

LOFAR Self-Calibration

Using a Blackboard
Software Architecture



Overview

- The Setting
- Architectural Design
 - Design Considerations
 - Blackboard Pattern
- System Overview
 - Subsystems
 - Strategies and Steps
- Current Status and Future Work



The Setting

- Raw UV-data must be calibrated iteratively to
 - Correct for instrumental effects, like
 - Band-pass
 - Beam shape
 - Correct for ionospheric effects
 - Faraday and phase rotation
 - Create a local sky model, containing
 - Point sources
 - Gaussian sources
 - Extended sources



Architectural Design

- Design Considerations
 - Data Volume
 - Distributed Processing
 - Scalable Architecture
- Blackboard Pattern



The Challenge

- Data Volume of an average observation will be *tens of terabytes*.
 - Size must be reduced, otherwise...
 - Long-term archiving will be impossible
 - Local (post-)processing will be impossible
- Data Rate will be in the order of a *gigabyte per second*.
 - Data must be distributed, because...
 - A single hard-disk cannot write a GB/s
 - A single computer system cannot handle a GB/s



Divide and Conquer

- Data should be distributed such that, for all processing:
 - It does not have to be reordered
 - Large chunks can be processed locally
 - Only little data have to be exchanged



Divide and Conquer

- Data could be distributed along a few axes
 - *Time* is a bad choice
 - The data size of a single time-slot is too large to be sent to a single machine
 - *Baseline* is a better choice
 - However, it will lead to imaging problems
 - *Frequency* is the best choice
 - Images are usually made per channel, so imaging could be done locally



Scalable Architecture

- Heterogeneous cluster
- Decouple the computing nodes
 - Process locally. Bring the process to the data
 - Communicate through a global shared memory
- Several architectural patterns describe this approach

