NOAO DPP Document PL008

The NOAO High-Performance Pipeline System: Architecture Overview

[1]T. Cline, [2]F. Pierfederici, [3]R. Swaters, [3]B. Thomas, [1]F. Valdes

National Optical Astronomy Observatory
Data Products Program

November 16, 2006
To appear in ADASS XVI [P4.19]

[1]NOAO Data Products Program, P.O. Box 26732, Tucson, AZ 85732
[2]NOAO LSST Program, P.O. Box 26732, Tucson, AZ 85732
[3]Department of Astronomy, University of Maryland, College Park, MD 20742

Table of Contents

1 Introduction **2**

2 Pipeline Management System **2**
2.1 Pipeline Description Language 2
2.1.1 Actions 2
2.1.2 Events 3
2.2 Pipeline Manager 3
2.2.1 Module Manager 3
2.2.2 Blackboards 4

3 Services **4**
3.1 Node Manager 4
3.2 Directory Server 4
3.3 Pipeline Selection Utility 5

Abstract

The NOAO High-Performance Pipeline System is an infrastructure which provides event driven execution of scientific data processing pipelines within a distributed parallel system. The architecture includes the Flipper Pipeline Management System (Pierfederici 2005) and an assortment of services which manage hardware resources, calibration libraries, and metadata databases. The pipeline system infrastructure is separate from the pipeline applications which are built from host-callable programs and data processing systems. In this paper we describe the components of the pipeline system. The NOAO Mosaic Pipeline utilizes this system and is described in companion papers (Swaters & Valdes 2006, Valdes & Swaters 2006).

Keywords: techniques: image processing

1 Introduction

Automated data processing pipelines are needed to handle the significant data volume produced by the astronomical instruments of today and those proposed for the future. NOAO has developed a pipeline infrastructure designed to meet the needs of the astronomical community. We call this infrastructure the NOAO High-Performance Pipeline System (NHPPS).

The NHPPS is composed of a Pipeline Management System (PMS) and a collection of services which provide communication and management functionality across a distributed multi-node environment. In section 2 we discuss the PMS and its components. Section 3 provides an overview of the various services used by our system.

2 Pipeline Management System

The core of the NHPPS is the PMS which is responsible for executing the actions of a pipeline when the proper conditions are meet. The NHPPS PMS was developed under the name FLIPPER. It was originally based on the OPUS system (Rose et al. 1995) in use at the STScI; however, it has been updated to take advantage the XML language and to reduce the overall complexity of system configuration. The PMS manages the individual pipelines which together create a Pipeline Application.

The components of the PMS include an XML grammar (the Pipeline Description Language or PDL) which describes pipelines to the PMS, a Pipeline Manager which is the process representing an instance of a pipeline, Module Managers which perform the actions of the pipeline, and Blackboards which are used to post messages about the status of dataset processing.

2.1 Pipeline Description Language

We have developed a PDL using the XML language to describe pipelines and their actions to the Pipeline Management System. The PDL provides easy configuration of several features of the NHPPS such as process resource constraints (minimum required available disk space and maximum execution time), three different classes of pipeline actions (Setup, Primary, and Cleanup) which are executed when certain events have occurred (Time, File, Status, and the special ExitCode).

A pipeline is a collection of modules that define *settings* and *actions* which are executed when certain *events* occur. The modules typically reference each other within a pipeline; for instance, to specify that a previous module must be completed before another module may start. Each pipeline within the Pipeline Application is defined within its own XML file.

2.1.1 Actions

Actions perform the work of the pipeline. The NHPPS is designed to perform up to 3 different sets of actions within a module: Setup, Primary, and Cleanup.

Setup actions are performed before the Primary action and serve to setup the execution environment for the Primary action, perform bookkeeping actions on the blackboard, etc.

The primary action is the action the module is written to execute. There is only one Primary action in each module and it is typically a *Foreign* action which executes a program or script via a command line.

The cleanup actions are sets of actions, one of which may execute based upon the Exit Code event after the Primary action is completed.

2.1.2 Events

The events which trigger the start of execution for a module are Time, File, and Status events. There is a special type of event, the Exit Code event, which may be used to trigger optional Cleanup actions in the module.

Time events repeat at fixed intervals between a start and end time, or occur only once at a specified time.

