Re-engineering data processing for resilience: ALMA's new message-passing backbone

Alberto Maurizio Chavan,[1] Rachel Rosen,[2] Tomás Staig[3]

[1]ESO, Garching, Germany [2]NRAO, Charlottesville, VA, USA 2 [3]ALMA ADC, Santiago, Chile

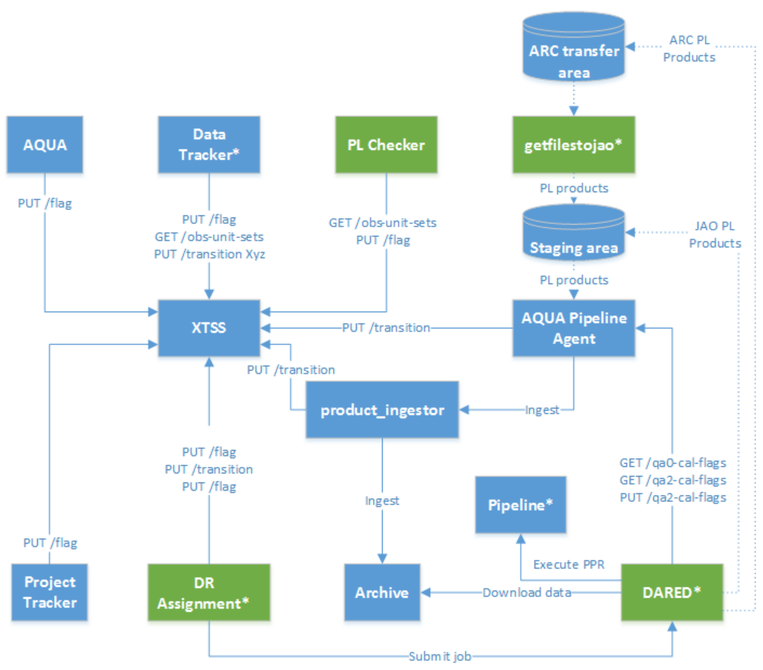
Processing of the data taken by the ALMA Telescope occurs over four continents and fourteen timezones. Our initial system to support integrated data processing was created out of existing and ad-hoc components connected via a variety of protocols. The design was focused on processing as much data as possible without manual intervention. Once all the data and control flows were fully operational, the system was re-designed to be more resilient, where individual component failures will not necessarily affect the overall flow of data. Standardizing on asynchronous message passing allowed the system to gracefully deal with critical components having limited (or no) availability without loss of data.

The resulting system, called *ADAPT* (for ALMA Data Processing Toolchain), will go live for ALMA's Cycle 7 observations.

ALMA Computing Centers

ALMA Quality Assurance program relies on large computational facilities at the Santiago Central Offices (SCO), Chile and ALMA's Regional Centers (ARCs)

Pre-ADAPT Data Processing



In 2017 ALMA changed its centralized data processing to a distributed system, putting together a heterogeneous system including some ad-hoc user scripts.

ADAPT Goals

- Improve reliability, availability
- Fault tolerance and recoverability
- Location transparency (components may be deployed anywhere)
- Visibility (monitoring)
- Testability
- Deployability

High-level Functional Requirements

- Support previous operational model, no restrictions
- Provide dashboard-like view of all system elements, their availability, and interactions
- Allow batch tools to report error conditions and general information

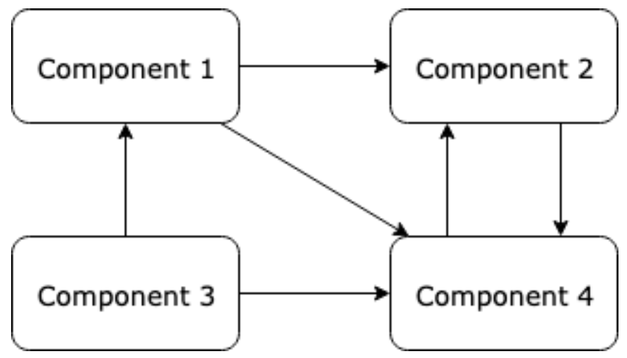
Architectural Principles

Resilient: The system stays responsive in the face of failure and can recover gracefully