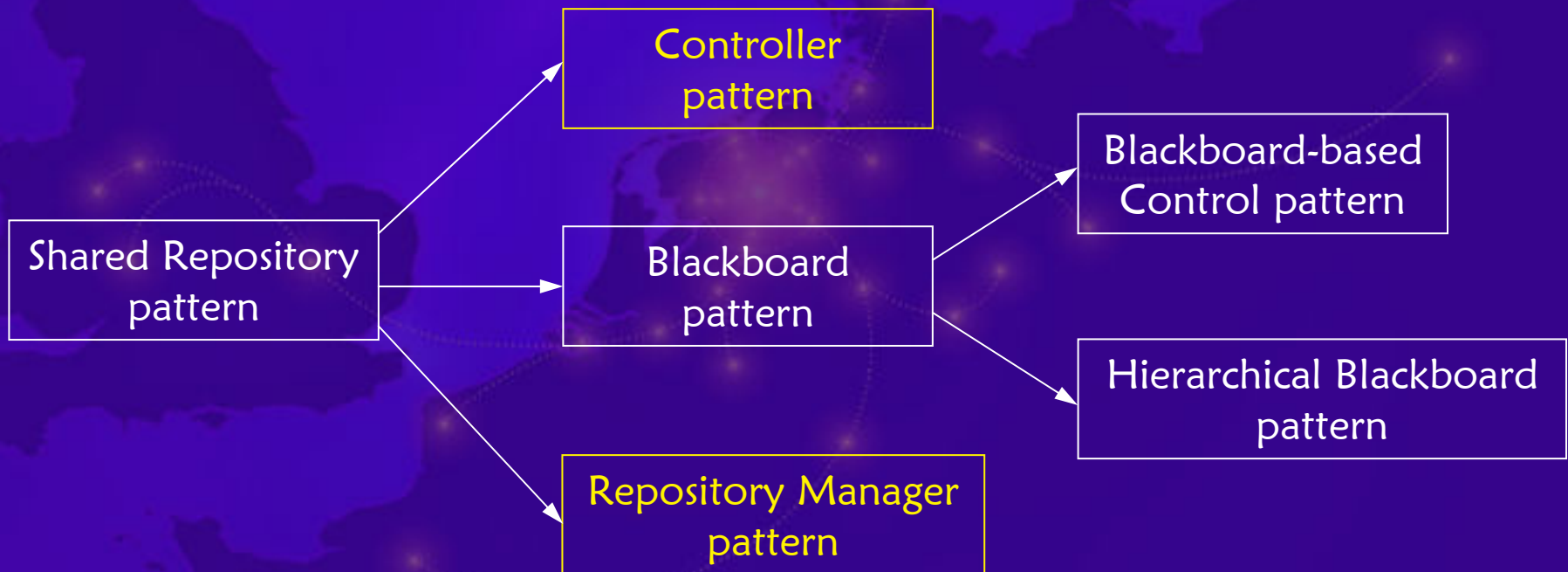


Blackboard Pattern

- Ideal for solving problems for which no predetermined algorithm is known.
But...
- “Best” Self-Calibration algorithm can be chosen from a relatively short list of strategies in advance
- The Shared Repository pattern is probably a better match

Shared Repository Pattern[†]

Specialization hierarchy of patterns based on the Shared Repository pattern



[†] Philippe Lalanda, in Proceedings of PLoP'98



Shared Repository Pattern

- Controller pattern
 - Introduces a control component in the system that rules the system and schedules activation of other components
 - Can be applied to deterministic problems where sequences of components activation can be determined off-line and coded in the controller
- Repository Manager pattern
 - Is applicable in a distributed environment
 - Introduces a repository manager that sends notification of data creation or modification to the software components
- Blackboard pattern
 - Refines the Controller pattern to deal with non-deterministic problems



Separation of Concerns

- Maximize scalability by complete separation of global and local controller
 - Global controller issues sequences of commands
 - Local controllers control the so-called “kernels” that execute the commands
 - A database system acts as a shared memory for storing commands and results

