URSI GASS 2021, Rome, Italy, 28 August - 4 September 2021

The IDIA PROCESSMEERKAT pipeline: Fast CASA Processing on a Cloud-Based HPC Cluster

Jordan D. Collier*(1)(2), Bradley Frank(1)(3)(4), Srikrishna Sekhar(1)(5) and Andrew R. Taylor(1)(6)
(1) The Inter-University Institute for Data Intensive Astronomy (IDIA), Department of Astronomy, University of Cape Town, Private Bag X3, Rondebosch, 7701, South Africa
(2) Western Sydney University, Locked Bag 1797, Penrith, NSW 2751, Australia
(3) South African Radio Astronomical Observatory (SARAO), 2 Fir Street, Observatory, 7925, South Africa
(4) Department of Astronomy, University of Cape Town, Private Bag X3, Rondebosch 7701, South Africa
(5) National Radio Astronomy Observatory, 1003 Lopezville Road, Socorro, NM 87801, USA
(6) The Inter-University Institute for Data Intensive Astronomy (IDIA), Department of Astronomy, University of the Western Cape, Bellville, South Africa

Abstract

The IDIA PROCESSMEERKAT pipeline is an efficient, user-friendly pipeline that has been widely tested and documented. It has been used to successfully calibrate and image MeerKAT data down to an RMS of several µJy / beam, and supports continuum, polarisation and spectral-line use cases. It runs on the ilifu cluster, making dynamic use of resources and containers, and presents a good framework for implementing pipelines in a HPC environment. The current version performs full-polarisation *a priori* calibration and produces quick-look images. The next version, which introduces self-calibration, is currently being tested.

1 Introduction

The IDIA PROCESSMEERKAT pipeline is a fully automated end-to-end, full polarisation pipeline that is efficient, flexible, scalable, and user-friendly, having been widely tested and documented[1]. It is a publicly available[2] parallelised software package for high performance processing of radio interferometric data, developed at the Inter-University Institute for Data Intensive Astronomy (IDIA). The pipeline is primarily implemented for ilifu, a Tier 2 Data Intensive Research Facility that constitutes a joint cloud platform for Astronomy and Bioinformatics. Ilifu uses OpenStack to deploy Ubuntu VMs of various flavours for the user community, according to their computational requirements, while the software is made available via custom-built Singularity containers. The ilifu SLURM cluster currently consists of more than ∼90 worker nodes, each with 256 GB of RAM and 32 CPUs, with a shared storage system of ∼6 PB (usable), representing a pathfinder science regional data centre, and a good framework for solving many of the broader challenges of the SKA.

The pipeline is designed to calibrate and image data from the MeerKAT telescope [1], South Africa's precursor to the Square Kilometre Array (SKA). MeerKAT is an interferometer with 64 13.5 m-diameter dishes that operates from 580–3500 MHz over a 856 MHz bandwidth, with a typical resolution of ∼8 arcsec over its longest baseline of 8 km, and typical sensitivity over 8 hours of a few µJy/beam. The pipeline's use cases include continuum, polarisation, and spectral line data products, produced for Large Survey Projects (LSPs[3]) such as the MeerKAT International GHz Tiered Extragalactic Exploration (MIGHTEE [2]) survey and the Looking At the Distant Universe with the MeerKAT Array (LADUMA [3]) survey. Although the pipeline is primarily designed for MeerKAT data on the ilifu cluster, its design is flexible, in order to support processing data from different radio telescopes, and with different HPC systems. Currently, it has successfully processed MeerKAT, Giant Metrewave Radio Telescope (GMRT) and Karl G. Jansky Very Large Array (JVLA) data, and has been adapted to different environments with minor changes, including SLURM clusters in the USA and Australia, and the Distributed Research using Advanced Computing (DiRAC) facility, which uses IRIS grid infrastructure.

