ASTRON
Netherlands Institute for Radio Astronomy

Streaming Visibility Processing study
Final review report

Teamcenter Document No. 013000 revision 1.3

	Organization	Date
Authors:		
S.J. Wijnholds, T.J. Dijkema and S. Yatawatta	ASTRON	March 12, 2026
Checked:		
A.W. Gunst	ASTRON	
Approval:		
A.W. Gunst	ASTRON	

Distribution list:

Group:	Others:
ASTRON T.J. Dijkema A.W. Gunst S.J. Wijnholds S. Yatawatta Review panel C. De Breuck (ESO) J. González (ESO, chair) F. Norambuena (JAO) J. Garcia (JAO) A.R. Offringa (ASTRON) J.W. Steeb (NRAO) E. Schmid (ESO) K. Sugimoto (NAOJ)	Observers C. Lagos (UWA; ESAC observer) M. Diaz-Trigo (ESO) L. Humphreys (ESO) M. Zwaan (ESO)

Document history:

Revision	Date	Chapter / Page	Change	Author
0.1	Aug. 28, 2025	All	Creation	S.J. Wijnholds
0.8	Nov. 6, 2025	All	First full draft	S.J. Wijnholds, S. Yatawatta, T.J. Dijkema
1.0	Nov. 10, 2025	All	Update after internal review	S.J. Wijnholds S. Yatawatta T.J. Dijkema
1.1	Nov. 11, 2025	Title page	Signed version for release	A.W. Gunst
1.2	Dec. 15, 2025	All	Update after final review	S.J. Wijnholds
1.3	Mar. 11, 2026	All	Final updates	S.J. Wijnholds

ASTRON

Table of contents:

1 Introduction

The ALMA Development Working Group has proposed three science drivers for ALMA developments over the coming years [1]:

- Origin of galaxies: trace the cosmic evolution of key elements from the first galaxies (Z > 10) through the peak of star formation (z = 2 – 4) by detecting their cooling lines, both atomic ([CII], [OIII]) and molecular (CO), and dust continuum, at a rate of 1-2 galaxies per hour.
- Origin of chemical complexity: trace the evolution from simple to complex organic molecules through the process of star and planet formation down to solar system scales (~10 – 100 AU) by performing full-band frequency scans at a rate of 2 – 4 protostars per day.
- Origins of planets: image protoplanetary disks in nearby (150 pc) star formation regions to resolve the Earth forming zone (~1 AU) in the dust continuum at wavelengths shorter than 1 mm, enabling detection of the tidal gaps and inner holes created by planets undergoing formation.

Achieving these science goals requires a significant increase in processed IF bandwidth to efficiently conduct redshift surveys or chemical spectral scans as well as improved spectral resolution to observe critical molecular transitions simultaneously.

This proposal resulted in plans for the Wideband Sensitivity Upgrade (WSU), which calls for 16 GHz per polarization [2] and a digital system that ideally supports correlation of the full 16 GHz of bandwidth per feed for up to 80 dish inputs with up to ~1 million spectral channels. Depending on the details of the observation, in particular integration time per visibility and number of frequency channels, this will result in an output data rate that can be as high as several 10s of GBytes/s. At these rates, it quickly becomes challenging to buffer a full observation before starting the data reduction to produce science-ready data products. The aim of the Streaming Visibility Processing (SVP) project [3][4] therefore is to develop a streaming processing solution that performs initial visibility processing in real-time up to the point where the data is reduced to more tractable rates without detrimental effects on the scientific quality of the end products.

Other major radio astronomical facilities like the Low Frequency Array (LOFAR) [5][6], the Square Kilometre Array (SKA) [7][8][9] and the 2000-dish Deep Synoptic Array (DSA-2000) [10] are running into similar challenges. LOFAR, for instance, still uses a large data buffer consisting of hard drives to store the raw correlator data. These data are deleted after pre-processing, whose main steps are flagging, averaging and compression, and which typically reduces the raw data volume by one to two orders of magnitude. As such a large buffer and associated I/O are costly, LOFAR is currently looking into a streaming pre-processing solution. Parallel to this, the data processing software for the SKA is currently being developed, which includes use cases for a real-time calibration pipeline and a fast-imaging pipeline. The use cases for these pipelines ask for a very low latency (order of seconds) between the observation and the output of the pipeline, which calls for a near real-time solution. DSA-2000 envisions to take streaming processing to the extreme by not even storing the raw visibilities and constructing images on the fly instead. How to do that is currently being investigated in the Radio Camera Initiative (RCI) [11]. A secondary goal of the SVP project is therefore to use this synergy and leverage the available expertise to benefit ALMA.

These benefits could not only lie in the implementation of streaming visibility processing itself, but also in demonstrating advanced processing techniques like compression, baseline-dependent averaging and direction-dependent calibration and imaging, on ALMA data.

For ALMA, the need to calibrate the observation based on the Water Vapor Radiometer (WVR) data poses a major challenge for streaming visibility processing. Currently, ALMA uses interpolation of the WVR data to match the time cadence of the WVR data to that of the visibility data and to improve the SNR of the WVR measurements. This implies that calibration corrections to visibilities for at a given time can only be made after the required number of WVR data samples is collected to perform the implementation for that time. This would require significant data buffering. In the past, initial corrections were made using extrapolation based on WVR data collected so far, but that method will, obviously, provide lower calibration quality. This issue was explored at length during the first-year review and several discussions thereafter. It was concluded that it would be difficult, even with positive outcomes, to get streaming visibility processing implemented in the actual production system. In those discussions, a new use case was identified: low-latency imaging for quality assurance (QA). This would be a nice addition to the real-time quick look (RT QL) system, which currently does not produce images. The original aims for this project as described in [4] were therefore adjusted towards this new use case as described in the updated statement for work [12].

After these discussions, the aims of this study can be summarized as follows:

- Simulate the capture of the visibility output stream from the ALMA correlator for several representative ALMA observations using an ALMA emulator provided by ESO.
- Demonstrate the ability to reliably receive the data via the Data Distribution Service (DDS) interface and pass it on to the streaming interface of DP3. DDS is a middleware protocol and API standard for data-centric connectivity. It integrates the components of a system together, providing low-latency data connectivity, extreme reliability, and a scalable architecture [13];
- Perform initial processing steps in a streaming fashion to produce an equivalent output data set that can be loaded and processed in CASA.
- Assess the feasibility and impact of lossy compression on the final data products produced by the ALMA imaging pipeline. Specifically. Baseline-dependent averaging (BDA) [14] and Dysco [15] compression were considered in this study.
- Demonstrate low-latency imaging for QA.
- Assess the steps needed to bring streaming data processing to production and advise on a possible implementation plan.

2 Streaming data processing vision

The temporal and spectral resolution of the raw visibility data of a radio interferometer is often dictated by (1) the need to limit time and frequency smearing on the longest baselines and (2) measures to carefully flag or remove interfering signals. The initial processing of these raw visibilities typically involves steps like flagging, initial

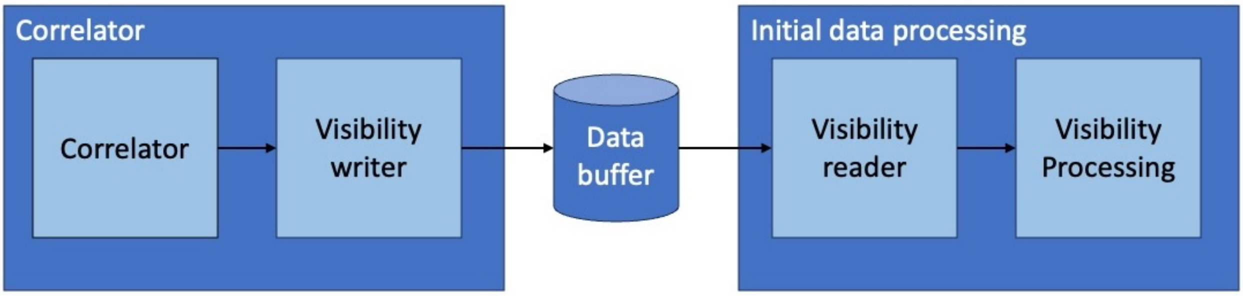

Figure 1 Top-level system diagram of current systems

corrections, compression and (baseline-dependent) averaging. This initial processing often reduces the raw visibility data volume by a significant factor that depends on the observation's needs. This reduction facilitates both archiving of the observation and further processing, like self-calibration and imaging, which is usually more computationally demanding per visibility than pre-processing. Currently, correlators write the visibility data to a storage buffer or, for ALMA, to the final archive, from where the visibilities are read to undergo initial processing that results in a new and smaller set of visibilities that is used to produce the scientific product. The exact data volume reduction will also depend on the data representation, which can vary from 8 bits to 32 bits for data stored in ASDM format (usually half-precision format is used in ASDM while data stored in a Measurement Set is usually represented in single-precision format). This situation is depicted in Figure 1.

The main idea behind streaming visibility processing is to avoid the intermediate storage of the (uncompressed) raw visibility data by performing the initial processing in a streaming fashion. This solution has the following benefits:

- The requirements on the storage buffer size becomes less demanding, reducing the costs for this storage buffer.
- The storage buffer access performance requirements are relaxed, permitting a cheaper and more energy efficient solution.
- Once the first initial processing steps are performed as real-time streaming operations, the system becomes much more flexible as the infrastructure for streaming data processing is in place and adding further functionality in the real-time path has come in reach. It also enables new applications like low-latency imaging for QA as demonstrated in Sec. 8.

To illustrate the first two points, we take LOFAR as an example. The LOFAR correlator produces visibilities with a spectral resolution of 0.763 kHz and an integration time of

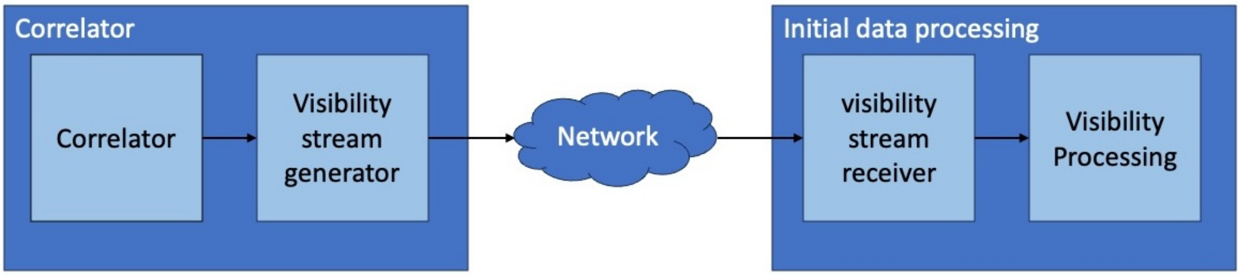

Figure 2 Envisioned top-level system diagram

1 s, mainly driven by the needs for RFI flagging. The LOFAR pre-processing pipeline starts by flagging the data at this high spectral and temporal resolution. After flagging, the data are typically averaged by a factor of 16 in frequency and a factor of 2 in time. Also, lossy compression with Dysco [15] can be performed on the data, which reduces the data volume by another factor of 5 while introducing only a slight increase in system noise. In total, the pre-processing pipeline thus can reduce the visibility data volume by about two orders of magnitude.

The need for a storage buffer for the raw visibility data can be avoided by replacing the visibility writer of the correlator with a process generating a stream of visibilities that can be sent over a network. A visibility stream receiver can receive this visibility stream from the network and feed it into the initial processing steps. This vision is sketched in Figure 2. Besides reducing storage buffer requirements for the buffering of raw visibility data, streaming visibility processing also opens avenues towards (near) real-time initial processing of the visibility data. In this project, this is exploited to demonstrate low-latency imaging for QA by the Astronomer On Duty (AOD).

3 Development plan and deliverables

The project started with a software design study with a twofold purpose:

1. To define a streaming interface with a sufficiently high level of abstraction that it can be made compatible with the streaming data format generated by both the LOFAR correlator and the ALMA correlator with the aim to maximize the synergy between these projects and minimize duplication of efforts.
2. To investigate whether the processing functions currently available in DP3 for common initial operations, like flagging, averaging, phase rotation and direction-independent calibration, need tailoring to ALMA specific needs.

This design study resulted in a design for streaming initial processing of visibilities that can be used as a reference during the implementation stage of this project (***Deliverable D1***). This design document was delivered to ESO on April 5, 2024, and accepted by ESO April 15, 2024 [16]. The design is presented in the next section with some additional details that were uncovered while working towards the next deliverables.

Ultimately, the goal of this project is to demonstrate streaming processing of visibility data. This will be demonstrated by using the ALMA emulator as stand in for the ALMA correlator. With help from Justo Gonzalez (ESO), we have created a callback function in the ALMA emulator that can open a socket to send requested data and meta-data to another process external to the ALMA emulator (***Deliverable D2***). This will be demonstrated in Sec. 5 of this report.

The emulator can be used to support the adaptation of the Default Pre-Processing Pipeline (DP3) software [17][18] that LOFAR uses for initial visibility processing, needed to perform streaming data processing. Streaming data processing will be introduced as an additional mode in DP3. This new feature can thus be merged into the existing Open Source DP3 repository. This repository is being maintained for LOFAR and SKA and, as such, adheres to coding standards with requirements on coding style and quality of developer and end user documentation to ensure maintainability and reusability. The successful adaptation of DP3 is demonstrated by capturing a stream of visibilities from the ALMA emulator and storing those in a valid Measurement Set (***Deliverable D3***) as described in Sec. 6.

More averaging, either baseline-dependent [14] or baseline-independent, will fundamentally lead to increased time and frequency smearing, which, in principle, will cause a decrease in SNR of the observed sources. Dysco [15] is a lossy compression method that will thus also decrease the SNR of the observed sources. Therefore, both methods to reduce the raw data volume come with a small (less than 1% if configured properly) SNR decrease [14][15]. To convince the ALMA science community that this is an acceptable price to pay for a significant reduction in visibility data volume, we need to assess the impact of lossy compression on the final spectra and images produced by the standard ALMA pipelines as well as the performance of the decompression algorithms (***Deliverable D4***). The results of this assessment are presented in Sec. 7.

A potential showstopper for (baseline-dependent) averaging in the ALMA context is the fact that the water vapor density along the line of sight can vary on short timescales (~10 s). Data from water vapor radiometers (WVR) are used for water vapor phase calibration (*wvrgcal*) [19]. Implementing this in a streaming fashion is not straightforward as a second stream with WVR data needs to be aligned with the stream of visibility data and moving averages using sliding windows centered on the current time slice are used. In principle, the visibility buffers in DP3 can be adapted to accommodate this but will require large memory buffers as calculated in Sec. 4.2.1. Removing these short-term gain variations is an enabler for BDA. As Dysco compression can be applied to the uncorrected visibilities, it does not introduce the additional complication of streaming water vapor gain correction and thus provides visibility data volume reduction with lower pipeline complexity. It should also be noted that, currently, there is no variant of Dysco supporting visibility data to which BDA has been applied, so, currently, we can either use Dysco or BDA. It is expected that a future version of Dysco will support BDA. It is also possible to compress the data with Sisco, a lossless compression mechanism with the downside that it reduced the data volume by only ~20%.

As discussed in the previous section, these complications resulted in a re-evaluation of the original project goals outlined in [4] resulting in a shift in focus towards a low-latency imager for QA (***Deliverables D5 and D6*** in [12]), which is described in Sec. 7.2. This section also discusses how to use the low-latency imager in ALMA operations (***Deliverable D7***). The study is concluded by this report (***Deliverable D8***), which presents the outcomes from this study. It is accompanied by the release of a code package containing the features implemented for this study.