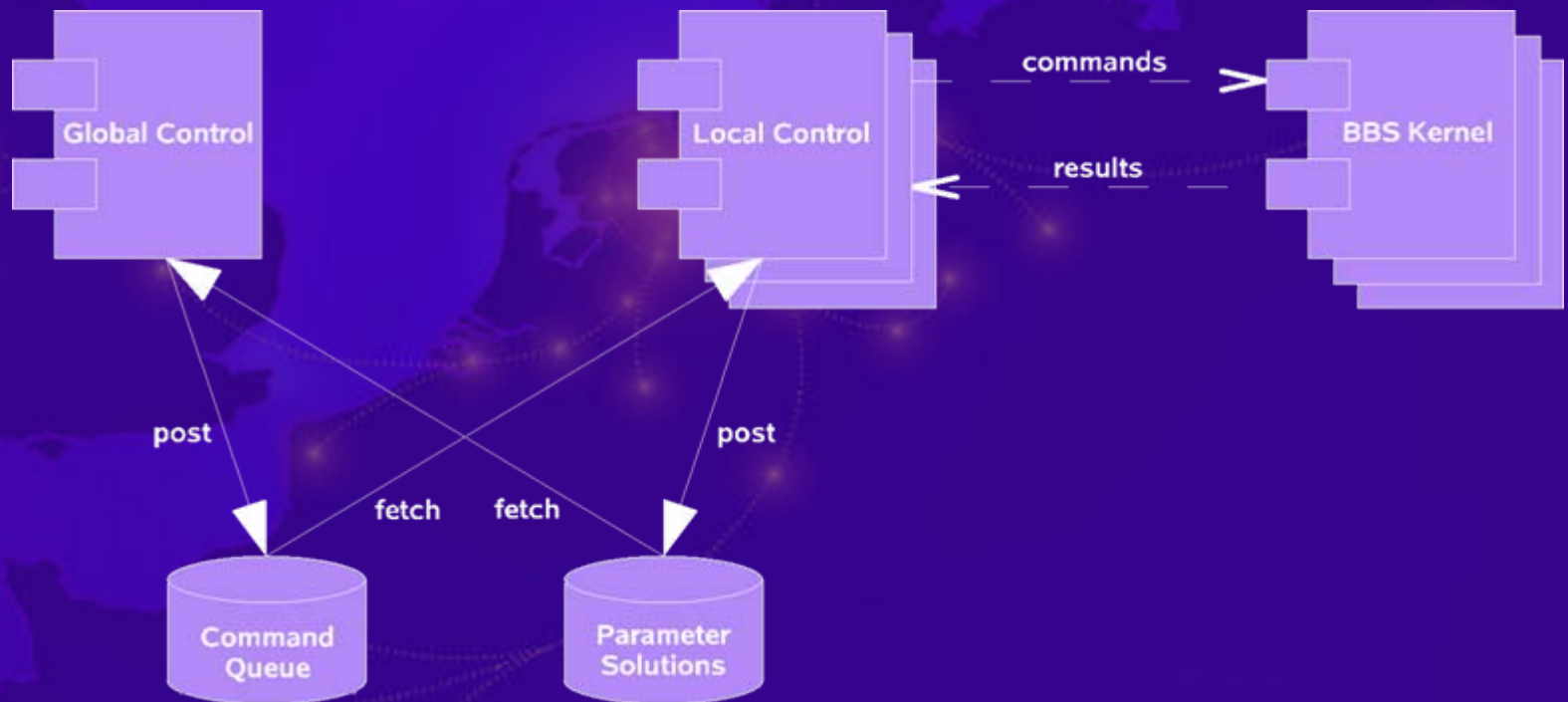


Blackboard Self-Cal System

- Subsystem overview
 - 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
 - BBS Database
 - Consists of two databases — Command Queue and Parameter Database — that together form the blackboard

BBS Control subsystem

- Global design of the BBS Control system





BBS Control subsystem

- Global Control
 - Acts as the main process
 - Posts one or more commands (steps) to the Command Queue
- Local Control
 - Controls a BBS Kernel subsystem
 - Fetches the next command from the Command Queue and forwards it to the BBS Kernel subsystem
- Command Queue
 - Stores the commands to be executed by the BBS Kernel and the status results returned by each kernel
- Parameter Solutions
 - Stores (intermediate) solutions of the model parameters calculated by the BBS Kernel.

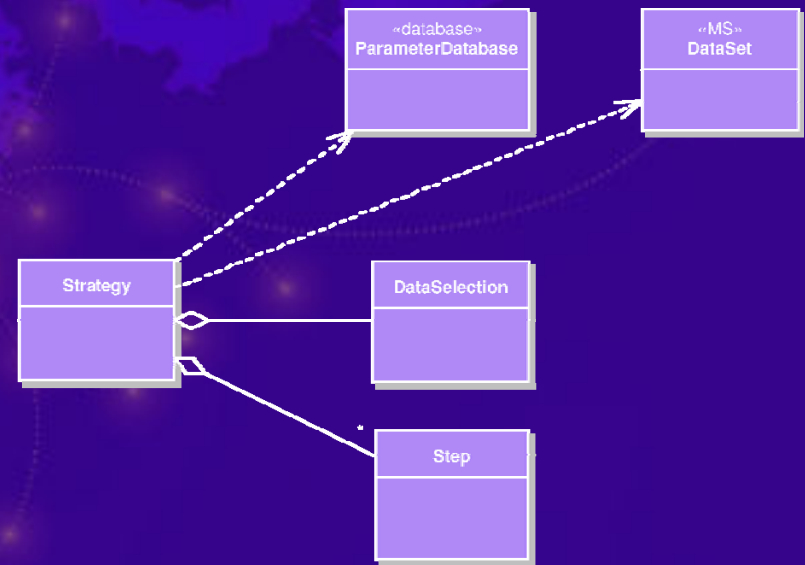


BBS Kernel subsystem

- Implements the Measurement Equation
- 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

