Astronomical Data Analysis Software and Systems XIII
ASP Conference Series, Vol. 314, 2004
F. Ochsenbein, M. Allen, and D. Egret, eds.

The ALMA Prototype Science Pipeline

Lindsey E. Davis, Brian E. Glendenning, and Doug Tody

National Radio Astronomy Observatory, PO Box O, 1003 Lopezville Road, Socorro, NM 87801

Abstract. In this paper we describe the ALMA Prototype Pipeline Project, a joint ALMA Computing IPT / NRAO Interferometry Software Division initiative to develop a Python based pipeline processing capability for ALMA.

1. Introduction

The primary function of the ALMA science pipeline is to automatically calibrate and image ALMA interferometry data and store the reduced data in the ALMA science archive. The ALMA pipeline system will provide the processing heuristics, in the form of Python scripts, and the software infrastructure required to execute the pipeline in the ALMA computing environment. The data reduction modules will be provided by the AIPS++ group. The ALMA prototype pipeline project is a joint ALMA Computing IPT / NRAO Interferometry Software Division initiative to develop a Python based pipeline processing capability for ALMA using the ALMA Common Software (ACS) framework and AIPS++ data processing modules.

2. The Science Pipeline Requirements

The pipeline software infrastructure must support:

- reuse of legacy science software
- an average data rate of 6 MB / second
- development, maintenance, and testing by scientific staff
- deployment at multiple sites with varied computing resources

To meet these requirements the pipeline architecture must:

- be open, distributed, and component based
- support a powerful scripting language
- provide robust error handling and logging systems
- be portable, scalable, and efficient
- handle resource allocation transparently
- be compatible with the offline reduction system

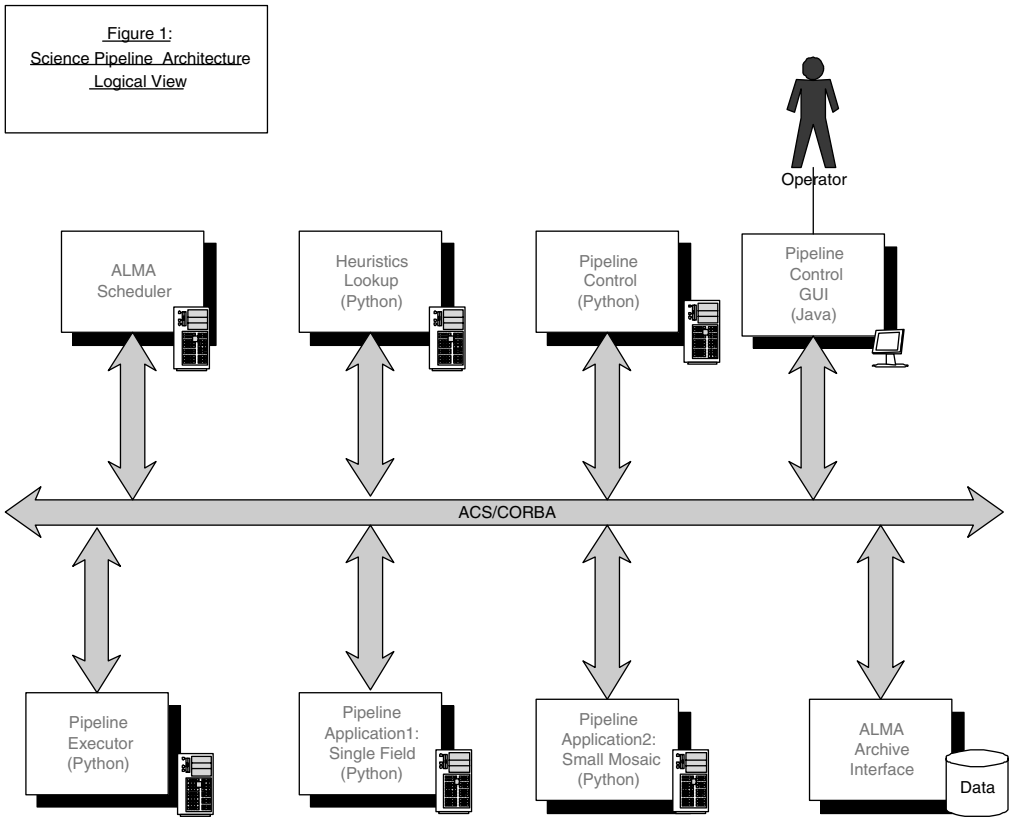

Figure 1. The logical view of the science pipeline architecture.

3. The Science Pipeline Architecture: The Logical View

Figure 1 shows the logical view of the ALMA science pipeline. The ALMA scheduler issues a pipeline processing request. The heuristics lookup module selects the appropriate pipeline application script. The pipeline application script is queued for execution by the pipeline executor. The pipeline application script retrieves the required input data from the ALMA archive, calls the AIPS++ components to do the actual processing, and writes the pipeline results to the ALMA archive. When processing is complete the pipeline executor issues a pipeline processing complete event to the ALMA scheduler. The pipeline control module and GUI provide mechanisms whereby the pipeline operator may intervene in pipeline execution. The CORBA based ACS system manages communications between components of the pipeline, and provides basic lifecycle management, event notification, logging, and error handling services.

