NOAO SDM Document PL101=ODI001=CDR-02

ODI Automatic Calibration Pipeline Application Design

F. Valdes[1]

National Optical Astronomy Observatories
Science Data Management

October 13, 2011

[1]NOAO Data Products Program, P.O. Box 26732, Tucson, AZ 85732

Copyright © 2011 by the authors.

Table of Contents

List of Tables

1 Overview

The ODI Automatic Calibration Pipeline (AuCaP) is a workflow that runs under the control of the Open Grid Computing Environment (OGCE) system. As the OGCE name indicates, the computation takes place in a grid computing environment. The computation software orchestrated by OGCE for the automatic calibration of ODI data is what is herein called the ODI AuCaP Application. In this document we describe this application software that is layered under the OGCE system ([6], [1], [3]).

As a grid application interface to OGCE, the AuCaP software appears as simple host commands with command arguments mapping to parameters supplied and consumed by OGCE as parameters. The complexity of the design appears in a) how the overall calibration logic is decomposed into the host commands to achieve efficient, data parallel execution and b) the details of what is inside the host commands. The workflow design is described in other documents such as [7]. The algorithms and how they address the ODI science requirements is given in [2]. The subject of this document is software that makes up the host commands executed by the OGCE workflow.

There are four levels of software structure in the AuCaP application design. At the top level is a wrapper script that creates the host command interface (§2) for OGCE. Within this wrapper there is a compute node orchestration infrastructure that manages the processing across the cores of the node and supports the modularization of the computation into a number of steps (§3). Each of these modular steps is a host command processing some piece of the data and performing some specific operation (§4). These steps are almost entirely scripts in a scripting environment; mostly IRAF command language (CL) scripts with some Python scripts. Finally within the scripting language are calls to the core processing tools (§5). This structure is straightforward but we also illustrate it by the onion layer diagram in figure 1.

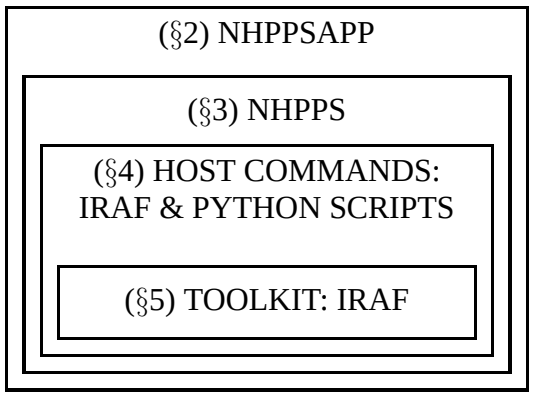

Figure 1: AuCaP Onion Layer Diagram

In the case of any Python scripts the core tools would be the libraries and applications developed by others for astronomical image processing.

Since most of the steps are IRAF CL scripts, the core tools are the large suite of astronomical data processing tools provided by IRAF. Note that a key element of this application design is use of

the decades of software development within IRAF for image processing as opposed to developing new core computational tasks. However, when no appropriate computational tool is available for an ODI-specific calibration step it will be developed by the AuCaP developers. These will be typically be as new IRAF tasks to take advantage of the IRAF core libraries.

2 NHPPSAPP: The Host Wrapper Layer

Rather than write custom wrappers for each OGCE workflow element (called *services* since they are packaged into web services by OGCE) the architecture makes use of a single generic wrapper converter script called NHPPSAPP. As described in §3, the node orchestrator is the NOAO High Performance Pipeline System (NHPPS), hence, the name.

In practice there are simple wrappers around NHPPSAPP to simplify the argument passing and give mnemonic names to the services. For the purposes of this document we consider these wrappers to be part of the OGCE environment.

NHPPSAPP provides all the conversion between OGCE input and output parameters and the standard structure of NHPPS pipelines; the pieces of the end-to-end workflow design. This wrapper sets up and runs the NHPPS orchestration system and triggers the processing with the appropriate input data. When the processing completes the orchestration is shut down and the results formatted and passed on as output OGCE parameters. Note that these results include data products, log information, and exit status information.

