URSI AP-RASC, Sydney, Australia, 17 – 22 August 2025

Improving scalability and computational efficiency of self-calibration and imaging pipelines for LOFAR and SKA

Stefan J. Wijnholds*(1), Tammo Jan Dijkema(1), Herman G.J. Groot(2), Maik Nijhuis(4), André R. Offringa(1), Chiara Salvoni(3), Nicolas Slusarenko(2), and Sebastiaan van der Tol(1)
(1) ASTRON, Dwingeloo, The Netherlands;
e-mail: wijnholds@astron.nl, dijkema@astron.nl, offringa@astron.nl, tol@astron.nl
(2) S[&]T, Delft, The Netherlands; e-mail: herman.groot@stcorp.nl, nicolas.slusarenko@stcorp.nl
(3) CGI, Rotterdam, The Netherlands; e-mail: chiara.salvoni@cgi.com
(4) TriOpSys, Utrecht, The Netherlands; e-mail: maik.nijhuis@triopsys.nl

Abstract

Modern self-calibration pipelines can produce high-quality radio interferometric imaging results at the cost of significant computational effort. The computational cost is expected to increase even further with the data volumes produced by the Square Kilometre Array (SKA). We are developing, improving and maintaining a code base for self-calibration and imaging. To identify performance improvements, we perform benchmarking on a quarterly cadence, In this contribution, we present the becnhmarking results from February 2024 and February 2025 and discuss the improvements made to achieve that progress as well as our plans for the near future.

1 Introduction

Self-calibration and imaging codes have matured to a level that enables high-quality imaging results but need significant computing resources. A prime example of this is the full field-of-view image reconstructed from an observation with the international Low Frequency Array (LOFAR, [1]) telescope at the cost of 250,000 core hours [2]. As data volumes will only increase in the future with the construction of the Square Kilometre Array (SKA) Low [3] and Mid [4] telescopes, scalability and computational efficiency need to improve significantly to reduce time to science and make future radio observations financially and environmentally sustainable.

With our team, we are developing, improving and maintaining a code base for self-calibration and imaging with LOFAR and SKA. We benchmark our self-calibration pipeline on a quarterly cadence to monitor our progress in terms of computational efficiency and scalability and to identify bottlenecks and further improvements. In this paper, we present benchmarking results from February 2024, discuss what we learned from them and which improvements we have made. The effect of these improvements is demonstrated by showing our latest benchmarking results (February 2025). We conclude by sketching our next steps.

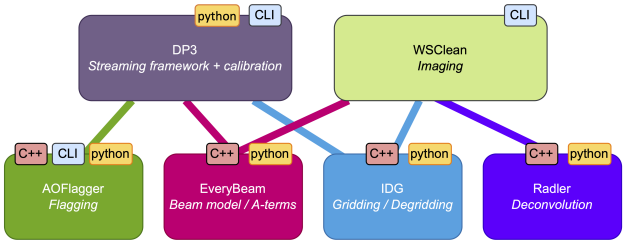

Figure 1. Overview of components in our software stack with their dependencies. The boxes on the top of each component indicate the available interfaces: command-line interface (CLI), C++ or Python [5].

2 Code base

An overview of our current software stack is shown in Fig. 1 [5]. The main components are the Default Pre-Processing Pipeline (DP3) and WSClean that use four other components, which can also be used independently. DP3 [6] can perform various operations, such as flagging (a.o. using the AOFlagger [7]) and direction-(in)dependent calibration, on the data in a pipelined way, so that the data are read and written only once. These operations are implemented as modules referred to as DPSteps, which can be chained to form a data processing pipeline. DP3 is written in C++ as a stand-alone program with a command-line interface (CLI). Python bindings have been added to allow integration of DPSteps in Python-based workflows.

WSClean [8, 9] is a generic wide-field imager for radio astronomy which supports multi-scale wideband deconvolution. It supports W-Stacking as well as Image Domain Gridding (IDG) [10] for imaging. The resulting dirty images can be deconvolved using various deconvolution methods available in the Radio Astronomical Deconvolution Library (Radler), including (wideband multi-scale) CLEAN [11, 12]. WSClean uses Radler's C++ interface by linking to Radler as shared library. The deconvolution methods in Radler are accessible in Python-based workflows via Python bindings.

Table 1. Pipeline performance on a single DAS-6 node.

cycle	calibrate	predict	image
1	00:21:13	00:04:26	05:06:30
2	01:05:36		04:56:28
3	03:35:50	00:03:59	04:36:13

Both DP3 and WSClean need a model of the direction-dependent instrument response in some of their processing stages. This is provided by EveryBeam, which can calculate the instrument response over the field-of-view for both interferometer arrays of dishes and phased array telescopes. DP3 and WSClean link to EveryBeam as a shared library, but other users can also call its functionality from Python.

