Compiled Language</td>
<td>Speed, robustness</td>
</tr>
</tbody>
</table>
### Layers of user interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Interface</td>
<td>Submit high level commands, queries</td>
</tr>
<tr>
<td>Graphical User Interface</td>
<td>Interactive execution of standard operations</td>
</tr>
<tr>
<td>Scripting</td>
<td>Controlled execution of repeated or complex operations</td>
</tr>
<tr>
<td>Command Line Interface</td>
<td>Interactive execution, debugging scripts</td>
</tr>
<tr>
<td>Compiled Language</td>
<td>Speed, robustness</td>
</tr>
</tbody>
</table>

- **Ease of learning**: Higher up the scale means easier to learn.
- **Power**: Higher up the scale means more powerful.

*Wednesday, 15 April 2009*
Astronomical scripting in Python

- Many examples
  - see e.g. Topical software at scipy.org

<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIPS + ParselTongue</td>
<td>Radio astronomy processing using venerable AIPS package</td>
</tr>
<tr>
<td>AIPY</td>
<td>Astronomy Interferometry in Python (including MIRIAD)</td>
</tr>
<tr>
<td>CASA + casapy</td>
<td>Radio astronomy processing for ALMA, EVLA</td>
</tr>
<tr>
<td>IRAF + PyRAF</td>
<td>Optical/IR processing</td>
</tr>
<tr>
<td>MeqTrees</td>
<td>Fitting parametrized measurement equations</td>
</tr>
<tr>
<td>AR runtime</td>
<td>Python access to VO tools</td>
</tr>
<tr>
<td>PyEphem</td>
<td>General purpose ephemeris tools</td>
</tr>
<tr>
<td>PyFITS</td>
<td>Access to FITS files</td>
</tr>
<tr>
<td>ppgplot</td>
<td>Tim Pearson's PGPLOT plotting library</td>
</tr>
<tr>
<td>matplotlib</td>
<td>MATLAB-like plotting library</td>
</tr>
<tr>
<td>S2PLOT</td>
<td>Visualization of data, including 3D</td>
</tr>
<tr>
<td>SWIG</td>
<td>Connect python to e.g. C, C++, FORTRAN</td>
</tr>
<tr>
<td>SciPy</td>
<td>General scientific processing capabilities</td>
</tr>
</tbody>
</table>
AIPS + ParselTongue

- AIPS = 30 year old data reduction package
  - Very powerful package with poor interface - very limited interpreter
- ParselTongue = python interface to AIPS
  - See http://www.radionet-eu.org/rnwiki/ParselTongue
  - Given new life to old package
  - But scripts tend to be very long-winded

- Excessively simple example:

```python
from Wizardry.AIPSData import AIPSUVData
from AIPS import AIPS

from pylab import plot, show

AIPS.userno = 667

data = AIPSUVData('MULTI', 'UVDATA', 1, 3)
u = []
v = []
for visibility in data:
    u.append(visibility.uvw[0])
    v.append(visibility.uvw[1])

plot(u, v, '.')
show()
```
MeqTrees

- Simulator for any telescope with a Measurement Equation
- See [http://www.astron.nl/meqwiki/](http://www.astron.nl/meqwiki/)
Other possibilities

<table>
<thead>
<tr>
<th>MATLAB</th>
<th>Very powerful package for matrix-based operations - very expensive but cheap for students</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCTAVE</td>
<td>Open source MATLAB workalike</td>
</tr>
<tr>
<td>IDL</td>
<td>Commercial package heavily used in e.g. NASA</td>
</tr>
<tr>
<td>GNU Data Language (GDL)</td>
<td>GNU Data Language - Open source IDL workalike</td>
</tr>
<tr>
<td>PERL</td>
<td>Very powerful but obscure scripting language</td>
</tr>
</tbody>
</table>

- MATLAB widely used in some fields
  - Scripts published for “reproducible research” in signal/image processing
- OCTAVE good enough for many purposes
- For some high level work, MATLAB/OCTAVE are superior to python

```matlab
function count = logsourcecount(S)
    a = [0.841, 0.540, 0.364, -0.063, -0.107, 0.052, -0.007];
    range = 0.6;
    count = -2.5 * S + (sum(a.*S.^range));
    return
endfunction
```
Why python?

