Authors: Kjeld v.d. Schaaf Ronald Nijboer	Date of issue: 2007-03-28 Kind of issue: public	Scope: LOFAR calibration Doc.id: LOFAR-ASTRON-SDD-050
	Status: final Revision nr: 2.0	

LOFAR Calibration Implementation
Global Design of the Major Cycle

Verified:			
Name	Signature	Date	Rev.nr.
Marcel Loose Joris van Zwieten John Romein		2007-03-28 2007-03-28 2007-03-28	2.0 2.0 2.0

Accepted:		
Work Package Manager	System Engineering Manager	Program Manager
Ronald Nijboer	Andre Gunst	Jan Reitsma

Distribution list:

Group:
Review committee E. van den Heuvel (chair, University of Amsterdam) A. Berg (SARA) W. Brouw (University of Groningen) R. Schilizzi (SKA-ISPO/U. Leiden) C.H. Slump (University of Twente) A.B. Smolders (NXP Semiconductors) E. Stolp LOFAR Project Mark Bentum Ger de Bruyn André Gunst Michiel van Haarlem Hanno Holties Auke Latour Marcel Loose Ronald Nijboer Jan Noordam Ruud Overeem Jan Reitsma John Romein Michael Wise Joris van Zwieten For information Marco de Vos

Document history:

Revision	Date	Section	Page(s)	Modification
0.1	2006-08-16		-	Outline
0.2 – 0.6	2006-08-29 – 2006-10-12		All	Wrote all text; review by calibration “stuurgroep” and development team
1.0	2006-10-18			Release for CAL-Review (6-8 November 2006)
2.0	2007-03-28		All	Update for Systems CDR (April 2007)

Table of contents:

1 Introduction

This document describes the translation of the calibration strategy described in [1] into a software system. Seen from the software engineering perspective, this document contains the architecture description of the LOFAR calibration software package.

The scope of the current version of this document is the calibration implementation for initial operation of LOFAR. We believe that during commissioning of LOFAR and even during the operational life of LOFAR the calibration strategy will evolve and mature. This implies that the Selfcal implementation will have to change and evolve with any newly obtained insights. The architecture described here is intended to be open to such changes.

Currently we are working on an implementation for the Standard Imaging Mode (see section 4.1 of [2] and Figure 1). This implementation is directly relevant to the Surveys, EoR, and Transients KSP pipelines (section 4 of [2]), which can all be derived from the Standard Imaging Pipeline. Moreover, specific parts of the implementation (e.g. ionospheric calibration) are relevant to all the other KSP pipelines. In time the implementation will evolve into a full implementation of the Major Cycle as described in [1].

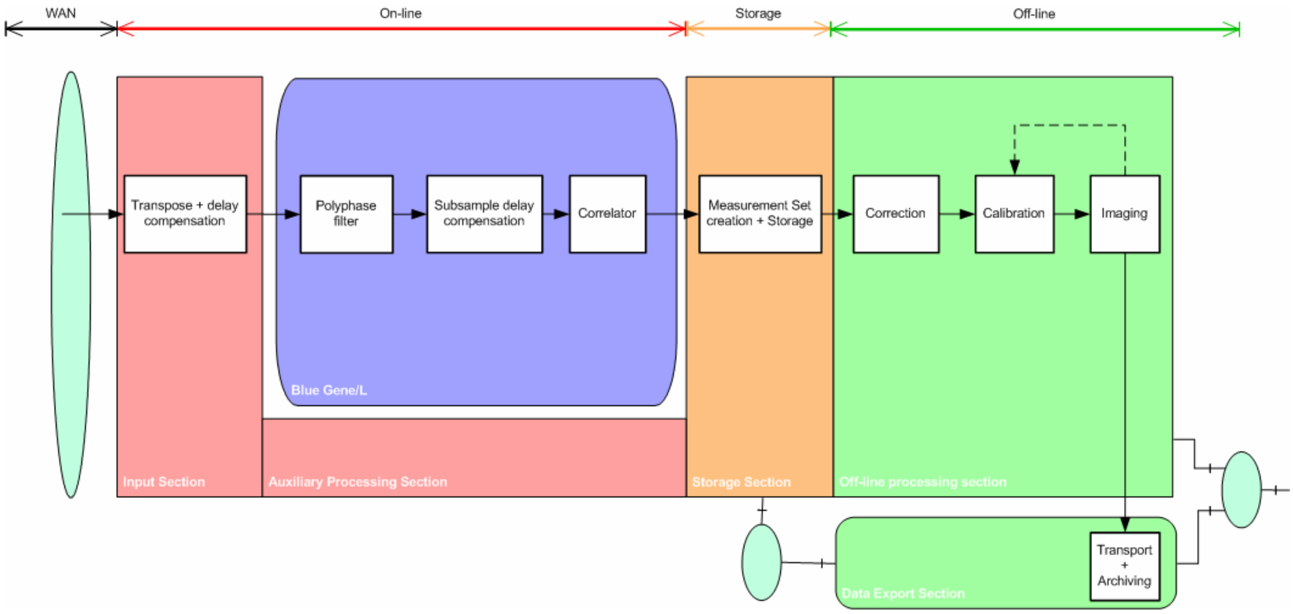


Figure 1 *Processing scheme for the Standard Imaging Mode (from [2])*

Section 2 of this document revisits the Measurement Equation and the modelling of the instrument, the ionosphere and the sky. In section 3 we describe what the self-calibration processing pipeline (i.e. the Major Cycle) looks like and which software components are needed. Section 4 describes the top-level software architecture and in section 5 some performance aspects are discussed. Finally, in section 6 we briefly describe the current status of the implementation. The planning is described in a separate document [3] as part of the full software planning.

2 The Measurement Equation

The Measurement Equation (ME) [4][5] is the core of the LOFAR Selfcal process. The whole calibration process is based on “tuning” the parameters of models underlying the ME in order to minimise the difference between the model-based predicted data, $\overset{\omega}{V}_{ij}$, and the observed data. The general LOFAR ME is given in [1] as:

$$\overset{\omega}{V}_{ij}(t, f) = (J_i \otimes J_j^*) \sum_k (J_{ik} \otimes J_{jk}^*) \cdot S \cdot \overset{\omega}{I}_k$$

The 2 x 2 Jones matrices (J) describe the instrument and the environment (i.e. the ionosphere) and they depend on station (i) and position in the sky (or source direction k). The 4 vector I describes the source in its Stokes parameters. t denotes time, f denotes frequency, and S denotes the transformation between Stokes parameters and correlation components (XX, XY, YX, YY for linear polarization).

In order to limit the number of parameters (by allowing for larger data domains) the Jones matrices can be split into more physical parts [1]:

$$J_i = B_i G_i$$
$$J_{ik} = E_{ik} P_{ik} I_{ik} F_{ik} K_{ik}$$

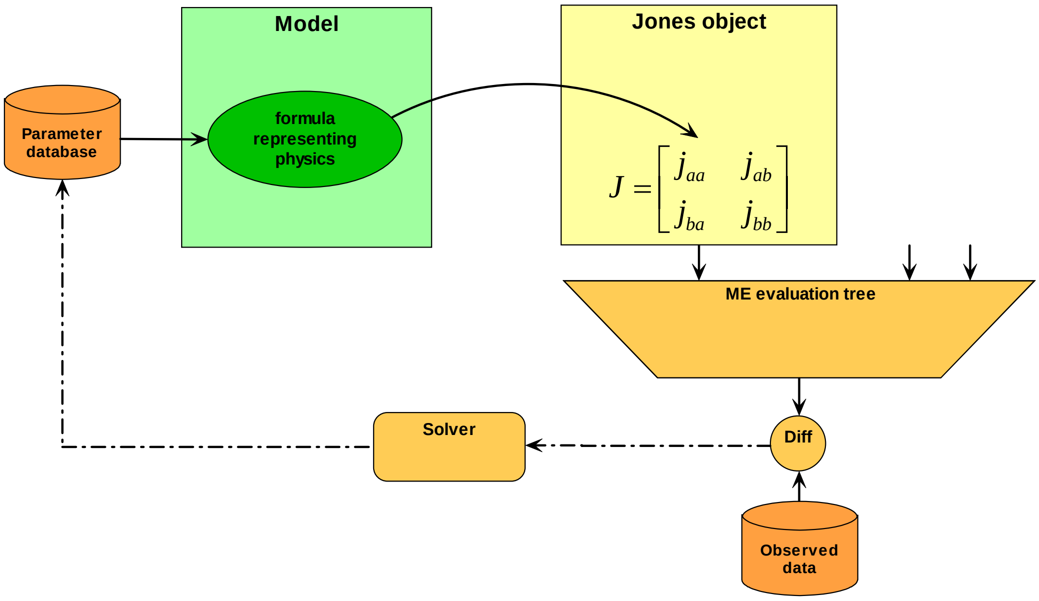

Model
Jones object
Parameter database
formula representing physics
$J = \begin{bmatrix} j_{aa} & j_{ab} \\ j_{ba} & j_{bb} \end{bmatrix}$
ME evaluation tree
Solver
Diff
Observed data

Figure 2 *Role of Jones matrix and parameterised models in the solve process. Note that we solve for the model parameters and not for the elements in the Jones matrix directly.*

The relation between the Jones matrices, models and parameters is illustrated in Figure 2. The Jones matrix elements are needed to evaluate the ME for a given interferometer ij, time-frequency point (t, f) and source direction k . The values i, j, t, f and k identify the data point for which the ME contribution is evaluated. The Jones matrix elements for a given data point are calculated based on a model of a physical phenomenon at hand. For example, the I-Jones matrix will be calculated based on a model of the ionosphere. Such a model consists of a formula, may include a look-up table or may be a complete simulation on its own. In all cases the model will contain parameters that tune the physics behind the model or tune empirical formulas. Those are the parameters we solve for in the Selfcal process.

Note that the model may also be absent, in which case the model parameters correspond directly to the Jones matrix elements. Splitting of the Jones matrix and the actual model allows us to integrate extensive (external) simulations and solve for the parameters of those simulations. This feature can be used in the bandpass modelling leading to the B-Jones where we combine passbands from different measurements and may solve for the relative weights of those contributions.

In section 2.1 we discuss the instrument models used for the B- and E-Jones matrices. The environment related models needed for the I and F-Jones matrices are discussed in section 2.2. The source contributions and sky model catalogues are discussed in section 2.3. Note that the different models will change with the evolution of LOFAR, through commissioning and early operations. Therefore, the model descriptions merely reflect our current understanding of the instrument, the environment, and the sky.

2.1 Description of the instrument models

This section describes the instrument models for the RF bandpass and beamshape of the LOFAR stations. These models are associated with the B and E-Jones matrices. The bandpass is a UV-only effect, i.e. it does not dependent on the pointing direction. The beamshape depends on the pointing direction and frequency.

2.1.1 Bandpass (B-Jones)

The bandpass model describes the frequency dependence of the analogue electronics in the station signal chains. The model describes the average behaviour of all signal chains in a station voltage beam, resulting in the B and G-Jones matrices
The physics driving the bandpass model are:

- The amplifier in the antenna. The frequency behaviour of this amplifier is measured in the lab.
- Coax cable from antenna to receiver board. Both the phase delay and attenuation as function of frequency are known from (lab) measurements
- Analogue filters on the receiver board. Depending on the receiver board settings the signal is filtered by low pass, high pass and bandpass filters. The phase and gain behaviour of these filters are measured in the lab.
- Digital filter passbands. These are known with 100% accuracy
- RF Cross-talk between channels may occur but is not modelled.

