REIMPLEMENTING THE BULK DATA SYSTEM WITH DDS IN ALMA ACS

B. Jeram, G. Chiozzi, R. Javier Tobar, ESO, Garching bei Muenchen, Germany
R. Amestica, NRAO, Charlottesville, VA, U.S.A
M. Watanabe, NAOJ, Tokyo, Japan

Abstract

Bulk Data (BD) is a service in the ALMA Common Software [1] to transfer high volumes of astronomical data from many-to-one, one-to-many or many-to-many clients. The main application is in the Correlator, (responsible for processing raw data from the antennas), which retrieves data from antennas on up to 32 computers. Data is forwarded to a master computer and combined to be sent to consumers. The throughput requirement both to/from the master is 64 MBytes/sec, differently distributed based on observing conditions. Requirements for robustness make the application very challenging. The first implementation, based on the CORBA A/V Streaming service [2], showed weaknesses. We therefore decided to replace it, even though ALMA was about to start operations; therefore it was essential to provide for careful testing. We have chosen the DDS (Data Distribution Service) [3] as core technology, because it is a well supported standard, widespread in similar applications. We have evaluated mainstream implementations, with emphasis on performance, robustness and error handling. We have successfully deployed the new BD, making it easy to switch between old and new for testing purposes. We discuss challenges and lessons learned.

INTRODUCTION

The Atacama Large Millimeter/Sub-Millimeter Array (ALMA) [4] is a radio telescope that comprises 54 12-meter and 12 7-meter antennas operating in the millimeter and sub-millimeter wavelength range, with baselines of up to 16km. It is an international collaboration between Europe, North America and East Asia astronomical organizations and is located at an altitude above 5000m on the Chajnantor plateau in the Chilean Atacama desert. Early science operations started in 2011, and the observatory was inaugurated in March 2013. The SW that operates the telescope is built upon the ALMA Common Software (ACS) [1] - a Container-Component CORBA based middleware. ACS provides a set of packages including development tools, common services and design patterns to build and deploy distributed systems.

BULK DATA

The ACS Bulk Data (BD) is an ACS service, which allows transferring high volumes of data using as much as possible of the available bandwidth in different sender/receiver configurations.

The bulk data is used in ALMA to transfer data between 6 ALMA sub-systems: antenna/array control system, two correlator control systems, the archive, the online telescope calibration and the offline system. Two kinds of scientific data go through this service: interferometric data from the correlators (the baseline (BL) and the compact array (ACA) correlators) as well as total power data. A correlator [5] is a specialized computer that, correlates/combines the signals received by an array of several antennas. The correlator can handle multiple arrays simultaneously, and each array produces an independent stream of data.

Data from the correlator are retrieved by clusters of 16 and 32 Correlator Data Processing (CDP) real-time computers respectively for BL and ACA correlators. These nodes process the raw lag data into spectral results, transferred to a central master computer using BD in many to one configuration (one for each correlator). The CDP master combines data from the nodes and forwards them to three receivers: Archive, Telescope Calibration and Real-Time Filler, all located about 30km away, using bulk data in one to many configurations.

Total power data collected from one or more arrays presents another set of data streams which is also distributed to these three types of receivers.

We can have therefore several concurrent data streams. The requirement for the system is to be able to handle a peak data rate for all the streams of 64Mbytes/sec.

BASIC CONCEPTS

The BD was from the beginning designed to keep the underlying details hidden from the developer, hiding the underlying DDS, or A/V behind a generic API [6].

Although the new Bulk Data implementation (BDNT) is based on ALMA requirements, it was designed and implemented to be usable also in other contexts.

The system API is based on the following concepts:

- A ***sender*** is an entity that sends data out.
- A ***receiver*** is an entity that receives data
- Data are transferred from the Sender(s) to the Receiver(s) on a ***flow***.
- One or more flows are grouped inside a ***stream***.
- We refer to each stream and each flow with a name.

A specific data transfer path is identified uniquely by the combination of stream and flow names.

A simple high level protocol is provided in order to:

- Create/destroy streams and flows
- Connect/disconnect to streams and flows
- Inform receivers that data is going to be sent
- Transmit and receive data
- Inform receivers about transmission completeness

Great care has been taken in designing interfaces and implementing code that enables graceful handling of errors in connection/disconnection of partners and in data transmission: a requirement was for a system able to cope with crashes in partners, and with networking problems.

The current bulk data service is implemented in C++.