2 Design

Currently, the pipeline performs full-polarisation *a priori* calibration of MeerKAT observations, which are typically several TB in size. As part of its off-the-shelf design, it uses the Common Astronomy Software Applications (CASA) version 6 library within a purpose-built container. In order to parallelise over the cluster, we use a combination of partitioning the data into smaller chunks both in time and frequency. To partition the data along the time axis, we use CASA's internal Multi-MeasurementSet (MMS) format, which splits the data into sub-MSs, each a single scan of an observed field. Each sub-MS is handled by a single message passing interface (MPI) worker during processing by CASAMPI. We further partition the data along the frequency axis into spectral windows (SPWs), each of which

[1] https://idia-pipelines.github.io/docs/processMeerKAT
[2] https://github.com/idia-astro/pipelines
[3] http://public.ska.ac.za/meerkat/meerkat-large-survey-projects

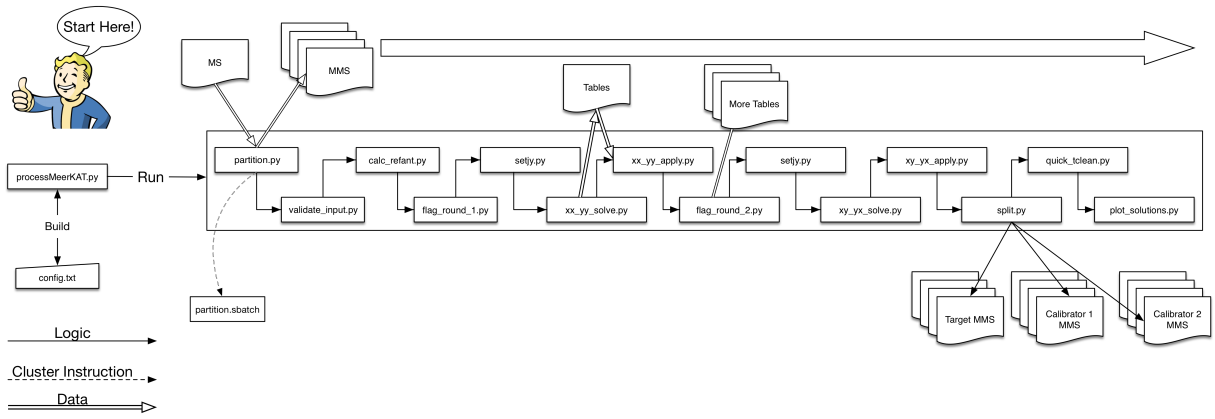

Figure 1. A flowchart showing the steps undertaken when running the cross-calibration routine within the IDIA PROCESSMEERKAT pipeline. After the partition step, each SPW follows these steps by default.

produces an MMS and launches an independent, concurrent instance of the pipeline. This not only decreases the run-time of the pipeline, but also produces a more reliable wideband continuum and polarisation calibration, by separately solving for each SPW.

The first step of the pipeline is a quick interactive 'build' step that reads the input MeasurementSet (MS) and extracts information such as calibrator and target field IDs, and the number of scans, which allows parameters for the field selection and resource usage to be set within a configuration file. This file contains configurable parameters, with sensible defaults that are suitable for the majority of use cases. This allows the user to easily configure the pipeline, such as selecting a SPW (frequency) range to process, a reference antenna, time and frequency averaging parameters, and frequency ranges to flag. Another config section allows the user to input custom scripts to run, different containers and MPI wrappers, and SLURM resource parameters, all of which can also be passed directly into the PROCESSMEERKAT script itself. Once the configuration file is built, the user then performs the automated 'run' step, which produces a set of SLURM SBATCH scripts with dynamic resource allocations that are submitted to the SLURM queue to run the pipeline.

We split the pipeline into discrete steps, each a logical unit within our calibration scheme that supports MPI, or a single CPU, according to CASA's limitations. Each step is a different pipeline script, or a user's own script, run within default or user-specified container, which is contained within an SBATCH file that calls the script with either a single CPU, or with the resources requested by the user. Figure 1 shows the steps undertaken when running the default set of scripts. The pipeline provides additional bash utility scripts to interact with the pipeline by presenting a summary of all of the jobs, a script to kill all jobs, a script to find any errors, and to display the run times.

3 Algorithm

The algorithmic details of the various CASA tasks used within a single instance of the pipeline (e.g. one SPW) are presented within its documentation[4], which we summarise here. The pipeline begins by converting the input MS into several MMSs over multiple SPWs. Within each SPW, the data is partitioned in N sub-MSs using MSTRANSFORM, where N is the total number of scans over all fields, each of which is handled by an MPI worker. After validating the input data and pipeline parameters, antenna statistics are then optionally calculated, in order to identify a reference antenna, and to potentially flag out bad antennas. Following this, one round of pre-calibration flagging is performed, to remove the worst RFI. This is done using a static mask set in the config file, with the default set to cover frequency ranges known to have persistent RFI, and then a combination of the 'clip', 'extend', and 'tfcrop' modes within FLAGDATA.