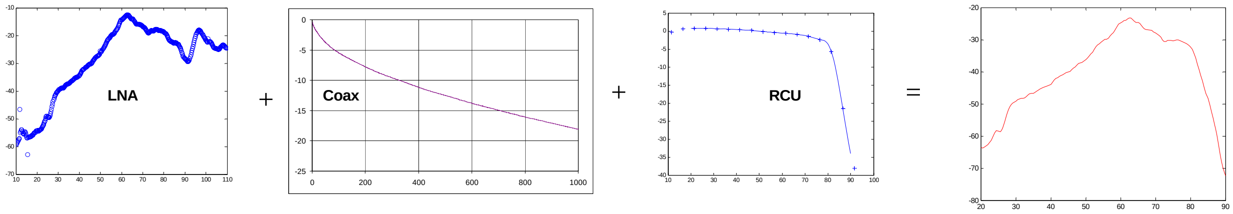


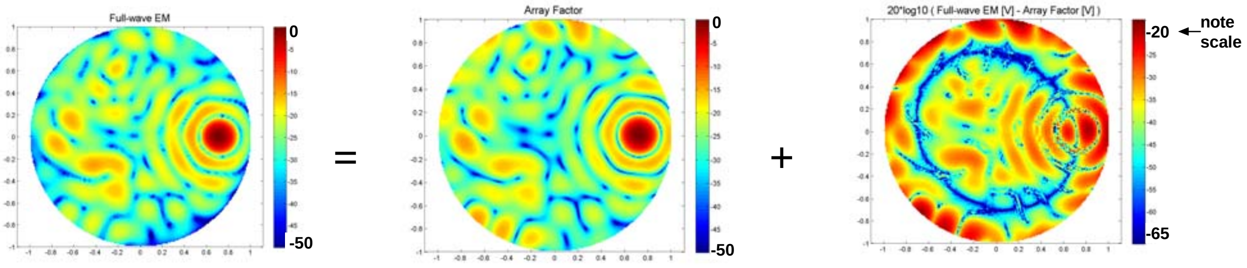
Figure 3 *The total station bandpass is built from individual lab measurements of the LNA, coax and RCU passbands. The total contribution is the product of these individual effects.*

The total bandpass is a combination of the underlying phenomena, as shown in Figure 3. The total model is built as a product of the three individual measured passbands, which are available in look-up tables. The passbands shown in the figure are the amplitude behaviours only; the phase response can be measured in the same way and added.

The parameters that go into the bandpass model should describe the time behaviour of the frequency response as well as the instability and variations in the components. The time behaviour will mainly be temperature effects. The temperature dependence of the various contributions can be measured as well, but rather should be modelled with simple functions. The model parameterisation should allow for time dependent changes as well as static variations in the individual contributions. A possible approach is to sample the temperature of the LNA location, coax cable and RCU boards and use these values as additional dimension in the model. Note however that this also introduces additional degrees of freedom since the observed temperature must be related to the laboratory measurement temperature tables.

We have not yet worked out a complete model. For the time being a simplified model is used with the combined amplitude characteristic of the complete IF chain available as a look-up table multiplied by a correction function in time (2nd degree polynomial).

2.1.2 Beamshape (E-Jones)



Station Beam

Element pattern & array effects

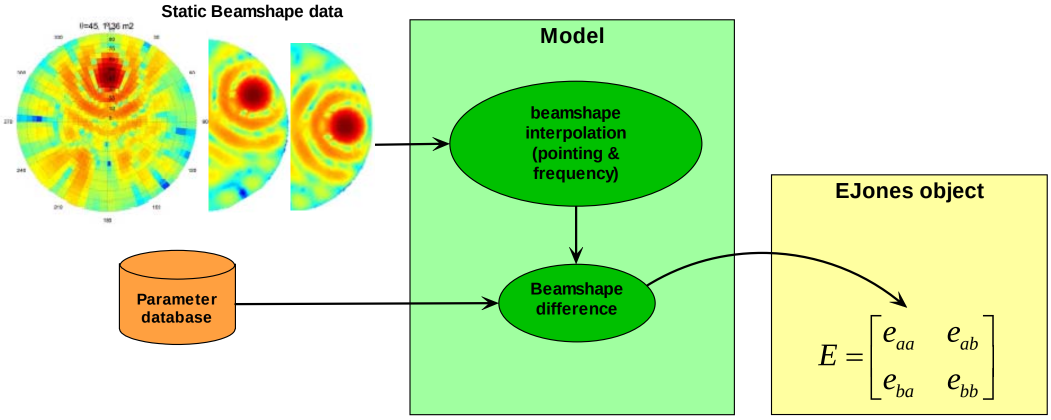
"Residual Beam"

Figure 4 *A full EM simulation of the station beam is built from an array factor and a "residual beam". The array factor contains all geometrical effects of the station beam along with the average element pattern. The colour scales are in dB and the residual beam scale is ~25 dB below the other two.*

The physics behind the total beamshape is very complex; involved are the geometrical effect, mutual coupling and other effects. Simulation efforts performed by the antenna group of Astron led to accurate results that are partly validated by direct measurements. At the same time these simulations show how complex and computationally intensive the calculations are. Therefore it does not seem feasible to base the beamshape estimate for each pointing[1] and frequency channel on a full EM simulation.
However, a very large part of the station beam is based on the so-called *array factor*, as is shown in Figure 4. Since the array factor is relatively cheap to calculate, we can give a rather accurate beamshape estimate in a real-time or we can interpolate (or actually rotate) between calculated array factors with pointings per minute or so. These pre-calculated array factors form the static part of the beamshape model, as illustrated in Figure 5. A similar approach will be followed with interpolation in the frequency dimension.
The array factor includes an element pattern, which may vary between observations, for example due to different reflection behaviour of the soil. We can use this variation by solving for the weight factors of a few variations in element pattern parameters.

[1] Note that the (digital) pointing direction changes every second while tracking a source.

We may also estimate the strongest contributions in the “residual beam” to increase the accuracy of the beamshape model. However, the residual beam shown in Figure 4 clearly demonstrates that it will be parameter intensive to model the residual beam accurately, meaning that we will need a lot of data to get enough information to solve for so many parameters. In practice this means that it will only be possible to solve for the residual beam pattern on a coarse time, frequency grid and interpolation will have to be used to get estimates for intermediate values.



Static Beamshape data
Model
beamshape interpolation (pointing & frequency)
Parameter database
Beamshape difference
EJones object
$E = \begin{bmatrix} e_{aa} & e_{ab} \\ e_{ba} & e_{bb} \end{bmatrix}$

Figure 5 *Overview of the Beamshape model associated to the E-Jones matrix. The beamshape model is based on simulated beamshapes for a number of pointings and frequencies that are used as static (look-up) data.*

2.2 Description of the environment related models

The I and F-Jones matrices describe the influence of the ionosphere on the observed visibilities. The physics covered in this model is the phase variation and Faraday rotation of the observed electro magnetic waves travelling through the different layers in the ionosphere. As described in the calibration framework document [1] we will use a so-called Minimum Ionospheric Model (MIM) which is an empirical all-sky continuous description of the observed phase rotation caused by the ionosphere. The implementation of the MIM will be relatively simple, using e.g. spatial piecewise polynomials or periodic functions to model the effect of the global electron density and Travelling Ionospheric Disturbances (TIDs). The model will be global in the sense of not being station specific. Instead, the geometry of the station layout is used to model the effect of a global electron density and TIDs over multiple stations, especially over the core stations. The outer remote stations are far apart and additional pointings can be used to model the region in between. Although the description of the global electron density and TID may not be single continuous functions over the entire array, the MIM will be a single global model used to calculate the I-Jones matrices for all stations. The implementation of the MIM is illustrated in Figure 6.

The empirical MIM will be constrained to specific physical effects, in particular the frequency dependence of the phase rotation and some continuity constraints on the travelling waves. Tomography with multiple beam directions can be used to improve the accuracy of the model.

Apart from the parameterised MIM functions, we need to solve for the relative phases between the stations and apply phase locking to keep the solutions in phase. This part of the ionospheric model will be updated separately from the iterative parameter solving in the calibration.

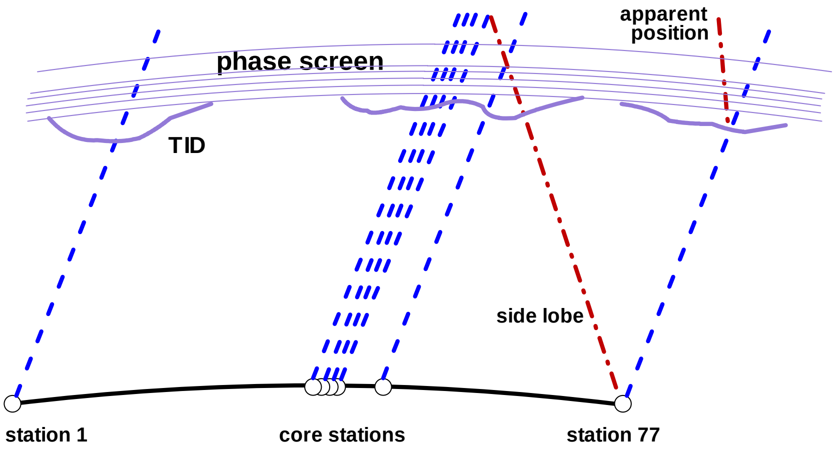


Figure 6 *Illustration of the MIM model for the ionospheric phase rotation. The MIM models the observed phase screen associated with the ionosphere. The model contains a parameterised total electron content (TEC) layer which is the basic phase screen through which we observe. Tomography may be used to completely map out the 3D distribution within this layer. Added to a global electron density are the disturbances which travel over the array (TID). Note that the ionosphere also alters the apparent position of a source by refracting the waves.*

2.3 Sky Model and catalogues

The calibration framework as described in [1] distinguishes between different types of sky sources (Cat-I-III). Cat I and Cat II sources are known, e.g. from previous measurements. Cat I are the strongest sources in the Field of View (FoV) and they are used to determine the instrumental and environmental models during the self-calibration. Cat II sources are weaker and typically subtracted from the data, in order to get a deeper map. Cat III sources are the sources that are in the data, but that we have not found yet.

The Local Sky Model (LSM) contains all known sources relevant for the data reduction of an observation. Hence, the LSM contains Cat I and Cat II sources plus sources that may enter through the station sidelobes (the so-called A-Team: CasA, CygA, VirA, TauA …) When Cat III sources are found and put in the LSM, they turn into Cat II sources (by definition).

The LSM only contains our knowledge of a part of the sky. As such it can be seen as a subset of the Global Sky model, which contains all knowledge of the sky brightness distribution we have. For each observation, the LSM sources must be selected from the GSM. Dedicated tooling will be developed to perform the selection process and visualise the resulting catalogues.

2.4 Dependency between model parameters

The Selfcal implementation cannot assume that all parameters in the instrument and environment models are completely independent. Hence we have to address such dependencies in the (solve) strategy.

