Stimela and CARACal

Towards system agnostic, portable and and configurable data reduction pipelines


Stimela

Radio Astronomy Software*

01 | Often written by astronomers, not software developers

02 | Difficult to build/install (Time consuming)

03 | Difficult to install multiple packages on the same system

04 | CASA can do most things. But unnecessarily large, complex and rigid. Does not play well with others.

*views from 2014

What's required of a *good* pipeline

1. **Version control + issue tracking**

   github gitlab **sourceforge** svn

2. **Documentation**

   readthedocs **github** gitlab

3. **Packaging**

   pypi kernsuite **well managed ppa**

4. **Continuous integration**

   travis jenkins **integrity** gitlab **github**

5. **Isolation + portability**

   docker podman/buildah **singularity** udocker

Pipeline/Recipe

A set of tasks connected in a suitable way so as to produce a known result given a set of known input data and parameters. Furthermore, a pipeline must:

1. Be versioned
2. Exit with a non-zero exit status if it fails

Cab (pipeline task)

An application that processes or visualizes data.

1. A Cab takes inputs in a standardized form
2. Applications can be wrapped to conform to a Cab standard
3. Cabs are executed using container technology
    a. Stimela supports **docker, singularity, and podman**

WSClean Cab

```
pipeline.add("cab/wsclean",
             "imager_example_robust_{:d}".format(i),
             {
                 "msname":   MS,
                 "weight":   "briggs {:d}".format(robust),
                 "prefix":   "{:s}_robust-{:d}".format(PREFIX, robust),
                 "npix":   2048,                  # Image size in pixels
                 "scale":   2,                     # Size of each square pixel
                 # Perform 1000 iterarions of clean (Deconvolution)
                 "niter":   1000,
                 "pol" : "I",
                 "multiscale": True,
                 "multiscale-scales" : [0,2],
             },
             label="Imaging MS, robust={:d}".format(robust),
             cpus=2,
             memory_limit="2gb")
```

631 lines (631 sloc) | 25.8 KB Raw

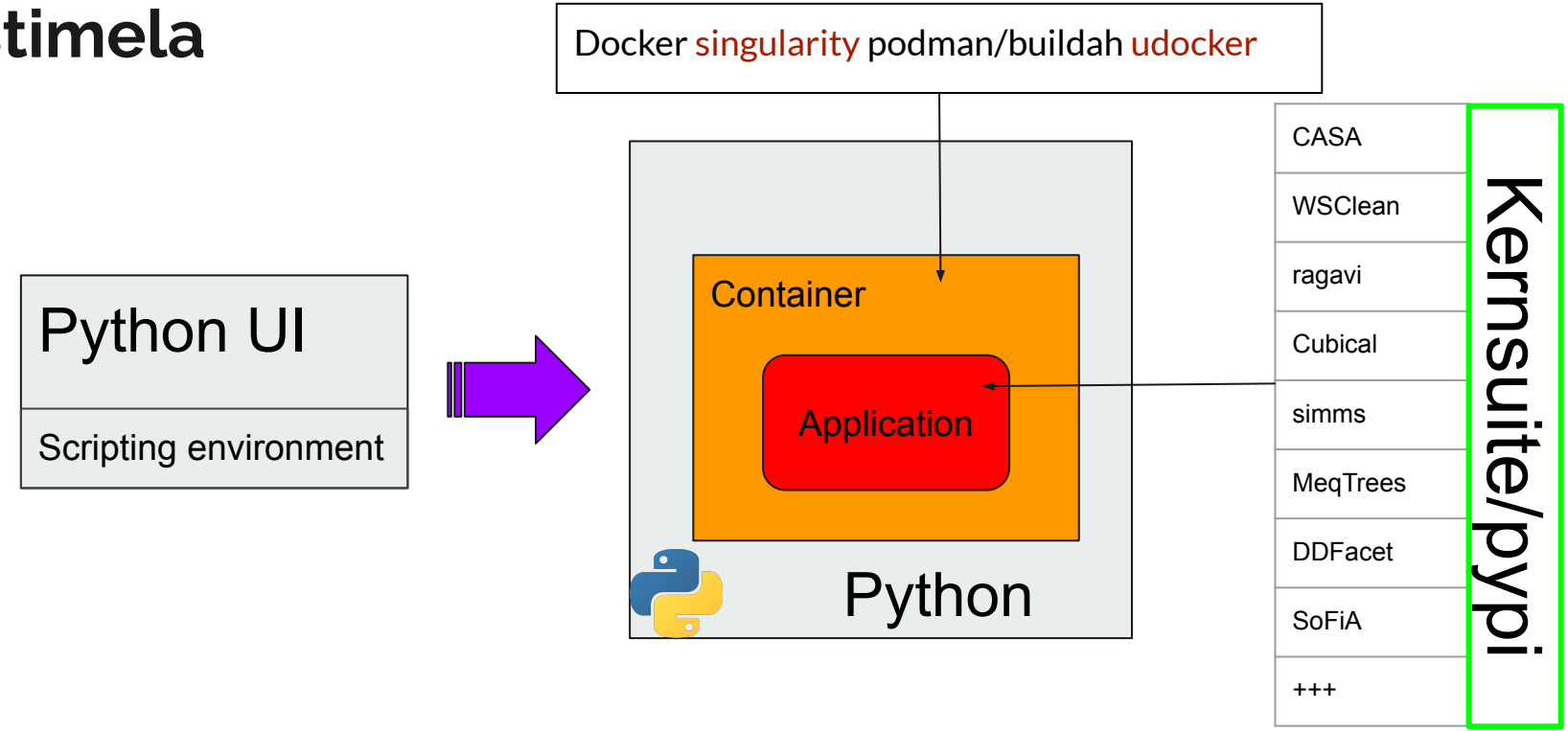

```
{
    "task": "wsclean",
    "base": "stimela/wsclean",
    "tag": ["1.6.0-1", "1.2.3", "1.6.3"],
    "description": "WSClean imaging software",
    "version" : ["2.6", "2.8", "2.9"],
    "prefix": "-",
    "binary": "wsclean",
    "junk" : [],
    "msdir": true,
    "parameters": [
        {
            "info": "MS(s) to be imaged. If multiple mses are specified, they need to be phase-rotated to the same point on the sky",
            "name": "msname",
            "io": "msfile",
            "default": null,
            "dtype": "list:file",
            "required": true
        },
        {
            "info": "Prefix for output products. Default is prefix of MS",
            "name": "prefix",
            "io": "output",
            "default": null,
            "dtype": "file",
            "mapping": "name"
        },
        {
            "info": "Specify number of computing threads to use, i.e., number of cpu cores that will be used. None means use all  cores",
            "dtype": "int",
            "default": null,
            "name": "threads",
            "mapping": "j"
        },
        {
            "info": "Limit memory usage to the given fraction of the total system memory. This is an approximate value.",
            "dtype": "float",
            "default": 100,
            "name": "mem"
        },
```

