A Reusable Process Control System Framework for the Orbiting Carbon Observatory and NPP Sounder PEATE missions

Chris A. Mattmann, Dana Freeborn, Dan Crichton, Brian Foster,
Andrew Hart, David Woollard, Sean Hardman, Paul Ramirez,
Sean Kelly, Albert Y. Chang, Charles E. Miller
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109, USA
mattmann@jpl.nasa.gov

Abstract

We describe a reusable architecture and implementation framework for managing science processing pipelines for mission ground data systems. Our system, dubbed "PCS", for Process Control System, improves upon an existing software component, the OODT Catalog and Archive (CAS), which has already supported the QuikSCAT, SeaWinds and AMT earth science missions. This paper focuses on PCS within the context of two current earth science missions: the Orbiting Carbon Observatory (OCO), and NPP Sounder PEATE projects.

1 Introduction

Data volume and computational needs for Earth science missions at NASA are growing by orders of magnitude. The low cost of disk storage space and the increasing power and pervasiveness of high performance computing have engendered an era in which previously unimaginable science questions can be answered in years rather than decades. These science questions range from the study of sea surface temperatures to observe maritime pollution, to measuring atmospheric chemical composition for weather forecasting, to obtaining a better understanding of the Earth's global carbon cycle and climate change as a whole.

A significant portion of any space-based NASA earth science mission is a Ground Data System (GDS). The GDS is responsible for receiving raw spacecraft data as delivered from a ground station[1], and processing the information through several focused series of steps with the goal of delivering the scientific value encoded in the data to interested scientists, both locally at an instrument team center, and then to universities, decision makers, and the broader science community. The processing that a GDS must perform ranges from mundane activities including data (un-)marshalling (removal of special space "header" information), and subsetting, to more involved processes including temporal and spatial positioning, calibration, and statistical analysis, to complex scientific assimilation including prospective and retrospective physical modeling of a scene.

Beginning with Automated Multi-Mission Operations System (AMMOS) Multi-mission Ground Data System (MGDS) in the early 1990s, our work has focused on building reusable software components for GDS systems. As an example, the Central Data Base (CDB) Subsystem of the MGDS included data base management software comprised of metadata and file management, file transfer capabilities, user interfaces and data storage facilities to support multi-mission telemetry data streams for current and future planetary missions. This demanded that the CDB architecture adhere to the architectural principles of extensibility, scalability, and reusability. Because the CDB was and is part of a larger system that included controlled, centralized hardware, these architectural principles of CDB were satisfied for AMMOS by simply ensuring that the CDB was data and policy driven.

Our ensuing work on the Alaska SAR Facility (ASF) and NASA Scatterometer (NSCAT) projects, made clear two significant trends: 1) neither of these missions were part of the controlled, centralized system for which the CDB was developed and 2) the data management requirements for these two missions were different from each other and AMMOS. This meant that 1) hardware and platform choices could not be assumed and 2) additional capabilities not originally required for AMMOS had to be developed. In or-

[1] A strategically placed data center on Earth with ample ground-to-space bandwidth and connectivity for receiving satellite data.

der to meet mission schedule and cost constraints, developers for each project independently employed a method we coined "rapid adaptation" of the original CDB software that resulted in two very successful mission data systems with ultimately very few similarities or shared code.

At the time the NSCAT follow-on mission (SeaWinds on ADEOS II) was ramping up, a technology task originally funded by the NASA Office of Space Science was focused on architecting and developing a common, standards-based software framework dubbed Object Oriented Data Technology (OODT) [12]. OODT provided "out of the box" core data management software services while remaining adaptable to address the (potentially evolving) requirements that are unique from mission to mission.

Several authors of this paper supporting SeaWinds and the OODT technology task decided to collaborate to create a platform- and database-independent service for managing files and tasks. The result of this collaboration was the OODT Catalog and Archive Service component that was architected to be reusable, reliable and scalable. The SeaWinds (on QuikSCAT and ADEOS II) and Advanced Communications Technology Satellite (ACTS) Mobile Terminal (AMT) projects benefited greatly from employing the CAS component to support their science data systems. QuikSCAT is in its 10th year of a planned 2-year mission and is continuing to function in a nearly lights out mode. Hardware has been added to the system to support the unplanned data and processing volumes (reprocessing of 7 years of data completed in 6 months, simultaneous with daily operations) by simply changing the software configuration. No software engineers were required to extend the system.

