STATUS OF ALMA SOFTWARE

T.C. Shen, J. Ibsen, R. Olguin, R. Soto, ALMA, Santiago, Chile

Abstract

The Atacama Large Millimeter /submillimeter Array (ALMA) will be a unique research instrument composed of at least 66 reconfigurable high-precision antennas, located at the Chajnantor plain in the Chilean Andes at an elevation of 5000 m. Each antenna contains instruments capable of receiving radio signals from 31.3 GHz up to 950 GHz. These signals are correlated inside a Correlator and the spectral data are finally saved into the Archive system together with the observation metadata. This paper describes the progress in the deployment of the ALMA software, with emphasis on the control software, which is built on top of the ALMA Common Software (ACS), a CORBA based middleware framework. In order to support and maintain the installed software, it is essential to have a mechanism to align and distribute the same version of software packages across all systems. This is achieved rigorously with weekly based regression tests and strict configuration control. A build farm to provide continuous integration and testing in simulation has been established as well. Given the large amount of antennas, it is imperative to have also a monitoring system to allow trend analysis of each component in order to trigger preventive maintenance activities. A challenge for which we are preparing this year consists in testing the whole ALMA software performing complete end-to-end operation, from proposal submission to data distribution to the ALMA Regional Centers. The experience gained during deployment, testing and operation support will be presented.

OVERVIEW

ALMA software [1] is a very complex system, developed by distributed teams across three continents. Every six months a new release takes place and once it is deployed at the Operation Support Facility (OSF), it is responsibility of the ALMA Department of Computing (ADC) in Chile [2] to take care of its maintenance. ALMA has scheduled for Sep 30th of 2011 to start operating in the so called "Early Science" mode. In preparation for such a big challenge, ADC has designed and implemented an entire support system around the core ALMA software in order to successfully achieve this important milestone.

In the following sections a description of the software verification and maintenance processes, together with the required infrastructure will be described.

ALMA OPERATION SUPPORT PROCESSES

Array Element Configuration

ALMA is composed of 66 antennas, photonics references, Correlators and control computers which need to be configured according to the science verification purpose. Due to ALMA still being under the construction phase, reconfiguration of the system is an activity run on daily basis.

ALMA observation software saves the system configuration variables in the Telescope Monitor and Configuration Database (TMCDB), a relational database built on top of Oracle R11. Each array element has an average of 1500 variables, in addition to the deployment information. It is unviable to do it manually, (as the first antenna has been done) and a template based approach has been developed, allowing to configure a new element in matters of minutes.

As soon as the number of antennas increases, it is hard to keep track of their status and deployment configuration. Several web based applications were developed to depict live deployment information.

Regression Tests

A new version of ALMA software is released every six months. The new release is deployed and verified on site together by North American, European and local software staffs. After the acceptance, the local software group is responsible for supporting and maintaining this new production release, together with guarantying the system's stability. Modifications in this operational release are driven either by software patches for bugs fixing or by adding new functionalities. A new weekly iteration of this release is deployed at the control computers in order to consolidate these modifications and a battery of regression tests is executed to guarantee system reliability.

The regression tests suite aims to verify the software functionality through the execution of the same tasks performed during science operations. Additionally, tests focused on the validation of the ALMA software deployment and are also executed.

The pass criteria applied for every test is simple and atomic. The idea is to avoid misinterpretation about the test completeness. For example:

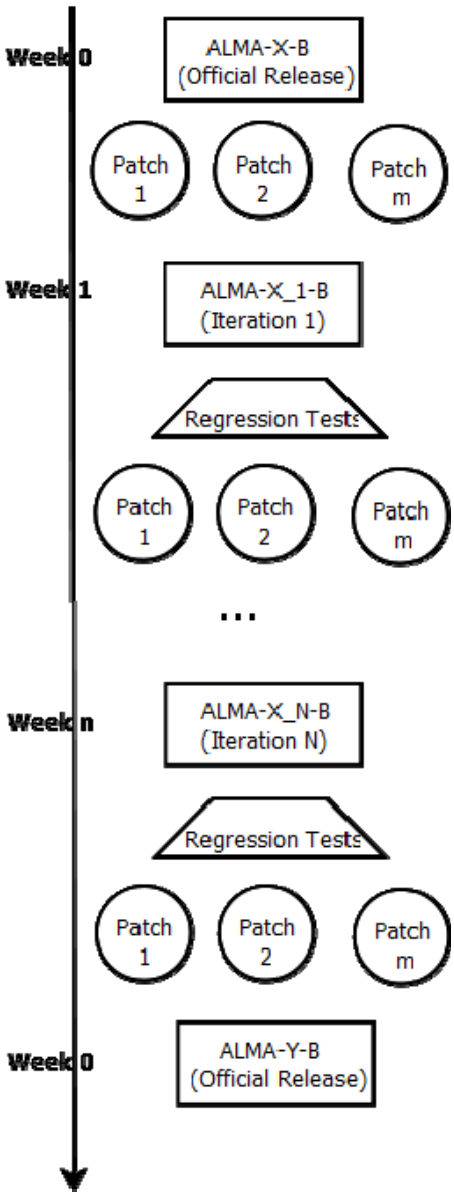

Figure 1: Regression Tests Workflow.