While the logic of running the NHPPS system as a self-contained host command is non-trivial the wrapper is still a small piece of software (500 lines of C-shell scripting) plus environment setting source files.

A notable aspect of this wrapper interface architecture is that the simple host commands that run NHPPS pipelines make them suitable for easy reuse. This might be in alternate OGCE workflows or for export (though all the requisite software, e.g. NHPPS, IRAF, and Python, would also need to be exported and installed as well).

3 NHPPS: The Node Orchestration Layer

The Node Orchestration Layer provides workflow management within a single high-performance compute node as opposed to workflow management across grid nodes. Because these nodes generally have multiple CPU cores, efficient use of a node requires more than a simple script of steps. At one extreme, one could make use of complex multi-threaded applications. However, as noted in §5 we will make use of existing, general purpose, image processing toolkits that are not typically single large applications nor, often, multi-threaded. In particular, the IRAF toolkit consists of non-threaded, general purpose tasks.

To make efficient use of multiple cores with a toolkit consisting of general purpose commands requires making use of data parallel threads at the operating system level. This means that the parallel sequencing of steps, both functionally – different independent steps on the same data element – and data – different data elements on the same step – requires an orchestrator. We

have found one of the best orchestrators for this purpose is the NOAO High Performance Pipeline System (NHPPS). This is described in detail in [4]. As noted in the reference, a nice feature of NHPPS is that the steps forming a pipeline are simply described in an XML-based pipeline definition language (PDL).

In a nutshell, NHPPS is an event-driven workflow orchestrator, where steps are executed based on prerequisite conditions such that the occurrence of the conditions for a particular step is what is meant by an ”event”. As long as the prerequisite conditions are met any number of steps and instances of steps on pieces of data may be executed. The running of multiple executions is handled by the operating system that can make appropriate use of cores, memory caches, and swapping strategies. With enough parallel steps and parallel data elements this orchestrator, which also provides throttle control to avoid swap bottlenecks when there are too many processes, has been found to be extremely efficient at utilizing multiple cores with independent granular steps typical of the logic required by AuCaP.

Let us consider an example of how the orchestrator supports the processing. ODI exposures are distributed across a set of grid nodes with one exposure per node. The host command is to run the OTA calibration steps of bias subtraction and dome flat fielding (actually other things would be done but this is for illustration). The pipeline workflow divides up the exposure by OTA. Each OTA is handled as a separate sub-workflow. NHPPS sequences the steps of bias subtraction and dome flat fielding for each OTA in parallel. The 64 ”threads” are then being executed by the OS across the cores, such as a 12 core node. The NHPPS throttling would keep the number of OTAs being handled at one time to something like 12 instances. See [5] for an actual OGCE workflow demonstration similar to this example and based on the AuCaP layers presented here.

There are two key software components forming this layer. These are a node manager that is the key element and one or more pipeline managers. These work together to interpret the PDL, handle all the conditions and actions, and execute the host command modules.

As an additional aspect of this architecture, though not the driver, is that by developing the various AuCaP services under NHPPS and wrapped by the NHPPSAPP wrapper, it is simple to run the entire AuCaP workflow as a purely NHPPS application on a local cluster (NHPPS also provides cluster orchestration through the node managers working together). This makes development and testing easier and allows for alternate processing then just in a grid environment.

4 The Computation Scripting Module Layer

An NHPPS pipeline consists of a set of steps, also call modules or stages, that perform some macro piece of the workflow on a particular type of dataset. These steps are orchestrated by NHPPS, as describe earlier, based on events. NHPPS executes a command when the prerequisite conditions are met. This command can either be a host command or a plug-in of the NHPPS system. In most cases the command is an host command.

