Efficient remote interactive pipelines using CASA and Jupyter

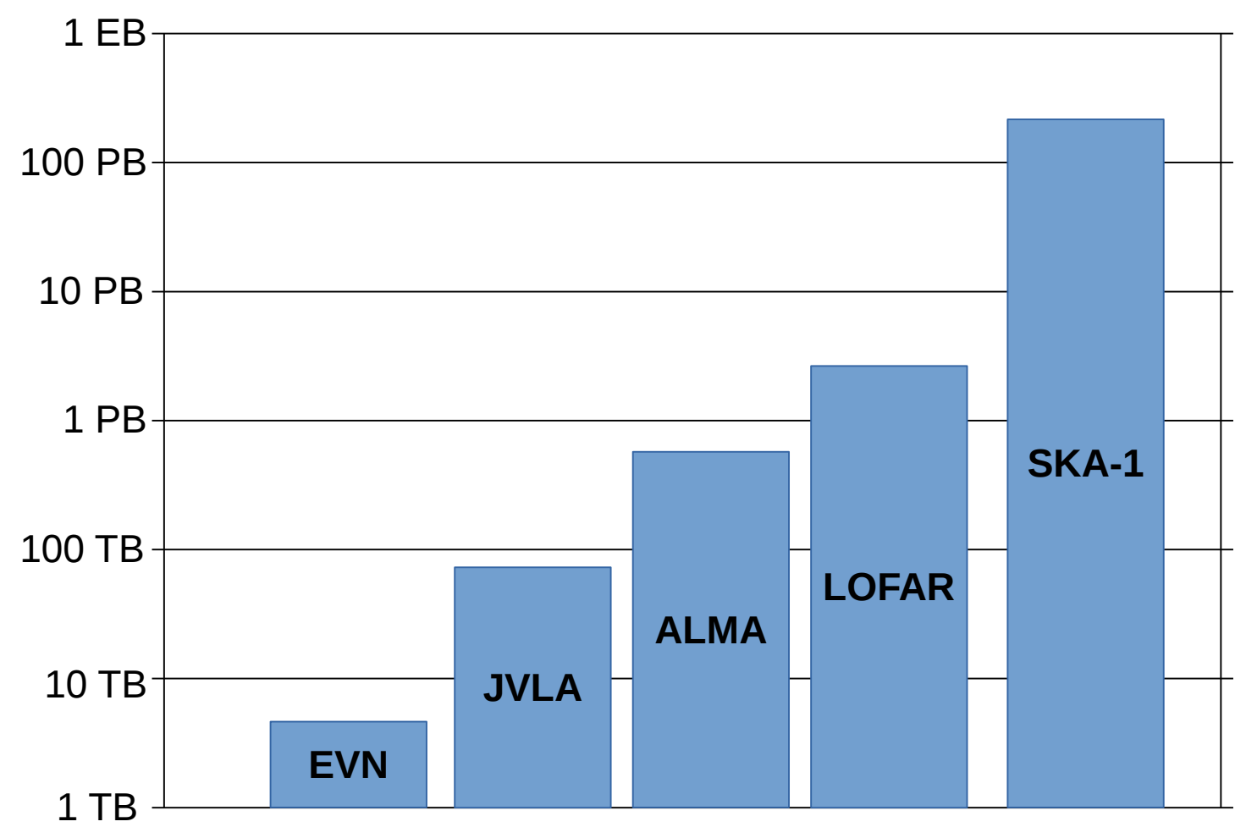
JIVE
Joint Institute for VLBI
ERIC

Aard Keimpema, Mark Kettenis, Des Small, Arpad Szomoru (JIVE),
and Tammo Jan Dijkema (ASTRON)

ESCAPE
European Science Cluster of Astronomy & Particle physics ESFRI research Infrastructures

Remote data processing

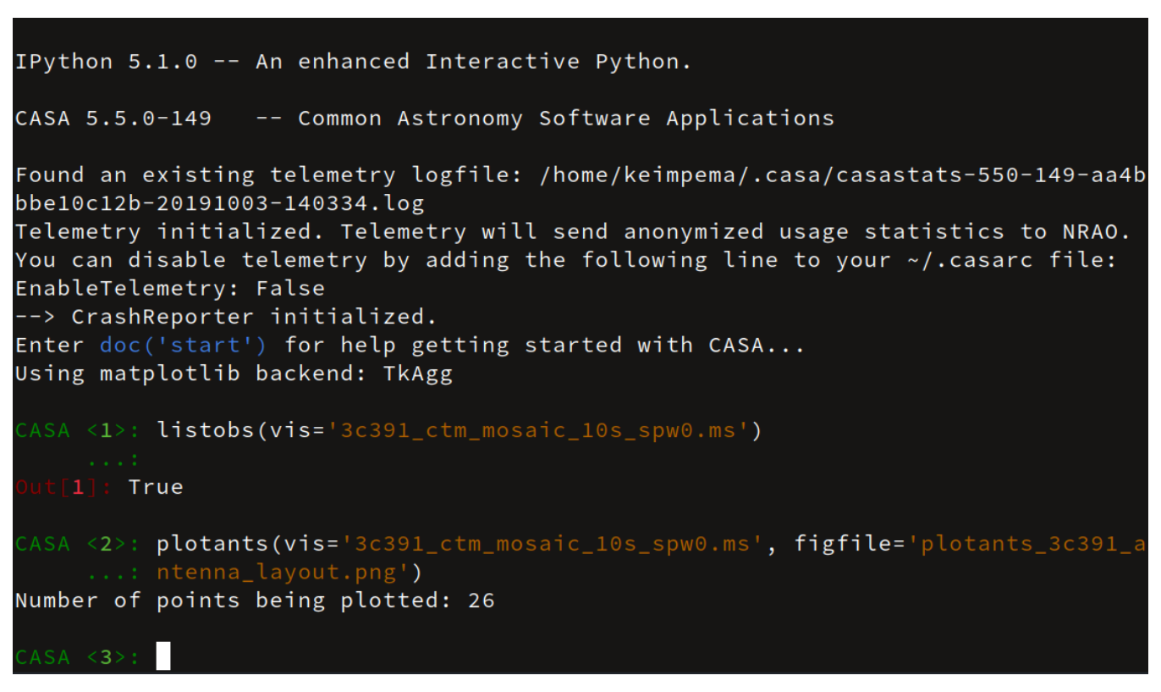
- Data volumes have increased dramatic
- The SKA will produce 1 PB of archivable data per **day**
- Near data processing using remote pipelines will be a necessity
- Need both *batch* and *interactive* processing
- Current CASA based remote pipelines (e.g. Alma, MeerAKTHI) lack interactivity
- Embedding CASA in Jupyter notebooks allows remote interactive pipelines to be created
- Pipelines embedded in Jupyter notebooks are **self-documenting** and fully **repeatable**