The work was organized according to the milestones in Table 1 with deliverables planned as listed in Table 2.

Table 1 Project milestones

Milestone	Description	Date
M0	Kick-off	T0

M1	Project Management Plan (PMP) and design documentation delivered and approved	T0 + 6 months
M2	First-year review	T0 + 12 months
M3	Delivery of assessment report	T0 + 18 months
M4	Updated Minimum Viable Products document including low-latency imager	T0 + 20 months
M5	Delivery of demonstrator	T0 + 25 months
M6	Final study review data package delivery	T0 + 25 months
M7	Final study review	T0 + 26 months
M8	Provisional acceptance of the full scope of work	T0 + 27 months

Table 2 Overview of deliverables

Deliverable	Description	Milestone
D0	Document: Updated Project Management Plan	M1
D1	Document: Design of streaming interface utilizing the ALMA emulator provided by ESO and initial processing steps to be conducted in a streaming fashion	M1
D2	Demonstration Successfully install and run the ALMA emulator provided by ESO using several representative data sets from the public ALMA archive	M2
D3	Code: DP3 refactored to work with the ALMA Bulk Data produced by the emulator. Demonstrate the ability to reliably receive the data via the DDS interface and pass it on to the streaming interface of the DP3.	M2
D4	Report: Assess the impact of lossy compression on the final spectra and images produced by the ALMA pipeline as well as the performance of the decompression algorithms.	M3
D5	Code: Low-latency imager for Quality Assessment (QA)	M5
D6	Report and code: Demonstrator of low-latency imager on ALMA data generated by the ALMA emulator, with as a stretch goal to apply the Water Vapour Radiometer extrapolated corrections taken from the TelCal single flow.	M5
D7	Report: Look ahead document, wherein is described how to use the low-latency QA imager in ALMA	M5
D8	Report and code: Final report and delivery of the code package to an ESO/ALMA software repository.	M6

4 Software design

The aim of this study was to demonstrate the ability to capture a stream of visibilities from the ALMA emulator. As this was a small project, we wanted to maximize use of readily available software, so we decided to use the Default Pre-Processing Pipeline (DP3) software suite [18] for visibility processing. The design of DP3 already focuses on performing operations on visibilities in a streaming manner, so that visibilities are read and written only once. Another consideration was that streaming visibility processing is a useful capability for LOFAR as well.

In view of these considerations, our design focused on realizing a visibility stream interface to DP3 that was suitable for ALMA and LOFAR. This implied that we did not strive for a fully generic solution within this project's context. However, given the significant differences that already exist between ALMA and LOFAR and that the top-level design is quite general, we anticipate that this targeted solution will be suitable for use in other contexts with relatively small modifications.

4.1 Top-level system design with visibility stream

Figure 1 shows the typical top-level system diagram for current systems. On the correlator side, the incoming signals are correlated to form visibilities that are stored to disk by a visibility writer. For example, in the LOFAR case, the visibility writer is a process called OutputProc and runs on the same machine as the correlator software. The data transfer between the LOFAR correlator and OutputProc is handled by a Message Passing Interface (MPI). The correlator sends data in several subbands, where a subband is a set of consecutive frequency channels that can be reduced (or written to Measurement Set) independently. A spectral window can be split into multiple subbands. The path of the visibilities for each subband from the correlator to disk is pre-determined (network interfaces, port numbers, etc.) and the metadata is already written before the actual data flow starts. In the current stage, DP3 processes one spectral window at a time. If combining multiple spectral windows is necessary for S/N, that can, in principle, be done in a next stage. This will likely be an essential feature in the context of the ALMA WSU, which can support 160 heterogeneous SPWs. The non-streaming input variant of DP3 already can combine SPWs into one. A streaming input-step which combines SPWs is conceivable but was not considered in this project.

To demonstrate streaming visibility processing, the visibility writer process of the correlator had to be replaced by a process generating a stream of visibilities that could be sent over a network. A visibility stream receiver could then receive the visibility stream from the network and feed it into the initial processing steps. This is sketched in Figure 2.

To realize this, two aspects need particular attention:

1. As the correlator operates in real-time, continuously receiving new data from the telescope to correlate, the visibility stream generator will effectively have to be a data publisher that can continuously stream visibility data over the network, regardless of whether there is an active subscriber receiving the visibility stream or whether the subscriber experiences hiccups, i.e., the communication between the visibility stream generator and the visibility stream

receiver(s) is one-way communication. Only operations such as starting, stopping and data link configuration can be communicated via a control channel. In view of bandwidth available per link, data may need to be multiplexed over multiple streams. The control channel can be used to configure such multiplexing as needed.

2. As the correlator is part of the telescope system, it is controlled by the monitoring and control system of the telescope. The visibility writer therefore knows or can easily request the meta-data required to be stored in the data products. As the visibility stream receiver will only receive data and a minimum amount of variable meta-data, the visibility processing block in Figure 2 will have to receive (most of) the meta-data, required to perform the visibility processing and to store the results in a Measurement Set, from another source.

In the next sections, we discuss how these aspects were addressed by suitable changes to the DP3 software suite and by defining a suitable interface between the correlator and DP3.

4.2 The Default Pre-Processing Pipeline (DP3) software suite

4.2.1 Structure of DP3

DP3 runs a pipeline in clearly defined stages:

- ***Initialize***: In this stage, all the processing steps are connected in a linked list data structure to create a pipeline, consisting of a sequence of steps called *DPSteps*. Examples of DPSteps are reading data, performing flagging, or writing data. The *initialize* stage is independent of the data.
- ***Get fields to read***: At this stage, the kind of data required by each step (data, flags, weights, (u, v, w)-coordinates) are combined to get a complete overview of what should be read from the input Measurement Set to run the pipeline successfully. For example, if a pipeline does not require the weights, the weights do not need to be read.
- ***Tune At***: This stage reads meta-data, such as the number of channels, and adjusts the steps accordingly.
- ***Process***: This stage pipes the data through the steps, feeding the steps the data for one sampling interval at a time.

Since the design of DP3 revolves around reading visibilities in time order, it is particularly well suited for streaming processing. In practice, DP3 was only used for reading visibilities from Measurement Sets and writing to Measurement Sets. Since this was the only use case, some small deviations from the initial design philosophy had crept in, which were prohibitive for real streaming of visibilities. Hence, some changes in DP3 were required to enable streaming visibility processing.

Data processing in DP3 is done in processing steps referred to as DPSteps. Each DPStep expects two objects as input, a DPInfo object containing the required meta-data to perform processing and a DPBuffer object containing the data to process. While reading visibilities in time order, the reader will produce a sequence of DPBuffer objects, each containing the data for one sampling interval. If a DPStep requires processing on multiple sampling intervals, it accumulates incoming DPBuffers until sufficient sampling intervals have been collected. For example, an Averager step configured to average over 5 sampling intervals will accumulate 5 DPBuffers, averages them into one new DPBuffer and passes that to the next step.

DP3 has been used successfully in other projects to calibrate visibility data over time intervals up to several tens of minutes at the expense of large memory usage (over 100 GByte per compute node). This shows that, in principle, DP3 can buffer data in a processing chain waiting for supporting data, such as data from WVRs. This allows us, in principle, to collect sufficient Precipitable Water Vapor (PWV) data to apply the current sliding window averaging, derive direction-independent gains from the PWV data and apply those gains to the visibilities if sufficient memory is available in the machine performing the streaming data processing. The existing wvrgcal approach operates in batch mode, i.e., given a batch of WVR measurements as input, it produces the phase correction factors for the antennas as output. This can be converted to an online algorithm, where we update the phase correction coefficients continuously, in an online manner. This could be realized by a state-space model of the transform from WVR to correction factors. As new WVR measurements are received this model is updated as well as producing the updated phase correction factors. The state space model can be as simple as a running average/variance or more complicated as a Kalman filter. Even the 'remove cloud' algorithm in the WVR correction essentially boils down to modifying the underlying statistical model that will be used in the online wvrgcal algorithm described above. Due to the cloud effect, the continuum emission has a slope (in frequency) and the wvrgcal algorithm should take this into account when converting the WVR data via path lengths and delays to phase correction factors. Nevertheless, the online wvrgcal algorithm in SVP should be capable of handling this.

Assuming an ALMA observation with 50 dishes and 2 polarizations, each channel and each time dump will produce 5100 real-valued numbers or 10200 Bytes of data represented in half precision. Assuming 595200 frequency channels and a correlator dump time of 3.072 s, this will result in 1.97 GByte of data per second. After receiving the data, DP3 will store the data in single processing format. Buffering the data for 1, 2 or 3 minutes, respectively, will thus require a data buffer of 0.237, 0.474 or 0.711 TByte. Thus, if we want, for example, to buffer data for 1 minute on machines having 100 GByte of RAM available for buffering, we need at least 3 such machines. The advantage of such a setup is that the visibility stream can be split into the same number of streams, which brings down the I/O rate and processing requirements per receiving node as well (with 3 nodes, this would be 5.27 Gbps per node).

4.2.2 Adaptations needed to receive a visibility stream

When an input Measurement Set is available, meta-data that have not been touched by DP3, such as the 'OBSERVATION' table, are copied from the input Measurement Set directly for writing the output Measurement Set. When handling a visibility stream from a correlator, these tables will have to be created from scratch, either without data or with data provided at the *tune at* stage. A new DPStep thus had to be written that could receive visibilities from a streaming protocol and translate it to data and meta-data in the native format of DP3.

It would be impractical to provide all necessary static meta-data via the visibility data stream coming from the correlator. Static meta-data that does not change with time,

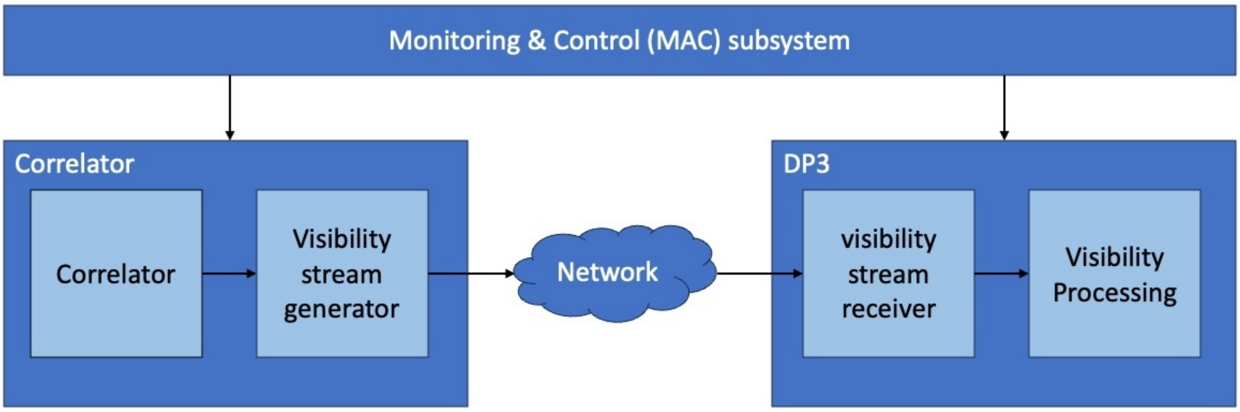

Figure 3 System diagram of proposed solution

should be communicated only once to save bandwidth. For static (meta-)data, DP3 would therefore require a second source of information. A new output DPStep was written that could construct a valid Measurement Set from scratch, i.e., without copying the base structure from an input Measurement Set and could store the static (meta-)data obtained from this second source of information in the constructed output Measurement Set.

Architecturally, the real-time processing of a visibility stream from a correlator operating in real-time as well can be considered as a subsystem of the telescope. It therefore makes sense to assume that real-time processing of the visibility stream is controlled by the telescope's Monitoring and Control (MAC) subsystem. This situation is sketched in Figure 3.

ALMA uses CORBA to control the various components. In such a context, the processing component performing SVP using DP3 can be added as an additional CORBA component to provide management of the input data stream from the correlator, and to enable control of and provide a feedback link from the DP3 process(es), for example to signal problems. It should be noted that the proposed architecture ensures a large degree of independence between the correlator subsystem and the DP3 process(es).

In this structure, the SVP subsystem (running in DP3) has two interfaces:

1. **Interface between correlator and DP3**
   The correlator is a real-time system that is not designed to buffer data. The most logical solution was therefore to have the correlator publish a visibility stream. The receiver of this visibility stream needs to be able to reconstruct the visibility stream even if packets arrive out-of-order or get dropped. The published packets therefore should contain a minimal amount of meta-data to interpret the visibilities in the respective packets, such as timestamp (or packet number) and frequency channel. We considered ZeroMQ to set this up but found that it uses buffering causing the visibility stream to be published in bursts and that it is not thread safe. We therefore resorted to defining an orchestrator on the receiver side with a one-way UDP-based data channel to stream the visibilities (ALMA already uses UDP) and a low-bandwidth TCP-based

control channel to coordinate between the orchestrator and the publisher. Owing to the control channel, the publisher can, e.g., be requested to split the data over multiple data channels to support large bandwidths.

2. **Interface between MAC subsystem and DP3**
   Via this interface, DP3 is set up and static meta-data, such as intended pointing direction and telescope configuration, is provided. The meta-data provided should be sufficient to perform the streaming visibility processing and store the processed visibilities in a self-contained Measurement Set. Time-dependent meta-data other than weights and flags (such as time-dependent pointing or radiometer data) can currently not be handled by DP3, so they were not included in initial demonstrations in this project. However, we designed the prototype in such a way that additional data streams can be added later.
   As the aim of this project was to deliver a proof-of-concept demonstration and the interface with the MAC subsystem is strongly telescope-dependent and not time-critical, we created a simple file-based interface to enable the proof-of-concept demonstration, i.e., we started DP3 manually and provided the required meta-data in files accessible to DP3.

In the ALMA context, it is important to note that Control compiles a number of additional flagging events that only become available at the end of an observation. This limits the number of operations that can meaningfully be done in a streaming fashion on ALMA operations. A detailed analysis of these constraints was outside the scope of this study.

4.2.3 Detailed design of the interface between the correlator and DP3

4.2.3.1 Key design considerations

- The data protocol should not be hardcoded to be that of ALMA or that of LOFAR. This can be achieved by separating the concerns for packet transmission (low-level design) and packet format (high-level design). In such a setup, the high-level design can be easily adapted to ALMA or LOFAR while the low-level design remains the same.
- Apart from a new input DPStep, the infrastructure of DP3 should be touched as little as possible.

4.2.3.2 Packet transmission and stream control

Figure 4 shows an overview of the transmission and reception of the visibility stream. It shows the processing, running in threads, on the ALMA emulator ("TELCAL") and DP3 ("Receiver" and "Worker") side respectively. On the ALMA side, the transmission of data operates as a series of callbacks (event driven). At the start of each blob, the cbStart() callback is executed and at the end of each blob, the cbStop() callback is executed. In between these two, the cbReceive() callbacks are executed to transmit a frame (a frame is a smaller block of data that the blob is divided into, typically 64kB in size). Each frame by itself has no semantic structure and they need to be assembled at the reciever to re-form the blob. An additional transmission of metadata is also performed, which is quite asynchronous to the frame transmission.

