Scalability Model for the LOFAR Direction Independent Pipeline

A.P. Mechev [a], T.W. Shimwell [b], A. Plaat [c], A.L. Varbanescu [d], H. Intema [a], H.J.A Rottgering [a]

[a] *Leiden Observatory, Niels Bohrweg 2, 2333 CA Leiden, the Netherlands*

[b] *ASTRON, Oude Hoogeveensedijk 4, 7991 PD , The Netherlands*

[c] *Leiden Institute of Advanced Computer Science, Niels Bohrweg 1, 2333 CA Leiden, the Netherlands*

[d] *University of Amsterdam, Spui 21, 1012 WX Amsterdam, the Netherlands*

Abstract

Understanding the performance of the LOFAR Direction Independent and Direction Dependent Pipelines is important when trying to optimize the throughput for large surveys. Making a model of completion time will enable us to predict the time taken to process a data set, optimize our parameter choices, help schedule future LOFAR processing services, and predict processing time for future large radio telescopes. We tested the full LOFAR `prefactor` target calibration pipeline by scaling several parameters, notably number of CPUs, data size and size of calibration sky model. Furthermore, we tested the overhead incurred by downloading and extracting the data and queuing jobs on the distributed cluster tasked with processing the LOFAR Two-Meter Sky Survey. We present these results as a comprehensive model which will be used to predict processing time for a wide range of processing parameters. We also discover that the calibration time scales favorably in time with smaller calibration models, while the calibration results do not degrade in quality. Finally, we validate the model and compare predictions with production runs from the past six months.

Keywords: Radio Astronomy, Performance Analysis, Performance Modelling, High Performance Computing

1. Introduction

Astronomy is entering the big data era with many projects creating Petabytes of data per year. This data is often processed by complex multi-step pipelines consisting of various algorithms. Understanding the scalability of astronomical algorithms theoretically, in a controlled environment, and in production is important to making prediction for future projects and upcoming telescopes.

The Low Frequency Array (LOFAR) (Van Haarlem et al., 2013) is a leading European low-frequency radio telescope. LOFAR's core stations are in the Netherlands, however it can collect data from sites across Europe. As it is an aperture synthesis array, LOFAR data needs to undergo several computationally intensive processing steps before obtaining a final scientific image.

To create a broadband image, LOFAR data is first processed by a Direction Independent Calibration pipeline followed by Direction Dependent Calibration software (Van Weeren et al., 2016; Williams et al., 2016; Smirnov and Tasse, 2015; Tasse et al., 2018). Direction Independent LOFAR processing can be parallelized on a high throughput cluster, while the Direction Dependent processing is typically performed on a single HPC node.

The LOFAR Surveys Key Science Project (SKSP) (Shimwell et al., 2017; Shimwell et al., 2018) is a long running project with the aim of creating a deep image of the northern sky at low frequencies. The broadest tier of the survey, Tier 1, will create more than 3000 8-hour observations at a sensitivity below 100 μJy. We have already processed more than 500 of these observations using the `prefactor` direction independent software (Horneffer et al., 2018).

While the current imaging algorithms can process data averaged by a factor of 64, it is important to understand how LOFAR processing scales with processing parameters, such as averaging parameters. With increasing LOFAR observation rates, data sizes and scientific requirements, users need to be able to predict the time and computational resources required to process their data.

To study the scalability of LOFAR processing, we set up processing of a sample SKSP data on an isolated node on the `GINA` cluster at SURFsara, part of the Dutch national e-infrastructure (Templon and Bot, 2016). We tested the software performance as a function of several parameters, including averaging parameters, number of CPUs and calibration model size. Additionally, we tested the performance of the underlying infrastructure, i.e. queuing and download time, for the same parameters. Finally, we compared the isolated tests with our production runs of the `prefactor` pipeline to measure the overhead incurred by running on a shared system.

We discover that the intensive LOFAR processing steps

Email address: `apmechev@strw.leidenuniv.nl` (A.P. Mechev [a]) scale linearly with data size, and calibration model size. Additionally, we find that these steps scale as an inverse power law with respect to the number of CPUs used. We discover that the time to download and extract data on the `GINA` cluster is linear with size up to 32GB, however becomes slower beyond this data size. In addition to this overhead, discover that queuing time on the `GINA` cluster grows exponentially for jobs requesting more than 8 CPUs. We validate these isolated tests with production runs of LOFAR data from the past six months. Finally we combine all these tests into a single model and show its prediction power by testing the processing time for different combinations of parameters. The major contributions of this work can be summarized as:

- A model of processing time for the slowest steps in the LOFAR Direction Independent Calibration Piepline.
- A model of queueing time and file transfer time which can be used by current or future jobs processed on the `GINA` cluster.
- Insight into calibration speed and quality with calibration models of radically different size and sensitivity.

We introduce LOFAR processing and other related work in Section 2 and describe our software setup and data processing methods in Section 3. We present our results and performance model in Section 4 and discussions and conclusions in Section 5.

2. Related Work

In previous work, we have parallelized the Direction Independent LOFAR pipeline on a High Throughput infrastructure (Mechev et al., 2017). While this parallelization has helped accelerate data processing for the SKSP project, creating a performance model of our software is required if we are to predict the resources taken by future jobs. This model will be particularly useful in understanding how processing parameters will affect run time.

Performance modelling on a distributed system is an important field of study related to grid computing. A good model of the performance of tasks in a distributed workflows can help more efficiently schedule these jobs on a grid environment (Sanjay and Vadhiyar, 2008). Many of the systems require knowledge of the source code and an analytical model of the slowest parts of the code (Xu et al., 1996). Many systems exist to model the performance of distributed jobs (Barnes et al., 2008; Xu et al., 1996; Kuperberg et al., 2008; Witt et al., 2018), with some employing Black Box testing (Yang et al., 2005; Kavulya et al., 2010) or tests on scientific benchmark cases (Carrington et al., 2006). Such performance analysis does not require intimate knowledge of the software and can be applied on data obtained from processing on a grid infrastructure.

Empirical modelling is useful in finding bugs in parallel code (Calotoiu et al., 2013) and modelling the performance of big data architectures (Castiglione et al., 2014). The insights from these models are used to optimize the architecture of the software system or determine bottlenecks in processing. We plan to use empirical modelling to determine how the LOFAR `prefactor` performance scales with different parameters.

3. Processing Setup

Using the CVMFS LOFAR software install described in (Mechev et al., 2017), we processed a typical LOFAR SKSP observation [1] at multiple different averaging parameters. Changing these parameters will decrease the final data size (as seen in Table 1). We test the processing time for different averaging parameters by running 15 runs per parameter step.

The processing was done on a dedicated node of the SURFsara `GINA` cluster, f18-01. The node is a typical processing node used by our LOFAR Surveys processing, however it is dedicated for the tests in order to ensure there's no contamination by other processes. The node is described in Section 3.3.

There are two sources of latency that need to be studied for a true end-to-end model of LOFAR processing. The first is the performance of the LOFAR software on the Dutch grid for a wide range of processing parameters. The second is the overhead, such as job queuing and data movement. Both of these effects depend on similar parameters such as data size and number of CPUs used. We will examine these effects in our study of the LOFAR processing performance by studying the performance at different parameter steps. While some parts of the processing software may change, the infrastructure parts of our performance model can be used independently of the the processing software and can even be applied to other scientific projects running on the `GINA` cluster.

We processed the sample data set with the LOFAR `prefactor` pipeline. The `prefactor` version used was the same as we use for the LOFAR SKSP broadband surveys (Horneffer et al., 2018).

3.1. Processing Metrics

The goal for our scalability model is to understand the effect of several parameters on the job completion time of LOFAR software. We do this by testing the processing time for various values of data size, number of CPUs used and sky model size.