Annual data volumes for a number of instruments

Common Astronomy Software Applications package (CASA)

- The de facto standard data reduction package for radio astronomy
- Core libraries split of in separate CASACORE package; LOFAR pipeline is based on CASACORE
- CASA is the standard data reduction package for ALMA, and the VLA.
- SKA pipeline will likely be based on CASA / CASACORE
- Mostly implemented in C++ but contains python bindings to all tasks
- CASA user interface is through customized iPython interpreter

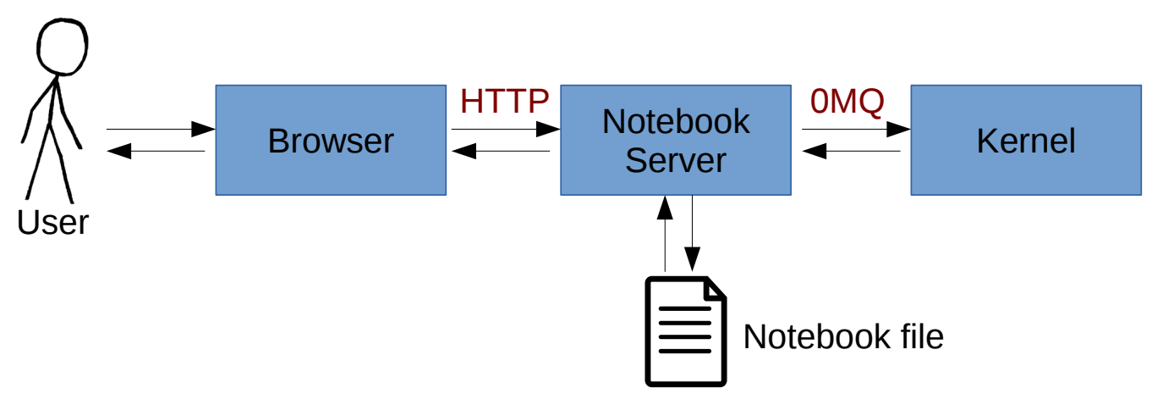


```
IPython 5.1.0 -- An enhanced Interactive Python.

CASA 5.5.0-149   -- Common Astronomy Software Applications

Found an existing telemetry logfile: /home/keimpema/.casa/casastats-550-149-aa4b
bbe10c12b-20191003-140334.log
Telemetry initialized. Telemetry will send anonymized usage statistics to NRAO.
You can disable telemetry by adding the following line to your ~/.casarc file:
EnableTelemetry: False
--> CrashReporter initialized.
Enter doc('start') for help getting started with CASA...
Using matplotlib backend: TkAgg

CASA <1>: listobs(vis='3c391_ctm_mosaic_10s_spw0.ms')
     ...:
Out[1]: True

CASA <2>: plotants(vis='3c391_ctm_mosaic_10s_spw0.ms', figfile='plotants_3c391_a
     ...: ntenna_layout.png')
Number of points being plotted: 26

CASA <3>: 
```

CASA iPython interpreter

Jupyter notebooks

- Multi-language web-based interactive documents which can contain live code, text, and images.
- Successor to the iPython project
- Support for over 40 programming languages
- Python kernel has MATPLOTLIB integration
- Documentation at *https://jupyter.org/*

