Astronomical Data Analysis Software and Systems XVI
ASP Conference Series, Vol. 376, 2007
R. A. Shaw, F. Hill and D. J. Bell, eds.

Virtual Astronomical Pipelines

Rahul Dave[1], Pavlos Protopapas, Matthew Lehner

Harvard-Smithsonian Center for Astrophysics, Cambridge, MA, USA

Abstract. The sheer magnitude of databases and data rates in new surveys makes it hard to develop pipelines to enable both the analysis of data and the federation of these databases for correlation and followup. There is thus a compelling need to facilitate the creation and management of dynamic workflow pipelines that enable correlating data between separate, parallel streams; changing the workflow in response to an event; using the NVO to obtain additional needed information from databases; and modifying the observing program of a primary survey to follow-up a transient or moving object. This paper describes such a Virtual Astronomical Pipeline (VAP) system which is running in the TAOS project. The software enables components in the pipeline to react to events encapsulated in XML messages, modifying and subsequently routing these messages to multiple other components. This architecture allows for the bootstrapping of components individually in the development process and for dynamic reconfiguration of the pipeline as a response to external and internal events. The software will be extended for future work in combining the results of surveys and followups into a global virtual pipeline.

1. Motivation

A pipeline is a structured sequence of collections of processes, or stages. Control, notification and error messages, and science data flow through the pipeline from one stage to another. Currently, most pipelines are static with flows described ahead of time and hidden in logic embedded in code; this is especially true of analysis pipelines. However, stages in many systems, such as those used for multiple observational purposes, are long-running and dynamic. These stages undergo change in the state of processes through data or message flow. An example is a change in observational program of a survey due to a gamma-ray burst (GRB) notification on the GCN network.

Such dynamic pipelines are crucial in the new astronomy. By providing a descriptively defined workflow that is quickly changeable, event processing can be facilitated at high rates and data volumes. On-line analysis for follow-ups and comparison wth other surveys is likely to be at a remote sites due to the large size of datasets, and a changeable description enables reuse of code at these sites through web services. Additional machines and processes can be easily added and removed, facilitating redundancy. Furthermore, an incremental mode of software development that takes one from prototype pipelines on laptops to production pipelines on mountain clusters can be supported.

[1]Most of this work was done while at the University of Pennsylvania, Philadelphia, PA, USA

2. TAOS and the VAP Prototype

The Taiwan-America Occultation Survey (TAOS) project uses four 0.5-m telescopes to scan large (3 $\deg^2$) fields on the sky looking for Kuiper-belt objects (KBOs), but is also set up to pre-empt its scheduled search to respond to GCN GRB alerts. TAOS runs a fully automated, high cadence (5 Hz) observation pipeline. The pipeline processes about 30 GB of data per night through observation, photometry, statistics and storage stages. The data is reprocessed in the day with fake occultation events added in for calculating efficiency. The light curves produced are also useful for transient and transit searches.

While our immediate concern was to have a flexible and incrementally developed high performance pipeline for TAOS, we defined an Open Pipeline Architecture so that our software and specifications will be usable by other projects, especially our own future ones. Processes in the pipeline communicate through events, or messages, which determine the next step in the pipeline execution. These messages are routed by intermediate routers to other processes. Data flow between processes is peer-to-peer for performance reasons, and uses a standard IO interface regardless of transport implementation. A simple trust model implements security for the messaging, especially for the subscription or rule messages used to dynamically reconfigure the workflow in the system. The architecture consists of:

Nodes and Stages Pipeline nodes are the processes that make up the pipeline, with nodegroups that are collections of nodes (such as telescope mount control daemons with different parameters). A pipeline stage may consist of multiple nodes and is conceptually a collection of processes that happen at the same time or place (e.g. the daemons that control the workings of an individual enclosure such as the camera and mount daemons). A stagegroup is a collection of such stages, such as the collection of enclosure processes for all of our four telescopes.

Virtual Pipeline Message Format (VPMF): We have defined a simple message format for the pipelines with the aim of being able to route messages based on the value of any part of their payload. A message consists of a header section, a key-value payload dictionary, and a data section which can carry a binary, such as a C struct. The header section consists of `to` and `from` pipeline components, an `opcode` to describe the purpose of the message, an `msgmode` (one of request, notification, or subscription), and a `domain` (either workflow, system, or application). A `priority` can be used to preempt other messages in situations requiring urgency (see Figure 1).

The XML-RPC data format is used for messages; this makes it simple to write pipeline components in almost any programming language. We have written a C library `libpmsg` to integrate messaging into the TAOS daemons, providing both synchronous and asynchronous interfaces. We have also written Ruby and Python libraries for other components of our system.