While the CAS component successfully supported SeaWinds and AMT, the following JPL earth missions, Orbiting Carbon Observatory (OCO) and NPP Sounder PEATE, needed to support far more complex processing (greatly increased data volumes and processing throughput) and various hardware and platform configurations. This forced us to rethink the CAS component implementation which resulted in 1) the refactoring of the CAS component into two distinct components, the File Manager and the Workflow Manager and 2) the development of a third component to provide a standard interface to various hardware and platform configurations, the Resource Manager.

The refactoring of the CAS into the File Manager and the Workflow Manager components solved several issues. First, it decoupled the initiation of a workflow from the ingestion of a file. Therefore, while workflows can be initiated based on the ingestion of a particular file or file type, they can also be initiated based on other events such as a specific time of day, an operator request or a software request. Second, the refactoring provides developers and system designers the ability to utilize only the components they need. And third, the refactoring supports independent evolution of the components, and thus capabilities. The combination of these three refactored CAS components have come to be known as the Process Control System, or PCS.

In addition to the File Manager, Workflow Manager and Resource Manager components that provide common reusable capabilities for file and metadata management, pipeline processing and job submission, we have also developed reusable interfaces to these components to provide additional commonly required capabilities for science data management systems. To support the automation of file ingestion, we have developed a configurable push-pull framework and crawler framework. To provide easy integration of science code in order to support all phases of algorithm development (testbed, operations and science computing facility), the PCS Task Wrapper has been developed.

In this paper we will describe our core PCS components, their architecture, how they helped us solve problems on OCO and NPP Sounder PEATE, and how they are positioning us for the future of Earth science mission work. We believe such work will necessitate the same spirt of architectural reuse, understanding and mission specific adaptation that led to the genesis of the modern PCS and that will ultimately lead to its future evolution. We will argue in this paper that our PCS uniquely positions us in the state of the art in constructing large-scale, distributed, data-intensive GDS software for NASA Earth science missions.

The rest of this paper is organized as follows. Section 2 provides further background and related efforts in the areas of grid computing, workflow systems and science data systems. Section 3 describes the core PCS architectural components in greater detail. Section 4 presents our experience leveraging the PCS on OCO and NPP Sounder PEATE. Section 5 rounds out the paper with conclusions and highlights our planned future work.

2 Background and Related Work

Since the development of the computational grid [8] as a means for the virtualization and sharing of processing and storage resources across organizational and geographic boundaries, many groups and organizations have recognized the power of the grid as an enabler of large-scale scientific research. In this paper, we discuss ongoing software projects and research initiatives relevant to the PCS.

2.1 Grid Systems

The Globus toolkit [9], developed by The Globus Alliance, is a collection of open-source software tools for developing distributed computing systems and applications. The toolkit provides users with a suite of software components and libraries that can either be used individually or

packaged together to implement the many aspects of a distributed, service-oriented infrastructure including security, resource and data discovery, access, and management, and communication modules customized for a particular grid-based effort.

2.2 Workflow Systems

The past ten years have witnessed an explosion in the number of workflow languages and software systems developed to support scientific workflows. Yu and Buyya [15] attempted to taxonomize these scientific workflow systems, largely according the underlying technologies with which they were built. In addition to this taxonomy, Woollard, et. al., presented a characterization of workflow systems based the intended scientific use [14]. Specifically, the authors classified certain workflow systems as Production Systems, of which both the OCO and NPP Sounder PEATE ground data systems are examples.

2.2.1 Condor

Condor [11] is a grid-based job scheduling system developed at the University of Wisconsin Madison which aims, among other things, to improve the effective usage of available distributed computing and storage resources by detecting and exploiting machine idle cycles. Condor provides mechanisms for job queuing, setting scheduling policies, and general resource management and monitoring. Condor insulates users from the particulars of the details of the underlying infrastructure by transparently handling decisions about when and where jobs will be scheduled, monitoring their execution, and producing notifications of completion. While originally designed to operate in a workstation environment, a variant of Condor, Condor-G [10], leverages the Globus toolkit to provide a Condor implementation that is interoperable with Globus-based grids.

2.2.2 Pegasus

Pegasus [7] is similar to Condor in that it provides a layer of abstraction between the jobs to be processed and the hardware that they will eventually be processed on. Developed at the USC Information Science Pegasus is capable of dynamically assigning computational workflows with multiple processing steps to a large number of grid-based compute nodes based on resource availability. In addition to generating an initial workflow mapping, Pegasus offers the ability to transparently remap a workflow, increasing the reliability of the system in the event of failure in a small number of compute nodes.