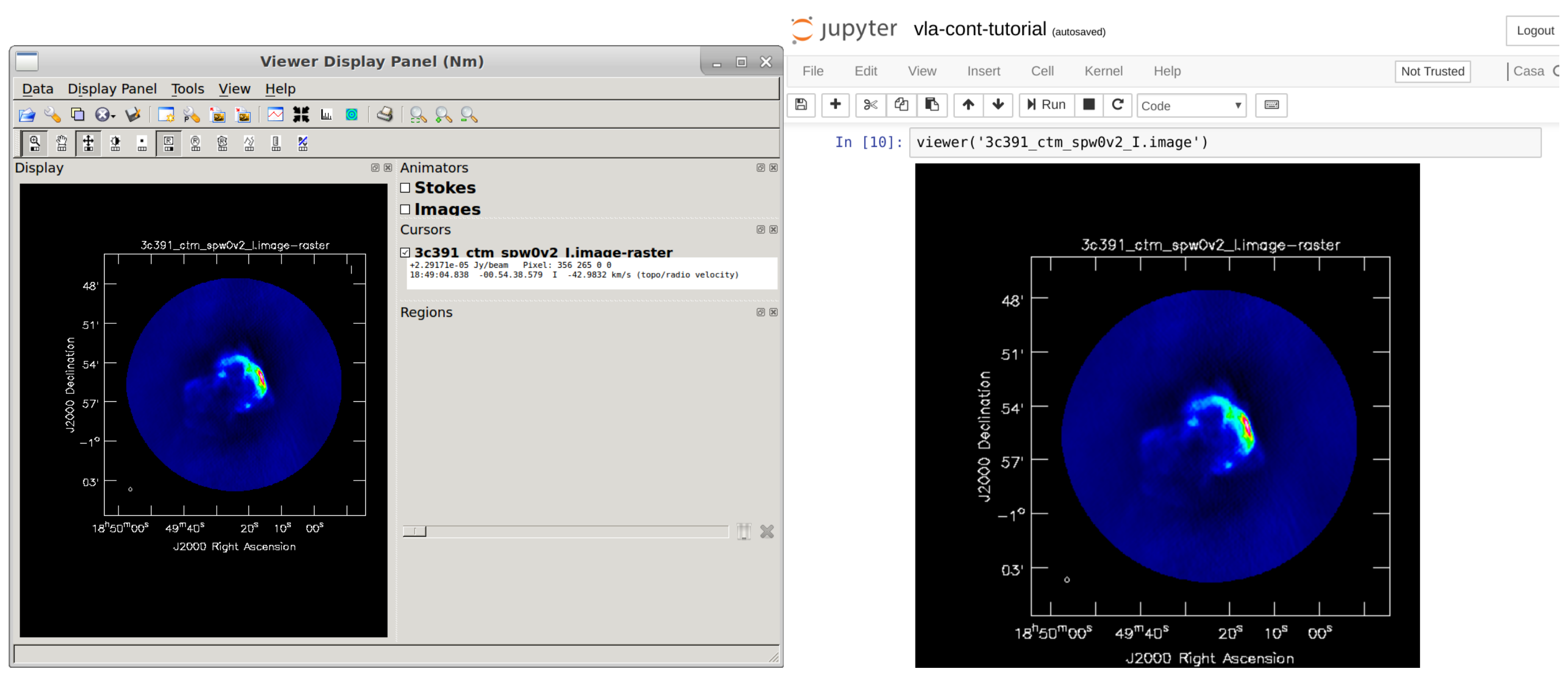


User interacts with notebook server through web interface, all language specifics are contained inside the kernel

Implementation

- CASA tasks are written in C++ but have python bindings
- Many CASA tasks spawn a GUI which is implemented using the Qt widget library
- The Jupyter-CASA kernel is based on Jupyter's python kernel
- CASA GUI tasks are wrapped such that they don't open a GUI but output to a file instead, the kernel embeds the results in the notebook
- Tasks wrappers are implemented as decorators which preserve call signatures and docstring of tasks
- Requires custom build of CASA which is distributes as DOCKER and SINGULARITY images

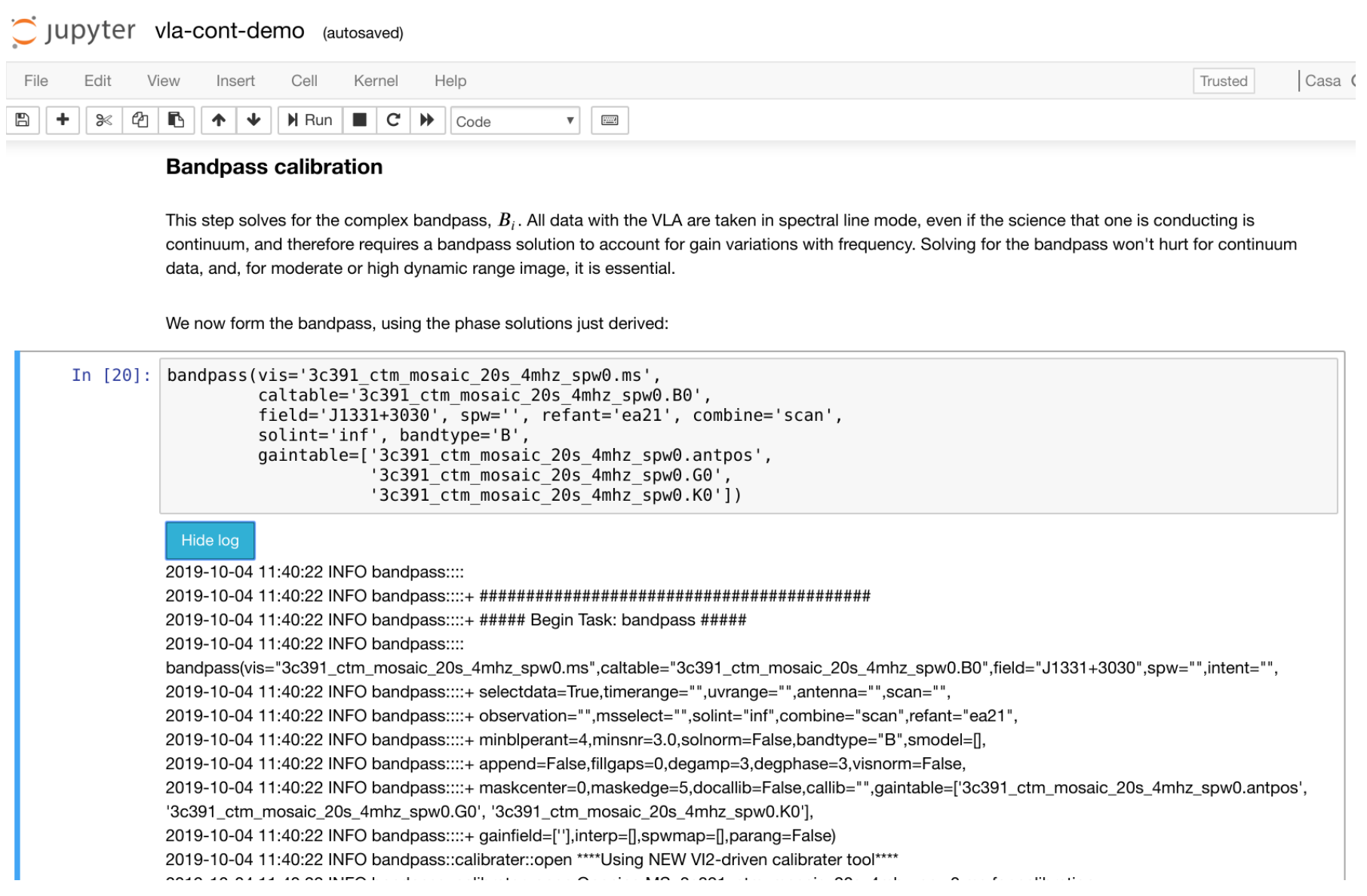
Jupyter CASA kernel feature highlights



```
+2.29171e-05 Jy/beam   Pixel: 356 265 0 0
18:49:04.838  -00.54.38.579  I  -42.9832 km/s (topo/radio velocity)
```

```
In [10]: viewer('3c391_ctm_spw0v2_I.image')
```

