Computing and Software System Design Description: ONL and MCL

020.50.10.00.00-0002 DSN

Status: **RELEASED**

PREPARED BY	ORGANIZATION	SIGNATURE
R. Hiriart, Software Engineer	ngVLA, NRAO	Signed by: Rafael Hiriart DD5A92F18606426... 1/8/2026

APPROVALS	ORGANIZATION	SIGNATURES
R. Rosen, CSS IPT Lead	ngVLA, NRAO	Signed by: Rachel Rosen 4602AA3F0A7C428... 2/5/2026
Eric Murphy, Project Scientist	ngVLA, NRAO	Signed by: Eric Murphy 82D35C8655C14D1... 2/5/2026
J. Kern, AD, DMS	DMS, NRAO	Signed by: 014419F0FBC647B... 2/5/2026
R. Selina, Project Engineer	ngVLA, NRAO	Signed by: R. Selina 9740299D489B4DD... 2/5/2026
P. Kotzé, System Engineer	ngVLA, NRAO	Signed by: P.P.A. Kotzé 9E6E22E5C0FB448... 2/5/2026

RELEASED BY	ORGANIZATION	SIGNATURE
W. Hojnowski, Project Manager	ngVLA, NRAO	Signed by: William Hojnowski 722BEBEC6DE84DB... 2/5/2026

Change Record

Version	Date	Author	Affected Section(s)	Reason
I	2025-03-17	R. Hiriart	All	Initial draft
A	2025-08-13	M. Archuleta	All	Minor edits and formatting for pdf and release.
B	2026-01-06	R. Hiriart	All	Several modifications to complete CDR post-review actions.

Table of Contents

List of Figures

List of Tables

1 Introduction

This document describes the software architecture for the Online Data Acquisition system (ONL) and the Monitoring and Control system (MCL). These systems are part of the ngVLA Computing and Software System (CSS). The general architecture of the CSS system is shown in Figure 1.

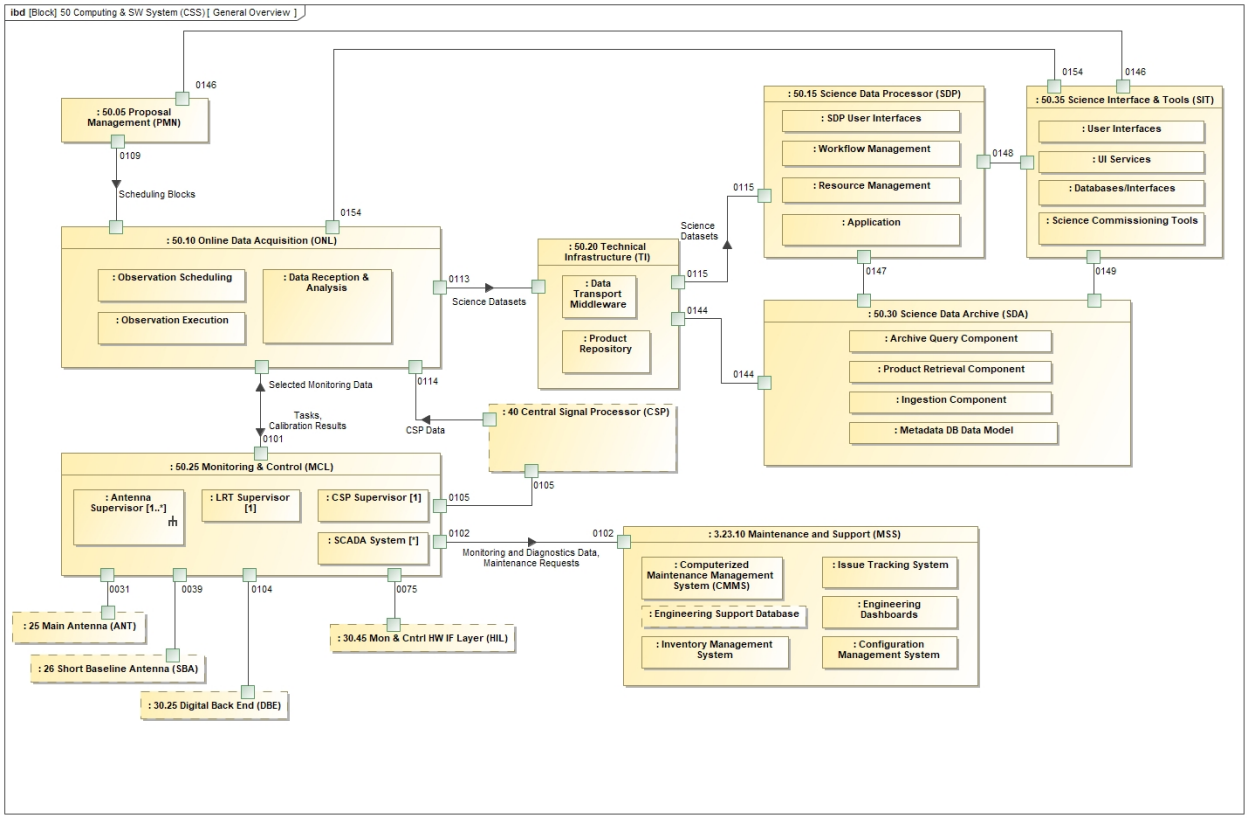

Figure 1. General overview of the ngVLA Computing and Software System (CSS). The numbers labeling each port identify interfaces that need to be formally specified by Interface Control Documents (ICD) and are kept under change control.

The ONL system is responsible for scheduling and managing the execution of astronomical observations, which are specified by Scheduling Block data structures. It receives science data from the Central Signal Processor (CSP), formats it, and forwards it to the Technical Infrastructure (TI) system. The TI system buffers and transmits the data to the Science Data Center, where it undergoes offline processing (SDP), archival (SDA), and subsequent end-user access and visualization (SIT). The ONL system processes CSP baseband data to perform real-time calibration and to detect radio-frequency interference (RFI).

The MCL system provides highly available supervisory monitoring and control services for the Array Operator. It acts as an intermediary between the ONL system and the underlying hardware, translating high-level directives into low-level control commands. Figure 2 shows an expanded view of the interfaces between the MCL system and hardware elements. The MCL system receives all hardware monitoring data

through these interfaces, applying condition monitoring algorithms and issuing preventive maintenance requests and diagnostic data to the Maintenance and Support System (MSS).

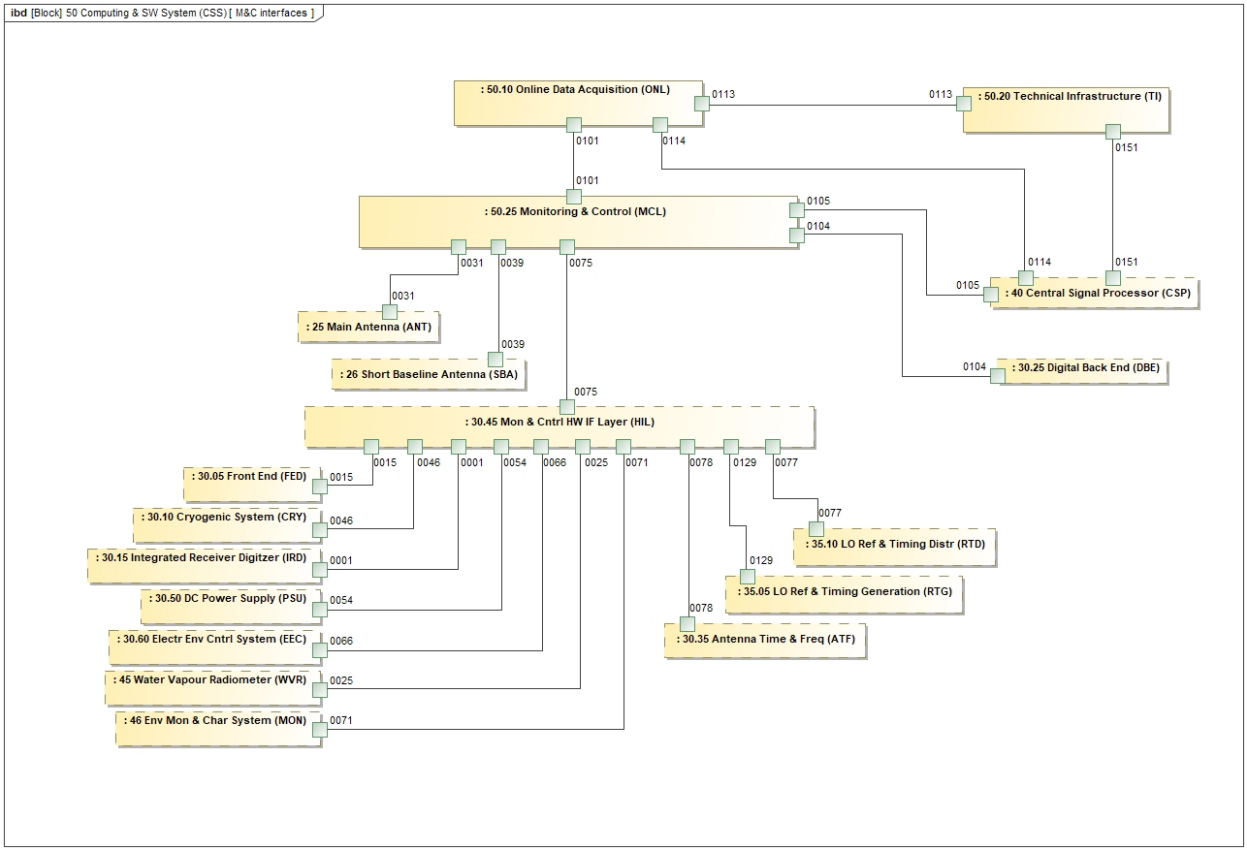

Figure 2. Monitoring and control interfaces. The numbers labeling each port identify interfaces that need to be formally specified by Interface Control Documents (ICD) and are kept under change control.

1.1 Requirement Management

The requirement management process for this part of the system follows the guidelines defined in the ngVLA Requirements Management Plan [RD01]. Consequently, a major source of software requirements is the System Requirement specification [RD02], derived from science requirements and stakeholder requirements and use cases. This is supplemented by several Lifecycle Concept and Use Case Specification documents written by the ngVLA Science Operations group [RD03-RD12], which cover in much more detail areas specifically related with the software systems. The derivation of requirements from these documents—managed by the CSS IPT—follows the recommendations defined in the INCOSE Guide to Needs and Requirements [RD13]. The overall process is shown in Figure 3. The Lifecycle Concepts were transformed in an Integrated Set of Needs [RD14-RD16], from which the Design Input Requirements and the Architecture & Design are derived.

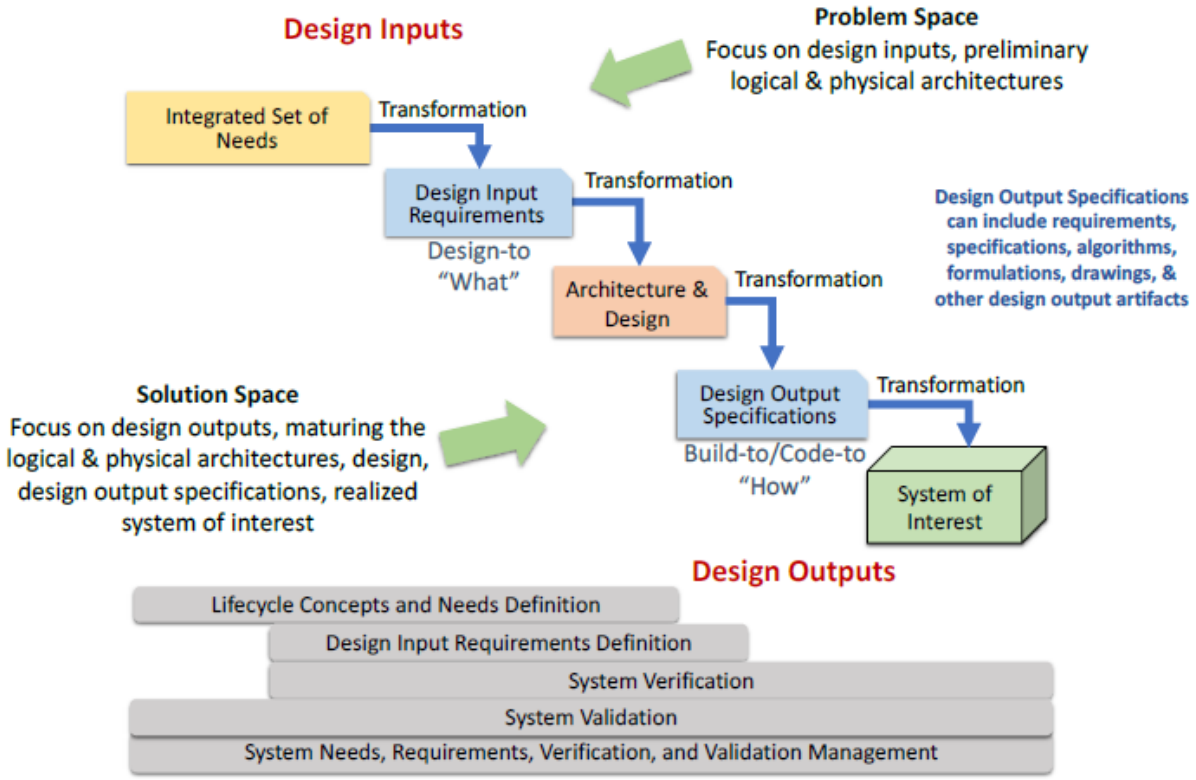

Figure 3. Needs and Design Input Requirements Development and Management (from INCOSE Guide to Needs and Requirements [RD13]).

For this iteration, the Needs were used directly to derive the Conceptual Architecture—skipping the transformation to Design Input Requirements, mostly because of insufficient manpower in the IPT to be able to follow the INCOSE process strictly, and because although further definition is still needed in some cases to be able to transform the Needs into well-formed Requirements, the Integrated Set of Needs is perceived at this point to be sufficiently defined to allow the definition of the Conceptual Architecture. The Design Input Requirements will be derived from the Integrated Set of Needs before the PDR. This is an important step to enable validation and verification activities and will become especially important as the Needs coming from GPA/USNO stakeholders are integrated. The Design Input Requirements conciliate differing and potentially conflicting Needs into an agreed-upon set of the design requirements.

1.2 Architecture Definition Process

The architecture is defined by following the Attribute Driven Design (ADD) process [RD17]. Figure 4 shows its steps and artifacts.

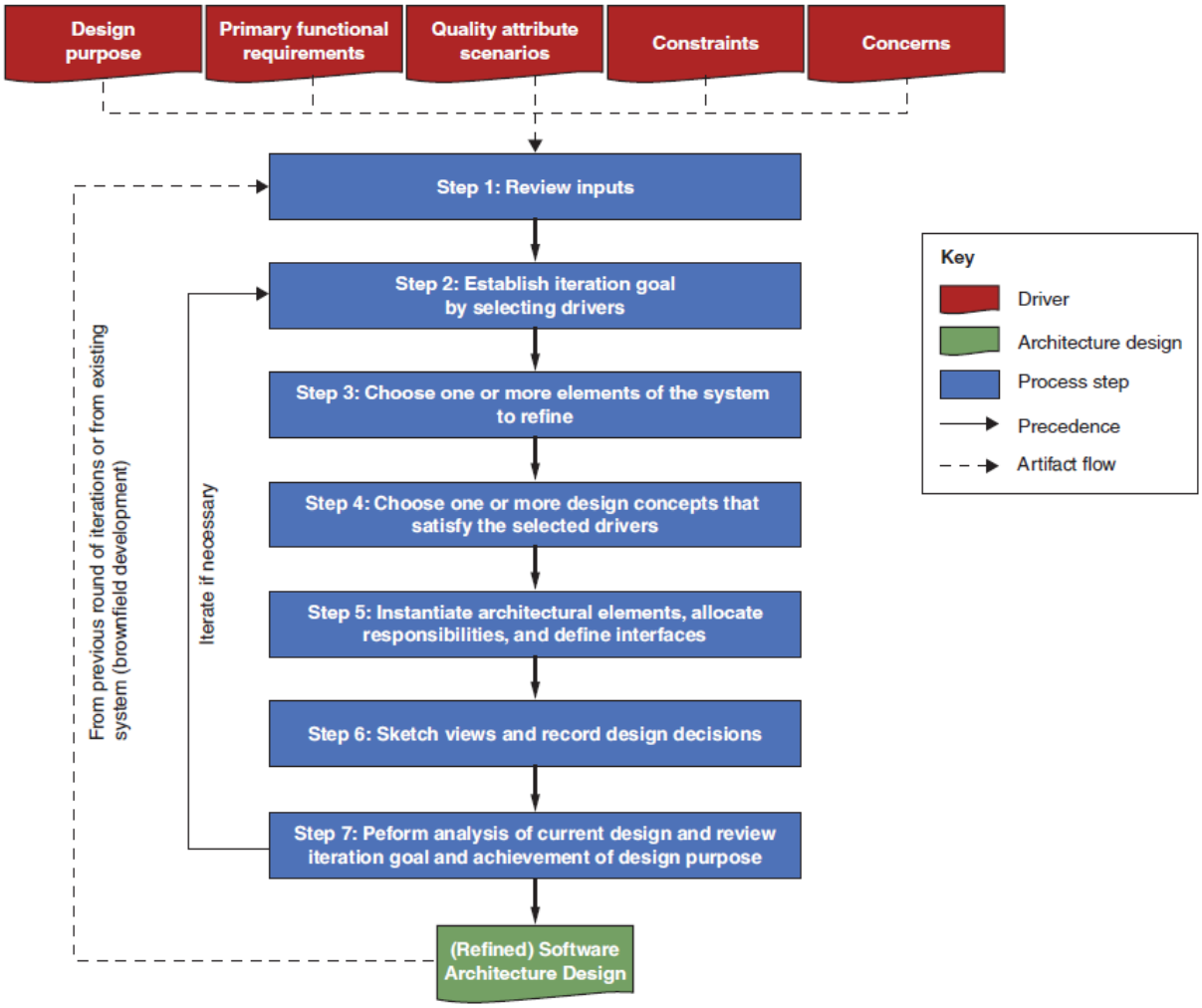

Figure 4. Steps and artifacts of the Attribute Design Process (from Software Architecture in Practice, 4th Edition [RD17]).

For this iteration, the design purpose is the definition of the Conceptual Architecture.

The Primary Functional Requirements are presented in section 5. The Quality Attribute Scenarios are presented in section 3. Other constraints and concerns are discussed in the text when relevant.

Design decisions are recorded in Architectural Definition Records (see [RD18]) which can be found in section 7.

The Software Architecture Design is maintained in a SysML/UML model in Cameo, along with the Requirement Specification database. This allows for the maintenance the traceability between requirements and software elements and specify in a detailed and unambiguous way key aspects of the

architecture. See the ngVLA Architecture Modeling Plan [RD19] for further information about the SysML model.

2 Reference Documents

Reference documents are any documents containing information complementing, explaining, detailing, or otherwise supporting the information included herein.