Dependency between parameters can have two main causes:

- It is not possible to distinguish between physical phenomena based on the available data
- Physical phenomena are not modelled accurately enough

Based on the model implementations described in section 2.1 we can expect dependencies between the following parameters/models:

- Frequency dependency of IF chain versus beamshape
- Phase rotation of IF chain versus ionosphere, especially for the outermost stations where we do not have multiple observations through the same ionospheric patch

We can detect dependencies between parameters in an indirect way during the execution of the calibration strategy. In order to do this we introduce alternative paths in the strategy and analyse the resulting parameters and convergence of parameters afterwards. The possibility to do this was one of the main drivers to apply the BlackBoard architectural pattern (see [6]). There are also methods to study the dependencies between parameters in a more off-line analysis approach based on the history database of parameter values. This issue of dependencies between parameters is strongly related to local minima in minimisation algorithms, see also section 5.5.

3 Calibration

The calibration strategy is described in detail in [1]. We give a description here that is focussed on the implementation in an operational software system. This introduces new viewpoints like control, data management and timing aspects.

3.1 Functional description

The challenges for LOFAR calibration as described in [1] are summarized as follows:

- the pathological ionosphere,
- the crowded fields
- the variability of the station beams (in time and frequency as well as over stations),
- the high station beam sidelobes,
- the large data volumes, and
- the high accuracy needed.

In order to deal with these challenges LOFAR calibration consists of several stages. First of all, part of the calibration of UV-plane effects can be performed on-line, i.e. in the central processor based on all station inputs, but prior to the correlation. This allows for calibration on timescales that are shorter than the correlator dump time. Although the design allows for this kind of calibration, it is not expected that it will be available in the early years of LOFAR operations.

Off-line calibration starts with an external calibration (correction for known errors, like differences in cable lengths, and correction for the bandpass) and flagging of the worst RFI. After external calibration uv-plane effects are calibrated, including electronic gains and fitting of parameters for the ionospheric phase. Finally, calibration of image-plane effects takes the form of a Major Cycle using both visibility data and image data:

1. Solve for Instrumental parameters using the visibility data and Cat-I sources.
2. Subtract the Cat. I sources and all strong (A-team) sources anywhere in the sky as accurately as possible from the UV-data
3. Subtract all other (known, Cat. II) sources from the UV-data using interpolated values for the instrumental and ionospheric predictions
4. Image the residual UV-data.
5. Find residual values for the Cat. I & II sources and extract additional unknown sources (Cat. III, which will then be upgraded into Cat. II).
6. Go to 1.

The whole LOFAR calibration chain is described in Figure 7 to Figure 9. In the next section all components in the chain are described.

3.1.1 On-line calibration

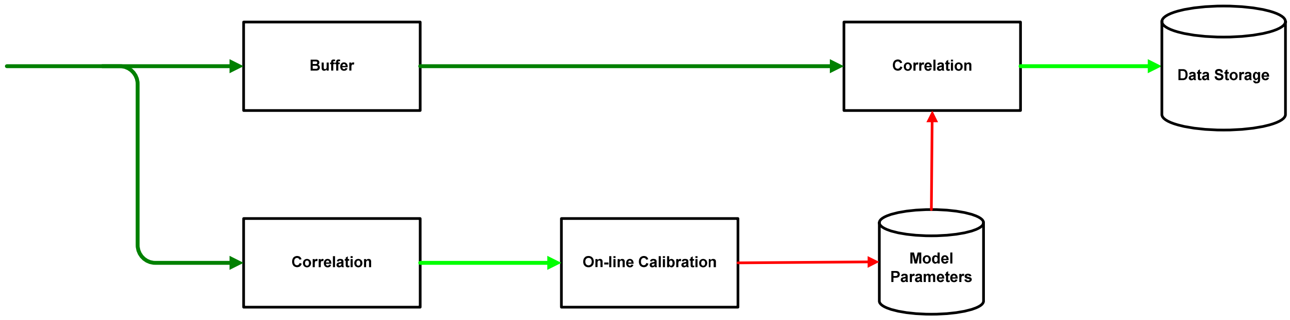


Figure 7 *Flow chart of the on-line calibration. The on-line calibration process operates on a sampled subset of the total data stream (lower part of figure). The model parameters calculated in the on-line calibration are used as correction factors in the correlation of the main data stream. This main data stream is delayed by using a buffer.*

On-line calibration gives one the possibility to correct the visibility data on time-scales shorter than the correlator dump time. This is only possible for effects that are station dependent and that are constant over the sky. On-line correlation is implemented by copying the data stream before correlation. One stream is then buffered, while the other is correlated and a coarse calibration is performed. The outcome of this calibration is then used to correct the buffered data stream while that stream is correlated. This is shown schematically in Figure 7. After correlation of the main stream, the resulting visibility data is stored.

The correlator software is described in more detail in [7]; here we just give a summary. The correlation block consists of the following steps:

- Delay Tracking / Fringe Control,
- Polyphase Filter (195 / 156 kHz subbands transform in 256 0.76 / 0.61 kHz channels)[2],
- application of phase correction (final step fringe control),
- multiplication,
- and integration (768 or 608 time samples are integrated into one 1 s sample).

As such the correlator block takes two station streams on a 5 or 6.25 microsecond basis to produce 1 baseline stream on a 1 second basis (and this for all station combinations including auto-correlations). The incoming data is for 77 stations and 156 / 206 subbands. The final visibility data consists of 40000 0.8 kHz or 52000 0.6 kHz frequency channels, per 1 s per 3003 baselines (cross-correlations and auto-correlations). This data is then stored for post-processing.

In the integration step the Station Flags are transformed into a sample weight. The correlator block produces weights and flags as meta-data.

Before correlation the data can be buffered for up to 30 seconds. By copying the data in a second stream and correlating it, a first, coarse calibration of complex gains can be performed that is then applied before correlation.

[2] Note that the station system can operate with two sample clock settings, leading to two different frequency resolutions. The first value given is for the 200 MHz clock, the second for 160 MHz; see [7] for more details.

3.1.2 Off-line UV-plane calibration

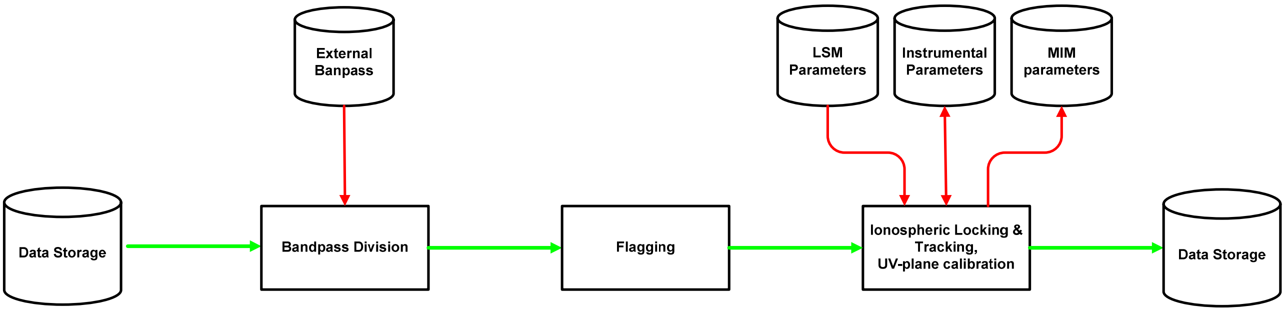


Figure 8 *Flow diagram of the data preparation processes. This flow starts with the UV data from the correlator which is stored on disk. This data enters a processing pipeline starting with bandpass division, then flagging and finally an ionosphere tracking and locking process and correction for electronic gains.*

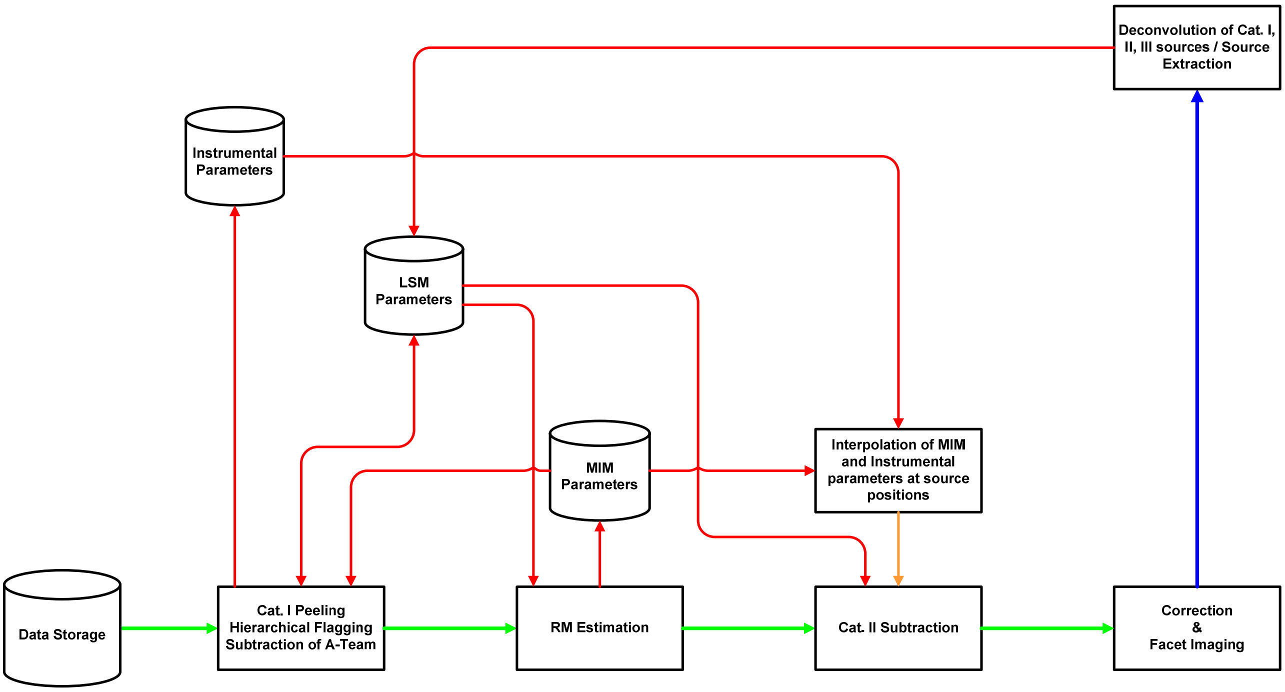
Before the actual Self-Calibration will run, the visibility data is pre-corrected. First the data is corrected for the bandpass and other pre-known off-sets. The bandpass is a station based frequency profile and it is externally given. The bandpass is determined either by measurements or by simulations.

After the bandpass correction, the data is flagged, identifying (the strongest) RFI.

Finally, a first order ionospheric model is determined from the data. This is done by first bringing all station phases in lock. The procedure is to start with three stations and include others one by one. When all stations are in lock, we solve for the time dependent phase screen in the ionosphere model (MIM). At the same time the data is corrected for the electronic gains (G-Jones).

After these preparation steps (which are performed once) the data is stored again.