On the DP3 side, two threads are used, one for receiving messages from the ALMA emulator, the "Receiver" and another one for arranging the received data in DP3's

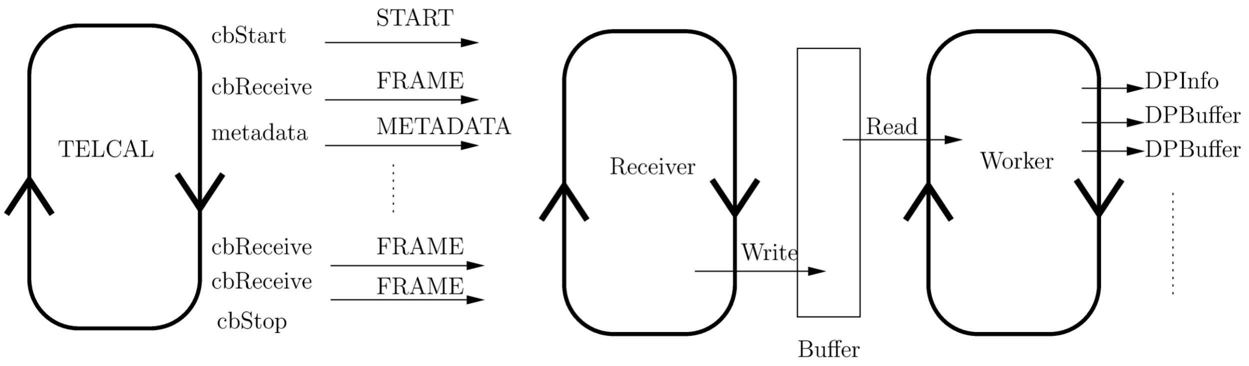

Figure 4 Overview of the processes (threads) running on the ALMA emulator (left) and DP3 (right)

internal meta-data and data structures, the DPInfo and DPBuffer objects respectively. These DPInfo and DPBuffer objects can be passed to consecutive DPSteps for processing of the data and storage in Measurement Set format.

The receiver and the worker share a common buffer to write the received data and to read from it. This enables us to start reading with some delay and can be used to handle packets receiving out-of-order. This is needed to support the creation of the DPInfo object, which requires the meta-data to be parsed first. Unfortunately, that will not happen, because the complete header of the ALMA data needs to be read by the emulator, and the transmission of a Blob (see section below), including its header, is split over several frames. Each frame is received into the buffer by the receiver. The worker will start reading the buffer from its origin at an appropriate time and start parsing the message, extracting the meta-data and data arrays. The binary data arrays will be converted to XTensor objects to be attached to the DPInfo or DPBuffer structures, thereby limiting the number of memory copies to one.

After abandoning the idea of using ZeroMQ for the interface between the correlator and DP3, we started working towards a receiver with the ability to orchestrate the communication by using a UDP-based data channel and a TCP-based control channel. In the current implementation, the socket connections between the ALMA emulator and the receiver are bi-directional, but we only focused on the data flow from left to right in Figure 4. In the future, we can extend this with a control channel to provide feedback to the ALMA emulator, for example for load balancing, configuring multiplexing of the visibility data over multiple streams, increasing wait time, etc. The details require a discussion with the ALMA developers on what improvements will be made to their software in the future.

4.2.3.3 Packet format

Figure 5 gives an overview of the structure of an ALMA Blob. As discussed in the previous section, each Blob will be split into multiple frames that are consecutively streamed over a socket to a destination outside the ALMA emulator. On the receiving side, these frames are put in a buffer that gets parsed by the worker to convert the contents of the ALMA Blob into contents of appropriate DPInfo and DPBuffer objects. In this setup, we do not assume a one-to-one mapping between the contents of each frame and the contents of the DPBuffers, so the publisher does not need to know this to construct the frames.

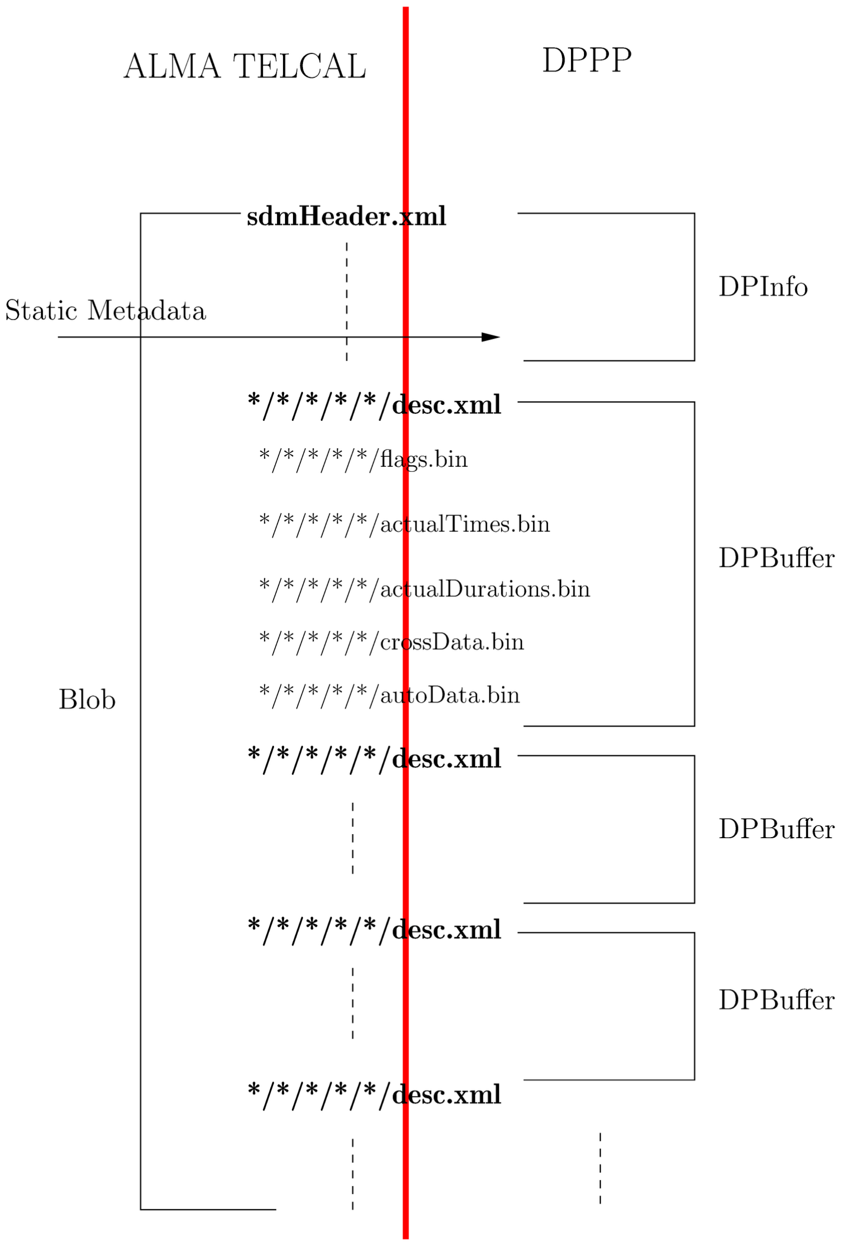

Figure 5 Structure of an ALMA Blob according to the ALMA Data Model (left) and its mapping to DPInfo and DPbuffer objects in DP3.

5 Running the ALMA emulator

The ALMA emulator (TELCAL simulator) provides a data stream that is expected from the ALMA correlator. So instead of running a real correlator, TELCAL enables us to get the input data streams. The main limitation is scaling up to large streams such as a real correlator can produce. TELCAL takes care of this by buffering the entire data set to reach a sustained about rate of 70 MB/s. To run TELCAL with large volumes of streaming data, we need to have a capable computer, with several 100 GB memory and CPU power. We have adapted the TELCAL emulation environment to run only the SVP data manager, thus avoiding the need require a high memory footprint. However, the current ALMA SVP demonstrator only processes channel average data which is a limitation in the sense that processing a high-peak data rate is not

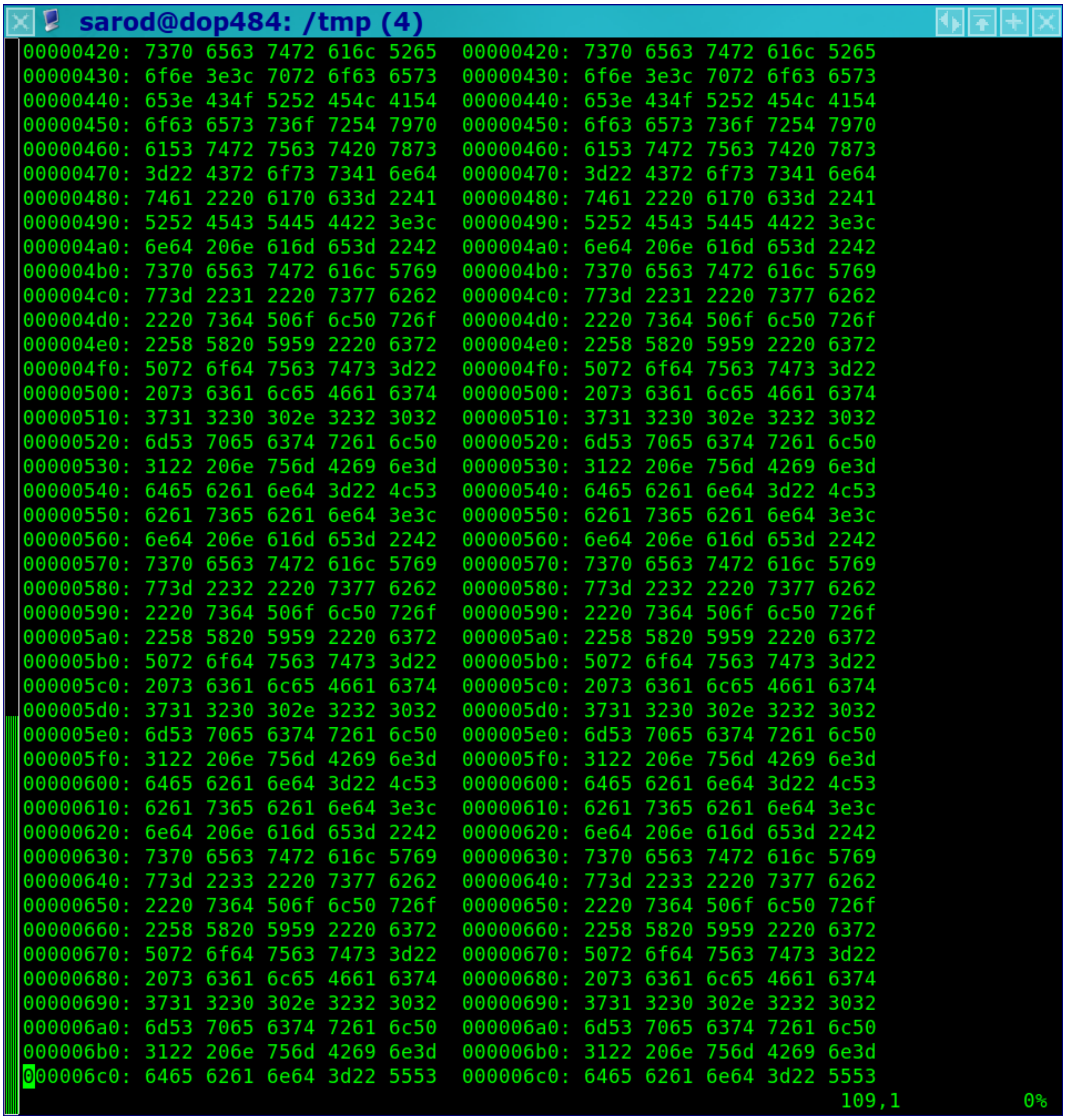

Figure 6 Comparison of data transmitted (left) and received (right) in hexadecimal format

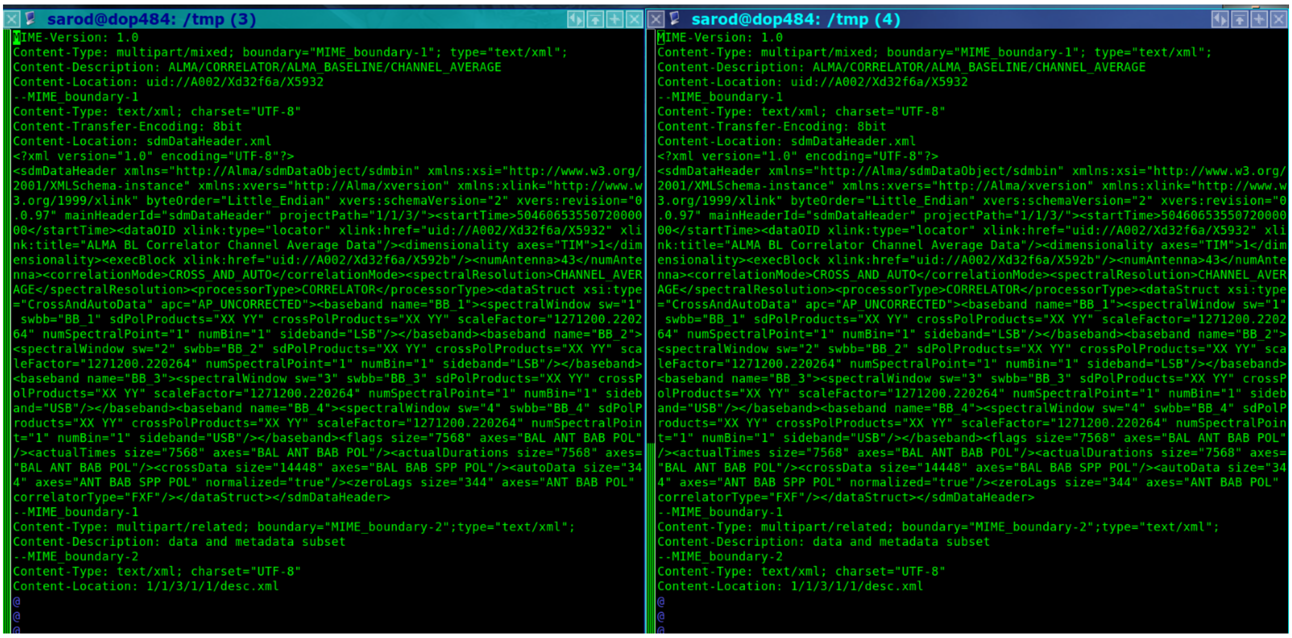

Figure 7 Comparison of the data transmitted (left) and received (right) showing only the text data

exercised. This is also the reason for running tests with the LOFAR correlator to test SVP of large data streams (full spectral resolution) and to check the scalability. In this LOFAR test, we achieved a peak data transfer rate of 100 MB/s while spawning 20 streams received by 20 DP3 processes running in parallel. These DP3 processes performed averaging and writing the data to a Measurement Set.

The TELCAL simulator is invoked within a docker container that is used by the standard ALMA software and clients can connect to the emulator using the network interface from any computer. Each client only needs to know the host name of the computer on which the ALMA emulator is running and the port numbers for which the data streams are set up. Multiple clients can connect to one ALMA emulator to receive multiple data streams, provided that each client uses a unique port to connect to the emulator. The port numbers to use can be pre-specified using, e.g., a configuration file.

To demonstrate Deliverable D2 (Successfully install and run the ALMA emulator provided by ESO using several representative data sets from the public ALMA archive), we installed the Docker container containing the ALMA emulator on a laptop, on which the receiver was also built as stand-alone program, i.e., outside DP3. We replayed ALMA observation with UID A002/Xd32f6a/X592b provided by Justo Gonzalez, captured the frames published by the ALMA emulator by the stand-alone receiver and compared the visibility data sent and received.