The data used by the LOFAR surveys is stored at a time resolution of 1 second intervals and frequency resolution of 16 channels per Subband (equivalent to 12kHz

[1] LOFAR Observation ID L658492, co-ordinates [17h42m21.785, +037d41m46.805] observed by the LOFAR High Band Array for 8 hours between 2018-06-20 and 2018-06-21.

channel width). While some of the processing steps such as flagging of Radio Frequency Interference and removal of bright off-axis sources needs to be done on the high-resolution data, later steps can be performed on averaged data. To make processing faster, the raw data is averaged in time and frequency, decreasing the input data size to later tasks. For the LOFAR surveys project, our final averaging parameters are 8 seconds per sample and 2 channels per Subband. This corresponds to a reduction in size by a factor of 64. In section 4.1.1, we measure the performance or the `prefactor` pipeline for data sizes between the raw data of 64GB/Subband and the averaged data of 1GB/Subband. The tested data sizes and parameters are shown in table 1. It should be noted that the gsmcal_solve and gsmcal_apply steps act on a concatenation of 10 Subbands. As such, if the input data is 1GB, the input to these steps will be 10GB. The concatenation of `prefactor` data is described fully in (Mechev et al., 2017) and (Van Weeren et al., 2016).

Data set	Time averaging parameter (sec)	Channels per Subband	Size (Gb)
1GB	8	2	1.235
2GB	4	2	2.459
4GB	2	2	4.906
8GB	1	2	9.802
16GB	1	4	18.00
32GB	1	8	36.72
64GB	1	16	66.88

Table 1: Averaging parameters and final data sizes tested for the sample LOFAR SKSP observation. The raw data is 64 GB per Subband. The LOFAR SKSP data processing uses averaging parameters of 8 seconds and 2 channels per Subband. This reduces the raw data by a factor of 64. We highlight the data size used in the LOFAR SKSP Tier 1 survey.

The slowest step of the `prefactor` pipeline is the gsmcal_solve step, which performs the gain calibration against a model of the radio sky. We obtain this model through the TGSS sky model creator[2]. By default this service creates a text sky-model from the TGSS survey (Intema et al., 2017) and sets a threshold of sources brighter than 0.3 Jy. Lowering this threshold creates longer sky-model files with more faint sources, while increasing it will return only the few brightest sources. Since sky model calibration requires converting the sky-model into UV data (van Diepen and Dijkema, 2018), a longer sky model will increase the time taken to gain calibrate a data set. We created 7 sky models with minimal flux ranging between 0.05 Jy and 1.5 Jy. The resulting files are listed in Table 2. For production[3],

[2]Accessible at the TGSS ADR portal.

[3]The query used to obtain model 3 is `http://tgssadr.strw.leidenuniv.nl/cgi-bin/gsmv3.cgi?coord=265.590770833,37.69633472220001&radius=5&unit=deg&deconv=y&cutoff=0.3`

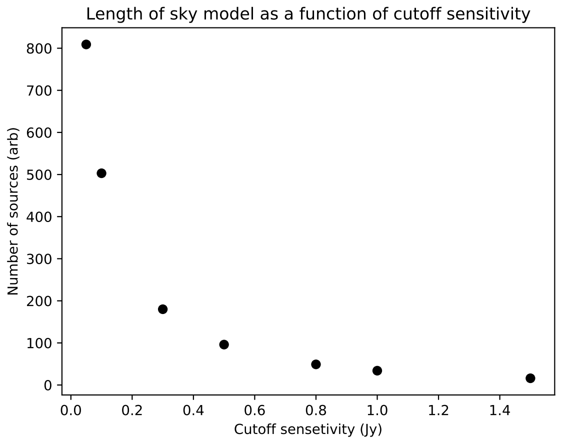

Figure 1: The size of the sky model (measured in number of sources) increases exponentially as we decrease the flux cutoff of the model (i.e. increase the sensitivity).

we used the sensitivity parameters for model 3. Each line of these model files corresponds to one source, modelled either as a point or an ellipse), hence the second column also lists the number of sources per sky model file.

It is important to note that the model of the sky depends on the direction of observation. As such, our test is only a heuristic for predicting the run-time based on the calibration model length. The model size and calibration solution quality will differ for other observations and these results will require a further in depth study before changing the calibration strategy. Additionally, it is notable that the number of sources is an exponentially dependent on the minimum sky model sensitivity (seen in Figure 1). According to this relationship, even a modest decrease in sensitivity can significantly decrease the size of the model.

Sky model #	min sensitivity	# lines
model 1	0.05 Jy	809
model 2	0.1 Jy	503
model 3	0.3 Jy	180
model 4	0.5 Jy	96
model 5	0.8 Jy	49
model 6	1.0 Jy	34
model 7	1.5 Jy	16

Table 2: List of test sky models. Model 3 is created with the parameters used in our production processing of LOFAR data. All models include objects within 5 degrees from the centre of the pointing.

Finally, the number of CPUs used by each step is a parameter that can be optimized for the entire pipeline. While increasing the number of CPUs can make some steps run faster, requesting jobs that reserve a large number of CPUs will take longer to launch on shared infrastructure. In order to understand these two effects, we study the queuing time and processing time as a function of number of CPUs. For the parameter steps we choose to test 1, 2, 3, 4, 8 and 16 CPUs.

3.2. Infrastructure Performance

Since our jobs are launched on a cluster supporting several different use cases, the requested resources are allocated by a job queue, in our case implemented by the glite workload management system (Marco et al., 2010). As queuing jobs can take significant amount of time, we test the queuing time as a function of number of requested CPUs. In order to do that, we create test jobs that log the launch time and submit them, requesting 1, 2, 3, 4, 8 and 16 CPUs. We run several tests for each parameter step to ensure that we capture system variability at different times of day during the week and the weekend.

Besides queuing, time is also spent during downloading and unpacking data, as well as uploading and packing the results. Despite using no compression to pack the data, untarring and tarring large files still takes time depending on the system workload. We measure the time taken to transfer and unpack data of different sizes. The data sizes we chose were 0.5GB, 1GB, 2GB, 4GB, 8GB, 16GB, 32GB and 64GB. As our largest data sets are 64GB and our smallest results are ∼0.2GB, these values span a realistic range expected for LOFAR data processing. We test this by uploading mock data to the dCache storage pool at SURFsara and launching a small 1 CPU job which downloads and untars the data, logging the start time of each step. We present the results of this test in the next section.

3.2.1. Software Versions

For the current test, we use the LOFAR software stack, version 2.20.2(Dijkema, 2017). This software was compiled on a virtual machine and distributed using the CERN CVMFS virtually mounted file system. We use this software version and distribution method as it is the same software version and distribution used to process the data for the LOTSS Data Release 1.

3.3. Test Hardware

The LOFAR software was tested on a reserved node on the SURFsara `GINA` cluster. The node, f18-01 has 348 GB of RAM, 3TB of scratch space [4]. The CPU is an Intel Xeon(R) Gold 6148 CPU with 40 cores clocked at 2.40GHz. As this hardware node was reserved, there was no other scientific processing aside from our tests, meaning there was no resource contention aside for that inherent in the LOFAR software. In the results section, we compare these isolated runs with processing results over the past two years.

[4]More detailed specifications are at the `GINA` specification page linked here

4. Results

Using a test data set, we tested the LOFAR `prefactor` target pipeline on the SURFsara `GINA` cluster. First we will present the tests done in an isolated environment and then compare them to the run time in production on a shared infrastructure. We will integrate all the results in a complete model which can be used to predict processing time for a variety of parameters. Finally, we will make some predictions on the run time of our processing based on the model and validate these predictions.

Since we are processing a sample data set in the context of the LOFAR Surveys project, we will compare these tests with the production runs of our pipeline. In production, the parameters chosen are a data size of 1GB, a sky model length of 180 lines and 8 CPUs for the gsmcal_solve step.

4.1. Isolated Environment tests

We first tested the LOFAR software in isolation in order to determine the scalability of processing time in terms of data size. We run the entire `prefactor` target pipeline which which removes Direction Independent Calibration errors from a LOFAR science target. In the following sections, we present the models obtained from these tests.

4.1.1. Input Data Size

LOFAR data can be averaged to different sizes based on the scientific requirements. Smaller data sets are processed faster, so it is important to understand the effect of data size on processing time. We show the processing time for our test data set, averaged to different sizes for several `prefactor` steps in Figures 2, 3, 4, 5, 6. The figures also plot linear fits for consecutive pairs of parameter steps, in gray dashed lines, used to help guide the selection of parametric model.

All of the steps show linear behavior with respect to input data size, while the gsmcal_solve step is best fit by two linear relationships, for data smaller and larger than 16 GB. The linear fit to the run times are shown in Equations 1a-1e. The equations show the processing time as a function of the data size ($\mathcal{S}$), with the slope in the units of seconds/byte. The fits are also shown in Figures 2 to 6 as a black dashed line.