LIMITATIONS OF THE CORBA A/V

We implemented the first version of BD [6] on top of the CORBA A/V streaming service (see Fig. 1), using the TAO implementation [7]. At the time of selection (more than 10 years ago) the CORBA specification for the A/V service was very promising and TAO provided it. Unfortunately, this implementation did not adequately handle anomalous conditions and errors, and has not been regularly maintained. At present time there are essentially no alternative implementations.

To further complicate matters, during the years from initial implementation to full operation of ALMA, new, unforeseen requirements for BD have surfaced, which are not easily covered using the A/V service.

Thanks to the extensive usage over more than 10 years, we have identified several issues in TAO A/V implementation that are very relevant for our production environment:

- Poor error handling and lack of reliability and robustness in case of instability with some other parts of the ALMA SW. This also made debugging difficult.
- No full thread-safety. The ALMA SW must handle several arrays in parallel and therefore BD must be capable of creating/destroying connections while data transfer on other streams is in progress. Lack of thread safety caused several issues difficult to debug.
- Lack of proper resource clean up makes problematic re-connection and re-initialization of BD applications.
- Optimized for one-to-one transfers. Although this was the original requirement, we "discovered" that we actually needed also one-to-many. We had to re-design BD, increasing complexity, and stretching A/V capabilities.
- Distributor. The TAO A/V implementation provides plug-ins for different communication protocols. UDP makes it possible to multicast the data, but does not work reliably, so we have found it necessary to use TCP. To satisfy the "send data to 3 receivers" requirement we had to develop a "distributor" which on one side receives and on the other side distributes data to many receivers. In order to achieve the required data rate of 64Mbytes/sec we would have had to deploy the distributor on a dedicated machine with at least 4 network cards. This imposes serious limitations on the deployment architecture and reduces flexibility.
- Limitations in stream management. Using TAO A/V we could not implement applications able to handle multiple streams. We had to fall back to separate components per each array: in the case of 6 arrays this means that we would need 6x3 receivers + 6 distributors.
- Rigid connection protocol. A/V requires connecting senders, distributors and receivers in a strict order. This increases the complexity of the system and makes startup and error recovery more difficult to implement.

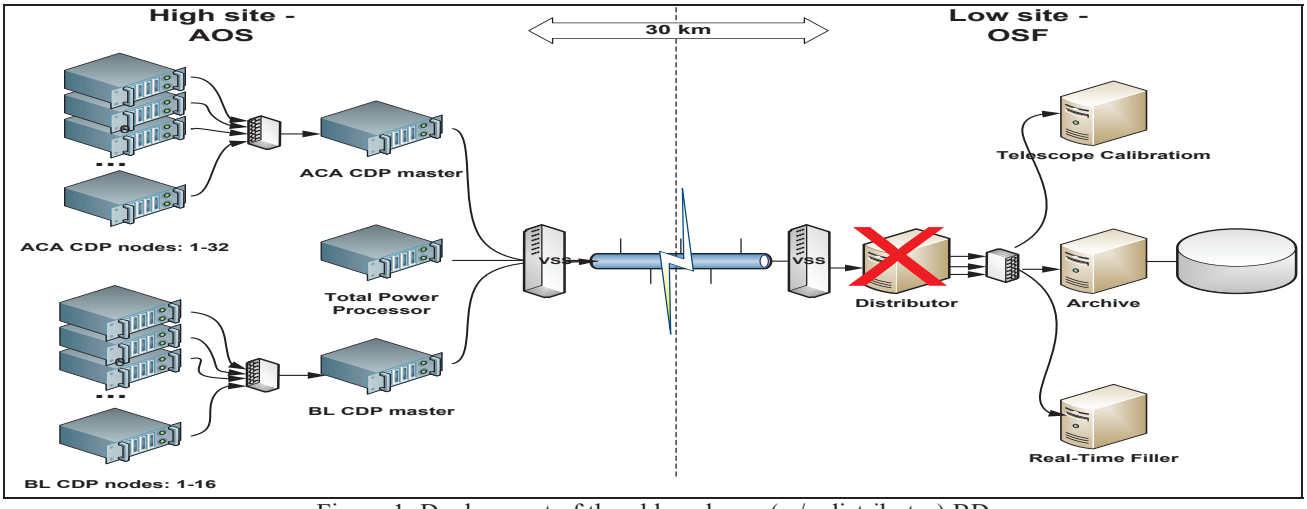

Figure 1: Deployment of the old and new (w/o distributor) BD.

DDS IMPLEMENTATION

These issues drove us to retarget our implementation to a more reliable mechanism, which we refer to as BD New Technology (BDNT).

Since ALMA was already in the early science operation phase, our main concern was to avoid disrupting commissioning and operations by changing a core component of the ALMA SW.