Ref. No.	Document Title	Rev./Doc. No.
RD01	Requirements Management Plan	020.10.15.00.00-0001 PLA
RD02	System Requirements	020.10.15.10.00-0003 REQ
RD03	Observation Preparation Concept	020.10.05.05.00-0010 PLA
RD04	ngVLA Observation Scheduling Concept	020.10.05.05.00-0012 PLA
RD05	ngVLA Observation Execution Concept	020.10.05.05.00-0013 PLA
RD06	Conceptual Narrative for Time Domain Science	020.10.05.05.00-0016 PLA
RD07	ngVLA Proposal Process Concept	020.10.05.05.00-0011 PLA
RD08	Telescope Support Concept	020.10.05.05.00-0008 PLA
RD09	Maintenance & Support Concept	020.10.05.05.00-0007 PLA
RD10	ngVLA Subarraying Operational Concept	020.10.05.05.00-0014 PLA
RD11	Scientific User Support and Outreach Concept	020.10.05.05.00-0009 PLA
RD12	ngVLA Calibration Concept	020.10.05.05.00-0015 PLA
RD13	Guide to Needs and Requirements	T. Katz, K. Orr and L. Wheatcraft
RD14	CSS Stakeholder Needs - PMN	020.50.00.00.01-007 REQ
RD15	CSS Stakeholder Needs - SDA, SDP, SIT, TI	020.50.00.00.01-0009 REQ
RD16	CSS Stakeholder Needs - ONL, MCL, MSS	020.50.00.00.01-0008 REQ
RD17	Software Architecture in Practice	L. Bass, P. Clements and R. Kazman
RD18	Software Architecture: The Hard Parts	N. Ford, M. Richards, P. Sadalage and Z. Dehghani
RD19	ngVLA Architecture Modeling Plan	020.10.20.00.00-0003 PLA
RD20	Building Event-Driven Microservices	A. Bellemare
RD21	ngVLA Subarraying Operational Concept	020.10.05.05.00-0014 PLA
RD22	Scheduling Algorithms	P. Brucker
RD23	ngVLA Multi-subarray Scheduling Algorithm	ngVLA Computing Memo #9
RD24	Scheduling preemptive multiprocessor tasks on dedicated processors	L. Bianco, J. Blazewicz, P. Dell'Olmo and M. Drozdowski. *Performance Evaluation*, no. 20, pp. 361-371, 1994.
RD25	Scheduling, Theory, Algorithms and Systems	M. Pinedo
RD26	ngVLA Data Rates and Computational Loads (Update)	R. Hiriart
RD27	RFI Mitigation for the ngVLA: A Cost-Benefit Analysis	ngVLA Memo #70

Ref. No.	Document Title	Rev./Doc. No.
RD28	Size-of-Computing Estimates for ngVLA Synthesis Imaging	ngVLA Computing Memo #4
RD29	ngVLA Calibration Concept	020.10.05.05.00-0015 PLA
RD30	Computing and Software System: Glossary	020.50.00.00.00-0008 LIS

3 Quality Attribute Scenarios

Table 1 outlines the key quality attribute scenarios relevant to the ONL and MCL systems. A quality attribute (QA) refers to a measurable or testable characteristic of a system that reflects how effectively it meets stakeholder expectations beyond its basic functionality [RD01]. While functional requirements specify *what* a system must do, they do not dictate its architectural design. Instead, quality requirements—also referred to as non-functional requirements and typically expressed through quality attribute scenarios—play a critical role in shaping the system's architecture. Indeed, for any given set of required functionalities, numerous architectural configurations may satisfy them. The selection of the most appropriate architecture is therefore guided by how well each alternative supports the desired quality attributes.

The Architectural Value and Technical Risk column estimates these metrics using the following scale:

Architectural Value

- High: Must-have requirement
- Medium: Important but would not lead to project failure if omitted
- Low: Nice to have but not worth much effort

Technical Risk

- High: Meeting the requirement is perceived to be challenging
- Medium: Concerning but not carry high risk
- Low: It can be accomplished confidently

Requirements which are high in either metric should be given priority in the architecture design and development.

The "Design Concepts" column identifies tactics, reference architectures, or architectural patterns employed to address each requirement. These concepts are further elaborated upon in the subsequent sections.

Other important architecturally significant requirements (ASRs) derived from the Conceptual Lifecycle documents and system requirements are listed in Section 5. Given that these requirements are mostly functional, they do not drive the definition of the architecture, but they are included for the purpose of demonstrating compliance.

Table 1. Quality attribute scenarios.

ID	Name	Text	Arch. Value / Tech. Risk	Design Concepts
QAS001	Availability - Antenna hardware failure recovery	While executing an observation, an antenna fails due to a hardware error. The system detects the problem, informs the operator and flags the data. The operator troubleshoots the problem, recovers the antenna, and re-incorporates it in the observation.	H/M	The following tactics support fault-tolerance: - Hardware errors will be normally detected at the supervisor level - Supervisors communicate the fault to the Executor, who keeps a state machine of Tasks and their execution status - Scheduling keeps a state machine of CSBs and their execution status - Fault-tolerance strategies can be implemented at the Task level in the Executor or at the CSB level in the Scheduler - The system is flexible to allow the re-incorporation of antennas into an observing subarray – by dynamically connecting services to the corresponding streams. - Services involved in the observation execution are stateless, or are stateful only within a CSB execution, allowing the system to seamlessly repeat sets of Tasks or CSBs in case of failures.
QAS002	Availability - No data loss and recovery from H/W or S/W faults	While executing an observation, a hardware or software error brings down one of the processes involved in executing the observation. All the observation data up to the last successfully executed CSB is preserved automatically. Depending on the type of failure, the system is recovered automatically or manually (e.g. the system could restart the dead process automatically on a different server, or the operator	H/M	Besides the fault-tolerance tactics documented in QAS001, the following tactics allow the system to support this requirement: - All the metadata that is transmitted through the Event Broker is saved with replication. - The CSP data is formatted and immediately stored to disk. If a Receiver or Formatter fails, then the CSB can be repeated by the Scheduler, with the container orchestrator starting the required services in a different cluster node.

ID	Name	Text	Arch. Value / Tech. Risk	Design Concepts
		could decide to switch to a previous software version for this subarray), and the observation can be resumed from the last successfully executed CSB. Excluding the time that it takes the operator to detect the problem and initiate the follow-up action on the system, it should be possible to resume the observation in less than 10 (TBC) minutes.		- The system supports different software versions by containerization.
QAS003	Availability - Loss of connectivity with antennas	The system should be able to tolerate degraded M&C communications between the central systems and the antennas, to the extent that this is possible. This is particularly important for long-baseline antennas, which would be connected through ISP networks that present a higher degree of latency variability. For example, temporary problems that occur while the system is executing a CSB (but not configuring and initiating a CSB) should be handled robustly.	H/L	The Executor configures the execution of a CSB by enqueuing timestamped Tasks into the Antenna Supervisor. The Antenna Supervisor executes the queue independently of the Executor and tolerates temporary loss or degraded connectivity. The enqueueing of CSB Tasks is retried several times in case of communication failures.
QAS004	Availability - Initialization latency	After a full system shutdown and restart, the system shall reach the operational state in less than 10 minutes.	M/M	The architecture avoids any centralized services that can become a performance bottleneck during startup. For example, configuration data can be replicated locally to nodes by means of a distributed configuration system.
QAS005	Availability - M&C fault tolerance	A server in the monitoring and control system fails during normal operation, the system informs the	M/M	Level 2 services are static and designed to be either stateless or their stateful operations to be idempotent (e.g. repeated pointing commands to same target cause no

ID	Name	Text	Arch. Value / Tech. Risk	Design Concepts
		operator and continues to operate with no downtime.		effect). Communication between Level 3 and Level 2 is asynchronous, allowing the system to restart Level 2 services and continue operations.
QAS006	Availability - Subarray fault isolation	After a hardware or software failure in the components involved with the operation of a subarray, other subarrays are unaffected.	H/M	All Level 3 services involved in the operation of a subarray are dynamic and completely independent from other subarrays. Level 2 services are static but independent from each other.
QAS007	Deployability - Support for continuous deployment	A new software feature or bug fix is implemented, tested and committed by a software developer. The change is available to be deployed and tested in a test subarray immediately. The rest of the system, working still with the previous stable version(s), is unaffected.	H/M	This requirement is supported by - Dynamic and independent Level-3 subarray services - Use of containerized services - Most Level-2 services are independent and can be deployed with different instances supporting different versions - OPC UA can support multiple versions at the API level.
QAS008	Interoperability - Incorporation of third-party S/W components	The system should allow the integration of third-party components in architectural areas where these components are available and are compliant with ngVLA's requirements.	M/L	The architecture is based on open standards such as OPC UA and MQTT.
QAS009	Modifiability - Simple alarm management modification	A trained Array Operator is able to introduce a simple change in alarm management system (e.g. changing the alarm priority or a new alarm rule), test and deploy the modification in 15 minutes, without having to bring down the system or affect other array operations.	M/L	Commercial SCADA system such as Ignition supports this requirement. The key characteristics is that these systems support changes by configuration, without requiring code modifications.
QAS010	Modifiability - Observing	A Telescope Support Scientist is able to modify an observing mode	H/M	Observing modes are supported by - Heuristics to break down an SB

ID	Name	Text	Arch. Value / Tech. Risk	Design Concepts
	mode modification	(e.g. changing the heuristics to select a calibrator, or the sequence of scans), testing the changes in a simulated environment, and when ready deploying and testing the modifications in a test subarray (or instructing the Array Operator to do so) without affecting other array operations.		into a set of CSBs at the observation preparation level - Calibration management at the Scheduling level - Task translation in the Executor. These services can be made easy to modify by allowing these operations to be modified by configuration, for example introducing tactics such as scripting and domain-specific languages. The architecture supports these tactics by separating science-specific functions from other technical functions, as shown in the Executor design.
QAS011	Observability - Collection of troubleshooting data	A Telescope Support Engineer or Scientist should be able to gather all the data pertinent to troubleshoot a reported problem in 10 minutes. This includes operator logs, system logs, relevant monitoring data and science data.	M/L	All logs, tasks, monitoring data, flag commands, and other operational information pass through the Event Broker, where they are automatically retained for a configurable period. For long-term storage, this data is stored in the Engineering/Monitoring database.
QAS012	Performance - Scheduling algorithm	The Array Operations Group executes the scheduling algorithm with a time horizon of one week, and the scheduler provides the results in less than one hour.	H/H	This requirement is supported by choosing appropriate algorithms and parallelizing their implementation. For example, graph algorithms can be parallelized using frameworks such as Apache Spark.
QAS013	Performance - Closed loop complex gain calibration latency	The TelCal subsystem receives the data from a complex gain calibrator scan. The solutions are determined and applied in the phasing system in less than 3 seconds.	H/M	The data streams from the CSP are partitioned by spectra (e.g. by subbands). Each stream is received and processed in parallel by independent Spectral Reception and Processing Units. Latency is optimized by reducing the amount of data copying, using RDMA and shared memory.

ID	Name	Text	Arch. Value / Tech. Risk	Design Concepts
QAS014	Performance - Data reception and formatting throughput	The Data Reception and Analysis system is able to receive, format and send raw data products to the Data Transport subsystem with a throughput of 1056 Gbps.	H/H	The data streams from the CSP are partitioned by spectra (e.g. by subbands). Each stream is received, formatted and buffered in parallel by independent Spectral Reception and Processing Units. Storing to the buffer parallel filesystem is optimized by means of Parallel IO.
QAS015	Security - Message eavesdropping	An attacker has been able to compromise the system and attempts to eavesdrop the messages flowing through the network (for example, to gain insight for a subsequent attack, or access confidential metadata). However, network messages are protected, preventing the attacker to gather confidential information.	M/L	- The architecture establishes a DMZ between Level 4 and Level 3 and for the long baseline antennas between Level 3 and Level 2. No direct communication is allowed to pass through the DMZ. - Event Broker event messages are encrypted. - OPC UA commands are encrypted.
QAS016	Security - Unauthorized access to internal control network	An external attacker gains access to the system exploiting vulnerabilities on Internet-facing systems. He attempts to access the internal control network but it is prevented from doing so. The unauthorized access is detected and the system maintains an audit trail.	H/M	- The DMZ prevents unauthorized access between network segmentation zones. - An Intrusion Detection System (IDS) monitors the network and detects the unauthorized attempt. - The system keeps several logs and audit trails that document the threat.
QAS017	Security - Message spoofing, alteration or replay	An attacker has gained access to the internal control network and attempts to send an unauthorized control messages (e.g. by message spoofing, alteration or replay). The system detects and prevents the attack, informs security personnel and writes details of the attack in an audit trail log.	M/M	- OPC UA prevents message spoofing by providing the ability to sign messages. - Additionally, OPC UA messages always contain a valid SessionId, SecureChannelId, RequestId, Timestamp and the correct sequence number. - OPC UA PubSub messages counters message spoofing threats

ID	Name	Text	Arch. Value / Tech. Risk	Design Concepts
				by providing the ability to sign messages. - The system keeps several logs and audit trails that document the threat.
QAS018	Security - Database attack	An attacker gain access to the system and attempts to corrupt or destroy system databases (e.g. Scheduling Block database, Calibrator database). The system detects the attack, keeps an audit trail, and the data is restored within one day.	H/M	- Database are secured with defense-in-depth tactics such as physical security, network segmentation, use of firewalls, access control lists, application security and end-user authentication and authorization. - Databases are periodically backed up with backups stored in secure locations. - An Intrusion Detection System (IDS) monitors the network and detects the unauthorized attempt. - The system keeps several logs and audit trails that document the threat.
QAS019	Security - Rogue server or publisher	An attacker builds a malicious server (e.g. an OPC UA server) or installs an unauthorized instance of a genuine server in the system. The system prevents the application of commands coming from the unauthorized server and keeps confidential information protected. The system detects the attack and keeps an audit trail.	M/M	- OPC UA client applications counters the threat of rogue servers by validating Server Application Instance Certificates and the user of private keys to encrypt and sign messages. - OPC UA subscriber applications counter the effect of rogue Publisher by validating the signature on the published messages.

In summary, the architecture main quality drivers are availability, operational efficiency, performance (high data rates for data reception from the CSP and TelCal calculations; scheduling algorithm latency) and cyber-security.

Performance is not expected to present major challenges in other scenarios like the subarray re-configuration or observation execution. For these, there may be sizable latencies, but they are expected to be mostly in the hardware. The main requirement for the software system is the capacity to parallelize these operations, e.g. hiding latencies by configuring the system while the antenna is slewing. This is

considered as part of the design of the Executor. Latencies involved in the CSP subarray reconfiguration may be of some concern, so its interface with the ONL system considers the possibility of performing some of the operations required for the next scan while the current scan is still running. For these scenarios, there may be some tradeoffs with security that should be considered (tradeoffs usually occur between security and performance), but these are not expected to be significant as real-time control is constrained to Level 1. The DMZ doesn't introduce overheads in these scenarios, given that once a CSB is submitted to be executed from Scheduling, all the interactions involved are constrained to Levels 1-3.

The availability scenarios, in particular QAS001 and QAS002, are quite complex and will be specified in detail for the PDR. At the conceptual level, these quality attributes are mostly expressing the need for the software to be able to detect and tolerate failures and allow the recovery and re-incorporation of antennas during an observation. The following points are relevant:

- Detection is not particularly difficult but occurs at different levels. The Antenna Supervisor can check certain conditions, receive alarms, or receive failure responses to OPC UA operations. TelCal can generate QA metrics that can also detect problems, like larger than usual antenna pointing offsets or large phase standard deviation.
- Scheduling and Observation Executor keep an observation state machine that keeps track of the status of each CSB execution and the observation in general.
- Automatic responses could be triggered in cases where it is possible, at the level of the Supervisor, Executor, or Scheduling.
- Events are received by the Operator in the Alarm Panels in the SCADA UI, so he can try to recover the antenna manually.
- Upon antenna failures, if the observation still has enough sensitivity with the remaining antennas, the observation continues, otherwise it is aborted.

Having an antenna in a subarray means that it will be listening to hardware commands and try to execute them, unless the type of failure is of such nature that the Antenna Supervisor decides to stop executing the command queue right away. Normally, if the Operator needs to apply recovery procedures, it will stop the Antenna Supervisor execution of commands, that is, the antenna is no longer "in" the subarray.

Provided that the antenna was recovered, it is then re-incorporated in the observation, so it will start executing commands in the next CSB. One problem is that any software component that crashed and was restarted may lose its software state in the process. This state could involve metadata elements distributed at the beginning of the observation, or references to other components that need to be re-acquired. If the components are mostly stateless, this becomes easier. The Event Broker also has an event "re-playability" capacity, which can be useful if it needs to recover state variables. Using containerization and a container orchestrator such as Kubernetes allows to automatically restart a software component that crashed, but as explained above, any state needs to be recovered before the component can resume its operations.

A table collecting the main scenarios and design concepts for each one of them can be found in Table 10, section 11.

4 General Description of Subsystem

A general view of the architecture of the ONL and MCL systems is shown in Figure 5. The architecture adopts the Purdue model to address network security requirements (see section ADR001: Adopt the Purdue Model for network security). The Purdue model defines 5 layers, which are adapted slightly to the ngVLA system architecture and observatory operations:

- **Level 0 Physical Process:** The array hardware elements. Includes the devices that receive, process and transmit the sky signal (Antenna Receiver Front End, Integrated Receiver Digitizer (IRD), Digital Back End (DBE), Central Signal Processor (CSP)), and devices that generate and distribute Local Oscillator (LO) reference signals and time synchronization signals.
- **Level 1 Basic Control:** Devices responsible for sensing and manipulating physical processes. Most of the antenna electronic devices are controlled by means of Hardware Interface Layer (HIL) Boards, except for the Antenna Control Unit (ACU) and the DBEs, which incorporate their own control interfaces. The CSP is controlled by the CSP Monitor and Control (CMC) system and LO Reference and Timing devices are controlled by means of HIL boards.
- **Level 2 Supervisory Control:** Control systems that supervise, monitor and control the physical process. Includes systems that collect, process and archive monitoring data.
- **Level 3 Observation Operations:** Systems involved in managing the observation workflow to produce the raw data products. Includes the Dynamic Scheduler, which produce observation schedules for short timeframes (few next hours).
- **Level 4 Planning and Logistics:** Systems involved in planning and supporting the observatory operations. Includes the Long-Term Scheduler, which produce observation long-timeframe schedules for planning purposes (mid-term: 1-2 weeks, long-term: observing season).
- **Level 5 Observatory Network:** The outermost network layer, providing public access to services such as the Proposal Management System (PMN). Levels 4 and 5 communicate through the observatory IT network.

In the Purdue model, the lower layers are assumed to present higher risks if their assets were to be compromised, hence the network is segmented to progressively restrict the connections that are allowed between upper and lower layers. An additional level is added between levels 4 and 3 to separate the IT network from the control system network (a.k.a. Operational Technology (OT) network). This constitutes level 3.5, the Industrial Demilitarized Zone (DMZ). No direct traffic is allowed between levels 4 and 3. Instead, the data that needs to be exchanged does so through secured proxy servers or databases. The DMZ usually includes a firewall to restrict the traffic between levels 4 and 3.5, a firewall to restrict the traffic between levels 3.5 and 3, and a switch to connect the applications that are deployed in the DMZ. For ngVLA, these include:

- Patch Management service.
- Virtual Private Network (VPN) server or Remote Access Server (RAS).
- Intrusion Detection System.

- The Engineering Support Database, which collects monitoring data and logs from the system for engineering support and troubleshooting.

Long baseline antennas are connected to the Central Electronics Building through the public Internet, so a DMZ zone is established between levels 2 and 3. Additional components are deployed in the Antenna DMZ:

- The Antenna Proxy, a simple bi-directional queue application to establish the connection between the Central Electronics Supervisory Control and the Antenna Supervisory Control.
- The Monitoring Broker, which collects Pub/Sub monitoring messages from level 1 and transmits them to the Central Electronics Supervisory Control for processing, distribution and archival.