There are two ways to test the correctness of the data flow in Figure 4. The first option is to save the Blobs transmitted by TELCAL on the sender and save the Blobs received by DP3, both on disk. We have done this and got 100% accuracy in the reception of the data as illustrated by the screenshots in Figure 6 and Figure 7. We ran similar tests on a few other data sets obtaining similar results. The second option is to import the ALMA data in CASA using the *importasdm* tool into Measurement Set format and compare this with the Measurement Set generated by DP3. This will be done in the next section to demonstrate that Deliverable D3 has been achieved.

6 Receiving ALMA bulk data by DP3

To demonstrate Deliverable D3 (DP3 refactored to work with the ALMA Bulk Data produced by the emulator), we integrated the orchestrator in a new input DPStep. This DPStep uses two threads as shown in Figure 4 to receive the packets from the visibility stream and convert their contents to DPInfo and DPBuffer objects. This input DPStep was connected to an MSWriter output DPStep to produce a Measurement Set. Static meta-data needed to produce the output Measurement Set was provided in the form of an empty Measurement Set (skeleton). This forms a nice starting point for incremental updates to the code in which an increasing amount of static meta-data is extracted from the received ALMA Blobs.

Using the same ALMA observation as before, we validated that the visibilities were received correctly by comparing the visibilities that we replay using the emulator to those that are stored in the output Measurement Set created by DP3. To this end, the data replayed by the ALMA emulator was imported in CASA using the *importasdm* and its visibility content compared with the visibilities in the output Measurement Set.

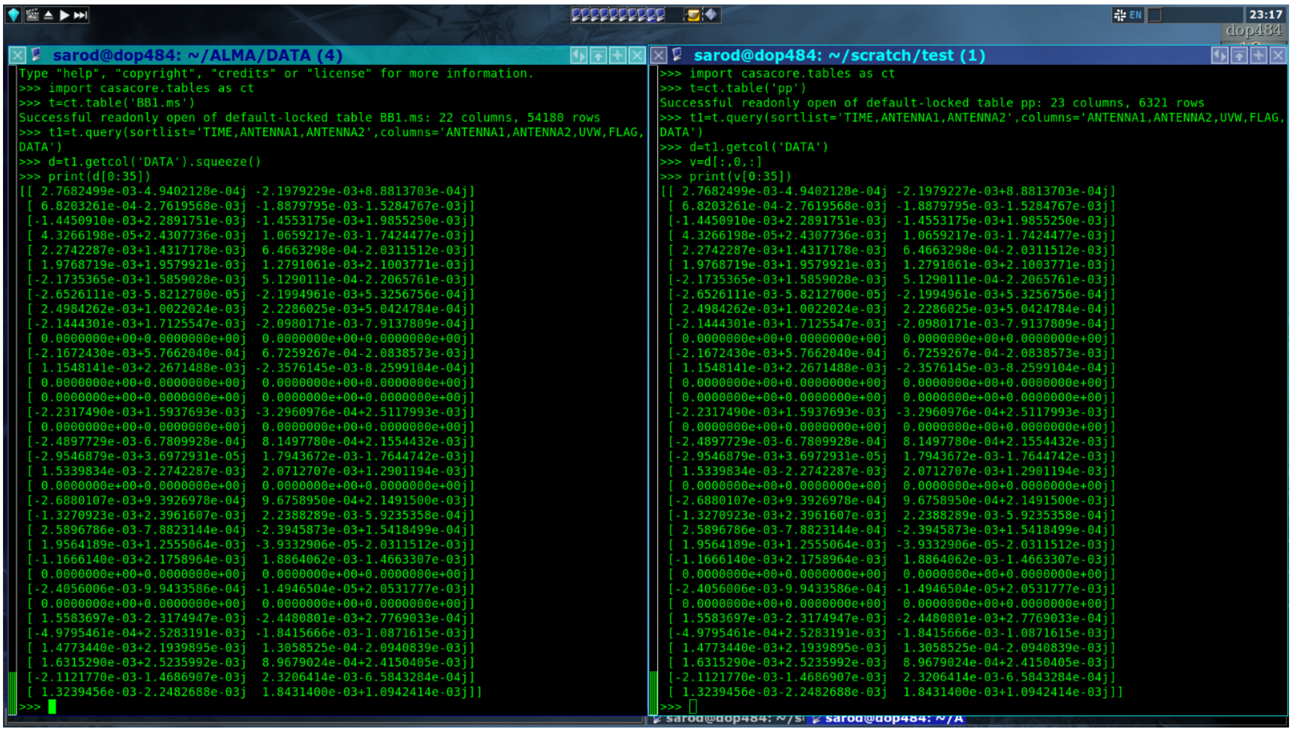

Figure 8 This figure shows a comparison of the original ALMA data (left), converted into Measurement Set format using CASA's importasdm tool, and data written to the output Measurement Set by DP3 (right). Each row shows the visibilities of one baseline.

Figure 8 shows the result of this comparison, showing that we have 100% agreement between the original data and the data streamed into DP3.

One detail about the ALMA Science Data Model (ASDM) binary format which is not entirely clear is the padding that might be present in the 'Content-Location' and the actual binary content. For example, we had to use a padding of 2 bytes to read in the binary cross correlation data to produce the result in Figure 8.

The code used for demonstration of deliverables D2 and D3 is available in the following repositories:

- The adapted ALMA emulator used is located on the ASTRON-Streaming SOC branch of the ALMA git repo at https://bitbucket.alma.cl/projects/ALMA/repos/almasw/browse?at=refs%2Fheads%2FASTRON-Streaming-SOC
- The DP3 version with input DPStep for receiving streaming visibilities can be found on the SVP branch of DP3 in the ASTRON git repository at https://git.astron.nl/RD/DP3/-/tree/svp_imaging?ref_type=heads

7 Impact of lossy compression in ALMA context

7.1 Baseline-dependent averaging

The key idea behind streaming visibility processing (SVP) is that it enables performing initial processing on visibility data before writing the visibilities to disk for the first time as explained in Sec. 2. This becomes particularly attractive if the initial processing steps result in a data volume reduction, for example by Dysco compression [15] or baseline-dependent averaging (BDA) [14]. An assessment of the suitability of these compression techniques in the ALMA context was therefore included as Deliverable D4 of the SVP ALMA development study [4]. In this section, which is adapted from [20], we present the assessment of the usefulness of BDA in ALMA context. An assessment of the suitability of Dysco compression for ALMA observations is described in Sec. 7.2, which is adapted from [28].

As the CASA workflows used to produce the science-ready images available through the ALMA archive are stored along with the data sets, a straightforward test would be to pick a representative data set, apply BDA to it, decompress the data again and then run it through the same workflow as the original data. Such a compression and decompression procedure will be needed anyway for CASA workflows as CASA cannot handle BDA-ed data. After discussions during the first-year review, it was noted that the extent to which BDA can be applied to ALMA data, may be limited due to precipitable water vapor (PWV) conditions on the ALMA site. It was therefore decided to assess the impact of this limit on the potential benefits of BDA first, before testing BDA on ALMA data.

7.1.1 Description of baseline-dependent averaging

After fringe stopping and direction-independent calibration at the phase center, the visibility phase of a source away from the phase center of the observation varies as a function of time and frequency. This visibility phase variation causes decorrelation of the signal when integrated over a finite time or frequency interval, an effect known as time or frequency smearing [21]. The rate of change is proportional to the baseline length. The correlator dump time is therefore chosen such that the time smearing effects on the longest baseline remain acceptable for sources near the edge of the telescope field-of-view, thereby oversampling the visibilities on the shorter baselines. This led to the idea of BDA [14]. For the choice of the width of the frequency channels, other considerations, such as spectral line width, usually play a role as well. Here, we will therefore focus on BDA in time.

BDA reduces the visibility data volume, which saves storage space and bandwidth when transferring data. It can also reduce processing costs required for calibration and imaging [22] as many operations iterate over the visibility data. For this assessment, we will therefore use the visibility data volume reduction as figure-of-merit, so we need to calculate how much averaging is possible for a given baseline length given that the fringe rate is proportional to the baseline length. This implies that averaging over *N* samples is possible if the baseline length *B* satisfies

$$\frac{B^{-\alpha}}{B_{\mathrm{max}}^{-\alpha}} \geq N,$$

where $\alpha = 1$ and B_{max} is the length of the longest baseline. This condition ensures that the decorrelation due to time smearing on the shorter baseline is never more than that on the longest baseline. This condition implies that averaging over *N* samples is acceptable when

$$B \leq N^{-1/\alpha} B_{\mathrm{max}}.$$

This expression provides the most basic BDA scheme. When designing a BDA scheme, other considerations may also play a role. For example, it may be convenient if all integration intervals are aligned on a reasonably short cadence. If this cadence is, e.g., $N_{\mathrm{max}} = 12$, *N* may be restricted to the values 1, 2, 3, 4, 6 and 12. Practically, there will always be a value for the maximum number of samples over which integration is feasible due to, e.g., calibration intervals or scan length.

7.1.2 ALMA-specific limitations

The varying precipitable water vapor (PWV) content of the atmosphere above the ALMA site has a significant impact on the feasible integration time. The observing conditions on the ALMA site have been studied extensively using data from several years of ALMA observations [23]. The results indicate that significant integration of the signal is only feasible after applying gain corrections based on data from the water vapor radiometers (WVRs). The results also show that the decorrelation increases roughly proportional to the square root of the integration time and is baseline dependent with the least decorrelation on the shortest baselines. However, the RMS phase caused by the PWV content scales as $B^{0.21}$ [23] for integration times up to 60 s. Taking into account the proportionality to the square root of time, this implies $\alpha = 0.42$, which is significantly less than $\alpha = 1$ applicable in the case of time smearing due to Earth rotation. To ensure that BDA does not cause more decorrelation than the RMS phase variations caused by the PWV content of the atmosphere, we thus need to assume $\alpha = 0.42$ when designing a BDA scheme for ALMA.

In [24], it was noted that, on the shortest baselines, the RMS phase scales as $B^{0.65}$ for baselines shorter than 500m. They also introduced an offset to fit a power law to the measured RMS phase as function of baseline length. Putting this together results in a more general model for the RMS phase as function of baseline length:

$$\phi_{\mathrm{RMS}} = \begin{cases} \gamma_{\mathrm{short}} B^{-\alpha_{\mathrm{short}}/2} + \beta \text{ if } B < 500 \text{ m} \\ \gamma_{\mathrm{long}}\, B^{-\alpha_{\mathrm{long}}/2} + \beta \text{ if } B \geq 500 \text{ m} \end{cases}$$

As the offset β signifies an RMS phase contribution that is independent of baseline length, it causes a contribution to decorrelation that is independent of time and frequency smearing. For this analysis, we will therefore take $\beta = 0$. Factoring in the square root relation with time, the RMS phase scaling as $B^{0.65}$ on the shortest baselines would imply $\alpha_{\mathrm{short}} = 1.3$, which is steeper than the constraint imposed by time smearing. We will therefore take $\alpha_{\mathrm{short}} = 1$. We will use $\alpha_{\mathrm{long}} = 0.42$ as above.

7.1.3 Results

7.1.3.1 Data volume reduction calculations

The BDA scheme under PWV constraint described in the previous section was applied to various ALMA configurations as used during cycle 12[1]. For each baseline, the feasible level of averaging was determined based on averaging over a single averaging interval on the longest baseline with a maximum of averaging over $N_{\mathrm{max}} = 10$ samples. For Frequency Division Mode (FDM) observations, where the data is already averaged over 6 s, this already implies an integration time of 60 s on the shortest baselines. If data for a given baseline can be averaged over *N* samples, this reduces the data volume for that baseline to a fraction $f_{kl} = 1/N$ where *k* and *l* are the indices of the dishes forming the baseline. We can then calculate the data volume for a given configuration after applying BDA as a fraction of the original data volume as

$$F = \frac{1}{N_{dish}^2} \sum_{k,l=1}^{N_{\mathrm{ant}}} f_{kl},$$

where N_{dish} is the number of dishes in the configuration. Note that this calculation includes the autocorrelations.

7.1.3.2 Results for ALMA only

In this section we present the results for observations using only the array of 12-m ALMA dishes. Figure 9 shows one of these ALMA configurations, the corresponding baseline distribution and the resulting data volume fraction per baseline as an example. The results for all configurations used in cycle 12 are summarized in Table 3 assuming both a simple power law with $\alpha = 0.42$ as well as the more general relationship for the RMS phase as function of baseline length as described in Sec.

---

[1] The Matlab-code used for is available at:
https://bitbucket.alma.cl/projects/ALMA/repos/almasw/browse/TELCAL/SVPCL/BDA_assessment_ALMA.m?at=refs%2Fheads%2FASTRON-Streaming-SOC
https://bitbucket.alma.cl/projects/ALMA/repos/almasw/browse/TELCAL/SVPCL/readConfig.m?at=ASTRON-Streaming-SOC

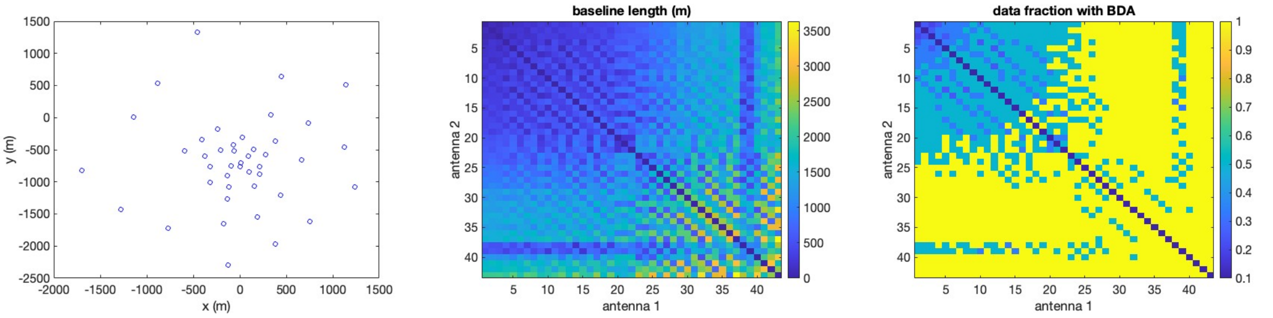

Figure 9 Configuration number 7 if the array of 12-m dishes (left), baseline lengths (middle) and fraction of data left after applying BDA under PWV constraint (right).

7.1.2. To illustrate the impact of the PWV constraint, results for BDA based on fringe rate with $\alpha = 1$ are included for comparison. This shows that the PWV constraint limits the data volume achievable using BDA significantly. Another limiting factor is that the array of 12-m dishes has been optimized to provide a reasonable instantaneous (u,v)-coverage thereby avoiding a strong central concentration of dishes.

Table 3 Data volume reduction achievable with BDA for the array of 12-m ALMA dishes.