3 February 2024 benchmarking results

For benchmarking, we used data from a LOFAR HBA observation covering 137.8 – 149.5 MHz on a field flanking the North Celestial Pole. The data was reduced on the DAS-6 research cluster, initially on a single node with two Intel Xeon E5-2660 CPUs with 40 cores running at 2.60 GHz and 256 GB RAM. To demonstrate scaling, later benchmarks were run on three such DAS-6 nodes.

Table 1 shows the wall clock time to run three full major cycles on 20% of the data on a single DAS-6 node as measured in February 2024. Each self-calibration cycle (except the second) consists of three stages: direction-dependent calibration, prediction and subtraction of bright sources outside the field-of-view and imaging (including deconvolution). Despite the fact that only three self-calibration cycles were run on just 20% of the data, the pipeline already requires about 20 hours to complete. Experience showed that adding a few additional major cycles including a final cycle using 100% of the data would take a factor 3.7 more time. Our goal for the year was thus to make scalability and computational efficiency improvements, so that a full pipeline would complete in a reasonable amount of time.

One should note that the time to make an image is quite consistent across the self-calibration cycles while the time taken by calibration is increasing from one cycle to the next due to the increasing calibration model complexity. This results in more source components to predict model visibilities for and more directions to solve for direction-dependent gain solutions. Performance improvements in calibration were therefore important despite these results suggesting otherwise.

4 Pipeline improvements

4.1 Parallelisation

During calibration of low-frequency observations, it is beneficial to use large solution intervals in frequency to constrain the solutions using clock and Total Electron Content (TEC) fitting. Temporally, there is no a priori relationship between gain solutions that are several hours apart. Therefore, parallelism in calibration can be achieved by chunking the observation in time, having different calibration processes work on different time chunks. We realised this by defining a set of DP3 processes, each performing calibration on a different time chunk of data, and having that work distributed by Dask.

For imaging, it natural to split the data into frequency chunks linked to coarse frequency channels. Even in continuum imaging, coarse frequency channels are used to capture the smooth spectral variations of the continuum sources and derive their spectral indices. In this case, we decided to orchestrate parallelisation across multiple nodes by WSClean itself using the Message Passing Interface (MPI) for communication between nodes. This enables us to incorporate domain-specific knowledge into the orchestration. For example, the complete imaging problem involves a gridding task per facet (patch of the image) and per frequency chunk. The simplest way to orchestrate these gridding tasks is to define a list of them and have the available compute nodes work through them in a round robin fashion. This will result in a situation in which data for the same frequency chunk is read multiple times for multiple facets, which may cause an I/O bottleneck. In our scheduler, we tried to avoid such situation by mapping tasks related to the same frequency chunk to the same node. This allowed us to work with local data locks instead of central data locks. It also paves the way to store the data for a specific frequency chunk on a specific node potentially reducing I/O between central storage and compute nodes significantly. This approach also allows further optimisations, such as reading data once for multiple facets.

4.2 Performance improvements

Reuse of model visibilities
In some cycles, phase and amplitude calibration are done in two separate stages to enable the use of different solution intervals. Each calibration stage triggered prediction of model visibilities. However, if the model is not updated between the two stages, exactly the same complex values are calculated twice. To avoid such unnecessary calculations, we added the option to buffer the model visibilities when they can be reused in the next calibration stage.

Image-based predict
One of the primary cost drivers in the later self-calibration cycles is visibility prediction for calibration. Our pipeline currently uses direct predict, predicting visibilities for each individual source component. This is expensive when the number of source components becomes very large. We therefore experimented with image-based predict using the w-gridder. Due to the overhead of doing a large fast Fourier transform (FFT), this turned out to be beneficial only when predicting visibilities for a large time interval at once. For our scenario (Dutch LOFAR array, up to about 50 cali-

Table 2. Pipeline performance on three DAS-6 nodes.

cycle	calibrate	predict	image
1	00:11:40	00:02:40	04:35:29
2	00:17:56		04:50:00
3	01:42:26	00:06:29	04:49:58
4	00:23:45	00:03:18	04:45:01
5	00:58:41	00:05:45	04:56:23
6	00:13:24	00:01:15	05:31:50
7	02:45:58	00:53:43	06:30:59

bration directions), image-based gridding turned out to be beneficial only when the source model had over 40,000 source components, which may only happen in the final self-calibration cycles. We therefore made image-based predict optional for each individual self-calibration cycle. The larger number of stations in SKA-LOW may shift the balance. Finally, efforts are underway to implement w-towers [13] in WSClean, which may also shift the balance.