4.1.2. Calibration Model Size

To test the effect of the calibration model size on run time, we tested our calibration with several different lengths of the sky model file. We created these models by changing the maximum sensitivity using values ranging from 0.05 Jy to 1.5 Jy. The most sensitive model (0.05 Jy) had 809 sources while the 1.5 Jy model had only 16 sources.

Figure 7 shows that the calibration time is directly proportional to the length of the sky model. Figure 8 shows the run time as a function of the processing parameter: the cutoff sensitivity. As the relationship between the number

$$T_{predict\_ateam} = 5.19 \times 10^{-8}\mathcal{S} + 4.20 \times 10^{1} \tag{1a}$$

$$T_{ateamcliptar} = 4.57 \times 10^{-9}\mathcal{S} - 8.42 \times 10^{0} \tag{1b}$$

$$T_{dpppconcat} = 3.51 \times 10^{-8}\mathcal{S} + 4.20 \times 10^{1} \tag{1c}$$

$$T_{gsmcal\_solve} = \begin{cases} 7.38 \times 10^{-7}\mathcal{S} - 8.20 \times 10^{1} & |\mathcal{S} <= 1.6 \times 10^{10} \\ 1.04 \times 10^{-6}\mathcal{S} - 4.04 \times 10^{3} & |\mathcal{S} > 1.6 \times 10^{10} \end{cases} \tag{1d}$$

$$T_{gsmcal\_apply} = 2.07 \times 10^{-8}\mathcal{S} - 1.38 \times 10^{1} \tag{1e}$$

Equation 1: Equations describing the processing time of five **prefactor** steps as a function of the input data size ($\mathcal{S}$) in bytes.

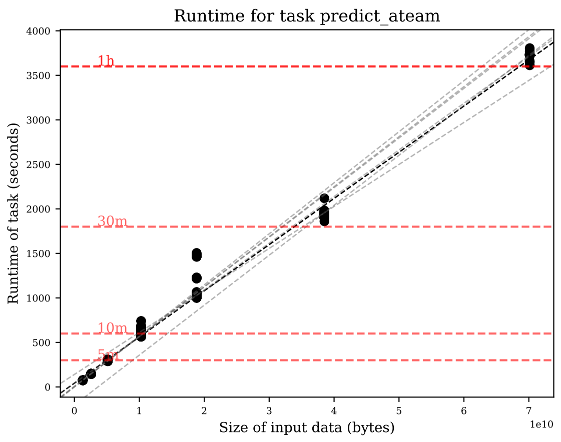

Figure 2: Tests of the predict_ateam step for input data size ranging from 1GB to 64 GB. This step calculates the contamination from bright off-axis sources. Dashed lines are shown connecting each pair of points, to highlight the trend. The linear model fit to this data (Equation 1a) is shown in black.

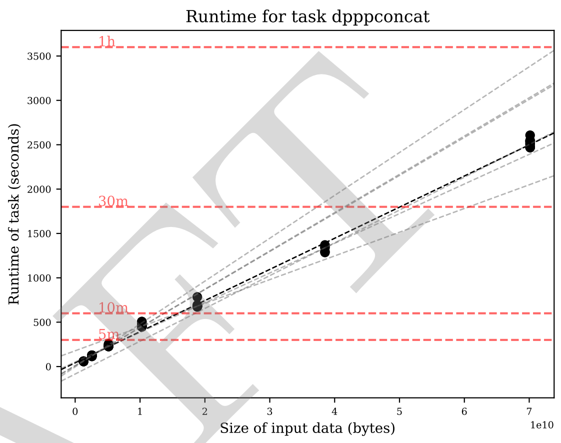

Figure 4: Tests of the dpppconcat step for input data size ranging from 1GB to 64 GB. This step concatenates 10 files into a single measurement set. The linear model fit to this data (Equation 1c) is shown in black.

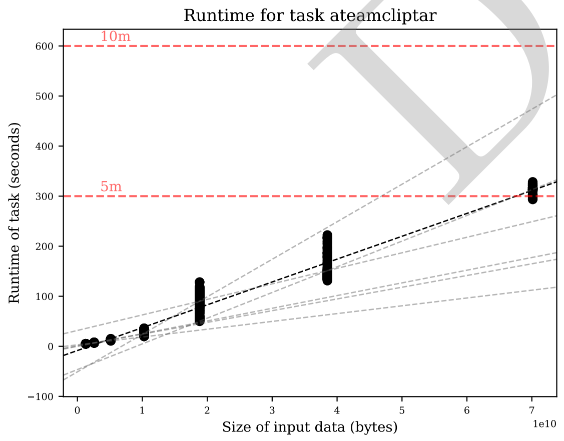

Figure 3: Tests of the ateamcliptar step for input data size ranging from 1GB to 64 GB. This step removes the contamination from bright off-axis sources. The linear model fit to this data (Equation 1b) is shown in black.

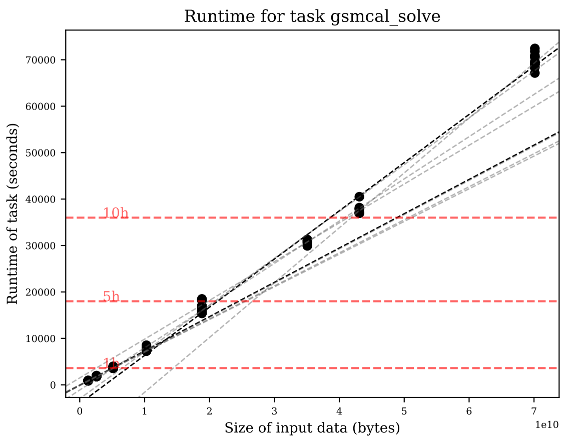

Figure 5: Tests of the gsmcal_solve step for input data size ranging from 1GB to 64 GB. This step performs gain calibration of the concatenated data set against a sky model. It is the slowest and most computationally expensive **prefactor** step. We fit two linear models, for data below 16GB and above 16GB. The models, shown in (Equation 1d) are shown in a black dashed line.

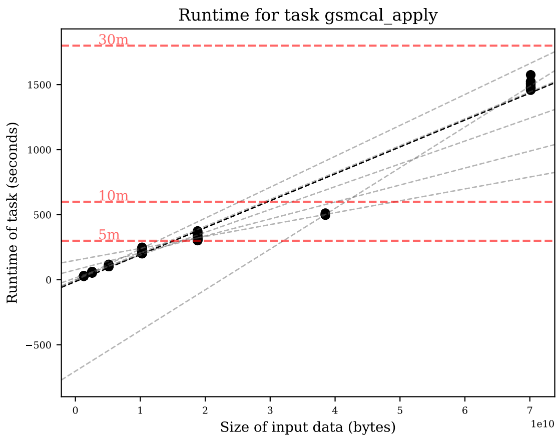

Figure 6: Tests of the gsmcal_apply step for input data size ranging from 1GB to 64 GB. This step applies the calibration solutions to the data. The linear model fit to this data (Equation 1e is shown in dark blue.

$$T = 1185 \cdot \mathcal{F}^{-0.854} \tag{2}$$

Equation 2: Processing time for the gsmcal_solve step as a function of the flux cutoff of the calibration model ($\mathcal{F}$) in Jansky

of sources and cutoff sensitivity is a power law, here we see the same relationship holding for processing time.

We model the run time as a function of the cutoff frequency using a power law, and fit the data to the function $y = \alpha \cdot \mathcal{F}^{-k}$. Our fit found the best model to be shown in Equation 2, where $\mathcal{F}$ value is the cutoff flux in Jansky and T is the run time in seconds.

We show four images made from data sets calibrated with a 0.05Jy (top left), 0.3Jy (top right), 0.8 Jy (bottom left) and 1.5 Jy (bottom right) cutoff in Figure 9. The statistics for the four images, taken from the regions in green on Figure 9) are shown in Table 3.

Calibration Model Flux Cutoff	RMS Noise (Jy)	std dev (Jy)
0.05Jy	0.00402834	0.004026
0.3 Jy	0.00402311	0.004020
0.8 Jy	0.00404181	0.004039
1.5 Jy	0.00410204	0.004105

Table 3: Statistics for an empty region for the four images shown in Figure 9. The 0.3Jy model, here shown shaded in gray, is the one used in production.