The host commands in AuCaP are predominantly composed of IRAF executable tasks (see §5). IRAF provides not only this toolkit of image processing executables but a framework and scripting environment to easily tie them together through one of the IRAF command languages into

a logical pipeline module. For AuCaP we use the classic IRAF CL for most of the scripting. Unlike interactive astronomical use, where the CL is both a session command environment and interpreter of scripts within the session, AuCaP modules use the CL as a host command interpreter shell; a Unix execution feature often referred to a *a hash-bang script*. The modules begin with a hash-bang directive to run the IRAF CL interpreter followed by the IRAF script. The CL provides a command line argument functionality to deliver host command line arguments to the script environment. These IRAF scripts appear just like any Unix host command found in the execution path.

One thing to understand about the the AuCaP application design is that it is also modular in the sense that the connection between steps is through files and independent executions. In other words, there is no use of shared memory. As noted earlier, the use of an orchestrator that can manage many processes in parallel makes the I/O penalty less severe and the processing is still as efficient as one can have without rewriting the steps and underlying software to use shared memory.

5 The Core Toolkit Layer

The ODI detectors and mosaic focal plane are not fundamentally different from other astronomical CCD cameras. Therefore, there is no need to redevelop tools already widely used for calibrating data from such instruments. There are a number of toolkits one can chose from. Because NOAO will develop the AuCaP, the logical choice is the NOAO developed IRAF toolkit. Not only is it the logical choice but it is one of the most widely used and well-tested toolkits in astronomy.

IRAF provides a toolkit of image processing executables consisting of millions of lines of compiled and script code and people-decades of developer time. There is not much more to say about the toolkit. Though most of the pipeline makes use of existing IRAF tasks, some written as general image processing or standard CCD reduction tools and some written for other NOAO pipeline applications such as for the NOAO Mosaic Imagers, there will be a few tasks that will need to be written expressly for ODI. These will also be developed as IRAF executables in order to fit in with the other tools, to make use of the many core libraries such as for masks, and to be exposed for standard interactive IRAF use. The only significant tool that can be forecast as needing to be developed is to convolve static calibration exposures by the unique shift history of OTA-guided exposures.

References

[1] R. Singh. ICD: Workflow and Pipeline Interfaces for Tier 1 Processing. ODI CDR Document CDR-09, ODI-PPA, Oct 2011.

[2] R. Swaters and F. Valdes. ODI Automatic Calibration Pipeline Design Elements. ODI CDR Document CDR-18, ODI-PPA, Oct 2011.

[3] F. Valdes. Transforming NHPPS Pipeline Applications into Grid Applications. Draft SDM Pipeline Document PL019, NOAO/SDM, Jun 2010. http://chive.tuc.noao.edu/noaodpp/Pipeline/PL019.pdf.

[4] F. Valdes, T. Cline, F. Pierfederici, B. Thomas, M. Miller, and R. Swaters. The NOAO High-Performance Pipeline System. SDM Pipeline Document PL001, NOAO/SDM, Oct 2006. http://chive.tuc.noao.edu/noaodpp/Pipeline/PL001.pdf.

[5] F. Valdes and S. Marru. The ODI Demonstration Tier 1 Pipeline. Draft SDM Pipeline Document PL021, NOAO/SDM, Sep 2010. http://chive.tuc.noao.edu/noaodpp/Pipeline/PL021.pdf.

[6] F. Valdes and S. Marru. The Marriage of Mario (NHPPS) and Luigi (OGCE). In I. N. Evans, A. Accomazzi, D. J. Mink, & A. H. Rots, editor, *Astronomical Data Analysis Software and Systems XX*, volume 442 of *Astronomical Society of the Pacific Conference Series*, pages 211–+, July 2011. Also http://chive.tuc.noao.edu/noaodpp/Pipeline/PL023.pdf.

[7] F. Valdes and R. Swaters. ODI Pipeline Data Flow Design. SDM Pipeline Document PL013, NOAO/SDM, Oct 2009. http://chive.tuc.noao.edu/noaodpp/Pipeline/PL013.pdf.