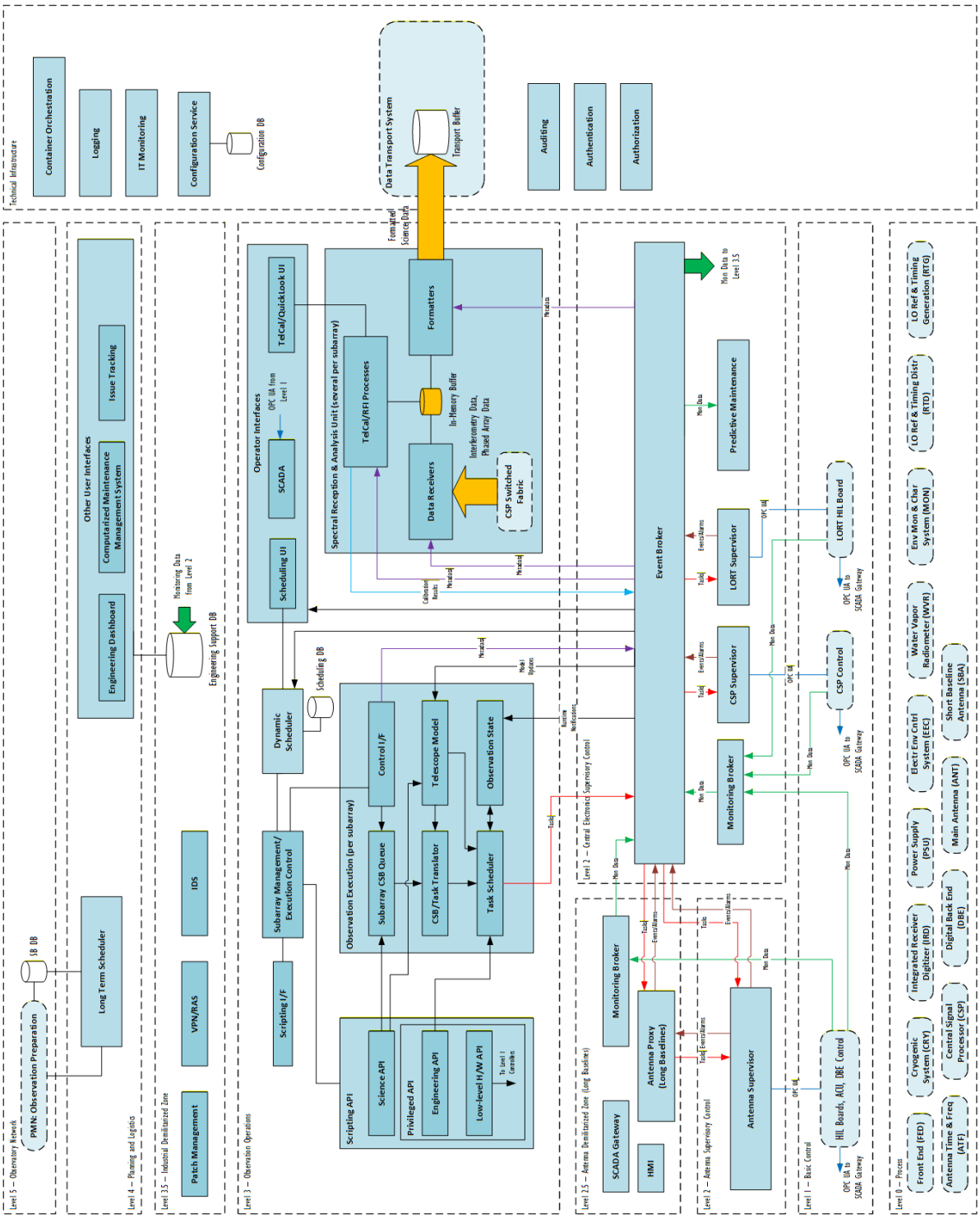

Figure 5. General ONL and MCL architecture. Rectangular boxes represent components that belong to the ONL system, while rounded segmented boxes represent external components.

External systems to ONL and MCL are assumed to be in level 5 - Observatory Network. Of particular importance is the Proposal Management System (PMN), which handles the Telescope Time Allocation process and the generation of Scheduling Blocks.

Level 4 includes services that support observation planning and maintenance operations. Observation planning is supported by the creation of statistical forecasting models (e.g. pressure plots) and the execution of the observation scheduling algorithm on medium and long timeframes. The Engineering Dashboard provides a user interfaces for engineering support and commissioning staff, integrating data from several sources (e.g. the Telescope Configuration DB, Engineering Support DB) and providing visualization, querying and scripting capabilities. Level 4 includes interfaces for the Maintenance and Support System (MSS), including the Inventory Management System, Computerized Maintenance Management System, Issue Tracking System, etc. These will probably be third-party tools.

The main functions of Level 3 – Observation Operations are selecting and executing SBs, receive data from the CSP Switched Fabric and provide interfaces for the Operator. The Dynamic Scheduler produces short-term schedules and controls their execution through the Subarray Management service. It receives pertinent runtime data such as current weather conditions and hardware status from the Event Broker. Several user interfaces (UI) are provided to the operator, including the SCADA UI, a Scheduling UI and TelCal/Quick-Look UI. The SCADA allows the Operator to supervise the health of the array, receive events and alarms, troubleshoot and recover from faults. The SCADA system normally interacts with the supervisory components in Level 2 (these can be considered *part* of a distributed SCADA system) but should also provide direct access to level 1 controllers. Allowing Level 3 to connect directly with Level 1 – an exception to the Purdue model – raises some security concerns that will be evaluated during the design phase. Other security controls, such as the application of the Zero Trust model, communication encryption and the application of the OPC UA security model can be used in combination to the Purdue model to make this interaction secure. Another vulnerability risk that needs to be carefully designed is the integration of a VPN or Remote Access Server for engineering support in the DMZ.

The Observation Execution service orchestrates the execution of the scheduled observations for a given subarray. As the execution of the schedule proceeds and subarrays are created and destroyed, corresponding instances of Level 3 services are created and destroyed as well. The observation execution services in level 3 are dynamic and tied to a given subarray. From the perspective of the ONL software, a subarray provides an execution context, which includes several instances of containerized services dynamically connected to other services by means of topic streams in the Event Broker. Note that the Event Broker in Level 2 is not instantiated per subarray but is static instead. There is only one, which handles different streams for each subarray. The observation execution workflows for each subarray is orchestrated by an instance of the Observation Execution service, which communicates with other services by means of asynchronous messages, with data and metadata collected with eventual consistency (see ADR004: Observation Execution Communication, Consistency and Coordination). No atomic transactions are required, as the final dataset is consistent when it is eventually sent to the Data Transport System. The metadata is distributed before the CSP data arrives, so in strict rigor there is a small interval of time where the received data may be inconsistent. However, as no services queries this data during this interval, this temporary inconsistency is acceptable. (Otherwise, a different approach would be needed

to ensure transactional consistency.) The use of streams in the Event Broker to connect the services belonging to a subarray execution context provides a flexible, fault-tolerant and scalable way of conducting observations. The use of containerized services (or microservices) allows for flexible deployment strategies, e.g. allowing different versions of the software to be executed in different subarrays.

The CSP outputs interferometry and phased array data through the CSP Switched Fabric. Depending on the subarray configuration and scan parameters, the data rate can be as high as 1056 Gbps (CON104). Several Data Receivers are instantiated to receive these data streams in nodes physically connected to the CSP Switched Fabric through 400 Gbps network links. These receivers allocate memory buffers and handle protocol handshaking functions. The data transmission is performed through Remote Direct Memory Access (RDMA), a high-performance data transmission protocol that can achieve very low latencies, due to the NICs directly writing the data into RAM or GPU memory, bypassing the CPU. Several Formatter threads or processes read from these memory buffers and format the data in their final science dataset format, merging the data from the CSP with the metadata received from other services. The data format is being defined by the XRADIO project and will be likely based on HDF5 or NetCDF. Both formats combine data and metadata in the same binary files and support Parallel I/O.

As data is received and buffered in-memory, it is read by telescope calibration (TelCal) and RFI detection processes, which compute and broadcast results to the Event Broker. Many of these processes (e.g. the complex gain and delay solvers, a.k.a. *antsol*) are good candidates to be implemented in GPUs. Starting instances of the Data Receiver, Formatter and Telcal processes, with the correct version of the software, is handled by the Subarray Management service. These processes receive metadata and other control messages from the Observation Execution services through the Event Broker.

The main services in level 2 are the Event Broker, the Supervisors (Antenna Supervisor, CSP Supervisor and LORT Supervisor) and the Monitoring Broker. Services in this level constitute the MCL system. Contrary to Level 3 services, Level 2 services are static: they are always running and are statically connected to Level 1 controllers through OPC UA. These services run with high availability, are containerized and are managed by a Container Orchestrator (e.g. Kubernetes). If one of these services goes down, it is automatically restarted by the Orchestrator, possibly in a different machine. Containerization allows for the deployment of more than one instance of these services, each with a different software version. However, only one of these instances is connected to the corresponding Level 1 device. OPC UA locking mechanisms can be used to ensure this.

The Event Broker (e.g. Kafka) receives events, stores them in a partitioned event stream (append only, similar to a database commit log), and provides them for consumption by other processes. The Event Broker architecture follows a model where multiple, distributed brokers work collaboratively in a cluster to provide several capabilities [RD20]:

- *Scalability.* Additional event broker instances can be added to increase the cluster's production, consumption, and data storage capacity.
- *Durability.* Event data is replicated between nodes. This permits a cluster of brokers to both preserve and continue serving data when a broker fails.
- *High availability.* A cluster of event broker nodes enables clients to connect to other nodes in the case of broker failure. This permits the clients to maintain full uptime.

- *High-performance.* Multiple broker nodes share the production and consumption load. In addition, each broker node must be highly performant to be able to handle hundreds of thousands of writes or reads per second.

Several user interfaces are provided for the Array Operator(s):

- Scheduling UI: This interface allows the Operator to manage subarrays and schedule observation executions on them. These operations can be performed automatically, manually, or in a semi-automatic mode where the system suggests actions that need to be accepted by the Operator. The interface presents to the Operator a view of the status of each subarray and their observations.
- Supervisory Control and Data Acquisition UI: This UI integrates a system of hierarchical views where the Operator can visualize the status of the array's hardware systems in different levels of detail. It includes interfaces to manage alarms, events and logs. This may be provided by third-party software such as a commercial industrial SCADA system. (Their integration is made possible by the selection of a standard protocol such as OPC UA.)
- Telcal and Quicklook UI: This interface allows the Array Operator and the Scientist On-Duty to visualize calibration results and other metrics necessary for observation quality control.

These interfaces should be customizable and extensible, allowing the users to adapt them to better fit their needs. They should also be highly secure, integrating role-based access control.

4.1 Monitoring and Control

Monitoring and Control functions are allocated in Level 2 and Level 1. Level 2 implements supervisory control, while real-time control loops are implemented in Level 1. Level 2 communicates with Level 1 through the Open Platform Communications Unified Architecture (OPC UA) protocol. This is an open, platform-independent, service-oriented, flexible, and secure communication protocol for industrial automation. By implementing the M&C system on top of this standard, third party components—such as SCADA systems and data streaming solutions—can be potentially integrated, allowing development resources to concentrate on domain specific functionality. This is particularly important for ngVLA given the anticipated tight development schedule. It is important, on the other hand, to recognize the potential pitfalls of integrating third-part components (e.g. complexity, loss of design control, challenging integration, risk of dropped support in the future, risk of non-backward compatible changes). These will be evaluated case-by-case during the design stage of the project.

One of the key strengths of the OPC UA protocol is the independence of the core OPC UA services from the underlying data transport mechanism and encoding. OPC UA supports both client/server connection-oriented communication protocols such as TCP or HTTPS, and publisher/subscriber connectionless protocols such as UDP, AMQP or MQTT. Payloads can be encoded either in a binary format tailored for performance, or commonly used formats such as JSON or XML. This flexibility allows OPC UA to adapt to different infrastructure scenarios, e.g. directly connected antennas versus ISP-connected long-baseline antennas. Furthermore, this capability offers a pathway for addressing the possibility of future technological obsolescence.

The main parts of the M&C system are:

1. OPC UA servers: These are Level 1 servers that implement the OPC UA interfaces.
2. Supervisor services: These components supervise the OPC UA servers, establishing client/server connections. They translate high-level Task commands into hardware-specific OPC UA commands. They oversee the successful completion of these commands, sending alarms or notifications in case of faults.
3. Supervisory Control and Data Acquisition (SCADA) system: This system connects with the OPC UA servers and provides the Array Operator with user interfaces to visualize the state of the hardware systems and allow troubleshooting operations. They also provide interfaces for alarm and event management, and access to historic data. These systems offer extensive customization capabilities allowing Array Operators to tailor its functionality to their specific needs and operational requirements. Customizable dashboards enable operators to view real-time data in formats that are most relevant to their tasks, while alarm configurations can be fine-tuned to ensure prompt response to critical events.
4. Monitoring collection system: Monitoring data is streamed from Level 1 devices to one or more Monitoring Brokers. Consumers are subscribed to specific monitoring channels. This is a flexible approach to distribute monitoring data across the system. For example, high-frequency monitoring data can be sent to local edge-computing processing or packetized for remote visualization. Normally sampled data is sent to the Event Broker, which acts as a central stream aggregator, buffer and distributor. Data is stored in a database optimized for time-series.

Only the first two items require custom code. The last two involve mostly integration and configuration efforts, not custom code. An important driver in this architecture is avoiding development effort for areas where robust third-party solutions can be found. Selecting standard protocols is an enabler for their integration.

An accurate estimation of the monitoring data load will need to wait until the Level 1 ICDs are completed, but an upper bound can be estimated for 263 antennas, 20 devices per antenna and 100 monitoring points per device, sampled every second. This results in 526,000 samples/second. At 8 bytes per sample this generates an aggregated bandwidth of 4.2 Mbytes/sec. The bandwidth required is not very high, but the number of samples per second can prove challenging. This is one of the reasons to prefer an Event Broker versus a Message Queue. The throughput reported for a system like Kafka, for example, reaches the millions of events per second, while typical message queues (e.g. RabbitMQ) can only handle tens of thousands of events per second before saturating (these limits will be demonstrated with prototypes). Similarly, the required number of transactions per second would overwhelm a standard SQL database, while a time-series database is specifically designed to handle large volumes of time-stamped data.

4.2 Scheduling Stages

Allocating telescope resources to proposed projects—and defining, executing, and post-processing the required observations—involves multiple scheduling decisions made at various levels of abstraction and timescales. While the Telescope Time Allocation system handles PI Proposals in an abstract, telescope-independent manner, the execution of observations requires operations tailored to the specific telescope hardware. While the TAC review process demands scheduling estimates for the entire observing season, the online scheduler can only effectively plan observations a few hours in advance, due to the inherent

uncertainty in environmental conditions that impact observation quality. These constraints have led to the development of a hierarchy of schedulable entities—data structures that represent observations with increasing levels of detail—each serving distinct purposes at different stages of the scheduling process, as outlined in Table 2.

Table 2. Schedulable entities, along with their characteristics, creation process, and the scheduling processes that involve them.

Schedulable Entity	Characteristics	Created by	Scheduling Processes
Observation Specification	An abstract, telescope-independent structure.	TTA	TAC Schedule Forecast. A statistical forecast (e.g. based on pressure plots), not an actual schedule. On the other hand, the TAC review may use the Long-Term Scheduler for simulating allocation scenarios.
Phase 1 Scheduling Block	A more concrete, telescope-dependent structure. They identify a prioritized list of reference subarrays and label calibration scans with intents, but not yet identifying specific calibrators. Phase 1 SBs are created at the end of the Proposal Review process. They include the identification of OBC's and SBC's.	Observation Preparation	Long-Term Scheduling, spanning whole Observing Seasons. A coarse grain schedule is produced for long-term planning purposes.
Phase 2 Scheduling Block	Concrete, executable SB. A reference subarray is selected from the prioritized list, and calibrators are assigned to each calibration scan. The Phase 2 SBs are produced as part of the Mid-term scheduling process.	Observation Preparation	Mid-Term Scheduling, spanning a few weeks. Online Dynamic Scheduling, spanning a few hours.
Calibratable Scan Block	Part of the Phase 2 SB, describes a set of calibration and target scans that can be calibrated as a unit.	Observation Preparation	Online Dynamic Scheduling. The dynamic scheduler may update CSBs to

Schedulable Entity	Characteristics	Created by	Scheduling Processes
			account for current environmental conditions. CSBs are sent to the Executor.

As we traverse down the table, the timescale of the scheduling processes become shorter, and the entities become more specific and detailed. This includes the definition of the calibration scan parameters (SBCs, OBCs), which may be introduced only as overheads in the Observation Specification, being coarse at the level of Phase I, and completely specified at the CSB-level.

These processes ultimately produce a stream of CSBs that are sent to the Executor. By dividing observations—which typically last several hours—into minute-long CSBs, the system gains greater flexibility to respond to changing conditions and faults. Interruptions to SB execution, as well as the addition or removal of antennas from a subarray, are generally performed at CSB boundaries.

4.3 Online Dynamic Observation Scheduling

At its core, the scheduling function can be defined in the following terms: given a number of Scheduling Blocks, a time-frame, a set of reference subarrays[1], a mapping between the Scheduling Block and one or more reference subarrays (with one being the primary subarray and the others being secondary choices), an algorithm to rank the SBs based on several criteria and weights, an objective function to optimize, and a set of scheduling constraints; produce an optimized schedule of subarrays and the SBs that will be executed in each one of them for the given timeframe.

Once this optimized schedule has been produced, the Scheduler coordinates its execution either in fully automatic mode or in an advisory mode that requires a confirmation from the Array Operator before proceeding to execute the next action. A manual mode is also provided where the Array Operator is fully in charge of creating and executing observation schedules (or incrementally modifying a schedule created by the Scheduler, for example).
The Scheduler selects SBs from a pool that contains more SBs that can be observed during the observing season. The objective function has been characterized as maximizing [RD21]:

The sum of effective baseline hours observed for A-ranked projects that are successfully completed.

The requirement to schedule not only SBs but also generate the schedule of subarrays places the algorithm in the category of Multi-Processor Task (MPT) scheduling [RD22]. In this case, each task (SB) needs several machines (antennas) for its execution. Many algorithms for MPT scheduling problems are known to be NP-Hard. A starting point for developing a suitable algorithm for ngVLA is to analyze the suitability of

[1] A Reference Subarray is one of a pre-defined set of approximately 20 distinct, complementary groupings of ngVLA antennas, defined primarily to facilitate scheduling, planning and observation tracking. The actual subarray used for the executing a SB can differ slightly. See [RD21] for details.

existing algorithms. However, first the problem needs to be represented in a compatible form. Common objective functions in scheduling theory are the minimization of the weighted sum of the tasks completion times or minimizing the total schedule span. In our case, the task weight is a function of the rank and other parameters and constraints. An algorithm that minimizes the weighted sum will try to schedule as many highly weighed tasks as possible first. However, this objective function may be stronger than needed, if we care mostly about observing as A-ranked SBs as possible in an observing season, but not necessarily that these are observed first in the schedule. For this reason, using the schedule span as objective function maybe more appropriate. Compressing the total length of the schedule for A-rank SBs will certainly maximize the utilization of the array and the baseline-hours.

For minimizing the weighted sum, an existing algorithm relies on enumerating all possible subarray configuration solutions as a graph and finding an optimal solution by calculating the shortest path [RD23]. This may be appropriate for online or mid-term scheduling, but not for long-term scheduling as the size of the graph grows exponentially. On the other hand, minimizing the schedule span can be transformed into a linear optimization problem [RD24]. In practice, the complexity costs of these algorithms may require us to adopt other alternatives, such as simulated annealing or other meta-heuristic methods [RD25].

The development of the scheduling algorithm for ngVLA is a challenging problem. As the requirements are not quite settled, it may be a task better carried out following an agile approach. Algorithm development will be addressed before PDR and will probably continue during and after construction. The software should allow swapping of different algorithms, as they are developed and tested.

It is possible to differentiate the core functions related with the execution of the algorithm from other auxiliary function. The core functions are:

- Scheduling Algorithm
- SB Pool Management
- SB Ranking/Weighting
- Reference Subarray Mapping
- Scheduling Constraints
-

One way of architecting the Scheduler is applying the *microkernel* architectural style. In this style, a *core system* is defined, containing the minimal functionality required to run the system. This core is extended through *plug-in* components, as illustrated in Figure 6. The diagram shows several extensions requested explicitly in the functional requirements.