4.1.3. Number of CPUs

One parameter that can be optimized is the number of CPUs requested when the job is launched. We investigated the processing speedup as a function of the number of CPUs for the `prefactor` target pipeline. From the steps tested, only the gsmcal_solve step shows a significant

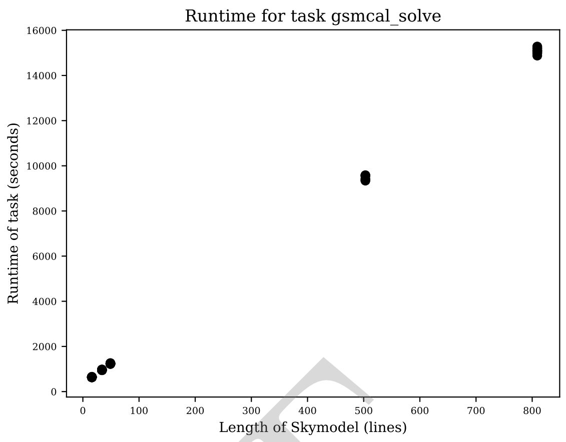

Figure 7: The processing time of the gsmcal_solve step is linear with the size of the sky model as measured by the number of sources.

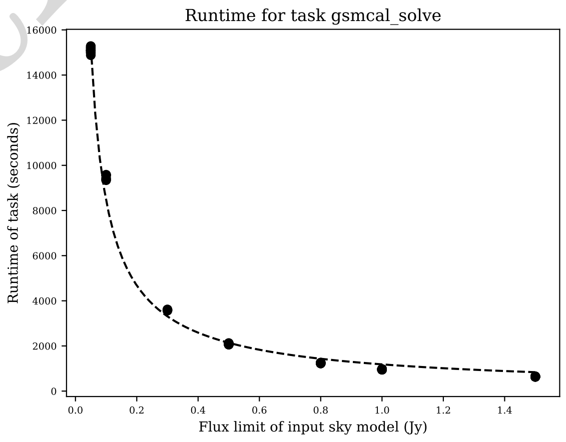

Figure 8: The run time of the gsmcal_solve step as a function of the cutoff sensitivity is not linear. As shown in Figure 1, the number of sources increases exponentially as the minimum sensitivity decreases. The dashed line shows the model fitted in Equation 2.

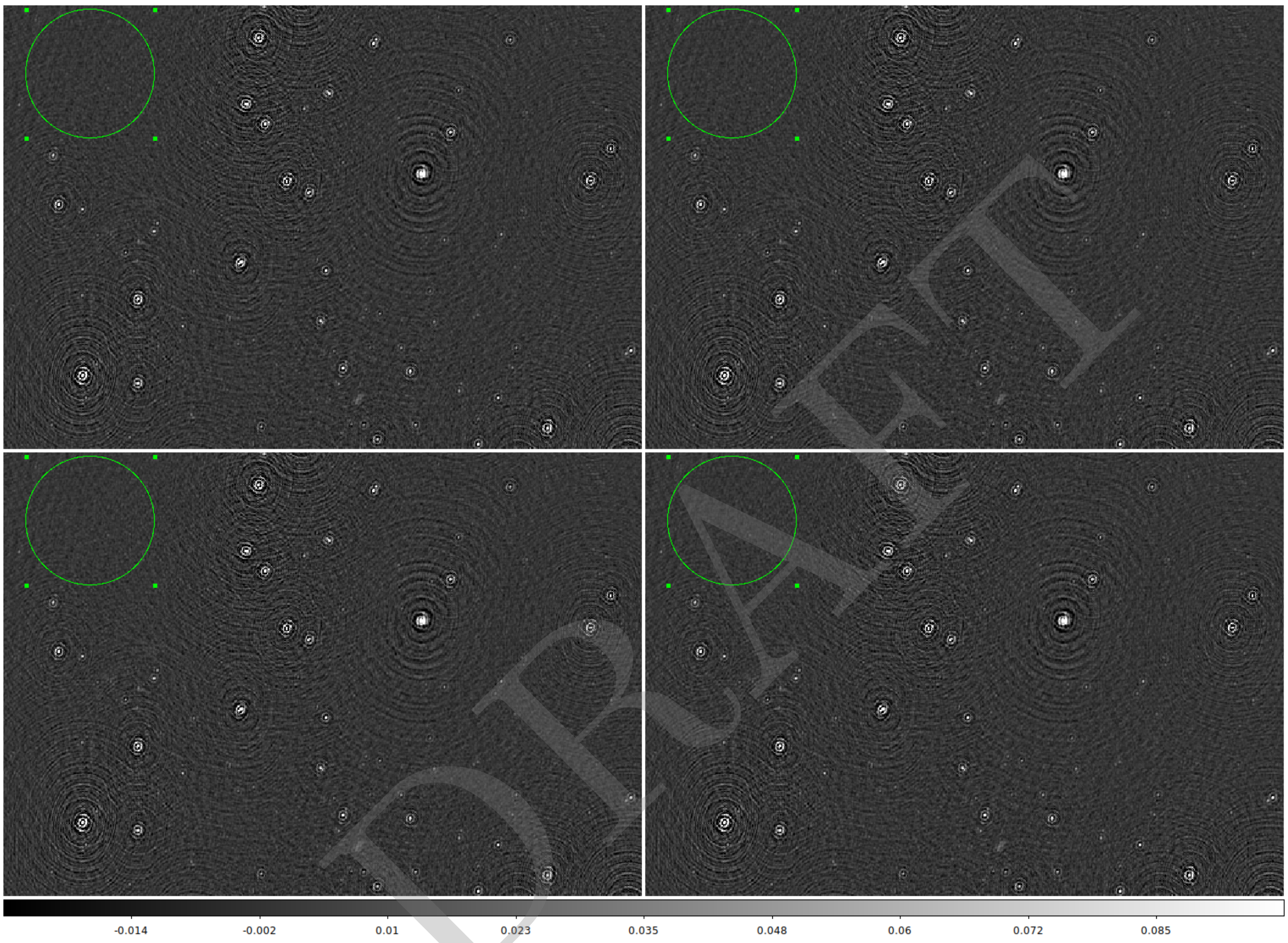

Figure 9: Four dirty images made with `wsclean` from the data set. The four images were calibrated with sky models of various flux cutoffs ranging from 0.05Jy (top left) to 1.5Jy (bottom right). Flux statistics for the green regions in the four images are listed in Table 3.

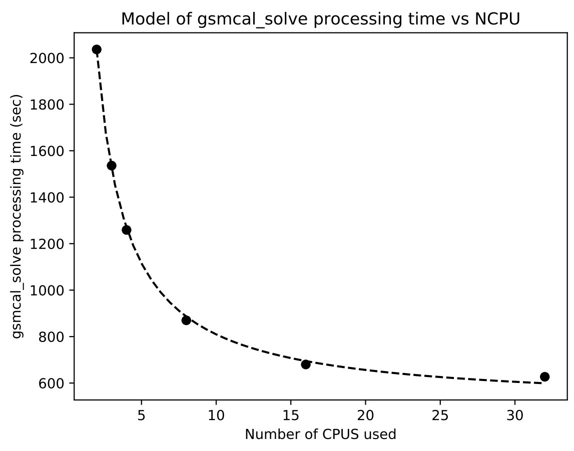

Figure 10: The processing time of the gsmcal_solve step decreases exponentially with the number of CPUS requested. The model in Equation 3 is shown in a dashed line

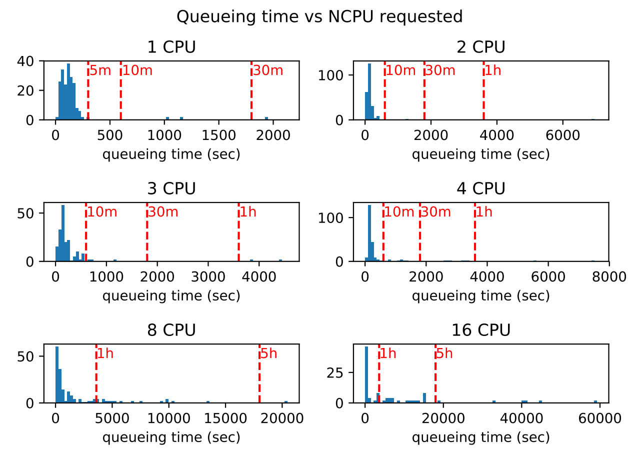

Figure 12: Test randomly submitting jobs to the GINA with different number of requested CPUs.