After this, parallel-hand calibration is performed, in order to obtain better statistics for a second round of flagging, following a standard routine of solving for delay, bandpass and gain calibration, and bootstrapping from the flux calibrator to all fields. The second round of flagging is then performed as before, but using tighter thresholds and also including the 'rflag' mode, in an attempt to flag out all RFI.

After this, the user can optionally perform full-Stokes cross-hand calibration, or otherwise a second round of the previous calibration routine. Under the former recipe, a new set of bandpass, delay and parallel-hand gains are computed. Using `gaintype='T'`, the phase calibrator can be used to solve for a polarisation-independent gain. The frequency-dependent leakages are solved for using the unpolarised primary calibrator, and the XY-phase using the polarisation calibrator.

[4]https://idia-pipelines.github.io/docs/processMeerKAT/calibration-in-processmeerkat/

Following this, the corrected data for the target field and any other fields specified by the user are split out for further processing. This includes options for file format, and averaging in time and frequency, but by default, no averaging is performed, and the file is an MMS. A quick-look image is produced from TCLEAN without using self-calibration, W-projection, or thresholding. After this, various plots are written summarising the calibration solutions and the potential presence of RFI.

Lastly, if more than one SPW is processed, the pipeline will perform a concatenation, which produces a quick-look continuum cube over all processed SPWs, for the target field and any other fields specified by the user. Furthermore, concatenated MSs or MMSs over all SPWs are written for each of those fields, for further processing, such as self-calibration and science imaging. An example quick-look image as output from the pipeline is shown in Figure 2.

4 Benchmarking

As mentioned in section 2, parallelism is achieved by the pipeline through two schemes: 1) the internal CASA mechanism to convert an MS into an MMS by partitioning into scans (in time), handled by CASAMPI; 2) the pipeline mechanism to partition an MS into multiple independent SPWs (frequencies) that are concurrently processed.

We performed benchmarking of the pipeline under a combination of these different parallelism schemes. To do this, we processed a typical MeerKAT UHF-band (544–1088 MHz) 32k data set, observed over 9.0 hours. We tested both 32k data at the full native spectral resolution of 32,768 channels, with only XX and YY correlations present (6.0 TB raw, inflated to ∼16 TB during processing), and 1k data pre-averaged to 1024 channels, with all correlations present (353 GB raw, inflated to ∼960 GB during processing). When using parallelism scheme 1, we selected 16 tasks over 3 nodes (48 tasks/CPUs) and 32 tasks/CPUs on 1 node, respectively for the 32k and 1k data. When using parallelism scheme 2, we selected 8 SPWs over ∼470 MHz (free of RFI), each processed concurrently.

Figure 3 shows the cumulative wall-time over the different steps, from the initial partition of raw data, through to splitting out the calibrated fields, given by cross-calibration runs using a combination of parallelism schemes. The top and bottom panels show the wall-times for the 32k and 1k data, respectively. The blue, orange, green and red lines show pipeline runs that used no parallelism, and parallelism schemes 1 (MMS), 2 (SPWs), and both (MMS + SPWs), respectively. The former two are omitted from the top panel due to their extreme wall-times (>100 h). The solid line and shaded grey region shows the median and robust standard deviation, derived over all the SPWs. Using both parallelism schemes for both 32k and 1k data achieves a median run-time of ∼0.5 hours and ∼5 hours, respectively, with very little deviation over each SPW.