4. The Science Pipeline Architecture: The Physical View

Figure 2 shows the physical view of the ALMA science prototype pipeline. The interfaces to the ALMA scheduler and archive are not part of the prototype pipeline project and have been omitted from the figure. The heuristics lookup,

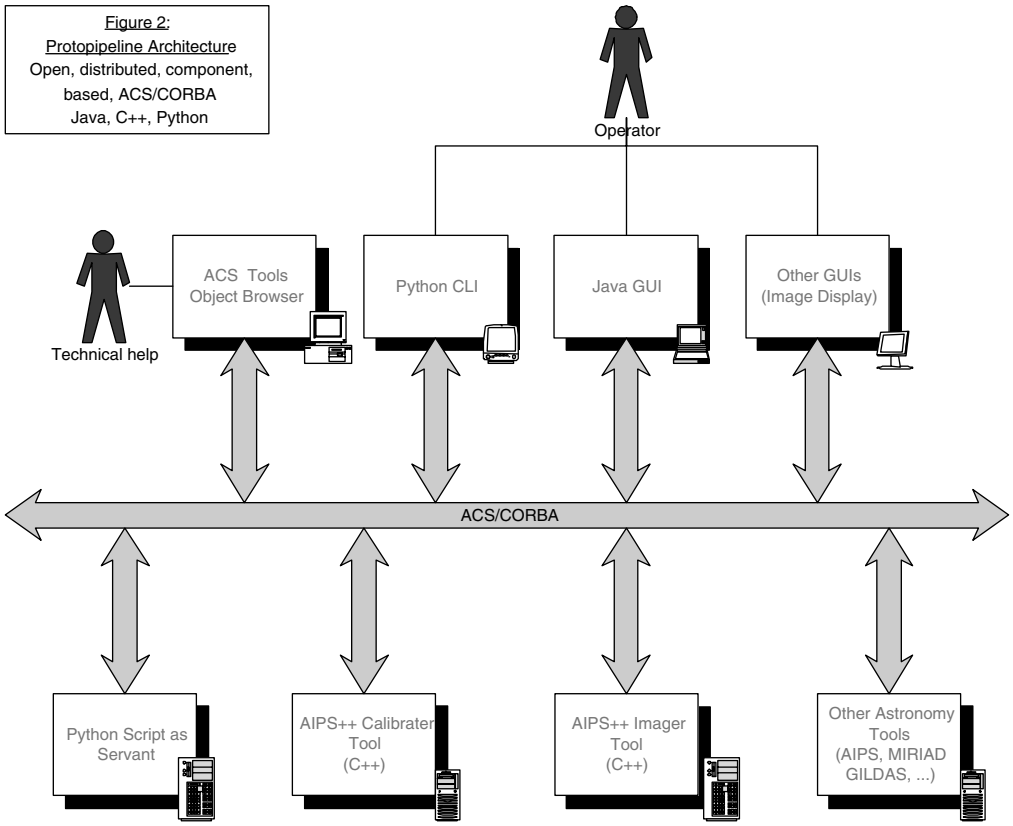

Figure 2. The physical view of the science pipeline architecture.

pipeline executor, pipeline control, and pipeline application modules from Figure 1 will be implemented in Python. The pipeline application modules may be executed as Python commands (Python CLI) or wrapped as ACS components and run inside an ACS Python container (Python script as servant). The two AIPS++ C++ modules may be executed as Python commands (the Python CLI) or called from within Python pipeline application scripts. The pipeline control GUI will be written in Java (Java GUI). ACS will manage communications between the pipeline objects and provide basic services and tools (e.g. Object Explorer). The prototype pipeline architecture is open and will support importing visualization tools and data reduction components from other systems in the future.

5. The Prototype Pipeline Development Plan

The primary goal of the ALMA prototype pipeline project is to develop a Python based pipeline processing capability for ALMA using ACS to execute AIPS++ data processing components. A simple VLA 8 GHz GRB survey data pipeline will be used to drive the prototype pipeline development effort, help focus AIPS++ component development, and demonstrate pipeline processing capa-

bility. The final prototype pipeline will be scripted in Python using AIPS++ C++ components. The project development plan is divided into three phases A, B, C. The goals and current status of each phase are summarized below.

- Phase A Goals
  - Demonstrate basic task execution including messaging.
  - Connect some key AIPS++ components to ACS.
- Phase A Status: Complete
  - AIPS++ vlafiller, constants, quanta, and imager ported to ACS.
- Phase B Goals
  - Add remaining data processing components required for GRB use case.
  - Demonstrate a basic Python scripting capability for the AIPS++ components.
  - Demonstrate a Java client capability for AIPS++ components.
- Phase B Status: In progress
  - Python test scripts written for vlafiller, constants, quanta, and imager.
  - Ports of remaining major AIPS++ components underway.
- Phase C Goals
  - Implement and test prototype VLA GRB pipeline in Python.
  - Run the pipeline as a servant.
- Phase C Status: In progress
  - ACS team have implemented Python as servant capability IN ACS.
  - GRB use case development completed.

6. Current Status and Future Plans

The prototype pipeline project is on track to meet its primary goal of providing a Python based pipeline processing capability for ALMA. The prototype pipeline design already satisfies the first three requirements outlined in section 2. The processing efficiency issue is being successfully addressed by the AIPS++ group. A secondary goal of the prototype pipeline project is to evaluate ACS for use as a future analysis framework. This effort will address longer term issues of portability, scalability, resource allocation, and compatibility with the off-line system.

More information about the ALMA ACS, pipeline, and offline systems (AIPS++) can be found at the ALMA TWiki site at:

http://almasw.hq.eso.org/almasw/bin/view/Main/WebHome

Acknowledgments. We are grateful to T. Cornwell for proposing the migration of AIPS++ to ACS and developing the migration plan. We would also like to acknowledge the hard work of the protopipeline team members Dongshan Guo, Darrell Schiebel, Raymond Rusk, Wes Young. Finally we thank the ALMA ACS team and Dave Fugate in particular for rapid development of the ACS Python capabilities.