2.3 Science Data Processing Systems

Science Data Processing Systems provide the base level of service needed to effectively manage the vast quantities of intermediate and final data products generated by large-scale, computationally intensive research tasks. While there are a large number of systems in operation, we focus our discussion on those which provide services distinctly similar to the PCS.

2.3.1 S4PA

The Simple, Scalable, Script-based Science Product Archive (S4PA) [3], is a storage architecture developed and deployed at NASAs Goddard Space Flight Center in support of the operation of the Goddard Earth Science Data and Information Services Center (GES DISC). As cost was a primary factor in the development of S4PA, the developers have taken pains to streamline the system. Hosting the primary copy of all data online reduced the need for costly physical media distribution, and utilizing the UNIX directory structure, in combination with metadata-encoded filenames, provides a simplified mechanism for archive and retrieval.

As its name implies, the S4PA is primarily a data archive service. The PCS, as described in this paper, addresses data archiving, but takes a more architecturally grounded approach, eschewing scripts in favor of first-class architectural components and connectors to implement complete, end-to-end data processing pipelines. Furthermore, as complete science data processing pipelines are composed of a large number of complimentary, interconnected services, a formal architectural underpinning helps to provide unity and cohesion among the constituent components.

2.4 Standards

Grid-based science data processing systems have matured sufficiently for common themes, lessons, and challenges to emerge among the many participants. As a result, there are several ongoing efforts to codify the shared knowledge and experience into formal standards. We discuss the Open Grid Framework and the Open Archives Initiatives Protocol for Metadata Harvesting.

2.4.1 OGF

The Open Grid Forum [2] is actively developing standards and specifications with the goal of spreading the adoption of grid-based software systems. The OGF is comprised of business, government, scientific, and academic organizations and focuses on interoperability as the key to expanding

the utilization of grids. Through both advocacy and policy, the OGF represents an independent voice on the role of grids, and their potential to aid modern research.

2.4.2 OAI

The Open Archives Initiative [1] also promotes standards for interoperability and has developed, among others, the Protocol for Metadata Harvesting (OMI-PMH). The goal of the OMI-PMH is to improve application interoperability by enabling consistency in the way *metadata* (data about data) is exposed, accessed, and interpreted. By providing a flexible, extensible standard interface to the rich array of application-specific metadata currently stored in non-uniform, distributed repositories, the OAI hopes to facilitate the broader accessibility and usability of distributed data resources.

3 PCS Core Architecture

In this section, we describe the PCS core components. The three PCS manager components, File Manager, Workflow Manager, and Resource Manager, are daemon-like web service components responsible for answering basic questions regarding file locations, metadata, task control and data flow, and resource availability, monitoring, and usage. The three PCS frameworks together implement one of two critical higher level services in data processing systems: (1) managing the ingestion and acquisition of remotely acquired datasets, handled via the Crawler Framework and Push Pull components ; and (2) managing pipeline processing, product ingestion and data production, handled via the PCS Task Wrapper. We will describe each component in greater detail below. The overall PCS architecture described in this architecture is given in Fig. 1.

3.1 File Manager

The File Manager component is responsible for tracking, ingesting and moving file data and metadata between a client system and a server system. The File Manager is an extensible software component that provides an XML-RPC external interface, and a fully tailorable Java-based API for file management. The critical objects managed by the File Manager include:

Products - Collections of one or more files, and their associated Metadata.

Metadata - A map of key to multiple values of descriptive information about a Product.

References - Pointers to a Product file's original location, and to its final resting location within the archive constructed by the File Manager.

Product Type - Descriptive information about a Product that includes what type of file Uniform Resource Identifier (URI) [5] generation scheme to use, the root repository location for a particular Product, and a description of the Product.

Element - A singular Metadata element, such as "Author", or "Creator". Elements may have additional metadata, in the form of the associated definition and even a corresponding Dublin Core [4] attribute.

Versioner - A URI generation scheme for Product Types that defines the location within the archive (built by the File Manager) where a file belonging to a Product (that belongs to the associated Product Type) should be placed.

Each Product contains one or more References, and one Metadata object. Each Product is a member of a single Product Type. The Metadata collected for each Product is defined by a mapping of Product Type to one or more Elements. Each Product Type has an associated Versioner.