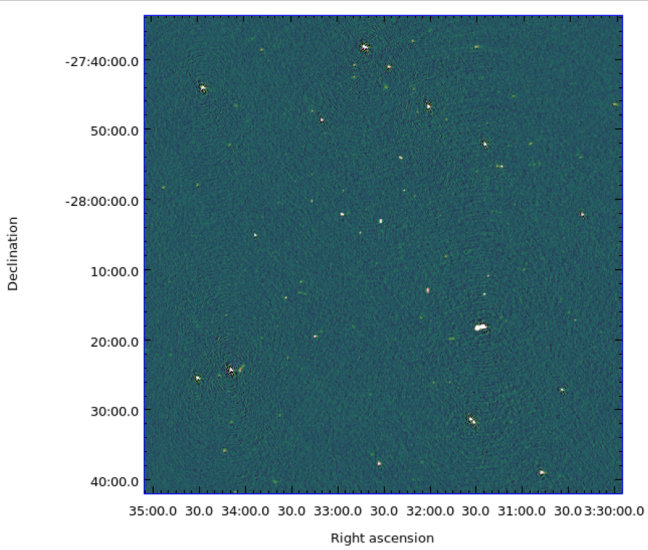

Figure 2. A selected quick-look image cross-calibrated by V1.0 the pipeline. This particular image is of the CDFS field, observed over 8 hours, reaching ∼80 µJy / beam over a bandwidth of 10 MHz.

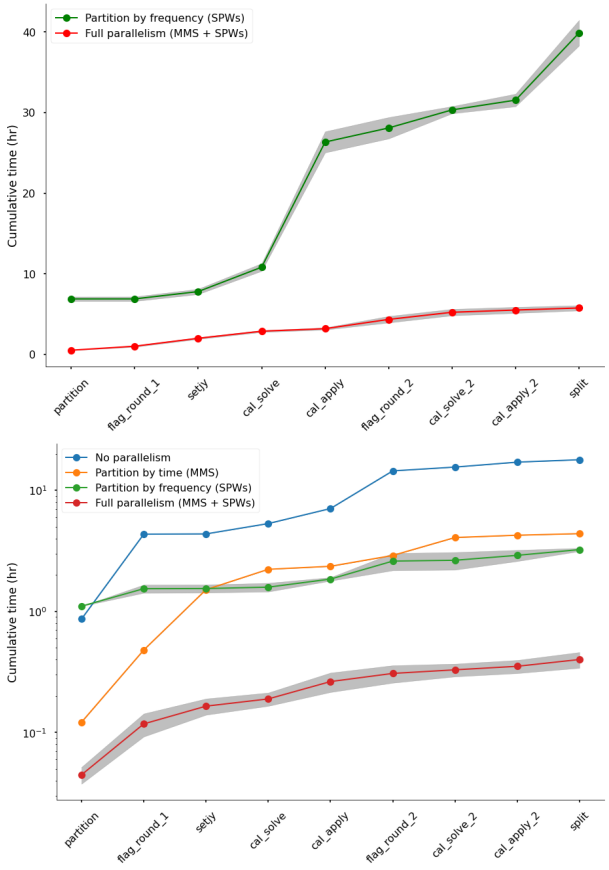

Figure 3. The cumulative wall-times over the different cross-calibration steps for IDIA pipeline runs to process a 9.0 hour MeerKAT UHF-band observation at 32k (6.0 TB raw - top panel), and 1k (353 GB raw - bottom panel), according to the labelled parallelism schemes discussed in Section 4. The solid line and greyed area show the median wall-time and robust standard deviation over all SPWs.