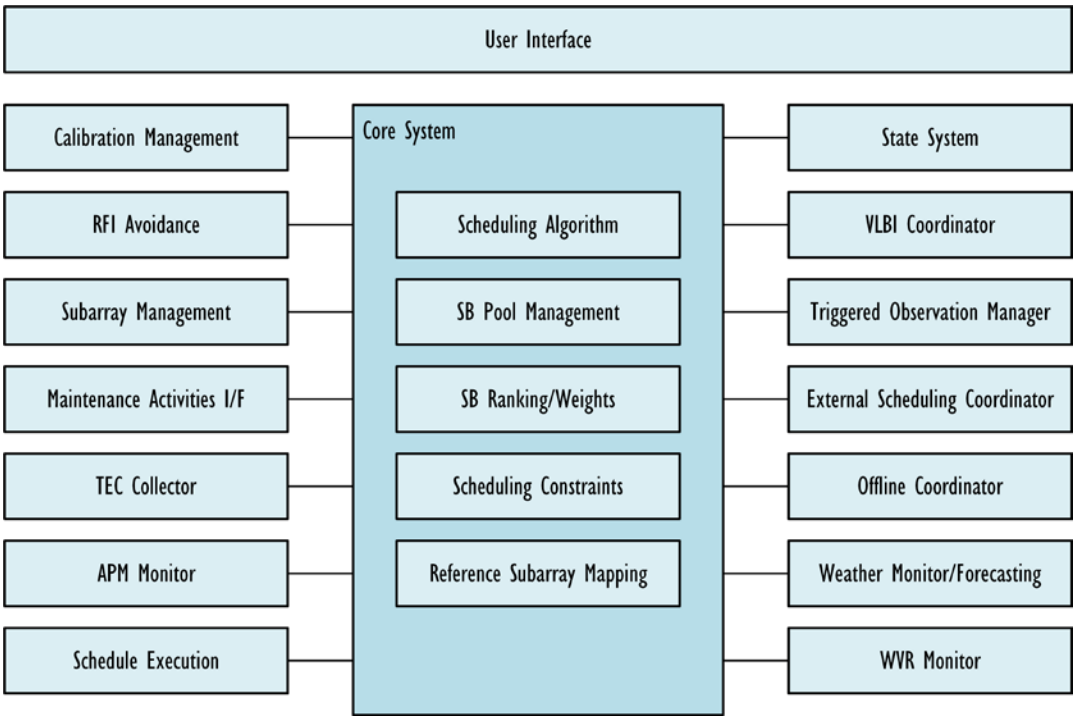

Figure 6 – Scheduling microkernel architecture.

4.4 Subarray Creation

As defined in the CSS Glossary [RD30]:

> **Subarray:** A subarray is a group of system resources that: are scheduled as a set and have the same hardware resources (such as antennas, correlator, and computing). The antennas do not necessarily point in the same direction. Subarrays can operate concurrently (i.e. independently and at the same time), or they can be coordinated (see simultaneous and synchronous subarray definitions).

In this architecture, a subarray is realized in the system as a collaboration of components and communication topics in the Event Broker. The steps that are followed to create a subarray and execute observations are:

1. The Dynamic Scheduler decides the schedule of subarrays and the SBs that will be executed in each one of them.
2. The Scheduler sends a request to the Subarray Management service, which creates an Executor and several Spectral Reception and Analysis Units. If necessary, the subarray-related dynamic topics in the Event Broker are created as well.
3. Resources are assigned to a subarray by connecting the Executor and the SRAUs to the corresponding static topics in the Event Broker, forming the structure shown in Figure 7. The assignment of subarray resources is defined in the Event Broker by means of these connections.

4. The Scheduler starts submitting CSBs to the Subarray Management service, which dispatch them to the Executor. The Executor receives the CSBs in a queue and executes each one of them, by sending Task messages to the Hardware Tasks topic. For a given subarray, the Scheduler only sends CSBs to a single Executor.

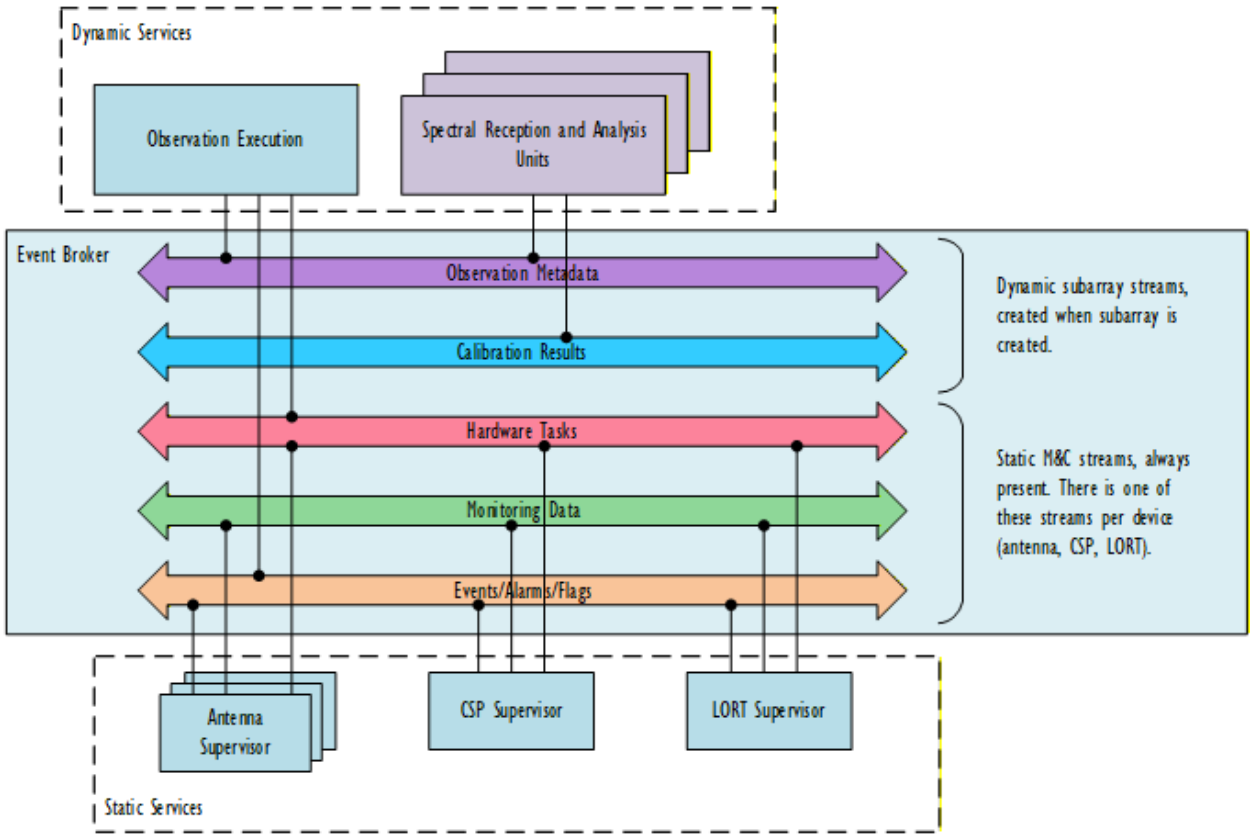

Figure 7. Subarray structure. A subarray consists of a collaboration of dynamic (Level 2) and static (Level 1) components that communicate between each other through the Event Broker.

The allocation of hardware resources to a subarray is simply the process of connecting the Observation Execution service to the appropriate Hardware Task streams and Event/Alarm streams. These connections can be made dynamically while an observation is ongoing, allowing the system to seamlessly manage the disconnection and reconnection of antennas. Supervisors, in general, are designed to be stateless. Stateless programming in an approach where components don't retain information from previous interactions, with each request containing all necessary information to complete the task. This leads to benefits like easier scalability, higher fault tolerance and simpler debugging. Supervisors serve as task interpreters, receiving instructions from the Observation Executor and translating them into low-level commands for the underlying hardware. Ideally, they do not retain state variables. Since Tasks are timestamped, they are transient, expiring once the system's time surpasses their designated timestamp. Most hardware operations listed in Table 3 are stateless, except for configuring the CSP subarray, the CSP scan sequence and the ACU mode and position. The scan sequence operation is stateful only within

a CSB, as scan sequences are configured specifically for each CSB, and there's no state between scan sequence configurations. The CSP subarray state consist of the antenna allocations. Antennas are allocated to a CSP subarray at the beginning of an observation and the current allocation can be obtained by querying the CMC (CSP Monitoring and Control). Altering the CSP subarray composition within the set of antennas assigned to the subarray at the beginning of the observation does not require additional interactions with the CSP, but incorporating external antennas will require additional CSP commands (e.g. destroying and recreating the CSP subarray). In the case of the ACU, the state consists of its mode and current position, which can be recovered by the Antenna Supervisor from the ACU itself.

4.5 Observation Execution

The M&C functions of the ONL system are implemented in the services deployed in Level 2 and are only supervisory. Real-time control loops are implemented in Level 1. Table 3 shows the most important control functions involved in performing an observation.

Table 3. Important control functions.

H/W Device	Control Functions
1. FrontEnd	1.1 Control of the attenuator bias. These are read from the configuration database and set by the HIL board. There is no need for active control.
	1.2 Control the noise diode - turn it on/off and adjust the gain. Adjusting the gain is necessary for solar mode observations.
2. Cryogenic System	2.1 Supervisory control of the internal feedback loops (Variable Frequency Drive).
	2.2 Supervisory control of the dewar vacuum system. Send alarms in case of temperature oscillations. If the temperature increases momentarily, it may be necessary to trigger a pump-out.
3. Integrated Receiver and Digitizer	3.1 Adjust attenuators based on statistics from the ADC. The statistics are retrieved from the DBE.
	3.2 Alternatively, the system may incorporate total power detectors.
	3.3 There is a command to synchronize the sampler clocks, with the drift being reported in a monitoring point.
	3.4 It is possible that the optical transmitters may be turned on/off, avoiding sending signal from the bands that are not being used in the DBE. Alternatively, the DBE inputs may be multiplexed. A multiplexor would switch between bands 1-5 and band 6.
	3.5 Calibration coefficients need to be provided to the DBE for conversion of the IQ pair to USB/LSB, including sideband separation coefficients. These parameters depend on the IRD connected to the DBE.
4. Digital Back End	4.1 Tuning the number of bits. The bit depth is configurable to 2, 4, 6, 8, 12 or 16 bits quantization per subband. In practice, 8-bits is the default and 16-bits is available when the transmitted bandwidth is < 10 GHz (bands 1, 2, 3).

H/W Device	Control Functions
	4.2 Antenna time may differ from central time, due to propagation delays. The LO has round-trip control, but the timing signal does not. This is not a problem for interferometric observations, as the difference is in the order of nanoseconds, but it may be necessary to re-sync for time-domain (pulsar) observations.
	4.3 Subband selection. The DBE divides the bandwidth from the receiver in fixed-position 218.75 MHz subbands.
5. Water Vapor Radiometer	5.1 Receive the WVR data, optionally applying it in the incoming data from the CSP.
6. DC Power Supply System	6.1 Supervisory monitoring and control of voltage and temperature sensors.
7. Environmental Control (HVAC and Glycol cooling loop)	7.1 Supervisory control. Setpoints are configurable.
8. Central Signal Processor	8.1 Supervisory control of system voltages and temperatures.
	8.2 Scan sequence configuration, including data reception configuration.
	8.3 Delay tracking and fringe rotation command streaming.
	8.4 Beamforming command streaming.
	8.5 Array phasing command streaming. Solutions are computed from complex gain and reference pointing scans and are applied in subsequent scans. The solutions need to be computed and applied in 3 seconds or less since the end of the calibration scan.
9. Local Oscillator Reference and Timing	9.1 Monitor timing signal drift between Maser and Global Navigation Satellite System (GNSS) Antenna.
	9.2 Issue re-sync control command if the drift strays beyond allowable margins.
	9.3 For the Reference and Timing Distribution system, it may be necessary to issue a re-sync with clock edge command, if the signal strays beyond allowable margins.
10. Antenna Control Unit (ACU)	10.1 Send antenna trajectory command stream (azimuth and elevation).

It is possible to classify these functions in the following types:

- *Device configuration functions*: Configurations are loaded into devices from the Configuration Database either at system startup, or before starting a scan.

- *Basic supervisory M&C functions*: In this case the system does not actively control closed real-time feedback loops, but monitors parameters, issuing alarms or triggering corrective actions when the parameters go beyond their allowed operation ranges.

- *Scan configuration functions*: These functions prepare the system for the execution of a scan. Once the changes introduced in the devices have settled, the data received from the CSP can be considered valid. Examples in this category are the IRD attenuator setting, setting the number of bits in the DBE, command the antennas to track the target source, and configure the CSP.

- *Online calibration functions*: Closed control loops involving the computation and application of calibration solutions. These solutions are computed from data received from a calibration scan and applied in the next scans.

- *Command streaming functions*: Functions such as delay tracking, fringe rotation, beamforming and antenna trajectory commands, that require the control system to generate a stream of control commands.

Many of these operations need to be performed on different telescope hardware devices, so they are inherently parallel. An important requirement is the need to optimize the execution by minimizing overheads and maximizing on-source time. Another important driver is fault tolerance. When faults occur, the system should preserve correct data acquired so far and continue observing to the extent that it is possible.

As discussed in section 4.2, the Dynamic Scheduler breaks down an SB execution into a sequence of Calibratable Scan Block (CSB) structures, which represent a sequence of scans that are executed and calibrated together, and therefore constitutes the unit of execution (while the SB constitutes the unit of scheduling). The function of the Executor is to execute each CSB, breaking it down into Tasks that are submitted to be executed by the telescope hardware devices. While the CSB constitutes a *science view* of scans that need to be observed, the set of Tasks represent a *technical view* of the operations that need to be performed in the hardware—albeit still an abstract representation that need to be translated into specific hardware protocols. It is the Executor responsibility to execute each CSB in an efficient and robust way. This involves scheduling each one of these hardware tasks according to the system timing (e.g. how much time is needed for each scan setup operation) and communication (what is the communication latency to transmit commands) constraints.

The Executor architecture is shown in Figure 8. Scheduling sends the CSB structures to the Executor, where they are enqueued. Each CSB is translated into a Task graph, which represents all the hardware operations that need to be performed to execute it. The Task graph is fed into a Task scheduler, which incorporates timing constraints and optimizes the execution.

Scripting interfaces are made available for commissioning and troubleshooting. The Science API allows the user to create and enqueue CSB structures, and two privileged APIs that provide access to the Executor at the Task and OPC UA levels. Scripts can be generated from the SB and CSB data structures, or these structures can be made available to the scripts by importing them into their namespace during execution. It is still to be decided whether the execution will always involve the execution of a script, or the CSBs

can be executed directly. To a certain extent the question is about where the complexity should be allocated, either in the script or the Task Translator. Standard modes may not require the execution of a script and the complexity of executing a CSB is allocated in the Task Translator. On the other hand, it may be advantageous to implement edge cases or observing modes under development using the scripting interface. It may be appropriate to implement both execution paths.

This architecture is chosen to support fault tolerance; facilitate the management of time constraints and subarray simultaneity/synchronicity; separate science interfaces from engineering interfaces; and decouple telescope-specific functions from general functions, facilitating maintenance and evolution.
Note as well that Task Translator has flexibility in commanding the parts of the telescope, for example, it can easily partition a subarray and command different sets of antennas to point to different targets at the same time.

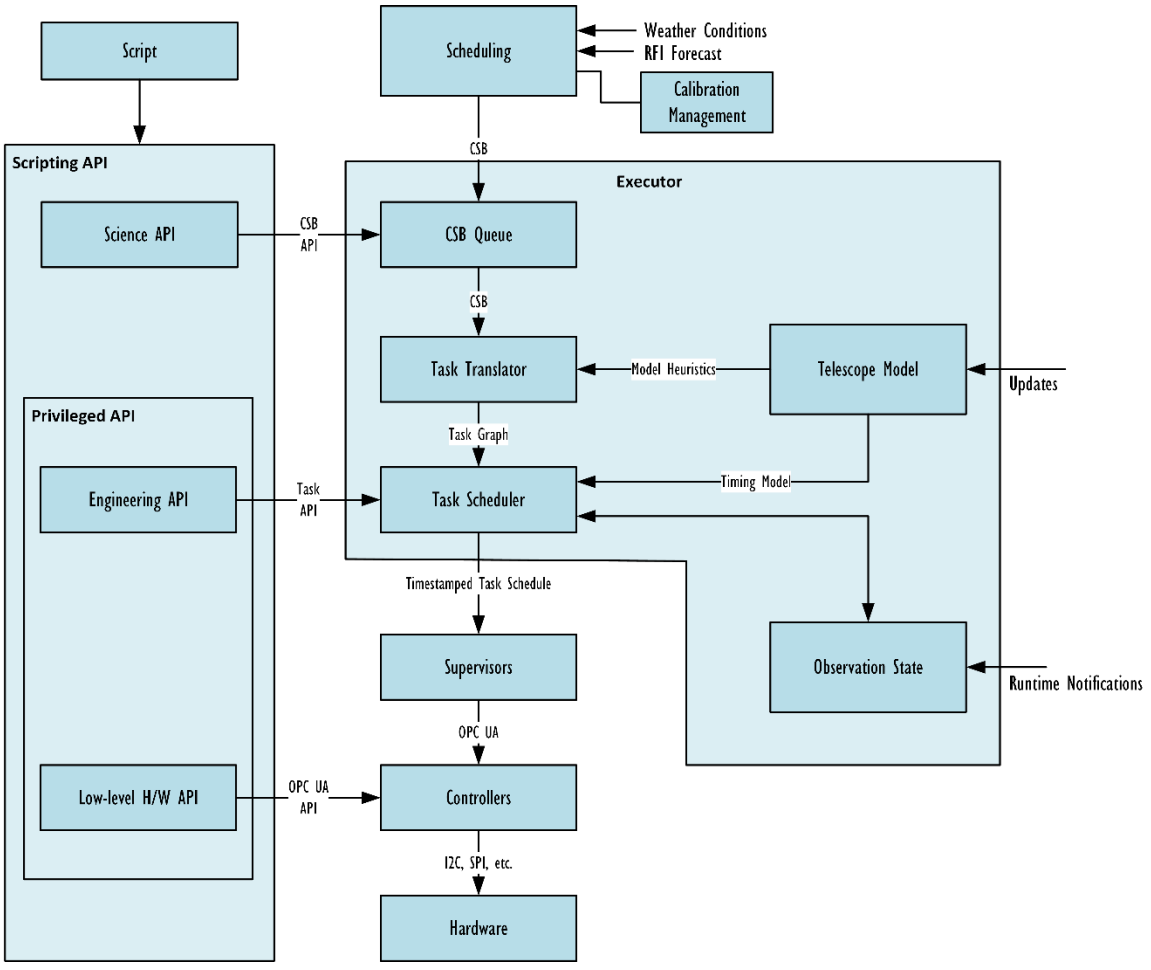

Figure 8. Observation executor architecture. For clarity, the Subarray Management, Scripting I/F and the Executor Control I/F components, shown in Figure 5, have been omitted.

An example of the Executor operation and related data structures is shown in Figure 9.