• Complete scripting language
  • Mature
  • Very rich set of capabilities

• Open Source
  • Available on most architectures

• Interpreted rather than compiled
  • Up to 5 times more productive

• Widely used in science
  • Powerful standard libraries

• Interactive development
  • Can build scripts line-by-line

• Multiple programming models
  • Objects, imperative, functional

• Extensible
  • Python modules
  • C/C++ modules
My preferred approach

- **Choose your tools**
  - I use casapy - has excellent synthesis tools and powerful data access

- **Script all steps**
  - Observing setup, schedule, archive access, data reduction, visualization, analysis, modeling

- **Incrementally develop scripts**
  - Add functionality and rerun (if feasible)
  - Print and plot lots of diagnostics

- **Keep results in one directory (or tree)**
  - Add notes file pointing to scripts
CASA imaging of 406 ATCA pointings

```python
msname = "cena-core.0.ms"
algorithm = 'multiscale'
qcellsize = '10arcsec'
npix = 2*1024
nchan = 13
start = 1
spw = range(2)
fields = range(1, 406)

im.open(msname)

for field in fields:
    model = 'cena-core.'+str(field)+'.'+algorithm
dirty = model+'.dirty'
restored = model+'.restored'
mask = model+'.mask'
print model
#
# Specify the data and image parameters
#
print im.selectvis(spw=spw,field=field)
print im.defineimage(nx=npix, ny=npix,
cellx=qcellsize, celly=qcellsize,
stokes="I" , spw=spw,
phasedcenter=field, nchan=1, mode='mfs')
print im.setoptions(ftmachine='ft')
print im.uvrange(uvmin=50,uvmax=5000)
#
# Weight the data
#
print im.weight('briggs', robust=0.0, npixels=2*npix)
print im.sensitivity()
print im.setmcontrol(scaletype='SAULT', stoplargenegatives=0,
cyclefactor=3, cyclespeedup=1000)
scales = [0,8,16,32,64]
print im.setscales('uservector', uservector=scales)
niter = 1000
threshold = '5.0mJy/beam'
print im.boxmask(mask, blc=[npix/4, npix/4],
                 trc=[3*npix/4-1,3*npix/4-1])
print im.setbeam('60arcsec', '40arcsec', '0deg')
print im.clean(algorithm='multiscale',
               model=model,
               residual=model+ '.residual',
               image=model+ '.restored',
               displayprogress=F,
               mask=mask,
               threshold=threshold, niter=niter, gain=0.75)
```
CASA

- Large C++ library for astronomical processing
  - See http://casa.nrao.edu/
- IPython interface
  - Cannot yet import CASA into already running python
- Tool-based or task-based interface
  - Task-based interface non-pythonic!
- Tool interface most useful for scripting

```python
ia.maketestimage('image.large', overwrite=True)
innerquarter=ia.setboxregion([0.25,0.25],[0.75,0.75],frac=true)
ia.fromimage(outfile='image.small', infile='image.large',
              region=innerquarter, overwrite=True)
ia.close()

ia.open('image.small')
cl1 = ia.fitsky(models=['gaussian'])
print cl1.keys()
# 'pixels','pixelmask','converged','return'
print cl1['return'] # 'return' field holds a componentlist record
if cl1['converged']:
    ia.close()
    ia.open('image.large')
cl2 = ia.fitsky(models=['gaussian'], estimate=cl1['return'])
print cl2['return']
ia.close()
```
Career options in astronomy

- **Information Technology now central to astronomy**
  - Competitive advantage goes to people with significant IT capabilities
- **Astronomer**
  - Ph.D. in astronomy or equivalent
  - Scripting very desirable
  - Compiled languages optional
- **Instrument specialist**
  - Ph.D. in astronomical techniques or equivalent
  - Scripting essential - most processing is ad-hoc
  - Compiled languages optional
- **Science software developer**
  - Ph.D. in astronomy or astronomical techniques or equivalent
  - Scripting essential
  - Compiled languages very desirable
- **Software developer in Astronomy**
  - Degree in Computer Science or equivalent
  - Scripting and compiled languages essential
Thank you