With this in mind we proceeded in the following way:

- Conducted a thorough review of the requirements, significantly changed in the 10 years since inception.
- Investigated possible candidate technologies, by analysing the market, looking at similar projects and developing prototypes. Criteria for selection were:
  - Satisfy the BD technical requirements
  - Mature and widely used technology
  - Well supported
  - Foreseeable long life span
  - Possibly used by others in our organizations
- Selected a specific implementation, based on realistic tests at the operations site.
- Developed a full scale prototype
- Developed BDNT to allow having both new and old systems at the same time, with the ability of switching between the two at run time and of having some parts of the system using the new implementation while other parts were still using the old one.
- Developed an offline suite of performance tests.
- Ported applications to the new platform.
- Organized testing campaigns on the site, deploying and testing the BDNT during technical time and switching back quickly to the old system during operation, to compare the behaviour or whenever requested by the observatory operation team.
- Left it in operation after careful commissioning.

The process, although taking longer than anticipated, has been very successful and BDNT has been put in operation without any major disruption to operations

In line with the criteria mentioned above, we selected the DDS technology, because of its:

- widespread use in institutes as well as in industry and military applications;
- availability of several implementations (free and commercial);
- availability of (commercial) support;
- use by other projects in ESO [8] and already evaluated as a replacement for other ACS services [9][10];
- availability of reliable multicast mode, allowing us to get rid of the distributor component (and dedicated computer) for sending data to multiple receivers at once

In order to select a specific DDS vendor we developed a prototype implementing core features. We tested several products (OpenDDS, RTI, OpenSplice and CoreDX) [11] against the requirements, with particular attention to performance for many-to-one and one-to-many communication (multicast).

Each implementation was extensively tested in particular for the use cases where the old BD suffered the most - error handling, reliability and robustness:

- sudden disconnect (crash) of any sender/receiver(s)
- slow / blocking receiver(s)
- graceful connect/disconnect of any receiver at any time – in particular while transferring data
- simulation of network problems such as slow network or cut cables
- parallel sending of data (flows)

The prototypes also helped us to get a feel for the difficulties in porting an application between different implementations: our conclusion is that the DDS API is not (yet) as well standardized and interoperable as CORBA, but differences are manageable.

We also considered it necessary to test the pure DDS behaviour in the (network) infrastructure of the ALMA operational site, with the simplest possible test applications. The network infrastructure there is quite complex because it has to connect the computers at the high site (the AOS, where antennas and correlators are located) with the computers in the control room at the low site (the OSF, 30 kms away). The traffic has to cross several subnets interconnected with different switches.

As expected, it took some time and experimenting before we could get multicasting working with the required performance. Thanks to a very effective collaboration with IT network specialists and debugging down to the level of RTPS packets with network sniffers, we could solve several real world issues such as:

- configuring network switches to enable multicast and IGMP to prevent flooding the network with multicast traffic
- tuning DDS QoS parameters like the TTL (time to leave) for UDP packets
- properly configuring network cards and drivers (like network bonding) to achieve required throughput for computers that send and receive data
- switching DDS discovery from multicast to unicast to make it work with real-time Linux kernels lacking multicast support

In the end, we managed to multicast data from the high site to 3 computers (with 1Gbit NIC) at the low site at a data rate of more than 900Mbit/sec using RTI and CoreDX.

After evaluating all four different implementations we selected RTI, a commercial DDS. We chose RTI mainly because its implementation proved to be very stable, it provided good performance, and came with good documentation and support.

Having chosen the DDS vendor, we moved on to the actual implementation, introducing several improvements with respect to the old BD, some thanks to DDS and some to our improved understanding of the requirements.

The most significant changes were:

- improved error handling, stability and robustness;
- simplified, more intuitive, API;

- simplified and more flexible deployment, thanks to reliable multicast (no more dedicated distributor computer and distributor components);
- support for more than one stream per component (no longer a separate component set per each array);
- flexible configuration: it is now possible to define default DDS QoS profiles for different environments (operations, testing…). These QoS settings can be overridden for each stream and flow in the configuration database, where the streams and flows are defined;
- the connection order does not matter anymore and it is very easy to add a new receiver, for example to dump the data for testing purposes;
- built-in support for trouble shooting; several mechanisms were added for (early) detection of problems with applications: checking (average and instantaneous) receiver's data processing time, stricter checking of the BD high level protocol, logs containing protocol statistics (packets and heartbeats sent, packets resent …), several configurable levels of debug log messages, giving finer control over the details and volume of log messages;
- availability of generic bulk data sender, receiver and receiver-sender (to simulate the CDP master) applications. These tools allow simulating different scenarios and loading conditions at any time. Such "synthetic tests" have proven extremely useful during the testing and in particular during the deployment of the software based on the new BD.