3.1.3 Calibration of Image Plane effects: the Major Cycle



Deconvolution of Cat. I, II, III sources / Source Extraction
Instrumental Parameters
LSM Parameters
MIM Parameters
Interpolation of MIM and Instrumental parameters at source positions
Data Storage
Cat. I Peeling Hierarchical Flagging Subtraction of A-Team
RM Estimation
Cat. II Subtraction
Correction & Facet Imaging

Figure 9 *Flow diagram of the Major Cycle*

After preparation of the data the actual Self-Calibration can start. Self-Calibration is implemented as a number of Major Cycles through the visibility domain and through the image domain. Each cycle starts with the original, pre-corrected visibility data and in each cycle the instrumental, environmental, and sky model parameters are improved.

The steps of the Major Cycle are:

1. Use the brightest sources (Cat. I) to fit Instrumental model parameters. Subtract the Cat. I sources and all strong sources that enter through the sidelobes (the A-team). This step consists of several sub-steps and iterations between the Cat-I and A-team sources. In between these sub-steps additional flagging of RFI may be needed. (Note that the flags that are set are kept in next cycles.)
2. Subtract all remaining known sources (Cat. II) using the instrumental and environmental errors.
3. Correct the visibility data per facet. Make a residual image per facet.
4. In the (facet) images find residual values for the Cat. I & II sources and extract additional unknown sources (Cat. III, which will then be added to the Sky model and upgraded to Cat. II).

Note that all steps in the Major Cycle will produce meta-data. This is not shown explicitly in Figure 9.

After all calibration has taken place, residual visibilities, and all model parameters will be stored[3]. Residual visibility data may be stored for short periods. This is not indicated explicitly.

[3] Actually the model parameters can also be stored continuously during the calibration

3.2 Components of the Major Cycle

This section describes the software components that together provide the functionality described in section 3.1. In software engineering terms this corresponds to a first architectural view, the functional description can be seen as the requirements set. The process of converting those requirements into a component model is not further described here; rather we present the end-result only in Figure 10.

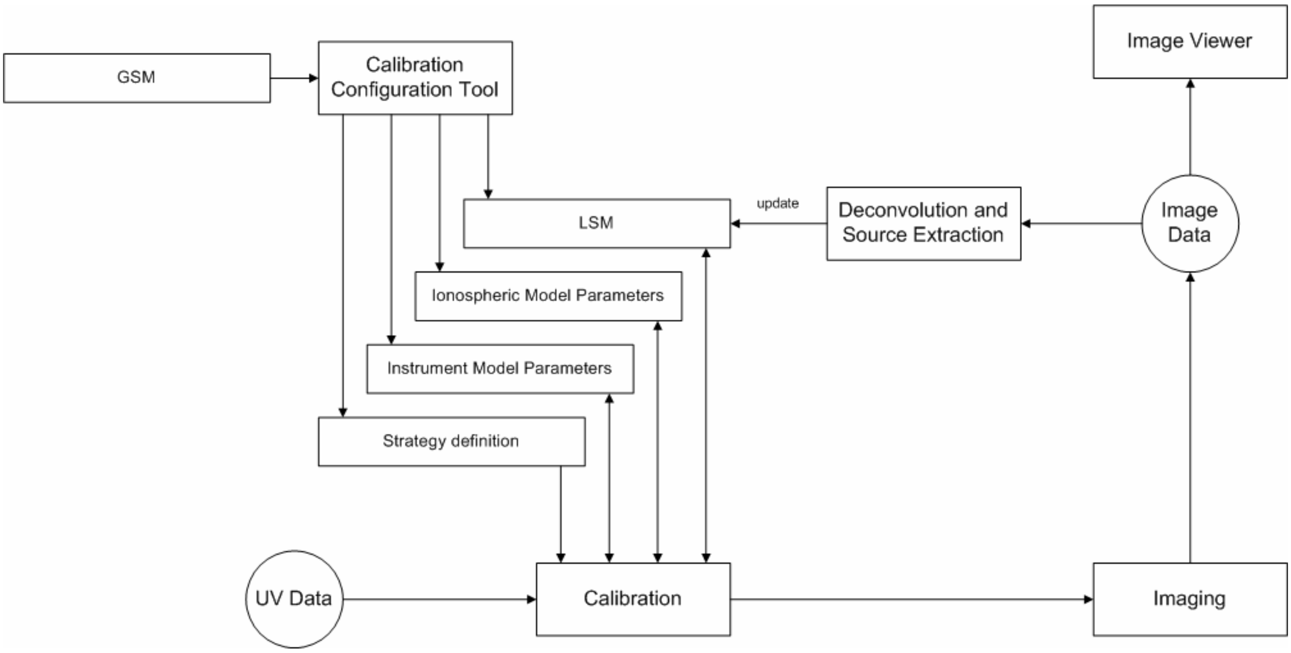


Image Viewer
GSM
Calibration Configuration Tool
update
Deconvolution and Source Extraction
Image Data
LSM
Ionospheric Model Parameters
Instrument Model Parameters
Strategy definition
UV Data
Calibration
Imaging

Figure 10 *Software components in the Major Cycle. The arrows indicate the flow of data and parameters in the system. The user can inspect the performance and progress of the major cycle execution while it is executed in an automated way.*

Configuration tool
The configuration tool is used to completely specify a calibration run and to configure the details of that run. The configuration tool is part of the Scheduling and Specification (SAS) software, which is described in [8]. Specification consists of the selection of pre-defined calibration scenario templates. Such a template defines the exact execution order of the steps described in section 3.1 such as the number of Major Cycle loops or a stop criterion. One level deeper the template describes for example the parameter solving order in the inner loop. After selection of, and possibly modifications of, the template we have to configure the calibration run by setting facet sizes, convergence criteria etc. The template will contain default values that can be modified at will.

The whole calibration process is supposed to run in a completely automated mode based on those parameter files. However, it is possible to specify interaction points in the Major Cycle flow. These interaction points may be compared to breakpoints in a debugger. At the interaction point the user may inspect the results of the calibration so far and modify the specification or configuration for the remainder of the calibration run. This results in updated parameter files and corresponding updated procedures in each component.

Sky Models (GSM & LSM)

Part of the specification and configuration process is the creation of a Local Sky Model (LSM) based on all sky knowledge available in the Global Sky Model (GSM). The result of the selection process is a set of Cat-I, Cat-II, and A-team sources (or source models).

The selection tool selects sources from the GSM based on queries which can be (combinations of) spatial criteria, source strength criteria, polarisation criteria etc. This selection can be performed “by hand” using visual tools as well as completely automated.

Parameter tables

The specification and configuration process, including the creation of the LSM catalogues, results in a set of parameter files that together define the complete calibration run. These files are sent via the monitoring and control (MAC), [9], and application configuration and control (ACC), [2], subsystems to the individual processes at execution time (see [10] for more details).

UV Data File

The observed visibility data is stored in the UV data file component. Since the calibration component is implemented as a distributed system, the UV data is also distributed over potentially many storage nodes. Currently uv-data files are stored in the AIPS++ Measurement Set format. The possibility to implement this format based on a HDF5 data format is being considered.

Calibration

The calibration component performs the actual solving for model parameters based on the full measurement equation and visibilities in the observed UV data file. Along with the model parameters, the calibration receives a completely configured “recipe” of the solve process in terms of Strategies and Steps, see section 4.2.1.

The calibration component implements the fitting process of the observed data to the model functions with the model parameters as free parameters. Due to the iterative nature and the need for a distributed and parallelised implementation the calibration component is actually a major effort to build.

Imaging

The imaging component performs the transformation from UV plane to image plane. Images are created per facet after the (residual) visibility data is corrected for the centre of the facet. The visibility data is potentially distributed over many storage nodes and this will also be true for the images that are produced. The imaging component will not be built from scratch, but rather based on existing AIPS++ libraries. Distribution will be implemented within the same framework as the calibration component.

Image file

The image file will be stored in the AIPS++ image format. The possibility to implement this format based on a HDF5 data format is being considered. The image files will be distributed over multiple nodes, similar to the UV data files. An export facility in FITS image format will be provided too.

Deconvolution and Source Extraction

Residual images that are produced in the Major Cycle need to be deconvolved to obtain updates of source parameters. Deconvolution will be performed with part of the Point Spread Function (e.g. only the main lobe). The PSF can be modelled such that the variation of the PSF over the image can be taken into account. After deconvolution the apparent source contributions are corrected for station beam attenuation to find (approximated) intrinsic source contributions. These intrinsic source contributions are used to update the (Local) Sky Model.

Note that the deconvolution and source extraction component may depend on the imaging component, since a combined implementation of those components allows for optimised sharing of data etc.

Image Viewer
Images should be displayable. Since images can be distributed over many storage nodes, the Image Viewer should be able to handle this. LOFAR KSPs may need the Image Viewer for analysing images. Hence, additional features may be needed or shared with the imaging and deconvolution components.

4 Software architecture

4.1 Implementing the control of the Major Cycle

The Major Cycle is a co-operation between multiple applications running on distributed computers. The control system has to take care of the appropriate sequence of execution in terms of sequential operations and possible parallel execution. This functionality is not too different from other applications running in LOFAR, see [8] for a description of the LOFAR control structure. The LOFAR MAC system is designed to control distributed applications, where the *Application Configuration and Control* (ACC) subsystem plays an important role. The fact that the Major Cycle contains multiple applications is reflected in an additional controller level in the MAC observation control tree. This controller level is also used to control processing pipelines. To control the Major Cycle, a specialised controller is created that has knowledge of the required dynamical behaviour of the applications constituting the major cycle, see Figure 11.
The Major Cycle controller also guides the construction of direct communication channels between the components and sharing of files or data streams between the components.

The configuration of the Major Cycle execution is similar to the configuration of any processing application or pipeline in LOFAR: in the System Administration and Scheduling (SAS) configuration GUI a dedicated set of panels will be available to set the key settings of the Major Cycle, such as number of global iterations and the requested source detection algorithms. The remainder of the settings, such as file names, file distribution, et cetera are generated automatically and may be overruled in expert mode.

As described in section 3.2 the control structure can be operated interactively in the sense that the user or operator can adapt the settings for the remainder of the calibration process during the run.

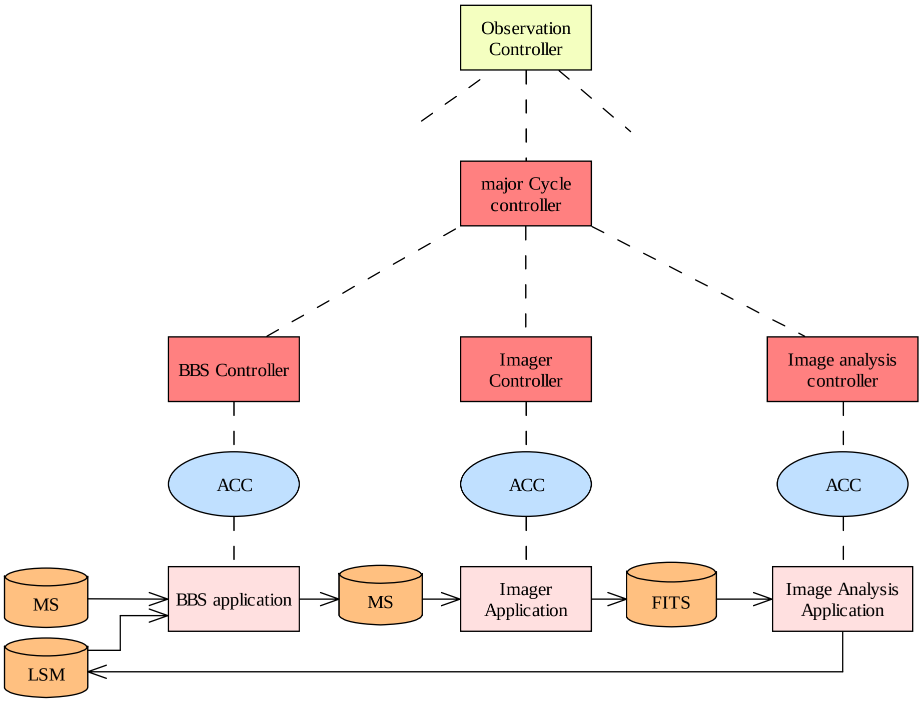


Figure 11 *Control of the Major Cycle in the MAC system. The MAC system contains a hierarchy of controllers to control observations. Part of such a hierarchy is the major cycle controller with leaf nodes to control the various applications constituting the major cycle.*

4.2 Implementing the Calibration component

The calibration component of the Major Cycle is implemented in the BlackBoard Selfcal (BBS) application. This section describes the architecture of the calibration component. More details on the design can be found in [11].