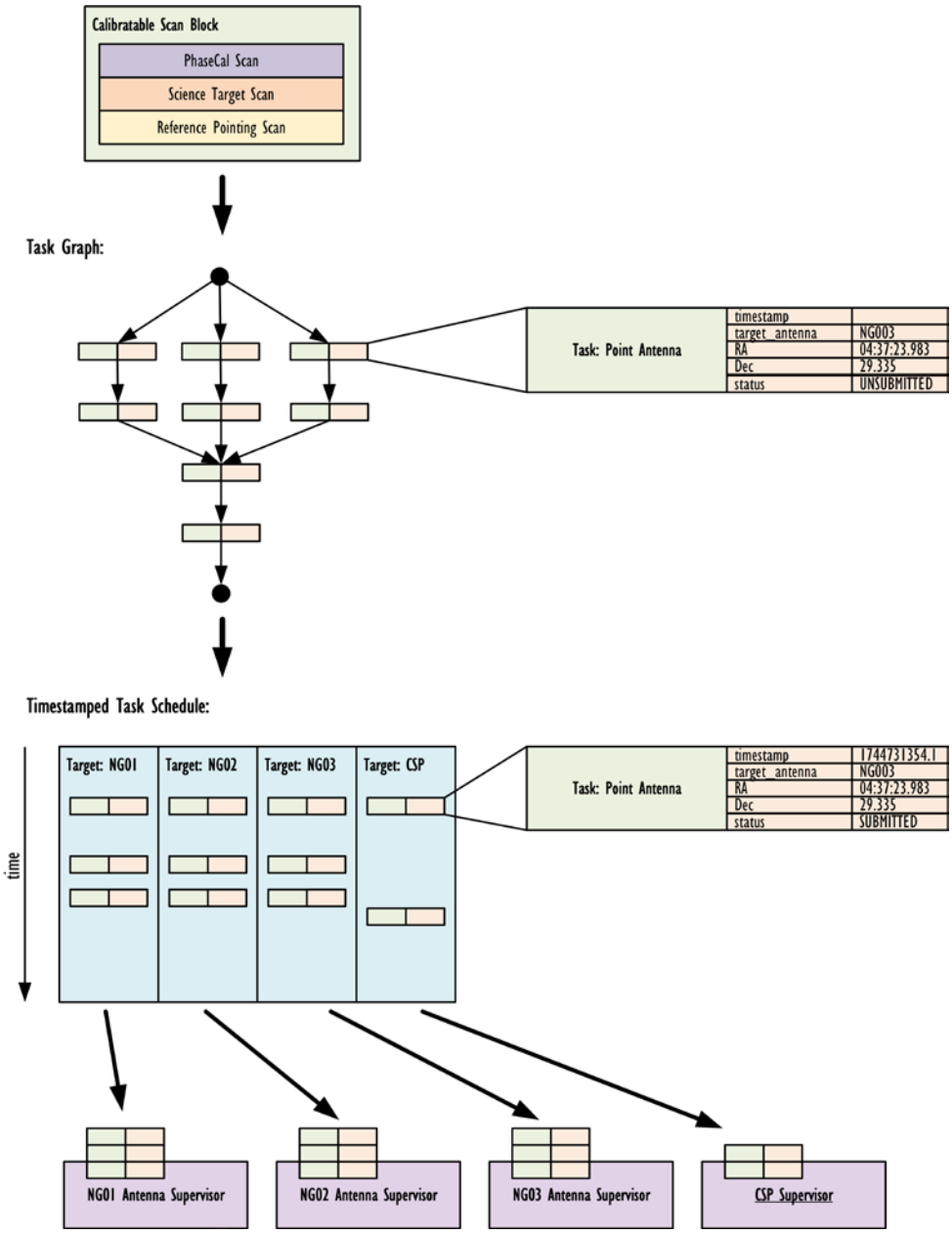

Figure 9. Observation execution structures.

As discussed, the Task Translator converts each CSB in a Task Graph, where each node corresponds to an operation that needs to be executed by the hardware, and the edges represent inter-task dependencies, as some operations depend on other operations that need to be executed previously. At this point, the nodes have unassigned application timestamps. The Task Translator uses a Telescope Model for this translation. This component includes telescope-specific heuristics, for example, constraints for antenna

elevation, antenna cable-wrap, tuning models (for ngVLA this is a simple mapping between frequency ranges and static sub-bands), and other parameters.

The Task Graph is taken by the Task Scheduler, which constructs a schedule of operations, parallelizing tasks and assigning them timestamps in the future. The tasks are sent to queues in each Supervisor. It is the responsibility of each supervisor to execute each task at the correct time.

The Task Scheduler maintains an Observation State model, keeping track of the completion of each task and their super-structures (CSB, SB, etc.). The Observation State model is a state machine that receives notifications from each Supervisor to update the state of the observation execution. This pattern was introduced in ADR004: Observation Execution Communication, Consistency and Coordination.

The Science API provides an interface to construct and submit CSBs, control parameters exposed by the Telescope Model and query the state of the execution from the Observation State. The Telescope Model and Observation State connect with the Array Operator user interfaces applying the MVC pattern.

The Scripting API includes the Science API and two privileged scripting interfaces, the Engineering API allows for the direct submission of Tasks, and the Low-level H/W API provides direct access to Level I hardware controllers by means of their OPC UA interfaces.

The Executor maintains a graph of the tasks that need to be executed in the hardware, their status, and their dependencies, which allows it to make recovery decisions in case of faults.

Timing constraints are delegated to the Task Scheduler. When a Task Graph is executed, timestamps are assigned based on a model of the involved latencies (including transmission latencies), and the Tasks are sent to the Supervisors. The latency model can be made adaptive, by updating it according to the reported latencies from different operations. Supervisors send notifications of the completion of the submitted tasks. If a notification is not received within a given interval of time, the Task is considered as failed. If a completion notification arrives late, the Executor flags the corresponding interval.

Simultaneous subarray observations (CSS18012) and synchronous subarray observations (CSS18016) are supported by the Task Scheduler assigning the same timestamps to operations occurring in different subarrays. If two subarrays need to be coordinated together, they are executed by the same Executor. Differences between the operations that need to be carried in these two subarrays are simply handled by expressing them in the Task Graph. For example, the subarray antennas can be partitioned in two sets, each one observing different targets simultaneously.

The operation of the Task Scheduler is exemplified in Figure 10. The Task Graph incorporates several of the operations from Table 3. The latency of each operation, including communication costs, is used to set the weight of each outgoing edge. To ensure sufficient time for all operations to complete, the Task Scheduler calculates the *maximum spanning tree* of the graph. Its durations are used to construct the schedule shown in the top right corner of the figure.

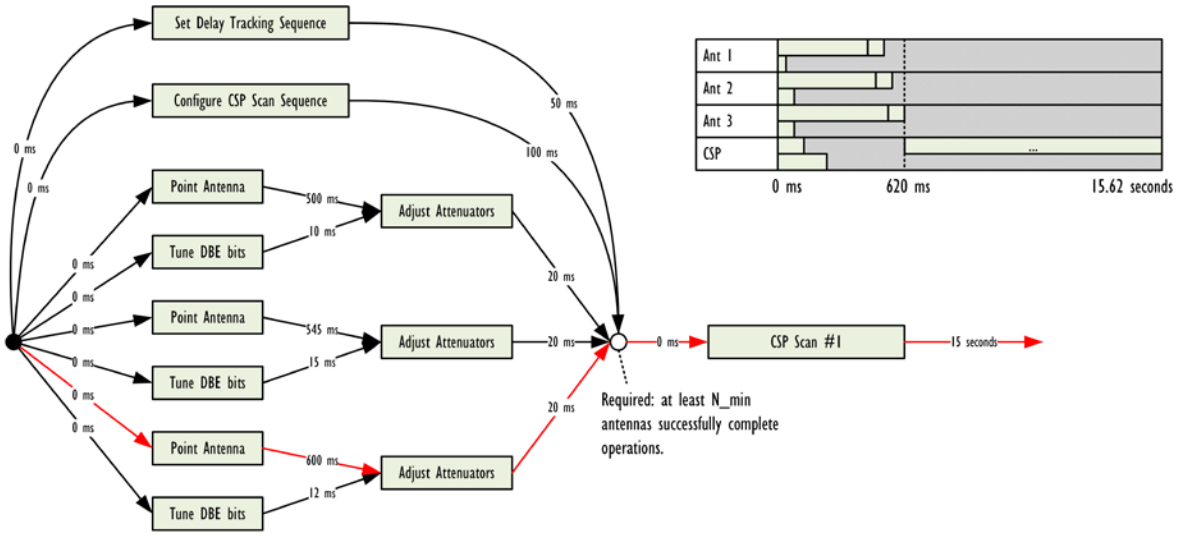

Figure 10. A Task Graph example. The edge weight is determined by the latency of the vertex (operation) immediately preceding, including communication overheads. The edges in red show the maximum spanning tree, necessary to guarantee that all operations have enough time to be completed.

Synchronization nodes are used to wait for parallel Tasks to complete. They can establish criteria for successful completion, as shown in the diagram, where it is required at least N_min antennas to complete for the execution to continue. If more than N_min antennas fail, then the corresponding CSB is marked as FAILED. Several strategies to deal with failed antennas are possible. One possibility is to allow the Task Scheduler to adjust the scan time to compensate for failed antennas, maintaining constant the assigned antenna-hours. Another possibility is continuing the observation with the assigned scan time but allow the Scheduling service to submit additional CSBs to compensate for the lost sensitivity. Implementing these strategies is facilitated by the introduction of these structures in the architecture, and the separation of SB Scheduling, CSB Scheduling, and Task Scheduling.

4.6 *Data Reception & Analysis*

The data from the CSP is received by several machine nodes in the CSP Back End (CBE) cluster, where it is formatted in the final science dataset format (XRADIO) and delivered for transmission to the Data Processing Center. In parallel, the incoming data from the CSP is processed by TelCal to compute calibration solutions, detect RFI and perform other operations like quantization correction. In some cases, TelCal solutions are applied back into the hardware system, e.g. complex gain calibration solutions are applied for phasing the array, or pointing offsets derived from reference pointing scans are incorporated in the commands sent to the antenna ACU as pointing offsets. The visibility data stream comes from the CSP at an average of 21 GBytes/sec, with a peak of 324 GBytes/sec for the most demanding cases from the EOP [RD26]. As there is no buffering for this data until delivering it to the Data Transport System, the reception, formatting and computation processes need to keep up with the peak throughput. This is only possible by parallelizing these processes and the data streams from the CSP, as shown in Figure 11.

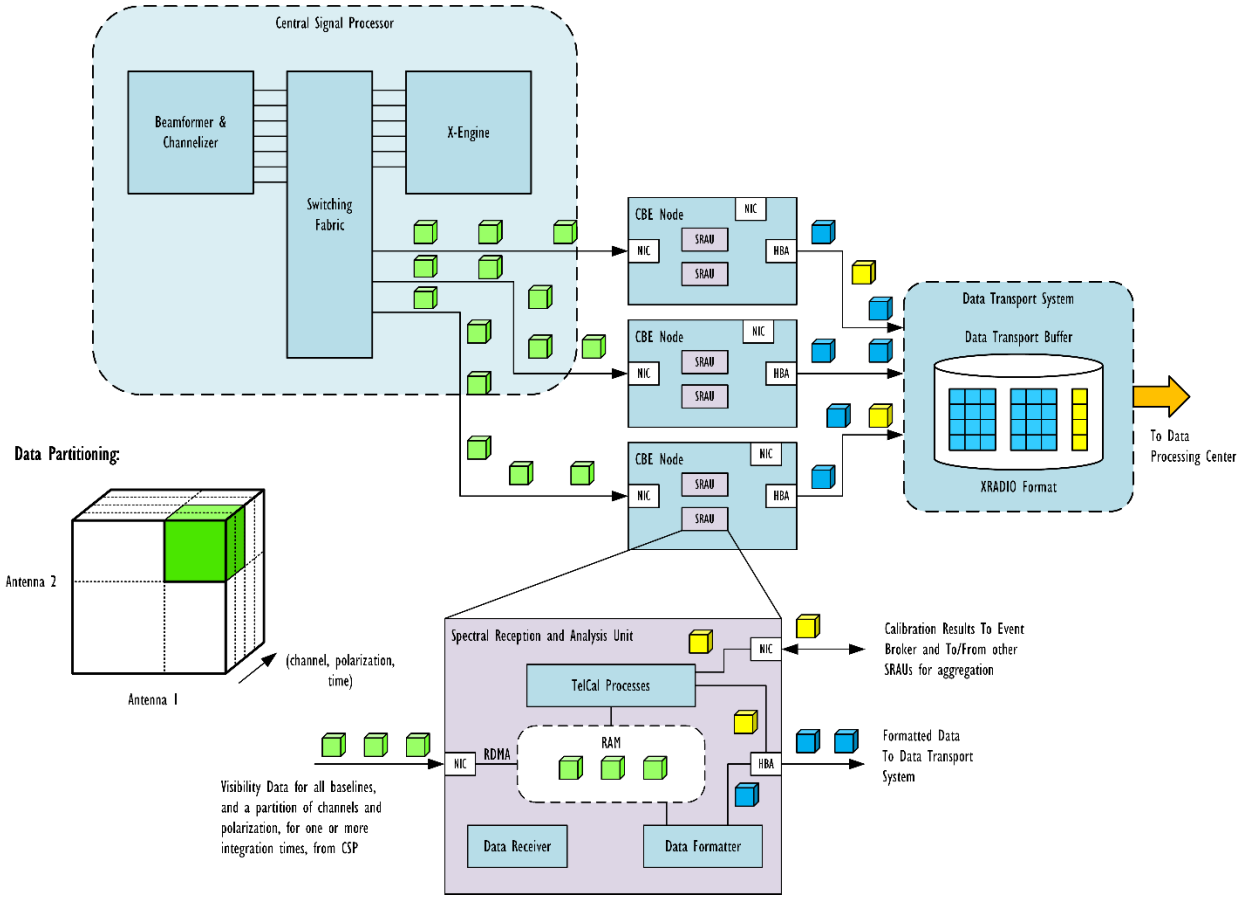

Figure 11. Data Reception and Analysis Architecture. The small boxes aim to illustrate how the data is partitioned and streamed. Green boxes represent raw visibility packets, blue boxes represent formatted data, and yellow boxes calibration results.

The data stream from the CSP is partitioned by spectra and polarization as shown in the lower left corner of Figure 11. The streams are divided in packets where each packet contains the visibilities for all baselines for a given interval of spectral channels, for one or more integrations and polarization products. Maintaining the spectral interval and the polarization products fixed for each stream, the CSP sends packets containing data for subsequent integrations to the same CBE receiver. The interval of spectral channels for each packet is defined as part of the protocol between the CSP and the ONL system. It is defined in terms of the capabilities for each CBE node (maximum bandwidth, computational capabilities and memory) and the characteristics of the observation (time resolution, spectral resolution, polarization products, and number of antennas). Each of these data streams is processed by an independent Spectral Reception and Analysis Unit, which includes collocated processes for Data Reception, Data Formatting and TelCal processing. To optimize throughput, the data is received in a shared area of memory where it is read by the Data Formatters and TelCal processes, avoiding unnecessary data copying. The data is directly sent to CPU or GPU memory from the CSP using Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE). The Data Receiver performs the protocol handshaking and coordination,

e.g. passing on the RDMA memory handles to the CSP and coordinating access to the memory buffers by the other processes.
The spectral partition used in data reception can be made to coincide with the partition of the whole receiver bandwidth in subbands introduced by the DBE, but not necessarily so. The LO down-conversion will introduce instrumental artefacts that will need to be calibrated with on-sky observations, but the subbands contained inside each baseband pair are not expected to require additional calibration to be concatenated together in wider spectral windows.

The most computationally demanding TelCal process is solving antenna complex gains (antsol). A description of the algorithm can be found in section 10. The visibilities for a gain calibration scan are time-averaged, and the Least-Squares algorithm is applied to solve for the antenna gains. During each iteration, the algorithm solves the residual delay as well, by means of applying a linear fitting over the averaged visibilities. The bulk of the calculations in each iteration comes from averaging the visibilities in time, delay-correcting each time-averaged visibility for delay and averaging the corrected visibilities in frequency. As the time-averages are calculated, visibilities contaminated by RFI are excluded. An estimation of the computational cost for flagging has been documented [RD27]. The computational load of these operations is linear with the data rate (although while time-averaging depends on the integration time, the subsequent frequency averaging does not). A corner-case is shown in Table 4. This case, requiring both high spectral resolution and high time resolution, is probably unlikely as the channel sensitivity would be very low, but it is useful for understanding the data rate limits that a node is able to receive and process. This case would be close to saturating a 400 Gbps link with the CSP. The total computational load of 236 GFLOPS would probably require GPU processing, which is feasible as the operations involved are mostly calculating averages and other statistical measures. About 8 nodes would be needed to process the peak 320 GBytes/sec bandwidth from the EOP. Normally, a single Spectral Reception and Analysis Unit (SRAU) will process one spectral quantum of data, but it is possible for multiple SRAUs to communicate data to a common aggregation process if processing a spectral window that is larger than the data limits of a single SRAU is necessary, for example, in cases where higher sensitivity is needed.

Table 4. Estimated antsol and RFI flagging computational load for 263 antennas, 16e3 channels, 100 msec integration, and 10 iterations, for each polarization.

Metric	Value
Data rate	44.10 GBytes/sec
RFI flagging computational load	54.40 GFLOPS
Antsol computational load	181.93 GFLOPS

Constraint CON104 limits the maximum data rate coming out of the CSP to 132 GBytes/sec, a figure that derives from the ROP [RD28]. This figure was updated to 320 GBytes/sec for the EOP [RD26]. Either way, these are large bandwidths for writing into a file system, considering that the bandwidth to a single NVMe SSD disk is in the order of 2-4 GBytes/sec. In general, there will be numerous SRAUs for multiple spectral chunks per subarray, for many concurrent subarrays, writing data into many files. Achieving high bandwidths on writing into a parallel file system under these circumstances can be difficult for several reasons: excessive locking, block misalignment, inadequate block size, network saturation, metadata server congestion, etc. These details depend largely on low-level details of the parallel file system implementation and the connecting networking. Middleware such as MPI-IO allows for the optimization of accessing the filesystem, in combination with data formats such as Parallel NetCDF or HDF5, which define and

communicate the layout of the data to the MPI-IO layer. These are compatible with XRADIO data format, which being based on the xarray Python package, can use either PNetCDF or HDF5 as storage backend.

On the other hand, achieving high bandwidths in the transmission to the Science Data Center may be easier. There is a tradeoff between the bandwidth that will be procured to the Science Data Center versus the size of a filesystem necessary to buffer the CSP output before its transmission. According to current prices (0.05 $/Gbyte for SSD storage, 250,000 $/year for a 400 Gbps fiber optic connection), it seems to be the case that implementing a large buffering filesystem is less expensive than contracting the transmission bandwidth at the peak data rate. Additionally, a buffer filesystem is necessary to tolerate disruptions in the fiber connectivity and avoid operational downtime. Given the high data rates, the filesystem buffer will need to be in the order of the tenths of PetaBytes.

The minimum required bandwidth to the Science Data Center is given by the average output data rate from the CSP. The maximum used bandwidth is set by the peak data rate from the CSP. From the ROP and EOP analysis, most of the observations generate sub-average loads, with few cases pushing the bandwidth several orders of magnitude higher. This suggests strategies at the scheduling level to decrease the need to contract high networking bandwidths: the system can schedule low-bandwidth observations temporarily to allow the buffer to empty out before scheduling another high-bandwidth observation. On the other hand, this strategy may not be necessary, given that even at peak data rates, the link between the Central Electronics Building and the Science Data Processing center is equivalent to another long-baseline antenna, so contracting connectivity at the peak data rate is a small component of the total networking budget for the project.

These tradeoffs will be analyzed further as part of the Technical Infrastructure design, being developed in the context of the NRAO-LCCF collaboration.

4.7 Supporting Databases

The ONL, MCL and MSS systems depend on several supporting databases which are described in Table 5.

Table 5. Supporting databases.

Name	Description	Used By	Updated By
Calibration Database	This database manages OBC calibration data such as antenna positions and their relative offsets to the focal plane, antenna pointing parameters, delay/phase models, etc.; and SBC calibration attributes and pointers to calibration solution products in the Product Repository (SDA), for re-use when possible.	PMN, ObsPrep, Scheduling service, SDP Workflows	SDP Calibration Workflows (OBC).