Python UI

User has access a diverse set of applications via a simple python interface.

```python
import stimela

MS = "sim-meerkat.ms"

pipeline = stimela.Recipe("Simulate point sky", msdir="msdir",
                          indir="inputs", outdir="outputs")

pipeline.add("cab/simms", "makems", {
    "msname": MS,
    "tel": "meerkat",
    "direction": "J2000,0deg,-30deg",
    "freq0": "1.32GHz",
    "dfreq": "1MHz",
    "nchan": 32,
    "dtime": 1,  # in seconds
    "synthesis": 1,  # in hours
}, label="Create an empty meerkat MS")

# the simulator cab is wrapper of the MeqTrees turbo-sim.py script
pipeline.add("cab/simulator", "simsky", {
    "msname": MS,
    "skymodel": "points_sky.lsm.html",
    "addnoise": True,
    "sefd": 541,
    "column": "DATA",
}, label="Simulate point sources into MS")

for i, robust in enumerate([-1, 0, 1]):
    pipeline.add("cab/wsclean", f"image_{i}", {
        "msname": MS,
        "column": "DATA",
        "scale": "1asec",
        "size": 1024,
        "niter": 5000,
        "name": MS[:-3],
        "weight": f"briggs {robust:.2f}",
    }, label=f"Imaging simulation with Briggs robust={robust}")

pipeline.run()
```

stimela


Docker singularity podman/buildah udocker
Python UI
Scripting environment
Container
Application
Python
CASA
WSClean
ragavi
Cubical
simms
MeqTrees
DDFacet
SoFiA
+++
Kernsuite/pypi

Key Takeaways

1. Task definition is standardized to allow a simple and unified UI. Radio interferometry tools essentially become Python functions (even CASA!)

2. User does not need to install any of the applications supported by stimela. The applications are containerized and can be downloaded from an online registry (hub.docker.com).

3. Stimela is not a pipeline, but a framework that allows the creation of pipelines.

4. Container technology guarantees reproducibility and stability, while the simple UI facilitates transparency and debugging.


CARACal

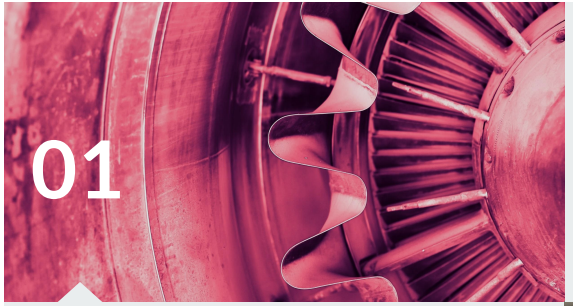
Graduate students

Feb 2019
Busy week

Hogsback,
Eastern Cape,
South Africa

Rhodes University | SARAO | INAF | University of Pretoria | Wits University

What is it?



Configurable, portable and system agnostic. Easy to install

Stimela-based end-to-end data reduction pipeline for RI data. Not limited to MeerKAT data

Can do both continuum and spectral line reductions. Includes full direction-dependent treatment and mosaicing.

How does it work

```yaml
general:
  prefix: mypipelinerun

getdata:
  dataid: ['']
  extension: ms

transform:
  enable: true
  field: calibrators

flag:
  enable: true
  field: calibrators
  label_in: cal
  flag_autocorr:
    enable: true
  flag_spw:
    enable: true
    chans: '*:856~880MHz , *:1658~1800MHz, *:1419.8~1421.3MHz'
    ensure_valid: false
  flag_mask:
    enable: true
    mask: meerkat.rfimask.npy
    uvrange: '0~1000'
  flag_rfi:
    enable: true
    flagger: aoflagger
    aoflagger:
      strategy: firstpass_QUV.rfis

crosscal:
  enable: true
  label_in: 'cal'
  set_model:
    enable: true
```

User Interface (YAML)

Backend

- crosscal_worker.py
- ddcal_worker.py
- flag_worker.py
- getdata_worker.py
- inspect_worker.py
- line_worker.py
- mask_worker.py
- mosaic_worker.py
- obsconf_worker.py
- prep_worker.py
- selfcal_worker.py
- transform_worker.py

workers = stimela recipes

Data Products

Gain tables and plots

RAW visibility plots

Continuum images

Spectral-line cubes

Detailed and summary log files

Jupyter notebook (or HTML) reports

Calibrated visibilities and plots

Source catalogs

Over to you, Josh.

https://caracal.readthedocs.io https://github.com/caracal-pipeline/caracal

https://github.com/ratt-ru/Stimela

https://github.com/ratt-ru/scabha