

LOFAR Self-Calibration Using a Blackboard Software Architecture

G. Marcel Loose

ASTRON, Dwingeloo, The Netherlands

Abstract. One of the major challenges for the self-calibration of the new generation of radio telescopes is to handle the sheer amount of observational data. For LOFAR, an average observation consists of several tens of terabytes of data. Fortunately, many operations can be done in parallel on only part of the data. So, one way to take up this challenge is to employ a large cluster of computers and to distribute both data and computing power. This paper focuses on the architectural design of the LOFAR self-calibration system, which is loosely based on the Blackboard architectural pattern. Key design consideration was to provide maximum scalability by complete separation of the global controller—issuing sequences of commands—on the one side; and the local controllers—controlling the so-called ‘kernels’ that execute the commands—on the other side. In between, resides a database system that acts as a shared memory for the global and local controllers by storing the commands and the results.

1. Introduction

The LOFAR telescope, once fully operational, will produce raw data sets that are *very* large. These data sets need to be calibrated in an iterative way to correct for instrumental effects, like band-pass and beam shape; and ionospheric effects, like Faraday and phase rotation. In this paper, we will focus on the architectural design of the LOFAR self-calibration system from a system control point-of-view.

2. Architectural Design

The data volume of an average LOFAR observation will in the order of tens of terabytes. This size must be reduced, otherwise it will be impossible to do long-term archiving and local (post-)processing. Moreover, the output data rate of the correlator will be in the order of a gigabyte per second. Neither a single disk, nor a computer system can handle this data rate, therefore data must be distributed.

2.1. Design Considerations

Data should be distributed such that: all processing can be done without re-ordering; large chunks of data can be processed locally; and only few data have to be exchanged. Data can be distributed along a few axes.

Time is a bad choice, because a time-slot contains a lot of data (up to several gigabytes), which may lead to problems in the on-line system when all data

of a time-slot are sent to a single machine. It will also make parallelization of imaging difficult, because the data of all time-slots have to be combined.

Baseline seems a better candidate. However, it will lead to imaging problems, because data from large amounts of gridded Fourier transformed data have to be sent around.

Frequency is the best choice. Images are usually created per channel, so, in principle, imaging could be done locally. This distribution scheme matches nicely with the way the correlator and the on-line system is designed.

One important requirement is scalability. A heterogeneous cluster will be a fact of life, since hardware will be bought in stages and will be replaced in stages. Therefore, unnecessary coupling between the different computing nodes should be avoided, and data should be processed locally as much as possible. Communication between the computing nodes should be done using a global shared memory. Several architectural patterns describe this approach. Among these, the Blackboard pattern (Buschmann et al. 1996) is probably the best-known.

2.2. Blackboard pattern

The Blackboard architecture is ideal for solving problems for which no predetermined algorithm or solve strategy is known. However, the “best” algorithm to perform a self-calibration run can be chosen from a relatively short list of calibration strategies in advance, giving us a significant performance gain. In fact, the Shared Repository pattern (Lalanda 1998), which can be seen as a generalization of the Blackboard pattern, is probably a better match for the BlackBoard Selfcal (BBS) system. It realizes indirect communication using a repository as shared memory. Figure 1 shows the specialization hierarchy of patterns based on the Shared Repository pattern.

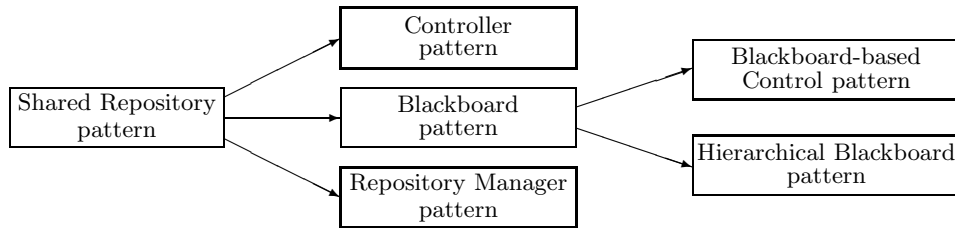


Figure 1. Patterns based on the Shared Repository pattern (Lalanda 1998). The two patterns in a thick framebox are of special interest for BBS.

Controller pattern introduces a control component in the system, which rules the system and schedules activation of other components. This pattern can be applied to deterministic problems where sequences of components activation can be determined off-line and coded in the controller using various techniques.

Repository Manager pattern is applicable in a distributed environment. It introduces a repository manager which sends notification of data creation or modification to the software components.

3. System Overview

3.1. Subsystems

The Blackboard Self-Cal system is split into three parts. BBS Control takes care of the distributed processing by means of the Blackboard pattern. BBS Kernel does the actual processing; it executes a series of steps, where each step consists of an operation like SOLVE or CORRECT. The BBS Database consists of two databases—Command Queue and Parameter Database—that together form the blackboard. This separation of concerns maximizes scalability.

BBS Control is responsible for controlling the execution of a self-calibration strategy (figure 2). A strategy describes one iteration in the so-called *Major Cycle* (Nijboer & Noordam 2006). It defines the relationship between the observed data and the model data, and consists of an ordered list of commands that will be executed by the BBS Kernel subsystem. The key idea is that the BBS Kernel keeps a subset of the data (the so-called *work domain*) in memory. As many commands as possible are executed on these data before the next chunk of data is accessed.