Name	Description	Used By	Updated By
Calibrator Database	This database stores calibrator source characteristics, in order select them for SBC, OBC and RNC calibration during SB/CSB creation/updating.	PMN, ObsPrep, Scheduling service, SDP Workflows	SDP Calibration Workflows (OBC).
RFI Database	This database stores characteristics of detected interferers.	Scheduling service, SDP Workflows	RFI Manager.
Scheduling Block Database	This database stores Scheduling Block structures.	ObsPrep, Scheduling, Commissioning Tools.	ObsPrep, Scheduling.
Telescope Configuration Database	This distributed database provides a central repository for the configuration needed by ONL and MCL services. This data is kept under version control.	ONL, MCL services, Commissioning Tools.	Telescope Configuration Service.
Engineering Support Database	This database supports engineering operations, storing data such as monitoring streams, alarms and system logs.	MSS, Commissioning Tools.	MCL.

In the context of the ONL system deployment, the first 4 databases: Calibrator DB, Calibration DB, RFI DB and SB DB; are all used primarily by Scheduling and can be deployed in Level 4, probably behind frontend microservices. The last 2 databases: the Telescope Configuration DB and Engineering Support DB, are needed by services behind the DMZ so they should be deployed at Level 3.5, behind the firewall.

The databases involved in calibration activities and RFI flagging require additional discussion. These databases, the entities that they hold, and the software elements that make use them are shown in Figure 12. The Calibrator and Calibration databases are queried by PMD, ObsPrep and Scheduling during the different scheduling stages (see section 4.2) to select the calibrators that should be included in the SB (or Observation Specification) for SBC calibration types; and select suitable OBC calibrations to be used when calibrating each observation dataset downstream in offline processing. When none can be found, or the calibration validity time has expired, the system automatically generates, and schedules special SBs as described in [RD12]. In the case of reference gain calibration, a three-tier process is triggered. All decisions and traces are recorded in the SB.

Besides being updated by the SDP OBC calibration workflows, the Calibrator and Calibration databases are used by the SDP data reduction pipelines. Use cases to consider are:

1. The pipelines need access to calibrator image models for cases where the calibrator is partially resolved on long baseline observations.
2. The pipelines will compare the flux densities it measures from calibrators in the SB with the flux density history of those calibrators to check for anomalous changes.

3. The pipelines will need access to antenna beam patterns to apply primary beam corrections to images already produced, or to apply them to the data during calibration.
4. The pipelines will need access to bandpass solutions to apply them to the data during the calibration workflows.

The RFI Database is updated by the RFI Manager and is used by Scheduling and the SDP workflows (e.g. for flagging RFI-affected channels prior to on-the-fly channel averaging when computing calibrator gain solutions).

Scheduling Blocks specify observations, which result in raw data products and calibration products, stored in the SDA Product Repository. As part of the calibration process, the calibrator and calibration attributes are produced and ingested into their respective databases.

For OBC calibrations that determine parameters that need to be loaded into the telescope hardware devices, configuration documents are generated from calibration tables and are stored in the Telescope Configuration database.

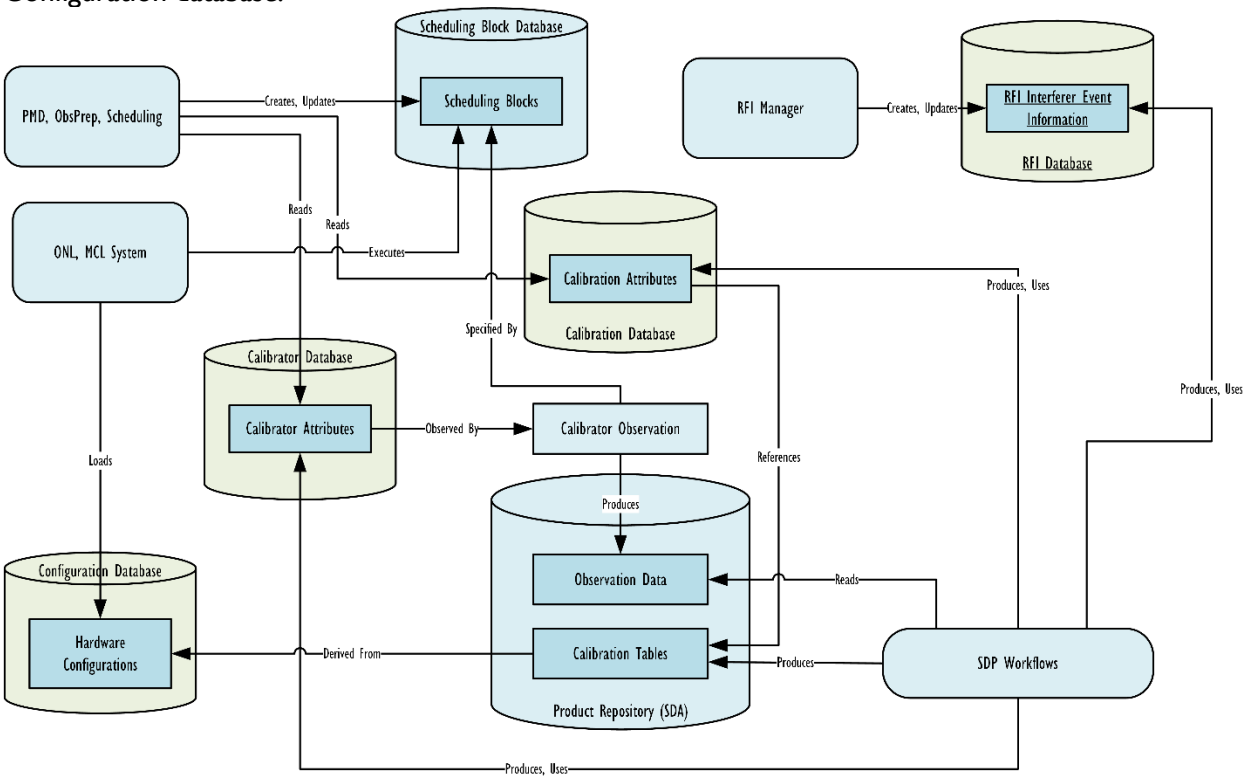

Figure 12. Context diagram for databases that support online operations. The databases in green are "boundary" elements in the interface between ONL and SDP.

This interaction determines a general pattern, where Scheduling (or the Operator through a UI) decides to trigger the updating process for Calibrator Attributes, the OBC Calibration Attributes, or the Hardware Configurations; and special SBs are generated, scheduled, and executed, to be processed by the SDP to update the respective databases. These four databases: Calibrator DB, Calibration DB

Configuration DB and RFI DB; need to be accessed by the SDP and the ONL systems, which are deployed in different centers.

There are several ways that sharing these databases between SDP and ONL can be implemented:

1. Databases can be replicated between geographical centers.
2. Databases won't be accessed directly but services will be constructed in front of them (e.g. following a microservice architecture).

The tradeoffs involved in deciding between these alternatives, and prototypes to explore the feasibility of database replication (it seems to be well supported by most major databases), along with deciding the best database type in each case (e.g. relational, key-value, document, column, graph, time-series) will be decided before the PDR. This activity is coupled with the specification of the data models, which will probably be more effectively performed following an agile development process.

4.8 Interfaces

The interfaces between the ONL and MCL systems and other systems in the ngVLA architecture are shown in Table 6.

Table 6. ONL and MCL system-level interfaces.

ID	System 1	System 2	Description
0101	ONL	MCL	ONL sends real-time Tasks and calibration results; and receives a subset of the monitoring data. Communication is internal to the Event Broker.
0109	ONL	PMN	The Scheduling DB is the boundary between these systems. This interface includes the definition of the Scheduling Block structure. This interface may also include coordination functions to handle triggered observations.
0112	ONL	SDP	This interface includes the definition of the data format for the data products written by ONL and transmitted to the SDP for processing. The Calibrator, Calibration and Configuration databases are updated by the SDP and read by ONL. This interface may also include the status of SDP processing resources for the purpose of scheduling (mostly to avoid buffer overflows).
0113	ONL	TI	This interface specifies software deployment details and computational, networking and storage requirements needed by the ONL system. It also includes interfaces to common software infrastructure such as logging, IT monitoring and container orchestration.
0114	ONL	CSP	This interface specifies software aspects of the transmission of CSP data (interferometric, phased and pulsar data), including the specification of the data

ID	System 1	System 2	Description
			format. The physical aspects of the networking interface are specified in interface 0151, between CSP and TI.
0154	ONL	SIT	ONL sends notifications to SIT, providing a way for end-users to track the status of their observations.
0031	MCL	ANT	OPC UA interface with the Main Antenna (ANT).
0073	MCL	MON	OPC UA interface with the Environmental Monitoring and Characterization (MON) system.
0075	MCL	HIL	This interface aggregates all the OPC UA interfaces for the devices controlled by the HIL boards.
0084	MCL	INF	Interface with Array Infrastructure (INF). Includes interfaces to PLCs that control file alarm systems, intrusion detection alarms and UPS systems.
0086	MCL	OPS	Interface with Operations Buildings (OPS). Includes interfaces to PLCs that control file alarm systems, intrusion detection alarms and UPS systems.
0095	MCL	NSB	Interface with ngVLA Site Buildings (NSB). Includes interfaces to PLCs that control file alarm systems, intrusion detection alarms and UPS systems.
0102	MCL	MSS	MCL transmits monitoring data and generated maintenance tickets to the MSS system.
0103	MCL	TI	It specifies software deployment requirements for the MCL system, defining requirements for computational power, input/output bandwidth and storage.
0104	MCL	DBE	OPC UA interface to the DBE.
0105	MCL	CSP	OPC UA interface to the CSP.
0039	MCL	SBA	OPC UA interface with the Short Baseline Antenna (SBA).

5 Architecturally Significant Needs and Requirements

Table 7 shows the Architectural Significant Requirements, selected from the stakeholder's Needs assigned to the ONL and MCL systems and the System Requirements. The Compliance column explains how the architecture supports this requirement. Derived Requirements will replace the Needs in this table in future versions of this document.

Table 7. Architectural Significant Requirements.

ID	Name	Text	Arch. Value / Tech. Risk	Compliance
CSS13099	Observatory selection of calibrators	The Telescope Support Scientist needs that the observation preparation and scheduling processes select the SBC calibrators to be observed in an SB, according to: observing frequencies, sub-array characteristics, observing conditions and observation goals.	H/L	The Observation Preparation subsystem selects SBC calibrators when Phase 2 SBs are generated, likely during mid-term scheduling planning. The selected calibrators and their observation parameters can be modified by Scheduling (via its Calibration Management plug-in) according to current observing conditions.
CSS13105	Scheduling of OBC observations queued by validity time expiration	The Telescope Support Scientist needs that when an OBC parameter is needed and past its validity time limit – or is expected to be needed (above some probability) at a time when the parameter is no longer valid – the scheduler enqueues a new OBC observation to update the value of this parameter.	H/L	The Calibration Management Scheduling plug-in manages OBC calibrations, generating OBC SBs for calibrations close to their expiration date.
CSS13113	Integration of WVR data in dynamic scheduling	The Telescope Support Scientist needs the dynamic scheduling system to integrate WVR data to estimate tropospheric fluctuations for higher frequency bands (4, 5, and 6).	M/M	WVR data is collected from the WVR system as monitoring data in the Event Broker. This data stream is processed by a TelCal process, which solves and publishes delay results in the Event Broker. Results are sent to Scheduling by means of its WVR Monitor plug-in.
CSS13114	Integration of WVR data for low overhead tropospheric delay correction	The Telescope Support Scientist needs the real-time calibration correction system (RNC) to integrate WVR data to correct for tropospheric delay fluctuations.	M/M	WVR data is collected from the WVR system as monitoring data in the Event Broker. This data stream is processed by a TelCal process, which solves and publishes delay results in the Event Broker. Results are received by the CSP Supervisor, which incorporates it into the delay corrections sent to the CSP.
CSS13115	Integration of phase RMS for real-time	The Telescope Support Scientist needs the dynamic scheduling system to	M/M	A TelCal process calculates the phase RMS from the received interferometric data and publishes

ID	Name	Text	Arch. Value / Tech. Risk	Compliance
	assessment of phase fluctuations in dynamic scheduling	integrate the phase RMS of interferometric data during calibration scans, or science target scans if they are strong enough, into the algorithm.		the results in the Event Broker. Results are received by the Data Collector, from where they are sent to Scheduling via its Weather Monitor plug-in.
CSS13117	Three-tier process for selecting the reference gain calibrator	The Telescope Support Scientist needs the system to select the reference gain calibrator using a three tier process, as described in @020.10.05.05.00-0015-PLA, section 6.2.	M/M	Tier 1 is handled by the Calibration Management service and the Scheduling Calibration Management plug-in. For Tiers 2 and 3, the Calibration Management service creates the respective OBC SBs. These are observed and processed by the SDP. Results are ingested into the Calibration Database, from where they are read by the Calibration Management service, closing the loop. As the interaction encompasses components deployed in the Central Electronics Building and the Data Processing Center, the Calibration Database will be deployed in both locations and will be replicated.
CSS13121	On-the-fly calibrator selection for triggered transient observations	The Telescope Support Scientist needs that the ONL system selects the reference gain calibrators for triggered observations at the time of executing the observation.	M/M	The interaction is similar to CSS13117, but in this case observations of OBC SBs and SDP processing is performed with higher priority.
CSS13132	Pointing model coefficients tracked as a function of time	The Telescope Support Scientist needs that the absolute pointing model coefficients be tracked as a function of time.	M/M	Pointing model coefficients are stored in the Configuration database, under version control. Version control is an important requirement for the Configuration database.
CSS13136	Real-time reference offset pointing correction	The Telescope Support Scientist needs that reference pointing solutions be computed and applied in real-time on the ONL system (RNC).	H/M	Reference pointing solutions are solved by a TelCal process, which publishes the results in the Event Broker. Results are received by the Antenna Supervisor, which incorporates the offsets in the pointing commands sent to the

ID	Name	Text	Arch. Value / Tech. Risk	Compliance
				ACU. Results are also sent to QL display interfaces.
CSS13157	Deprecation of current solutions after antenna maintenance	The Telescope Support Scientist needs that the OBC calibration database be able to recognize when solutions are deprecated after antenna maintenance cycles.	M/M	Scheduling interacts with maintenance components through the Maintenance Activities I/F plug-in and deprecates OBC calibrations. The Calibration Management plug-in creates OBC SBs to be scheduled.
CSS13158	Scheduling of polarization calibration short-observations after antenna maintenance	The Telescope Support Scientist needs the scheduling system to schedule short single-scan observations to update the polarization calibration parameters after antenna maintenance events.	M/M	The interaction with maintenance activities is handled similarly to CSS13157. The interaction with the SDP is handled similarly to CSS13117.
CSS13165	Recording of deterministic flags	The Telescope Support Scientist needs that deterministic flags be recorded, including antenna shadowing, edge channel heuristics and known RFI sources.	H/M	Flags originate from several software components. They are all collected in the Event Broker, which stores them persistently for a time window (in a scalable and fault-tolerant way). They are collected into an SDM table at the end of the observation.
CSS13167	Flagging for hardware failure conditions before computing gains solutions for phasing the array	The Telescope Support Scientist needs the phasing system to apply hardware failure flags before computing gain solutions for phasing the array.	H/M	Flags originate from several components, and are collected in the Event Broker. They are read by TelCal, which discards flagged visibilities before solving for the complex gain.
CSS13169	OFF positions check for no spectral emission	The Telescope Support Scientist needs the ONL system to check OFF positions by automated procedures, ensuring that these positions are free of spectral emission.	M/M	This requirement is handled similarly to CSS13117.
CSS13173	Switched power	The Telescope Support Scientist needs the switched	H/M	Switched power data is collected by the CSP Monitor and Control

ID	Name	Text	Arch. Value / Tech. Risk	Compliance
	measurements recording	power measurements to be included in the science dataset, for offline processing.		(CMC) subsystem and exposed as a monitoring stream. Data is sent to the Event Broker, from where it is collected and formatted into the SDM.
CSS13174	Application of relative amplitude correction when phasing the array	The Telescope Support Scientist needs that switched power amplitude correction be applied to the data prior to phasing the array.	H/M	Switched power data is collected by the CSP Monitor and Control (CMC) subsystem and exposed as a monitoring stream. Data is sent to the Event Broker, from where it is received by TelCal, which incorporates it into the phasing solution. Alternatively, it can be received directly by the CSP Supervisor, if it is straightforward to incorporate into the phasing solutions.
CSS13190	High-frequency favorable condition monitoring	The Telescope Support Scientist needs that the system monitors surface weather parameters, phase fluctuations from current ongoing observations, and PWV sensed with the WVR to determine if current conditions allow high frequency observations.	M/M	The relevant data is either weather monitoring data or TelCal results. They are sent to the Event Broker, from where they are read by the Weather Monitor Scheduling plug-in, which manipulates the SB ranks/weights accordingly. If conditions deteriorate, Scheduling can decide to interrupt an on-going observation. This is handled by stopping the flow of CSBs into the Executor.
CSS13192	Fast switching cycle optimization	The Telescope Support Scientist needs the online system to select the optimum switching cycle appropriate for current atmospheric phase fluctuations.	M/M	This is managed by Scheduling, which adapts the CSBs sent to the Executor based on current atmospheric phase fluctuations, computed by TelCal, as described in CSS13190.
CSS13193	Fast switching feasibility due to calibration overheads	The Telescope Support Scientist needs the online system (scheduler) to check for the fast switching observation feasibility, considering the	L/M	The Calibration Management Scheduling plug-in calculates fast switching calibration overheads from separation angle, observing elevation, slew and settling times, cycle time and scan lengths; and

ID	Name	Text	Arch. Value / Tech. Risk	Compliance
		corresponding calibration overheads.		sets the SB rank/weights accordingly.
CSS13194	Dynamic calibrator cycle times and scan lengths	The Telescope Support Scientist needs that the online system be able to change the calibrator cycle time and scan length dynamically to respond to changing conditions.	M/M	The Calibration Management Scheduling plug-in receives weather monitoring data and TelCal results from the Event Broker (via the Weather Monitoring plug-in in some cases), and computes the calibration cycle time and scan length for each CSB.
CSS13195	Fast switching observation interruption	The Telescope Support Scientist needs that the online system (scheduler) be able to interrupt a fast switching observation if the current conditions deteriorate beyond acceptable limits.	M/M	The Calibration Management or the Weather Monitor Scheduling plug-ins monitor the current conditions, interrupting the execution of an SB if conditions deteriorate beyond acceptable limits. Interrupting an SB consists of interrupting the streaming of CSBs to the Executor.
CSS13198	PWV monitoring from WVR data	The Telescope Support Scientist needs that WVR data be used to monitor PWV, in order to avoid high Tsys conditions during high frequency observations.	M/M	WVR data is received by the Event Broker, from where it is received either by a TelCal process to estimate PWV, or the estimation is performed directly by Weather Monitoring Scheduling plug-in (depending on the required computational capacity). The Scheduling plug-in changes the weight of high-frequency SBs accordingly.
CSS13199	Array phasing real-time calibrations	The Telescope Support Scientist needs that the following per-antenna calibration terms be applied in real-time when phasing the array: bandpass; polarization D; gain phase and delay possibly including both solutions from bright compact science targets themselves as well as those transferred from secondary calibrators, and possibly also including WVR data; and amplitude weighting to	H/M	The required per-antenna calibration terms are either retrieved from the Calibration database or solved dynamically by TelCal. Either way, they are received by the CSP Supervisor, which forms the corresponding Jones matrices and loads them into the CSP, which applies them in the antenna data prior to summation.