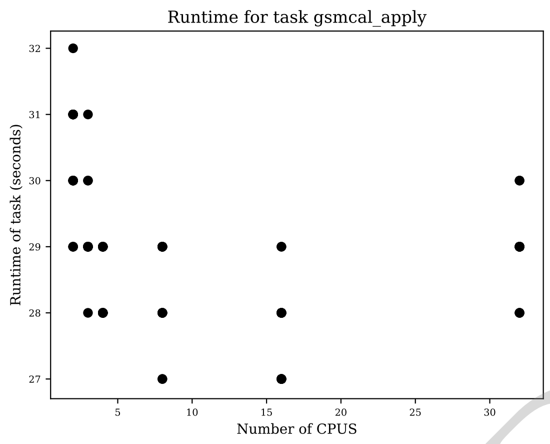

Figure 11: The step that applies the calibration solutions, gsmcal_apply, does not show a speedup when run on multiple cores.

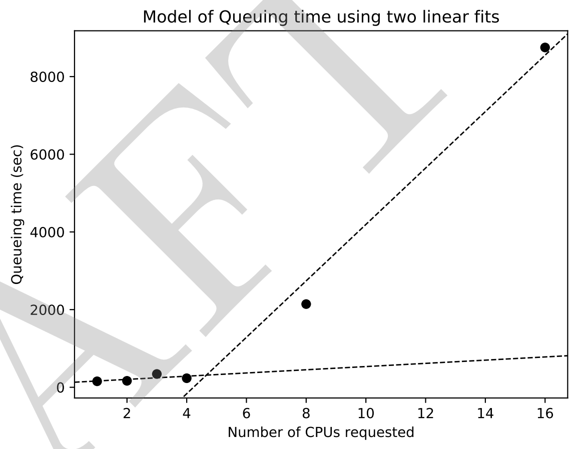

Figure 13: The queuing model built from two linear fits to the queuing times. We use the 75th percentile of the queuing data as a upper bound of job queuing.

speedup as the number of CPUs is increased. The run time of this step is an inverse power law with respect to the number of CPUs as seen in Figure 10. Unlike the solving step, the step applying the calibration solutions (gsmcal_apply) is constant in time with respect to the number of CPUs as seen in Figure 11.

We fit an inverse proportional model to the raw data, shown in Equation 3, with the parameter ($\mathcal{N}$) being the number of CPUs used.

$$T = 503.37 + \frac{3062.6}{\mathcal{N}} \tag{3}$$

Equation 3: Processing time for the gsmcal_solve step as a function of ($\mathcal{N}$), the Number of CPUs used by the process.

4.2. Queuing Tests

Aside from performance of the LOFAR software, we measured the queuing time at the GINA cluster, as a function of the number of CPUs requested. This data was obtained between 16 Nov 2018 and 10 Dec 2018 for 1, 2, 3 ,4, 8, and 16 CPUs per job. A histogram of the queuing time for these jobs is shown in Figure 12. Statistics for these runs are in Table 4. We use the 75th percentile of the queuing time for each parameter step to fit a model. This scenario will include 75% of runs and is a good trade-off between ignoring and including outliers.

We fit two linear models for this queueing time. One model for 1-4 CPUs and one for 4-16 CPUs. The model, as a function of Number of CPUS, $\mathcal{N}$ is in equation 4. The two models are plotted against the 75th percentile of the queuing times (last column in Table 4) in Figure 13.

NCPU requested	Mean time (sec)	Median time (sec)	75th percentile (sec)
1 CPU	150.5	116.2	154.1
2 CPU	201.1	125.8	165.8
3 CPU	296.2	152.0	243.0
4 CPU	498.9	167.7	233.7
8 CPU	1944.2	428.4	2142.4
16 CPU	7079.0	696.4	8750.6

Table 4: Statistics for queuing time for different values of CPUs requested.

$$T = \begin{cases} 49.3 \cdot \mathcal{N} + 120 & |\mathcal{N} <= 4 \\ 726 \cdot \mathcal{N} - 3071 & |\mathcal{N} > 4 \end{cases} \quad (4)$$

Equation 4: The model for the Queuing time as described by two linear models.

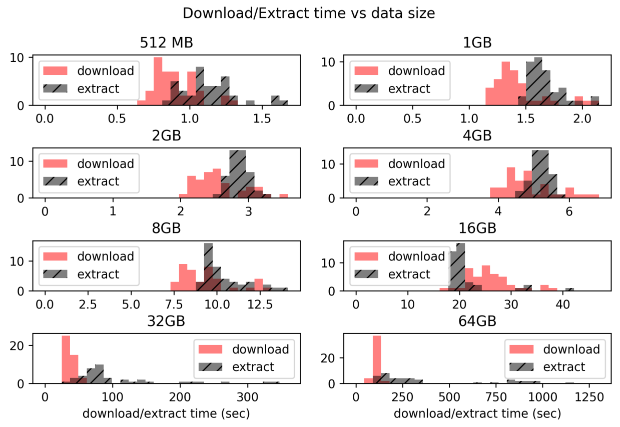

Figure 14: A histogram of the download and extracting times of multiple data sizes on the GINA worker nodes. Download and extract times are comparable for data up to 8GB, however above that, the extracting time dominates.

4.3. Transfer and Unpacking Time

We tested the downloading and unpacking time for data sizes ranging from 512MB to 64GB. We discovered that the unpacking of files below 64GB scaled linearly with file size, however unpacking data larger than 16GB becomes considerably slower than downloading it.

Figure 14 shows the histogram of the download tests, and Figure 15 displays the tests as a function of data size. Both figures show that extracting of the 32 and 64GB data sets has more slow outliers than the downloading of this data.

We fit a power law model to the time taken to transfer and unpack the data. In this case, we also consider the 75th percentile of these times in order to capture the majority of runs and ignore outliers. The plot of the data and our model can be seen in Figure 15 and the model is in Equation 5, as a function of the input data size, $\mathcal{S}$ in Gigabytes.

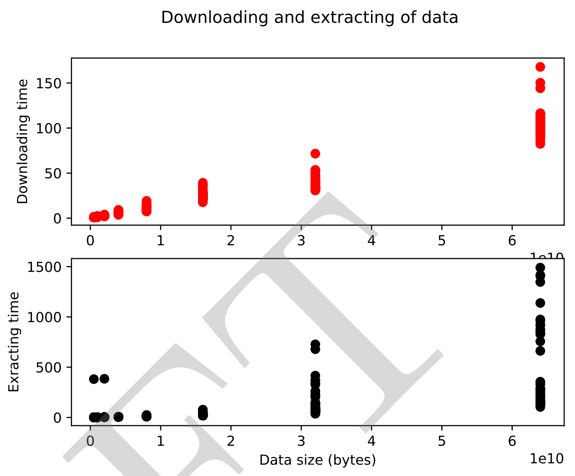

Figure 15: A scatter plot of the download and extracting times of multiple data sizes on the GINA worker nodes. The difference between download and extract time for the 32 and 64 GB data sets can be seen.

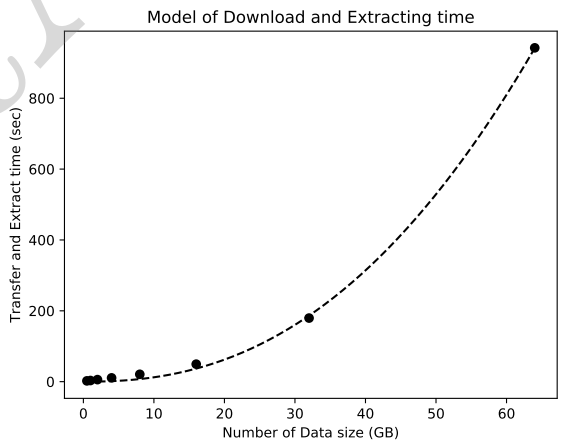

Figure 16: Fit of an exponential model to the Download and Extraction time for different data sizes. For the transfer overhead, we took the 75th percentile from the data shown in Figure 14. The model in Equation 5 is shown in a dahsed line.

$$T = 5.918 \times 10^{20} \cdot \mathcal{S}^{2.336} \quad (5)$$

Equation 5: Model of the downloading and extracting time as a function of the data size ($\mathcal{S}$) in bytes.