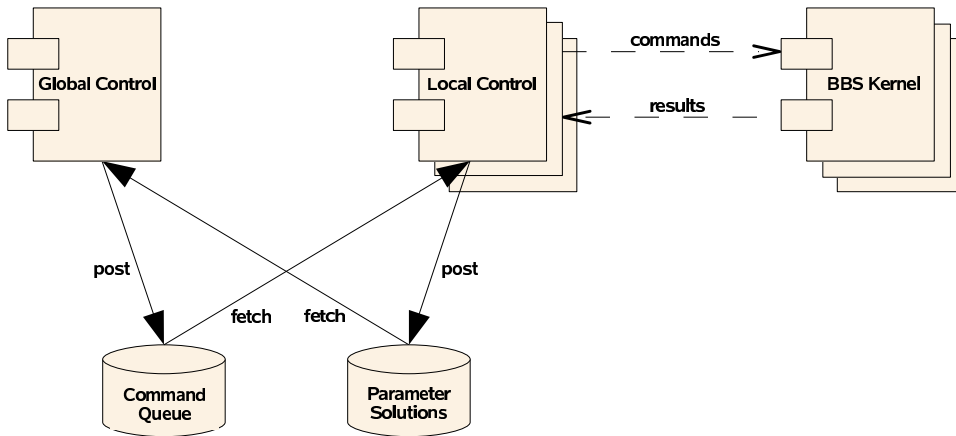


Figure 2. Design of the BBS Control system. Global Control posts commands to the Command Queue. These commands are asynchronously retrieved by Local Control and forwarded to the BBS Kernel. After execution of the command, BBS Kernel returns the result to Local Control, which in turn posts it to the Parameter Database. Global Control checks the quality of these solutions and takes appropriate action.

BBS Kernel implements the Measurement Equation (Hamaker, Bregman & Sault 1996), which models the response of an interferometer given a description of the sky, the environment, and the interferometer. It consists of two components. The *kernel* implements operations like PREDICT, SUBTRACT, and CORRECT. The *solver* calculates estimates for the model parameters by minimizing the difference between model and observation.

BBS Database consists of two different databases. The Command Queue stores the command to be executed by the BBS Kernel and the status results re-

turned by each kernel. The Parameter Database stores (intermediate) solutions of the model parameters calculated by the BBS Kernel. Access to the Parameter Database is minimized in order to avoid performance penalties.

4. Current Status and Future Work

Currently, the control framework runs on a 12-node Linux cluster, using a PostgreSQL database as shared memory. Figure 3 shows the result of one of the first successful calibration runs on a gigabyte-sized dataset.

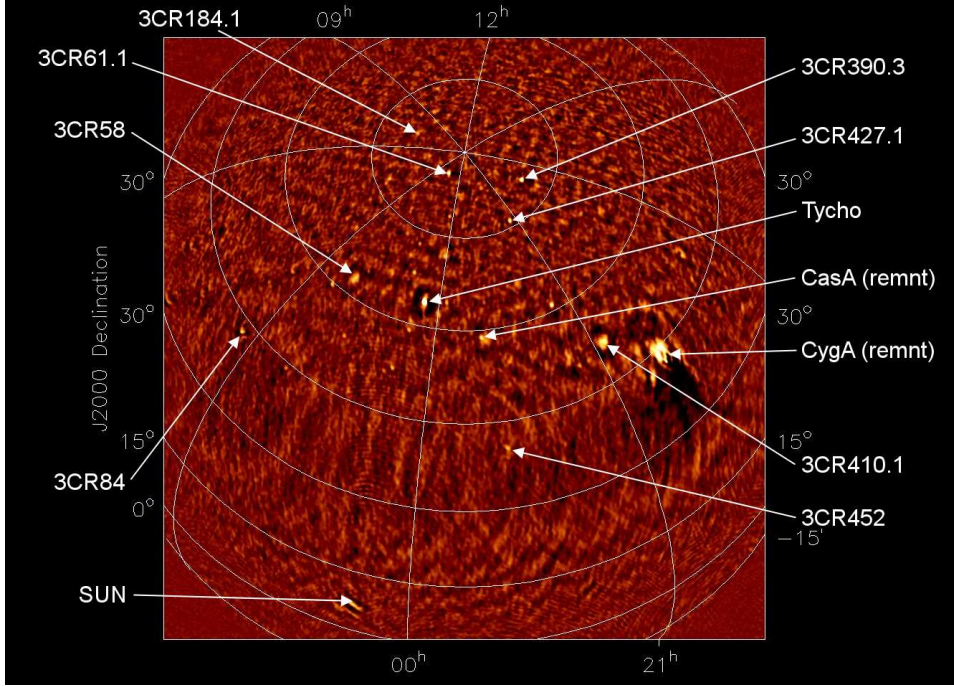


Figure 3. Image produced from a 16-hour CS-1 observation on 30 March 2007, using six 160 kHz subbands between 47.5 and 65 MHz. Note the remnants of two strong radio sources, CasA and CygA, after their subtraction.

The scalability of the framework will be tested in the coming period, as the cluster is expected to grow to several hundreds of nodes. Support for calibration of beam-shape, and ionospheric effects will be added to the BBS Kernel.

References

- Buschmann, F., et al. 1996, Pattern-Oriented Software Architecture, Volume 1 (John Wiley & Sons Ltd.)
- Hamaker, J. P., Bregman, J. D., & Sault, R. J. 1996, A&A, 117, 137
- Lalanda, P. 1998, in Proceedings of PLoP'98 (Pattern Languages of Programs'98), http://jerry.cs.uiuc.edu/~plop/plop98/final_submissions/P24.pdf
- Nijboer, R. J., Noordam, J. E. 2006, in ASP Conf. Ser. 376, ADASS XVI, ed. R. A. Shaw, F. Hill, & D. J. Bell (San Francisco: ASP), 237