		Remaining data volume (%)		
Configuration	B_{max} (m)	$\alpha = 0.42$	$\alpha_{\text{short}} = 1$ $\alpha_{\text{long}} = 0.42$	$\alpha = 1$
1	161	92.6	62.8	62.8
2	314	85.5	41.4	41.4
3	500	82.0	40.3	40.3
4	784	81.4	53.8	41.1
5	1398	79.8	63.4	38.0
6	2517	77.7	72.9	37.8
7	3638	81.2	79.2	39.7
8	8548	72.5	72.1	33.6

7.1.3.3 Results for ALMA combined with ACA

ALMA's WSU will also enable cross-correlations between the signals from the 12-m ALMA dishes and the 7-m dishes of the ALMA Compact Array (ACA). In this section we present the results for such observations combining the two arrays. Figure 10 shows one of these configurations, the corresponding baseline distribution and the resulting data volume fraction per baseline as an example. The results for all configurations used in cycle 12 are summarized in Table 4. To illustrate the impact of the PWV constraint, results for BDA based on fringe rate with $\alpha = 1$ are included for comparison. This shows that the PWV constraint limits the data volume achievable using BDA significantly. Even with this more stringent constraint, the achievable data volume reduction for observations including the ACA is higher than of observations only using the array of 12-m dishes due to the higher fraction of short baselines provided by the ACA, pushing the achievable volume data reduction for the four most extended configurations from 20-25 % to 25-35 %. For the most compact array configurations (configurations 1-4), the achievable data volume reduction is equal to

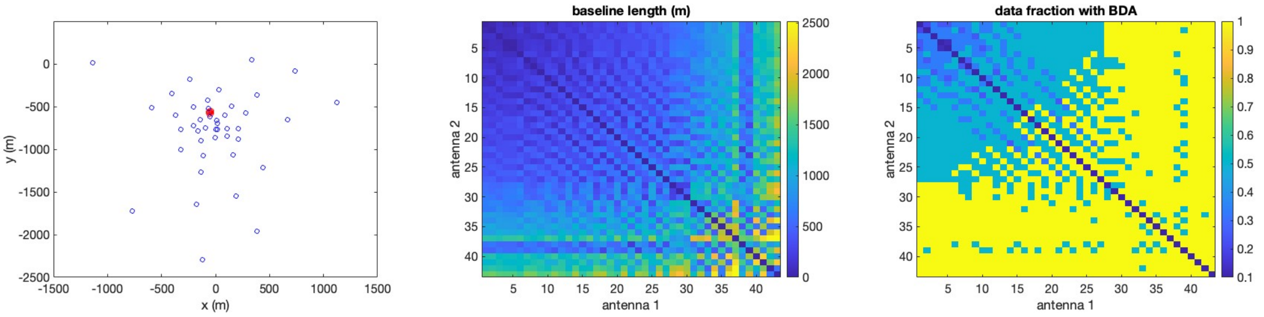

Figure 10 Configuration number 6 of the array of 12-m dishes indicated in blue combined with the ACA of 7-m dishes indicated in red (left), baseline lengths (middle) and fraction of data left after applying BDA under the PWV constraint (right).

or close to the data volume reduction achievable based on fringe rate when the model from [24] is assumed, potentially providing a data volume reduction of about a factor 2.

Table 4 Data volume reduction achievable with BDA for the array of 12-m ALMA dishes combined with the ACA of 7-m dishes.

		Remaining data volume (%)		
Configuration	$B_{\max}$ (m)	$\alpha = 0.42$	$\alpha_{\text{short}} = 1$ $\alpha_{\text{long}} = 0.42$	$\alpha = 1$
1	249	88.5	52.5	52.5
2	338	86.5	47.6	47.6
3	500	83.7	41.5	41.5
4	784	80.4	51.3	38.7
5	1398	75.5	57.8	34.4
6	2517	71.6	65.5	33.2
7	3638	75.0	72.0	34.6
8	8548	65.6	64.8	29.3

7.1.3.4 Summary of results

The calculations of achievable data volume reduction show that the optimization of the array of 12-m dishes for instantaneous (u,v)-coverage combined with the PWV constraint on the BDA scheme limits the data volume reduction to 20-25 % even for the most extended configuration. For comparison, the achievable data volume reduction possible for BDA schemes only constrained by fringe rates is 60-65% for these configurations. For observations combining the array of 12-m dishes with the ACA of 7-m dishes, BDA under PWV constraint can provide a 25-35 % visibility data volume reduction for the most extended configurations. If the model from [24] can be assumed, data volume reductions of about a factor 2 are achievable for the most compact array configurations. We did not consider BDA for observations with only the ACA as the very compact nature of this array will already allow very long integration times on even its longest baselines, limiting the usefulness of BDA.

7.1.4 Testing BDA on ALMA data

We have experimented with an actual ALMA Measurement Set from the stakeholder acceptance tests to test BDA on actual ALMA data. For this experiment, we used a subset of `2017.1.00750.T_tclean_exe1.ms` (uid___A002_Xd20b17_X2f8a_target) with `FIELD_ID 2`, `SCAN_NUMBER 15`, `SPECTRAL_WINDOW_ID 0`. We applied BDA by runing DP3 with the following options

```
msin=subset.ms
msout=subset_bda.ms
steps=[bdaaverager]
bdaaverager.maxinterval=60
bdaaverager.timebase=500
```

Many parts of CASA expect Measurement Sets to be regular, though the Measurement Set specification does not require this. To show the assertions on regularity in CASA, we have tried to run a simple `tclean` command on this Measurement Set. On the uncompressed data set, this runs fine. On the BDA compressed data set, we run into an assertion error. Unfortunately, this assertion error does not point straight to the root cause (the Measurement Set is not regular).

We have tried to work around the fact that CASA does not support BDA compressed Measurement Sets by uncompressing the compressed data into the original data set and running tests on that. Details on this method are described in Secs. 7.2.2 and 7.2.3 on Dysco compression. However, in the workaround we found an issue where uncompressing BDA compressed data to a regular data set (the DP3 step `BDAExpander`), the number of times in de decompressed data is not exactly equal to the original data. In other words, the round-trip regular-BDA-regular does not yield the original data shape, though the 'decompressed' data is regular. The effect of this issue is minor, only one or two correlations are missing. However, this has prevented us from running stakeholder tests to see the effect of running BDA on ALMA test data.

7.1.5 Conclusion and recommendation

In this memo, a BDA scheme was defined that considers the constraints imposed by PWV conditions at the ALMA site summarized in ALMA memo 624 [23] and [24]. This enforces less aggressive averaging than traditional BDA schemes designed to limit the decorrelation due to time and frequency smearing [14]. The impact of this more stringent constraint became apparent when the achievable visibility data volume reduction was calculated for various ALMA configurations: for the four most extended configurations considered, the achievable data volume reductions changes from 60-65% to 25-35% for observations combining the array of 12-m ALMA dishes with the ACA of 7-m dishes. For the most compact ALMA configurations, the achievable data volume reduction is about a factor 2 when the model presented in [24] can be assumed. For the array of 12-m dishes the relative impact of applying the PWV constraint is less than for the array configurations including the ACA due to the (u,v) configuration, which is not strongly centrally condensed. Contrasting with LOFAR, the Dutch LOFAR array is very centrally condensed with 24 core stations located in an area with about 3-km diameter and 14 remote stations outside the core area spread over an area with about 150-km diameter. Owing to this strong central concentration, a data volume reduction of ~4 is regularly achieved for the Dutch LOFAR array.

Although the visibility data volume reduction is less in the ALMA context where PWV conditions need to be considered, significant reductions are still possible, especially for the most extended configurations in combination with the ACA and for the most compact configurations, where the RMS phase drops faster with baseline length. Based on the analysis of site conditions [23][24], BDA must be applied after WVR gain correction. With calibration and imaging software being developed that can process BDA-ed visibilities [22], applying BDA will not only reduce storage requirements for archiving data, but is likely to reduce memory usage and processing resources needed for calibration and imaging as well. However, when the anticipated benefits of visibility data volume reduction mainly lie with data storage in the ALMA context, Dysco compression [15] may be a better option providing a larger data volume reduction. Before using BDA in an ALMA pipeline, further testing on ALMA observations is recommended.

7.2 Dysco

This section describes the results of a simple test to assess applicability of Dysco compression to ALMA observations. It exploits the fact that the CASA workflows used to produce the science-ready images available through the ALMA archive are stored along with the data. We picked a few representative ALMA imaging data sets, compressed them, uncompressed them and then attempted to reproduce the original imaging results by processing the uncompressed visibilities by the appropriate CASA workflow. In this section, we demonstrate Dysco compression in an off-line setting, starting with Measurement Sets. Dysco can also be enabled in the streaming context, by enabling the compression in the configuration of DP3. We have demonstrated this in the first-year review. This completes Deliverable D4 of this study together with the assessment of BDA described in the previous section.

7.2.1 Dysco compression

Dysco [15] is a lossy compression algorithm, with corresponding implementation as a casacore storage manager. The compression of Dysco follows a two-step approach: visibilities of similar amplitude which are expected to have a similar distribution are grouped together, and subsequently quantized. In [15] it is shown that the lossy compression adds a (small) amount of extra noise. For scenarios common for the LOFAR telescope, the extra noise is an acceptable price to pay for compression rates of up to a factor of 6, leading to less storage of visibilities and faster reading/writing of those visibilities in cases where I/O dominates the processing time. The processing costs of Dysco are also sufficiently small to compress visibilities in a streaming context. For example, compression of a LOFAR Measurement Set of 6.8 GByte, which is larger than the ALMA test data sets, took 88 s wall clock time, most of which was spent on I/O.

7.2.2 Test setup

To test the performance of Dysco on ALMA data we have used the data in the ‘stakeholder acceptance tests’ of the CASA software. We have run these tests with CASA 6.7.0. This version is, as of the writing of this report, not yet officially supported for the ALMA pipeline. However, we did not have a suitable machine available for running the officially supported CASA 6.6.0.

One of the stakeholder acceptance tests does not pass when not compressing dysco, therefore we have omitted this test (`test_cal_mfs_IQUV: polarized calibrator imaging - mfs IQUV`). Presumably the reason this test to fail is that we are running on CASA 6.7.0.

7.2.3 Work-around for lack of support for ALMA data format

We have tested with Dysco[2] version 1.3. Dysco does not currently support Measurement Sets with multiple spectral windows, fields or scans. When running the current version of Dysco on a typical ALMA Measurement Set, this will lead to an error message like:

```
This measurement set is not 'regular'; at table row 1510, timeblock
index 150,   timeblock offset 10 the index for antenna1 is not the
same as for previous timesteps. In other words, not all timesteps had
the same baselines. This is required to be able to compress with
Dysco.
```

In principle, Dysco could be altered to support multiple spectral windows, fields and scans. The compression, which happens per baseline, should also happen per spectral window, field and scan, since the noise statistics can be expected to vary between them.

We have not altered Dysco. Instead, we have extracted rows from the test Measurement Sets per spectral window, field and scan, run Dysco on the extracted rows, and copied the result back to the original Measurement Set. In this way, the test Measurement Set holds exactly the data that it would hold if Dysco were compressed on the complete Measurement Set. The experiment is described in the following pseudo-code[3]:

```
for each scan in all_scans.MS:
    1. extract this scan into one_scan.MS
    2. dysco-compress one_scan.MS into one_scan_compressed.MS
    3. copy visibilities from one_scan_compressed.MS back into
       all_scans.MS
    4. add data size of one_scan_compressed.MS to 'dysco_size'
```

For this assessment, we used the default settings of Dysco, which are

- `bits per data val = 8`
- `bits per weight = 12`
- `distribution = TruncGaus with sigma=2.5`
- `normalization = AF`

[2] https://github.com/aroffringa/dysco/

[3] https://bitbucket.alma.cl/projects/ALMA/repos/almasw/browse/TELCAL/SVPCL/run_dysco_test.py?at=refs%2Fheads%2FASTRON-Streaming-SOC

`bits per data val` is the number of bits per float for visibility data. Because a visibility is a complex number, the total nr bits per visibility will be twice this number. The compression rate is n/32. `bits per weight` is the number of bits per float for the data weights. The storage manager will use a single weight for all polarizations, hence with four polarizations the compression of weight is 1/4 * n/32. `distribution` is the distribution used for the quantization of the data. The truncated gaussian and uniform distributions generally produce the most accurate results. The default is `TruncGaus with sigma=2.5`, which is approximately optimal for bitrates 4-8. Alternative options for `distribution` are `uniform`, `gaussian` and `student t`. `Normalization` is `AF` (Antenna-Frequency). Alternatives are `RF` (Row-Frequency) and `Row-normalization`. In this context, a 'row' is a row of a measurementset, i.e. unique combination of time and baseline.

Table 5 shows the compression rates achieved for a number of Measurement Sets. The columns 'Data size' and 'Dysco size' show the size on disk of the columns DATA and CORRECTED_DATA and exclude size of all metadata and weights[4]. We conclude that Dysco on ALMA data, with its default settings, achieves a compression factor of around four.

Table 5 Dysco compression rates for various Measurement Sets

Measurement Set name	Total size (Mb)	Data size (Mb)	Dysco size (Mb)	Compression factor
2017.1.00750.T_tclean_exe1.ms	1409	951	244	3.9
2017.1.00750.T_tclean_exe2.ms	1061	671	174	3.9
2018.1.00879.S_tclean.ms	6127	5786	1449	4.0
E2E6.1.00020.S_tclean.ms	332	237	62	3.8
E2E6.1.00034.S_tclean.ms	187	74	20	3.8
uid___A002_Xfafcc0_X6adf.ms	1536	1219	324	3.8
uid___A002_Xfafcc0_X70fe.ms	1206	915	243	3.8

7.2.4 Test results

Out of the 3290 unit tests, 36 fail when Dysco compression is applied. These are listed below. The second column shows the result after (simulated) compression, the third column (in green) shows the reference result. We should note that many of these tests are very sensitive to numerical noise, in particular metrics like maximum pixel value or image sum. Also, the tests on "`.model im_rms`" measure the RMS on the model image, which is just a metric to detect changes and should not be confused with the noise on the image or residual.

```
Test name                          Output     Reference  Delta
test_mosaic_cube_briggsbwtaper: Mosaic cube imaging wth briggsbwtaper -
field SMIDGE_NWCloud, spw 22
.model max_val                     0.5997     0.5603     -6.6%
.model im_rms                      0.0005396  0.0005318  -1.4%
```

[4] The size of the data is found by determining the index storage manager of the data columns with `showtableinfo`, and then finding the disk size of the corresponding file. For example, if `showtableinfo` gives storage manager index 17 for DATA and CORRECTED_DATA, we determined the file size of `one_scan.MS/table.f17*`

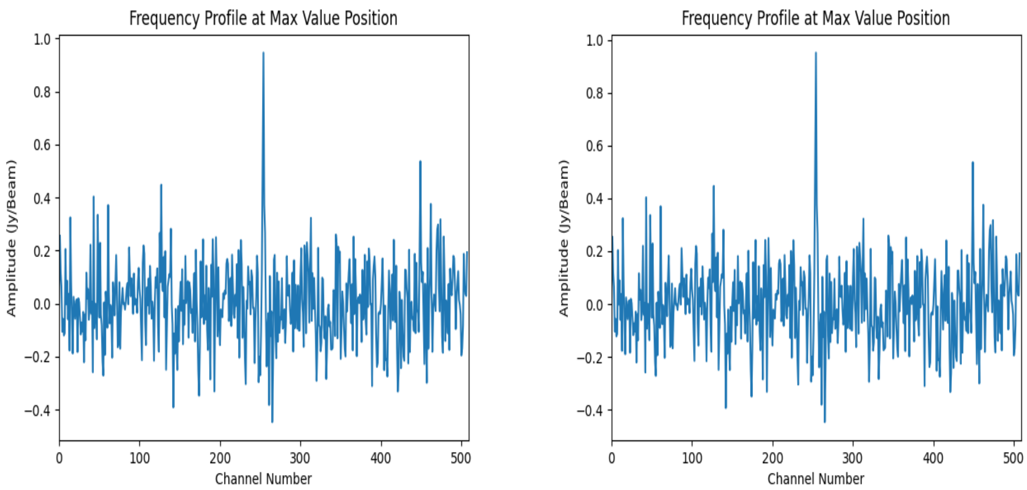

Figure 11 Spectra without (left) and with (right) Dysco compression.