Kernel embeds output from GUI programs directly into the notebook



Bandpass calibration

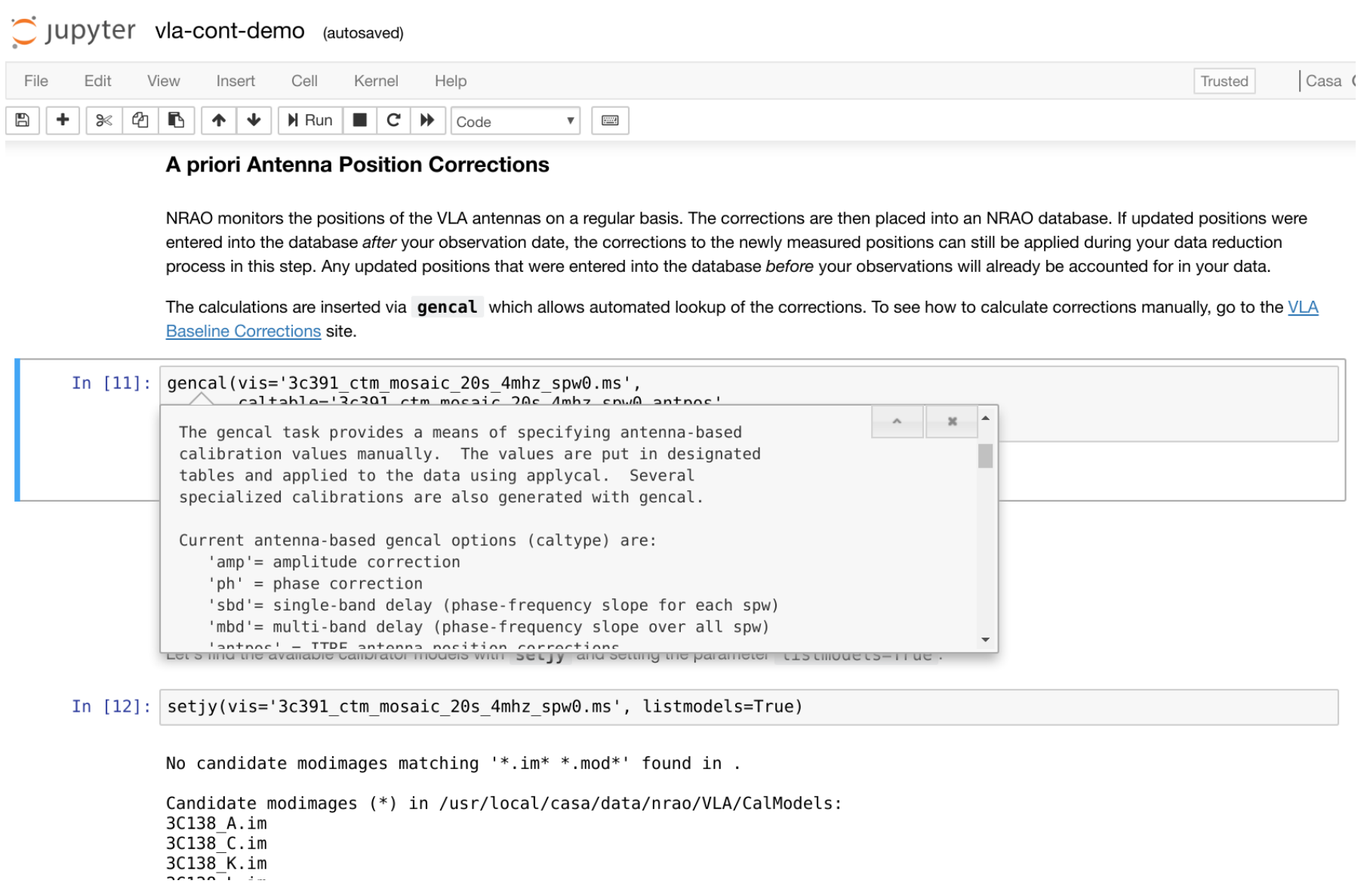
This step solves for the complex bandpass, B_i. All data with the VLA are taken in spectral line mode, even if the science that one is conducting is continuum, and therefore requires a bandpass solution to account for gain variations with frequency. Solving for the bandpass won't hurt for continuum data, and, for moderate or high dynamic range image, it is essential.