Reducing number of beam evaluations
In principle, the station beam of a phased array station changes continuously while tracking the targeted field. In practice, this variation is sufficiently slow to assume a constant model over a reasonably short period of time, e.g., 2 minutes. This can be exploited to reduce the number of beam evaluations by not evaluating the beam for every time instance but only after a chosen interval has passed. By buffering these beam predictions, the number of computations needed for visibility model prediction can be reduced.

5 February 2025 benchmarking results

In February 2025, we benchmarked our pipeline again using the same data set as in February 2024, but now using three nodes. In the meantime, we had also extended the pipeline to perform 7 self-calibration cycles in total, including a final cycle using 100% of the data. Table 2 summarises the performance results obtained. Comparing with the results obtained in February 2024 shown in Table 1 clearly shows the good scaling of the calibration stages to three nodes. The time taken by calibration varies between cycles due to the varying number of directions configured in direction-dependent calibration and due to the amount of data to calibrate (20% of the full observations for cycles 1 through 6, 100% for the last cycle).

Unfortunately, this is less apparent for imaging. To understand this, one has to realise that the imaging stages consist of four different activities: inversion (gridding and FFT to make a dirty image), prediction (FFT of the clean image and degridding to predict visibilities to close the major deconvolution cycle), deconvolution and filtering of the sky model. Our efforts focused on distribution of the (de)gridding work, but we did not touch deconvolution and filtering of the sky model. The latter two are still a single-node effort.

This is nicely illustrated by the CPU usage during the last two self-calibration cycles on the main node and the two worker nodes shown in Fig. 2 captured during a similar pipeline run. During calibration stages, all three nodes show over 70% usage of total CPU capacity. In the imaging stage, two distinct phases are visible. In the first phase, several major deconvolution cycles are executed. This phase is characterized by a spiky CPU usage pattern as the worker nodes are idle during image based deconvolution in each major deconvolution cycle. In the second phase, the sky model is filtered to reduce its complexity (measured as the total number of source components). This is done by merging source components if possible and discarding source components that are likely deconvolution artefacts. The aim is to save time in the next self-calibration cycle by reducing the number of source components in the model, but it is currently a single-node process that does not even fully exploit the available CPU capacity of the main node as visible in the top panel of Fig. 2.

6 Future work

Based on the results presented above, it is clear that reduction of the time spent on tasks running on a single node should be reduced significantly to enable scalability to more nodes. The following steps are currently being taken to achieve this:

- Enabling image-based predict by introducing w-towers [13] for visibility prediction and subpixel rendering of source components.
- Making sky model filtering optional: if sky model filtering takes more time than it saves, we may want to experiment with turning it off.
- Improving the performance of the sky model filtering function by using the available cores (even on a single node) more efficiently.
- Performing the corner turn between calibration (parallelised using time chunks) and imaging (parallelised using frequency chunks) on the fly exploiting the frequency chunk to node map introduced in WSClean.

Parallel to this, efforts to improve computational performance continue:

- Enabling Baseline-Dependent Averaging (BDA, [14, 15]), which reduces the visibility data volume and thus the number of visibilities to store and process.
- Overlapping I/O and processing during facet-based imaging.
- Introduction of GPU accelerated versions of compute-intensive functions.

7 Acknowledgements

This work was supported by the Foundation for Dutch Scientific Research Institutes (NWO-I) and the SKA Observatory (SKAO).