Description: *Create a manual array*
Test Execution: *Within the OMC, open the "Create Array" plugin. Create a manual array by selecting all available antennas, a Photonic Reference and a Correlator. (if another array was created, dispose of it previously)*
Pass Criteria: *A manual array has been created and all available antennas are assigned to it.*

Despite most of the tests being executed and the results verified manually, ADC has been working on the automation of the tests suite. A framework based on pyUnit has been implemented in order to allow the automatic execution of the tests and the verification of the passing criteria. The "Auto Regression Tests" framework takes the information from an XML configuration file, in which a test is defined. Every tests suit consists in: (1) cleanly initializing the system, (2) allocating the required resources, such as: the selection of available antennas, photonic references, and Correlators, (3) executing the specified tests, (4) releasing allocated resources and shutdown of the whole system, and (5) reporting the results. In cases, when the test suite collects scientific data, the data must be analyzed in order to verify the coherence of the observations. To cover this area, validation routines have been developed in collaboration with the science team, which in turn basically reduce the raw data by means of tools such as TPOINT, CASA, etc. Upon the event when a Boolean result is not easy to achieve visual inspections will be required.

In conclusion, regression tests provided an objective and reliable way to verify the system functionalities after the deployment of new software iteration. However, this process requires working continuously on the upgrade of test suites as scientific scripts evolve and new features being introduced in the system.

In the near future, the goal of this is to execute the same tests suite across all ALMA software development centres in order to verify and validate a fresh release.

End-To-End Tests

Aside from the weekly regression tests, which focus on the observation software, a complementary verification is done in parallel, which focuses on the end to end aspect. These tests start with (1) proposal submission phase, (2) proposal selection phase, (3) proposal scheduling and observation phase, (4) observation data replication from OSF to Santiago data centre and (5) data replication to ALMA regional centres.

Software Patch Workflow

The workflow starts with (1) bug detection and reporting, (2) debug and troubleshooting process, (3) patch preparation. (4) Control Board approval, (5) testing, (6) deployment in the operational environment, (7) and consolidation for the next software iteration.

ALMA uses JIRA as the main tickets reporting system. Together with JIRA, several custom data mining tools were developed to produce periodically statistics about downtime and the ticket associated. Other key process indicators are also generated in order to provide sufficient information to allocate efficiently the available resources.

Scalability Tests

These tests aim to validate in advance the capacity of ALMA software to deal with all the array elements when it is completed. Tests are focused on (1) diskless real-time machines booting process, (2) CORBA notification services, (3) TMCDB access during software start up, (4) monitoring system, and (6) dynamic resource allocation during observation.

Currently ALMA has 20 antennas integrated into the array. Scalability showed that the current software scales correctly up to 30 antennas, but problems start to occur

when the 40 antennas boundary is crossed, which is the number of antennas expected at the end of 2012.

ALMA SOFTWARE SUPPORT INFRASTRUCTURE

In order to successfully and efficiently perform the aforementioned processes, several computing infrastructures are required. In the following sections key infrastructures are depicted.

Virtualization

The virtualization technology has been the foundation to provide computing power in ALMA. Starting from 2009, several virtualization technologies have been tested, such as XEN, KVM. But the final choice was VMware ESX. The final decision was based on features of administration tools rather than a performance reason [3]. Virtualization is being used in several areas, not only in the testing and development environments but also in production environment, i.e: build farms, non real-time production servers (web, database, applications). Virtualization provides the flexibility and efficiency to deploy servers to accomplish dynamic ALMA operation needs.

After two years of production experience with virtualization, a new step has been taken: consolidate the hardware infrastructure in blade system. Two chassis with Dell M610 blade servers were procured and put into operation.

ALMA Software Build Farm

ALMA software is a four Gb. of source code which takes around ten hours to compile. Currently they are four different active versions being used. In order to support these four branches of software the concept of build farm had been introduced two years ago. Now there are six nodes configured to generate daily builds. Failures found during compilation are notified by email and remedy action is taken immediately.

Hudson has been chosen to be the main continuous integration tool. In addition of it, several custom scripts were created to generate change logs automatically from the code version control repository. These logs are useful for telling the nature of the modifications in the production branches and unsolicited modifications are investigated and reverted if there is no valid justification from the developers. Developers also benefit from the daily build in order to simplify the bug fixing and verification activities. Now having daily builds reduce the time to prepare a working environment to test patches.