test_mosaic_cube_eph: Mosaic ephemeris cube imaging with pcwdF - field Venus, spw 45

.model regn_sum	0.6414	0.6585	+2.7%

test_mosaic_cube_eph_briggsbwtaper: Mosaic ephemeris cube imaging with briggsbwtaper - field Venus, spw 45

.mask mask_pix	8322	8534	
.mask mask_regns	30	31	
.model max_val	0.03852	0.03946	+2.4%
.model im_sum	7.982	8.168	+2.3%

test_mosaic_cube_eph_pcwdT: Mosaic ephemeris cube imaging with briggsbwtaper - field Venus, spw 45

.model max_val	0.03857	0.03951	+2.4%

test_mosaic_cube_pcwdT: Mosaic cube imaging wth pcwdT - field SMIDGE_NWCloud, spw 22

.model max_val	0.5997	0.5603	-6.6%
.model im_rms	0.0005397	0.0005318	-1.5%

test_mosaic_mtmfs: Mosaic mtmfs imaging - field NGC5363, spw 16 & 22

.image.tt1 regn_sum	0.1299	0.1257	-3.2%

test_standard_cal: Calibrator image - field J2258-2758, spw 22

.residual im_sum	0.1201	0.1221	-1.7%

test_standard_cube: Standard (single field) cube imaging - central field of SMIDGE_NWCloud (field 3), spw 22

.image im_sum	171.54	168.24	-1.9%
.image regn_sum	67.318	66.089	-1.8%
.model max_val	0.3253	0.2860	-12.1%
.model im_rms	0.0002649	0.0002498	-5.7%
.model im_sum	0.9968	0.9263	-7.1%
.model regn_sum	0.9968	0.9263	-7.1%

test_standard_cube_briggsbwtaper: Standard (single field) cube imaging with briggsbwtaper

.image min_val_pos index 0	36	35	
.image im_sum	-26.3613	-25.7062	-2.5%
.residual min_val_pos index 0	36	35	
.model im_rms	0.0004158	0.0004073	-2.0%

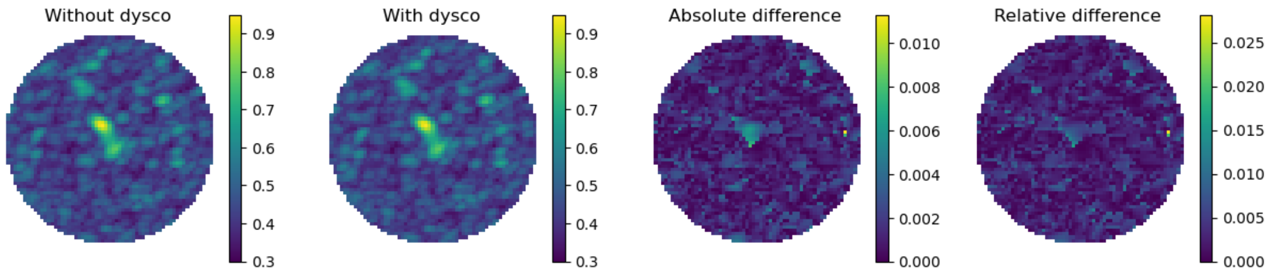

Figure 13 Example (`standard_cube.iter1.image.moment8`) pipeline images with and without Dysco compression. The two rightmost panels show the absolute and relative difference between the images with and without Dysco compression.

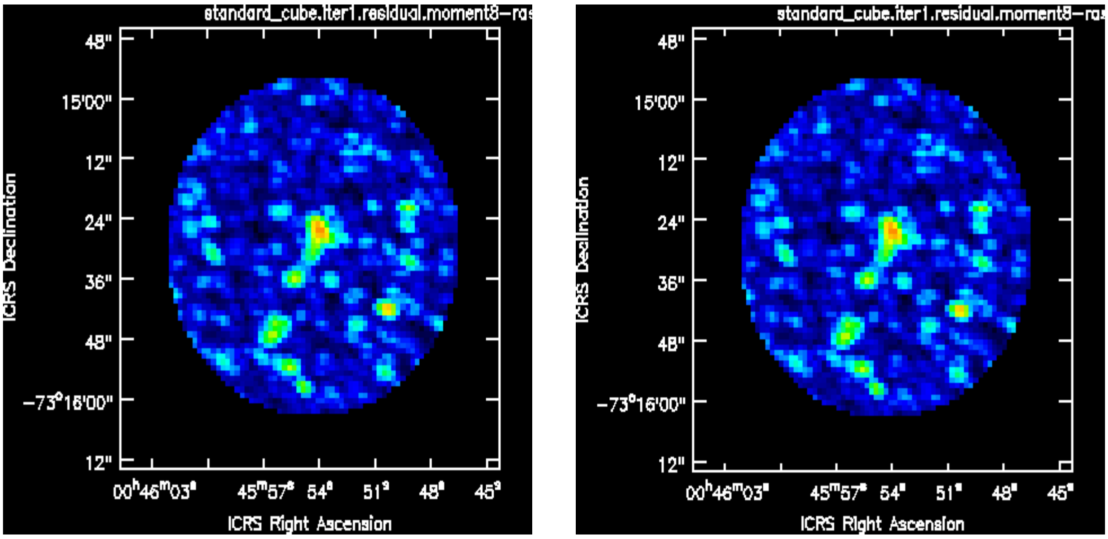

Figure 12 Residual images without (left) and with (right) Dysco compression.

test_standard_cube_eph: Single field multi-EB ephemeris cube imaging - field 21PGiacobini-Zinner, spw 20

.residual im_sum	233.46	236.25	+1.2%
.model min_val	-0.06779	-0.07262	+7.1%

test_standard_cube_eph_briggsbwtaper: Single field multi-EB ephemeris cube imaging with briggsbwtaper

.residual min_val	-0.3358	-0.3310	-1.4%
.residual min_val_pos	False	True	
.residual min_val_pos index 3	491	490	

test_standard_cube_eph_pcwdT: Single field multi-EB ephemeris cube imaging with pcwdT

.image min_val	-0.3757	-0.3665	-2.4%
.residual min_val	-0.3358	-0.3440	+2.4%
.residual min_val_pos	False	True	
.residual min_val_pos index 3	491	490	

test_standard_cube_pcwdT: Standard (single field) cube imaging with pcwdT and briggs

.image min_val_pos index 0	36	35	
.image im_sum	-26.439	-25.784	-2.5%

```
.residual min_val_pos index 0    36          35
.model im_rms                    0.0004159  0.0004073  -2.1%
```
test_standard_mtmfs: Single field mtmfs imaging - central field of NGC5363 (field 2), spw 16 & 22
```
.residual.tt1 im_sum             0.0005382  0.0005101  -5.2%
```

The stakeholder acceptance tests give their output on a 'weblog' page. The full output of these tests is available at https://bitbucket.alma.cl/projects/ALMA/repos/almasw/browse/TELCAL/SVPCL/dysco-weblog.tgz?at=refs%2Fheads%2FASTRON-Streaming-SOC

As an example, Figure 13 shows the resulting images with and without Dysco compression for the test `test_standard_cube`. The corresponding residual images are shown in Figure 12 and the respective spectra obtained at the maximum pixel value in Figure 11. The images are produced by the stakeholder tests without any modification to imaging parameters[5]. These tests have been written to test every aspect of imaging, catching regressions in the development of CASA. Visually, there is no difference between the images/spectra produced with or without Dysco compression with default settings. This holds for all images produced by the stakeholder acceptance tests. We have shown the output of one of the tests as an example. We also inspected the weblogs to see whether there were any differences in the stopping criterion of the `tclean` task and found that `tclean` stopped after a very similar number of iterations for the data with and without Dysco compression and did not reach the maximum number of iterations in either case.

The stakeholder acceptance tests collect many statistics on these images. The statistics are summarized in Table 6.

Table 6 Detailed statistics of stakeholder acceptance tests.

#	Test Name	Passed	Failed
1	test_mosaic_cube	153	
2	test_mosaic_cube_briggsbwtaper	153	2
3	test_mosaic_cube_eph	146	1
4	test_mosaic_cube_eph_briggsbwtaper	146	4
5	test_mosaic_cube_eph_pcwdT	146	1
6	test_mosaic_cube_pcwdT	153	2
7	test_mosaic_mfs	140	
8	test_mosaic_mfs_eph	136	
9	test_mosaic_mtmfs	207	1
10	test_mosaic_mtmfs_eph	203	
11	test_cal_mfs_IQUV	Not run	Not run
12	test_standard_cal	123	1
13	test_standard_cal_eph	119	
14	test_standard_cube	136	6
15	test_standard_cube_briggsbwtaper	138	4
16	test_standard_cube_eph	133	2

[5] A full description of the parameters used by the ALMA stakeholder tests is available at https://open-bitbucket.nrao.edu/projects/CASA/repos/casa6/browse/casatests/stakeholder/test_stk_alma_pipeline_imaging.py

#	Test Name	Passed	Failed
17	test_standard_cube_eph_briggsbwtaper	134	3
18	test_standard_cube_eph_pcwdT	134	4
19	test_standard_cube_pcwdT	138	4
20	test_standard_mfs	123	
21	test_standard_mfs_eph	119	
22	test_standard_mtmfs	189	1
23	test_standard_mtmfs_eph	185	
	Total	325	36

As Figure 13, Figure 11 and Figure 12 suggest a minimal impact on the output products, we wanted to understand better why some of the tests fail. The Dysco article [15] describes that the effect of the lossy compression is similar to extra noise on the observation, i.e. it should not introduce systematic effects. The image RMS on the resulting images can be expected to show an increased RMS.

None of the failing tests is failing on increased RMS. Some failing tests do mention RMS ("`.model im_rms`"), however this is the RMS in model images, which is just a metric to detect changes in test output, like most of the other failing tests, which measure maximum, sum, etc.

Table 7 lists the RMS of images before and after applying Dysco compression. We see that the image RMS increase introduced by Dysco with default compression rate is extremely small, explaining why the impact on the output products of the various pipelines is very minimal.

Table 7 image RMS before and after Dysco compression.

	.image im_rms before	.image im_rms after	increase
test_mosaic_cube	0.087515842	0.087521568	0.0065%
test_mosaic_cube_briggsbwtaper	0.087579295	0.087585492	0.0071%
test_mosaic_cube_eph	0.012333022	0.012333261	0.0019%
test_mosaic_cube_eph_briggsbwtaper	0.011388254	0.011388526	0.0024%
test_mosaic_cube_eph_pcwdT	0.011414423	0.011414643	0.0019%
test_mosaic_cube_pcwdT	0.087583181	0.087589379	0.0071%
test_mosaic_mfs	0.002017806	0.002017967	0.0080%
test_mosaic_mfs_eph	0.672798828	0.672798828	0.0000%
test_standard_cal	0.201288135	0.201286495	-0.0008%
test_standard_cal_eph	0.359356185	0.359356185	0.0000%
test_standard_cube	0.14397977	0.14399189	0.0084%
test_standard_cube_briggsbwtaper	0.141714705	0.141738897	0.0171%
test_standard_cube_eph	0.055886523	0.055880078	-0.0115%
test_standard_cube_eph_briggsbwtaper	0.056903111	0.056918096	0.0263%
test_standard_cube_eph_pcwdT	0.056912387	0.056919864	0.0131%
test_standard_cube_pcwdT	0.141719086	0.141743282	0.0171%
test_standard_mfs	0.00315945	0.003159796	0.0110%
test_standard_mfs_eph	0.359367109	0.359367109	0.0000%
test_standard_mtmfs_eph	0.361678874	0.361678874	0.0000%

Since we have used a work-around for running Dysco on ALMA data, we cannot directly compare runtimes of the test suite with and without Dysco. However, we can measure runtimes required for running Dysco, and the runtime of the total test suite, which gives a good impression of the relative time required for compression and decompression. Dysco compression and decompression were run with dscompress /

decompress version 1.3, using 8 cores (the default). Decompressing dysco compressed data adds virtually no overhead in runtime. The time required to run all tests was 71 minutes, as reported by CASA. Running compression and decompression on the test measurement sets took 14 minutes. This means that dysco compression adds about 20% runtime.

7.2.5 Conclusions and outlook

We conclude that Dysco compression on ALMA Measurement Sets could work when Dysco is altered to support multiple spectral windows, scans and fields. This change would be straightforward, and the numerical performance would be exactly as described in this report, i.e., the compression factor on visibility data with default settings would be around four. We expect this alteration to take ~ 1 month of programming and testing work.

The effect on images and spectra is very small. From the 1% failing test results, it seems that the test failures introduced by Dysco compression are quite small. Some of the failures may be due to a very tight test tolerance.

We did not test different compression settings to verify the compression rate versus the impact on RMS on resulting images. In principle, we expect this to be similar to the data tested in the Dysco paper, but it would be interesting to test this, to balance compression rate against noise increase.

Since 2020, the European ALMA Regional Center allows ALMA users to request calibrated data for a given dataset to be made available for download in MS format, the CalMS service[6]. Dysco compression could be a nice addition to this service as it allows compression of the calibrated MS thereby reducing the need to recalibrate the data to save storage capacity and also reducing the amount of data that needs to be downloaded from the service.

[6] https://almascience.eso.org/tools/eu-arc-network/the-european-arc-calms-service

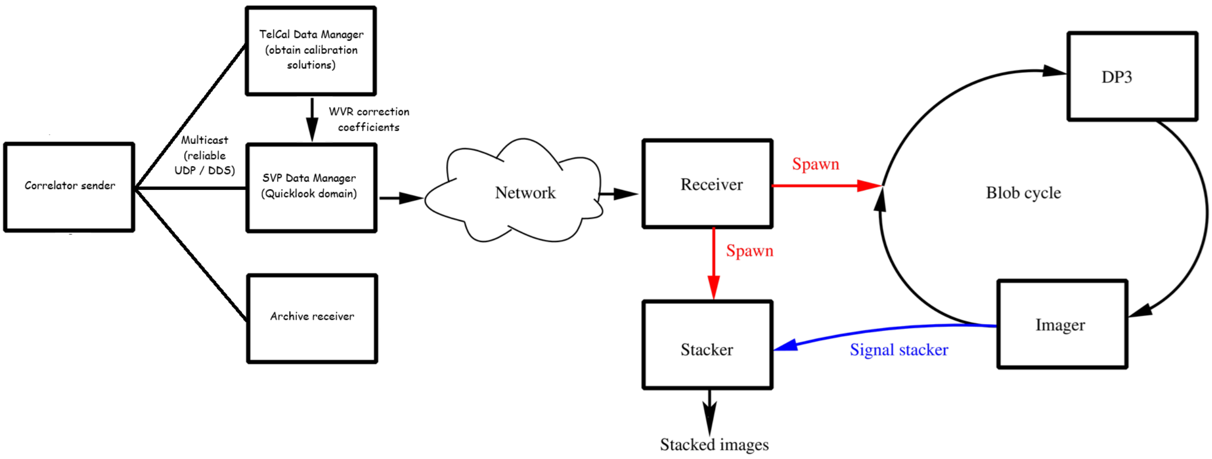

Figure 14 Architecture of the low-latency imager.

