Parametric models of SDP compute requirements

R. Bolton, P.C. Broekema, T.J. Cornwell, G. van Diepen, C. Hollitt, M. Johnston-Hollitt, L. Levin Preston, Á. Mika, R. Nijboer, B. Nikolic, S. Salvini, H. Rampadarath, A. Scaife, B. Stappers, P. Wortmann

Table of Contents

Summary

In this document the processing requirements for the Science Data Processor (SDP) compute platform are modelled. The parametric models are implemented in an iPython notebook which is described in an accompanying document [RD01]. The notebook can then be used to inform the costing of the SDP element [AD05].

The various pipelines that need to run on the SDP compute platform are described in the design document [RD02]. Each pipeline consists of a number of pipeline components, we model these components and the overall combination of them as they are used to form pipelines. The combination of these modelled quantities then leads to the design equations for the compute platform.

The computational cost of the calibration sub- components is modelled in this document, however, a detailed costing of the Calibration approach remains uncertain depending on a final Calibration and Processing approach. Calibration and correction of Direction Dependent Effects is an active area of research. Currently we are only able to model the cost of simple approaches based on facetting, which are unlikely to be the final word on direction dependent calibration and correction.

As a note of warning: the analysis here is focused on the compute rates required for the pipelines and components. Interpreting these to get to a system design is challenging as computational efficiency and the effects of data transfer limitations need to be considered.

1 Introduction

1.1 Purpose of the Document

The purpose of this document is to provide the modelling equations for the costing and scaling of the SDP compute platform, and to place these within the pipeline frameworks where the processing will occur. It is accompanied by an iPython implementation of the modelling equations for the calibration and imaging components. We have implemented the Non-Imaging Processing pipeline cost estimates in a spreadsheet. The design of the pipelines and their components is considered in RD02.

1.2 Scope of the Document

This document sets out the assumptions used in the models and gives the basic equations relating the parameters with required performance for all pipeline components currently under consideration. The initial analysis and the key outputs of the models is considered in RD01 and RD03. This document is concise and much of the underlying analyses and derivations can be found in supporting documents RD04, RD05, RD06, RD07, RD08, RD09, and RD10.

The version of this document that was submitted for milestone M7 (input for re-baselining) only contained the analysis of the imaging component. The current version of this document now also includes the analysis for the other components, including Non-Imaging Processing, and calibration. The main conclusion still holds true, though, the imaging component is computationally the most demanding component for the SDP SKA1 processing. For this reason the iPython implementation [RD01] still focuses on the imaging component. The other difference with the earlier version is that the current version is focusing on the modelling only and all the results (i.e. the “plugging in of numbers”) are moved to RD03. The following steps are not yet included in the current parametric modelling:

- Full Direction Dependent Calibration (only a very simple approach to DD Calibration can currently be modelled),
- LSM selection and update costs,
- The computation of the u, v, w coordinates if these are not stored.

Note that all FLOPS mentioned in this document are "required (or real) FLOPS" since our model adds up the number of operations needed to perform various tasks and derives compute rate estimates (in FLOPS) by dividing this operations total by the total time available for processing. Double precision is assumed throughout; identifying areas where single precision numbers could be used is part of our future development.

2 The Non-Imaging Pipelines

The Non-Imaging Components of the SDP that connect to Pulsar Search (PSS) and Pulsar Timing (PST) in CSP are trivially scalable from prototype to SKA-sized versions. For PSS (searching), despite the fact that the SDP post-processing brings together information from all tied-array beams being searched to reject spurious candidates, this information is in the form of candidate lists, not data, and there exist good constraints on our expectations of the numbers of candidates supplied by PSS. Therefore we anticipate no issues in scaling prototypes of the PSS post-processor in SDP. For PST (timing), the SDP post-processing per pulsar per observation is entirely independent of all other observations and so depending on the configuration each parallel observation of separate pulsars can either go to different nodes or even multiple pulsars to the same node. There are no difficulties in projecting how this scales from the algorithm operation on one example dataset to the full SKA timing experiment. Likewise, the Single Pulse / Fast Transients pipeline is also expected to be readily scalable.

2.1 Pulsar Search

The pulsar search pipeline will receive pulsar candidates from CSP in the form of a detected, folded data cube in PSRFITS format, accompanied by associated metadata in ASCII format. A description of the pulsar search parameters and the corresponding expected values are listed in Table 1. Each data cube will be of size $M_{\rm cand} = N_{\rm f} \times N_{\rm bin} \times N_{\rm subint} \times N_{\rm byte}$ = 1.049 GB, and each metadata file will be of size $M_{\rm metadata}$ ∼10 kB. In total per observation with SKA-Mid, that corresponds to $M_{\rm input} = N_{\rm beam} \times N_{\rm cand} \times (M_{\rm cand} + M_{\rm metadata})$ = 1.59 TB.

Parameter	Symbol	Expected value
Number of beams	$N_{\rm beam}$	1500 (SKA-Mid) 500 (SKA-Low)
Number of candidates per beam	$N_{\rm cand}$	1000
Number of frequency channels	$N_{\rm f}$	128
Number of pulse profile bins	$N_{\rm bin}$	128
Number of subintegrations	$N_{\rm subint}$	64
Typical observation length	$t_{\rm obs}$	600 seconds

Table 1: Pulsar search parameters

The processing pipeline for pulsar search consists of five major steps, as described below. All expected numbers of FLOP and RAM requirements are summarised in Table 2.

1. **Merge OCLDs (Optimal Candidate List and Data) from multiple beams.** This step involves coincidence matching of candidates from different beams. This is a sort-like algorithm, and using `quicksort` as a basis, the estimated number of flops is $1.4 \times N_{\rm beam} \times N_{\rm cand} \times N_{\rm param} \times \log_2(N_{\rm beam} \times N_{\rm cand} \times N_{\rm param})$, where $N_{\rm param}$ is the number of parameters to sort by (i.e. spin period, DM, R.A. and Dec.). This step will remove candidates associated with RFI and duplicate detections of real candidates. The remaining pipeline will only need to be run on the unique candidates that survive the coincidence matching, expected to be $\sim 10\%$ of the total. Hence, going forward the total number of candidates are $N_{\rm unique} = 0.1 \times N_{\rm beam} \times N_{\rm cand}$.

2. **Generation and extraction of candidate heuristics.** This step is the most computationally demanding of the pulsar search pipeline. The heuristic value calculation is estimated to require a maximum of 2 Gflop per $N_{\rm unique}$ candidate, and will make up the majority of the processing requirement for pulsar search. Details of the experimental approach used to arrive at this number is described in RD34. All heuristic values calculated in this step will be appended to the metadata file for each candidate.

3. **Candidate classification and machine learning.** Determining the computational cost of the machine learning algorithm is very difficult as it is strongly dependent on the algorithm that is being implemented. Assuming that the machine learning algorithm The Gaussian Hellinger Very Fast Decision Tree is being used, this step would require 3.5 Mflop per $N_{\rm unique}$ candidate. More details on this calculation can be found in RD34.

4. **Candidate selection.** This step simply involves checking a score given by the candidate classification to a detection threshold, and hence will only require 1 flop per $N_{\rm unique}$ candidate.

5. **Alert generation** The last step creates an alert to be sent to TM for each candidate that are selected in the previous step. This requires very low operation counts as well, estimated to TBD flop per selected candidate.

Task	Total FLOP [Gflop]	RAM required [GB]
Merge OCLDs from all beams	0.19	30
Generation and extraction of heuristics	300,000	317.6
Classification and machine learning	525	3.6
Candidate selection	0.00015	3.6
Alert generation	TBD	TBD
Total Gflop	300,525.62	–
Total GFLOPS (Gflop per second)	500.88	–

Table 2: Pulsar search processing summary. The RAM requirements assume that the data not needed in RAM in a specific step, is stored on the local disk or some other media.

The output from the pulsar search pipeline will consist of metadata files for all input candidates, and data cubes for all unique candidates. Hence, the total output data size per observation will be $M_{\rm output} = N_{\rm unique} \times M_{\rm cand} + N_{\rm beam} \times N_{\rm cand} \times M_{\rm metadata}$ = 172.3 GB.

2.2 Single Pulse / Fast Transients

The fast transient pipeline will receive single pulse candidates from CSP in the form of a dedispersed filterbank data file in PSRFITS format, accompanied by associated metadata in ASCII format. A description of the fast transient parameters and the corresponding expected values are listed in Table 3.

Each data file will be of size $M_{\rm burst} = N_{\rm samp} \times N_{\rm f} \times N_{\rm pol}$ = 2.62 MB, and each metadata file will be of size $M_{\rm metadata}$ ∼10 kB. Assuming $N_{\rm burst}$ are detected in each beam and sub-integration, that corresponds to $M_{\rm input} = N_{\rm burst} \times N_{\rm beam} \times (M_{\rm burst} + M_{\rm metadata})$ = 3.93 GB of input data every $t_{\rm subint}$ seconds.

The processing pipeline for fast transient search consists of five major steps, identical to the steps in the pulsar search pipeline (see section 2.1). All expected numbers of flop and RAM requirements are summarised in Table 4, assuming that $N_{\rm unique} = 20$ bursts survive coincidence matching every $t_{\rm subint}$ seconds.

The output from the fast transient pipeline will consist of metadata files for all input single pulse candidates, and data files for all unique single pulse candidates. Hence, the total output data size per

Parameter	Symbol	Expected value
Number of beams	N_{beam}	1500 (SKA-Mid) 500 (SKA-Low)
Number of candidate bursts per beam and sub-integration	N_{burst}	1
Number of samples	N_{samp}	640
Number of frequency channels	N_{f}	1024
Number of polarisations	N_{pol}	4
Sub-integration length	t_{subint}	10 seconds
Typical observation length	t_{obs}	600 seconds

Table 3: Fast transient pipeline parameters

Task	Total flops per t_{subint} [Gflop]	RAM required [GB]
Merge OCLDs from all beams	1.1×10^{-4}	0.03
Generation and extraction of heuristics	40	0.11
Classification and machine learning	0.07	4.8×10^{-4}
Candidate selection	2×10^{-8}	4.82×10^{-4}
Alert generation	TBD (but very small)	TBD
Total Gflop	40.07	–
Total GFLOP (Gflop per second)	4.0	–

Table 4: Fast Transient processing summary. The RAM requirements assume that the data not needed in RAM in a specific step, is stored on the local disk or some other media.

observation will be $M_{\mathrm{output}} = (N_{\mathrm{unique}} \times M_{\mathrm{burst}} + N_{\mathrm{beam}} \times N_{\mathrm{burst}} \times M_{\mathrm{metadata}}) \times t_{\mathrm{obs}}/t_{\mathrm{subint}}$ = 4.05 GB.

2.3 Pulsar Timing

The pulsar timing pipeline will receive coherently dedispersed, detected pulsar data cubes from CSP in PSRFITS format, which include all associated metadata. A description of the pulsar timing parameters and the corresponding expected values are listed in Table 5. Here the expected values are given for three different observing modes: ordinary pulsars, millisecond pulsars (MSPs) and Pulsar Timing Array (PTA) pulsars. A maximum of 16 tied-array beams will be available for pulsar timing. Since the beams are independent of each other, a combination of the different observing modes can be used simultaneously. This also implies that each pulsar can be processed completely independent of the others, and hence all 16 pulsars do not need to be processed on the same compute node or even the same compute island. Each input data cube will be of size $M_{\mathrm{pulsar}} = N_{\mathrm{bin}} \times N_{\mathrm{f}} \times N_{\mathrm{subint}} \times N_{\mathrm{pol}} \times N_{\mathrm{byte}}$.

The processing pipeline for pulsar timing consists of six major steps, as described below. All expected maximum numbers of flop and RAM requirements are summarised in Table 6.

- **Radio Frequency Interference mitigation** The next step involves removing any parts of the data cube that have been affected by RFI. The exact procedure to mitigate the RFI is still under investigation, , but estimates can be made assuming a commonly used algorithm for removing affected frequency channels ("channel zapping") and time samples ("lawn mowing"). Channel zapping is carried out on the bandpass of each sub-integration and compared to a median smoothed version of the bandpass. In this case, the flop count is a combination of that required to create the bandpass and that required for median smoothing, hence $(2 \times N_{\mathrm{bin}} \times N_{\mathrm{pol}} + N_{\mathrm{f}} \times N_{\mathrm{f,smooth}}^{1/2}) \times N_{\mathrm{subint}}$, where $N_{\mathrm{f,smooth}}$ is the number of frequency channels included in the smoothing window. In a similar way, the profile lawn mowing is used to replace RFI-affected phase bins with the local median

Parameter	Symbol	Expected value		
		Ordinary pulsars	MSPs	PTAs
Maximum number of beams	N_{MAXbeam}	16	16	16
Average number of beams	N_{beam}	8	3	3
Number of frequency channels	N_{f}	512	512	4096
Number of pulse profile bins	N_{bin}	2048	4096	4096
Number of subintegrations	N_{subint}	180	180	180
Number of polarisations	N_{pol}	4	4	4
Number of bytes per pixel	N_{byte}	4	4	4
Typical observation length [seconds]	t_{obs}	1800	1800	1800
Data size per pulsar [GB]	M_{pulsar}	3.02	6.04	48.32
Average data size per observation [GB]	M_{obs}	24.2	18.1	145.0
Maximum data size per observation [GB]	$M_{\mathrm{obs;MAX}}$	48.3	96.6	773.1

Table 5: Pulsar timing parameters

plus some random noise. Here, each sub-integration, frequency channel and polarisation of the pulse profile is compared to a median smoothed version of the same profile. The number of flops required for this analysis is $N_{\mathrm{f}} \times N_{\mathrm{subint}} \times N_{\mathrm{pol}} \times N_{\mathrm{bin}} \times N_{\mathrm{bin:smooth}}^{1/2}$, where $N_{\mathrm{bin:smooth}}$ is the number of phase bins included in the smoothing window.

- **Calibration** After the data cubes have been cleaned from RFI, they are passed on to the calibration step. Provided that the calibration solutions are given as input from TM, estimations of flops required for both flux calibration and polarisation calibration scale linearly with the total number of phase bins. That is for flux calibration the estimated number of flops is $N_{\mathrm{bin}} \times N_{\mathrm{f}} \times N_{\mathrm{subint}} \times N_{\mathrm{pol}}$, and for polarisation calibration $N_{\mathrm{bin}} \times N_{\mathrm{f}} \times N_{\mathrm{subint}} \times 20$.

- **Averaging and archive product generation** Thus far the pipeline has been using full-resolution data cubes, since RFI mitigation and calibration greatly benefit from high resolution data. The following steps require higher signal-to-noise values rather than high resolution, and therefore this step produces a partly averaged data cube to be used through the remaining steps. This branch also produces three additional versions of averaged data cubes, which will be sent to the Long-term Preservation and saved for future post-processing. The output consists of the following averaged data cubes in PSRFITS format:

  - Summed over the entire frequency band
  - Summed over all sub-integrations
  - Summed over the entire frequency band and all sub-integrations
  - Summed over part of the frequency band and part of the sub-integrations

  The estimated number of flops required for creating each of these data cubes is $N_{\mathrm{sum}} \times N_{\mathrm{bin}} \times N_{\mathrm{f}} \times N_{\mathrm{subint}} \times N_{\mathrm{pol}}$, where N_{sum} is the number of different sums that need to be carried out.

- **Time of arrival determination** The partly averaged data cube created in the last step, will be used for time-of-arrival (ToA) determination. It will determine the ToAs by cross correlating the current observation to a pulsar-specific template provided from TM. The flop count in this step can be estimated as $N_{\mathrm{bin}}^2 \times N_{\mathrm{f}} \times N_{\mathrm{subint}}$.

- **Residual determination and model update** This branch will use the currently best timing model to compute expected arrival times based on the model. It will then compare the expected arrival times to the observed arrival times passed on from the last step in the pipeline and generate

timing residuals as the difference between model and observation, and update the model of the pulsar. The flop count for this step is TBD, but will be small in the context of the full pipeline.

Task	Total flop per pulsar [Gflop]	RAM required [GB]
Radio Frequency Interference mitigation	109.3	48.3
Calibration	72.5	96.6
Averaging and archive product generation	120.8	48.7
Time of arrival determination	193.3	0.38
Residual determination and model update	TBD (but small)	0.015
Total Gflop	495.9	–
Total GFLOPS (Gflop per second)	0.28	–

Table 6: Pulsar Timing processing summary. Maximum numbers required per pulsar observed, assuming a PTA pulsar. 16 pulsars can be observed simultaneously, but processed independently. The RAM requirements assume that the data not needed in RAM in a specific step, is stored on the local disk or some other media.

3 The Imaging Pipelines

The SDP consists of a number of pipelines and processing stages, see RD02. The main visibility processing pipelines are shown in Figure 1 below.

Visibility data flows from the correlator into the Ingest pipeline which performs pre-processing and buffering. Buffered visibilities might get re-processed multiple times at this point. At observation time visibilities will be imaged right away by the FastImg pipeline in order to identify slow transients, and the RCAL self-calibration pipeline will do a first pass at solving calibration for supporting pulsar experiments. As these pipelines must keep up with latency requirements, the use of iterative visibility processing is limited here. Instead calibration and fast imaging will be performed using a constant LSM.

The remaining pipelines will have an entire observing track available from the the Buffer, which is processed asynchronously according to compute resource availability while further observations get under way. This allows for iterative self-calibration by the ICAL pipeline. After calibration has been solved, Data Product preparation pipelines (DPrepA, DPrepB, DPrepC) will produce the actual scientific results. See Table 7.

Pipeline	**Description**
Ingest	Ingest and condition visibilities from the correlator (Receive and Pre-process)
RCAL	Produce calibration solutions using existing Sky Model
ICAL	Produce calibration solutions by iterating on Sky Model (i.e. self calibration)
DPrepA	Produce deconvolved continuum Taylor term images in Stokes I
DPrepB	Produce coarse continuum image cubes in I,Q,U,V
DPrepC	Produce fine spectral resolution image cubes un I,Q,U,V
DPrepD	Produce calibrated, averaged (In time and freq) visibility data
FastImg	Detect variations in sky at high cadence ($\sim$ 1 second) by searching for significant deviations relative to the Global Sky Model. Produces a time ordered catalogue of image-plane variations.