The architecture of the calibration component is mainly determined by the following key requirements:

- The implementation must be flexible to changes in the calibration algorithm, in particular to the model implementation and strategy recipe.
- There must be interfaces and tooling to inspect the performance of the calibration algorithm
- The implementation must be highly optimised in order to run on an affordable[4] computer platform
- Given the intended target platform, the calibration component must be able to run on a large distributed system and use parallel execution.

With these requirements in mind we have analysed possible architectures for the implementation. Given the need for distribution and flexibility, an architecture with a central communication and control entity was soon selected. In particular the BlackBoard architectural pattern (see [12]) got our attention (see [6]). A rapid proto type of a Selfcal-like application based on a single controller with blackboard and multiple workers was developed and analysed. This prototype proved the feasibility of a distributed Selfcal application with hundreds of nodes based on the BlackBoard pattern.

In the description of the BlackBoard Selfcal architecture presented in this section we start with a description of the control granularity and distribution. Next, we describe the role of the blackboard for control and distribution.

4.2.1 Strategies, Steps, and Multi-Steps

The software engineering concepts of Strategies and Steps are used to completely define what the calibration component has to do. At the lowest level, we have the Step concept. A Step is the smallest control entity in the calibration component.

The following Step types exist:

- Predict Step: predict visibilities using the Measurement Equation;
- Solve Step: predict visibilities using the Measurement Equation, compare with the observed data and solve for parameters in the Measurement Equation models. This sequence may be executed iteratively within the same Step definition;
- Subtract Step: predict visibilities and subtract those from the observed data;
- Correct Step: correct visibilities for a given direction in the sky;
- Others like: Shift Step, Refit Step, and Flagger Step.

A calibration run consists of a sequence of those Steps called Multi-Steps. By grouping Steps together, or even grouping Multi-Steps, we can define our own “language” to define a particular calibration Strategy. For example, we could define a peel Multi-Step containing Solve, Shift, and Subtract Steps.

The Strategy concept is used to specify the subset of the data to operate on and the time-frequency domains to use in the underlying (Multi-) Steps.

The execution flow of the calibration application is illustrated in Figure 12. The flow starts with configuring the Strategy definition in a graphical interface in terms of (Multi-) Steps. This definition results in a parameter set

[4] The baseline target platform is a 500 node Linux cluster machine with dual processor nodes and possibly a high bandwidth interconnect system

file which is the input for the calibration controller. The controller interprets the strategy definition and issues commands (esp. Steps) to the worker nodes. On the worker node the command (Step) is interpreted and the appropriate local procedures are executed. Note that most of the commands will be Steps, but additional commands are envisaged for data pre-fetch from disk, parameter database access etc.; see [11] for more details.

The description so far does not involve the blackboard yet. This corresponds with the intended operation of the calibration component in single node operation when all UV data can be stored in a single file, say on the end-user's desktop system. Although this single node operation won't meet the performance requirements for the calibration of a LOFAR observation, it is a valuable mode for the tuning and development of new strategies.

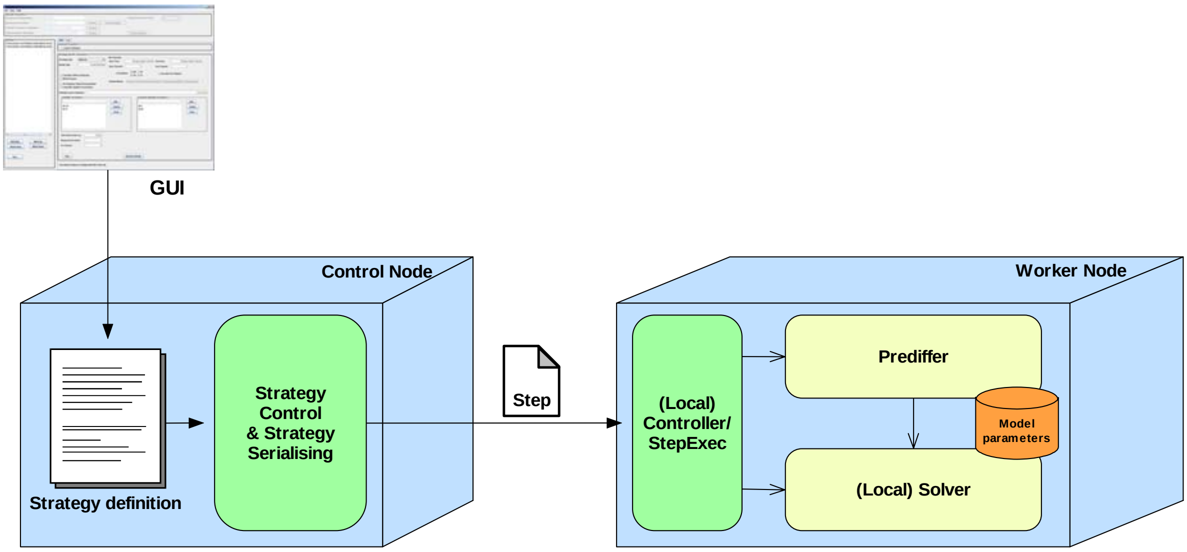


Figure 12 *Configuration and execution of a Strategy. A calibration run starts with the definition of the calibration Strategy in terms of Steps and Multi-Steps in the SAS GUI. This results in a parameter file containing the complete definition of the calibration run. This definition is interpreted by the (global) Strategy controller that dispatches small commands (e.g. Steps) to the worker node. The worker node performs the Predict, Solve, Subtract, or Correct Steps and reports the results back to the Strategy controller.*

4.2.2 The BlackBoard Pattern

We have chosen to base the architecture of the calibration component on the BlackBoard pattern. With this choice we attain the following goals:

- Control and processing are isolated from each other in the architecture. This separation serves separation of concerns in the description of the architecture as well as at execution time. The latter is especially important for the performance optimisation of the processing tasks.
- The blackboard pattern is a well known solution to shared data problems, which we have due to a single set of model parameters versus multiple nodes working on the observed data simultaneously.
- The blackboard pattern is very efficient and flexible in allowing for new or adaptive calibration strategies. It is effective in the freedom we have to implement very complex algorithms. It is flexible in that changes to the Strategy are very well isolated in a single code and execution thread of the central controller.

The reasoning and experiments leading to the final choice to select the blackboard patterns as basic element of the inner loop architecture is described in [6]. In the remainder of this section we describe the blackboard pattern as it is used here.

The generic operation of BlackBoard Selfcal is illustrated in Figure 13. The BlackBoard forms the main communication mechanism between the central controller and the distributed workers. Two flows of information go through the BlackBoard: control and status on the one hand, and results and quality metrics on the other hand. Not shown in the figure is that users and operators can inspect the progress of the calibration process through the BlackBoard, either during execution or afterwards. This is particularly important for tuning or de-bugging.

The global controller can also use the status and result data on the BlackBoard to optimise the algorithm or execution. Based on the results from the solver, the controller can decide if another iteration is needed or if a dependency may be solved by changing to another parameter.

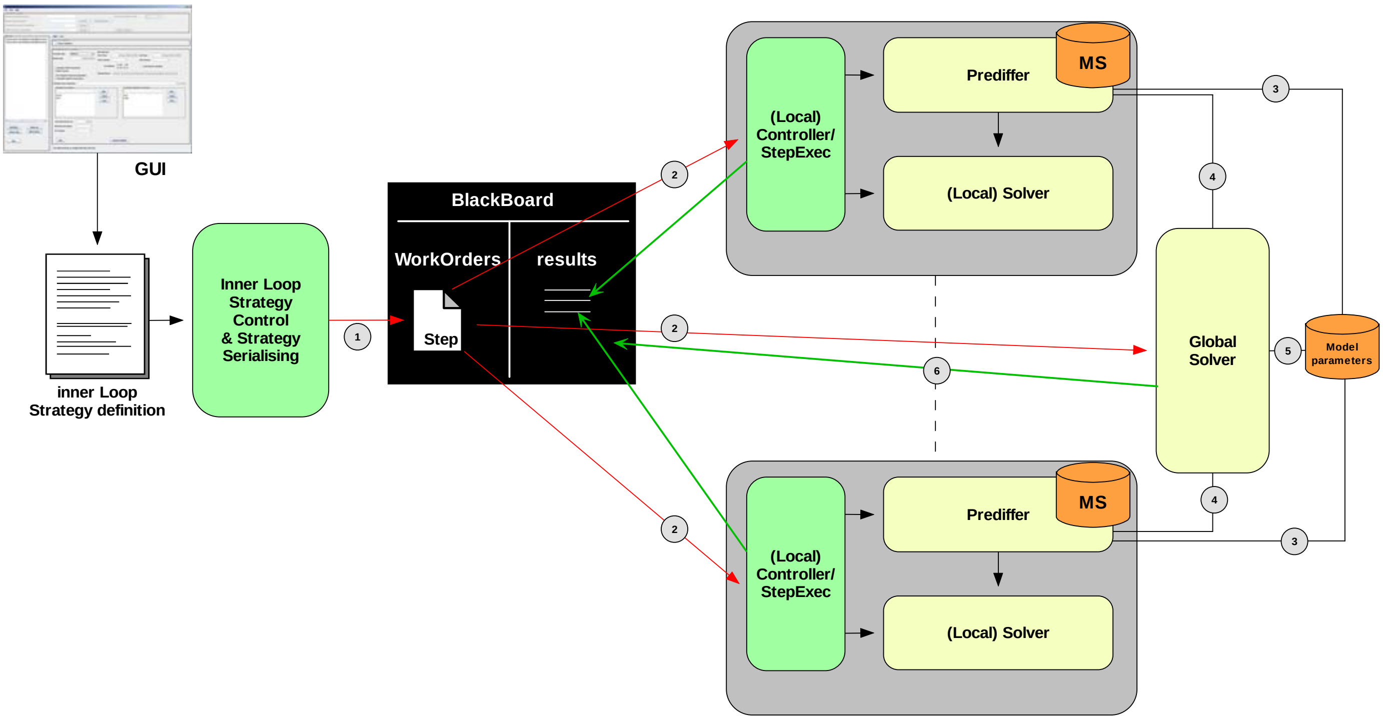


Figure 13 *Execution of Strategies in the distributed BlackBoard Selfcal implementation of the calibration component. A run starts with the definition of the calibration Strategy in terms of Steps and Multi-Steps in the SAS GUI. This results in a parameter file containing the complete definition of the calibration run. This definition is interpreted by the global controller that dispatches small commands (called Steps) to the BlackBoard (1). The worker nodes query the blackboard to find their commands (2). Based on the Step definitions they receive in this way they perform the Predict, Solve, Subtract, or Correct Steps and report the results back to the BlackBoard (green arrows). The worker nodes read model parameters from a shared parameter database (3). Global solve operations are performed on a central solver (4) which is also controlled via the blackboard (2b). Finally the updated parameters are written back into the shared parameter database (5) and quality metrics are written to the BlackBoard (6). For Steps that do not involve the full frequency band, the global solver is not used and local solvers are used instead.*

4.3 Implementing the Imager component

The imager architecture will also be based on the BlackBoard pattern. Very similar to the BlackBoard Selfcal, we will have a configuration, an image run definition and a central controller. The worker nodes of the imager will run the imager component. Each Imager work node will image UV data from a UV data file into a facet image. A dedicated process will “stitch” those facets together into a wide field image. This is illustrated in Figure 14.