Virtual Pipeline Routers (VPR): The VPR is the the main software product of the VAP system, and handles routing to nodes, nodegroups, stages, and stagegroups. It receives messages from nodes (there is one VPR per node and stage) and forwards them using the subscriptions. It is implemented in the TAOS system by a daemon called `PlatformD`, developed in Python using the Twisted library (`http://twistedmatrix.com`). The daemon implements message persistence, process handling, and multi-protocol communications with other routers

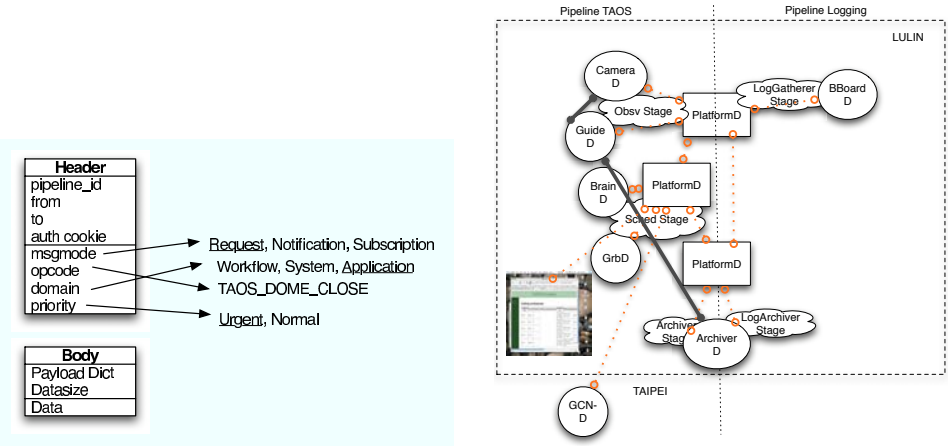

Figure 1. The left image in the panel shows the structure of a VPMF message. The right image shows two pipelines in the TAOS system which work together. The stages in the system, the nodes belonging to the stages, and the PlatformD daemons which are the VPRs are illustrated. Thick black lines represent peer-peer IO, while thin dashed orange lines illustrate message flow from nodes to their routers and then between routers. The Sched stage PlatformD acts as the VPP.

and nodes. The routing engine provides simple value matching and template substitution for the message header and payload dictionary based on the rules, but can delegate to a custom Python script if desired, the idea being to make simple pipelines simple to code and complex pipelines possible.

Virtual Pipeline Portal (VPP): There is one portal per pipeline, for authentication, authorization, control and status. The portal is implemented in a VPR; with custom routing rules enabling this behavior. It is possible to route messages between multiple pipelines through this special VPR. We use this facility to compose the larger TAOS pipeline from smaller, individually developed control, logging, and analysis pipelines. A Python based user shell, `bush`, has been implemented that speaks http to the portal and provides a simple shell language to control the pipeline.

VPIOI: Virtual Pipeline IO Bus: The IO bus is implemented in a C library, `libpio`, currently providing file, mmap and socket IO implementations for data transport, with reference counted data buffers, and the concept of sink and source pads which may be connected to multiple other pads to move data. This infrastructure creates a set of pipeline components connected directly to each other for high performance IO; with IO scheduling optionally controllable with VPMF messages.

3. Combining Pipelines: Constructing a Global Virtual Pipeline

The Virtual in VAP refers to our desire to compose pipelines at surveys together to form larger, virtual pipelines, or super surveys that take observations, compare them against existing databases, and schedule followup observations. We currently have support in the VPR code to enable message communication be-

tween pipelines, but no mechanisms to enable the advertisement of capabilities of pipelines, to move data between pipelines over VPIOI, or provide a security model for these communications.

As an example of the kind of application that will be enabled by a global, virtual pipeline, consider a new generation solar system survey project that seeks to replicate the Quaoar discovery model. This new survey (possibly Pan-STARRS) will detect moving objects at least 100 times more frequently than present surveys. Once a candidate is identified, it would be useful to constrain its orbit by going back to surveys taken at different times in the past. On projecting the orbit error circle back to those past times, we would see if we can find the object in the corresponding RA and Dec range. This would be done by comparing the past image against a known template for that part of the sky in which the orbit error circle lies. If one does find the object, a stronger constraint on the orbit is achieved and the process can be repeated against older surveys with a far more tightly constrained error circle. Follow-ups to nail down the orbit even further may now be scheduled at another telescope.

In this example complex, interacting steps must be performed by three interacting pipelines. These are the pipelines at (a) the detecting survey, (b) the data center or VO portal with archival data, and (c) the followup survey. An iterative virtual pipeline may be composed which permits ever tighter orbit constraints.

4. Conclusion and Future Work

Virtual pipelines provide a model for interaction between surveys, data centers, and the VO; and a way for long running workflows to be saved and repeatedly executed. We expect to release our implementation of VAP as open source software in 2007.

We will add functionality providing message routing and data transfer between pipelines, a kerberos like interpipeline security model, and a web server based pipeline portal. We will be supporting VO events and VO sinks and sources on the IO bus. Finally, we will provide the ability to add policies for handling errors descriptively. We hope that once the software is released, developers will be able to add functionality to make an already flexible system even better. Please email `rahuldave@gmail.com` with opinions and suggestions.

Acknowledgments. We are grateful to the Keck Foundation for their support under the aegis of the CUSP project.