Strategies and Steps

- Strategy
 - One iteration in the so-called *Major Cycle*[†]
 - Defines a relationship between the observed data and the model parameters.
 - Defines the size of a *work domain*
 - Consists of a number of steps

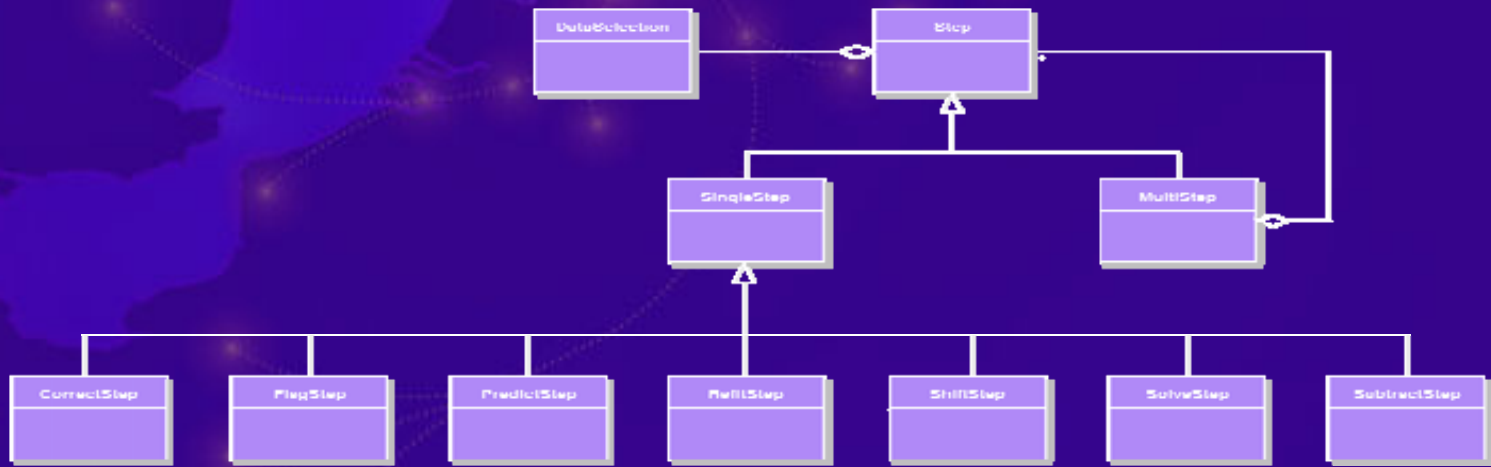


[†] J.E. Noordam, LOFAR Calibration Framework, ASTRON
ADASS 2007, GML

Strategies and Steps

■ BBS Step

- Is designed as a Composite pattern; each step can be made up of one or more (sub)steps
- Leaf classes define a single piece of work that can be handed over to the BBS Kernel subsystem