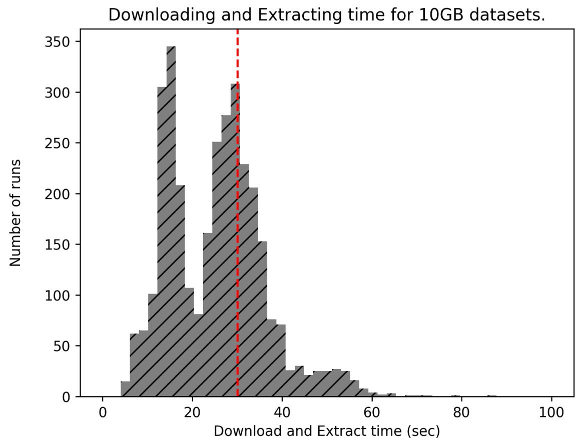

Figure 17: Downloading and extracting time for 10 1GB data sets performed in our production environment. Data from this test ranges from July 2018 to January 2019. The dashed red line shows the prediction obtained from section 4.3.

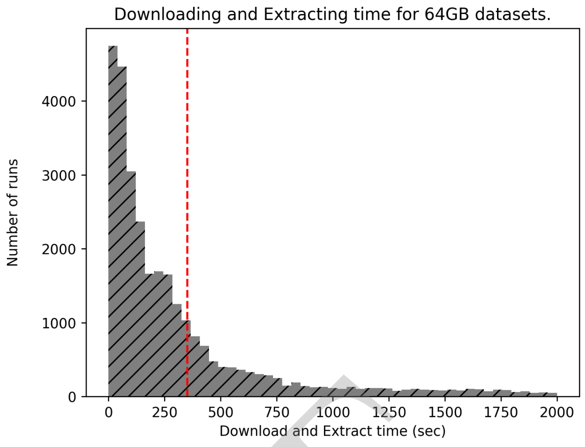

Figure 18: Downloading and extracting time for a 64GB data set performed in our production environment. Data from this test ranges from 07/2018-01/2019. The dashed red line shows the prediction obtained from Figure 5 in Section 4.3.

4.4. Comparison with production runs

Over the past two years, the LOFAR software has been running in production and collecting data on run time for each processing step. We have saved detailed logs for these runs starting in July 2018. We can compare this to the isolated model in order to determine the overhead incurred by processing LOFAR data on shared nodes.

Using the logs recorded by our processing launcher[5], we made plots showing the processing time for the downloading and extracting, and for the slowest steps, ndppp_prepcal and gsmcal_solve. The results are shown in Figures 17 and 18. We include predicted extract times from Section 4.3 as vertical dashed lines for both plots.

Finally, we present Figures 19 which shows a comparison of gsmcal_solve run times and our model's prediction for a 1GB data set. Figure 20 plots the processing time vs data size for these production runs and includes the model from Equation 1d. The significant overhead incurred on a shared infrastructure can be noted.

4.5. Complete Scalability Model

To incorporate all our data into a complete model, we consider the slowdown of each parameter as a multiplier to the time taken to process our base run. We incorporate the models for each parameter above for the model of the run time. We add the transfer and queuing time to the processing time to obtain a final function of all our parameters. We can use this function to predict the processing time for an arbitrary data set. The final performance model is in Equation 6.

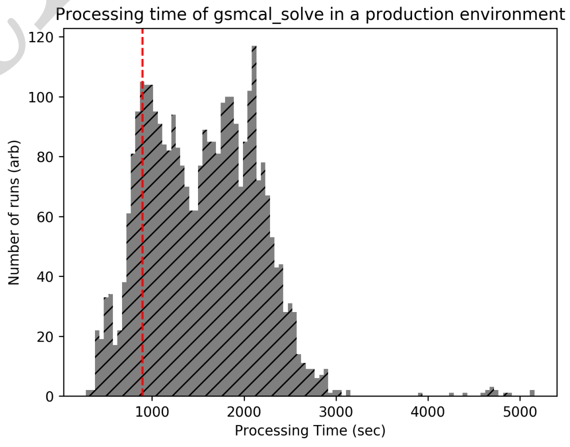

Figure 19: Processing time for the gsmcal_solve step in a production environment. Data from this test ranges from 07/2018-01/2019. The dashed red line shows the prediction for a 1GB run, obtained from section 4.3. It should be noted that the first peak corresponds to 512MB data, as seen in Figure 20.

[5]GRID_PiCaS_Launcher, `https://github.com/apmechev/GRID_picastools`

$$
\begin{aligned}
t_{gsmcal\_solve} = &\begin{cases} 49.3 \cdot \mathcal{N} + 120 & |\mathcal{N} <= 4 \\ 726 \cdot \mathcal{N} - 3071 & |\mathcal{N} > 4 \end{cases} \\
&+ 0.056 \cdot \mathcal{S}^{2.336} \\
&+ 3566 \cdot \frac{1}{3.012} \mathcal{F}^{-0.854} \cdot 9.97 \cdot 10^{-7} \mathcal{S} \cdot (0.1412 + \frac{0.8589}{\mathcal{N}}) \\
&+ 0.056 \cdot \mathcal{S}^{2.336}
\end{aligned}
\tag{6}
$$

Equation 6: Model of the total time of the gsmcal_solve step (t_{gsmcal_solve}) for the parameters $\mathcal{N}$, Number of CPUs; $\mathcal{S}$, Size of data in bytes and $\mathcal{F}$, cutoff calibration model flux in Jansky.

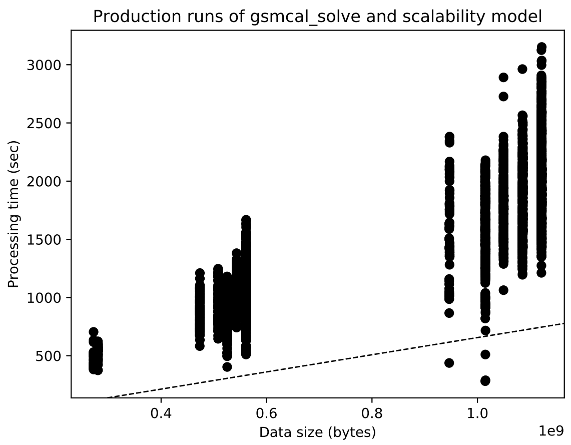

Figure 20: The scalability model for processing data through the gsmcal_solve step and the performance for production runs of this step between July 2018 and January 2019.

5. Discussions and Conclusions

The goal of this work is to understand the performance of the LOFAR Direction Independent Pipeline as processing parameters are scaled up or down. We increased the size of broadband LOFAR data by a factor of 64 in size, and discovered that all our processing steps scale linearly in time with respect of the input data size. We learned that for input data above 16GB in size, the slope of our scaling relation is higher than for the smaller data sets.

As the calibration step concatenates 10 input subbands, and the test system only has 380GB of RAM, data larger than 16GB, the discrepancy in slopes can be attributed to the large memory requirement for data larger than 200GB. Splitting the performance model in two also helps make a more accurate processing time prediction as fitting a single linear model would have a large negative y-intercept, predicting negative processing times for data smaller than 2GB.

When comparing our model's prediction with real processing runs over the past six months, we note that there are considerable overheads when processing data on an isolated node vs when running on a shared infrastructure (Figure 20). The overhead in processing is roughly a factor of two-three from our model. This discrepancy suggests that a model for gsmcal_solve needs to be built using data when running on a shared environment, to better predict processing time.

Overall, the slowest step was the gsmcal_solve step, and its run time scales more strongly with data size than the other steps (equation 1d has the steepest slope). This suggests that as data sizes increase, gsmcal_solve will increasingly dominate the processing time over the other steps (As seen in Figures 2 and 5). This effect is especially prominent for data larger than 16GB (160GB when 10 subbands are concatenated). As such, it is recommended to avoid calibration of data larger than 160GB.