Table 7: Imaging pipeline definitions

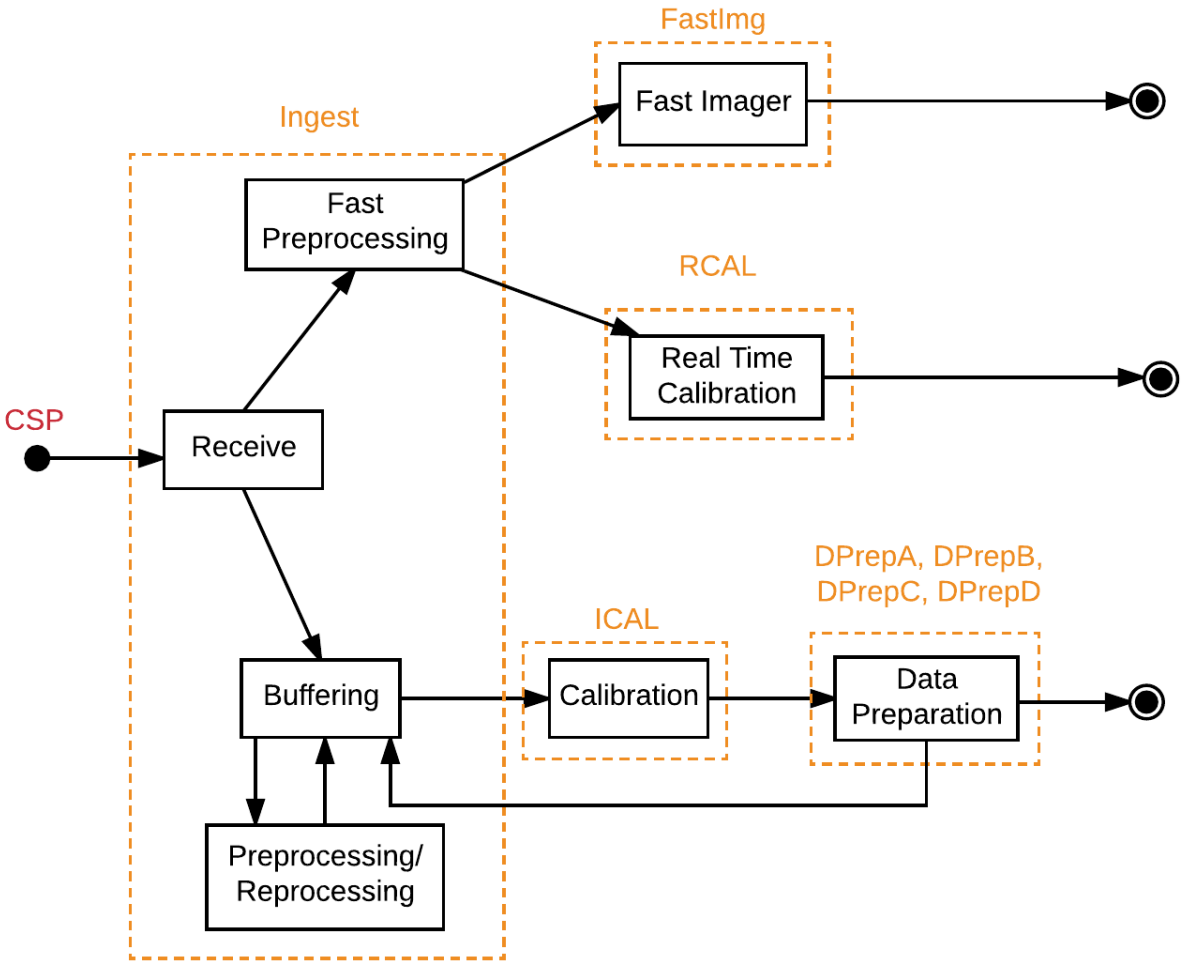

Figure 1: The high level architecture for Imaging along with the Imaging steps in the Performance Model, with the associated pipelines (names and boundaries in red). The ingest pipeline includes Receive, Buffering and both the Fast and Buffered Pre-Processing pipelines. The three final imaging pipelines are: DPrepA - Continuum imaging, DPrepB, - coarse channel spectral imaging, and DPrepC - fine channel spectral imaging. DPrepD simply prepares visibility data for delivery (by correcting it and averaging it), if it has been requested.

3.1 Pipelines Analysis

In estimating the compute rates of Pipelines, we have to amortise costs over pipelines both in Real-Time and Batch Processing. The consequence is that the derived processing rates are averages only, and not actual lower bounds for the required processing capability. For instance, it is no problem if batch processing takes longer than the observation time as long as "cheaper" batch processing or even telescope idle time can make up for it.

On the other hand, cutting it too close might cause blocking while transitioning from one pipeline to the other, so a certain amount of extra capacity in storage and compute will be required to allow smooth operation of the telescope. A more detailed accounting of these effects will be possible once we have developed a more refined model for telescope and Batch Processing scheduling.

In the following sections we will discuss the components used in every pipeline or class of pipelines, describing the scaling in context. The choices made for the equations and parameter values in the Performance Model are described in the Appendices. Thus the capabilities of the model can be gleaned from reading just this section, but deeper understanding requires the appendices.

3.2 Fundamental Rates

Processing costs generally scales either with the number of (input) visibilities and therefore observation time or with the number of (output) pixels. The former is more common, so we will give cost equations for rates per observation time. As explaiend in the previous section, we expect that after armotisation this corresponds roughly to rate per pipeline *execution* time.

The unaveraged visibility data rate from the correlator $R_{\mathrm{vis,corr}}$ is given by:

$$R_{\mathrm{vis,corr}} = \frac{N_{\mathrm{beam}} N_{\mathrm{pol}} (N_{\mathrm{bl}} + N_{\mathrm{ac}}) N_{\mathrm{f,corr}}}{t_{\mathrm{dump}}} \tag{1}$$

For reference, the visibility rate from each of the two SKA1 instruments is very similar and close to 40G visibilities per second. Internally this data will get reduced somewhat as described in section C:

$R_{\mathrm{vis,full\,FoV}}$: the average visibility rate for an observation after we have used baseline-dependent averaging, with time and frequency smearing limits set to the full field of view

$R_{\mathrm{vis,facet\,FoV}}$: average visibility rate *per facet* after further coalescing of data to meet time and frequency smearing limits set be the field of view of an image facet.

This means that the total number of handled visibilities works out as:

$$N_{\mathrm{vis,corr}} = R_{\mathrm{vis,corr}} t_{\mathrm{obs}} \tag{2}$$
$$N_{\mathrm{vis,FullFoV}} = R_{\mathrm{vis,full\,FoV}} t_{\mathrm{obs}} \tag{3}$$
$$N_{\mathrm{vis,facetFoV}} = R_{\mathrm{vis,facet\,FoV}} N_{\mathrm{facet}}^2 t_{\mathrm{obs}} \tag{4}$$

However, as noted previously the rest of the document will work with rates exclusively.

Similarly, the rate of production of pixels per observation time can be calculated as:

$$R_{\mathrm{pix}} = \frac{N_{\mathrm{SelfCal}} N_{\mathrm{major}} N_{\mathrm{beam}} N_{\mathrm{pol}} N_{\mathrm{f}} N_{\mathrm{pix}}^2}{t_{\mathrm{obs}}} \tag{5}$$

assuming that image cubes (of linear size N_{pix}) are made for every imaging cycle combining data from an observation of length t_{obs}. Generally we would expect $R_{\mathrm{pix}} < R_{\mathrm{vis,corr}}$, but note that a spectral pipeline with $N_{\mathrm{f}} = 30,000$ and $N_{\mathrm{pix}} = 50,000$ could easily produce data rates greater then the incoming visibility rate.

3.3 Ingest Pipeline

The Ingest pipeline receives the visibilities from the correlator and provides various conditioning operations (which are described by RD04 in detail):

1. Receive and buffer the data from the correlator.
2. Calculate data weights from the fraction of data flagged and the autocorrelation data.
3. Flagging the visibility data for known RFI and apparent bad data.
4. Demix to remove the effect of a small number of very bright sources (called the A-Team)
5. Averaging to reduce data rate while staying within the smearing limits.
6. Apply known calibration

The post-processed visibilities output from this pipeline will be used as inputs for the FastImg, Calibration and Imaging pipelines (see figure 1). The operations are described in more detail in RD04. In Table 9, we show the scaling of the Ingest components.

3.3.1 Receive

The processing chain for the visibility data starts with the Receive component (see Figure 1). The receive component does a number of calculations: For each visibility, the weight has to be calculated from the data fraction received from CSP and from the autocorrelation values, which requires 4 floating point operations per visibility. Furthermore, the UVW coordinates (in metres) have to be calculated per antenna for every correlator dump time, and these must be available on all compute islands. The cost of this calculation is estimated at 1000 flops per UVW (i.e. per antenna, per correlator dump time). A subtraction gives the UVWs per baseline, but most likely that is done at a later stage and not taken into account here. We estimate a cost of 23 flops per visibility. The total compute rate (FLOPS) in the Receive component is given in Table 8.

The receive memory bandwidth (B_{mem}) depends on the visibility rate. Out-of-order packets are not taken into account, because such cases will occur infrequently. The input from CSP is 10 bytes per visibility, the output is 13 bytes (data, flag, weight) per visibility. This gives:

$$B_{\mathrm{mem}} = 23 R_{\mathrm{vis,corr}} \tag{6}$$

Pipeline: Receive
$C_{\mathrm{Receive}} = 4R_{\mathrm{vis,corr}} + \frac{1000 N_{\mathrm{a}} N_{\mathrm{f,distribute}} N_{\mathrm{beam}}}{t_{\mathrm{dump}}}$

Table 8: Scaling for Receive

3.3.2 Flagging

Flagging using the AOFlagger requires $N_{\mathrm{flop/vis}} = 278$ floating point operations per visibility. This gives a total flagging compute rate of:

$$C_{\mathrm{Flag}} \sim N_{\mathrm{flop/vis}} R_{\mathrm{vis,corr}} \tag{7}$$

To put this in context; the maximum visibility rate for each of SKA1-Low and Mid is very similar (just under 40 G visibilities per second) so the flagging compute rate is around 0.01 PFLOPS. The $N_{\mathrm{flop/vis}}$ can be reduced to 150 FLOPS/visibility by using a Scale Invariant Rank (RD04). This would then reduce the flagging compute rate to $\approx$ 6 TFLOPS.

3.3.3 Demix

De-mixing is a spatial projection approach to remove signals from strong sources outside the FoV (e.g. the A-Team in the Northern Sky) from the visibility data. The approach is in use for the LOFAR telescope, because a direct approach used an $O(N^3)$ solver which was too costly.

Leaving out the negligible terms[1] from the equation in section 5.1.2 of RD04, the total cost of demixing can be expressed as:

$$N_{\text{flop}} = \left(154 N_{\text{Ateam}} + 84 + \frac{N_{\text{Ateam}}^2 + (33 + 24 N_{\text{it}}) N_{\text{Ateam}} + 64}{N_{t,\text{av}} N_{f,\text{av}}} \right) R_{\text{vis,corr}} \tag{8}$$

This gives the per visibility cost of demixing as:

$$N_{\text{flop/vis}} = \left(154 N_{\text{Ateam}} + 84 + \frac{N_{\text{Ateam}}^2 + (33 + 24 N_{\text{it}}) N_{\text{Ateam}} + 64}{N_{t,\text{av}} N_{f,\text{av}}} \right) \tag{9}$$

$N_{t,\text{av}}$ and $N_{f,\text{av}}$ are the number of time and frequency samples that are averaged together before demixing. Which for the LOFAR standard pipeline, are 5 seconds and 32 channels. N_{Ateam} = 4 (from LOFAR) and N_{it} = 50, yields for the buffered pre-processing pipeline of SKA1-Low: $N_{\text{flop}} \approx 27.9$ TFLOPS and $N_{\text{flop}} \approx 32.7$ TFLOPS for the fast pre-processing pipeline, with $N_{t,\text{av}}$ = 1 (RD04).

3.3.4 Averaging

Each averaging operation (using complex weights) will cost 8 flops per visibility going into the average. In the ingest pipeline, there is only one initial set of averaging, to reduce the visibility rate from $R_{\text{vis,corr}}$ to $R_{\text{vis,full FoV}}$ (see equation 51). (The imaging pipelines will use further averaging at the gridding step, see section 3.6.10.) The total cost is therefore

$$C_{\text{Average}} = 8 R_{\text{vis,corr}} \tag{10}$$

3.3.5 The Fast Pre-processing Pipeline

After the visibility data have been received, they have to be pre-processed before the fast calibration and imaging can be done. It consists of the flagging, de-mixing (or direct solve and subtract) and averaging.

Due to the latency requirements of the fast calibration and imaging pipelines, flagging and demixing must deliver solutions at high cadence. Hence, in the demixing equation $N_{t,\text{av}} = 1$ (see section 5.1.2 of RD04). Furthermore, it is possible to limit the processing time by limiting N_{it} in the solve loop.

3.3.6 The Buffered Pre-processing Pipeline

The Buffered Pre-processing Pipeline operates similar to the Fast Pre-processing Pipeline, but does not have latency requirements and reads the data from the buffers instead of receiving it in a streaming way. Therefore it is possible to use an extended time window in the flagging step to obtain better flagging results. Also it is possible to average in time in the de-mixing step to improve performance.

Flagging is done per baseline for a time/frequency window as large as possible. On the other hand the subtraction of bright sources needs all baselines for a limited time-frequency window. LOFAR experience has shown that a window of 64 channels and 100 time slots is sufficient to obtain very good flagging results. A transpose could be avoided by keeping all baselines of such a window in memory. This might also be possible for SKA-Low as it requires a buffer of 27 GBytes.

An important note is that if the flagging and bright source subtraction results of the Fast Pre-processing Pipelines are sufficient, it might be possible to omit the Buffered Pre-processing pipeline for some cases.

[1] For SKA1-Mid no demixing is needed, while it will be necessary for SKA1-LOW. Thus with $N_{\text{a}} = 512$, the term $121 N_{\text{a}}^{-1}$ in the original equation is negligible.

Pipelines: Ingest
Flag $278 R_{\mathrm{vis,corr}}$
Demix $\left(154 N_{\mathrm{Ateam}} + 84 + \frac{N_{\mathrm{Ateam}}^2 + (33+24N_{\mathrm{it}})N_{\mathrm{Ateam}} + 64}{N_{t,\mathrm{av}} N_{f,\mathrm{av}}}\right) R_{\mathrm{vis,corr}}$ with $N_{t,\mathrm{av}} = 1$ for fast pre-processing; $N_{t,\mathrm{av}} \sim 100$, $N_{f,\mathrm{av}} \sim 64$ for Buffered pre-processing
Average $8 R_{\mathrm{vis,corr}}$
Correct $8 N_{\mathrm{mm}} R_{\mathrm{vis,corr}}$
Extract LSM ...(not yet modelled)
QA ...(not yet modelled)

Table 9: Components and scalings for Ingest pipeline.

3.4 The Calibration Pipelines

There are two calibration pipelines: RCAL (real-time calibration) and ICAL (off-line calibration). RCAL uses an *a priori* LSM to calibrate against, thus allowing a solution within a short period of time. By these means the telescope is kept calibrated at all times. The off-line calibration is actually self-calibration: the LSM is iteratively updated to obtain both calibration and an improved LSM. ICAL thus requires continuum imaging. We have made use of some repeated sub-pipelines to improve legibility: see Figure 4. These sub-pipelines have the same structure in all pipelines.

There are two forms of calibration required for SKA: direction-independent effects (DIE) and direction-dependent effects (DDE). The former may be accommodated by applying a single correction factor to each visibility. For the DDE case, the gain is also a function of direction on the sky. As the correction is often antenna and therefore baseline specific, this correction is generally applied in UVW space by a convolution (A-projection). Calculating the convolution generally involves image-space multiplication of all direction-dependent effects at low resolution, followed by a small local Fourier Transform. There are ways that this can be combined efficiently with the gridding process, see [RD19].

For the DDE calibration, solving for the calibration parameters is difficult and time-consuming. The image-space approach can utilise a greedy algorithm to successively approximate gains to weaker and weaker sources. The convolution-space approach works with convolution differentials of the A-kernels with respect to the unknown parameters. For example, pointing errors can be derived this way.

At this point in the development of the Performance Model, only solutions for DIE are fully supported, though the correction of DDEs by A projection is modelled.

3.5 Imaging pipelines

There are three imaging pipelines: one for continuum imaging including MFS and two for spectral imaging (full and coarse resolution).

The fast imaging pipeline produces a residual image for every correlator dump. The LSM is subtracted from the visibility data so that the residual image should show transient or variable sources.

3.6 Scaling Behaviour of the Calibration and Imaging Components

Next we describe the scaling behaviour of the individual components used in the calibration and imaging components.

Figure 2: The Real Time Calibration pipeline RCAL. Visibility data are input at DFT and Snapshots. The output data are the calibration solution send to the Calibration Tables.

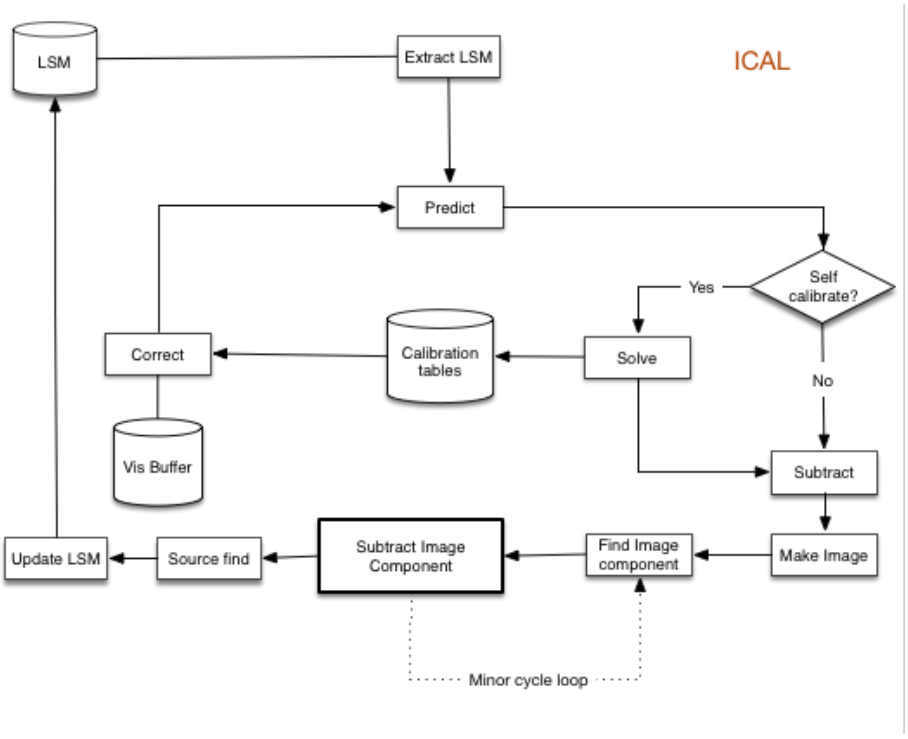

Figure 3: The Offline Calibration pipeline ICAL. The output data are the calibration solutions sent to the Calibration Tables. The LSM is updated every iteration. Each cycle around this loop constitutes a major cycle. This may occur with and without self-calibration. If self-calibration is performed then the gains are applied to the original (non-residual) visibility data and the number of major cycles is reset to zero.

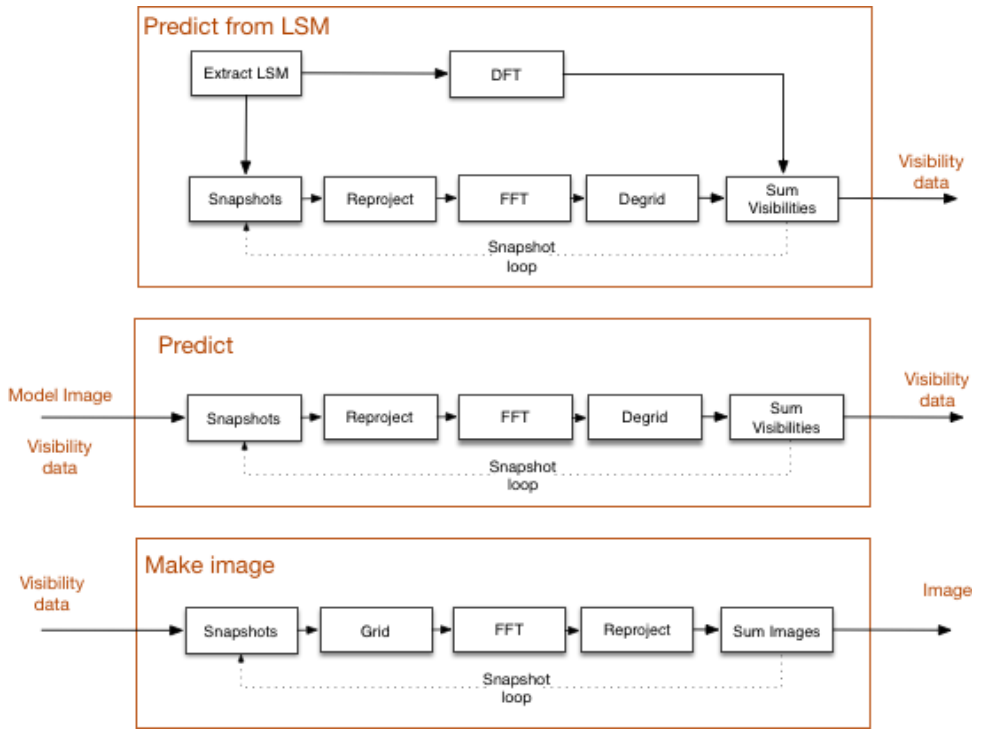

Figure 4: Predict and make image sub-pipelines.