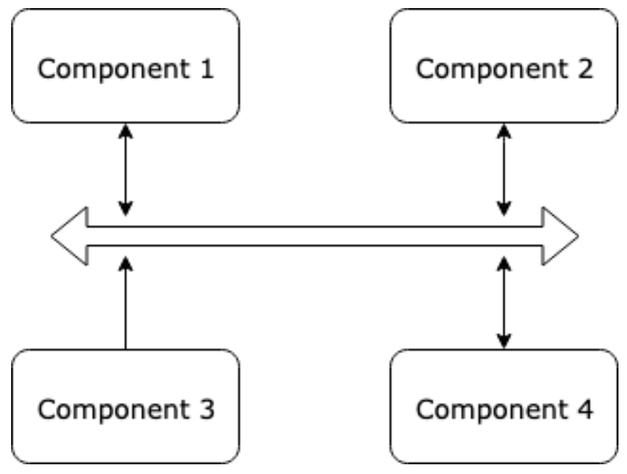
Elastic: The system stays responsive under varying workload and can react to changes in the input rate by increasing or decreasing the resources allocated to service these inputs

Message Driven: The system relies on asynchronous message-passing to ensure loose coupling, isolation and location transparency

Point-to-point vs Message Bus



Point-to-point *synchronous* communication can take place only when both partners are available at the same time: examples are telephone, HTTP, Remote Procedure Calls → *Tight coupling*

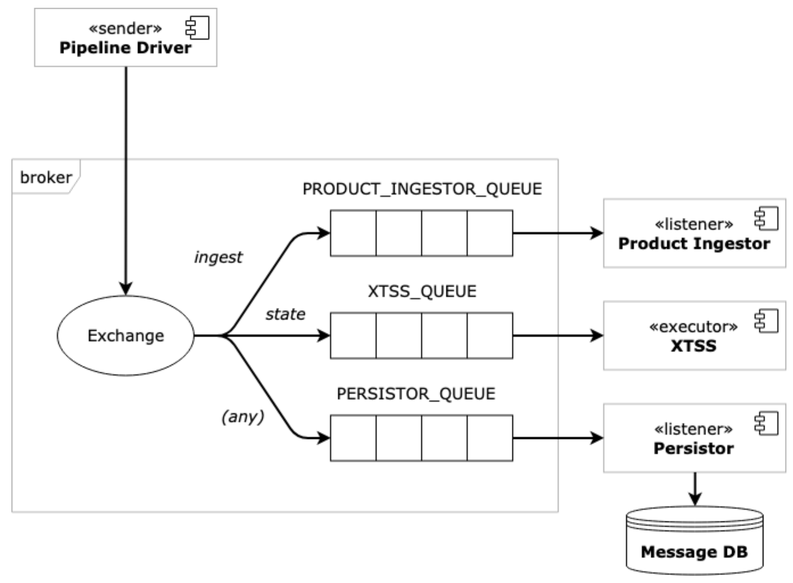


Asynchronous communication allows the receiver to be temporarily unavailable: messages wait safely until the receiver is ready to process them.

Examples are message bus, email, file transfer → *Loose coupling*

Note In a message-passing system the broker becomes a single point of failure. ALMA adopted a clustered deployment, distributed alternatives are also possible.

Message persistence



A persistence subscriber takes care of saving all messages to a database, updating their states as they progress along their life-cycle. While message brokers usually discard any messages after they have been safely delivered, ADAPT messages need to be explicitly persisted to satisfy the *visibility* (monitoring) requirement. A dashboard-type application presents a number of regularly updated system views, including message lists.

Automated integration testing

Implementing ADAPT was generally a relatively straightforward exercise, and testing individual ADAPT components was a matter of exercising the new APIs and ensuring no regression was introduced. Integration testing was a much more complex activity: it had to make sure ADAPT preserves all existing system functionality while adding value in terms of the project's goals and requirements. *Automated testing* provided part of the answer: a set of automated end-to-end tests were created, exercising several meaningful use cases against the production software. Test results were recorded and provided a benchmark to evaluate ADAPT against.

Many other use cases had to be verified manually, which was extremely labor intensive but did detect when internal APIs had not been specified completely or correctly. Deploying a test system across multiple locations requires extensive coordination efforts as well.

ADAPT will be deployed in production in the course of Cycle 7, somewhat later than originally planned.