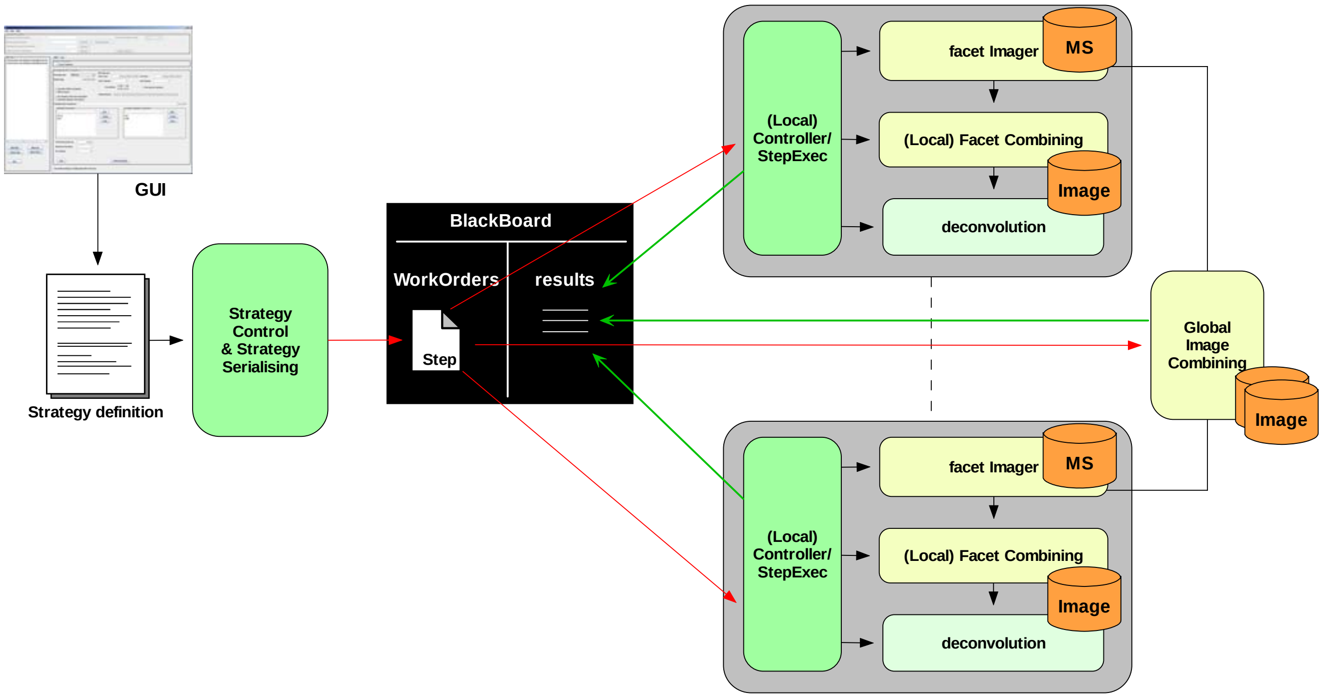


Figure 14 *Organisation of the BlackBoard Imager component. The configuration GUI, Image Strategy definition parameter file and the central controller are analogues to the calibration component (Figure 13). Dedicated workorders will be put on the BlackBoard for the imager work nodes.*

In the same architecture we may also include image analysis components. Deconvolution, for example, is an iterative process, much like the calibration process and it maps in the same way on the BlackBoard architecture as the calibration component [13]. A Global Solver and many Predict / Solve Steps based on image parameters are needed. Although the Solver and Step internals will be different the architectural view is the same as for the calibration component shown in Figure 13.

To facilitate integration of different components, the data access and caching behaviour of the worker nodes will also be subject to explicit control by the central controller.

The functionality of the imager and image analysis components will be based on the AIPS++ imager code.

5 Performance aspects

5.1 Distribution and parallelisation

As described in section 4.2.2, the BlackBoard is the main architectural construct to support distributed operation. In this section we describe how such distribution is organised and how parallel execution is used.

The BlackBoard is used to distribute the work of the calibration component over multiple nodes. In principle, each node will host one worker working on the UV data for, e.g. a single subband (~200 kHz). This is the simplest mapping of the total UV dataset of 165 subbands onto a cluster computer. Most of the calibration Steps operate on a single frequency subband and all nodes can work concurrently on those Steps. Solving for models over the entire observed frequency band, such as the bandpass, may be implemented using a global solver (see Figure 13) in which case the controller will impose a synchronisation between all nodes before the updated parameters from the global solver are issued to all nodes. Note that this synchronisation is not necessarily a full barrier type if the controller issues workorders (Steps) that are not dependent on the parameters of the global solve Step. More distribution is used within the worker nodes. Each node will run separate processes for predict, solve and data access. Those processes communicate using the shared memory system within the nodes.

The imager component will be distributed in a similar way with (integrated) frequency subbands / channels per node. Since the image will be built from smaller facets, further distribution within a worker node is straight forward.

As described above, the distribution of data and processing tasks is primarily based on distribution in the frequency domain. All software components could, in principle, also operate with a time based distribution of data and tasks. However, the code and data access patterns are optimised for distribution in frequency, which corresponds with the ordering of data in the output files of the correlator.

Parallelisation is used within the nodes to speed up parts of the work. We may use multiple CPU's and cores within the nodes to work in parallel on for example predict tasks over all channels in a subband. In the current BBS implementation threads and OpenMP are used to implement this type of parallelisation. More parallelisation and vectorisation is used in typical library functions such as FFT and matrix arithmetic. The vectorisation is based on the SSE instruction set on Intel processors[5].

Given this distribution and parallelisation scheme, the target platform for the calibration and imaging components is a ~165 node cluster computer with multi-core multi-processor SMP Intel architecture. Alternatively we may also build small units of multiple smaller cluster nodes with fast interconnects and shared access to a file server.

5.2 Model Parameters handling

A database is used to store all model parameters for a specific observation. This database is filled at the start of the calibration based on the parameter files generated by the SAS configuration GUI and scripts. From an architectural viewpoint it is decided to use a single database in order to have full control over parameter distribution and prevent having multiple copies of the same parameter.
In the detailed design this centralised database view is altered in two ways:

[5] Non-vectorised code will be used on processors that do not support SSE instructions.

1. Parameter values are cached at the worker nodes to prevent multiple database access during iterations. A write locking mechanism is used in the database to control the write access to a parameter if multiple copies exist. The update of new parameter values is explicitly controlled by the central controller
2. A distributed database with full replication can be used to increase access performance. The replication must be controlled by the central controller.

If parameter caching is applied, we think that a single (central) large database server will provide enough performance for a 200-500 node distributed calibration run. However, we have built a demonstrator blackboard based on a distributed database (option 2).

5.3 Data Handling

This section describes how disk access and data caching in RAM are used. The description is at two levels: on an architectural level we describe how disk and RAM usage is controlled and distributed. On a detailed design level we describe how the data access and tiling in the inner loop component is organised.

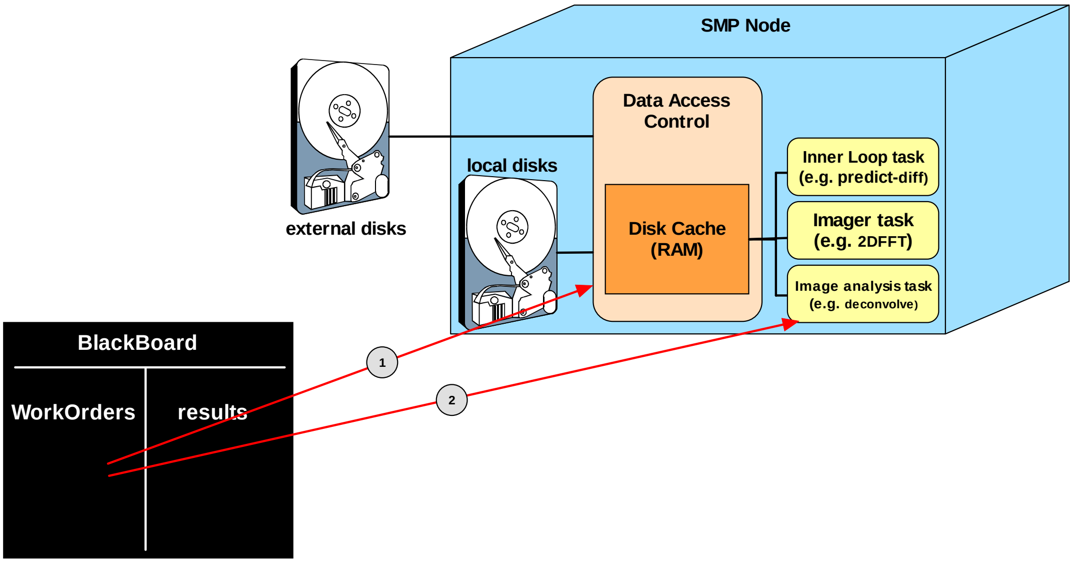


Figure 15 *Illustration of controlled disk access and caching in the BlackBoard controlled Selfcal and imaging components. All disk access and data caching is controlled by a dedicated process per worker node. Other tasks on the node access the cached data through shared memory or file handlers.*

Disk access and data caching is organised per worker node, as illustrated in Figure 15. On each node a data access controller process in running. This process accesses disks both internal and external and caches data in RAM. Other processes running on the same node can access data from that cache using shared memory. The data access control process will try to continuously read or write to all relevant disks since disk access bandwidth is a limiting factor in the calibration. It is important to note that the disk access and caching is shared between the calibration, imager and image analysis components. The disk caching can be explicitly programmed, or a system function like `memmap()` may be used like in the current BBS implementation.

The data access tilings in the calibration component are defined by the work domain and solve domain concepts. The workdomain is the selection of data that is read from disk by the data access controller process. The controller may also specify data selections, such as a subset of the baselines or frequency channel selection, or data integration and gridding operations.