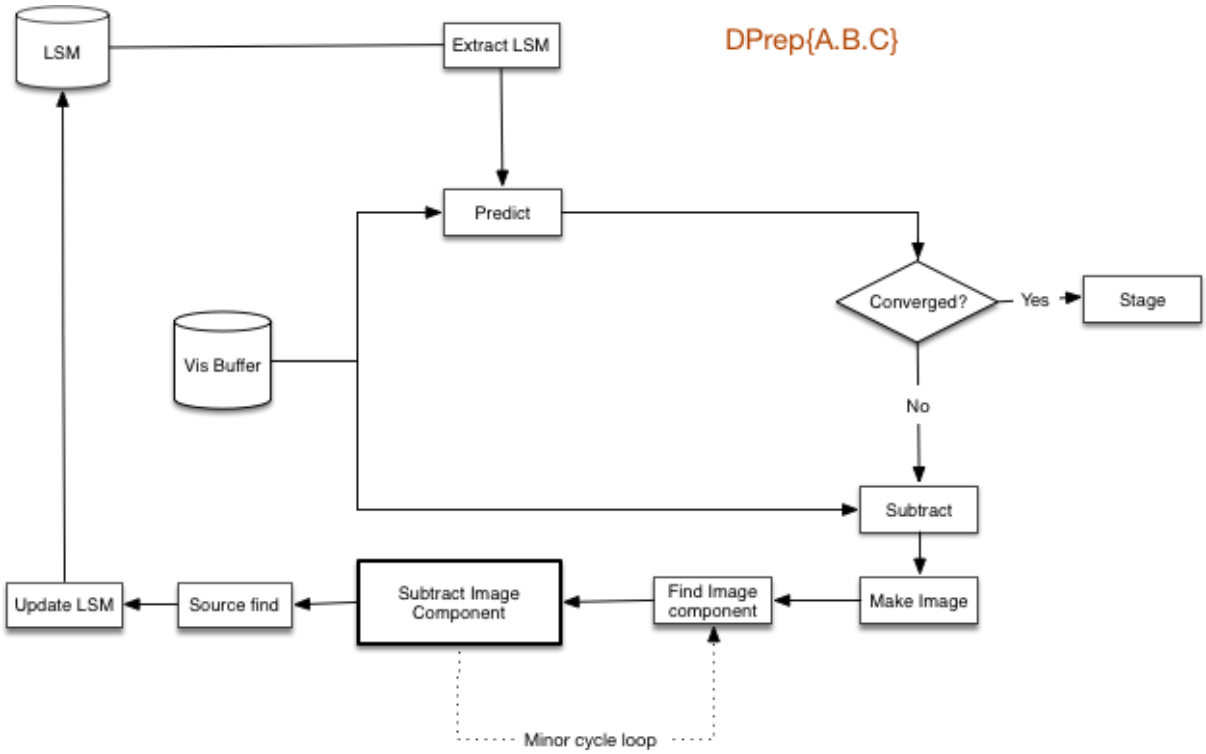

Figure 5: The Imaging pipelines DPrepA, DPredB, DPrepC. The output data are the restored images and associated products to be sent for staging.

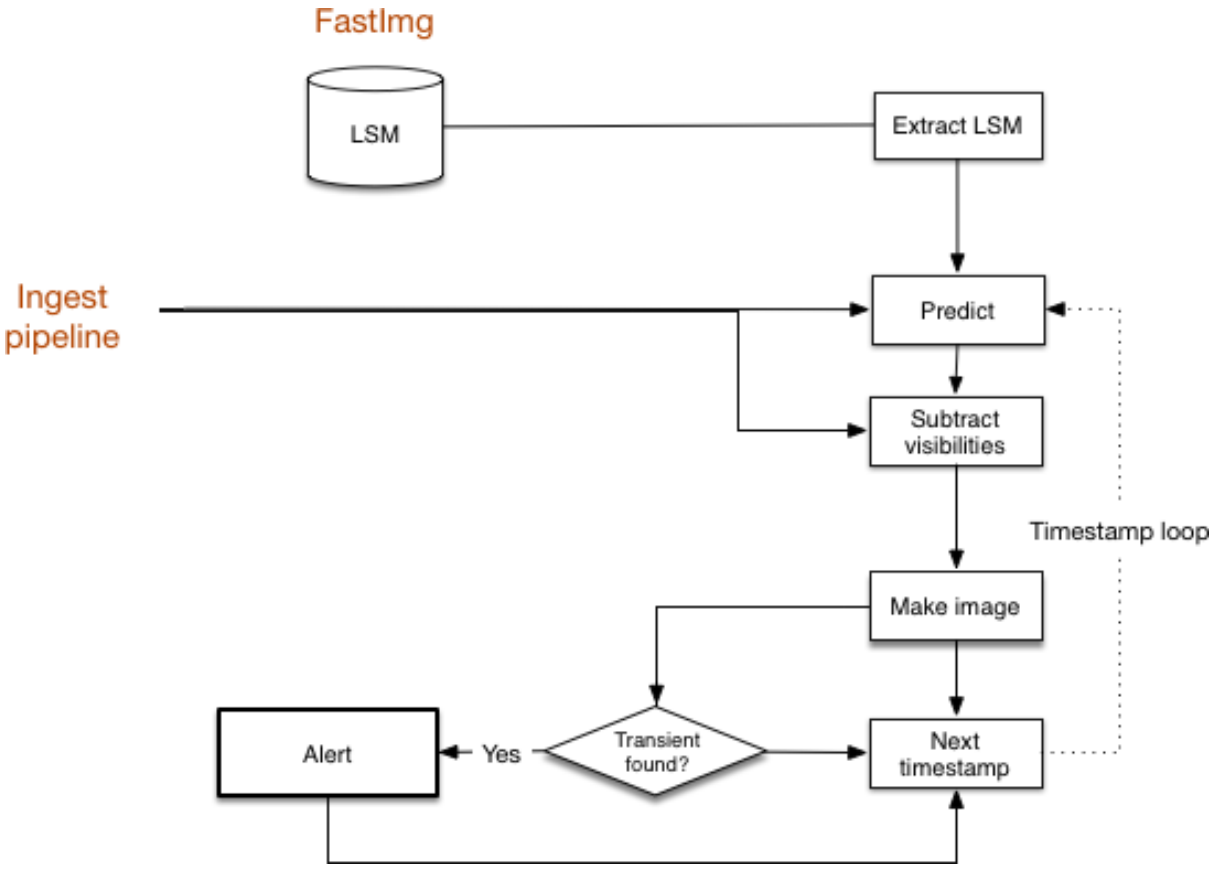

Figure 6: The Fast Imaging (FASTIMG) pipeline. This runs synchronously with observing.

The performance model does not include estimation of LSM related operations but we expect these to be fast database operations, and to not contribute significantly to the overall compute load or timing.

3.6.1 Extract LSM

Extract LSM extracts the relevant LSM from the GSM. This is a one time operation, executed at the start of observation.

3.6.2 Update LSM

Update LSM updates the LSM with the results of the deconvolution. This occurs every major cycle. It relies upon the source finder. The update is composed of two parts - discrete components that can be transformed by a DFT, and images (composed of pixels on a regular grid) containing other emission.

3.6.3 Notify GSM

Notify GSM notifies TM to an update to the GSM available. This is a one time operation, executed at the end of observation.

3.6.4 Predict via Direct Fourier Transform

The Direct Fourier Transform (DFT) is used for predicting visibilities from the discrete components in the (Local) Sky Model, and an FFT/degrid-based approach is used for the image component of the LSM.

This can be used either for comparison against the observed data in the Solver or for accurate deconvolution of (foreground) sources.

The DFT compute cost for a single 4-vector complex visibility, i.e. full polarisation, is presented in RD05. For large number of receivers, the computation cost will be dominated by the leading term in the expression, giving $32N_{\mathrm{a}}^2N_{\mathrm{source}}$ flops per 4 polarisation baseline array[2]. This is a cost per (single polarisation, single baseline) visibility of[3]:

$$N_{\mathrm{flop/vis}}^{\mathrm{predict,DFT}} \sim 16N_{\mathrm{source}} \tag{11}$$

Which gives a total compute rate for the DFT component of the predict of

$$C_{\mathrm{Predict}} = ((N_{\mathrm{SelfCal}}+1)N_{\mathrm{major}})16N_{\mathrm{source}} \times R_{\mathrm{vis,full\,FoV}} \tag{12}$$

where $R_{\mathrm{vis,full\,FoV}}$ is the visibility rate for predicting the visibilities at smearing limits set by the full field of view.

The (I)FFT cost of the Predict step is described below in 3.6.13.

3.6.5 Solve

For a general solve using StEFCal, there is a compute cost (leading term only) of

$$N_{\mathrm{flop/solution}}^{\mathrm{StEFCal}} = 48N_{\mathrm{a}}^2N_{\mathrm{it}}N_{\mathrm{pol}} \tag{13}$$

flops associated with every calibration solution (which is for all polarisations). (N_{it} is the number of iterations required, typically in the range 10 to 50). To estimate the total compute cost we must model the number of solutions that will be required.

There are multiple Direction-Independent Effects (DIE) to be solved:

Real-time gains: G The telescope is calibrated in real-time but using an *a priori* LSM. This is performed by the RCAL pipeline, for time windows of about 10 seconds. The solutions are sent to TM for the Telescope Model and also applied in the Ingest phase as soon as available. RCAL is required to be synchronous with observing, with up to 15 seconds of delay (SDP_REQ-662).

Fine time resolution gains: G The broad-band antenna-based gains may have significant structure at all scales down to the correlator dump time. Hence this must potentially be solved for every correlation dump time but only asynchronously. Self-calibration is required. The initial LSM is iteratively updated to conjunction with the estimation of the gains.

Bandpasses: B The bandpass will have time- and frequency-structure all potentially all scales. The complexity incurred is thus potentially very high, being roughly the number of channels higher than the complexity of the gain solution.

We understand how these DIE terms are to be solved well enough to calculate the resulting complexity: It is just the sum of the solution costs for all terms included in a given pipeline. The largest unknowns are the time- and frequency-scales for the different effects.

More difficult are the multiple Direction-Dependent Effects (DDE):

Instrumental Polarisation: P The instrumental polarisation is likely to be direction dependent for both Low and Mid.

Ionosphere: I "Calibration" of the ionosphere is still being investigated by LOFAR. There are multiple approaches.

Pointing errors: E Calibration of pointing errors has been demonstrated [RD18]. It is expensive and not in common use at the JVLA.

[2] This is for computing the visibilites one-sided

[3] Note for large N_{a}, $N_{\mathrm{bl}} \approx N_{\mathrm{a}}^2/2$

At the moment, we are not sufficiently comfortable of the understanding of DDE calibration to include the resulting complexity. This is concerning for Low since ionospheric calibration will nearly always be required. For Mid, the effect of the uncertainty is lower because only very deep integrations will require solution of pointing errors (or other DDEs).

In summary, modelling of RCAL includes gain solution for G, typically for each sub-band every 180s ($t^G_{sol} = 180$), and modelling of ICAL includes G, typically for sub-band at 1s ($t^G_{sol} = 1$s), and B, typically at 1 hour ($t^B_{sol} = 3600$s) intervals. We expect that both these models and associated timescales will be revised and refined with the delivery of the calibration strategy. For example, the relative calibration of the different beams should be considered. For the moment, we have assumed that beams are calibrated entirely separately. We have also not yet included calibration of direction-independent instrumental polarisation effects since this most likely to require careful examination in the construction of the calibration strategy.

The Bandpass solution (B) is assumed to have f^B_{sol} frequency terms (we do not know what value this should take, but assume 500 as a working assumption; it must depend on the calibration strategy (i.e. how bright the bandpass calibrator is or if the field itself is used as the bandpass calibrator), and on the characteristics of the receivers when built.

For the RCAL pipeline this gives a total cost in the solver of:

$$C^{\mathrm{RCAL}}_{\mathrm{Solve}} = \frac{N_{\mathrm{beam}} N_{\mathrm{pol}}}{t_{\mathrm{RCAL,G}}} \times 48 N_{\mathrm{it}} N_{\mathrm{a}}^2 \tag{14}$$

and for the ICAL pipeline:

$$C^{\mathrm{ICAL}}_{\mathrm{Solve}} = (1 + N_{\mathrm{SelfCal}}) N_{\mathrm{beam}} N_{\mathrm{pol}} \left(\frac{1}{t_{\mathrm{ICAL,G}}} + \frac{f^B_{sol}}{t_{\mathrm{ICAL,B}}} \right) \times 48 N_{\mathrm{it}} N_{\mathrm{a}}^2 \tag{15}$$

3.6.6 Subtract

The Subtract step subtracts the predicted visibilities from the observed visibilities. Given available predicted visibilities the Subtract itself will take 2 flops per (single polarisation) visibility.

$$N^{\mathrm{subtract}}_{\mathrm{flop/vis}} = 2 \tag{16}$$

Which gives a total compute rate for the subtract of

$$C_{\mathrm{Subtract}} = 2 \times ((1 + N_{\mathrm{SelfCal}}) N_{\mathrm{major}}) \times R_{\mathrm{vis,full\,FoV}}, \tag{17}$$

since the visibility subtraction must be done at the visibility rate for the full field of view.

3.6.7 Correct

The Correct step corrects observed visibilities with the instrumental Jones matrices. A 4 vector of Observed Visibilities is multiplied by a 4x4 Mueller matrix formed by combining 2x2 Instrumental Jones matrices; and we include the computational cost of generating Mueller matrices, at the cost of $6{*}4{*}4 = 96$ flops each. This leads to a compute cost of

$$C_{\mathrm{correct}} = 224 \times (1 + N_{\mathrm{SelfCal}}) N_{\mathrm{major}} \times R_{\mathrm{vis,full\,FoV}} \tag{18}$$

It should be noted that the number of operations would be considerably reduced using the Measurement Equation formalism (the constant would be 64 instead of 224).

Currently, the correct step is not included in the iPython implementation of the parametric model.

3.6.8 Visibility Weighting

Visibility weights are applied during gridding. For natural weighting, the weights are simply the inverse variances of the visibility. For uniform or Briggs weighting, the weights themselves are calculated using a pre-pass through the visibility u, v coordinates, forming a measure of the density of samples over the grid, and then the density is used to correct weights for over- or under-sampling (so a complex multiply-and-add, with two passes). This gives a cost of 2x8 flops per visibility. Since this occurs just at the gridding step, these weights can be applied to the coalesced visibilities, giving a total cost of:

$$C_{\text{weight}} = (1 + N_{\text{SelfCal}}) N_{\text{major}} N_{\text{facet}}^2 \times 2 \times 8 \times R_{\text{vis,facet FoV}} \tag{19}$$

3.6.9 Phase Rotation

Phase rotation is equivalent to a multiplication of the visibility data by a modulus unity complex factor. In addition, the u, v, w coordinates must also be re-computed. The coordinate computation corresponds to a 3×3 matrix multiply [RD26] (requiring 18 operations). We allow 10 operations for the phase rotation, giving a total of 28 **flops per visibility**:

$$N_{\text{flop/vis}}^{\text{Phase rotation}} = 28 \tag{20}$$

where N_{vis} is the number of visibilities. We need to phase rotate for each facet, and at this point the visibility rate must be appropriate for the full field of view $R_{\text{vis,full FoV}}$ (additional averaging to the faceted field of view limits can be conducted after phase rotation and before gridding).

$$C_{\text{phrot,back}}^{\text{ICAL}} = 28(1 + N_{\text{SelfCal}}) N_{\text{major}} N_{\text{facet}}^2 R_{\text{vis,full FoV}} \tag{21}$$

Additionally, if we do predict at the facet field of view, we will also need to phase rotate the predicted visibilities back, doubling the total number of operations needed for phase rotation.

$$C_{\text{phrot,predict}}^{\text{ICAL}} = 28(1 + N_{\text{SelfCal}}) N_{\text{major}} N_{\text{facet}}^2 R_{\text{vis,full FoV}} \tag{22}$$

$$C_{\text{phrot,total}}^{\text{ICAL}} = 56(1 + N_{\text{SelfCal}}) N_{\text{major}} N_{\text{facet}}^2 R_{\text{vis,full FoV}} \tag{23}$$

We assume that we gather visibility samples that are from the same antenna pairing and that are sufficiently local in time and frequency so as to fall within the same sub-cell in the uv-plane defined by the field of view of the image facet. A discussion of this gathering and the corresponding number of samples to cover the time and frequency axes is give in section C. The required time and frequency resolution is a function of baseline length, so we need to sum over all baselines.[4]

3.6.10 Coalesce Visibilities at the Gridding Step

As mentioned in section 3.3.4, each time an averaging is done the cost is 8 flops per input visibility. Just before gridding, *if a faceted approach is used* for each facet we gather together visibilities to reduce the visibility rate from $R_{\text{vis,full FoV}}$ to $R_{\text{vis,facet FoV}}$. This is done for each imaging cycle ($N_{\text{major}} \times N_{\text{SelfCal}}$). Thus we have a cost for the ICAL pipeline of

$$C_{\text{Coalesce}}^{\text{ICAL}} = 8(1 + N_{\text{SelfCal}}) N_{\text{major}} N_{\text{facet}}^2 R_{\text{vis,full FoV}} \tag{24}$$

[4] In practice, in the iPython implementation of the parametric model we bin the baselines into ten different bins in baseline length, and keep track of the number of baselines within each bin, assigning the maximum baseline length for that bin to each baseline; this speeds up the summation and whilst this slightly overestimates the compute load, the difference is small and certainly much, much less than other uncertainties. We treat all baseline-dependent calculations in the same way.

3.6.11 Grid and Degrid

We assume that 8 floating point operations are required for the update of each target grid point. It is important to acknowledge that the distribution of baselines affects the cost of gridding since the kernel size varies with baseline length (more strictly w). For every sample to be gridded (e.g. for each visibility) the number of flops is thus given by:

$$N_{\mathrm{flop/vis}} = 8(N_{\mathrm{pix,kernel}}(B_{\mathrm{i,j}}))^2 \tag{25}$$

where $N^2_{\mathrm{pix,kernel}}$ is number of pixels in the gridding kernel for convolutional gridding and can be estimated from equation 65 (which is a function of several parameters, including baseline length of the specific antenna pairing). Therefore to grid a single visibility from a long baseline is more costly than for a short baseline, so although this operation scales in principle with the visibility rate, a simple multiplier of e.g. $R_{\mathrm{vis,full\,FoV}}$ cannot be factored out. To get the total gridding operations count we must perform a sum over all baselines, accounting for data averaging that is allowed at each baseline (see section C) and parameterising the size of the convolution kernel with baseline length too (see equations 61 and 65).We have tabulated the histogram of baseline lengths (into ten bins) and we use that tabulation to aid in the calculation.