8 Low-latency imaging for QA

8.1 Software design

As explained in the introduction, streaming visibility processing for ALMA would require significant data buffering to apply the WVR corrections in an optimal way. As this could make streaming visibility processing impractical, this was discussed at length during the first-year review. More practical calibration approaches requiring less buffering would yield lower quality images. It was proposed that this could be useful for ALMA operators if we could build a low-latency imager that could be used to assess telescope performance and data quality already during on observation, which would be a major improvement over the current situation, where the online RT QL system does not produce images and the QA0 imaging system produces images with a delay after the observation. As a result, it currently may only become apparent a few days later whether an observation was successful or not. Based on these considerations, the goals for this project were redefined to include development of a low-latency imager prototype.

To successfully implement a low-latency imager, several choices had to be made to ensure that the data flow from TELCAL (or the correlator) to the receiver runs uninterrupted and without any delay or data loss. The receiver receives blobs from the ALMA emulator via the network, and these blobs are sent to DP3 for creation of Measurement Sets as demonstrated in Sec. 6. Two additions are needed for imaging, the first is the imager, and the second is the stacking software to stack images made by the imager whenever possible.

The overall architecture of the low-latency imaging software system is shown in Figure 14. There are two additional components to complete the low latency imaging system. First is the Imager (WSClean [25][26] is used as an example) that will produce images from the MS data created by DP3. To avoid unnecessary disk I/O, the MS can be passed from DP3 to the Imager in memory, e.g., by using a RAMdisk. Second is the Stacker that will stack images together whenever possible and do any cleanup of intermediate files.

The SVP Data manager is conceived to run in parallel to TELCAL, using the same bulk data mechanisms as the rest of the ALMA subsystems, all based on DDS. Technically speaking the SVP Data Manager is a DDS bulk data receiver. In this view all DDS bulk receivers (TELCAL, Archive and SVP) run in parallel receiving the data via DDS multicast. The difference comes in what happens after the data is received. TELCAL buffers the data, processes, and sends it back to the Control system, the Archive receiver writes the data to disk and sends that to the archive frontend, and the SVP receiver does not buffer any data, and instead forwards the data to the SVP client machine. The SVP client machine will run DP3 and WSClean.

The Receiver plays the role of the controller in addition to the reception of data from TELCAL. At the beginning, the receiver will spawn the Stacker. Both will co-exist until the termination of the program (or the end of data flow). They will communicate via a Unix socket. Whenever a new blob of data begins to stream, the Receiver will spawn DP3 and the Imager sequentially. When the Imager is done, a signal will be sent to the Stacker about the new image that was just created.

The Stacker will examine the current stack of images and whether the new image can also be added to the stack. If not, the current stack will be written as a new image and a new stack will be initiated starting with the image that was just completed. The suitability for stacking is determined by matching the pointing center coordinates. The stacking will use inverse variance weighting, and the stacked images will be named according to the target (center) name. As the suitability for stacking is determined by matching the pointing center coordinate, the stacking operation is not limited by a low SNR.

8.2 Running the software and results

The low-latency imager can be started with a command like

```
./client -t localhost -p 5555 -d /home/sarod/scratch/software/bin/DP3 -w
/home/sarod/scratch/software/bin/wsclean -b "-mem 40 -scale 0.5asec -size
4000 4000"
```

The crucial information needed are the hostname or IP number (`localhost`) and port (`5555`) to make the connection to TELCAL. Secondly, the full paths to DP3 and WSClean need to be provided. In addition, custom imaging arguments can be provided using the `-b` option.

A caveat when running the low-latency imager is to make sure the resources (especially the memory) of the system are not exhausted (hence `-mem 40` option). The imaging is hungry for memory and if there is not enough memory, one imaging run will cause an extra delay causing a traffic jam of blobs. It is possible to use other imagers that use less memory or use GPU acceleration to avoid this problem. To get a feel for the memory usage of the imager, we imaged one minute of data for one channel (a typical calibrator snapshot) with a pixel size of 0.1 arcsec using various image sizes. The results are collected in Table 8.

Table 8 Memory usage for various image sizes

Image size	Memory usage (MB)
400 x 400	96
800 x 800	128
1600 x 1600	256
3200 x 3200	736
6400 x 6400	2560
12800 x 12800	9824

An initial phase calibration can also be applied before the imaging step, which can improve the quality of the images made. The overhead of the initial calibration is minimal, about 20% extra computations. The way to invoke an initial calibration is to modify the above command with the `-a` option like,

```
./client -t 10.0.0.2 -p 5555 -d /home/sarod/scratch/software/bin/DP3 -a
"steps=[gaincal] msout=. gaincal.sourcedb=phasecenter.skymodel
gaincal.caltype=scalarphase gaincal.applysolution=true
gaincal.overridesky=true" -w /home/sarod/scratch/software/bin/wsclean -b
"-mem 40 -weight natural -niter 300"
```

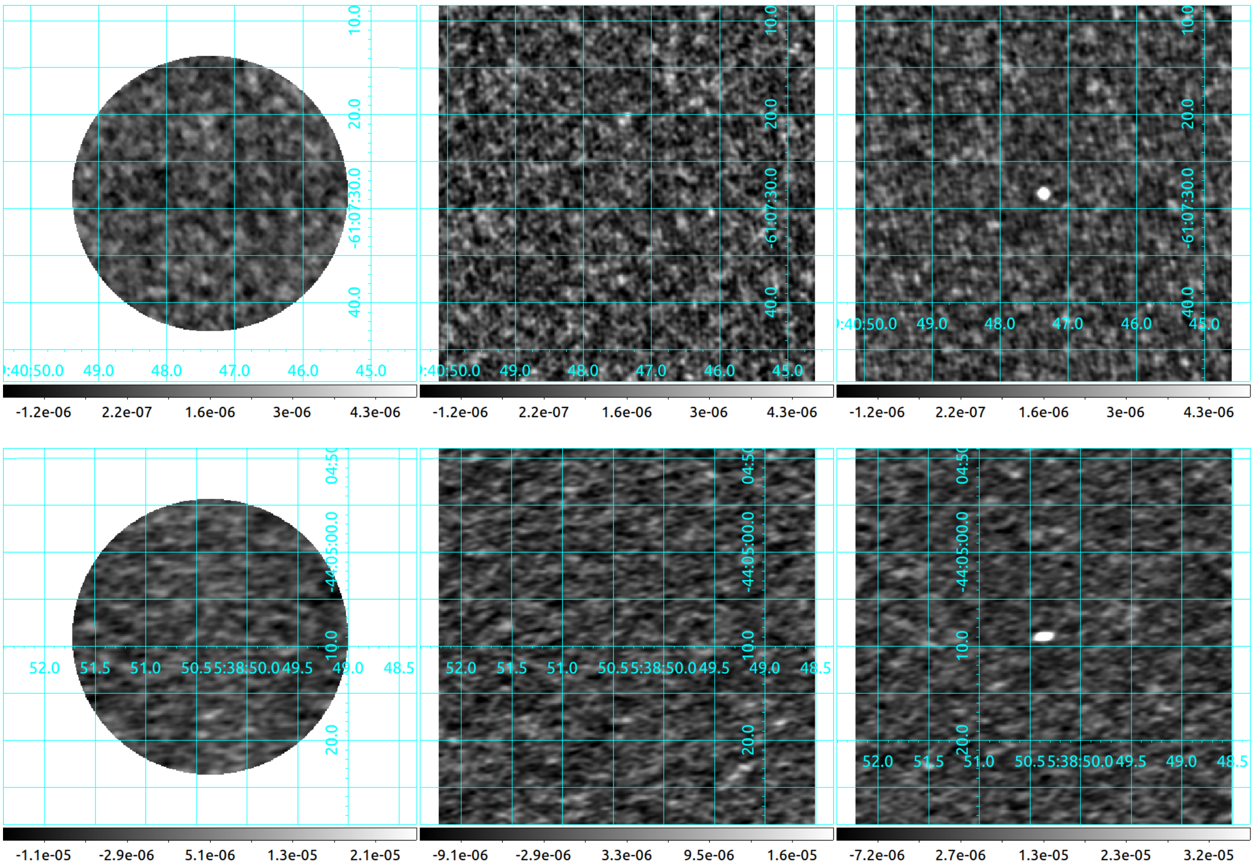

Figure 15 Images made using CASA (left) and made using the low-latency imager without calibration (middle) and with calibration (right) for J0940-6107 (top row) and J0538-4405 (bottom row).

Figure 15 shows images made using CASA (left) and made using the low-latency imager without calibration (middle) and with calibration (right). As only uncalibrated data is available during an observation, the CASA images are uncalibrated and are made by imaging each baseband and stacking the 4 baseband images together (only showing the area of the primary beam). The other two images are made by imaging each blob using all 4 basebands in multi-frequency synthesis and stacking. The top image is for J0940-6107 and the bottom image is for J0538-4405. The phase calibration clearly improves the image as even with low WVR the uncalibrated images do not show the source at the center.

To test the efficacy of our approach for various observing scenarios under varying observing conditions, we tested the low-latency imager on the following three observations:

- `uid___A002_Xedbc85_X9aa0`: A 2.4 GB B4 dataset with a disk (AB_Auriga) taken at PWV 0.191.
- `uid___A002_Xf57b0c_X44cf`: A 66 GB B7 dataset with a disk (SY_Cha) taken at PWV 0.352.
- `id___A002_Xb6d0c2_Xa746`: A 23 GB B4 dataset with a disk (GO_Tau) taken at PWV 0.291 with online WVR-correction included (by extrapolation, from 2016).

Table 9 summarizes the various performance measures while running the streaming imager. The experiment was performed using two computers, connected via a network with 1 GB/s maximum bandwidth. Note that the network bandwidth never exceeded 10% of this value for the streaming hence the network was not a bottleneck. The streaming time was given for 10 time samples and this should be multiplied with the total duration of the observation (~10 min) to calculate the total streaming time. Also, the bandwidth required to stream the full resolution data can be calculated by scaling up the above bandwidth to the full resolution channels.

Table 9 Summary of low-latency imaging performance

Observation	AB_Auriga, 138 GHz	SY_Cha, 300 GHz	GO_Tau
TELCAL Mem	10 GB	32 GB	10 GB
SVP Mem	6 GB	6 GB	6 GB
Stream time for 10-s duration, 4 channels	5 s	5 s	5 s
Runtime for calibration (DP3)	< 1 s	< 1 s	< 1 s
Imaging time (WSClean)	< 1 s	< 1 s	< 1 s
Network peak bandwidth (max 1 GB/s)	10 MiB/s	10 MiB/s	10 MiB/s
Baselines	820	946	861

Figure 16 and Figure 17 show the imaging results obtained for the target field (using averaged baseband data, one channel per baseband) using the low-latency imager including a calibration step for AB_Auriga and SY_Cha respectively. For reference, we also show the images obtained using CASA. The imaging results from CASA show that it is not possible to obtain a meaningful image from the raw data. Therefore, the use of a point source sky model as starting point for calibration was added as a stretch goal to be able to generate low-latency images. We also tried imaging of GO_Tau, a data set for which WVR correction was applied by extrapolation. Figure 18 shows the

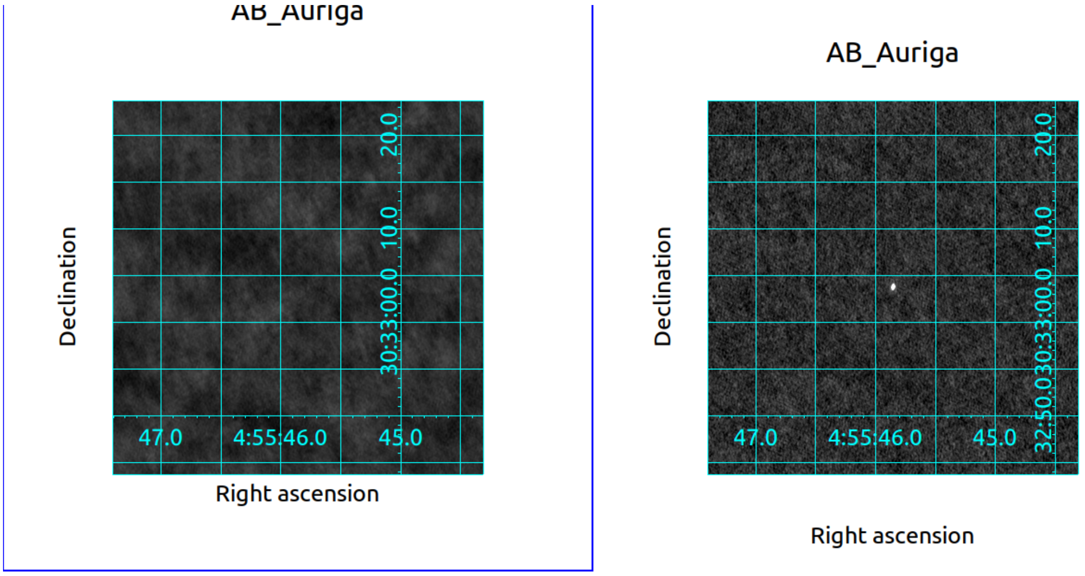

Figure 16 Low-latency imager result for AB_Auriga (right) with the CASA image (left) as reference.

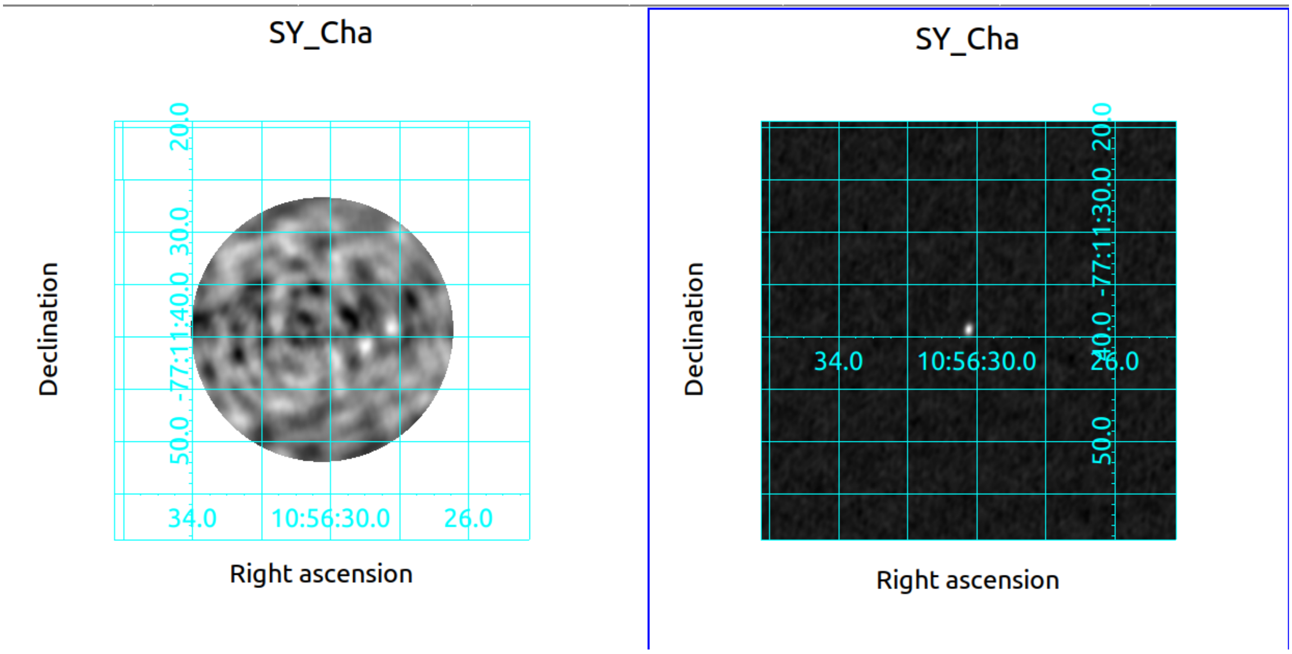

Figure 17 Low-latency imager result for AB_Auriga (right) with the CASA image (left) as reference.