ID	Name	Text	Arch. Value / Tech. Risk	Compliance
		maximize S/N ratio, possibly using the switched power data.		
CSS13201	Phased array real-time phase and delay calibration latency	The Telescope Support Scientist needs that the real-time phase and delay calibration for phasing the array be computed and applied within 3 seconds of the calibrator scan.	H/H	This is a latency requirement for phasing control loop. This requirement is fulfilled by parallelizing TelCal computations in Spectral Reception and Analysis Units, avoiding copying data unnecessarily, introducing hardware acceleration and incorporating a low-latency Event Broker (e.g. Kafka).
CSS13202	Phasing scaling and weights recording	The Telescope Support Scientist needs that all calibration factors (including phases, delays and amplitudes/weights) applied pre-sum be recorded in order for the sum to be re-scaled appropriately if necessary.	H/L	Jones matrices are assembled in the CSP Supervisor, who records them by sending them to the Event Broker. In addition, all the input data needed to compute the Jones matrices are also transmitted through the Event Broker, who records them automatically. This data can be collected in the SDM and/or in the Monitoring Database.
CSS14000	RFI avoidance in scheduling algorithm	The Telescope Support Scientist needs the scheduling algorithm to integrate RFI environmental constraints, such as the position and frequencies of known interferers.	M/M	The position and frequency of known interferers is stored in the RFI Database, managed by the RFI Manager. The RFI Avoidance Scheduling plug-in reads the database and transforms expected interference fields into scheduling constraints.
CSS14018	RFI Manager interface to offline processing algorithms	The Telescope Support Scientist needs the system to provide an interface between the offline RFI-processing system and the RFI manager, as an aid to tune the algorithms.	M/M	This interface is implemented by replicating the RFI Database between the Central Electronics Building and the Data Processing Center.
CSS18004	Tracking completion of observations	The Telescope Support Scientist needs that the Execution Fraction be computed and updated after each observation, in order	M/M	As the Executor executes CSBs, it keeps the state of each execution in an internal state machine (persisted via the Event Broker). As CSBs are completed, the Executor sends

ID	Name	Text	Arch. Value / Tech. Risk	Compliance
		to determine when the project is complete.		notifications to the State System Scheduling plug-in, which computes the effective baseline-hours for imaging projects and antenna-hours for phased array projects and keeps this information in a database that is replicated with the Data Processing Center, to be used in end-user presentation interfaces (e.g. workspaces).
CSS18005	Tracking Antenna Hours in phasing and total power projects	The Telescope Support Scientist needs that the Antenna Hours be computed and updated after each observation of a phasing or total power project, in order to determine when the project is complete.	M/M	As the Executor executes CSBs, it keeps the state of each execution in an internal state machine (persisted via the Event Broker). As CSBs are completed, the Executor sends notifications to the State System Scheduling plug-in, which computes the effective baseline-hours for imaging projects and antenna-hours for phased array projects and keeps this information in a database that is replicated with the Data Processing Center, to be used in end-user presentation interfaces (e.g. workspaces).
CSS18012	Simultaneous subarray observations	The Telescope Support Scientist needs that the system supports Simultaneous Subarray observations.	M/M	The Task Scheduler in the Executor sets the same timestamps to simultaneous subarray tasks.
CSS18016	Synchronous subarray observations	The Telescope Support Scientist needs that the system supports Synchronous Subarray observations.	M/M	The Task Scheduler in the Executor sets the same timestamps to synchronous subarray tasks.
CSS18023	Concurrent subarray operation	The Array Operations Support Staff needs that system operates numerous subarrays concurrently.	M/M	Concurrent subarrays are supported across all ONL subsystems: Scheduling, Executor, Event Broker, Data Reception and Analysis.
CSS19514	Observation preemption at CSB boundaries	The Science Telescope Support and Diagnostic Group needs the online	M/M	Interrupting observation at CSB boundaries is simply to command the Scheduler to stop sending CSB

ID	Name	Text	Arch. Value / Tech. Risk	Compliance
		system to interrupt observation at Calibratable Scan Block boundaries, in order to execute time-constrained time-domain observations.		execution commands to the Executor.
CSS2003	Observation Time Model	The observation preparation, execution, and scheduling tools shall support a scientific operations model of allocated time by subarray to an observation.	M/M	Concurrent subarrays are supported across all ONL subsystems: Scheduling, Executor, Event Broker, Data Reception and Analysis.
CSS2007	Observation Execution Logs	The system shall automatically generate execution logs, including the issuance of commands associated with an observation, to provide a record of system actions and to enable system debugging.	H/L	The Event Broker logs automatically all events, including all commands issued during an observation.
CSS2021	Scheduler Start/Stop/Restart	The scheduler shall be designed to make informed decisions about scheduling and early termination of observing blocks based on weather conditions, and to account for previously stopped observations in the ranking process.	M/M	The Scheduler executes SBs by sending CSB execution commands to the Executor. It can stop the execution at CSB boundaries if weather conditions deteriorate. The status of each CSB execution is kept in the Scheduling State System database, so it can resume partially executed SB when appropriate.
CSS2032	Observation Execution Interruption	The Online Subsystem shall provide a way for the Array Operator to abort a running observation in a sub-array. The observations running in other sub-arrays shall not be affected by this abortion.	M/M	An observation can be interrupted at CSB boundaries by the Scheduler and at the Task level by the Executor and Supervisors.
CSS2500	Concurrent subarrays	The Operation Specialist needs the scheduling system to schedule observations in	H/M	The Scheduler manages the execution of observations in multiple subarrays through the Subarray Management service. The

ID	Name	Text	Arch. Value / Tech. Risk	Compliance
		multiple concurrent subarrays.		execution of observations in each subarray is independent from each other.
CSS2508	Weekly scheduler	The Scheduling Support Scientist needs the scheduling system to provide a scheduler capable of producing a 1 week schedule.	M/H	The system is capable of accommodating various scheduling algorithms, customized to meet diverse requirements and timeframes.
CSS2513	Atomic Unit of Observing: Calibratable Scan Blocks (CSB)	The Operation Specialist needs the scheduling system to schedule a set of Calibratable Scan Blocks (CSB) as defined in the Project Data Model document.	M/M	Scheduling enqueues CSB execution commands into a subarray's Executor. The Executor executes individual CSBs.
CSS2514	Scripting interface for CSB execution	The Science Telescope Support and Diagnostic Group and the Commissioning Group need a scripting interface to be able to customize the execution of a set of Calibratable Scan Blocks.	H/M	The system supports scripting interfaces at several levels: a science interface at the CSB level, and privileged interfaces at the Task and OPC UA levels.
CSS2530	Ability to take antennas out or put antennas into observing	The Operation Specialist needs the scheduling system to allow antennas to be taken out, or put back into observing, at CSB boundaries.	H/M	Antennas are taken in or out of a subarray by connecting or disconnecting their Supervisor to the subarray command stream. In general, antenna operations are stateless.
CSS2535	Switching dynamic scheduling algorithm by configuration	The Array Operation Group needs that the scheduling tool allows for the replacement of the dynamic scheduling algorithm by configuration.	M/M	The scheduling algorithm is a core component of the Scheduler microkernel architecture. This architecture defines standardized interfaces that components must implement, allowing them to be interchangeable via configuration through dependency injection.
CSS2539	Incorporation of the availability of transmission, computing and	The Array Operations Division needs that the scheduling system incorporates the availability of transmission, computing	M/M	The Scheduling microkernel architecture incorporates these concerns through plug-ins. Transmission bandwidth and buffering storage are the main

ID	Name	Text	Arch. Value / Tech. Risk	Compliance
	storage resources in the dynamic scheduling algorithm	and storage resources in the scheduling algorithm.		constraints. These can be incorporated into the algorithm as additional resources for the execution of an SB, similarly to array resources (antennas).
CSS2541	Scripting interface for observation execution	The CSV Team, Array Operations Division, and Scientific Telescope Support and Diagnostic Group need the online system to provide a scripting interface to execute observations.	M/M	The system supports scripting interfaces at several levels: a science interface at the CSB level, and privileged interfaces at the Task and OPC UA levels.
CSS2542	Observation script SB context	The CSV Team, Array Operations Division, and Scientific Telescope Support and Diagnostic Group need that the scripting facility execute scripts in the context of an SB and provide access to the SB information from the script.	M/M	The scripting facility uses the observation preparation service to generate a Phase 2 SB, in order to account for the time spent in the script executions.
CSS2545	Immediate execution of SB	The Operation Specialist needs that the scheduling system allows for the execution of an SB in a subarray immediately, optionally interrupting the currently executing observation.	H/M	An observation can be interrupted at CSB boundaries by the Scheduler and at the Task level by the Executor and Supervisors.
CSS2549	SB version control	The Array Operations Division needs that the proposal management system keep the SBs under version control, tracking changes annotating who made the change and when.	H/M	This requirement suggests the use of a Non-SQL document-based database, instead of an SQL database, given that it is easier to implement version control on documents.
CSS5000	Persistent Configuration Data	The online system shall persistently store system configuration data. This configuration data includes the array center position, the antenna locations, the cable and electronic delay	H/M	The Configuration Database needs to be distributed, and support version control. A document-based database may be more appropriate than an SQL database.

ID	Name	Text	Arch. Value / Tech. Risk	Compliance
		model, alarm thresholds, and other parameters. Configuration data shall be kept under version control.		
CSS5001	Subsystem Automation	Individual antennas and subsystems within the array shall perform system configuration and monitoring functions without the need for human intervention. It is a goal that each subsystem be capable of reaching the operationally-ready Standby state after a full power cycle without human intervention.	M/M	Automation and reliability drive a hierarchical M&C architecture where supervisory controller processes and HIL controller boards control hardware elements autonomously, can be redundant, and are managed by a container orchestrator, which can restart them in case of failures.
CSS5500	Subarray support in observation execution	The Operations Specialist needs the observation execution software to support multiple subarrays.	H/M	Subarrays are managed by the Scheduling and Subarray Management services. The execution of observations for each subarray is performed by subarray-specific dynamic Level-3 services, which control hardware resources through Level-2 Supervisor services, allocated to the corresponding subarray.
CSS5503	Scripting interface	The Array Operations Division and the Engineering Operations Division need the observation execution system to provide a scripting interface that allows the execution of science, test, engineering and observatory-function observations from scripts.	M/M	The system supports scripting interfaces at several levels: a science interface at the CSB level, and privileged interfaces at the Task and OPC UA levels.
CSS5504	SB context for manual scripts	The Operation Specialist needs that manual scripts be associated with an SB in order to be scheduled through the scheduling system.	M/M	The scripting facility uses the observation preparation service to generate a Phase 2 SB, in order to account for the time spent in the script executions.

ID	Name	Text	Arch. Value / Tech. Risk	Compliance
CSS5505	Ability of SBs or scripts ability to command all parts of the array	The Operation Specialist needs the observation execution system to allow the SBs or the scripts derived from them, to command all parts of the array.	H/M	The system supports scripting interfaces at several levels: a science interface at the CSB level, and privileged interfaces at the Task and OPC UA levels.
CSS5506	Role-based permission system	The Array Operations Division and the Engineering Operations Division need the observation execution system to provide interactive interfaces, integrating a role-based permission system to control the operations that each user is allowed to perform.	M/M	Interactive interfaces and services will integrate with a Role-Based Access Control (RBAC) system, providing fine-grained authorization.
CSS8110	Scheduling Test Observations Using Specific Software Versions	Telescope Support Scientists need to schedule test observations using specific versions of the software.	M/M	Deploying specific versions of software is supported through containerization.

6 Technical Risks

Table 8 shows the distribution of the perceived architectural value and technical risk of all Quality Attribute Scenarios (Table 1) and Architectural Significant Requirements (Table 7).

Table 8. Architectural value and technical risk distribution for QAS and ASR requirements.

	Technical Risk		
Architectural Value	**Low**	**Medium**	**High**
Low	0	1	0
Medium	4	39	1
High	5	22	3

The requirements that present high technical risk are:

- QAS012: Performance - Scheduling algorithm
- QAS014: Performance - Data reception and formatting throughput

- CSS13201: Phased array real-time phase and delay calibration latency
- CSS2508: Weekly scheduler

Risks for the Scheduler (QAS012 and CSS2508) come from algorithm requirement uncertainty and the challenge of solving for optimal or near-optimal solutions in an acceptable time. As noted before, many problems in this area are known to be NP-hard and can explode combinatorially. It is suggested to mitigate this risk by starting requirement definition and prototyping following an agile approach as soon as possible. On the other hand, a non-optimal scheduler will mostly affect observation efficiency, which may not be critical in the first phases of the project. This fact could buy time for algorithmic research.

The technical risk for QAS014 and CSS13201 comes from the difficulty of achieving the required performance given the high data rates that need to be received, formatted and processed. Although theoretical estimations can give a sense of the difficulty, they cannot replace empirical measurements, so it is recommended that prototypes start as soon as possible, hopefully involving the CSP team to test the full end-to-end interaction involved in the data reception.

Lastly, scheduling risk may be present given the anticipated tight project schedule. Activities that are in the critical path for integration and commissioning should be given priority, and early prototyping should start as soon as possible.

7 Architectural Decision Records

7.1 ADR001: Adopt the Purdue Model for network security

7.1.1 Context

ngVLA will need to operate in an environment of increasing cybersecurity threats. Cybersecurity is a cross-cutting concern that needs to be considered across the whole system architecture. Given that the ONL and MCL systems communicates with controllers and devices that interact with the physical environment, these systems fall in the domain of Operational Technology (OT). This category imposes their own unique performance, reliability and safety requirements, in comparison with pure IT systems. We look for a comprehensive cybersecurity strategy to protect the ONL and MCL systems.

7.1.2 Decision

The ONL system network architecture will follow the Purdue model for network security.

SYS2702 mandates the adoption of the cybersecurity guidelines defined by the NSF-funded Trusted CI, the NSF Cybersecurity Center of Excellence; and AUI Cyber Security Policy.

Trusted CI Must 15 requirement, “Baseline Control Set”, states that:

> *Organizations must adopt and use a baseline control set.*

and recommends the adoption of CIS Controls [RD01]:

> *Trusted CI recommends adoption of the CIS Controls unless another baseline control set is legally or contractually required for selected information asset categories.*

In addition, regarding Must 16 requirement, "Additional & Alternate Controls", and specifically to ICS and SCADA system security, TrustedCI states:

> *ICS and SCADA systems present unique challenges due to a combination of mission criticality, the uniqueness of individual ICS and SCADA systems, especially in scientific contexts where systems may have a longer lifetime than the companies that produced them, and the difficulty of maintaining the security of ICS and SCADA components. CIS provides a companion guide for industrial control environments[*RD02*]. The Industrial Control System Cyber Emergency Response Team (ICS-CERT) is a recognized resource in this area. NIST also provides guidance on ICS security[*RD03*]. Moreover, the security of these devices has much in common with Internet of Things (IoT) devices for which the Department of Homeland Security (DHS) has published a set of strategic security principles[*RD04*].*

NIST and ICS-CERT recommend the adoption of a Defense-in-Depth strategy. From [RD03]:

> *Defense in depth is a multifaceted strategy that integrates people, technology, and operational capabilities to establish variable barriers across multiple layers and dimensions of the organization. Many cybersecurity architectures incorporate the principles of defense in depth. It is considered best practice and integrates into numerous standards and regulatory frameworks.*
> *The basic concepts are to prevent single points of failure in cybersecurity defenses and to assume no single origin of threats. From this position, cybersecurity controls are organized to provide layers of protection around the critical system and system components.*

Defense in depth utilizes these principles to systematically construct layers of security controls. These layers are:

- Layer 1 – Security Management
- Layer 2 – Physical Security
- Layer 3 – Network Security
- Layer 4 – Hardware Security
- Layer 5 – Software Security

The layers that mostly affect the software architecture are Layers 3 and 5. Specifically regarding network security, it is advised to apply the network architecture principles of segmentation and isolation, adopting either the Purdue model, ISA-95 levels, or a three tier IIOT system architecture. All these architectures incorporate a DMZ as an enforcement boundary between network segments. This is also recommended in CIS Control 12 [RD02]. We decide to adopt the Purdue model, as it maps well with ngVLA system architecture.

Other architectural decisions related with these security layers will be documented separately. For example, network monitoring, use of Zero Trust Architecture (ZTA), malicious code protection, data protection, access control and application software security are all controls that will need to be analyzed. Some of these will be developed during the design phase of the project.

7.1.3 Consequences

Adopting the Purdue model imposes restrictions on the communication paths between components, and forces to introduce buffers or proxies for data that needs to cross the DMZ. This can complicate the implementation.

7.2 ADR002: Selection of OPC UA as MCL protocol

7.2.1 Context

The communication protocol between hardware controllers and the Monitoring and & Control system needs to be defined. This system interfaces with:

- HIL boards that control the antenna electronics
- The Antenna Control Unit (ACU), which is a PLC accessible through an OPC UA interface
- HIL boards that control LORT devices
- The CSP and DBEs

These interfaces need to be defined formally, by means of Interface Control Documents (ICD), for which it is necessary to define the underlying communication protocol.

7.2.2 Decision

We select OPC UA, a standard protocol for industrial control systems at the supervisory level. This decision enables QAS008, “Incorporation of third-party S/W components”.

The following aspects need to be considered:

1. Standard or custom protocol? We decide to choose a standard industry protocol instead of defining our own, in order to reduce development costs, benefit from the potential integration of third-party components in the system and facilitate maintenance.
2. Field bus protocol or a supervisory-level protocol? A supervisory-level protocol is appropriate for MCL, given that
    a. Direct communication with the antenna electronics and LORT devices will be handled by the HIL, which acts like a translation layer between field bus (or chip) protocols and supervisory-level protocol
    b. The CSP, DBE and ACU incorporate their own internal protocols and control processors (with the ACU interface being OPC UA)
3. Are real-time capabilities needed? There are no real-time requirements for supervisory control.

OPC UA is transport-agnostic, providing flexibility on choosing the most appropriate transport protocol for our requirements and providing a path to integrate new protocols as the industry evolves. For example, publish-subscribe communication can be performed using MQTT or UDP multicast. Client-server communications can use a binary protocol, REST/HTTP or SOAP. It is possible to integrate OPC UA with TSN if real-time communication capabilities are needed.

Importantly, OPC UA incorporates a robust security model [RD05], addressing QAS015, QAS017, QAS019 and QAS020.

7.2.3 Consequences

OPC UA is perceived as a complex technology. On the other hand, OPC UA complexity arises from the fact that it formally defines a large number of mechanisms (e.g. discovery, notifications and alarms, security, redundancy, programs) that we would need to define ourselves if other apparently simpler protocols were adopted, decreasing interoperability.

7.3 ADR003: Observation Execution Communication, Consistency and Coordination

7.3.1 Context

We identify the following core high-level distributed services involved in the execution of an observation:

- Scheduling
- Observation Executor
- Antenna Supervisor, CSP Supervisor, LORT Supervisor (see ADR)
- Data Receivers and Formatters
- Telescope Calibration Solvers
-

Breaking down the functionality of performing an observation in multiple services is necessary to support availability, scalability, deployability, testability and maintainability. These attributes drive the decomposition of monolithic components into modular distributed services.

These services may need to be broken down further, but this initial decomposition is sufficient to analyze the tradeoffs involved with their mutual *dynamic coupling*, i.e. the way these services communicate at runtime (in opposition to their *static coupling* which is determined by their static dependencies).

Dynamic coupling is influenced by three decisions:

- Communication: synchronous or asynchronous.
- Consistency: atomic or eventual.
- Coordination: orchestrated or choreographed.
-

The effects of choosing between these 8 different possibilities can be analyzed as variations of the *Saga Pattern* [RD06].

7.3.2 Decision

The observation execution workflow will implement the *Parallel Saga* pattern, with asynchronous communication, eventual consistency, and orchestrated coordination.

Figure 13 shows an overview of the most important interactions involved in the observation execution workflow.