Gridding using a faceted approach requires N^2_{facet} griddings of the entire data, averaged appropriately for the facet field of view, onto a facet-sized image:

$$\begin{aligned} C^{\mathrm{ICAL}}_{\mathrm{grid}} =& \frac{(1+N_{\mathrm{SelfCal}})N_{\mathrm{major}}N^2_{\mathrm{facet}}N_{\mathrm{pol}}N_{\mathrm{beam}}N_{\mathrm{mm}}}{t_{\mathrm{obs}}} \times \\ & \sum_{\mathrm{i,j}}^{\mathrm{baselines}} 8(N_{\mathrm{pix,kernel}}(B_{\mathrm{i,j}}))^2 N_{\mathrm{f}}^{\mathrm{facet\,FoV}}(B_{\mathrm{i,j}}) N_{\mathrm{time\,samples}}^{\mathrm{facet\,fov}}(B_{\mathrm{i,j}}) \end{aligned} \tag{26}$$

$$\begin{aligned} =& (1+N_{\mathrm{SelfCal}})N_{\mathrm{major}}N^2_{\mathrm{facet}}N_{\mathrm{mm}} \times \\ & \sum_{\mathrm{i,j}}^{\mathrm{baselines}} R_{\mathrm{vis,facet\,FoV}}(B_{\mathrm{i,j}}) 8(N_{\mathrm{pix,kernel}}(B_{\mathrm{i,j}}))^2 \end{aligned} \tag{27}$$

where $R_{\mathrm{vis,facet\,FoV}}(B_{\mathrm{i,j}}) = N_{\mathrm{pol}}N_{\mathrm{beam}}N_{\mathrm{f}}^{\mathrm{facet\,FoV}}(B_{\mathrm{i,j}})N_{\mathrm{time\,samples}}^{\mathrm{facet\,FoV}}(B_{\mathrm{i,j}})/t_{\mathrm{obs}}$ is the facet-field of view visibility rate for the specific baseline (i,j) and $N_{\mathrm{beam}}N_{\mathrm{f}}^{\mathrm{facet\,FoV}}(B_{\mathrm{i,j}})$ is the number of time samples need to grid the facet FOV and is determined using equations 47 and 48.

We have two possible ways to perform the degridding operation. These produce the same intermediate product, that is predicted visibilities at smearing resolution sufficient for the full field of view.

Option A: With No faceting allowed in predict step: visibilities are de-gridded at full time and frequency resolution from a full size image:

$$\begin{aligned} C^{\mathrm{ICAL},A}_{\mathrm{degrid}} =& \frac{(1+N_{\mathrm{SelfCal}})N_{\mathrm{major}}N_{\mathrm{pol}}N_{\mathrm{beam}}N_{\mathrm{mm}}}{t_{\mathrm{obs}}} \times \\ & \sum_{\mathrm{i,j}}^{\mathrm{baselines}} 8(N_{\mathrm{pix,kernel}}(B_{\mathrm{i,j}}))^2 N_{\mathrm{f}}^{\mathrm{full\,FoV}}(B_{\mathrm{i,j}}) N_{\mathrm{time\,samples}}^{\mathrm{full\,fov}}(B_{\mathrm{i,j}}) \end{aligned}$$

$$\begin{aligned} =& (1+N_{\mathrm{SelfCal}})N_{\mathrm{major}}N_{\mathrm{mm}} \times \\ & \sum_{\mathrm{i,j}}^{\mathrm{baselines}} R_{\mathrm{vis,full\,FoV}}(B_{\mathrm{i,j}}) 8(N_{\mathrm{pix,kernel}}(B_{\mathrm{i,j}}))^2 \end{aligned} \tag{28}$$

where $R_{\mathrm{vis,full\,FoV}}(B_{\mathrm{i,j}}) = N_{\mathrm{pol}} N_{\mathrm{beam}} N_{\mathrm{f}}^{\mathrm{full\,FoV}}(B_{\mathrm{i,j}}) N_{\mathrm{time\,samples}}^{\mathrm{full\,FoV}}(B_{\mathrm{i,j}})/t_{\mathrm{obs}}$ is the visibility rate for the specific baseline i,j at full-field of view averaging, and $N_{\mathrm{beam}} N_{\mathrm{f}}^{\mathrm{full\,FoV}}(B_{\mathrm{i,j}})$ is the number of time samples need to grid the full FOV, and is determined using equations 47 and 48.

Option B: Alternatively, predict and sum visibilities per image facet (frequency and time resolution set by facet field of view, exactly as done for the backward step): this gives coarse visibilities (coarser by factor of N_{facet} compared to those in option A); then replicate these to put the visibilities on a finer time-frequency grid; then phase rotate each visibility N_{facet}^2 times (with different phase shift for each facet) to put visibilities on the same fine time- and frequency sampling as in the origin. The cost of this de-gridding option operation is essentially the same as the gridding of averaged data (i.e. as given in equation 26). Option B results in significant savings compared to option A, we have included this choice in the Performance Model. For some more details see RD35.

$$C_{\mathrm{degrid}}^{\mathrm{ICAL},B} = (1 + N_{\mathrm{SelfCal}}) N_{\mathrm{major}} N_{\mathrm{facet}}^2 N_{\mathrm{mm}} \times \sum_{\mathrm{i,j}}^{\mathrm{baselines}} R_{\mathrm{vis,facet\,FoV}}(B_{\mathrm{i,j}}) 8 (N_{\mathrm{pix,kernel}}(B_{\mathrm{i,j}}))^2 \tag{29}$$

3.6.12 Gridding Kernel Update

The gridding kernel potentially changes with a large number of variables: antenna1, antenna2, w, time, frequency, polarisation, direction on the sky. This recalculation can be a large computational burden. We see three differing approaches: calculate on demand, calculate on demand with caching, and pre-calculate. Pure on-demand is obviously wasteful, while pre-calculation is wasteful if we fill out more values than are needed. On-demand with caching and pre-calculation should be equivalent if the pre-calculation is done optimally. Therefore in what follows, we will analyse the case for on-demand with caching if necessary.

In this approach, the convolution kernel is kept in lookup tables indexed as necessary. The minimum indices are channel (or frequency), w. Typically the significant values of the kernel occupy only a limited region in this space and so the memory use can be optimised. There are additional indices that may be required. For the aperture array stations, the primary station beams will vary significantly in azimuth and elevation space. For the dish antennas, just elevation may be sufficient. In addition the stations may differ from each other because of the need to decohere on bright sources in the sidelobes. In this case, the number of gridding kernels will scale as the number of baselines.

The computation of the baseline-dependent convolution kernels is dominated by the FFT from the image plane. The number of flops required to re-calculate the (possibly baseline-dependent) convolution kernels are therefore set by the linear size in pixels of the far-field from which the (complex-to-complex) FFT is performed, N_{cvff}

$$N_{\mathrm{flop/kernel}} = 5 N_{\mathrm{cvff}}^2 \log_2(N_{\mathrm{cvff}}^2) \tag{30}$$

The convolution kernel itself has two major contributions: (1) the w-kernel and (2) the A-kernel. The support of the convolution kernel is given by

$$\begin{aligned} N_{\mathrm{cvff}} &= Q_{\mathrm{GCF}} \sqrt{N_{\mathrm{GW}}^2 + N_{\mathrm{AA}}^2} \\ &= Q_{\mathrm{GCF}} N_{\mathrm{pix,kernel}}(B_{\mathrm{i,j}}) \end{aligned} \tag{31}$$

where N_{GW} is the support of the w-kernel and N_{AA} is the support of the A-kernel. Q_{GCF} is an over-sampling factor, typically set to be eight.

In the gridding or degridding step, we may choose to model each antenna/station voltage pattern identically or differently. This introduces a factor $N_{\mathrm{A,kern}}$ the number of distinct A kernels that must be

calculated. This is which is unity for identical antennas/stations and for non-identical antennas could be as high as the number of baselines:

$$N_{\mathrm{A,kern}} = \frac{N_{\mathrm{a}}(N_{\mathrm{a}}+1)}{2} \tag{32}$$

The factor $N_{\mathrm{A,kern}}$ is chosen according to the dynamic range required. Since SKA1-Mid is made up of two distinct antenna types (the 13.5 m MeerKAT antennas and the 15 m SKA antennas) there will be at least $N_{\mathrm{A,kern}} = 3$ different A-kernels to handle (corresponding to the 13.5 m:13.5 m, 13.5 m:15 m and 15 m:15 m dish baseline combinations).

It is not yet clear to what extent convolution kernels will need to be pre-calculated and cached or to what extent they can be calculated on the fly, or at least locally on a compute island as needed (and re-used for multiple frequency and time samples for a specific baseline). In section E we consider to what extent kernels can be reused over frequency and time (the frequency re-use being set by considerations akin to frequency smearing, and time re-use limited by an update timescale ($t_{\mathrm{kernel,update}}$) which might be set by phase variations if these are corrected via the kernel. We estimate a total cost for calculation of these kernels assuming that these kernels must be regenerated for every imaging cycle. [5]

The total number of convolution kernels we need to calculate to handle the full observations data is a sum over all baselines of the number of kernels needed to cover the frequency and time axes for each baseline. The total cost to calculate all required kernels is therefore:

$$C_{\mathrm{Kernels}} = \frac{(1+N_{\mathrm{SelfCal}})N_{\mathrm{major}}N_{\mathrm{pol}}N_{\mathrm{mm}}N_{\mathrm{beam}}N_{\mathrm{facet}}^2}{t_{\mathrm{kernel,update}}} \left(\sum_{\mathrm{i,j}}^{\mathrm{baselines}} 5N_{\mathrm{cvff}}^2 \log_2(N_{\mathrm{cvff}}^2) N_{\mathrm{f,kernel}} \right) \tag{33}$$

We assume that a kernel caching strategy can be devised to reach close to this lower bound for each major cycle. Caching over major cycles may also be possible but we have not yet assumed that.

The frequency re-use of kernels (giving the baseline-dependent $N_{\mathrm{f,kernel}}$ and timescale for kernel update ($t_{\mathrm{kernel,update}}$)) is described in more detail in section E.

3.6.13 FFT and IFFT

We assume that a real-to-complex 2D FFT case be used in conjunction with gridding and degridding. For a real-to-complex or complex-to-real Fast-Fourier Transform working on a grid N_{pix} by N_{pix} the following number of flops are required for a grid of size N_{pix} by N_{pix}.(See the FFT Support Document [RD09] for more information.)

$$N_{\mathrm{flop}} = 2.5N_{\mathrm{pix}}^2 \log_2(N_{\mathrm{pix}}^2) \tag{34}$$

(note that for a complex-to-complex FFT the prefactor of 2.5 must be doubled).

The total FLOPS rate for the FFT stage is given by:

$$C_{\mathrm{FFT}}^{\mathrm{ICAL}} = \frac{(1+N_{\mathrm{SelfCal}})N_{\mathrm{major}}N_{\mathrm{facet}}^2N_{\mathrm{pol}}N_{\mathrm{beam}}(N_{\mathrm{subbands}}N_{\mathrm{Tt}})}{t_{\mathrm{snap}}} 2.5N_{\mathrm{pix,facet}}^2 \log_2(N_{\mathrm{pix,facet}}^2) \tag{35}$$

In this we have assumed that grids are combined to form Taylor term grids (for each imaging sub-band) and then the FFT is performed.

The IFFT is identical in cost:

$$C_{\mathrm{IFFT}}^{\mathrm{ICAL}} = \frac{(1+N_{\mathrm{SelfCal}})N_{\mathrm{major}}N_{\mathrm{facet}}^2N_{\mathrm{pol}}N_{\mathrm{beam}}(N_{\mathrm{subbands}}N_{\mathrm{Tt}})}{t_{\mathrm{snap}}} 2.5N_{\mathrm{pix,facet}}^2 \log_2(N_{\mathrm{pix,facet}}^2) \tag{36}$$

[5] This is an example only, and not a statement as to which implementation (pre-calculated or on-the-fly kernels or some middle ground) is best for SKA - the choice needs to be considered alongside the calibration strategy.

The factor $N_{\mathrm{subbands}}N_{\mathrm{Tt}}$ is used to cover the frequency axis for the ICAL pipeline since we have N_{Tt} Taylor terms in the frequency expansion for each of the N_{subbands} subbands. For the DPrepA pipeline, this is also the case, but the DPrepB and C pipelines need images made at a specified number ($N_{\mathrm{f,out}}$) of separate spectral channels, though the IFFT (to predict from the Taylor term sky model) can still be done in Taylor-terms (see the summary tables for each pipeline, Tables 12, 13).

3.6.14 Reprojection

The number of floating point operations for the re-projection is directly proportional to the total number of pixels in the grid (N^2_{pix}, i.e. the square of the linear dimension of the grid). Assuming bi-cubic interpolation and a pre-factor of 50 the number of operations per grid is:

$$N^{\mathrm{Reprojection}}_{\mathrm{flop}} = 50N^2_{\mathrm{pix}} \tag{37}$$

This is assuming interpolation from a 3×3 grid of pixels and about 5 flops per pixel to do the interpolation and addition of the images. Pixel weights should be re-usable for each of the polarisation products to allow efficient computation.

We need to multiply this by the total number of grids required for the observation to get a total number of operations.

$$C^{\mathrm{ICAL}}_{\mathrm{Reproj}} = \frac{(1+N_{\mathrm{SelfCal}})N_{\mathrm{major}}N^2_{\mathrm{facet}}N_{\mathrm{pol}}N_{\mathrm{beam}}(N_{\mathrm{subbands}}N_{\mathrm{Tt}})}{t_{\mathrm{snap}}}50N^2_{\mathrm{pix,facet}} \tag{38}$$

There is actually no dependence on N_{facet} because (apart from when no faceting is used) the $N^2_{\mathrm{facet}}N^2_{\mathrm{pix,facet}} = r^2_{\mathrm{facet}}N^2_{\mathrm{pix}}$ where r_{facet} is a factor (likely around 1.2-1.5) which includes the need for adjacent facets to overlap one another. Thus we can re-write the equation as:

$$C^{\mathrm{ICAL}}_{\mathrm{Reproj}} = \frac{(1+N_{\mathrm{SelfCal}})N_{\mathrm{major}}r^2_{\mathrm{facet}}N_{\mathrm{pol}}N_{\mathrm{beam}}(N_{\mathrm{subbands}}N_{\mathrm{Tt}})}{t_{\mathrm{snap}}}50N^2_{\mathrm{pix}} \tag{39}$$

where we use the factor $N_{\mathrm{subbands}}N_{\mathrm{Tt}}$ to cover the frequency axis for the ICAL and DPrepA pipelines as for the FFT, but again, we need $N_{\mathrm{f,out}}$ for DPrep B,C. The reprojection timescale is the same as the FFT timescale - it must be t_{snap} as this is the rate at which the images are made.

3.6.15 Image Spectral Fitting

The Multi-Scale Multi-Frequency Synthesis algorithm [RD38] has two implementations in common use: one in CASA and one in ASKAPsoft. These differ in the calculation of the Taylor terms images. CASA calculates a large number of dirty images and beam and then forms fractional frequency offset weighted moment images directly. ASKAPsoft calculates the moment images by gridding appropriately weighted visibilities, once for each Taylor term. These two calculations should be identical if done correctly and so the only advantages of one over the other lies in computational issues such as flops, memory usage, and interconnect bandwidth.

$$C^{\mathrm{ICAL}}_{\mathrm{SpecFit}} = 2\frac{(1+N_{\mathrm{SelfCal}})N_{\mathrm{major}}N_{\mathrm{pol}}N_{\mathrm{beam}}(N_{\mathrm{subbands}}N_{\mathrm{Tt}})N^2_{\mathrm{pix}}}{t_{\mathrm{obs}}} \tag{40}$$

The timescale for spectral fitting is t_{obs} since it is done on images combined from all snapshots.

3.6.16 Subtract Image Component

The minor cycle processing cost is composed of two parts, the identification of point sources and the subtraction of these identified sources in the image plane.

The compute cost for the point source identification is:

$$C^{\mathrm{ICAL}}_{\mathrm{minor\,clean}} = \frac{(N_{\mathrm{SelfCal}}+1)N_{\mathrm{major}}N_{\mathrm{minor}}N_{\mathrm{pol}}N_{\mathrm{beam}}(N_{\mathrm{subbands}}N_{\mathrm{Tt}})}{t_{\mathrm{obs}}}2N^2_{\mathrm{pix}} \quad (41)$$

The number of sources that need to be subtracted in the minor cycle depends on many characteristics of the observations being made, the telescope parameters and the characteristics of the imaging algorithm. Normally, some fraction of the synthesised point spread function is used as the beam patch that is used for subtraction. The number of pixels in the patch is denoted $N_{\mathrm{patch,pix}}$

To a first order, the compute cost for the subtraction is then:

$$C^{\mathrm{ICAL}}_{\mathrm{minor\,clean}} = \frac{(1+N_{\mathrm{SelfCal}})N_{\mathrm{major}})N^2_{\mathrm{facet}}N_{\mathrm{minor}}N_{\mathrm{pol}}N_{\mathrm{beam}}(N_{\mathrm{subbands}}N_{\mathrm{Tt}})}{t_{\mathrm{obs}}}2N^2_{\mathrm{patch,pix}} \quad (42)$$

where we have inserted a factor of N^2_{facet}, though essentially the total cost should be independent of whether the minor cycle is done with faceting in, since the number of minor cycle clean per facet would need to be adjusted accordingly, so the product $N^2_{\mathrm{facet}} \times N_{\mathrm{minor}}$ should be a constant if equivalent sources are cleaned.

Again the factor $(N_{\mathrm{subbands}}N_{\mathrm{Tt}})$ is appropriate for the ICAL and DPrepA pipelines but must be replaced by $N_{\mathrm{f,out}}$ for DPrepB and DPrepC.

3.6.17 Source find

All source finding techniques are assumed to operate on images or image cubes being produced by the various Imaging Pipelines.

The point source detection algorithms required for the sky model function by comparing each image pixel with some threshold noise level and then growing any regions thought to contain emission (e.g. RD30, RD31, RD32). The overall cost of each growing operation scales with the size of the emission (i.e. the size of the synthesised beam) and the number of such growing operations scales with the number of detected sources. If one assumes that the number of sources detected is (at worst) proportional to the input image size then the overall computational cost of point source detection will scale with the image size.

However, going much further is difficult since the source finding algorithms currently in use [RD07] are heavily driven by heuristics that are difficult to summarise in flops. We can make some progress though if we restrict consideration to sources that will end up as discrete models in the GSM i.e. the sources that are used for calibration. This is context dependent. For the moment, we work with the GSM source density, ρ_{GSM}.

The approximate number of sources that can be found is about:

$$\rho_{\mathrm{GSM}}\theta^2_{\mathrm{FoV}} \quad (43)$$

Each source must be found and then fitted in Stokes I at the reference frequency. The finding operation goes as the number of pixels included in the fit - about 100 typically. For the source fitting, at the bare minimum, we can expect a non-linear least squares algorithm with 6 parameters for each source. According to this simple model for the fitting process, the number of iterations is $N_{\mathrm{sourcefit}}$ O(10).

$$1200N_{\mathrm{sourcefit}}\rho_{\mathrm{GSM}}\theta^2_{\mathrm{FoV}} \quad (44)$$

This corresponds to about 1.2 kFlops per source found. We can make some estimates for the number of sources likely to be in the 6-hour duration maps produced routinely by the SDP: extrapolation of modelled source counts following Condon (2012) [RD36], shows that an SKA1-Low map made with 30% fractional bandwidth might have around 40,000 sources per square degree in it, so there might be $O(10^6)$ sources in a map. However, even allowing for 1 million sources, this corresponds to O(MFLOPS) when divided by the 6 hours observing time. Hence, even if we have underestimated this be a factor

of one million, source finding should still not be a significant compute burden compared with other pipeline components. Since it is both likely to be a small cost and the numerical values are uncertain, source finding is not included in our iPython model.