3.2 Workflow Manager

The Workflow Manager component is responsible for description, execution, and monitoring of Workflows, using a client, and a server system. Workflows are typically considered to be sequences of tasks, joined together by control flow, and data flow, that must execute in some ordered fashion. Workflows typically generate output data, perform routine management tasks (such as email, etc.), or describe a business's internal routine practices [14]. The Workflow Manager is an extensible software component that provides an XML-RPC external interface, and a fully tailorable Java-based API for workflow management. The critical objects managed by the Workflow Manager include:

Events - are what trigger Workflows to be executed. Events are named, and contain dynamic Metadata information, passed in by the user.

Metadata - a dynamic set of properties, and values, provided to a WorkflowInstance via a user-triggered Event.

Workflow - a description of both the control flow, and data flow of a sequence of tasks (or stages that must be executed in some order.

Workflow Instance - an instance of a Workflow, typically containing additional runtime descriptive information, such as start time, end time, task wall clock time, etc. A WorkflowInstance also contains a shared Metadata context, passed in by the user who triggered the Workflow. This context can be read/written to by the underlying WorkflowTasks, present in a Workflow.

Workflow Tasks - descriptions of data flow, and an underlying process, or stage, that is part of a Workflow.

Workflow Task Instances - the actual executing code, or process, that performs the work in the Workflow Task.

Workflow Task Configuration - static configuration properties, that configure a WorkflowTask.

Workflow Conditions - any pre (or post) conditions on the execution of a WorkflowTask.

Workflow Condition Instances - the actual executing code, or process, that performs the work in the Workflow Condition.

Each Event initiates one or more Workflow Instances, providing a Metadata context (submitted by an external user). Each Workflow Instance is a run-time execution model of a Workflow. Each Workflow contains one or more Workflow Tasks. Each Workflow Task contains a single Workflow Task Configuration, and one or more Workflow Conditions. Each Workflow Task has a corresponding Workflow Task Instance (that it models), as does each Workflow Condition have a corresponding Workflow Condition Instance.

3.3 Resource Manager

The Resource Manager component is responsible for excecution, monitoring and traacking of jobs, storage and networking resources for an underlying set of hardware resources. The Resource Manager is an extensible software component that provides an XML-RPC external interface, and a fully tailorable Java-based API for resource management. The critical objects managed by the Resource Manager include:

Job - an abstract representation of an execution unit, that stores information about an underlying program, or execution that must be run on some hardware node ,including information about the Job Input that the Job requires, information about the job load, and the queue that the job should be submitted to.

Job Input - an abstrct representation of the input that a Job requires.

Job Spec - a complete specification of a Job, including its Job Input, and the Job definition itself.

Job Instance - the physical code that performs the underlying job execution.

Resource Node - an available execution node that a Job is sent to by the Resource Manager.

Each Job Spec contains exactly one Job, and Job Input. Each Job Input is provided to a single Job. Each Job describes a single Job Instance. And finally, each Job is sent to exactly one Resource Node.

3.4 Crawler Framework

The Crawler Framework was an effort to standardize the common ingestion activities that occur both in data acquisition and archival, as well as those that occur in pipeline processing. These types of activities regularly involve identification of files and directories to crawl (based on e.g., mime type, regular expressions, or direct user input), satisfaction of ingestion pre-conditions (e.g., the current crawled file has not been previously ingested), followed by metadata extraction. After metadata extraction, crawled data follows a standard three state lifecycle: (1) *preIngestion* - where e.g., a file may be unzipped or pre-processed prior to ingestion; (2) *postIngest success*, indicating a successful ingestion has occurred and e.g., the origin data file from the ingest area should be deleted; and (3) *postIngest failure*, indicating that ingestion was not successful and some corrective action, e.g,. moving the failed file to a failure area for later examination, should occur.

To date, we have identified three types of Product Crawlers, where each Crawler varies along the lines of customized precondition verification, crawilng strategy, and need for metadata extraction. The StdProductCrawler assumes that a Metadata object has already been generated and included with a Product prior to ingestion, so no further work is required to generate Metadata from a Product – the Product is ready to be ingested. The MetExtractorProductCrawler is responsible for generating a Metadata object dynamically, as files are encountered during the crawling process. Finally, the AutoDetectCrawler uses a content type identification and regular-expressions to identify Product Types dynamically, and then defaults to the behavior of the MetExtractorProductCrawler for Product Types identified via content detection. The critical objects managed by the Crawler Framework are:

Crawler Action - is attached to one or more of the three phases, and when a ProductCrawler enters a given phases, all the CrawlerActions attached to that phase are executed. The valid phases are: preIngest, postIngestSuccess and postIngestFailure.