We now form the bandpass, using the phase solutions just derived:

```
In [20]: bandpass(vis='3c391_ctm_mosaic_20s_4mhz_spw0.ms',
                  caltable='3c391_ctm_mosaic_20s_4mhz_spw0.B0',
                  field='J1331+3030', spw='', refant='ea21', combine='scan',
                  solint='inf', bandtype='B',
                  gaintable=['3c391_ctm_mosaic_20s_4mhz_spw0.antpos',
                             '3c391_ctm_mosaic_20s_4mhz_spw0.G0',
                             '3c391_ctm_mosaic_20s_4mhz_spw0.K0'])
```

```
2019-10-04 11:40:22 INFO bandpass::::
2019-10-04 11:40:22 INFO bandpass::::+ ##########################################
2019-10-04 11:40:22 INFO bandpass::::+ ##### Begin Task: bandpass #####
2019-10-04 11:40:22 INFO bandpass::::
bandpass(vis="3c391_ctm_mosaic_20s_4mhz_spw0.ms",caltable="3c391_ctm_mosaic_20s_4mhz_spw0.B0",field="J1331+3030",spw="",intent="",
2019-10-04 11:40:22 INFO bandpass::::+ selectdata=True,timerange="",uvrange="",antenna="",scan="",
2019-10-04 11:40:22 INFO bandpass::::+ observation="",msselect="",solint="inf",combine="scan",refant="ea21",
2019-10-04 11:40:22 INFO bandpass::::+ minblperant=4,minsnr=3.0,solnorm=False,bandtype="B",smodel=[],
2019-10-04 11:40:22 INFO bandpass::::+ append=False,fillgaps=0,degamp=3,degphase=3,visnorm=False,
2019-10-04 11:40:22 INFO bandpass::::+ maskcenter=0,maskedge=5,docallib=False,callib="",gaintable=['3c391_ctm_mosaic_20s_4mhz_spw0.antpos',
'3c391_ctm_mosaic_20s_4mhz_spw0.G0', '3c391_ctm_mosaic_20s_4mhz_spw0.K0'],
2019-10-04 11:40:22 INFO bandpass::::+ gainfield=[''],interp=[],spwmap=[],parang=False)
2019-10-04 11:40:22 INFO bandpass::calibrater::open ****Using NEW VI2-driven calibrater tool****
```

Diagnostic output is embedded into the notebook through a toggle button



A priori Antenna Position Corrections

NRAO monitors the positions of the VLA antennas on a regular basis. The corrections are then placed into an NRAO database. If updated positions were entered into the database *after* your observation date, the corrections to the newly measured positions can still be applied during your data reduction process in this step. Any updated positions that were entered into the database *before* your observations will already be accounted for in your data.

The calculations are inserted via `gencal` which allows automated lookup of the corrections. To see how to calculate corrections manually, go to the VLA Baseline Corrections site.

```
In [11]: gencal(vis='3c391_ctm_mosaic_20s_4mhz_spw0.ms',
```

```
The gencal task provides a means of specifying antenna-based
calibration values manually.  The values are put in designated
tables and applied to the data using applycal.  Several
specialized calibrations are also generated with gencal.

Current antenna-based gencal options (caltype) are:
   'amp'= amplitude correction
   'ph' = phase correction
   'sbd'= single-band delay (phase-frequency slope for each spw)
   'mbd'= multi-band delay (phase-frequency slope over all spw)
```

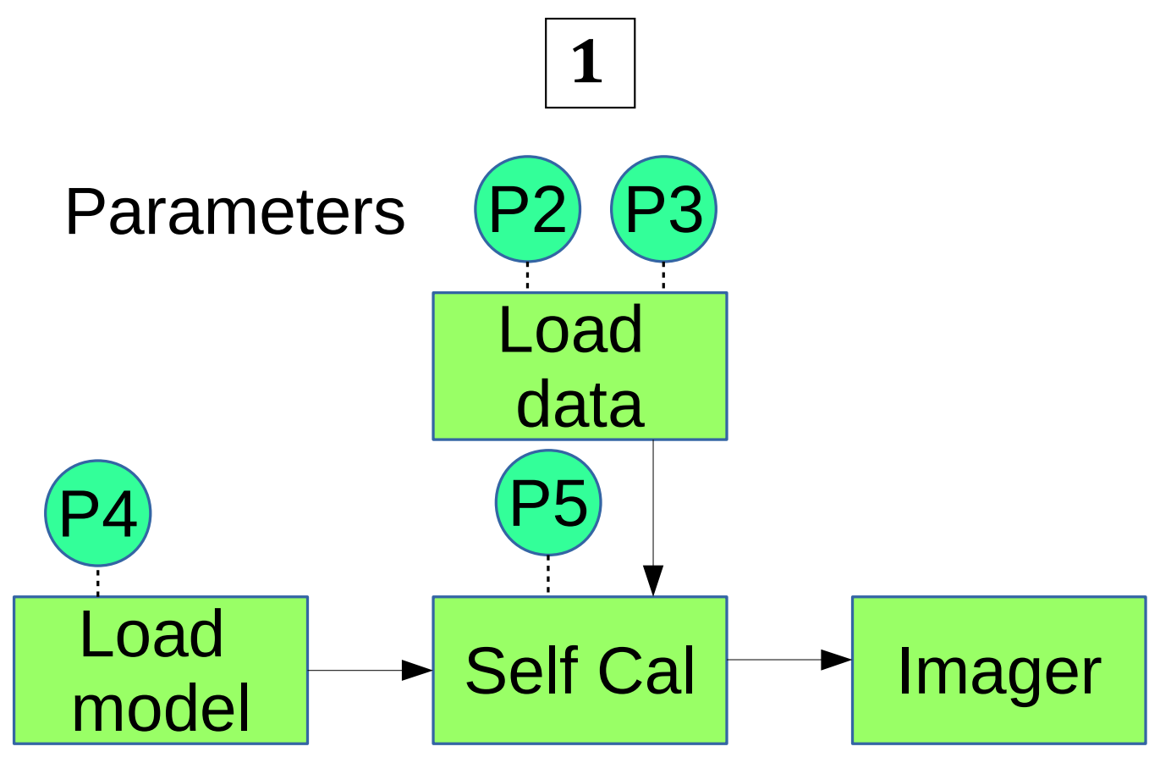
```
In [12]: setjy(vis='3c391_ctm_mosaic_20s_4mhz_spw0.ms', listmodels=True)

No candidate modimages matching '*.im* *.mod*' found in .

Candidate modimages (*) in /usr/local/casa/data/nrao/VLA/CalModels:
3C138_A.im
3C138_C.im
3C138_K.im
```