4 Summary of the Performance Model

The Performance Model (PM) is now capable of calculating the full computational load of all defined SDP pipelines. The verification of the Model will be ongoing in parallel with the benchmarking and prototyping. It must be remembered that the model mostly produces lower bounds because of the as yet unknown impact of other factors such as parallelism model, memory model, I/O capabilities, language. However, it is now true that the PM can be used to evaluate the performance of all the major wide-field imaging algorithms, such as facets, w-projection, a-projection, aw-projection, and aw-snapshots. This means that the PM can be used for evaluation of future such algorithmic advances.

The Tables 10 to 15 below summarise the results from each of the calibration and imaging pipelines. See Tables 8 and 9 for the Receive and Pre-Processing costs.

Pipeline: RCAL, Real time calibration
Extract LSM ...(not yet modelled)
Predict via DFT $C_{\mathrm{Predict}} = 16 N_{\mathrm{source}} \times R_{\mathrm{vis,full\,FoV}}$
Solve $C_{\mathrm{Solve}} = \frac{N_{\mathrm{beam}}}{t_{\mathrm{RCAL,G}}} \times 48 N_{\mathrm{it}} N_{\mathrm{a}}^2 N_{\mathrm{pol}}$ The time $t_{\mathrm{RCAL,G}}$ is the G solution interval.
Subtract $C_{\mathrm{Subtract}} = 2 R_{\mathrm{vis,full\,FoV}}$
Correct $C_{\mathrm{correct}} = 224 R_{\mathrm{vis,full\,FoV}}$
Phase rotation $C_{\mathrm{phrot,total}} = 28 R_{\mathrm{vis,full\,FoV}}$
$C_{\mathrm{de-grid}} = \left(\frac{N_{\mathrm{facet}}^2 N_{\mathrm{pol}} N_{\mathrm{beam}} N_{\mathrm{mm}}}{t_{\mathrm{obs}}} \right) \sum_{\mathrm{i,j}}^{\mathrm{baselines}} 8 (N_{\mathrm{pix,kernel}}(B_{\mathrm{i,j}}))^2 N_{\mathrm{f}}^{\mathrm{facet\,FoV}}(B) N_{\mathrm{time\,samples}}^{\mathrm{facet\,FoV}}(B)$
Gridding kernel update $C_{\mathrm{Kernels}} = \frac{N_{\mathrm{facet}}^2 N_{\mathrm{pol}} N_{\mathrm{beam}}}{t_{\mathrm{kernel,update}}} (\sum_{\mathrm{i,j}}^{\mathrm{baselines}} 5 N_{\mathrm{cvff}}^2 \log_2(N_{\mathrm{cvff}}^2) N_{\mathrm{f,kernel}})$
IFFT $C_{\mathrm{IFFT}} = \left(\frac{N_{\mathrm{facet}}^2 N_{\mathrm{pol}} N_{\mathrm{beam}} (N_{\mathrm{subbands}} N_{\mathrm{Tt}})}{t_{\mathrm{snap}}} \right) 2.5 N_{\mathrm{pix,facet}}^2 \log_2(N_{\mathrm{pix,facet}}^2)$
Re-project $C_{\mathrm{Reproj}} = \frac{r_{\mathrm{facet}}^2 N_{\mathrm{pol}} N_{\mathrm{beam}} (N_{\mathrm{subbands}} N_{\mathrm{Tt}})}{t_{\mathrm{snap}}} 50 N_{\mathrm{pix}}^2$

Table 10: Components and scaling for real time calibration pipeline RCAL.

Pipeline: ICAL, Self-calibration
Extract LSM; Update LSM; Notify LSM ...(not yet modelled)
Predict via DFT $C_{\text{Predict}} = (1 + N_{\text{SelfCal}})N_{\text{major}}16N_{\text{source}} \times R_{\text{vis,full FoV}}$
Solve $C_{\text{Solve}} = (1 + N_{\text{SelfCal}})N_{\text{beam}}N_{\text{pol}}\left(\frac{1}{t_{\text{ICAL,G}}} + \frac{f^B_{sol}}{t_{\text{ICAL,B}}}\right) \times 48N_{\text{it}}N^2_{\text{a}}$
Subtract $C_{\text{Subtract}} = 16 \times ((1 + N_{\text{SelfCal}})N_{\text{major}}) \times R_{\text{vis,full FoV}}$
Correct $C_{correct} = 224 \times (1 + N_{\text{SelfCal}})N_{\text{major}} \times R_{\text{vis,full FoV}}$
Weight visibilities $C_{\text{Weight}} = 16(1 + N_{\text{SelfCal}})N_{\text{major}}R_{\text{vis,full FoV}}$
Phase rotation $C_{\text{phrot,total}} = 2 \times (1 + N_{\text{SelfCal}})N_{\text{major}}N^2_{\text{facet}} \times 28R_{\text{vis,full FoV}}$
Coalesce $C_{\text{Coalesce}} = 8(1 + N_{\text{SelfCal}})N_{\text{major}}N^2_{\text{facet}}R_{\text{vis,full FoV}}$
Grid $C_{\text{grid}} = (1 + N_{\text{SelfCal}})N_{\text{major}}N^2_{\text{facet}}N_{\text{mm}} \times \sum^{\text{baselines}}_{\text{i,j}} R_{\text{vis,facet FoV}}(B_{\text{i,j}})8(N_{\text{pix,kernel}}(B_{\text{i,j}}))^2$
De-grid $C_{\text{de}-\text{grid}} = C_{grid}$
Gridding kernel update $C_{\text{Kernels}} = \frac{(1+N_{\text{SelfCal}})N_{\text{major}}N^2_{\text{facet}}N_{\text{pol}}N_{\text{beam}}}{t_{\text{kernel,update}}}(\sum^{\text{baselines}}_{\text{i,j}} 5N^2_{\text{cvff}} \log_2(N^2_{\text{cvff}})N_{\text{f,kernel}})$
FFT $C_{\text{FFT}} = \left(\frac{(1+N_{\text{SelfCal}})N_{\text{major}}N^2_{\text{facet}}N_{\text{pol}}N_{\text{beam}}(N_{\text{subbands}}N_{\text{Tt}})}{t_{\text{snap}}}\right) 2.5N^2_{\text{pix,facet}} \log_2(N^2_{\text{pix,facet}})$
IFFT $C_{\text{IFFT}} = C_{\text{FFT}}$
Re-project $C_{\text{Reproj}} = \frac{(1+N_{\text{SelfCal}})N_{\text{major}}r^2_{\text{facet}}N_{\text{pol}}N_{\text{beam}}(N_{\text{subbands}}N_{\text{Tt}})}{t_{\text{snap}}}50N^2_{\text{pix}}$
Image spectral fitting $C_{\text{spectralfit}} = 2\frac{(1+N_{\text{SelfCal}})N_{\text{major}}N_{\text{pol}}N_{\text{beam}}(N_{\text{subbands}}N_{\text{Tt}})N^2_{\text{pix}}}{t_{\text{obs}}}$
Identify Component $C_{\text{Identifycomponent}} = \frac{2(1+N_{\text{SelfCal}})N_{\text{major}}N_{\text{minor}}N_{\text{beam}}N_{\text{pol}}N_{\text{subbands}}N_{\text{Tt}}N^2_{\text{pix}}}{t_{\text{obs}}}$
Subtract Image component $C_{\text{Subtractimagecomponent}} = \frac{2(1+N_{\text{SelfCal}})N_{\text{major}}N_{\text{minor}}N_{\text{pol}}N_{\text{beam}}(N_{\text{subbands}}N_{\text{Tt}}N^2_{\text{patch,pix}})}{t_{\text{obs}}}$
Source Find Insignificant

Table 11: Components and scaling for ICAL

Pipeline: DPrepA - Make Taylor Term Continuum Images in Stokes I only.
Extract LSM; Update LSM; Notify LSM ...(not yet modelled)
Subtract $C_{\text{Subtract}} = 16 N_{\text{major}} R_{\text{vis,full FoV}}$
Weight visibilities $C_{\text{Weight}} = 16 N_{\text{major}} N_{\text{facet}}^2 R_{\text{vis,facet FoV}}$
Phase rotation $C_{\text{phrot,total}} = 2 N_{\text{major}} N_{\text{facet}}^2 28 R_{\text{vis,full FoV}}$
Coalesce $C_{\text{Coalesce}} = 8 N_{\text{major}} N_{\text{facet}}^2 R_{\text{vis,full FoV}}$
Grid $C_{\text{grid}} = 8 N_{\text{major}} N_{\text{facet}}^2 N_{\text{mm}} \sum_{\text{i,j}}^{\text{baselines}} R_{\text{vis,facet FoV}}(B_{\text{i,j}}) 8 (N_{\text{pix,kernel}}(B_{\text{i,j}}))^2$
De-grid $C_{\text{de-grid}} = C_{grid}$
Gridding kernel update $C_{\text{Kernels}} = \frac{N_{\text{major}} N_{\text{facet}}^2 N_{\text{beam}}}{t_{\text{kernel,update}}} \sum_{\text{i,j}}^{\text{baselines}} 5 N_{\text{cvff}}^2 \log_2(N_{\text{cvff}}^2) N_{\text{f,kernel}}$
FFT $C_{\text{FFT}} = \left(\frac{N_{\text{major}} N_{\text{facet}}^2 N_{\text{beam}} (N_{\text{subbands}} N_{\text{Tt}})}{t_{\text{snap}}} \right) 2.5 N_{\text{pix,facet}}^2 \log_2(N_{\text{pix,facet}}^2)$
IFFT $C_{\text{IFFT}} = C_{\text{FFT}}$
Re-project $C_{\text{Reproj}} = 50 \frac{N_{\text{major}} r_{\text{facet}}^2 N_{\text{beam}} (N_{\text{subbands}} N_{\text{Tt}}) N_{\text{pix}}^2}{t_{\text{snap}}}$
Image spectral fitting $C_{\text{spectralfit}} = 2 \frac{N_{\text{major}} N_{\text{beam}} (N_{\text{subbands}} N_{\text{Tt}}) N_{\text{pix}}^2}{t_{\text{obs}}}$
Identify Component $C_{\text{identifycomp}} = \frac{2 N_{\text{major}} N_{\text{minor}} N_{\text{beam}} (N_{\text{subbands}} N_{\text{Tt}}) N_{\text{pix}}^2}{t_{\text{obs}}}$
Subtract Image component $C_{\text{Subtractcomponent}} = 2 \frac{N_{\text{major}} N_{\text{minor}} N_{\text{beam}} (N_{\text{subbands}} N_{\text{Tt}}) N_{\text{patch,pix}}^2}{t_{\text{obs}}}$
Source Find Insignificant
Note that this pipeline works in total intensity only. $N_{\text{pol}} = 1$ in the definition of $R_{\text{vis,full FoV}}$ and $R_{\text{vis,facet FoV}}$ (equations 51 and 52).

Table 12: Components and scaling for DPrepA.

Pipelines: DPrepB & DPrepC - Make frequency image cubes
Extract LSM; Update LSM; Notify LSM …(not yet modelled)
Subtract $C_{\text{Subtract}} = 16 \times N_{\text{major}} \times R_{\text{vis,full FoV}}$
Weight visibilities $C_{\text{Weight}} = N_{\text{major}} N_{\text{facet}}^2 16 R_{\text{vis,facet FoV}}$
Phase rotation $C_{\text{phrot,total}} = 2 \times N_{\text{major}} N_{\text{facet}}^2 \times 28 R_{\text{vis,full FoV}}$
Coalesce $C_{\text{Coalesce}} = 8 N_{\text{major}} N_{\text{facet}}^2 R_{\text{vis,full FoV}}$
Grid $C_{\text{grid}} = N_{\text{major}} N_{\text{facet}}^2 N_{\text{mm}} \times \sum_{\text{i,j}}^{\text{baselines}} R_{\text{vis,facet FoV}}(B_{\text{i,j}}) 8 (N_{\text{pix,kernel}}(B_{\text{i,j}}))^2$ Note: Frequency averaging may be restricted by experiment for spectral line experiments.
De-grid $C_{de-grid} = C_{grid}$
Gridding kernel update $C_{\text{Kernels}} = \frac{N_{\text{major}} N_{\text{facet}}^2 N_{\text{pol}} N_{\text{beam}}}{t_{\text{kernel,update}}} (\sum_{B_{\text{i,j}}} 5 N_{\text{cvff}}^2 \log_2(N_{\text{cvff}}^2) N_{\text{f,kernel}})$
FFT $C_{\text{FFT}} = \left(\frac{N_{\text{major}} N_{\text{facet}}^2 N_{\text{pol}} N_{\text{beam}} N_{\text{f,out}}}{t_{\text{snap}}} \right) 2.5 N_{\text{pix,facet}}^2 \log_2(N_{\text{pix,facet}}^2)$
IFFT $C_{\text{IFFT}} = C_{\text{FFT}}$
Re-project $C_{\text{Reproj}} = \frac{N_{\text{major}} r_{\text{facet}}^2 N_{\text{pol}} N_{\text{beam}} N_{\text{f,out}}}{t_{\text{snap}}} 50 N_{\text{pix}}^2$
Image spectral fitting $2 \frac{N_{\text{major}} N_{\text{pol}} N_{\text{beam}} (N_{\text{subbands}} N_{\text{Tt}}) N_{\text{pix}}^2}{t_{\text{obs}}}$
Identify Component $C_{\text{Identifycomponent}} = \frac{2 N_{\text{major}} N_{\text{minor}} N_{\text{beam}} N_{\text{pol}} N_{\text{subbands}} N_{\text{Tt}} N_{\text{pix}}^2}{t_{\text{obs}}}$
Subtract Image component $C_{\text{Subtractimagecomponent}} = \frac{N_{\text{major}} N_{\text{minor}} N_{\text{pol}} N_{\text{beam}} (N_{\text{f,out}})}{t_{\text{obs}}} 2 N_{\text{patch,pix}}^2$
Source Find Insignificant
Note that B and C are identical apart from the number of output channels: for C we anticipate very fine channelisation so that we need to grid and FFT all at a frequency resolution set by the output cube.

Table 13: Components and scaling for DPrepB & DPrepC.

Pipeline: FastImg
Extract LSM; Notify LSM ...(not yet modelled)
Subtract $C_{\text{Subtract}} = 16 N_{\text{major}} R_{\text{vis,full FoV}}$
Weight visibilities $C_{\text{Weight}} = 16 N_{\text{major}} R_{\text{vis,facet FoV}}$
Phase rotation $C_{\text{phrot,total}} = 2 \times N_{\text{major}} \times 28 R_{\text{vis,full FoV}}$
Coalesce $C_{\text{Coalesce}} = 8 N_{\text{major}} N_{\text{facet}}^2 R_{\text{vis,full FoV}}$
Grid $C_{\text{grid}} = N_{\text{major}} N_{\text{facet}}^2 N_{\text{mm}} \times \sum_{\text{i,j}}^{\text{baselines}} R_{\text{vis,facet FoV}}(B_{\text{i,j}}) 8 (N_{\text{pix,kernel}}(B_{\text{i,j}}))^2$
De-grid $C_{\text{de-grid}} = C_{grid}$
Gridding kernel update $C_{\text{Kernels}} = \frac{(N_{\text{SelfCal}} N_{\text{major}}) N_{\text{facet}}^2 N_{\text{pol}} N_{\text{beam}}}{t_{\text{kernel,update}}} (\sum_{B_{\text{i,j}}} 5 N_{\text{cvff}}^2 \log_2(N_{\text{cvff}}^2) N_{\text{f,kernel}})$ We assume we can re-use kernels for timescales longer than the snapshot timescale
FFT $C_{\text{FFT}} = \left(\frac{N_{\text{facet}}^2 N_{\text{pol}} N_{\text{beam}} (N_{\text{subbands}} N_{\text{Tt}})}{t_{\text{snap}}} \right) 2.5 N_{\text{pix,facet}}^2 \log_2(N_{\text{pix,facet}}^2)$ t_{snap} here is the fast imaging timescale.
IFFT $C_{\text{IFFT}} = \left(\frac{N_{\text{facet}}^2 N_{\text{pol}} N_{\text{beam}} (N_{\text{subbands}} N_{\text{Tt}})}{t_{\text{snap}}} \right) 2.5 N_{\text{pix,facet}}^2 \log_2(N_{\text{pix,facet}}^2)$
Source Find Insignificant

Table 14: Components and scaling for FastImg

Pipeline: DPrep D - Correct and Average visibility data (no imaging operations).
Average $C_{\text{correct}} = 8 \times R_{\text{vis,corr}}$
Correct $C_{\text{correct}} = 8 N_{\text{mm}} \times R_{\text{vis,corr}}$

Table 15: Components and scaling for DPrepD

A Time Averaging – Covering the Time Axis

The parametric model has several different ways of partitioning data and solutions in time; we outline these here.

A.1 Correlator Time Resolution, t_{dump}

The SDP receives visibility data from the CSP with time resolution given by the correlator "dump" time (t_{dump}), or (inversely) at the dump *rate* $R_{\mathrm{vis,corr}} = t_{\mathrm{dump}}^{-1}$. This is the finest temporal resolution that the SDP has access to.

A.2 Time Smearing Limit, t_{smear}

The (orientation-independent) time smearing limit is given by RD10:

$$t_{\mathrm{smear}}(B) = \frac{\epsilon_{\mathrm{f}} \lambda}{\Omega_{\mathrm{E}} \theta_{\mathrm{FoV}} B} \tag{45}$$

where ϵ_{f} corresponds to the fraction of a uv-cell (of size $\theta_{\mathrm{FoV}}^{-1}$) that we gather time samples across (this is linked to the amplitude loss at the edge of the field of view following equation 1 in RD10), and B is the baseline length. Because this is linked to the *uv* cell size it is affected by faceting and may be different for the backward and forward steps.

We use this smearing time limit to enable averaging (or coalescing) of visibility data at the gridding step, provided that other criteria are upheld. See section C.

A.3 Snapshot Time, t_{snap}

We assume a snapshotting approach, where the full observation (which may be several hours) is broken up into shorter "snapshots" which are gridded separately and then re-projected. Because the *uvw* points within an individual snapshot are closer to a 2D plane than they would be for the full observation the convolution kernels needed to deal with array non-coplanarity are smaller and hence gridding costs are reduced. We select a snapshot time which minimises total compute load. See section H.1.

The *uv* plane grids need to cover each snapshot time, at whatever frequency resolution is appropriate, so the number of snapshots is a multiplicative factor for FFT operations.

A.4 Observation Length, t_{obs}

This is the total length of an individual observation and the longest timescale considered by the SDP, sometimes referred to as a synthesis imaging run. Here we assume 6 hours is the typical value, though this is parametrised as t_{obs}.

B Covering the Frequency Axis

The parametric model has several different ways of covering frequency space. This can be quite complicated and confusing, so we introduce them here. We express the frequency resolution in terms of the number of frequency samples needed to cover the full band, N_{f}, for a number of different scenarios.

We give some example numbers here because their orders of magnitude can be instructive.

B.1 Number of Channels at Correlator Frequency Resolution, $N_{\mathrm{f,corr}}$

This is the frequency resolution coming out of the correlator, and is the finest resolution that the SDP has access to. The number of frequency "channels" is denoted $N_{\mathrm{f,corr}}$. Currently the maximum value of $N_{\mathrm{f,corr}}$ is 65,536 (2^{16}) for each of Low and Mid.