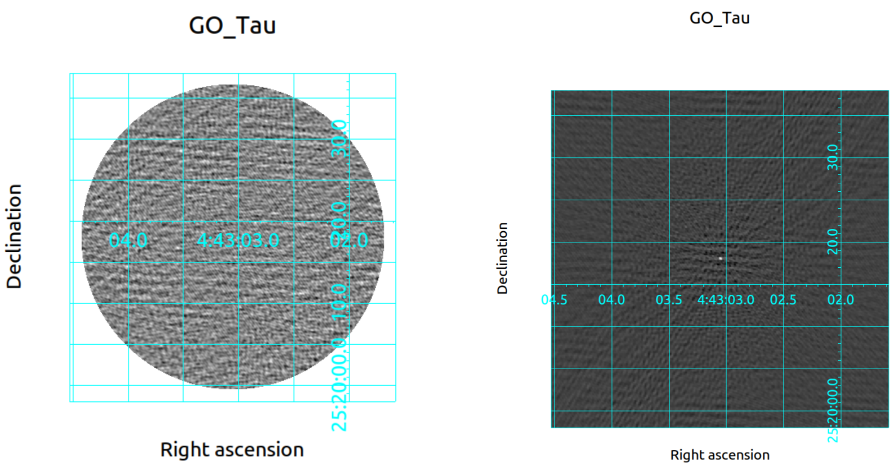

Figure 18 CASA image of GO_Tau observation with WVR correction by extrapolation (left) and with the low-latency imager (right).

CASA result after this WVR corrections, indicating that WVR correction is not enough to obtain a meaningful image.

To demonstrate that the low-latency imager can keep up with the data production rate during regular observing, Table 10 shows timing results for AB_Auriga, which has several calibrator scans interleaved with three target scans. We report the total integration time (summing up the integration time for each field) as well as the streaming and imaging time taken by SVP. It should be noted that the SVP time is dominated by the time needed to stream the data and not by the time taken for calibration and imaging.

Table 10 Detailed timing results for AB_Auriga

Field	Total integration time	Total SVP time
AB_Auriga	12 min	6 min
J0511+2927	15 min (3 x 5 min)	5 min
J0159+2744	1 min	2 min
J0510+1800	4 min	2 min

8.3 Availability of the software

The software required to run the above is in the ASTRON-Steaming-SOC branch of the ALMA git repository [27]. Justo González (ESO) has also made a docker image that will build DP3 and the Imager.

It should be noted that the antenna positions generated by CASA are static (requiring the task `getantposalma` to get updated values) while the ones generated by DP3 are the updated antenna positions (after position calibration) available in the meta-data in the stream.

8.4 Future work

Although some nice results were achieved with the demonstrator for the low-latency imager (Deliverable D5 and D6), further steps are required to make a low-latency imager for QA available to ALMA operators. In this section, we sketch those steps (Deliverable D7).

Improved robustness
In the current implementation, several datasets that were used for testing threw the exception `std exception detected: std::bad_alloc`. This is possibly caused by issues mapping the data and associated flags in the ALMA blobs to appropriately sized DPInfo objects, e.g., with correct actual data duration. The cases in which this occurs need to be understood, so that the software can be made more robust to the various situations that may occur in typical ALMA observations. Ideally a feedback mechanism should be enabled where the client side can send TELCAL messages to cope with such situations.
Estimated effort: 1.5 months

Improved calibration
Calibration in DP3 is currently done under the assumption that there is a dominant point source in the phase center. When there is sufficient data, this is a good assumption to start a self-calibration process if the main source is in the phase center, even if it has a more complex structure. However, in the first calibration pass, this biases the solution towards a point source, especially if there are only few visibilities available. This may be one of the reasons why both calibrated images shown in Figure 15 look like a point source. This bias can be reduced by using more data, for example by combining more frequency channels under a smoothness constraint in calibration, or by using an improved calibration model for the calibrator sources.
Estimated effort: 1 month

Correct visibilities using WVR data
For the demonstration, a few observations were selected with low precipitable water vapor. The current calibration approach may struggle in cases with high precipitable water vapor due to lack of SNR. This can be mitigated by applying a correction based on (extrapolation of) the data sequence produced by the WVRs up to the time of the current snapshot, a strategy that has been applied for ALMA in the past. This would require implementation of a receiver for a second stream of data from the WVRs, aligned with the visibility data stream or application of the WVR correction in TELCAL before the (corrected) visibilities are streamed.
Estimated effort: 2 months

Full stack imaging
In the current scheme, a new image stack is started every time ALMA switches target. As many full ALMA observations involve frequent switching between the calibrator field and the target field, this approach does not result in a full stack for neither the target nor the calibrator field. Ideally, we should keep all stacks open until the end of the observation (although intermediate stacks could still be interesting to retain), so that we can also make a full stack image for the complete observation.
Estimated effort: 2 weeks

Support for multiple streams / flows
The demonstrator processes only a single flow. For the ALMA WSU, we would be interested in processing up to 160 flows simultaneously. Scaling up to handle many flows can be done by encapsulating the aforementioned software by a high-level distributed computing framework such as MPI. The MPI software will launch multiple clients to connect to multiple data flows. This also enables to use a network of computers to receive all data streams, thus distributing the compute load.
Estimated effort: 2 months

Integration with other ALMA subsystems
As a final step towards use of the low-latency imager for QA in ALMA observation, the low-latency imager needs to be integrated with other ALMA sub-systems, particularly the Observation Tool. This would not only smoothen the operation processes but also enables passing of metadata for the next observation to the low-latency imager beforehand. Based on the metadata, certain imaging parameters like resolution and image size can be derived from the observing frequency and configuration. It should already be possible to do the latter by shortly buffering the ALMA blobs in the receiver until the packet with static metadata has arrived and the Imager can be spawned with appropriate settings, but that would be only a temporary solution when further integration is foreseen. This also needs the feedback mechanism mentioned above.
Estimated effort: 3 months

9 Conclusions

The aims of the Streaming Visibility Processing (SVP) study can be summarized as follows:

- Simulate the capture of the visibility output stream from the ALMA correlator for several representative ALMA observations using an ALMA emulator provided by ESO.
- Demonstrate the ability to reliably receive the data via the Data Distribution Service (DDS) interface and pass it on to the streaming interface of DP3.
- Perform initial processing steps in a streaming fashion to produce an equivalent output data set that can be loaded and processed in CASA.
- Assess the feasibility and impact of lossy compression on the final data products produced by the ALMA imaging pipeline. Specifically. Baseline-dependent averaging (BDA) [14] and Dysco [15] compression were considered in this study.
- Demonstrate low-latency imaging for QA.
- Assess the steps needed to bring streaming data processing to production and advise on a possible implementation plan.

This report provides an overview of the progress made during this development study.

Specifically, we have successfully completed the following deliverables:

- ***Deliverable D1*** *(Document) – Design of streaming interface utilizing the ALMA emulator provided by ESO and initial processing steps to be conducted in a streaming fashion*
  This design document was delivered to ESO on April 5, 2024, and accepted by ESO April 15, 2024 [16]. An updated version of the proposed design was

described in Sec. 4, considering the experience gained during the implementation of the next deliverables.

- ***Deliverable D2*** *(Demonstration) – Successfully install and run the ALMA emulator provided by ESO using several representative data sets from the public ALMA archive*
  This demonstration was described in Sec. 5.
- ***Deliverable D3*** *(Code) – DP3 refactored to work with the ALMA Bulk Data produced by the emulator. Demonstrate the ability to reliably receive the data via the DDS interface and pass it on to the streaming interface of the DP3.*
  This was demonstrated in Sec. 6. The code is available under Open-Source license on the git repositories of ALMA (adapted ALMA emulator) and ASTRON (augmented version of DP3).
- ***Deliverable D4*** *(Report) - Assess the impact of lossy compression on the final spectra and images produced by the ALMA pipeline as well as the performance of the decompression algorithms.*
  Reports on lossy compression using Baseline-Dependent Averaging (BDA) [20] and Dysco compression [28] were delivered to ESO. The contents of these two memos were reproduced in Sec. 7 of this report.
- ***Deliverable D5*** *(Code) - Low-latency imager for Quality Assessment (QA)*
  The software required to run the above is in the ASTRON-Steaming-SOC branch of the ALMA git repository [27]. Justo González (ESO) has also made a docker image that will build DP3 and the Imager.
- ***Deliverable D6*** *(Report and code) - Demonstrator of low-latency imager on ALMA data generated by the ALMA emulator, with as a stretch goal to apply the Water Vapour Radiometer extrapolated corrections taken from the TelCal single flow.*
  This was successful achieved, even going beyond the nominal goal by implementing direction-independent calibration using DP3 assuming a point source in the phase centre as calibration model. This is described in Sec. 8 of this report.
- ***Deliverable D7*** *(Report) - Look ahead document, wherein is described how to use the low-latency QA imager in ALMA.*
  The further steps needed to incorporate the low-latency QA imager in ALMA operations were described in Sec. 8.4, including an estimate of the required development effort.
- ***Deliverable D8*** *(Report and code) - Final report and delivery of the code package to an ESO/ALMA software repository.*
  This is covered by this report. The code developed for this project has been reviewed and merged into the appropriate repositories as mentioned in this report.

Appendix A List of acronyms

ALMA	Atacama Large Millimeter/submillimeter Array
AOD	Astronomer On Duty
API	Application Programming Interface
ASDM	ALMA Science Data Model
ASTRON	Netherlands Institute for Radio Astronomy
BDA	Baseline-Dependent Averaging

BL	Baseline
CASA	Common Astronomy Software Applications
CORBA	Common Object Request Broker Architecture
DDS	Data Distribution Service
DP3	Default Pre-Processing Pipeline
DSA	Deep Synoptic Array
ESAC	European Science Advisory Committee
ESO	European Southern Observatory
IF	Intermediate Frequency
I/O	Input / Output
JAO	Joint ALMA Observatory
LOFAR	Low Frequency Array
MAC	Monitoring and Control
MS	Measurement Set
MPI	Message Passing Interface
NAOJ	National Astronomical Observatory of Japan
NRAO	National Radio Astronomy Observatory
PMP	Project Management Plan
PWV	Precipitable Water Vapor
QA	Quality Assurance
RCI	Radio Camera Initiative
RT QL	Real-Time Quick Look
SKA	Square Kilometre Array
SNR	Signal-to-Noise Ratio
SPW	Spectral Window
SVP	Streaming Visibility Processing
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WSU	Wideband Sensitivity Upgrade
WVR	Water Vapor Radiometer
ZeroMQ	Zero Message Queue

Appendix B References

[1] J. Carpenter et al., “The ALMA Development Roadmap,” ESO, June 2018.
[2] J. Carpenter et al., “The ALMA 2030 Wideband Sensitivity Upgrade,” ALMA Memo 621, October 2022. Available at https://library.nrao.edu/public/memos/alma/main/memo621.pdf
[3] S.J. Wijnholds, A.W. Gunst and T.J. Dijkema, “Streaming visibility processing for ALMA - Scientific, technical & managerial proposal,” ESO-516656, January 2023.
[4] C. de Breuck et al., “Streaming Visibility Procession for ALMA – Statement of Work,” ESO-516566, ver. 1.1, August 2023.
[5] M.P. van Haarlem et al., “LOFAR: The Low Frequency Array," A&A, 556, A2, pp1-53, August 2013; DOI: 10.1051/0004-6361/201220873.
[6] www.lofar.org
[7] M.G. Labate et al., “Highlights of the Square Kilometre Array Low Frequency (SKA-LOW) Telescope,” SPIE JATIS, 8, 1, March 2022; DOI: 10.1117/1.JATIS.8.1.011024.

[8] G.P. Swart, P.E. Dewdney, and A. Cremonini, "Highlights of the SKA1-Mid telescope architecture," SPIE JATIS, 8, 1, January 2022, DOI: 10.1117/1.JATIS.8.1.011021.
[9] www.skao.int
[10] www.deepsynoptic.org
[11] www.radiocamera.io
[12] C. de Breuck et al., "Streaming Visibility Procession for ALMA – Statement of Work," ESO-516566, ver. 3, May 2025
[13] http://portals.omg.org/dds
[14] S.J. Wijnholds, A.G. Willis, and S. Salvini, "Baseline-dependent averaging in radio interferometry," MNRAS, 476, 2, pp2029-2039, May 2018; DOI: 10.1093/mnras/sty360.
[15] A.R. Offringa, "Compression of interferometric radio-astronomical data," A&A, 595, A99, pp1-13, November 2016; DOI: 10.1051/0004-6361/201629565.
[16] S.J. Wijnholds, T.J. Dijkema and S. Yatawatta, "Design for streaming visibility processing," ASTRON TC doc. no. 012738, April 2024.
[17] G. van Diepen, T.J. Dijkema, and A.R. Offringa, "DPPP, Default Pre-Processing Pipeline," Astrophysics Source Code Library, record ascl:1804.003, April 2018.
[18] dp3.readthedocs.io/en/latest/
[19] B. Nikolic, S.F. Graves, R.C. Bolton, and J.S. Richer, "Design and Implmenetation of the wvrgcal Program", ALMA Memo 593, July 2012. Available at https://library.nrao.edu/public/memos/alma/memo593.pdf
[20] S.J. Wijnholds, "Assessment of the potential of Baseline-Dependent Averaging (BDA) in ALMA observations," ASTRON TC doc. no. 012894, July 2025.
[21] A.H. Bridle and F.R. Schwab, "Bandwidth and time-average smearing." In: G.B. Taylor, C.L. Carilli and R.A. Perley, "Synthesis Imaging in Radio Astronomy II," Astronomical Society of the Pacific Conference Series, vol. 180, 1999.
[22] C. Salvoni et al., "Calibration and Imaging Pipeline Processing Baseline-dependent Averaged Visibilities," URSI Atlantic and Asia-Pacific Radio Science Conference (AT-AP-RASC), Gran Canaria (Spain), 30 May – 4 June 2022. DOI: 10.23919/AT-AP-RASC54737.2022.9814290
[23] L.T. Maud et al., "Updates to ALMA Site Properties: using the ESO-Allegro Phase RMS database," ALMA Memo 624, April 2023
[24] S. Matsushita et al., "ALMA Long Baseline Campaigns: Phase Characteristics of Atmosphere at Long Baselines in the Millimeter and Submillimeter Wavelengths," Publications of the Astronomical Society of the Pacific, vol. 129, no. 973, February 2017. DOI: 10.1088/1538-3873/aa5787
[25] A.R. Offringa et al., "WSClean: an implementation of a fast, generic wide-field imager for radio astronomy," Monthly Notices of the Royal Astronomical Society, vol. 444, no. 1, pp. 606-619, October 2014. DOI: 10.1093/mnras/stu1368
[26] https://wsclean.readthedocs.io/en/latest/
[27] https://bitbucket.alma.cl/projects/ALMA/repos/almasw/ASTRON-Streaming-SOC
[28] T.J. Dijkema and S.J. Wijnholds, "Assessment of the usefulness of Dysco to compress ALMA imaging observations," ASTRON TC doc. no. 012893, November 2025