Integrated help for all CASA tasks

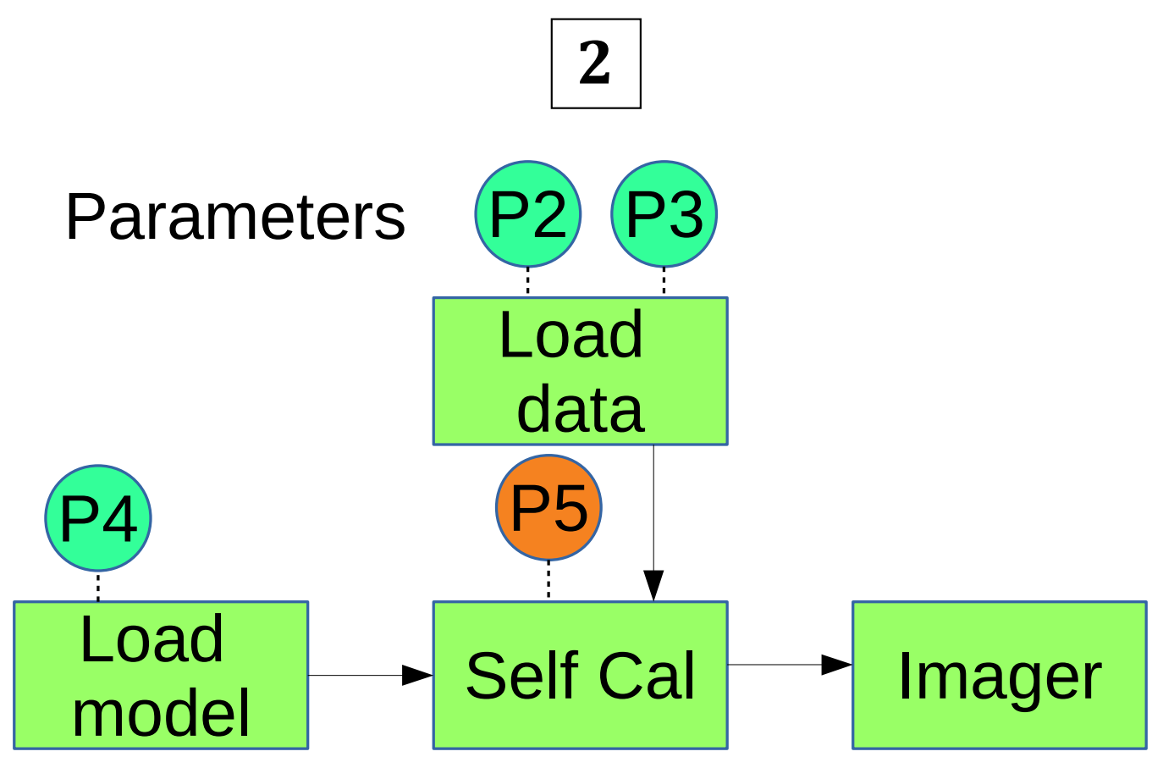
Minimal re-computation framework

- Running pipelines is an *iterative* process
- When inputs to pipeline change often only a subset of tasks needs to be re-executed
- The minimal re-computation framework automates this process by tracking the inputs and dependencies between tasks and only re-executes the tasks which are necessary
- Implemented by efficiently caching intermediate results using ZFS copy-on-write
- Implemented in separate Jupyter-CASA branch