Additionally we discovered that the calibration scales linearly in time as a function of the length of the calibration model, however it is a power law as a function of the model's cutoff sensitivity. This is because of the (expected) power law relation between the number of sources and cutoff sensitivity, seen in Figure 1. We can use this discovery to accelerate the processing time by increasing the flux cutoff to the LOFAR direction independent calibration to 0.5 Jy. Doing so will execute the calibration step in 60% of the time, saving 83 CPU-h per run. Over the remaining 2000 `prefactor` runs left in the LOTSS project, this change in sensitivity will save more than 167k CPU hours. Figures 9 show a data set calibrated with sky models with cutoff sensitivities listed in Table 3, and figures A.21 and A.22 show the calibration solutions obtained by calibrating with skymodels of cutoff ranging from 0.05 Jy to 1.5 Jy. These results suggest that performing gain calibration with less complex, and thus smaller, calibration models will not degrade image and solution quality while taking less than 20% of processing time. Table 3 also confirms this result.

We tested downloading and extracting test LOFAR data of various sizes. Both downloading and extracting was linear with respect to the data size for data up to 32 GB in size. Beyond those sizes, there was more scatter in data extraction. This is because load on the worker node's file-system can be unpredictable and can affect the extraction negatively. Nevertheless, when comparing our extraction tests and processing for the past 6 months, the predictions by our models (Figure 14) correspond to the

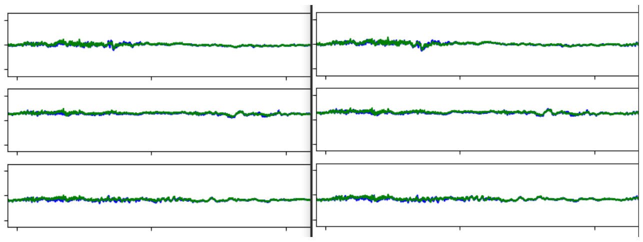
Figure A.21: The calibration (phase) solutions for the test dataset obtained when calibrating with sky models of 0.05 Jy cutoff (left) and 0.3Jy cutoff (right). The data shows the phase solutions for baselines including stations CS003HBA0, CS003HBA1 and CS004HBA0, with respect to the reference station, CS001HBA0.

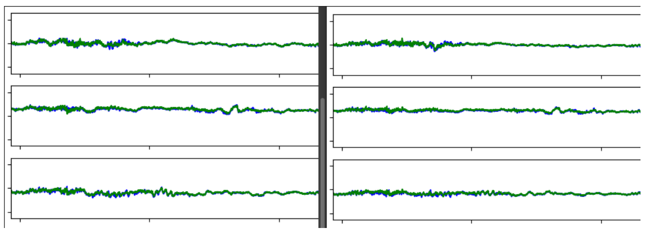
Figure A.22: The calibration (phase) solutions for the test dataset obtained when calibrating with sky models of 0.8 Jy cutoff (left) and 1.5Jy cutoff (right). The data shows the phase solutions for baselines including stations CS003HBA0, CS003HBA1 and CS004HBA0, with respect to the reference station, CS001HBA0.

production runs (Figures 17 and 18). Part of the LOFAR SKSP processing is done on shared infrastructure, which requires requesting processing time ahead of time for each grant period. Being able to predict the amount of resources required to process data each grant period is required to make a reliable estimate on what resources to request.

Moreover, providing LOFAR processing as a service to scientific users requires estimating the processing time for each request. This needs to be done in order to determine whether the user has sufficient resources left in their resource pool. Knowing the performance of the software pipelines as a function of the input parameters will help predict the run time for each request and the resources consumed. Knowing this will make it possible to notify users how long the request will take and how much of their quota will be used up.

Finally, a performance model of the LOFAR software will help make predictions on the time and resources needed to process data for other telescopes such as the Square Kilometer Array (SKA). Once operational, the SKA is expected to produce Exabytes of data per year. Processing this data efficiently requires understanding the scalability of the software performance to facilitate scheduling and resource allotment. Overall, these results will help guide algorithm development in radio astronomy, as well as be useful to schedule future processing jobs and optimize resource usage.

Appendix A. Calibration Solutions for the sky model tests

The output of the calibration step is a data set corrected for direction independent effects, as well as a set of calibration solutions. Figures A.21 and A.22 show the calibration solutions for core station obtained when calibrating with sky models with minimum flux cutoffs of 0.05, 0.3, 0.8 and 1.5Jy.

Appendix B. Parametric model parameters and fit accuracy

In this section, we note the uncertainties to the models fit in Equations 1-5.

Appendix B.1. Fits quality of run time vs input size model

The models of the processing time vs input size were fit as a linear regression. In this work we present such models for the gsmcal_solve, gsmcal_apply, dpppconcat, predict_ateam and ateamcliptar, the five slowest steps. The resulting models, calculated by the `scipy linregress`(Jones et al., 2001–) routine, are shown in Equation 1. We present the R^2 values, P values and standard error below, in Table B.5.

`prefactor` step	R^2	P value	Standard Error
predict_ateam	0.996	0	1.92×10^{-10}
ateamcliptar	0.979	0	3.94×10^{-11}
dpppconcat	0.999	1.2×10^{-128}	1.78×10^{-10}
gsmcal_solve <=16GB	0.995	3.12×10^{-75}	6.80×10^{-9}
gsmcal_solve >16GB	0.951	7.07×10^{-40}	1.58×10^{-8}
gsmcal_apply	0.989	5.6×10^{-82}	3.12×10^{-10}

Table B.5: Fit parameters for the models in Equation 1.

Appendix B.2. Fit of run time vs calibration model flux cutoff

The run time vs Flux cutoff model shown in Equation 2 is defined by the equation $y = a \cdot x^{-k}$ and two parameters, a and k. The covariance matrix for these two parameters is shown in Equation B.7. The standard deviation for the fit of the parameters a and k is 26.134 and 7.624×10^{-3} respectively.

Appendix B.3. Fit of the NCPU model

The covariance matrix for the fit parameters of equation 3, a and k in $y = a + \frac{k}{N}$ are shown in Equation B.8. The standard deviation of the fits for a and k are 13.11 and 48.20 respectively.

$$\begin{bmatrix} 6.83 \times 10^{2} & -1.94 \times 10^{-1} \\ -1.94 \times 10^{-1} & 5.81 \times 10^{-5} \end{bmatrix} \quad \text{(B.7)}$$

Equation B.7: The covariance matrix of the parameters in model in Equation 2.

$$\begin{bmatrix} 171.94 & -504.11 \\ -504.11 & 2322.95 \end{bmatrix} \quad \text{(B.8)}$$

Equation B.8: The covariance matrix for the parameters for the model predicting run time vs Number of CPUs used, shown in Equation 3.

Appendix B.4. Fit for the queuing time model

The statistics of the model fit parameters for the queuing time model (Equation 4) are in Table B.6. The queuing model is fit to the 75th percentile of the queuing times for each parameter step. Since this results in a single number for each step, the model's P values are larger than the models from the other sections.

Value of $\mathcal{N}$	R^2	P value	Standard Error
$\mathcal{N} \leq 4$	0.382	0.381	37.086
$\mathcal{N} > 4$	0.986	0.075	86.293

Table B.6: Goodness of fit parameters for the model in Equation 4. Since the model is split into two parts, we treat each section as a single linear model.

Appendix B.5. Fit of the download and extract model

Equation B.9 shows the covariance matrix for the two parameters a and k, $y = a \times 10^{k}$ with the best fit values shown in Equation 5. The standard deviations of the fits for a and k are 0.016 and 0.068 respectively.

Acknowledgements

APM would like to acknowledge the support from the NWO/DOME/IBM programme "Big Bang Big Data: Innovating ICT as a Driver For Astronomy", project #628.002.001.

This work was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative through grant e-infra 160022 & 160152.

References

B. J. Barnes, B. Rountree, D. K. Lowenthal, J. Reeves, B. De Supinski, and M. Schulz. A regression-based approach to scalability prediction. In *Proceedings of the 22nd annual international conference on Supercomputing*, pages 368–377. ACM, 2008.

A. Calotoiu, T. Hoefler, M. Poke, and F. Wolf. Using automated performance modeling to find scalability bugs in complex codes. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 45. ACM, 2013.

$$\begin{bmatrix} 2.53 \times 10^{-4} & 1.08 \times 10^{-3} \\ 1.08 \times 10^{-3} & 4.60 \times 10^{-3} \end{bmatrix} \quad \text{(B.9)}$$

Equation B.9: The covariance matrix for the parameters for the model for Download and Extract time, shown in Equation 5.

L. Carrington, A. Snavely, and N. Wolter. A performance prediction framework for scientific applications. *Future Generation Computer Systems*, 22(3):336–346, 2006.