Precondition Comparator - is used by MetExtractorProductCrawler and AutoDetectProductCrawler. They are part of those ProductCrawlers customized implementation of precondition verification that identify appropriate times to stifle or allow metadata extractor and ultimately ingestion, to occur.

Metadata Extractor - is run by the MetExtractorProductCrawler and the AutoDetectProductCrawler to generate Metadata for a Product file based on some business rules and logic.

3.5 Push Pull Framework

The Crawler Framework supports many generic ingestion services, including metadata extraction, crawling, and ingestion, however, one service that necessitated further work was the development of a protocol layer allowing a ProductCrawler to obtain content using protocol plugins that download content using implementations of remote protocols such as HTTP, FTP, WinNT file system, HTTPS, etc.

The Push Pull Framework is responsible for remote data acquisition and acceptance over modern web protocols, such as those mentioned above. The Push Pull Framework is flexible in that it provides the ability to plug in different Metadata Extractors, Data Protocols, Content Types, etc. The framework supports parallel file transfers and data downloads, email-based push data acceptance using IMAP, SMTP protocols, and the ability to configure "Virtual" remote directories (based on Metadata such as Date/Time) from which files can be downloaded.

The critical objects managed by the Push Pull Framework are:

Retrieval Method - defines the manner in which files are retrieved from remote sites. It is given a configuration file, a the Parser for the file, and a FileRetrievalSystem (which handles all the complexities of multi-threaded file downloading). There are currently two out-of-the-box RetrievalMethods: RemoteCrawler and ListRetriever. RemoteCrawler is a configurable remote site directory and file regular expression filterable crawler. ListRetriever will download a given list of file URIs [5].

Parser - parses a given configuration file into a VirtualFileStructure which is use to filter URIs to download.

Protocol - handles file transfer and communication via some transfer protocol. Currently implemented Protocols include: sftp, ftp, http, imaps, file (localhost).

3.6 PCS Task Wrapper

The PCS Task Wrapper framework is responsible for standardizing the setup, process initiation, execution and file management tasks surrounding execution of NASA Product Generation Executives, or PGEs. PGEs codify a scientific algorithm, some step in the overall scientific process involved in a mission science workflow.

The PCS Task Wrapper provides a stable operating environment to the underlying PGE during its execution lifecycle. If the PGE requires a file, or metadata regarding the file, the PCS Task Wrapper is responsible for delivering that information to the PGE in a manner that meets its requirements. If the PGE requires knowledge of upstream or downstream PGEs in a sequence of executions, that information is also made available, and finally if information regarding disk space, node information such as CPU availability, etc., is required the PCS Task Wrapper provides this information to the underlying PGE. After this information is collected, the PGE is executed and its output Product file and Metadata generation is managed via the PCS Task Wrapper framework. The PCS Task Wrapper is responsible for marshalling output Products and Metadata back to the File Manager for use in downstream data processing and pedigree. In support of this, the PCS Task Wrapper leverages the Crawler Framework to ingest (during pipeline processing) the output Product files and Metadata produced by the PGE.

As can be gleaned from the above discussion, the PGE Task Wrapper is really the unifying bridge between the execution of a step in the overall processing pipeline, and the available PCS component services and the information that they collectively manage.

The critical objects managed by the PCS Task Wrapper are:

PGETaskInstance - an abstract class which contains a generalized set of actions usually performed when running PGEs. Every variable and method is protected, thus allowing subclasses to easily modify just those generalized actions which need to be customized for different PGE.

Pge Config File Builder - builds a PgeConfig object and set additional Metadata which codifies the information necessary for orchestrating a PGE through its lifecycle. The PCS Task Wrapper is based on a simple but powerful XML syntax which allows a scientist to simply fill out an xml file to describe the necessary steps to execute a PGE.

Config File Property Adder - builds the Pge Config file object and sets custom PGE Metadata. This allows for a general PgeConfigBuilder with different ConfigFilePropertyAdders for setting PGE specific fields in the PgeConfig object.