File events occur when a file matching a search pattern is found within a given directory. These files are called 'trigger files' within the NHPPS and are a very useful method for starting the flow of execution within a pipeline due to their ease of creation.

Status events occur when the blackboard status for a particular module matches a certain value.

Exit Code events are generated when a modules primary action is completed. This value is typically the value returned from the modules primary action. It may be used to instruct the module to perform a set of cleanup actions, based on the value of the Exit Code. This functionality allows a module to execute one of a number of different sets of cleanup actions based upon the Exit Code. By specifying different sets of cleanup actions, the module is able to handle cases where the primary action ended successfully, with a partial result, or failed.

2.2 Pipeline Manager

Each instance of a pipeline in a Pipeline Application has a corresponding Pipeline Manager (PM). The PM is primarily responsible for parsing its pipelines XML file, setting up Module Managers for each module in the pipeline, providing scheduling guidance to the Module Managers, and creating shared Blackboards.

2.2.1 Module Manager

Each instance of a module within a pipeline has an associated Module Manager which performs two key tasks 1) checking that the events necessary for the module to execute have occurred and 2) setting the environment and executing the modules actions when the required events have occurred.

For the pipeline architect the main function of the MM is to execute a desired action which is typically a program. The program may be specific to a single module but often is more generic. In either case the program needs context information when it is run. The module manager provides this in two ways. Programs may be called with arguments as specified in the pipeline description language (PDL). The MM also sets a number of standard environment variables. These include the dataset name or identifier, the pipeline name, the module name, the type of event, the module's

blackboard flag when it is triggered, the start time, the process identification, and logical directories associated with the pipeline.

2.2.2 Blackboards

Blackboards store messages posted by the MMs. These messages may be requests for other MMs to begin processing data, status of current data-set processing, or even the status of the MM itself. Blackboards are visible to *all* of the MMs in the pipeline, and therefore MMs are able to, and do, 'communicate' with each other by posting and reading messages on the blackboards.

The PM creates two distinct blackboards in memory, which may optionally be mirrored on disk. These two blackboards track 1) the progress of individual data-sets through the pipeline and 2) the status of pipeline modules to include, among other things, whether they are active or inactive, which data-set they are processing, and when they began processing.

As we have mentioned, these blackboards are shared among all of the MMs in the PM. This allows each module to check the status of the other modules in the pipeline.

3 Services

The targeted execution environment for the NHPPS is a multi-node processing cluster. The services described in this section enable the NHPPS to take advantage of a distributed processing environment.

3.1 Node Manager

The NM is primarily responsible for starting, pausing, resuming, and stopping pipelines. Additional tasks include tracking available resources on a node, and communicating with the Directory Server (DS) to locate other available NMs within the pipeline processing cluster.

A Node Manager (NM) is started on each node which is a member of the pipeline processing cluster on which the Pipeline Application is run. At startup, the NM contacts the DS to register itself as available for processing datasets. Note, this requires the DS to be running first. If no DS is running then the node will be unaware of other processing nodes, however it may still run a pipeline application although processing will be restricted to a single machine.

3.2 Directory Server

The Directory Server (DS) is responsible for maintaining a list of the nodes within the processing cluster on which the NM has been started. The NM's occasionally will need to determine which nodes are in the processing cluster and this list is sent to the NM when requested.

3.3 Pipeline Selection Utility

The pipeline selection utility, `pipeselect`, is a key tool in building a distributed pipeline. This tool communicates with the DS and NM servers and is most frequently called from within pipeline modules to discover other available pipelines, rank their resources, and provide trigger information. In application a module in a pipeline wishing to trigger another pipeline must first discover the instances of the pipeline. It specifies the name of the desired pipeline, whether a list of the available pipelines is to be returned or a ranked list for a desired number of instances, and the required disk space available to the pipeline. In addition to the arguments the utility uses a configuration file that currently defines whether a desired pipeline may be anywhere in the processing cluster, must be on the same node as the calling pipeline, or must be on any node other than the local node.

References

Pierfederici, F. 2005, ASP Conf. Series, 347,614

Rose, J., et al. 1995, ASP Conf. Series, 77, 429

Swaters, R. and Valdes, F. 2006, to appear in ASP Conf. Ser. XVI, [P4.20]

Valdes, F. and Swaters, R. 2006, to appear in ASP Conf. Ser. XVI, [P4.21]