Standard Test Environment

ALMA software was designed to run on top of a set of servers which are grouped as the minimum unit under the concept of Standard Test Environment (STE). [3]. In order to fulfil the requirement to commission and verify antennas in parallel, multiple STE were deployed at the OSF. Each STE has the same network subnets and addresses definition in order to guaranty the same environment to run ALMA software. To cope with the repetitive network addresses, virtual routing and forwarding (VRF) technology has been deployed and overlapping IP addresses can be used without conflicting with each other.

STE is also the platform being used to develop and test ALMA software under simulation. There is a total of 13 STEs across ALMA. Some of them are as simple as a set of three servers, and in the other hand, the one dedicated to control the array will eventually have up to 135 servers in it.

To administrate the configuration, installation and maintenance of such an amount of computers, an approach based on Puppet has been tested and validated. It is expected to be the official tool at the OSF soon. This tool will allow for an agile and efficient way to manage dynamic infrastructure.

Monitoring System

To operate the ALMA Observatory, aside from the previously mentioned array elements, it is likely important to make reference of the operation support infrastructure, such as, computer rooms, network devices, operation building facilities, power generators, etc. To guarantee a smooth and successful observation, both areas must work co-ordinately. By having two sites, the ALMA Operation Site (AOS) located at 5000 m., and the OSF located at 3000m over sea level, make the maintenance activities even more complex. In this context, the M&C (Monitor & Control) system plays a fundamental role on defining the live heath status of components of the entire Observatory. The design of the M&C must ensure a proper information distribution to users and specialists for decision making and must also provide alarms to trigger corrective actions, if required.

When ALMA reaches its full capacity with 66 elements in the array, there will be around 1 million monitor variables, which will generate information at a data rate of 40 GB/day. It's really a big challenge to maintain this M&C system, not only in the operational aspects but also solving the storage problem of such amount of data throughout time.

The ALMA Observatory M&C system is a three tiers software: (a) data acquisition layer, (b) data distribution and processing layer and (c) data display layer. Due to historical reason, the array M&C system was designed without considering the support infrastructure's M&C system and vice-versa. The first one is based on ACS BACI properties [1] and the second one is provided by an industrial solution provider. Both systems have components in the aforementioned three layers and works perfectly in the context where they were conceived. But unfortunately neither of both is flexible enough to interoperate with other. Therefore in order to integrate both systems, a third framework is introduced: Zenoss, this framework has several features which make it very suitable for this purpose: template oriented design for

quick devices configuration, vast range of monitoring protocols, powerful events engine, build-in notification services, Zope based core written in python and round robin database as main storage, etc.

By using Zenoss, as the interfaces which access both systems, it is possible to retrieve and consolidate information. A prototype implementation has demonstrated that is feasible to provide consolidated graphical users interfaces or dashboards according to user profiles, such as: array operators, facilities managers, instrumentations and software teams. Additionally, Zenoss provides a very convenient way to storage time series data by using round robin database (RRD). This automatically drops old data and eliminates additional maintenance work. Historical data saved in RRD allows providing trend analysis, characterize overall system behaviour and predict potential failures.

Currently array M&C and the implementation based on Zenoss are still under development. For infrastructure, the off the self system is currently installed and is being fine tuned.

CONCLUSION

ALMA has started to operate in "Early Science" mode since September 30th of 2011. And the preparation achieved in terms of software verification, has proven to be a key factor to achieve this huge milestone for the ALMA communities.

Testing infrastructure and simulation are important resources which are usually relegated to the least priority of astronomical projects. In our experience, these must be considered as part of the original design.

Regression tests, performed systematically, are essential to guaranty stable software and detect bugs introduced involuntarily by developers.

A prototype and semi-production monitoring system has been implemented and deployed. This tool produced relevant information for the maintenance department in order to schedule preventive and corrective activities.

Projects with a big number of elements must validate the scalability feature of its software design and implementation. We suggest that the earlier the better.

ALMA is the first astronomical project constructed and operated under industrial standards, therefore, it is important to include good practices from the industrial area.

ACKNOLEGMENT

The authors wish to acknowledge the contribution and cooperation of ALMA CIPT and especially to ALMA ADC Software Group, which works are reflected in this paper.

REFERENCES

[1] J. Schwarz, A. Farris, and H. Sommer, "The alma software architecture", Proceedings of SPIE, 5496, p. 190 (2004).

[2] V. Gonzalez, M. Mora, Other, "Fist year of ALMA site software deployment: Where everything comes together", Proceedings of the SPIE, 7737, p. 7731Z-77371Z-8 (2010).

[3] Zambrano, M., Arredondo, D., Other "Experience virtualizing the ALMA Common Software", ADASS XIX, 434, 477. (2010)