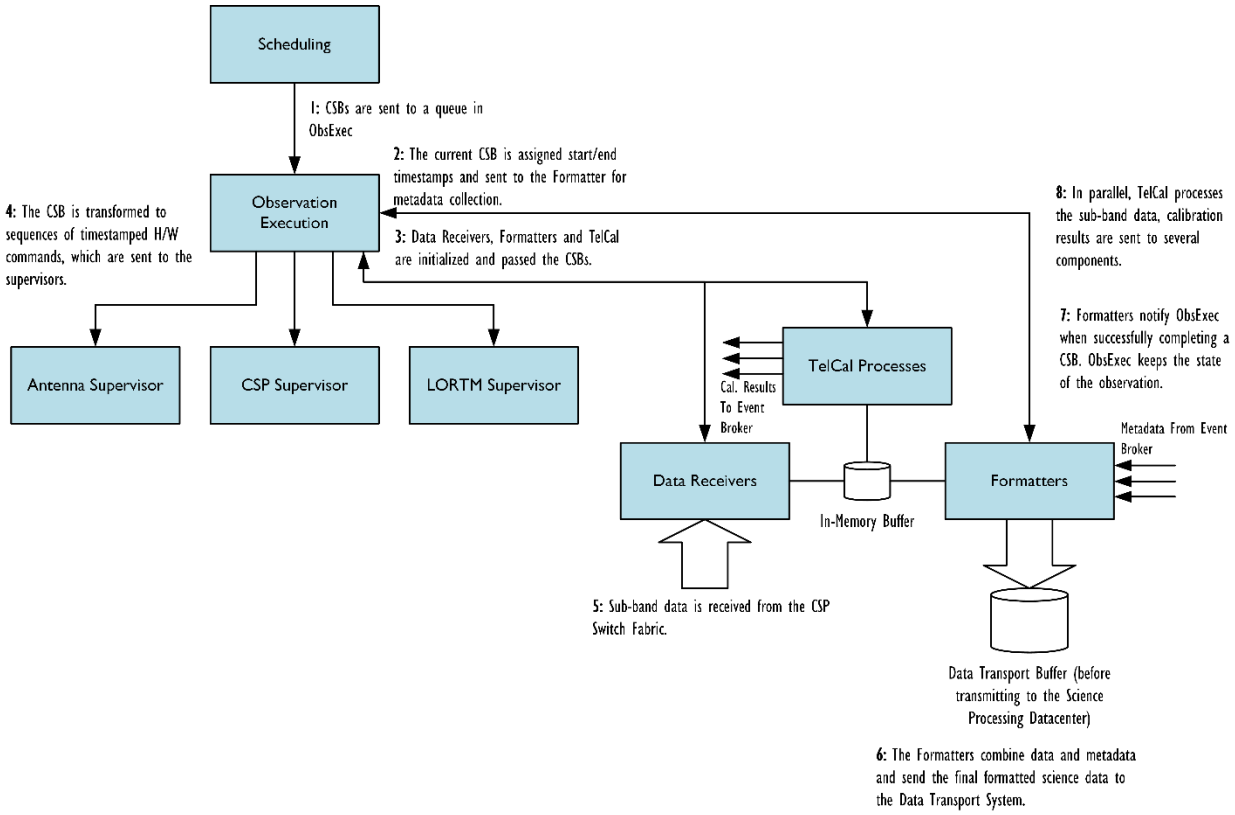

Figure 13. Main observation execution interactions.

There are no shared data elements distributed in multiple services that require atomicity. The data received from the CSP is combined with metadata elements to form the final science datasets that are written into the transport buffer. Scans, grouped as Calibratable Scan Blocks (CSB), establish units of data acquisition. Each scan is first configured in the hardware, and after a period of time where the hardware completes the scan setup, the data becomes valid, it's processed by the CSP and it is sent to the Data Receivers. It is important to keep data and metadata consistent, but eventual consistency is sufficient for this way of operation. The data and metadata received needs to be consistent when the Formatter formats the SDM dataset files. Metadata elements are known a priori, so they can be sent to the Formatter with anticipation. The only exception are the flags, which can be sent asynchronously. This is fine, however, as flags are stored in a Flag Table separate from the visibility datasets.

In this architecture, the Observation Executor and the Scheduler act as the *mediators* or *orchestrators*. The workflow is relatively simple: in case of failures, the most likely path will be to repeat the affected scan/CSB after the elements involved in the failure have been recovered (e.g. by restarting a failed process or switching to an earlier stable software version) or pause the observation for manual recovery procedures. Given this simplicity, service coordination could be choreographed instead, but other requirements push us in the direction of an orchestrated workflow, for instance the need to support scripting in the Executor.

Asynchronous communication support better availability, performance and elasticity. Synchronous communication could be an advantage in case of a complex workflow where atomic consistency is required, but this is not the case.

The Parallel Saga pattern has a low coupling level, low complexity, high responsiveness/availability and high scale/elasticity (see [RD06] for a detailed analysis of the tradeoffs with other saga patterns).

7.3.3 Consequences

Eventual consistency can introduce complexity into the Executor. The Executor would need to keep track of all the scans that are pending to be completed, i.e. both data and metadata have successfully arrived to each Formatter. One pattern that can simplify the implementation of the Executor is the *State Machine*. In this case, the Executor would maintain a state machine representing the state of each scan, and the state of each sequence of scans, i.e. the CSBs. This is shown in step 7 in Figure 13.

8 Future Work

The prototyping activities that should be given priority based on their perceived risk are identified in section 6. These are:

- Prototype the scheduling algorithm, in their different timescales (long, mid, and dynamic).
- Prototype data reception, formatting, buffering and transmission, optimally involving the CSP team for end-to-end testing.
- Prototype online (RNC) calibration loops.
- Prototype architectural parts in the critical path for commissioning and integration activities, e.g. MCL components, the Executor and the Event Broker. This would require at least preliminary versions of the relevant ICDs, and the development of a hardware simulator as real hardware devices will not be available for integration yet. This will be useful for continuing hardware-in-the-loop testing.

In addition, four suggestions were discussed during the Conceptual Design Review:

1. Evaluate the elimination the Supervisor components when sending commands to the hardware. The Executor would send commands directly to Level 1 devices instead.
2. Evaluate the use of pull monitoring instead of publish/subscribe.
3. Evaluate the elimination the Event Broker, sending messages directly from producers to consumers.
4. Specify the characteristics of a CSB (in particular its minimum duration) and revise QAS003, “Availability - Loss of connectivity with antennas”.

The discussion of these alternatives are documented in JIRA tickets NGVLARID-460, NGVLARID-459, NGVLARID-458, and NGVLARID-457 respectively. Evaluating these suggestions requires prototyping or additional system information not yet available (e.g. the completion of the hardware ICDs). These activities should be completed and the corresponding architectural decisions documented by ADRs.

After the CDR, the next steps will be to continue completing the architecture following the ADD process. Both requirements and architecture will be developed in Cameo, using the SysML language. Activities that should be completed before the PDR are:

- Define data models, in particular the Scheduling Block data model and the new Science Data Model.
- Define hardware interfaces. A method for doing this in Cameo using SysML has been defined [RD19].
- Derive requirement specifications from the integrated set of needs. Map them to software elements defined in this architecture. Generate traceability matrices.
- Elaborate software views, e.g. use case, structural, data model, behavioral and interfaces views.

9 Acronyms

Term	Meaning
AD	Applicable Document
AUI	Associated Universities Incorporated
CBE	Correlator Back End
CSP	Central Signal Processor
CSS	Computing and Software System
CSSRR	Computing and Software System Requirements Review
DBE	Digital Back End
EOP	Envelope Observing Program
GPA	Geodetic Precision Array
HMI	Human-Machine Interface
HVAC	Heating Ventilation and Air Conditioning
ICD	Interface Control Document
ICS	Industrial Control System
IMC	Coupled Imaging/Calibration
IPT	Integrated Product Team
LORT	Local Oscillator Reference and Timing
MCL	Monitoring and Control System
MQTT	Message Queueing Telemetry Transport
MSS	Maintenance and Support System
MVC	Model-View-Controller
ngVLA	The Next Generation Very Large Array Project
NIC	Network Interface Card
NRAO	National Radio Astronomy Observatory
NSF	National Science Foundation
OBC	Observatory Provided Calibration
OPC UA	Open Platform Communications Unified Architecture
PMN	Project Management System
REST	Representational State Transfer
RID	Review Item Discrepancy
RNC	Realtime/Online Calibration
ROP	Reference Observing Program
SBC	Calibration Included in the Scheduling Block
SCADA	Supervisory Control And Data Acquisition
SDA	Science Data Archive
SDP	Science Data Processor
SIT	Science Interface and Tools System
SOAP	Simple Object Access Protocol
SOW	Statement of Work
TAC	Telescope Allocation Committee
TBC	To Be Confirmed

Term	Meaning
TBD	To Be Determined
TEC	Total Electron Content
TelCal	Telescope Calibration
TI	Technical Infrastructure
TSN	Time Sensitive Networking
TTAT	Telescope Time Allocation Tools
VLA	Very Large Array
VLBA	Very Long Baseline Array
WVR	Water Vapor Radiometry

10 Appendix 1. Antsol Computational Analysis

The most computing intensive process in TelCal is solving for the complex gain and delays from the stream of received visibilities. Given that there is already an implementation of this algorithm for the VLA, we reverse-engineer the code and estimate the computational load. The code can be found in SVN at file:///home/asg/svn/EVLA/telcal.

The details of the application of the least squares algorithm to the problem of solving the antenna-dependent complex gains can be found in S. Bhatnagar, "Computation of Antenna Dependent Complex Gains". As explained in this note, the normalized output from the correlator between antennas i and j can be expressed in terms of antenna-dependent complex gains as

$$\rho_{ij}^{Obs} = g_i g_j^* \rho_{ij}^o + \epsilon_{ij} \tag{1}$$

and the solution for the complex gains g_i can be found iteratively by applying

$$g_i^n = g_i^{n-1} + \alpha \left[\frac{\sum_{\substack{j \\ j \neq i}} X_{ij}\, g_j^{n-1} w_{ij}}{\sum_{\substack{j \\ j \neq i}} \left| g_j^{n-1} \right|^2 w_{ij}} - g_i^{n-1} \right] \tag{2}$$

with $X_{ij} = \rho_{ij}^{Obs} / \rho_{ij}^o$ and $w_{ij} = 1/\sigma_{ij}^2$. When observing an unresolved gain calibrator, ρ_{ij}^o has a magnitude that is proportional to the flux, and zero phase.

In the EVLA, this calculation is performed in the `AntennaSolution.java` class as part of the `telcal` process. This algorithm is colloquially called "*antsol*". The algorithm implemented inside `AntennaSolution` is more or less as described above, with a twist: during each iteration not only the complex gains are solved, but the delays for each subband are determined as well.

The main data structures held inside `AntennaSolution` are two multi-dimensional arrays:

- `Fringe[baseband][subband][polarization][antenna_1][antenna_2][channel]`
- `GainSolution[baseband][subband][polarization][antenna]`

These structures are initialized inside the function `createAntennaSolution()`. This function reads the data from an on-disk BDF file and performs a time-average to create these structures. Once these structures have been created, for each `(baseband, subband, polarization)` the function `computeGainSolutions()` is called (these calls are sequential, this is one spot where `antsol` could have been parallelized). This function returns a one-dimensional array containing the solutions for each antenna.

Inside `computeGainSolutions`, the solutions are calculated by iteratively calling the function `doOneRound()`. This function receives the fringe data for all baselines and channels:

`Fringe[antenna_1][antenna_2][channel]`

and returns an array

`GainSolution[antenna]`

containing the gain solutions for each antenna and the delay for this baseline.

The bulk of the computations required to determine each solution are performed inside the function `sumChans()`, which is called for each baseline, i.e. it receives a uni-dimensional array `Fringe[channel]`. This function returns an array containing three quantities. The first two are the real and imaginary components of the channel average of the delay-corrected visibilities:

$$S_{ij} = \left\langle V_{ijk} e^{i\tau_{ij}\Delta\omega(k-N_{1/2})} \right\rangle_k \tag{3}$$

where i, j corresponds to the baseline, $k \in [N_{1/2} - N, N_{1/2} + N]$ is the channel number, $\Delta\omega$ is the spectral resolution, and $N_{1/2}$ is the channel in the middle of the subband of bandwidth $2N$. The returned value of the complex S_{ij} is used in function `doOneRound()` to calculate the quantity

$$g_i^n = \frac{\sum_j S_{ij}\, g_j^{n-1} w_{ij}}{\sum_j \left| g_j^{n-1} \right|^2 w_{ij}} \tag{4}$$

This corresponds to Equation I setting $\alpha = 1$.

The third quantity returned by `sumChans()` is used in `doOneRound()` to estimate the delay error. In order to calculate this error, a simple linear regression is applied to the complex visibilities as a function of channel. The linear regression equations for a variable y as a function of x are

$$y = \alpha + \beta x \tag{5}$$

$$\beta = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} \tag{6}$$

$$\alpha = \bar{y} - \beta \bar{x} \tag{7}$$

where $\bar{x}$ and $\bar{y}$ are the arithmetic averages of x and y. In our case x corresponds to the channel and y corresponds to the complex visibility. Developing further equation (6):

$$\beta = \frac{\sum_i (x_i - \bar{x}) y_i - \bar{y} \sum_i (x_i - \bar{x})}{\sum_i (x_i - \bar{x})^2} = \frac{\sum_i (x_i - \bar{x}) y_i}{\sum_i (x_i - \bar{x})^2}$$

as $\sum_i (x_i - \bar{x}) = 0$ because the sum is performed for all the channels between $N_{1/2} - N$ and $N_{1/2} + N$. Hence, an estimation of the (complex) slope of the visibilities as a function of channel is

$$\beta_{ij} = \frac{\sum_k (k - N_{1/2}) S_{ijk}}{\sum_k (k - N_{1/2})^2} \tag{8}$$

where S_{ijk} are the visibilities corrected by a previous estimation of the delay:

$$S_{ijk} = V_{ijk} e^{-i\Delta\omega k \tau_{ij}}$$

The slope β is a complex quantity. The delay is determined as follows: β can be expressed in terms of the delay-corrected visibility as

$$\beta = \frac{dS_{ijk}}{dk} = -i\Delta\omega\tau_{ij} V_{ijk} e^{-i\Delta\omega k \tau_{ij}}$$

Multiplying both sides by V_{ijk}^* we get

$$\beta V_{ijk}^* = -i\Delta\omega\tau_{ij} \left|V_{ijk}\right|^2 e^{-i\Delta\omega k \tau_{ij}}$$

if the delay is relatively small, $e^{-i\Delta\omega k \tau_{ij}} \approx 1$ and

$$\tau_{ij} = -\frac{Im(\beta V_{ijk}^*)}{\Delta\omega \left|V_{ijk}\right|^2} \tag{9}$$

This is how `AntennaSolutions` calculates the delay (although $\Delta\omega$ is incorporated in τ_{ij} as part of its units). Equation (9) is the third parameter returned by `sumChans()`, with the caveat that the denominator in equation (8) is incorporated in `doOneRound()`, as it is a common factor for all baselines. This factor is

$$\sum_k (k - N_{1/2})^2 = \sum_{k'=-(N-1)}^{N} k'^2 = \sum_{k'=1}^{N-1} k'^2 + \sum_{k'=1}^{N} k'^2 = \frac{(N-1)N(2N-1)}{6} + \frac{N(N+1)(2N+1)}{6}$$

(N called the `delayRange` in the source code).

10.1 Computational Load Estimation

The computational load can be estimated from equations (3) and (4), for the complex gain, and equations (8) and (9), for the delay. In addition, the visibilities for each integration time are averaged to form the channel vectors that are used as input in these equations. Counting FLOPS and assuming 8 bytes per visibility:

1. Visibility time averaging:
    a. Operations: $2 \cdot N_{subbands} \cdot N_{pol} \cdot N_{baselines} \cdot N_{chan} \cdot N_{int}$ FLOPS
    b. Data Rate: $8 \cdot N_{subbands} \cdot N_{pol} \cdot N_{baselines} \cdot N_{chan} \cdot N_{int}$ Bytes/sec
    c. Arithmetic Intensity: 2 FLOPS / 8 Bytes = 0.25 FLOPS/Byte
2. Equation 3:
    a. Operations: $13 \cdot N_{subbands} \cdot N_{pol} \cdot N_{baselines} \cdot N_{chan}$ FLOPS
    b. Data Rate: $8 \cdot N_{subbands} \cdot N_{pol} \cdot N_{baselines} \cdot N_{chan}$ Bytes/sec

   c. Arithmetic Intensity: 13 / 8 = 1.63 FLOPS/Byte
3. Equation 4:
   a. Operations: $11 \cdot N_{subbands} \cdot N_{pol} \cdot N_{baselines}$ FLOPS
   b. Data Rate: $8 \cdot N_{subbands} \cdot N_{pol} \cdot N_{baselines}$ Bytes/sec
   c. Arithmetic Intensity: 18 / 8 = 2.25 FLOPS/Byte
4. Equation 8:
   a. Operations: $11 \cdot N_{subbands} \cdot N_{pol} \cdot N_{baselines}$ FLOPS
   b. Data Rate: $8 \cdot N_{subbands} \cdot N_{pol} \cdot N_{baselines}$ Bytes/sec
   c. Arithmetic Intensity: 11 / 8 = 1.375 FLOPS/Byte
5. Equation 9:
   a. Operations: $14 \cdot N_{subbands} \cdot N_{pol} \cdot N_{baselines}$ FLOPS
   b. Data Rate: $8 \cdot N_{subbands} \cdot N_{pol} \cdot N_{baselines}$ Bytes/sec
   c. Arithmetic Intensity: 14 / 8 = 1.75 FLOPS/Byte
   d.

For example, for the worst case of a "fat" subband, with 16e3 spectral channels per subband, the time resolution is 100 msec, with 263 antennas and 10 iterations:

- (1) computational load per subband: 110.25 GigaFLOPs/sec
- (2) computational load per subband: 71.66 GigaFLOPs/sec
- (3-5) computational load per subband: 14.81 MegaFLOPs/sec
- (1) data rate per subband: 44.10 GBytes/sec
- (2) data rate per subband: 4.41 GBytes/sec
- (3-5) data rate per subband: 0.28 MBytes/sec

PCIe4.0 16x, the interface to the GPU, is the limiting factor at 32.0 GBytes/sec. Two GPUs per node could handle the load. These estimations need to be confirmed with a prototype.

11 Appendix 2. Quality Drivers Summary Table

Table 9. Architecturally significant quality drivers summary.

Design Driver	Scenarios	Design Concepts
Availability	• Antenna hardware faults • Software bugs • IT hardware faults • Loss or delayed antenna connectivity • Initialization latency • M&C server faults • Subarray fault isolation	• Fault-tolerant Executor • Scheduling re-issues CSBs • Asynchronous communication and stateless services, to the extent possible • Supervisory M&C layer • Service replication • Streaming channels

Design Driver	Scenarios	Design Concepts
		• High-availability and persistence of communications, use of checkpointing • Subarray operation isolation • Use of containerization to support different software versions per subarray
Operational Efficiency	• Subarray-aware, performant and extensible scheduling algorithm • Extensive use of subarrays • Customization of Operator and Engineering interfaces • Facilitate development of new Observing Modes • System observability • Continuous deployment	• Scheduling algorithm and application of microkernel pattern • Change by configuration (i.e., not by coding) • Separation of concerns, modularity • Automatic collection of troubleshooting data • Microservice pattern
High Data Rates	• Closed loop complex gain calibration latency (array phasing) • Data reception and formatting throughput • Monitoring data throughput	• Data partitioning, parallelization • Avoid unnecessary data copying, use of shared memory • HPC technologies, e.g. RDMA, Parallel IO, GPU acceleration • High-throughput streaming systems (e.g. Kafka)
Cyber Security	• Threats to the M&C networks and equipment • Threats to databases • Threats to services • Threats to other hardware elements	• Apply reference cyber security architectures: Defense-in-Depth, Purdue model (network segmentation and establishment of DMZ) • Standard security technologies: TLS, X.509 Certificates, etc. • Zero Trust Architecture • Secured M&C protocols (OPC UA)