5.4 Deployment and using the Blue Gene/L

The blackboard pattern described in section 4.2.2 is an excellent mechanism to make the processing power of the Blue Gene/L supercomputer available for calibration. The blackboard is already used to distribute different types of commands to the kernel processes and solvers. In the same way, we can send specific commands to processes running on the BG/L.

Given the limitations in the run-time environment of the BG/L, we will not be able to run all code of the inner loop on BG/L. Especially the absence of support for multi threading and some system functions will limit the portability of the BBS components. However, it is most probably feasible to port the predict tree itself (i.e. the evaluation of the ME) to the BGL platform. When we add an interface to a static parameter set, we can use the BGL to calculate the predicted visibilities for large sets of Cat-II sources, which can then be subtracted from the UV data, see Figure 16. Also the 2DFFT in the core of the imager may be deployed in this way on the Blue Gene/L.

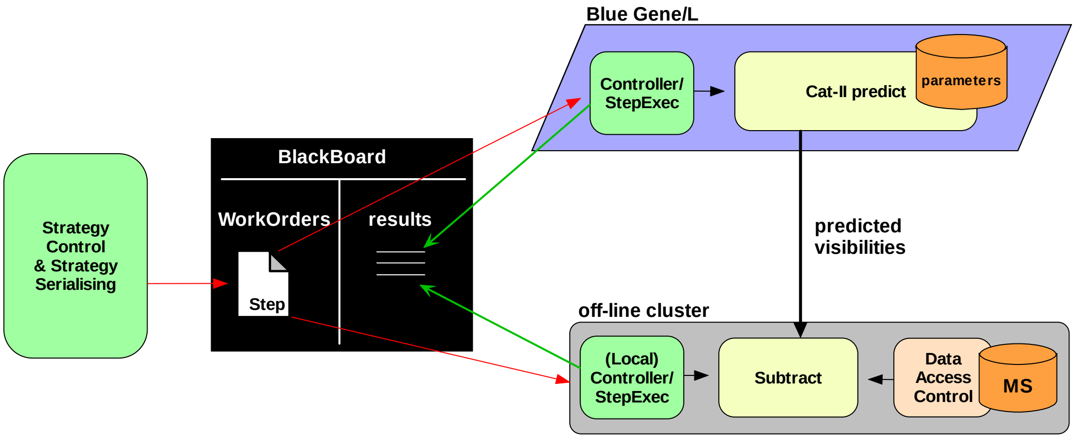


Figure 16 *Use of the BG/L for prediction of the Cat-II source contributions. The actual subtract form the UV data is performed on the off-line cluster where the UV data is available locally.*

The overall deployment of the calibration software on the CEP hardware systems in Groningen is shown in Figure 17. Following the dataflow through the system, we start with the correlator writing UV data files onto the storage system. If possible, on-line data preparation tasks like flagging will be performed on the Blue Gene/L prior to storage. The storage system may either be a cluster of file servers or it may have a SAN like architecture. The auxiliary cluster will be used to perform the off-line data preparation tasks, accessing the UV data on the storage system. The Major Cycle components run on the off-line cluster. These nodes deploy multiple tasks concurrently while the execution order of those tasks is controlled by the imager and calibration controllers. A database server is used to run the BlackBoard database.

As described above, the Blue Gene/L may be used as a kind of co-processor for specific tasks, assuming resources are actually available and not needed for correlation of other observations.

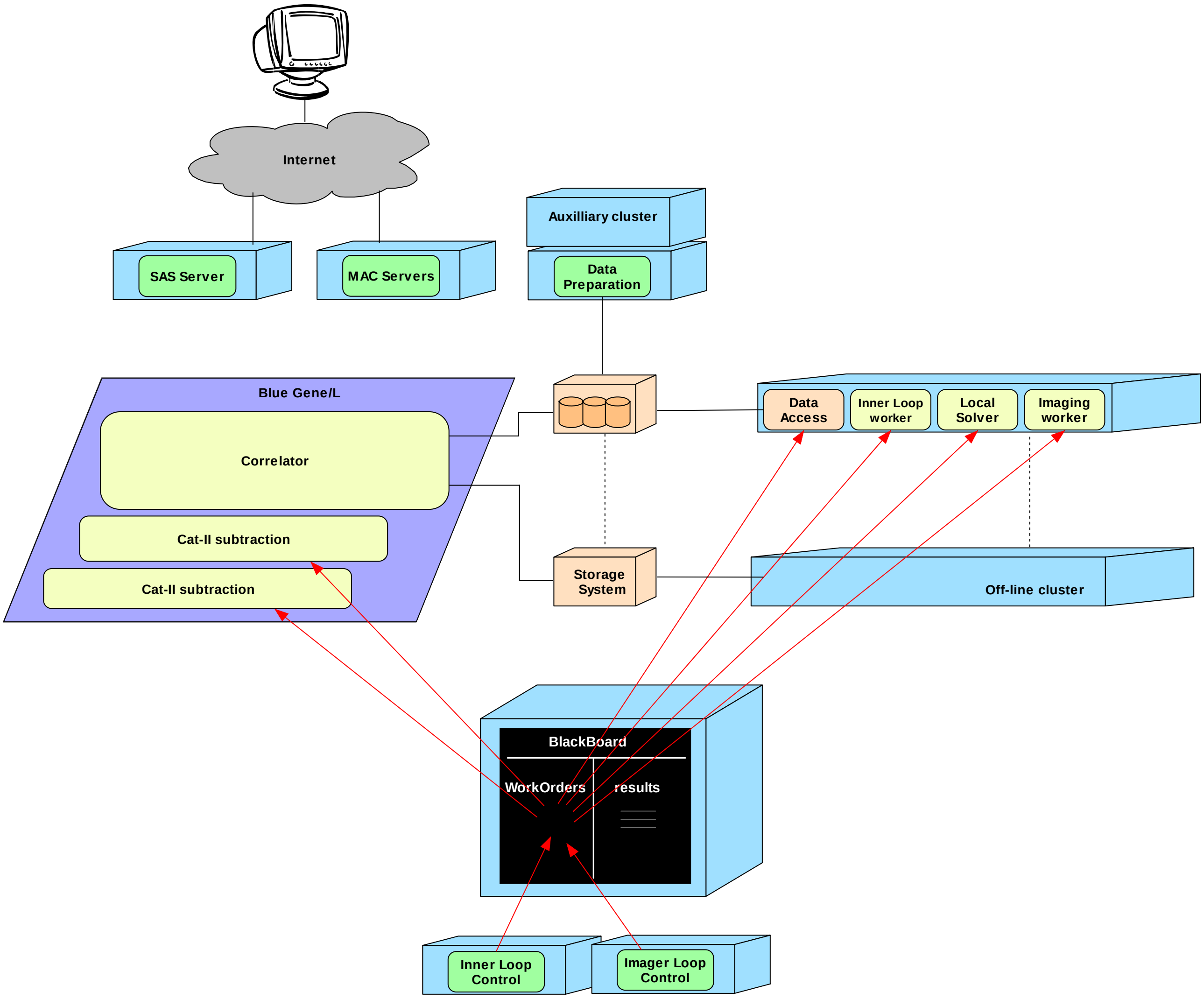


Figure 17 *Deployment overview of the calibration software. The main resources used are the file servers in the storage system, the off-line cluster and the server machines for the blackboard database and controllers. The data preparation tasks run on the auxiliary cluster and part of the inner loop and imager tasks may use the Blue Gene/L.*

5.5 Minimisation

The calibration component performs a minimisation by fitting the model parameters in such a way that the difference between the model prediction and the observed data is minimised. The minimisation process in the calibration can be optimised at several levels:

1. At a global level the order in which the parameters are solved for and the data traversal order determine the global optimisation strategy
2. The solver that is used to minimise a limited parameter set for a limited data selection (solvedomain)
3. The implementation of the models may differ in terms of accuracy and speed between stages in the global optimisation strategy

The architecture of the inner loop supports these three optimisation options and provides flexibility and ease of change to experiment with those. Below is a short description of how these optimisations can be made in the actual implementation of the calibration architecture once we start research on an optimal minimisation scheme. Note that those options are not yet implemented.

The global level strategy is defined by the SAS component and stored as Strategy definitions in a parameter file. The Strategy definitions provide the freedom to define new global minimisation algorithms; actually, we have designed this part with a Monte-Carlo like algorithm in mind as opposed to the strict parameter sequence in the "normal" scenario currently used. Defining a new global algorithm involves re-defining the Steps sequences and the workdomain/solvedomain loops in the controller code.

The solver itself can be replaced by another solver type. This will involve some work to wrap into the solver class and possibly change the call parameters. It will involve minor work to provide for instance additional gradients since these are standard outputs of the ME evaluation tree. Using different types of solvers might also be needed for the deconvolution component [13].

The model objects can get multiple model implementations and parameter sets. The execution of a Strategy setting will then contain additional information about the model variant to be used during execution of that scenario. While changing from model variant we may initialise the new model parameters based on the old ones. This functionality is added to the model object.

5.6 On-line Calibration

The architectures of the CEP system and of the calibration software both include the option to perform a calibration in real time and use the estimated model parameters to correct the data processing in real time. Such real-time calibration may be needed to smooth the ionospheric phase rotation correction for rotations of multiple degrees within the integration time of the correlator; see Figure 18. A similar real-time calibration may be applied to track and predict unknown RFI sources such as planes and cars.

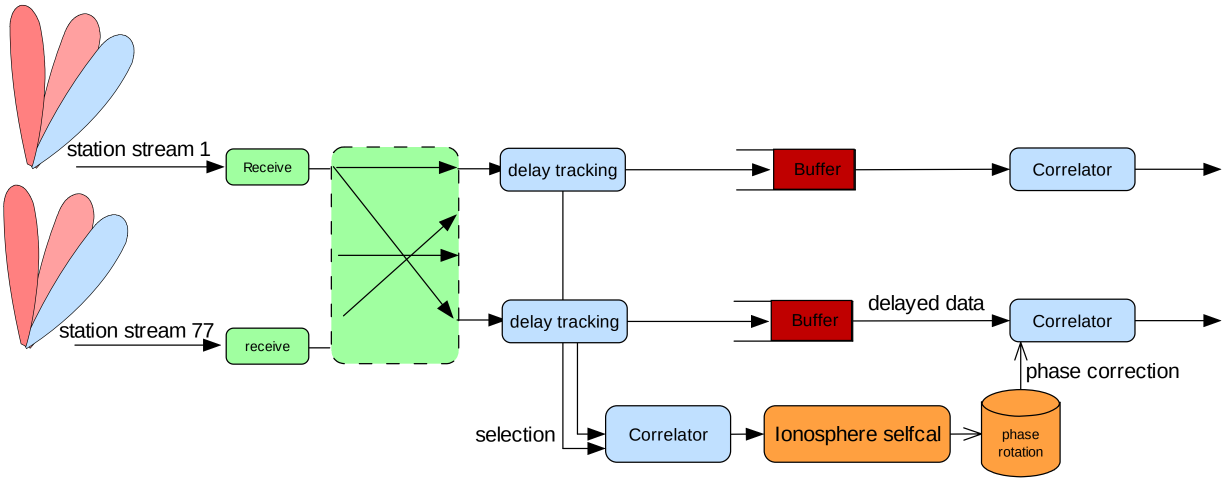


Figure 18 *On-line ionospheric calibration in the data flow. An on-line calibration component is attached to the data flow prior to the correlator. Based on a subset of the data a calibration solve loop is traversed for the I-Jones matrix in order to get the observed phase rotation parameterisation. Based on this phase rotation we can smoothly correct the full data stream before the correlator.*

The BlackBoard calibration implementation can be used for real-time operation as well; the data interface is already decoupled from the calculations and can be replaced by a stream input. Actually, the output from the data access controller process is already a stream. Since the real-time calibration will perform the same algorithms sequence all the time, we can directly attach the central controller to the worknodes and send the Step definitions only once.