A. Castiglione, M. Gribaudo, M. Iacono, and F. Palmieri. Exploiting mean field analysis to model performances of big data architectures. *Future Generation Computer Systems*, 37:203 – 211, 2014. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2013.07.016. URL http://www.sciencedirect.com/science/article/pii/S0167739X13001611. Special Section: Innovative Methods and Algorithms for Advanced Data-Intensive Computing Special Section: Semantics, Intelligent processing and services for big data Special Section: Advances in Data-Intensive Modelling and Simulation Special Section: Hybrid Intelligence for Growing Internet and its Applications.

T. J. Dijkema. LOFAR Imaging Cookbook. Available at http://www.astron.nl/sites/astron.nl/files/cms/lofar_imaging_cookbook_v19.pdf, 2017.

A. Horneffer, W. Williams, T. Shimwell, C. Roskowinski, D. Rafferty, A. Mechev, M. Dzieak, S. Bourke, T. J. Dijkema, M. Hardcastle, and J. Sabater. apmechev/prefactor: LOTSS Data Release 1, Nov. 2018. URL https://doi.org/10.5281/zenodo.1487962.

H. Intema, P. Jagannathan, K. Mooley, and D. Frail. The gmrt 150 mhz all-sky radio survey-first alternative data release tgss adr1. *Astronomy & Astrophysics*, 598:A78, 2017.

E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL http://www.scipy.org/. [Online; accessed January 23, 2019].

S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan. An analysis of traces from a production mapreduce cluster. In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 94–103, May 2010. doi: 10.1109/CCGRID.2010.112.

M. Kuperberg, K. Krogmann, and R. Reussner. Performance prediction for black-box components using reengineered parametric behaviour models. In *International Symposium on Component-Based Software Engineering*, pages 48–63. Springer, 2008.

C. Marco, C. Fabio, D. Alvise, G. Antonia, G. Alessio, G. Francesco, M. Alessandro, M. Elisabetta, M. Salvatore, and P. Luca. The glite workload management system. In *Journal of Physics: Conference Series*, volume 219, page 062039. IOP Publishing, 2010.

A. Mechev, J. B. R. Oonk, A. Danezi, T. W. Shimwell, C. Schrijvers, H. Intema, A. Plaat, and H. J. A. Rottgering. An Automated Scalable Framework for Distributing Radio Astronomy Processing Across Clusters and Clouds. In *Proceedings of the International Symposium on Grids and Clouds (ISGC) 2017, held 5-10 March, 2017 at Academia Sinica, Taipei, Taiwan (ISGC2017). Online at https://pos.sissa.it/cgi-bin/reader/conf.cgi?confid=293, id.2*, page 2, Mar. 2017.

H. Sanjay and S. Vadhiyar. Performance modeling of parallel applications for grid scheduling. *Journal of Parallel and Distributed Computing*, 68(8):1135 – 1145, 2008. ISSN 0743-7315. doi: https://doi.org/10.1016/j.jpdc.2008.02.006. URL http://www.sciencedirect.com/science/article/pii/S0743731508000464.

T. Shimwell, H. Röttgering, P. N. Best, W. Williams, T. Dijkema, F. De Gasperin, M. Hardcastle, G. Heald, D. Hoang, A. Horneffer, et al. The LOFAR Two-metre Sky Survey-I. Survey description and preliminary data release. *Astronomy & Astrophysics*, 598: A104, 2017.

T. W. Shimwell, C. Tasse, M. J. Hardcastle, A. P. Mechev, W. L. Williams, P. N. Best, H. J. A. Röttgering, J. R. Callingham, T. J. Dijkema, F. de Gasperin, D. N. Hoang, B. Hugo, M. Mirmont, J. B. R. Oonk, I. Prandoni, D. Rafferty, J. Sabater, O. Smirnov, R. J. van Weeren, G. J. White, M. Atemkeng, L. Bester, E. Bonnassieux, M. Brüggen, G. Brunetti, K. T. Chyży, R. Cochrane, J. E. Conway, J. H. Croston, A. Danezi, K. Duncan, M. Haverkorn, G. H. Heald, M. Iacobelli, H. T. Intema, N. Jackson, M. Jamrozy, M. J. Jarvis, R. Lakhoo, M. Mevius, G. K. Miley, L. Morabito, R. Morganti, D. Nisbet, E. Orrú, S. Perkins, R. F. Pizzo, C. Schrijvers, D. J. B. Smith, R. Vermeulen, M. W. Wise, L. Alegre, D. J. Bacon, I. M. van Bemmel, R. J. Beswick, A. Bonafede, A. Botteon, S. Bourke, M. Brienza, G. Calistro Rivera, R. Cassano, A. O. Clarke, C. J. Conselice, R. J. Dettmar, A. Drabent, C. Dumba, K. L. Emig, T. A. Enßlin, C. Ferrari, M. A. Garrett, R. T. Génova-Santos, A. Goyal, G. Gürkan, C. Hale, J. J. Harwood, V. Heesen, M. Hoeft, C. Horellou, C. Jackson, G. Kokotanekov, R. Kondapally, M. Kunert- Bajraszewska, V. Mahatma, E. K. Mahony, S. Mandal, J. P. McKean, A. Merloni, B. Mingo, A. Miskolczi, S. Mooney, B. Nikiel- Wroczyński, S. P. O'Sullivan, J. Quinn, W. Reich, C. Roskowiński, A. Rowlinson, F. Savini, A. Saxena, D. J. Schwarz, A. Shulevski, S. S. Sridhar, H. R. Stacey, S. Urquhart, M. H. D. van der Wiel, E. Varenius, B. Webster, and A. Wilber. The LOFAR Two-metre Sky Survey - II. First data release. *arXiv e-prints*, art. arXiv:1811.07926, Nov. 2018.

O. Smirnov and C. Tasse. Radio interferometric gain calibration as a complex optimization problem. *Monthly Notices of the Royal Astronomical Society*, 449(3):2668–2684, 2015.

C. Tasse, B. Hugo, M. Mirmont, O. Smirnov, M. Atemkeng, L. Bester, M. Hardcastle, R. Lakhoo, S. Perkins, and T. Shimwell. Faceting for direction-dependent spectral deconvolution. *Astronomy & Astrophysics*, 611:A87, 2018.

J. Templon and J. Bot. The dutch national e-infrastructure. To appear in Proceedings of Science edition of the International Symposium on Grids and Clouds (ISGC) 2016 13-18 March 2016, Academia Sinica, Taipei, Taiwan, Oct. 2016. URL `https://doi.org/10.5281/zenodo.163537`.

G. van Diepen and T. J. Dijkema. DPPP: Default Pre-Processing Pipeline. Astrophysics Source Code Library, Apr. 2018.

M. Van Haarlem, M. Wise, A. Gunst, G. Heald, J. McKean, J. Hessels, A. De Bruyn, R. Nijboer, J. Swinbank, R. Fallows, et al. LOFAR: The low-frequency array. *Astronomy & astrophysics*, 556:A2, 2013.

R. Van Weeren, W. Williams, M. Hardcastle, T. Shimwell, D. Rafferty, J. Sabater, G. Heald, S. Sridhar, T. Dijkema, G. Brunetti, et al. LOFAR facet calibration. *The Astrophysical Journal Supplement Series*, 223(1):2, 2016.

W. Williams, R. Van Weeren, H. Röttgering, P. Best, T. Dijkema, F. de Gasperin, M. Hardcastle, G. Heald, I. Prandoni, J. Sabater, et al. LOFAR 150-MHz observations of the Boötes field: catalogue and source counts. *Monthly Notices of the Royal Astronomical Society*, 460(3):2385–2412, 2016.

C. Witt, M. Bux, W. Gusew, and U. Leser. Predictive performance modeling for distributed computing using black-box monitoring and machine learning. *CoRR*, abs/1805.11877, 2018.

Z. Xu, X. Zhang, and L. Sun. Semi-empirical multiprocessor performance predictions. *Journal of Parallel and Distributed Computing*, 39(1):14 – 28, 1996. ISSN 0743-7315. doi: https://doi.org/10.1006/jpdc.1996.0151. URL `http://www.sciencedirect.com/science/article/pii/S0743731596901513`.

L. T. Yang, X. Ma, and F. Mueller. Cross-platform performance prediction of parallel applications using partial execution. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 40. IEEE Computer Society, 2005.