Strategies and Steps

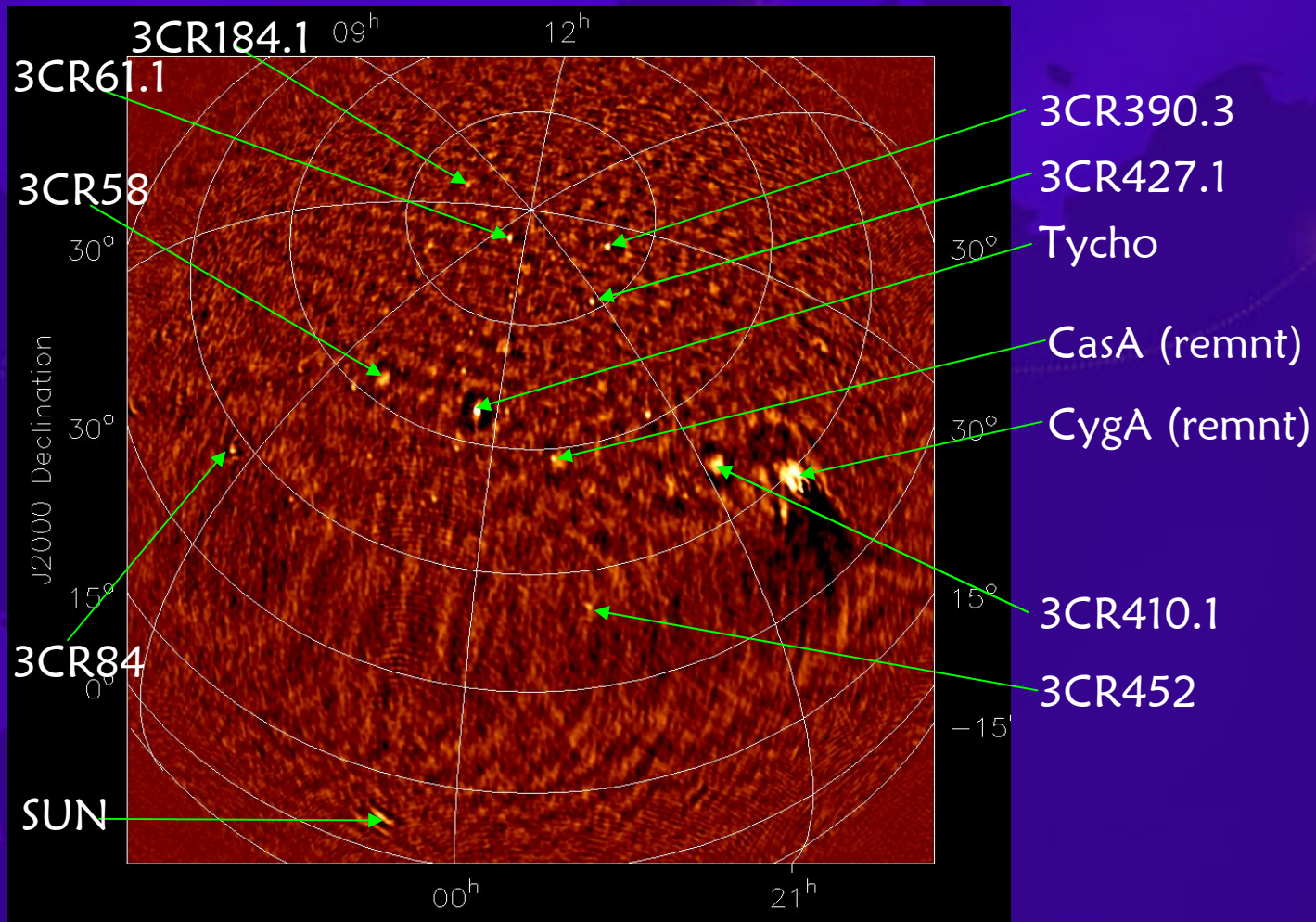
- Work domain
 - A subset of the data that fits in RAM memory
 - As many commands as possible are executed on these data before the next data chunk is accessed.
 - A strategy defines the size of the work domain (in time and frequency)



Current Status

- Control framework has been deployed on a 12 node cluster
- PostgreSQL database is used as shared repository
- First successful calibration runs on gigabyte-sized datasets

BBS Image





Future Work

- Test actual scalability of the control framework
- Add support to kernel for calibration of
 - Beam-shape
 - Ionospheric effects
 - Etc.



See Also

- ADASS Presentations
 - O4b.3: Distributed Processing of Future Radio Astronomical Observations
 - P9.9: LOFAR Core Station 1 post-processing and inspection tools
 - P9.15: Early LOFAR images with third generation calibration
- Thank you for your attention!