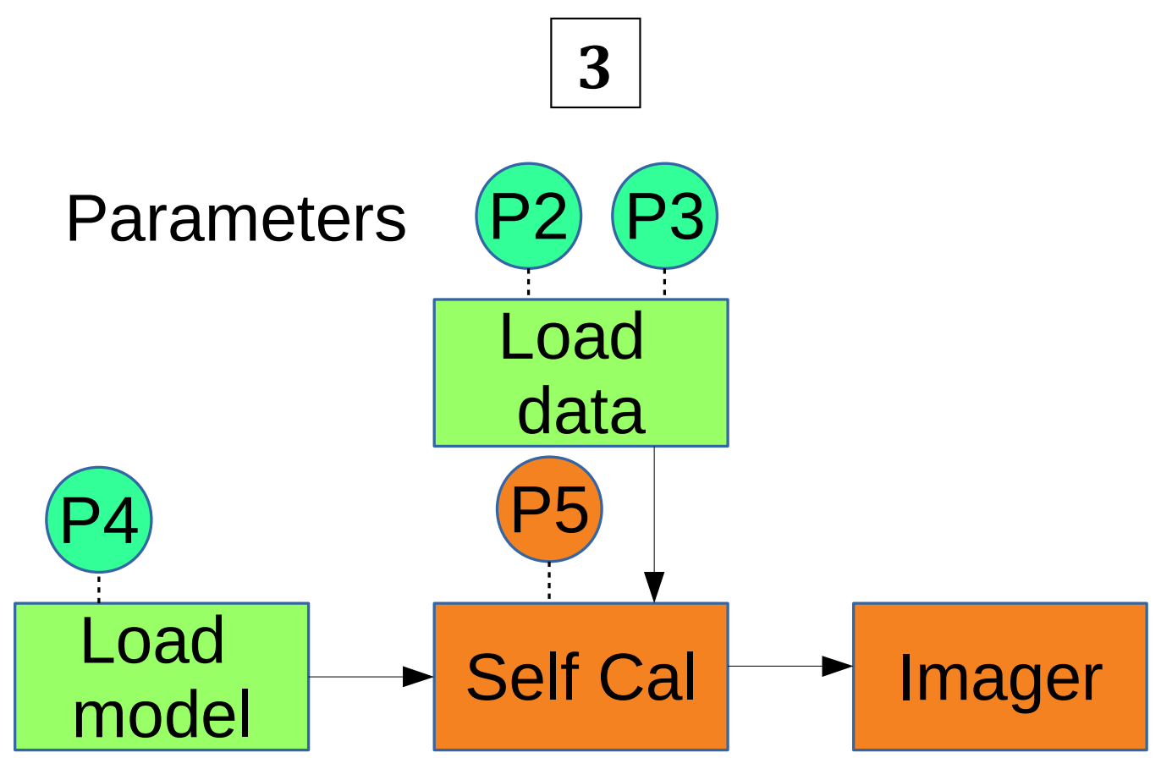
Bojan Nikolic (U. Cambridge), Des Small, and Mark Kettenis (JIVE)
Astronomy and Computing, 25, 133, 2018 (arXiv:1711.06124)



Mock pipeline, consisting of a number of tasks which take a set of parameters P1, P2, ..,



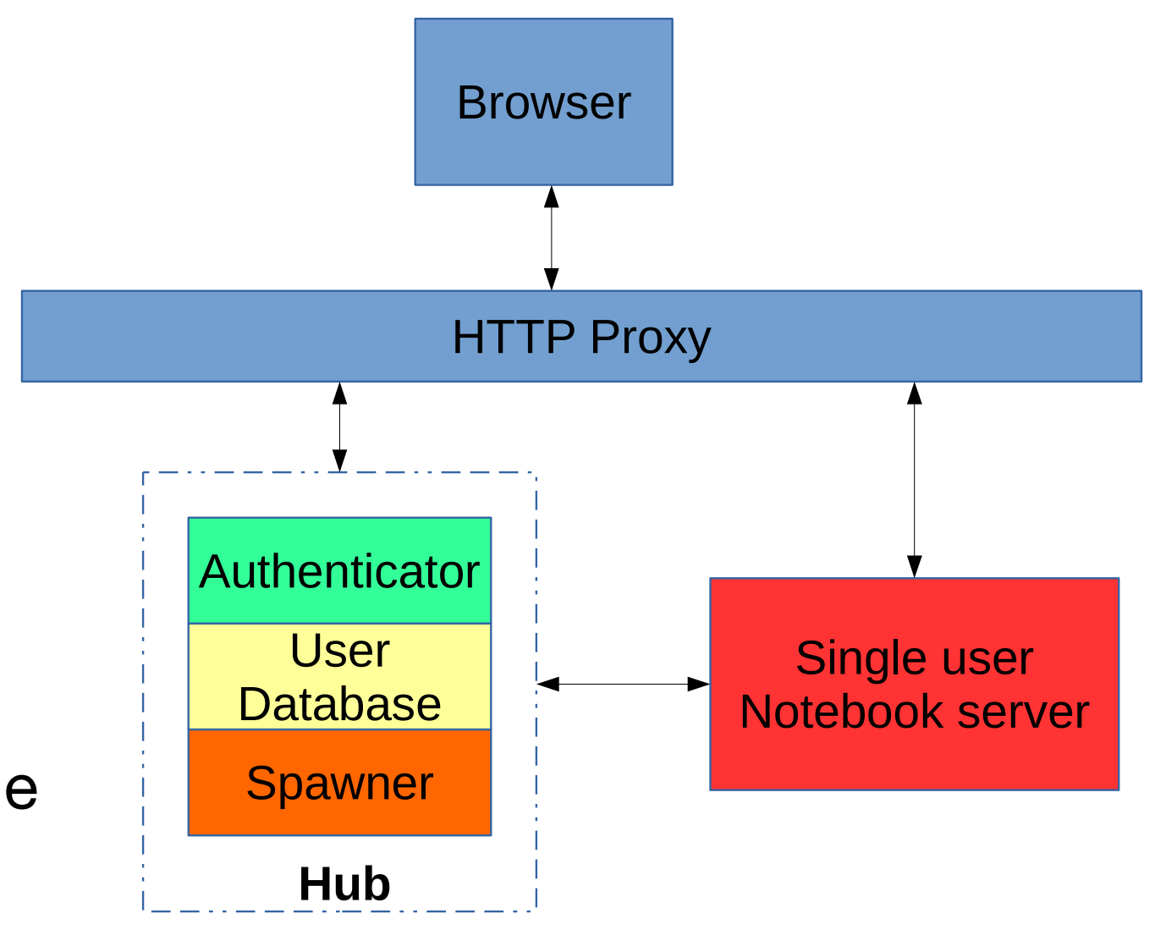
A parameter to the SelfCal task is altered