B.2 Number of Channels at the Frequency Smearing Limit, $N_{\mathrm{f,smear}}$

In the case of continuum imaging and fast imaging the visibilities can be binned in frequency to an extent determined by the bandwidth and the required output frequency resolution. To estimate this, we first define the number of channels $N_{\mathrm{f}} = \Delta f / \mathrm{d}f$, where Δf is the sub-band frequency range, and is given by [X] and $\mathrm{d}f$ is the maximum width of a single channel to avoid bandwidth smearing and can be found by taking into account the difference in the baseline length between the top (f_{top}) and bottom ($f_{top} + \mathrm{d}f$) of the channel: $B_{uv} = B \cdot (f_{top}/c)$; $\mathrm{d}B_{uv} = B \cdot (\mathrm{d}f/c)$.

To avoid bandwidth smearing, $\mathrm{d}B_{uv}$ cannot be larger than the fraction of a *uv* cell, ϵ_{f}, which might be approximately one-quarter to one-third. Thus, at the smearing limit, $\mathrm{d}f = \mathrm{d}B_{uv} \cdot c/B$ = $(\epsilon_{\mathrm{f}}/\theta_{\mathrm{FoV}}) \cdot (c/B)$.

Since the frequency smearing condition is a fractional bandwidth for the visibility data from a given baseline and a a given time we can bin in frequency in linearly spaced bins, provided that the scientific use case allows for it. At each baseline length the number of frequency channels thus required:

$$N_{\mathrm{f,smear}}(B) = \frac{\Delta f}{\mathrm{d}f} = \frac{\Delta f \theta_{\mathrm{FoV}} B}{\epsilon_{\mathrm{f}} c} \tag{46}$$

Such binning of visibilities provides a substantial reduction in data rate and, hence, a computational saving for ultra-wide bandwidth telescopes like the SKA.

To give some approximate examples, with $\epsilon_f = 0.33$ and with the image field of view out to the first null of the primary beam, to meet these frequency smearing constraints one needs around [X] frequency channels to cover a 2:1 bandwidth ratio with 150-km baselines and 15-m dishes; the numbers are almost linear with baseline: 50-km baselines need [X] channels, 20-km baselines need only [X] channels.

C Visibility Averaging and Coalescing

As has been outlined in sections **??** and A.2, the field of view to be imaged gives rise to limitations in the amount of averaging in frequency or time that is possible before the images will be corrupted. These limits are functions of baseline length, with more averaging allowed on shorter baselines.

In order to reduce the cost of the computational steps in the imaging pipelines, we assume that adjacent visibility points from the same antenna pairing (and which need the same convolution kernel) can be averaged together before any operations are performed. This averaging in both time and frequency is set by the smearing limits appropriate to the full field of view and baseline length, but of course, cannot be at finer resolution than coming out of the correlator, and nor can it be coarser than the timescale on which the convolution kernel is updated, $t_{\mathrm{kernel,update}}$, nor coarser in frequency than the required resolution of the final data products.

The time smearing limit is given in equation 45. We use this to estimate the number of time samples that can be combined, which is given by

$$N_{\mathrm{t,coalesce}}(B) = \mathrm{Max}\left(\mathrm{Min}\left(\mathrm{floor}\left(\frac{t_{\mathrm{smear}}(B)}{t_{\mathrm{dump}}}\right), \mathrm{floor}\left(\frac{t_{\mathrm{kernel,update}}}{t_{\mathrm{dump}}}\right)\right), 1\right) \tag{47}$$

To clarify the logic in equation 47: $\mathrm{floor}\frac{t_{\mathrm{smear}}(B)}{t_{\mathrm{dump}}}$ rounds down to the nearest integer the number of samples that can be combined before reaching the time smearing limit, and likewise the second floor function does the same w.r.t the kernel update rate. Then an overall lower limit of 1 is set since this is the minimum (if the smearing limit was actually less than a correlator dump time we could be in this scenario, which is possible if we attempt to image and clean sources from a field of view larger than the second null of the primary beam).

The total number of time samples needed for gridding for a full observation for a particular baseline ($B_{\mathrm{i,j}}$) is then:

$$N_{\mathrm{time\,samples}}(B_{\mathrm{i,j}}) = \frac{t_{\mathrm{obs}}}{(t_{\mathrm{dump}} \times N_{\mathrm{t,coalesce}}(B_{\mathrm{i,j}}))} \tag{48}$$

We can likewise gather together u, v, w points from the same antenna parings that come from adjacent frequency channels, provided that we do not exceed the frequency smearing limit or the limit imposed by the required frequency resolution of the output image cubes (which need $N_{\mathrm{f,out}}$ points to cover the frequency axis) or the number of FFT grids we need to cover the frequency axis to maintain distributability in the SDP ($N_{\mathrm{f,distribute}}$).

We estimate the number of such frequency channels needed at to prevent smearing at the full field of view for a particular baseline B is:

$$N_{\mathrm{f}}^{\text{full FoV}}(B) = \mathrm{Max}\left(N_{\mathrm{f,smear}}^{\text{full FoV}}(B), N_{\mathrm{f,distribute}}, N_{\mathrm{f,out}}\right) \tag{49}$$

Where $N_{\mathrm{f,smear}}^{\text{full FoV}}$ is given by equation 46, but with the full field of view applied.

This full-field of view smearing-limited averaging give a visibility rate many times lower than the correlator visibility rate, and we denote this this visibility rate $R_{\mathrm{vis,full\,FoV}}$.

However, if faceting is employed the uv cell size in the grids is increased (i.e. the field of view per facet is reduced) and we can perform a further data-reduction scheme (which we refer to as "coalescing") in which adjacent (already averaged) visibilities are gathered together immediately prior to gridding. The smearing equations are identical, but the facet field of view is used. The log expansion in equation 46 and the form of 45 mean that we get a near-proportional reduction in both the required number of frequency channels and time samples with N_{facet}. We estimate the number of such frequency channels needed at this gridding step for a particular baseline B is:

$$N_{\mathrm{f}}^{\text{facet FoV}}(B) = \mathrm{Max}\left(N_{\mathrm{f,smear}}^{\text{facet FoV}}(B), N_{\mathrm{f,distribute}}, N_{\mathrm{f,out}}\right) \tag{50}$$

Where $N_{\mathrm{f,smear}}^{\text{facet FoV}}$ is given by equation 46, but with the facet field of view applied.

The facet field of view smearing limits give another visibility rate, a factor of roughly [6] N_{facet}^2 times lower than $R_{\mathrm{vis,full\,FoV}}$, we denote this as $R_{\mathrm{vis,facet\,FoV}}$.

$$R_{\mathrm{vis,full\,FoV}} = N_{\mathrm{beam}} N_{\mathrm{pol}} \sum_{\mathrm{i,j}}^{\text{baselines}} \frac{N_{\mathrm{f}}^{\text{full fov}}(B) N_{\text{time samples}^{\text{full FoV}}}(B)}{t_{\mathrm{obs}}} \tag{51}$$

$$R_{\mathrm{vis,facet\,FoV}} = N_{\mathrm{beam}} N_{\mathrm{pol}} \sum_{\mathrm{i,j}}^{\text{baselines}} \frac{N_{\mathrm{f}}^{\text{facet FoV}}(B) N_{\text{time samples}^{\text{facet FoV}}}(B)}{t_{\mathrm{obs}}} \tag{52}$$

And, note that (of course) the factor of N_{bl} (number of baselines) that one might anticipate here is subsumed into the sum, which is specifically a sum over every single antenna pairing.

D The Resolution and Extent of Images and uv Planes

D.1 Image Size

The size of the images and the associated uv planes depends on the angular size of the pixels and on the angular size of the field to be imaged.

The full-width half-maximum of the primary beam size depends on the dish or station diameter (D_{s}) and the illumination pattern / element weighting strategy of the telescope. For example, it is about $1.17\,\lambda/D_{\mathrm{s}}$ for a -12 dB Gaussian tapered illumination and about $1.4\lambda/D_{\mathrm{s}}$ for LOFAR. The area to be imaged must necessarily include the full-width half-maximum of the primary beam in order to retain all of the information that the telescope is collecting. It is however usually necessary to image a larger area in order to be able to accurately subtract the sidelobes of the sources outside the primary beam.

[6] in practice, we don't quite get this scaling as we hit fundamental limits in frequency and time resolution arising from other constraints, and for high frequency-resolution spectral line experiments, we are unable to average in frequency.

We therefore define the diameter of the field of view θ_{FoV} to be some factor Q_{FoV} larger than the diameter of the first null of the Bessel function of first kind (i.e., the Airy disk):

$$\theta_{\mathrm{FoV}} = \frac{7.6\lambda_{\mathrm{sub,max}} Q_{\mathrm{FoV}}}{\pi D_{\mathrm{s}}} \tag{53}$$

where the relevant wavelength is the maximum wavelength to be used in the imaging of the single sub-band over which pixel sizes are going to be matched, $\lambda_{\mathrm{sub,max}}$. Therefore $Q_{\mathrm{FoV}} = 1$ corresponds to a field of view that encloses the first zero of the Bessel function of the first kind; $Q_{\mathrm{FoV}} = 1.8$ to the second zero; $Q_{\mathrm{FoV}} = 2.4$ to the third zero.

For reference, $Q_{\mathrm{FoV}} = 0.48$ corresponds to the full-width half-maximum of a 12 dB Gaussian tapered illumination and $Q_{\mathrm{FoV}} = 0.57$ corresponds to the full-width half-maximum of the LOFAR beam.

D.2 Image Plane Pixel Size

The size of the synthesised beam depends on the weighting scheme with a minimum of about λ/B_{max}, but in practice for well-filled arrays it is usually well approximated by a diameter of $\theta_{\mathrm{PSF}} = 3\lambda/2B_{\mathrm{max}}$. The size of the pixels can then be defined in terms of this practical synthesised beam size as:

$$\theta_{\mathrm{pix}} = \frac{3\lambda_{\mathrm{sub,min}}}{4B_{\mathrm{max}} Q_{\mathrm{pix}}} \tag{54}$$

where the relevant wavelength is the shortest wavelength to be used in the in the sub-band imaging, $\lambda_{\mathrm{sub,min}}$ and Q_{pix} is the oversampling factor above Nyquist. A value $Q_{\mathrm{pix}} = 1.5$ corresponds to three pixels per synthesised beam which is the minimum used in practice and the value $Q_{\mathrm{pix}} = 2.5$ corresponds to five pixels per synthesised beam. For comparison, the CASA routine Imager::advise [RD15] gives the pixel size as the inverse of twice the maximum uv distance, giving $\theta_{\mathrm{pix}}^{\mathrm{CASA}} = \lambda_{\mathrm{sub,min}}/2B_{\mathrm{max}}$ which corresponds to $Q_{\mathrm{pix}} = 1.5$ in the relation above.

D.3 Number of Pixels on Image or Grid Side

The linear size of the uv planes and images is then:

$$N_{\mathrm{pix}} = \frac{\theta_{\mathrm{FoV}}}{\theta_{\mathrm{pix}}} = \frac{10B_{\mathrm{max}} Q_{\mathrm{pix}} Q_{\mathrm{FoV}}}{\pi D_{\mathrm{s}}} \frac{\lambda_{\mathrm{sub,max}}}{\lambda_{\mathrm{sub,min}}} \tag{55}$$

where B_{max} is the maximum baseline length and D_{s} is the diameter of dish or station. As described above, the use of $\lambda_{\mathrm{sub,max}}$ and $\lambda_{\mathrm{sub,min}}$ is to allow sub-bands over which the pixel size is matched, decreasing the apparent efficiency but making operations across different frequency channels significantly easier to implement. The case $\lambda_{\mathrm{sub,max}} = \lambda_{\mathrm{sub,min}}$ is equivalent to having just a single fiducial wavelength.

E Re-use of Convolution Kernels

One significant contributor to the total number of FLOPS needed in the SDP is the calculation of the convolution kernels themselves. These kernels are antenna-pair specific and need to be updated every major imaging cycle.

To reduce this load, we assume that the same convolution kernel can be used across a range of different *uv* points adjacent in time and frequency.

E.1 Update Timescale for Convolution Kernels, $t_{\mathrm{kernel,update}}$

In the parametric model we assume that convolution kernels can be applied to adjacent visibility points from the same antenna pairings at the same frequency across a limited length of observing time. We call this the "kernel update timescale", $t_{\mathrm{kernel,update}}$.

Our current assumption is that kernel re-use is limited by the time variation in calibration solutions, rather than by imaging effects, and we adopt a default value of 10 s for this. Thus the number of convolution kernels needed to cover the observation is $N_{\mathrm{kernels,time}} = t_{\mathrm{obs}}/t_{\mathrm{kernel,update}}$.

E.2 Update Frequency Scale for Convolution Kernels

We have used a frequency smearing limit to estimate the range of frequencies that can be gridded simultaneously. This limit was dependent on the length of the baseline (in units of uv cells) and on the fraction of a uv cell that we allowed the visibility track to drift across.

Similar logic leads to a limit on the re-use of kernels: because the convolution kernels themselves are finite in size, using them a a frequency different from the one they are calculated for risks misplacing the convolved visibilities, especially at the edge of the convolution kernel. We want to ensure that errors from convolution kernel re-use remain much smaller than any occurring through averaging. This leads us to propose a new parameter, Q_{kernel} which links the fraction of a cell smearing is allowed through (ϵ_f) to the fraction of a uv cell that convolution kernel re-use is allowed to compromise, such that the kernel reuse cell fraction is lower by a factor of Q_{kernel} than ϵ_f. We assume a default value of 10 here, so if we allow smearing through (say) $1/3$rd of a cell, kernel re-use positional errors would be smaller than $1/30$th of a cell at the edge of the kernel.

$$N_{\mathrm{f,kernel}}(B) = \frac{\log\left(\frac{\lambda_{\mathrm{max}}}{\lambda_{\mathrm{min}}}\right)}{\log\left(\frac{\epsilon_f}{Q_{\mathrm{kernel}} N_{\mathrm{pix,kernel}}(B)}\right)}, \tag{56}$$

where $N_{\mathrm{pix,kernel}}(B)$ is the size of the convolution kernel, given in section G.3.

F Working Memory Required for Grids

The uv grids have N^2_{pix} pixels which have to be double precision complex values each requiring 16 bytes of internal storage, so the memory required per grid ($M_{uv,\mathrm{grid}}$, in bytes) is:

$$M_{uv,\mathrm{grid}} = 16 N^2_{\mathrm{pix}} \tag{57}$$

This size of working memory is required for gridding the visibilities, for gridding the point-spread functions (PSFs) and for the Fast Fourier Transforms (FFTs)[7] (which are taken to be in-place here).

The grids used for accumulating weights are real-valued, therefore:

$$M_{\mathrm{weight\_grid}} = 8 N^2_{\mathrm{pix}} \tag{58}$$

This is the minimum working memory requirement (expressed in bytes) for the image weight calculation.

The images used in the intermediate processing, i.e., the dirty images, PSFs and sensitivity maps as well as intermediate images for CLEAN are all also real-valued and probably require double precision. Thus:

$$M_{\mathrm{image}} = 8 N^2_{\mathrm{pix}} \tag{59}$$

This is the required working memory (in bytes) for any whole-image calculations. Re-projection requires minimum M_{image} if it operates on whole images but can relatively easily be decomposed into operations

[7] It is beneficial to include additional padding for the FFT operations, typically 20-30% in linear size. This is not yet included.

into sections of images. Minor cycle CLEAN likewise works on the M_{image} working set in the most straightforward implementation.

After all the processing is done Image data will usually be stored as single-precision 32-bit floating point numbers, but for very high dynamic range it might be required to store it with higher precision. Using 64-bit floating point numbers for storage might be overkill. Instead the mantissa of the 64-bit floats could be truncated to store it in, say, 48 bits. The DATA layer could take care of such "compression".

G Convolution Kernel Sizes

G.1 Size of the w-kernel

The sizing of the w-projection kernels is given [RD21] by the expression:

$$|u| = \sqrt{\left(\frac{w\theta_{\text{FoV}}}{2}\right)^2 + \frac{w\theta_{\text{FoV}}/2}{\pi\epsilon_\omega/\sqrt{w}}} \tag{60}$$

which shows the value of u for which the envelope of the amplitude of the convolution function declines to ϵ_ω of the maximum value. This is taken to be the radius of the convolution kernels. The linear size of the convolution functions in pixels is twice this radius divided by the size of the uv-plane cells (which are $1/\theta_{\text{FoV}}$), giving an overall expression $N_{\text{GW}}(w)$ which is:

$$N_{\text{GW}}(w) = 2\theta_{\text{FoV}}\sqrt{\left(\frac{w\theta_{\text{FoV}}}{2}\right)^2 + \frac{w^{3/2}\theta_{\text{FoV}}}{2\pi\epsilon_\omega}} \tag{61}$$

where θ_{FoV} is the field of view as previously and ϵ_ω is the amplitude level of the w function to which we want the kernel to extend.

This use of the first order expansion in ϵ_ω of the support of the w-convolution function is in contrast with previous work [RD22, RD23 and RD24]. The impact of this change is an increase in the size of the convolution function.

As described above, we size the kernel for maximum w deviation (Δw_{max}) given snapshot duration and maximum baseline length:

$$\Delta w_{\text{max}} = \frac{B_{\text{max}}\Omega_{\text{E}}t_{\text{snap}}Q_w}{2\lambda_{\text{min}}} \tag{62}$$

Here the factor of two in the denominator accounts for the fact that the distance from the centre of the array, which is approximately half the maximum baseline, is the appropriate distance to use. Parameter Q_w is in the range between 0 and 1 and allows for the possibility that w-correction is done for less than the maximal w deviation, e.g., because high-w visibilities are to be discarded or are imperfectly corrected, or to account for the fact that many visibilities will need kernels smaller than the kernel size set by Δw_{max} (e.g. RD23 recommends the range 0.1-0.3, see also RD25 for the discarding of high-w visibilities). N_{GW} is the size of the support of the w kernel evaluated at maximum w, i.e., $N_{\text{GW}} = N_{\text{GW}}(\Delta w_{\text{max}})$.

G.2 Size of the A-kernel

The A-kernel (of a homogeneous array) is the Fourier transform of the auto-correlation of the antenna power pattern, $P(l, m)$. Since this is effectively twice the size of the illumination pattern, the size of the A-kernel in uv-space, $N_{\text{AA}uv}$ is given by,

$$N_{\text{AA}uv} = \frac{2D_{\text{s}}}{\lambda} \tag{63}$$

In producing the A-kernel, $P(l, m)$ is truncated at some distance from the pointing centre (to exclude side-lobes, for example) and is padded with zeros (equivalent to multiplying with a box function).

Fourier transforming this will produce an oversampled A-kernel with resolution, Θ_{AA}. Thus, for a fixed dish diameter and observing wavelength the support size of the A-kernel N_{AA} is given by,

$$N_{\mathrm{AA}} = \frac{N_{\mathrm{AA}uv}}{\Theta_{AA}} \tag{64}$$

In practice, in we simply parametrise N_{AA} in the model, adopting a value of 9 uv grid cells for its total width.

G.3 Overall Convolution Kernel Size

We assume that the A and W terms in the convolution kernel combine in quadrature so the total size of the convolution kernels (in uv pixels) is:

$$N_{\mathrm{pix,kernel}} = \sqrt{N_{\mathrm{AA}}^2 + N_{\mathrm{GW}}^2} \tag{65}$$

When we calculate the kernels we oversample by a factor of Q_{GCF} so the support size (N_{cvff})is proportionally bigger than $N_{\mathrm{pix,kernel}}$.

H Imaging Assumptions

1. The convention is that each floating point operation is one of the basic operations (addition or multiplication). Even when an obvious fused multiply and add is present it is counted as two operations. All floating point operations are assumed double-precision. Some efficiency savings may possible once an analysis of which steps are possible in single-precision is done.

2. A traditional major-cycle model for CLEAN and self-calibration is assumed with both the “forward” step (modelling observed visibilities from a trial sky) and the “backward” step (estimating a sky brightness distribution from the observed visibilities) included in the performance model.