At this point the new BD could be given to the subsystems development teams for application porting and to run/extend their subsystem tests. During this period we provided support and made improvements and bug fixes, thanks to the valuable and timely feedback from the application developers.

After the porting of all the subsystems had been completed, we handed over the system to the integration and testing (ITS) team and we started testing the whole ALMA SW system with the new BD during technical time at the operational site. The main emphasis in the first tests was on the performance and robustness of the system as a whole, based on a list of tests covering in particular use cases that had caused problem with the old BD. The robustness test passed w/o problems, giving us confidence that the bulk data implemented with RTI DDS was very stable, robust and flexible. For example, it was now possible to recover from a crash of a component without having to restart the whole system.

When we started running observations during early science time, we also discovered some problems. For example, we noticed that in certain conditions the data rate was dropping, triggering timeouts (gracefully handled by the new implementation). It turned out that the problem was due to delays introduced by excessive logging in the application code of one of the receivers, introduced during the porting process.

This showed again that it is essential to look at the system as a whole - to evaluate the whole bulk data chain from deployment to application code.

We felt into another interesting trap when the full correlator was brought online and we went from 4 to 16 CDP nodes. At that point the new bulk data was configured to use reliable model for both: uni-cast UDP for many-to-one, and multi-cast UDP for one-to-many. We were getting unexpected timeouts when all nodes were sending out data at the same time to one computer with high throughput. We understood what was going on by analyzing the network traffic. The reliable model is implemented on top of UDP (intrinsically not reliable) where the acknowledgement mechanism (sending NACKs/ACKs) uses UDP as well. If the load of the network is very high, UDP datagrams get lost and many retries are needed, further increasing the load: at the end the performance drops dramatically. TCP instead (intrinsically reliable) handled these situations much better. We could test our supposition very easily just by changing the QoS configuration database to enable the TCP protocol for the many-to-one case. Now the CDP master (the part in the middle) works as receiver using the TCP protocol and on the other side as sender using multi-cast UDP.

CONCLUSION

We have been able to successfully replace the infrastructure used in ACS for a critical service like the Bulk Data without hindering the commissioning and early science operations of the observatory.

Although we could so far test the system with what is available: 54 out of 66 antennas, and 50% of the correlator data rate, we can be confident because we already see big improvements with respect to the old BD, in equivalent conditions. Actually we could only test the old BD with up to 40 antennas. As antennas were added to the array before the BD upgrade we had periodic problems that were very complex to debug, accompanied by system instabilities and could only be solved with considerable struggle.

This work has demonstrated that DDS has a very good and solid architecture, solving our problems, and the quality of commercial implementations if very high.

We have also learned from the issues that arose while commissioning highly distributed systems in a complex network infrastructure. No matter how high level your tools are, you need a good understanding of what is going on under the hood and the right instruments to analyze it: long live Wireshark!

REFERENCES

[1] G. Chiozzi et al., "ALMA Common Software (ACS), status and development", ICALEPCS'09, Geneva, TUP101.

[2] http://www.corba.org

[3] http://portals.omg.org/dds

[4] A. Farris et al., "The ALMA Telescope Control System", ICALEPCS'05, Geneva, WE3A.3-6O.

[5] J. Pisano et al, "ALMA Correlator Real-Time Data Processor", ICALEPCS'05, Geneva, PO2.067-4

[6] P. Di Marcantonio et al, "Transmitting Huge Amounts of Data: Design, Implementation and Performance of Bulk Data Transfer Mechanism in ALMA ACS", ICALEPCS'05, Geneva, PO1.032-6
[7] N. Surendran et al., "The Design and Performance of a CORBA Audio/Video Streaming Service", Proceedings of HICSS-32 vol. 8, Hawaii 1999, 8043.
[8] R. Karban et al., "Towards a State Based Control Architecture for Large Telescopes: Laying a Foundation at the VLT", ICALEPCS'11, Grenoble, FRBHMULT04
[9] J. Avarias et al., "Data Distribution Service as an Alternative to CORBA Notify Service for the ALMA Common Software", ICALEPCS'09 , Kobe, WEA006
[10] J.Avarias et al., "Introducing high performance distributed logging service for ACS', SPIE, San Diego, SPIE 7740.
[11] http://en.wikipedia.org/wiki/Data_Distribution_Service#List_of_Vendors