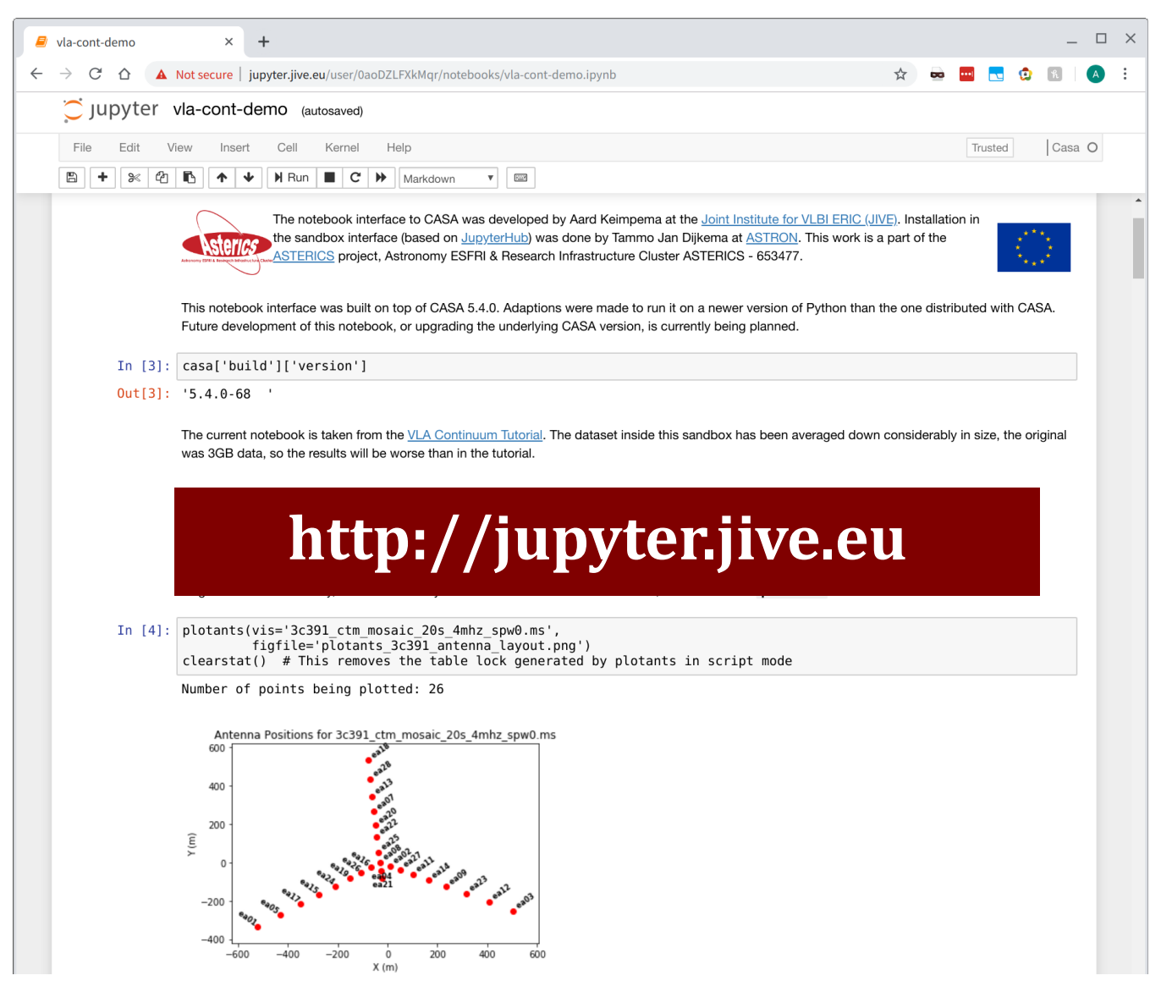
Minimal re-computation framework only re-executes the SelfCal and Imager tasks

Demonstration service

- Open service in which users can run a tutorial CASA notebook on real data.
- Multi-user service implemented using *JupyterHub*.
- Users connect to *http proxy* which then spawns a new DOCKER container for that user
- Both tutorial dataset and Jupyter CASA server are contained in the DOCKER image



JupyterHub architecture



The notebook interface to CASA was developed by Aard Keimpema at the Joint Institute for VLBI ERIC (JIVE). Installation in the sandbox interface (based on JupyterHub) was done by Tammo Jan Dijkema at ASTRON. This work is a part of the ASTERICS project, Astronomy ESFRI & Research Infrastructure Cluster ASTERICS - 653477.

This notebook interface was built on top of CASA 5.4.0. Adaptions were made to run it on a newer version of Python than the one distributed with CASA. Future development of this notebook, or upgrading the underlying CASA version, is currently being planned.

```
In [3]: casa['build']['version']
Out[3]: '5.4.0-68  '
```

The current notebook is taken from the VLA Continuum Tutorial. The dataset inside this sandbox has been averaged down considerably in size, the original was 3GB data, so the results will be worse than in the tutorial.

http://jupyter.jive.eu

```
In [4]: plotants(vis='3c391_ctm_mosaic_20s_4mhz_spw0.ms',
                 figfile='plotants_3c391_antenna_layout.png')
        clearstat()  # This removes the table lock generated by plotants in script mode
        Number of points being plotted: 26
```

Demonstration service

Important links

Jupyter-CASA kernel and documentation
https://github.com/aardk/jupyter-casa

Docker image
docker pull penngwyn/jupytercasa

Singularity image
singularity pull shub://aardk/jupyter-casa:docker

Demonstration service
http://jupyter.jive.eu