3. We assume imaging (see RD17 for an overview) is done using the AW-projection algorithm [RD18] together with the w-snapshots technique for controlling the size of the w convolution kernels [RD19]. Work is ongoing to analyse alternative algorithms but we do not believe this would fundamentally alter the numbers presented here.

4. Both the number of facets and the duration of the w-snapshot are optimised to maximise the overall processing throughput. Currently this optimisation is done by minimising the flop count required for imaging.

5. It is assumed that the sizes of w and A kernels combine in quadrature. The more conservative assumption would be to assume that the sizes combine linearly, giving a maximum of $\sqrt{2}$ increase in linear size and a factor of two increase in the FLOPS requirement for gridding and for the calculation of the kernels themselves.

H.1 Imaging Pipeline Geometry Assumptions

For the parametric modelling we assume the worst case scenario in which w is changing as fast as possible, i.e., the angular velocity of the Earth ($\Omega_{\mathrm{E}} = 7.27 \times 10^{-5}$ rad/s) times one half of the maximum baseline length.

It is assumed that the curvature of the Earth dominates the array non-coplanarity according to:

$$\Delta w_{\mathrm{min}} = \frac{B_{\mathrm{max}}^2}{8 R_{\mathrm{Earth}} \lambda} \tag{66}$$

where $R_{\mathrm{Earth}} = 6400 \times 10^3$ m is the radius of the Earth. This is for a plane with $w = 0$ at the centre of the array; smaller deviations are appropriate for a best-fitting plane but this depends on the distribution of the telescopes. This curvature limits the minimum useful w-snapshot time. We calculate this time to be the time that the change of w due to Earth's rotation is equal to the curvature of the earth, leading to:

$$t_{\mathrm{snap,min}} = \frac{2\lambda\Delta w_{\mathrm{min}}}{B_{\mathrm{max}}\Omega_{\mathrm{E}}} = \frac{B_{\mathrm{max}}}{4\Omega_{\mathrm{E}}R_{\mathrm{Earth}}} \tag{67}$$

H.2 Faceting

A description of the conventional faceting approach is given in RD27 while a description of the uv faceting is given in RD28. It is likely that any faceting in the SDP will be based on the uv faceting technique. In faceting, each visibility datum must be phase rotated to the direction of each facet and its u, v, w coordinates recomputed.

After the visibilities have been transformed, the imaging proceeds for each facet largely as before, but for a field of view which is N_{facet} smaller and the whole imaging operation is repeated N^2_{facet} times to account for the number of facets.

Since the reason for using facets in the SDP is to reduce working memory rather than control the w terms, relatively few facets should be sufficient (e.g. $N_{\mathrm{facet}} \sim 4$) while on the other hand the faceting approach is most relevant at the widest fields of view where $N_{\mathrm{pix,kernel}}$ is typically greater than 50.

In the model here, N_{facet} is a free parameter. However in the iPython implementation of the parametric model we currently choose the value of N_{facet} and t_{snap} to minimise the total number of FLOPS required for each pipeline. However, the minimum of this curve is quite shallow and we may well be forced (e.g. by interconnect rate limitations) to chose a lower value of N_{facet} than the one which minimises the FLOPS.

H.3 Working Memory Requirements

In this section some key working memory needs are identified and associated with particular elements of the imaging process. In general, there may be efficiency savings by keeping several working memory areas together so that some intermediate results or cached items are reused. Additionally, there are strategies of dividing grids such that smaller working memory sets can be used while retaining quite high efficiency. One example of this is faceting but simpler schemes, which operate only within a step are possible as well. This is not yet analysed here, but rather the approach is to identify the size of the individual grids assuming they are not further sub-divided. This is appropriate so for example an FFT can be efficiently done, or gridding without any pre-filtering on uv coordinates.

H.3.1 Target Grid Size in Faceting

The target grid size is defined per facet and can be calculated as

$$M_{\mathrm{target}} = \frac{M_{uv,\mathrm{grid}} \times r^2_{\mathrm{facet}}}{N^2_{\mathrm{facet}}} = \frac{16N^2_{\mathrm{pix}} \times r^2_{\mathrm{facet}}}{N^2_{\mathrm{facet}}} \tag{68}$$

Where r_{facet} is the ratio by which facets are larger than one might expect if naively dividing the total field of view linearly by the number of facets on a side - we typically expect them to be 20-50% bigger than that to give some overlapping regions on adjacent facets (so $r_{\mathrm{facet}} \sim 1.2 - 1.5$).

H.3.2 Convolution Kernel Cache Size

Beside the target grids, another important memory requirement is the memory used to cache the w-convolution kernels.

The cache size is calculated by assuming that the size of each kernel is $8N_{\mathrm{GW}}^2$ bytes, that the number of w-planes is the w-kernel oversampling times the size of the kernels in each angular direction (i.e., $N_{\mathrm{GW}}Q_{\mathrm{GCF}}$), that the oversampling is Q_{GCF} as previously assumed and that there is a separate kernel for each beam and facet:

$$M_{w,\mathrm{cache}} = N_{\mathrm{GW}}^3 Q_{\mathrm{GCF}}^3 N_{\mathrm{beam}} N_{\mathrm{facet}} \times 8\,\mathrm{bytes} \tag{69}$$

Here N_{GW} is the linear size of the convolution kernel in one of the angular dimensions and is evaluated at the optimal w-snapshot duration. The w-function cache above is sized to include the oversampled functions. When baseline-dependent kernels are recomputed repeatedly, the oversampling of the uv directions is regenerated. So in the case of such re-computation of kernels on the fly there is no need to permanently cache the uv oversampling of the w functions. This oversampling at the point of computing the combined Aw kernels is implemented in the LOFAR software.

I Appendix: Data Sizes

Visibilities consist of 1 complex float (single-precision) per time slot, per frequency channel, per beam, per baseline, and per polarisation. Hence, the total number of visibility points (including auto-correlations) for an observation is:

$$N_{\mathrm{vis}} = \frac{1}{2}\frac{t_{\mathrm{obs}}}{t_{\mathrm{dump}}} N_{\mathrm{f}} N_{\mathrm{beam}} N_{\mathrm{a}} (N_{\mathrm{a}} + 1) N_{\mathrm{pol}} \tag{70}$$

The SKA1 baseline numbers for these parameters are (see AD07, AD08 and the appendix of RD02):

- The minimum t_{dump} is specified as 0.9 s for Low and 0.14 s for Mid.
- The maximum number of frequency channels, N_{f}: 65,536 for both Low and Mid.
- Number of beams, N_{beam}: 1 for SKA1-Low and Mid
- Number of antennas (or stations), N_{a}: 512 for SKA1-Low, and 197 for Mid.
- Number of polarisations, $N_{\mathrm{pol}} = 4$, for both SKA1 instruments.

The maximum visibility rate from each instrument is very similar, at around 40 Giga-visibilities per second.

I.1 Visibility Meta Data

The visibility data has various attributes. The definition of the Measurement Set [RD13, RD14] offers a good starting point and makes baseline-dependent averaging possible. Several of the attributes can be a 2-dimensional array making it possible that multiple visibilities share the same metadata tags. In this way the metadata size can be kept to a minimum.

The size of such a data object is $55 + N_{\mathrm{f}} \times N_{\mathrm{pol}} \times 37$ Bytes, thus approximately 37 KBytes for $N_{\mathrm{pol}} = 4$ polarisations and $N_{\mathrm{f}} = 256$ frequency channels. For LOFAR (Low Frequency Array), where the data is stored in Measurement Sets of 256 frequency channels each, the meta data excluding DATA, FLAGS and WEIGHTS is less than 1% of the total data volume.

Alternatively, multiple baselines could be stored in a single data object, where it is required that all of them have the same shape (in case baseline-dependent averaging is done). The BASELINE field will become a 2-dim array.

Field	Type	Shape	Comment
DATA	`complex <float>`	$N_{\mathrm{f}}, N_{\mathrm{pol}}$	N_{f} = number of frequency channels N_{pol} = number of correlations (polarisations).
FLAGS	`bool`	$N_{\mathrm{f}}, N_{\mathrm{pol}}$	Not strictly needed; WEIGHT==0 is the same. A single flag per visibility is assumed to be sufficient. I.e., it is not needed to have separate flags for different types of flagging. Shape could be $N_{\mathrm{f}}, 1$ if all polarisations have the same flag (e.g., for SKA-Low).
WEIGHTS	`float`	$N_{\mathrm{f}}, N_{\mathrm{pol}}$	VisShape could be $N_{\mathrm{f}}, 1$ if all polarisations have the same weight (e.g., for SKA-Low).
UVW	`double`	3	In metres. If each frequency channel has its own time centroid, UVW has to be a 2-dim array with N_{f} as the first axis (and could be in wavelengths).
TIME	`double`		Mid of exposure interval (seconds) since the start of the observation. This gives a very high precision (pico seconds).
EXPOSURE	`double`		Exposure time in seconds. If it is constant for the full observation, it can be part of the observation metadata.
BASELINE	`int16`	2	Antenna ids forming the baseline. The antenna setup is described in the metadata.
OBSID	`int64?`		Id of observation in metadata.
FREQGROUP	`int16`		Id of frequency group in metadata.
CORR	`unsigned char`		Bit pattern indicating the polarisations used (XX,XY,YX,YY)
FIELD	`int16`		Id of field definition in meta data.

Table 16: Data fields from the Measurement Set.

I.2 Image Meta Data

The meta data describing an image can consist of various parts:

- The description of the axes of the image cube (spatial, spectral, polarisation, time). Typically the WCS (World Coordinate System) format will be used for it.
- The provenance data of the image describing the processing history of the data.
- Some metrics describing the quality of the image.

Usually the meta data makes up a small part of the image size, but if a lot of provenance data are to be kept, it can grow to up to 5% of the size of continuum images. Spectral line cubes are much larger, so the meta data size will always be a small fraction of their sizes. An example of an image format with full provenance is the LOFAR image format as described in RD16.

I.3 Buffer Requirements

I.3.1 Visibilities Buffer

The visibility buffer is sized by allowing space for two full observations of duration t_{obs}. Each visibility in the visibility buffer is assumed to consume M_{vis} bytes. This is currently understood to be at minimum

12 bytes: 8 bytes for a single-precision complex floating point visibility value and 4 bytes for time centroid and flags. The value of $M_{\mathrm{vis}} = 12$ however implies that the u, v, w coordinates are re-calculated at each read. The size of the buffer is then:

$$M_{\mathrm{buf,vis}} = 2N_{\mathrm{pol}}N_{\mathrm{vis}}M_{\mathrm{vis}}t_{\mathrm{obs}} \tag{71}$$

It is assumed that the uv data from the buffer are read once per full major cycle. The data also have to be written to the buffer. The required bandwidth to the buffer for the entire pipeline processing is:

$$R_{\mathrm{bw,I/O}} = (1 + N_{\mathrm{major}})N_{\mathrm{vis}}M_{\mathrm{vis}} \tag{72}$$

I.3.2 Model Sky Buffer

The reconstructed model sky will need to be held in memory during pipeline runs as well as being delivered as a data product at the end of the pipeline processing. The model image is a real-valued data-set with $N_{\mathrm{byte,pix}} = 8$ bytes per double precision value. The size of the buffer is then:

$$M_{\mathrm{buf,skymod}} = N_{\mathrm{pol}}N_{\mathrm{Tt}}N_{\mathrm{pix}}N_{\mathrm{byte,pix}} \tag{73}$$

Assuming that the Major Cycle is based on a multi-scale multi-frequency synthesis (MS-MFS) deconvolution [RD38], a number of additional different intermediate gridded data products will need to be held in memory at different stages of the cycle. The exact size of the required buffers per compute node depends on the detail of the distribution within the Major Cycle.

J Design Equations

There are a significant number of possible choices in the design of the SDP compute hardware. Examples are: types and specifications of the processing units; number of units per compute node; number of nodes per rack; nature of the storage system and the network bandwidths and topologies. The design equations drive constraints on these free parameters of the design from the computational requirements analysis presented in this document. The feasible parameter space defined by the design equations and the evolution of available COTS hardware components defines the design space of the compute platform of the SDP. The design equations below are derived from the parametric model described in this document. Scaling of the SDP deployments is based on these design equations, within constraints described in the SDP Data Processor Platform Design document [RD37]. This analysis is part of an ongoing piece of work and the numbers given here should not be used in earnest - we have included them here because they give context to our current design work only: they (along with the total compute rates) will change in the future as the models and use cases we apply them to develop.

While it is unlikely we can, at least initially, afford the aggregate capacities described below, these equations clearly show the ratios in which various characteristics are ideally needed. The final chosen design points will be selected to maximise the scientific output whilst fitting within a budget for capital and operational cost.

The key quantities that are addressed in the design equations are:

- N_{cu}: The number of compute units. These are defined as units with very high bandwidth to working memory and shared-memory parallelism.
- C_{peak}: The peak FLOPS capability of the compute unit
- C_{max}: The maximum FLOPS capability that a compute unit delivers in practice
- $R_{\mathrm{bw,max}}$: The maximum memory bandwidth of each compute unit to its main, high-throughput, working memory. For accelerator-type compute units this memory is taken to be the on-board memory.

- $R_{\mathrm{bw,I/O,max}}$: The maximum I/O bandwidth of each compute unit to buffer
- $M_{\mathrm{cu,work}}$: Size of working memory of the compute unit. This is the memory with bandwidth as described by $R_{\mathrm{bw,max}}$.
- $M_{\mathrm{cu,pool}}$: This is the slower working memory to which working grids etc. are swapped out to when not being actively worked on. For accelerator-based systems this could be DRAM on the main board or eventually new high-throughput NVRAM technology.
- $M_{\mathrm{cu,buf}}$: Size of the buffer attached to each compute unit (or its share of data-island local buffer).

In RD02 several processing pipelines are defined. Pipelines containing an Imaging Component are expected to be the most demanding to the system. Of these, the imaging processing pipelines are identified that need to be run consecutively on a fast visibility buffer: first the ICAL and then the DPrep A,B,C pipelines (as needed). In addition, the Fast Imaging Pipeline needs to run in parallel in a streaming fashion. It is assumed that floating point operations and memory bandwidth are the limitation for these pipelines; therefore the requirements for FLOPS rate and memory bandwidth are the combination of the requirements for these pipelines while other constraints are sized so that they are not a limiting factor. The simultaneous focus on both the peak FLOPS capability and the memory bandwidth is along the lines of RD29.

The following relationships can be established:

1. The peak FLOPS capability of the units taken together has to exceed the total FLOPS required for all pipelines combined, and when all are run at their maximum rates.
2. The total memory bandwidth available to all compute units must be greater than the total memory bandwidth summed for all required pipelines.
3. The visibility buffer of all compute units together must be sufficiently large to hold both the full-resolution visibilities and frequency-binned visibilities for continuum processing for 2 observations (to give a double buffer):
   $N_{\mathrm{cu}} M_{\mathrm{cu,buf}} > 2 \times (R_{\mathrm{vis,corr}} + R_{\mathrm{vis,full\,FoV}}) \times t_{\mathrm{obs}}^{\mathrm{max}}$.
   Additionally, the buffer will need to hold the intermediate grids and images. These have not yet been taken into account.
4. The total bandwidth to the visibility buffer must exceed the greatest of the bandwidths for any single pipeline.
5. If frequency-polarisation-beam parallelism only is to be used with no faceting or other image plane or uv-plane division, then:
   (a) Working memory per compute unit must be greater than the target grid memory: $M_{\mathrm{cu,work}} > M_{uv,\mathrm{grid}}$.
   (b) Each compute unit must have reasonably fast memory to keep the grids not being immediately worked on ($M_{\mathrm{cu,pool}}$). The experience of ASKAPSoft is that about ten copies are necessary, giving: $M_{\mathrm{cu,pool}} > 10 M_{uv,\mathrm{grid}}$.
6. When faceting is added as an additional parallelisation step and we use the numbers calculated for faceting from RD01 we get the design equations as given in Table 17.

   Clearly the details of how the tasks in the SDP will be scheduled are very important; further constrains on the number of compute islands and their properties will emerge as different parallelism schemes are considered in the next stage of the development of our model for the SDP.

Design equation	SKA1-Low	SKA1-Mid (Band 1)
$N_{\mathrm{cu}}C$	>35 PFLOPS	>74 PFLOPS
$N_{\mathrm{cu}}R_{\mathrm{bw,mem}}$	>69 PB/s	>148 PB/s
$N_{\mathrm{cu}}M_{\mathrm{cu,buf}}$	>23 PB	>23 PB
$N_{\mathrm{cu}}R_{\mathrm{bw,I/O}}$	>15 TB/s	>18 TB/s
$M_{\mathrm{cu,work}}$	>0.28 TB	>1.23 TB
$M_{\mathrm{cu,pool}}$	>2.84 TB	>12 TB
C	>2.8 TFLOPS	>8.1 TFLOPS

Table 17: SDP design equations.These are very preliminary and should not be used apart from to provide guidance.

We note that our computational intensity, i.e. $\frac{N_{\mathrm{cu}}C}{N_{\mathrm{cu}}R_{\mathrm{bw,mem}}}$, is extremely low based on these design equations. We recall Gene Amdahl's rules for a balanced system, that state that a system needs a bit of memory I/O and a byte of memory for each instruction. How these old rules apply to current day hardware is well beyond the scope of this document, but our applications require approximately 16 bits/s of memory bandwidth per FLOPS (i.e. 16 bits per flop). In contrast, current state of the art systems provide only one bit of memory I/O per (double precision) flop for typical accelerators, to one bit of I/O per 0.6 flop for a typical server processor. Observe that the design equations currently mandate a very imbalanced system, heavily focused on memory bandwidth rather than compute capacity. The trend over the last couple of decades has been that memory bandwidth growth does not keep up with the increase in compute capacity, so we can expect the discrepancy between model and realistic hardware solutions to widen, rather than close.

Our design equations therefore point to a possible mismatch between our application requirements and the available hardware. We do point our that our modelling to date has been focused on minimising compute requirements, not memory accesses. This has resulted in a steady decrease in required compute capacity, without regard for the memory access requirements.

It is clear that in the next phase of the project we have the opportunity to take the model described above to guide optimisation to a realistic hardware implementation. A more balanced required ratio of compute, memory bandwidth and memory capacity will encourage efficient use of the available hardware.

We are considering faceting to reduce the working memory size requirements for SDP. Table 17 shows that this requirement is currently modest in comparison to the flop count. We can conceivably balance the system by reducing the number of facets, which has a linear (but opposite) effect on both working memory size and memory bandwidth. An order of magnitude reduction in the number of facets results in two orders of magnitude lower memory bandwidth requirement, but an increase in working memory requirement by the same order. We currently optimise for minimum flop count, but we have noted that this is a shallow minimum, so a trade-off can be made where additional memory capacity and flops are sacrificed to reduce memory bandwidth. Such a detailed and comprehensive hardware / software co-design approach should result in a more achievable system design. In theory such a system should be able to achieve better computational efficiency compared to what we see in our benchmarks today.

Applicable and reference documents

Applicable Documents

The following documents are applicable to the extent stated herein. In the event of conflict between the contents of the applicable documents and this document, *the applicable documents* shall take precedence.

Reference Number	Reference
AD01	Dewdney, P. E. (2013). SKA1 System Baseline Design. SKA Office
AD02	McCool, R., Cornwell, T. (2013). Miscellaneous Corrections to the Baseline Design
AD03	SKA Phase 1 System (Level 1) Requirements Specification, T. Cornwell, SKA-OFF.SE.ARC-SKO-SRS-001_2
AD04	PDR.01 SDP.ARCH document
AD05	SDP Costing spreadsheet
AD06	Cornwell, T. J. (2015). SKA1 Telescope Calibration Framework. SKA Office (draft version)
AD07	SKA_100-000000-002 SKA1_Low SDP to CSP Interface Control Document
AD08	SKA_300-000000-002 SKA1_Mid SDP to CSP Interface Control Document
AD09	SKA-TEL-SDP-0000013, P. Alexander et al, SDP Architecture

Reference Documents

The following documents are referenced in this document. In the event of conflict between the contents of the referenced documents and this document, *this document* shall take precedence.