6 Development status

This section describes the status (mid March 2007) of the development of the LOFAR calibration production software, based on the BlackBoard Selfcal (BBS) architecture described in the former chapters. The current development is focussed on the data processing pipeline for LOFAR CS1: the CS1 Imaging Pipeline, see Figure 19.

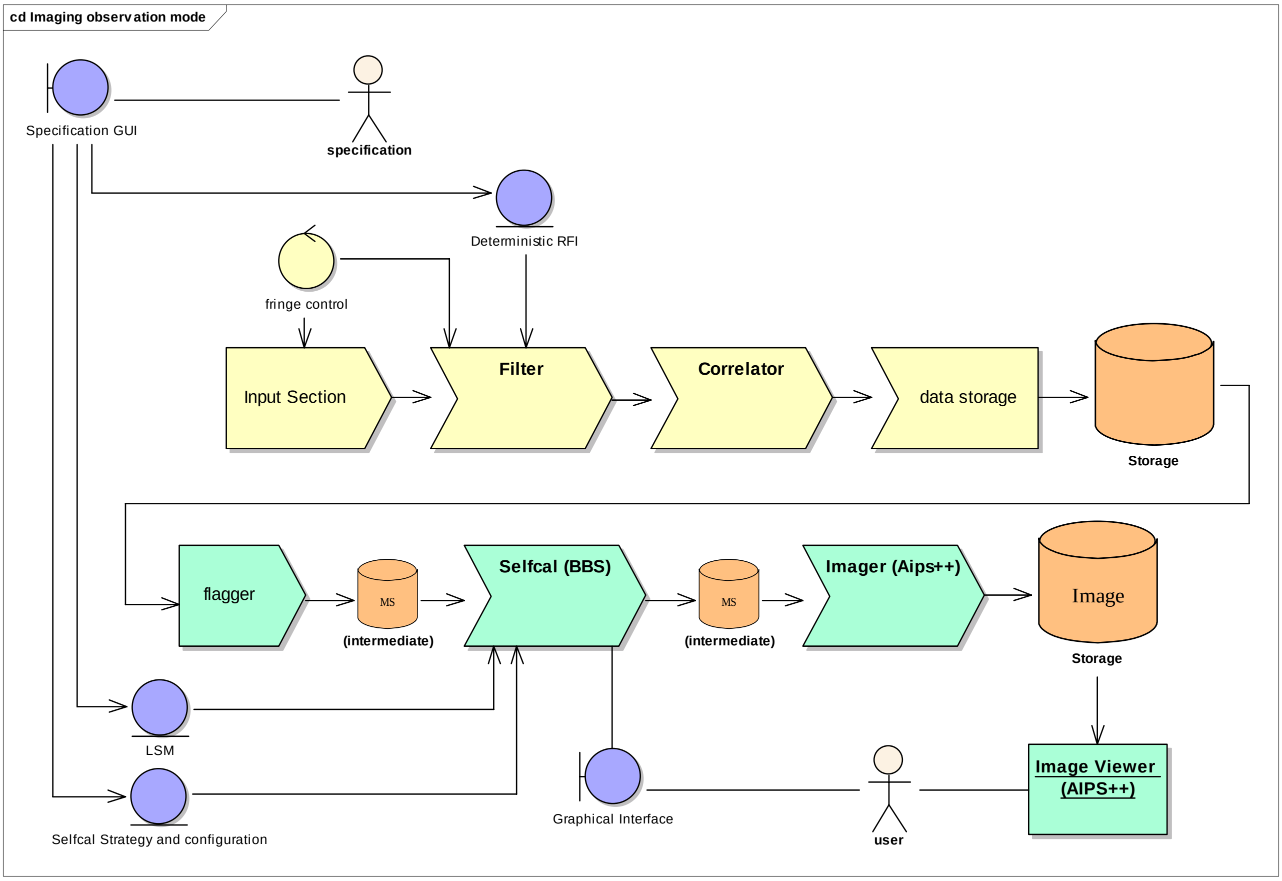


Figure 19 *Illustration of the on-line and off-line processing pipelines for imaging mode observations in CS1. The pipeline settings are defined in the Specification GUI (part of the SAS system). A few of the typical configuration files are shown as inputs to the processing pipelines. The on-line processing pipeline is shown in light yellow. This pipeline receives data from the stations. The data first goes through a poly-phase filter bank and next through the correlator, which calculates all cross and auto-correlations. The output from the correlator is written into a MS file.*
The off-line processing pipeline starts by reading the MS file and sending this data through the WSRT Flagger. The output from the Flagger is again written to file. The next pipeline step is the BBS Selfcal, which produces an updated MS file with residual or corrected data, which is converted into an image by the AIPS++ imager.

We are working towards the integration of SAS, MAC, OLAP, Flagger, BBS, and Imager. The integration is planned to be finished at the end of Q2 2007. At that point SAS and MAC can be used to schedule and control the Stations, the on-line processing and the off-line processing for calibration and imaging applications. We will be able to do beam tracking, delay tracking, and fringe stopping, have varying number of micro-stations per station and trade subbands against beams on the sky. After Q2 2007 SAS and MAC will mature and grow to cope with other KSP applications. The on-line processing will focus on optimizing Blue Gene performance, support of other observational modes, and integration with the off-line cluster.

Currently there is a 32 node computer cluster in Groningen available for on-line and off-line processing. 12 nodes are used as storage nodes and 18 nodes will be made available as off-line processing cluster. This cluster will be upgraded when more processing power is needed. We will start working towards a design of the off-line cluster and hope to have a first specification at the end of Q2 2007.

The CS1 off-line processing consists of a Pre-Processing component, including a Flagger component, a Selfcal component, and an Imaging component. The pre-processing component consists of several steps of splitting up and combining Measurement Sets. Also, the bandpass will be divided out and the worst RFI will be flagged. This Pre-Processing component is planned to be ready by April 2007.

The Flagger is based on the WSRT Flagger. It is extended to flag on residual data and in the end it will have to be integrated into the Selfcal component. There will also be a need for different Flagging heuristics. A first new heuristic (using a frequency based median, instead of a time based median) is also implemented.

The Selfcal component is implemented in a BlackBoard framework and the software is called BBS (BlackBoard Selfcal). The Blackboard framework is used to distribute and parallelize the selfcal application over multiple compute nodes. BBS has several ingredients: control, data interface, strategy nodes, solver, and Measurement Equation evaluation. The current version of BBS is based on:

- a single (compute) node,
- point sources,
- complex Jones matrices in different directions,
- bandpass correction (although this will now be done in the pre-processing component).

At this moment BBS is being distributed and parallelized. This means that apart from the local controller at each compute node, there will be a global controller and a BlackBoard; the latter being implemented by a database. This activity is planned to be finished in March 2007, but will probably be slightly delayed due to preparations for the System CDR.

BBS has a data interface towards the Measurement Set. This is functioning, but may need to be updated along the development path. Strategy nodes are: Predict, Solve, Subtract, Correct, Shift, Refit, and Flag. All these nodes are available except for the latter three. The solver is based on the AIPS++ fitting classes and is the same as the one used by the MeqTrees software (see below). Currently it is only possible to use the solver locally on one compute node. When global control is available, we will start working on a global solver, i.e. solving for parameters that share data over multiple compute nodes.

It is our impression that the BBS infrastructure is in a good state. However, the Measurement Equation evaluation part still needs a lot of work. Currently BBS is able to correct for the bandpass and to evaluate the ME for point sources thereby having separate solutions for different source directions. However, in BBS there are no beam models, no ionospheric models, and no other source models than point sources. Our understanding of these models still needs to be developed and that will be done with the use of the MeqTrees software. For now, we can start to revisit the design of the ME Evaluation part, so that BBS is ready when the first beam and / or ionospheric models come from MeqTrees.

Apart form the production software development we have also implemented an extensive “working prototype” software package, called MeqTrees, that is used to help develop the self calibration algorithms itself. At the same time the MeqTrees software also prototype many architecture and detailed design concepts which form a well-tested base for the development of the production software BBS. The MeqTrees software is in a mature state, given the scope in which it has to operate, and it is being used for calibration of CS1 data. This has resulted in a couple of calibrated all-sky images. At the same time the BBS software will also be used on

CS1 data. Cross-checks between the two systems will be used extensively to pinpoint potential problems in either of the packages.

The Imager component currently consists of the AIPS++ software. This allows us to make all sky images per Measurement Set, or say per subband. Images can be exported as AIPS++ images or as FITS files. They can be viewed by the AIPS++ viewer (for AIPS++ images) or the kvis viewer (for FITS files). At the same time ASTRON (by means of Ger van Diepen) and ATNF (by means of Tim Cornwell c.s.) are working on a distributed version of the Imager component. This version will be based on the BlackBoard framework and AIPS++ core libraries. It is planned to have a first operational version by August 2007.

7 References

[1] J.E. Noordam, LOFAR Calibration Framework, LOFAR-ASTRON-ADD-015, v1.0, 2006.

[2] K. v.d. Schaaf, A. Latour, LOFAR central Processing Facility, Architecture description, v.3, LOFAR-ASTRON-ADD-012, 2007.

[3] A. Latour, LOFAR Software Plan, LOFAR-ASTRON-PLN-034, 2007.

[4] J.P. Hamaker, J.D. Bregman, R.J. Sault, Understanding radio polarimetry I, Astron. Astrophys. Suppl. Ser., 117, 137 – 147, 1996.

[5] J.E. Noordam, The Measurement Equation of a generic radio telescope, AIPS++ implementation note 185, 1996.

[6] K. van der Schaaf, Requirements analysis and Architectural design of the BlackBoard SelfCal application, LOFAR-ASTRON-SDD-002, revision nr. 0.1, 2003.

[7] J.W. Romein, On-Line Application Processing (OLAP): Software Design Document, v.2, LOFAR-ASTRON-SDD-036, 2007.

[8] R. Overeem, Specification, Administration & Scheduling, Architectural Design Document, v.2, LOFAR-ASTRON-SDD-041, 2007.

[9] R. Overeem, E. Lawerman, LOFAR Low Frequency Array Monitoring and Control ADD, LOFAR-ASTRON-ADD-005, v.3.0, 2007.

[10] K. v.d. Schaaf, LOFAR Software Architecture, v.1.1, LOFAR-ASTRON-SDD-049, 2006.

[11] G.M. Loose, J.E. van Zwieten, BlackBoard SelfCal (BBS), Software Design Document, LOFAR-ASTRON-SDD-053, 2007.

[12] F. Buschmann, Meunier, Rohnert, Sommerlad & Stal, A System of Patterns, April 1999, John Wiley & Sons Ltd.

[13] T.J. Cornwell, G. van Diepen, Synthesis design for CONRAD, ATNF draft document, 2007.