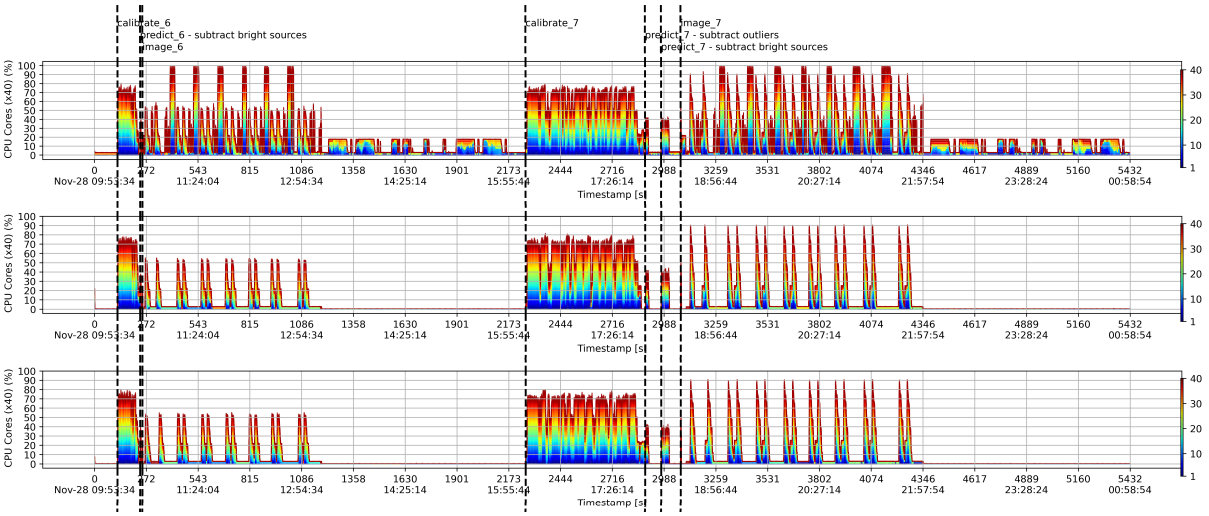

Figure 2. CPU uses of the main node (top row) and 2 worker nodes (bottom rows) during the last two self-calibration cycles.

References

[1] M. P. van Haarlem *et al.*, "LOFAR: The Low Frequency Array," *A&A*, vol. 556, no. A2, pp. 1–53, Aug. 2013, doi: 10.1051/0004-6361/201220873.

[2] F. Sweijen *et al.*, "Deep sub-arcsecond wide-field imaging of the Lockman Hole field at 144 MHz," *Nature Astronomy*, vol. 6, pp. 350–356, Jan. 2022, doi: 10.1038/s41550-021-01573-z.

[3] M. G. Labate *et al.*, "Highlights of the Square Kilometre Array Low Frequency (SKA-LOW) Telescope," *SPIE JATIS*, vol. 8, no. 1, Mar. 2022, doi: 10.1117/1.JATIS.8.1.011024.

[4] G. P. Swart, P. E. Dewdney, and A. Cremonini, "Highlights of the SKA1-Mid telescope architecture," *SPIE JATIS*, vol. 8, no. 1, Jan. 2022, doi: 10.1117/1.JATIS.8.1.011021.

[5] S. J. Wijnholds *et al.*, "Development of Python-based pipelines for LOFAR2.0 and SKA," in *URSI Atlantic Radio Science Conference (AT-RASC)*, Gran Canaria (Spain), May 2024, doi: 10.46620/URSIATRASC24/XDAS1340.

[6] G. van Diepen, T. J. Dijkema, and A. Offringa, "DPPP: Default Pre-Processing Pipeline," Astrophysics Source Code Library, record ascl:1804.003, Apr. 2018.

[7] A. R. Offringa, J. J. van de Gronde, and J. B. T. M. Roerdink, "A morphological algorithm for improving radio-frequency interference detection," *A&A*, vol. 539, no. A95, pp. 1–10, Mar. 2012, doi: 10.1051/0004-6361/201118497.

[8] A. R. Offringa *et al.*, "WSClean: an implementation of a fast, generic wide-field imager for radio astronomy," *MNRAS*, vol. 444, no. 1, pp. 606–619, Aug. 2014, doi: 10.1093/mnras/stu1368.

[9] A. R. Offringa and O. Smirnov, "An optimized algorithm for multiscale wideband deconvolution of radio astronomical images," *MNRAS*, vol. 471, no. 1, pp. 301–316, Jun. 2017, doi: 10.1093/mnras/stx1547.

[10] S. van der Tol, B. Veenboer, and A. R. Offringa, "Image Domain Gridding: a fast method for convolutional resampling of visibilities," *A&A*, vol. 616, no. A27, pp. 1–13, Aug. 2018, doi: 10.1051/0004-6361/201832858.

[11] J. A. Högbom, "Aperture Synthesis with a Non-Regular Distribution of Interferometer Baselines," *A&A Suppl.*, vol. 15, pp. 417–426, Jun. 1974.

[12] T. J. Cornwell, "Multiscale CLEAN Deconvolution of Radio Synthesis Images," *IEEE JSTSP*, vol. 2, no. 5, pp. 793–801, Oct. 2008, doi: 10.1109/JSTSP.2008.2006388.

[13] J. C. Kent, "Inteferometric Methods," Ph.D. dissertation, University of Cambridge, Combridge (UK), Jul. 2020.

[14] S. J. Wijnholds, A. G. Willis, and S. Salvni, "Baseline-dependent averaging in radio interferometry," *MNRAS*, vol. 476, no. 2, pp. 2029–2039, Feb. 2018, doi: 10.1093/mnras/sty360.

[15] C. Salvoni *et al.*, "Calibration and Imaging Pipeline Processing Baseline-dependent Averaged Visibilities," in *URSI Atlantic and Asia Pacific Radio Science Conference (AT-AP-RASC)*, Gran Canaria (Sapin), 30 May – 4 Jun. 2022, doi: 10.23919/AT-AP-RASC54737.2022.9814290.