Reference Number	Reference
RD01	SKA-TEL-SDP-0000041, F. Malan: iPython Performance Model Description
RD02	SKA-TEL-SDP-0000027, R. Nijboer: Pipelines Element Subsystem Design
RD03	SKA-TEL-SDP-0000038, R. Bolton: High Priority Science Objectives: Performance Analysis
RD04	SKA-TEL-SDP-0000028, G. van Diepen: Receiving and Pre-processing Visibility Data
RD05	SKA-TEL-SDP-0000029, S. Salvini: Pipelines: Calibration
RD06	SKA-TEL-SDP-0000030, A. Scaife: Imaging Pipeline
RD07	SKA-TEL-SDP-0000031, M. Johnston-Hollitt: Science Analysis Pipeline
RD08	SKA-TEL-SDP-0000017, S. Wijnholds: Baseline-Dependent Averaging
RD09	SKA-TEL-SDP-0000058, S. Salvini: Fast Fourier Transforms
RD10	SKA-TEL-SDP-0000057, C. Skipper: Time and Channel Averaging
RD11	Alexander et al., Software and Computing CoDR, Analysis of requirements derived from the DRM, WP2-050.020.010-RR-001
RD12	SKA-TEL-SDP-0000016, R. Bolton: SDP Archive Size Estimates
RD13	`http://casacore.github.io/casacore-notes/229.html`
RD14	`https://confluence.ska-sdp.org/pages/viewpage.action?title=MS+in+UML&spaceKey=ARCH`
RD15	Imager.cc C++ source, CASA source code SVN revision 30821, `https://svn.cv.nrao.edu/svn/casa/trunk`
RD16	`http://www.lofar.org/operations/lib/exe/fetch.php?media=:public:documents:casa_image_for_lofar_0.03.00.pdf`
RD17	Parameterized deconvolution for wide-band radio synthesis imaging, Urvashi Rao Venkata, 2010, PhD thesis
RD18	S. Bhatnagar, T. J. (2008). Correcting direction-dependent gains in the deconvolution of radio interferometric images. A&A, 419-429
RD19	Veenboer, Bram, Matthias Petschow, and John W. Romein. "Image-Domain gridding on graphics processors." 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE, 2017.

Reference Number	Reference
RD19	Cornwell, T. J., Voronkov, M. A., Humphreys, B. (2012). Wide field imaging for the Square Kilometre Array.
RD20	An Efficient Work-Distribution Strategy for Gridding Radio-Telescope Data on GPUs, John W. Romein, ICSfftti12, June 25fftti29, 2012, San Servolo Island, Venice, Italy. Source code available at: `ftp://ftp.astron.nl/outgoing/romein/Gridding-0.2.tar.bz`
RD21	Mitchell, D. a. (2014). Analysis of w-projection kernel size. SKA SDP Consortium
RD22	Rik Jongerius, S. W. (2014). An End-to-End Computing Model for the Square Kilometre Array. IEEE Computer, volume 47, number 9
RD23	A parametric model for the calibration and imaging costs of SKA1, T. J. Cornwell, 12 Feb 2013, SKA Draft Memo
RD24	SDP Element Concept. Paul Alexander, Chris Broekema, Simon Ratcliffe, Rosie Bolton, Bojan Nikolic, 2013, SDP-PROP-DR-001-1 (part of SKA SDP Consortium proposal)
RD25	Tasse, C. a. (2013). Applying full polarization A-Projection to very wide field of view instruments: An imager for LOFAR. A&A.
RD26	Faceted imaging in AIPS, Kogan and Greisen, AIPS Memo 113, 2009
RD27	Radio-interferometric imaging of very large fields - The problem of non-coplanar arrays, Cornwell, T. J.; Perley, R. A., Astronomy and Astrophysics (ISSN 0004-6361), vol. 261, no. 1, p. 353-364
RD28	Wide Field Imaging at Low Frequencies, Sault, R. J., The Universe at Low Radio Frequencies, Proceedings of IAU Symposium 199, held 30 Nov - 4 Dec 1999, Pune, India. Edited by A. Pramesh Rao, G. Swarup, and Gopal-Krishna, 2002., p.508
RD29	Roofline: An Insightful Visual Performance Model for Multicore Architectures, Williams, Waterman and Patterson, DOI:10.1145/1498765.1498785, April 2009, Vol. 52, No. 4, Communications of the ACM
RD30	Whiting, M., 2012, Duchamp: a 3D source finder for spectral-line data, Mon. Not. R. Astron. Soc., Volume 421, Issue 4, pages 3242fftti3256, April 2012
RD31	Hales, C. A., Murphy, T., Curran, J. R., Middelberg, E., Gaensler, B. M., Norris, R. P., blobcat: software to catalogue flood-filled blobs in radio images of total intensity and linear polarization, Mon. Not. R. Astron. Soc., Volume 425, Issue 2, pages 1365-2966, September 2012
RD32	Hancock, P.J., Murphy, T., Gaensler, B.M., Hopkins, A., Curran, J.R., (2012), Compact continuum source finding for next generation radio surveys, Mon. Not. R. Astron. Soc. 422, 1812fftti1824
RD33	Whiting, M., Humphreys, B., (2012) Source-Finding for the Australian Square Kilometre Array Pathfinder, Publications of the Astronomical Society of Australia, Volume 29, Special Issue 03, 2012, pp 371-381.
RD34	SKA-TEL-SDP-0000080, L. Levin Preston: SDP NIP compute requirements.
RD35	TCC-SDP-151123-1-1, Cornwell, T.J. (2016) Extension of the SDP performance model for cross data-island connectivity
RD36	DOI: 10.1088/0004-637X/758/1/23 Condon J., et al (2012) Resolving the Radio Source Background: Deeper Understanding Through Confusion, http://arxiv.org/abs/1207.2439
RD37	SKA-TEL-SDP-0000018, P.C. Broekema: SDP Data Processor Platform Design
RD38	A multi-scale multi-frequency deconvolution algorithm for synthesis imaging in radio interferometry, Rau, Urvashi, and Tim J. Cornwell, Astronomy & Astrophysics 532 (2011): A71.

List of abbreviations

AOFlagger	André Offringa Flagger
ASKAP	Australian Square Kilometre Array Pathfinder
CASA	Common Astronomy Software Applications
CDR	Critical Design Review
CPU	Central Processing Unit
CSP	Central Signal Processor
DDE	Direction Dependent Effect
DIE	Direction Independent Effect
DFT	Direct Fourier Transform
DM	Dispersion Measure
DRAM	Dynamic Random-Access Memory
EoR	Epoch of Reionisation
FFT	Fast Fourier Transform
flop(s)	Floating Point Operation(s)
FLOPS	Floating Point Operations per Second
FoV	Field of View
GPU	Graphics Processing Unit
GSM	Global Sky Model
I/O	Input/Output
ICAL	Imaging and Calibration Pipeline
ICD	Interface Control Document
IFFT	Inverse Fast Fourier Transform
JVLA	Jansky Very Large Array
LOFAR	LOw Frequency ARray
LSM	Local Sky Model
MPI	Message Passing Interface
MS-MFS	Multi-Scale Multi-Frequency Synthesis
MSP	Milli-second Pulsar
NVRAM	Non-Volatile Random Access Memory
OCLD	Optimised Candidate List and Data
PDR	Preliminary Design Review
RFI	Radio Frequency Interference
PCI	Peripheral Component Interconnect
PM	Performance Model
PSF	Point Spread Function
PSS	Pulsar Search
PST	Pulsar Timing
PTA	Pulsar Timing Array
RAM	Random Access Memory
RCAL	Real time Calibration Pipeline
RFI	Radio Frequency Interference

SDP	Scientific Data Processor
SKA	Square Kilometre Array
SKAO	SKA Organisation
StEFCal	Statistically Efficient and Fast Calibration
TBC	To Be Confirmed
TBD	To Be Decided
TM	Telescope Manager
ToA	Time of Arrival
NVRAM	Non-Volatile Random-Access Memory
WCS	World Coordinate System

List of symbols

ϵ_ω	Fraction of maximal amplitude of envelope of convolution function
ϵ_f	Fraction of a uv cell that we allow uv track to move in time- or frequency- smearing-limited averaging
η_{comp}	Computational efficiency
η_f	Fraction of frequency band
θ_{bs}	Beam squint
θ_{FoV}	Angular diameter of the field of view (rad)
θ_{pix}	Angular resolution of pixel size (rad)
θ_{PSF}	Angular diameter of the synthesised beam (rad)
$\lambda_{\mathrm{sub,max}}$	Maximum wavelength of an imaging sub-band (channels over which image and pixel size are matched)
λ_{max}	Maximum wavelength
λ_{min}	Minimum wavelength
$\lambda_{\mathrm{sub,min}}$	Minimum wavelength of an imaging sub-band (channels over which image and pixel size are matched)
λ	Wavelength
ρ_{GSM}	Number of discrete sources in the GSM
Ω_{E}	Angular velocity of the Earth (7.27 10-5 rad/sec)
b_{c}	Length of a complex number in bytes
B_{max}	Maximum baseline length
b_{r}	Length of a real number in bytes
B_{mem}	Memory bandwidth
B_{PCIe}	PCI-e bandwidth
b	Length of a complex number in bytes
B	Baseline length
C	Compute rate (FLOPS)
C_{Average}	Compute rate (FLOPS) to do averaging
C_{Coalesce}	Compute rate (FLOPS) to do coalescing (specifically, averaging visibilities just prior to gridding).
C_{Demix}	Compute rate (FLOPS) to do demixing on input visibilities
C_{Dirsub}	Compute rate (FLOPS) to do direct strong source subtraction on input visibilities

C_{FFT}	Compute rate (FLOPS) to do FFT
C_{Flag}	Compute rate (FLOPS) to do flagging on input visibilities
C_{Grid}	Compute rate (FLOPS) to do gridding
C_{Kernels}	Compute rate (FLOPS) for generation of convolution kernels
$C_{\mathrm{minor\,clean}}$	Compute rate (FLOPS) for minor cycle image plane subtraction (clean)
C_{max}	Maximum FLOPS that a compute unit delivers
C_{peak}	Peak FLOPS of compute unit
C_{PR}	Phase rotation compute rate (FLOPS)
C_{Predict}	Compute rate (FLOPS) to do DFT predict
C_{Receive}	Compute rate (FLOPS) to do receive
C_{Reproj}	Compute rate (FLOPS) for re-projection to account for snapshot approach
C_{Solve}	Compute rate (FLOPS) to solve for calibration solutions
C_{SpecFit}	Compute rate (FLOPS) for spectral fitting
C_{Subtract}	Compute rate (FLOPS) for subtraction
C_{Weight}	Compute rate (FLOPS) calculation and alteration of visibility weights prior to gridding step.
$\mathbb{D}$	Matrix of observed visibilities
D_{s}	Diameter of antennas/stations
f_{ref}	Reference frequency
Δf	Frequency resolution
$\mathbb{G}$	2x2 block diagonal matrix of complex gains
$\mathbb{M}$	Matrix of model sky visibilities
$M_{uv,\mathrm{grid}}$	Size of the uv-grid (bytes)
$M_{w,\mathrm{cache}}$	Cache size of gridding kernel
$M_{\mathrm{buf,skymod}}$	Size of the sky model buffer
$M_{\mathrm{buf,vis}}$	Size of the visibility buffer
M_{burst}	Size of single pulse candidate data cube
M_{cand}	Size of pulsar candidate data cube
$M_{\mathrm{cu,buf}}$	Size of buffer
$M_{\mathrm{cu,pool}}$	Size of slow working memory
$M_{\mathrm{cu,work}}$	Size of working memory of compute unit
$M_{\mathrm{image,ax}}$	Image size along one axis
M_{image}	Size of image (bytes)
M_{input}	Size of input data cube
M_{metadata}	Size of metadata file
$M_{\mathrm{obs;MAX}}$	Maximum data size per observation
M_{obs}	Average data size per observation
M_{output}	Size of output data cube
M_{pulsar}	Size of pulsar timing data cube
M_{target}	Target grid size (per facet)
M_{vis}	Size of visibility data (bytes)
$M_{\mathrm{weight\_grid}}$	Size of weight grid (bytes)

$M^{\text{cont}}_{\text{buf,vis}}$	Size of the visibility buffer needed for continuum imaging
$M^{\text{spec}}_{\text{buf,vis}}$	Size of the visibility buffer needed for spectral line processing
$N_{f,\text{av}}$	Number of frequency channels that are averaged (in de-mixing)
$N_{t,\text{av}}$	Number of time slots that are averaged (in de-mixing)
$N_{\text{A,kern}}$	Number of distinct A kernels that must be modelled
$N_{\text{A,kernel}}$	Number of distinct A models
N_{a}	Number of antennas or stations in the array
N_{AA}	Number of grid points for A-kernel support (along one axis)
$N_{\text{Aproducts}}$	Number of $A^T A$ tabulations needed
N_{apv}	Number of accesses per visibility
N_{Ateam}	Number of A-team sources
N_{avg}	Number of averages
N_{beam}	Number of beams simultaneously observed by the array
$N_{\text{bin:smooth}}$	Number of bins included in smoothing window
N_{bin}	Number of bins
N_{bit}	Number of bits
N_{bl}	Number of baselines (i.e. cross-correlations)
N_{ac}	Number of auto-correlations
N_{bpa}	Number of bytes per access
N_{burst}	Number of single pulse candidates
$N_{\text{byte,pix}}$	Number of bytes per pixel
$N_{\text{byte,vis}}$	Memory per individual visibility
N_{byte}	Number of bytes
N_{cand}	Number of pulsar candidates per beam
N_{cma}	Number of flops in complex multiply and add
N_{cu}	Number of compute units
N_{cvff}	Number of grid points along one axis for the gridding convolution function
N_{datait}	Number of data items required per source (e.g. position)
N_{f}	Number of frequency channels
$N_{\text{f,corr}}$	Number of frequency channels output by the correlator
$N_{\text{f,degrid}}$	Number of frequency channels to be de-gridded
$N_{\text{f,distribute}}$	Minimum number of frequencies needed to maintain distributability
$N_{\text{f,grid}}$	Number of frequency channels to be gridded
$N_{\text{f,kernel}}$	Number of convolution kernels needed to cover frequency axis for a particular baseline
$N_{\text{f,out}}$	Number of output frequency channels (to be FFT-ed and reprojected or output from the CSP pulsar timing backend
$N_{\text{f,smear}}$	Number of frequency channels set by smearing limit
$N_{\text{f,smooth}}$	Number of channels included in smoothing window
N_{f}	Number of channels
N_{facet}	Number of facets along one axis
N_{flop}	Number of floating point operations
$N_{\text{flop/grid}}$	Number of floating point operations per visibility

$N_{\text{flop/kernel}}$	number of flops to generate a convolution kernel
$N_{\text{flop/samp}}$	Number of floating point operations per sample
$N_{\text{flop/solution}}^{\text{StEFCal}}$	flops per calibration solution using StEFCal
$N_{\text{flop/vis}}$	Number of floating point operations per visibility
N_{grid}	Number of pixels in the grid (i.e. the square of the linear dimension)
N_{GW}	Number of grid points for w-kernel support (along one axis)
N_{it}	Number of iterations
$N_{\text{major,total}}$	Number of selfcal-major CLEAN cycles
N_{major}	Number of major CLEAN cycles
N_{MAXbeam}	Maximum number of beams simultaneously observed by the array
N_{minor}	Number of minor CLEAN cycles
N_{mm}	Factor to account for gridding of off-diagonal Mueller terms
N_{ops}	Number of operations
N_{param}	Number of parameters
$N_{\text{patch,pix}}$	Number of pixels of PSF to be used in minor cycle decorrelation
$N_{\text{pix,facet}}$	Number of pixels along one axis of a single image facet
$N_{\text{pix,kernel}}$	Linear size of combined A and w-kernels (without oversampling)
N_{pix}	Number of pixels along one axis of a grid
N_{pol}	Number of polarisation products
N_{rma}	Number of flops in real multiply and add
N_{samp}	Number of samples
N_{SelfCal}	Number of self cal loops
N_{source}	Number of sources
$N_{\text{sourcefit}}$	Number of iterations in source fitting
N_{subbands}	Number of subbands used in imaging (within a subband, pixel size and position is matched across frequency)
N_{subint}	Number of sub-integrations
N_{sum}	Number of sums
N_{Tt}	Number of Taylor terms
N_{unique}	Number of unique candidates
N_{vis}	Number of visibilities
N_t	Number of time slots
Q_{bw}	Quality factor
Q_{fcv}	Factor to allow for the reuse of functions between neighbouring frequency channels
Q_{FoV}	Quality factor defining how much bigger the diameter of the field of view is than the first zero of the Airy function
Q_{GCF}	Gridding oversampling factor
Q_{kernel}	Convolution kernel quality factor, for kernel re-use in frequency
Q_{MSMF}	Factor to account for multiple subtractions (multi-frequency multi-scale CLEAN)
Q_{pix}	Quality factor defining the oversampling of the synthesised beam
Q_t	Quality factor
Q_w	Correction factor for w values (0 to 1)
$\mathbb{R}_{\text{filt}}$	Filtered covariance matrix

$\mathbb{R}$	Covariance matrix
R_{IO}	IO rate
$R_{\mathrm{bw,I/O,max}}$	Maximum I/O bandwidth of compute unit to buffer
$R_{\mathrm{bw,I/O}}$	Bandwidth to buffer
$R_{\mathrm{bw,max}}$	Maximum memory bandwidth of compute unit
$R_{\mathrm{bw,mem}}$	Bandwidth to working memory
R_{Earth}	Radius of the Earth
r_{facet}	Facet overlap overhead
$R_{\mathrm{vis,corr}}$	Correlator output rate (visibilities/sec) instead
$R_{\mathrm{vis,facet\,FoV}}$	Visibility rate (per facet) for averaging scheme avoid smearing at facet field of view
$R_{\mathrm{vis,full\,FoV}}$	Visibility rate for averaging scheme avoid smearing at full field of view
R_{vis}	Number of visibilities per second
$R^{\mathrm{cont}}_{\mathrm{bw,I/O}}$	Bandwidth to buffer for continuum imaging
$R^{\mathrm{spec}}_{\mathrm{bw,I/O}}$	Bandwidth to buffer for spectral line processing
S	Telescope sensitivity
t_{comp}	Computation time
t_{obs}	Observation time
t_{res}	Time resolution (sec)
t_{solve}	Timescale for gain solutions
t_{dump}	Correlator dump time (sec)
$t_{\mathrm{kernel,update}}$	Update timescale for gridding kernel
t_{LSM}	Update timescale for the LSM
t_{smear}	Smearing Time resolution (sec) (function of baseline length)
t_{snap}	Snapshot duration
$t_{\mathrm{snap,min}}$	Minimum duration of a snapshot in which w-correction is not dominated by Earth's curvature
$t_{\mathrm{snap,opt}}$	Optimum snapshot duration (sec)
t_{subint}	Subintegration time
u, v, w	Visibility coordinates
uv	Visibility coordinates
$\mathbb{W}_{\mathrm{filt}}$	Spatial filter matrix
w	Visibility w coordinate
Δw_{max}	Maximum deviation of the w coordinate due to the Earth's curvature
Δw_{min}	Minimum deviation of the w coordinate due to the Earth's curvature

List of Figures

List of Tables