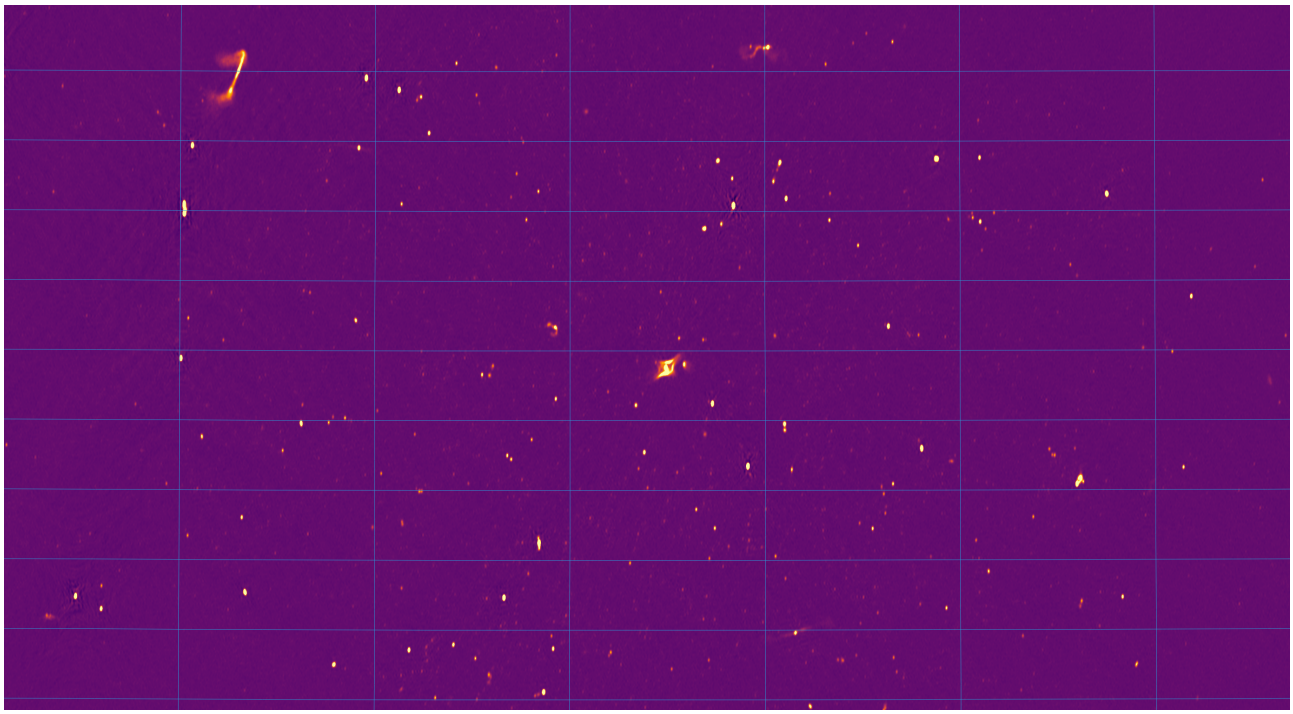

Figure 4. A selected image cross-calibrated with the IDIA pipeline, and then self-calibrated and imaged by the pipeline according to the algorithm summarised in Section 5. These data were observed over 5.5 hours, and reach ∼7 µJy / beam over a bandwidth of 100 MHz. This image was output from the Cube Analysis and Rendering Tool for Astronomy (CARTA).

5 Future Development

We are currently testing the next version of the pipeline, V2.0, in which we introduce self-calibration. During self-calibration, various imaging parameters, including the clean depth, image resolution and size, gridder and deconvolver options, are fully configurable by the user. By default, our self-calibration routine performs an initial TCLEAN using W-projection with a high threshold of the estimated RMS, after which source finding is performed by PyBDSF and used to produce a mask at the island boundaries. This mask is used to perform a deeper round of TCLEAN, which writes a model column to the data, used to solve for and apply the gains, and to optionally flag on the residual. This process is repeated for each loop of self-cal, with the solution intervals and total number of loops being specified by the user. Figure 4 shows an example image that has been cross-calibarated and self-calibrated with the IDIA pipeline according to this algorithm.

After this, the pipeline will introduce AW-projection, which is currently being implemented in TCLEAN in conjunction with the NRAO CASA Algorithm Research and Development Group (ARDG) group. This algorithm is implemented for any telescope that has performed holography, and will be used within our pipeline to accurately calibrate the off-axis leakages, as well as the frequency- and direction-dependent primary beam.

References

[1] T. Mauch, W. D. Cotton, J. J. Condon, and et al., "The 1.28 GHz MeerKAT DEEP2 Image,", vol. 888, no. 2, 61, p. 61, 2020. DOI: 10.3847/1538-4357/ab5d2d. arXiv: 1912.06212 [astro-ph.GA].

[2] A. R. Taylor and M. Jarvis, "MIGHTEE: The MeerKAT International GHz Tiered Extragalactic Exploration," in *Materials Science and Engineering Conference Series*, ser. Materials Science and Engineering Conference Series, vol. 198, 2017, p. 012 014. DOI: 10.1088/1757-899X/198/1/012014.

[3] B. W. Holwerda, S. L. Blyth, and A. J. Baker, "Looking at the distant universe with the MeerKAT Array (LADUMA)," in *The Spectral Energy Distribution of Galaxies - SED 2011*, R. J. Tuffs and C. C. Popescu, Eds., ser. IAU Symposium, vol. 284, 2012, pp. 496–499. DOI: 10.1017/S1743921312009702. arXiv: 1109.5605 [astro-ph.CO].