Science Pge Config File Writer - passes a PGE run information via configuration files. This object allows for any number of config files in any desired format to be generated describing PGE input and those files to be delivered to the PGE. The PCS Task Wrapper provides existing implementations, including a de-

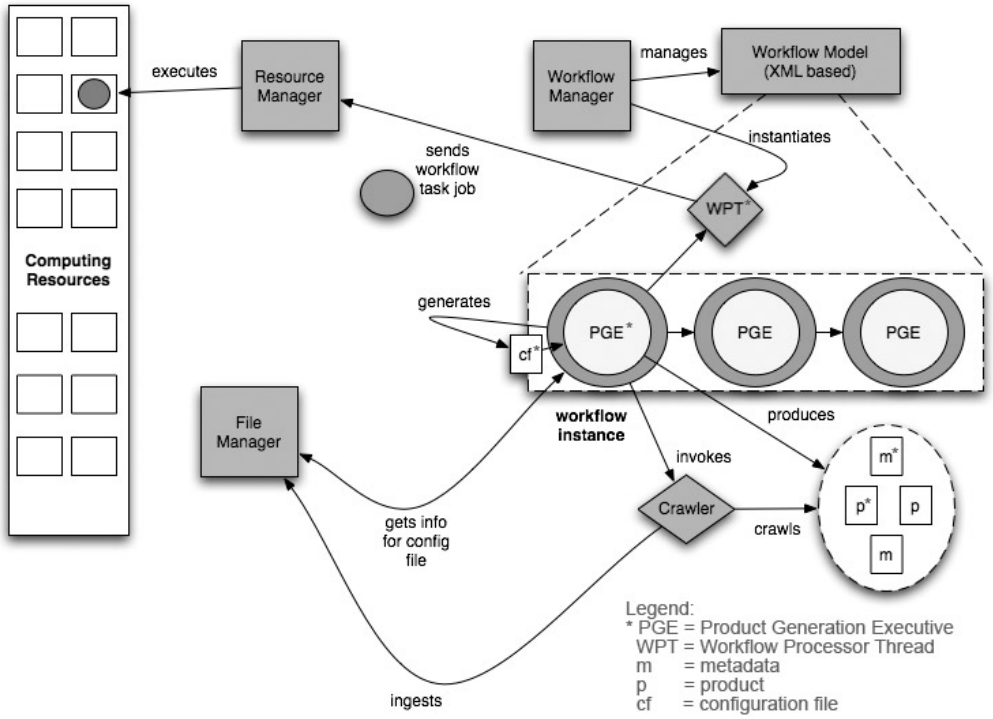

Figure 1. Component Interaction Within the PCS

fault XML Stylesheet Language (XSL) Transformation based SciPgeConfigFileWriter.

Pcs Met File Writer - aids in generating Metadata objects associated with PGE output products.

4 Experience and Evaluation

We have successfully applied the Process Control System (PCS) to existing NASA missions: the Orbiting Carbon Observatory (OCO) mission, and the NPP Sounder PEATE mission. Both systems involve tasks such as high throughput job processing, terabyte-scale data management, and science computing facilities.

4.1 Orbiting Carbon Observatory Mission

On OCO, the mission is using the File Manager to ingest MODIS, CloudSat and other ancillary data products for use in the high performance Level 2 Science Algorithm. To date, OCO has already used the PCS software to process over four terabytes of Fourier Transform Spectrometer (FTS) data provided by ground-based instruments located around the country (e.g., Park falls, Montana, and Darwin, Australia), and has used the software to support Instrument Thermal Vacuum (TVAC) testing, processing 100% of all data taken by the OCO instrument during TVAC. Also, the PCS supports a science computing facility in which variants of scientific software can be excursive prior to inclusion in an operations Pipeline.

4.2 NPP Sounder PEATE Mission

Specifically NPP Sounder PEATE has already used the File Manager and Workflow Manager to ingest and process hundreds of gigabytes of IASI data (and is in preparation to accept CRIMS data). Also on PEATE, the PCS is currently being used to re-catalog over fifteen million existing science data products from the NASA AIRS missions TLSCF. In addition, the Resource Manager will be used on NPP to support job processing across an eighty-node cluster.

4.3 Further Applications

In addition to the two aforementioned NASA missions, the PCS framework is being leveraged on reimbursable work for the National Cancer Institute (NCI)'s Early Detection Research Network (EDRN) [6]. JPL leads the informatics efforts on EDRN, and the PCS framework is being used in the collection, annotation and dissemination of raw scientific data supporting the early detection of cancer to scientists across the country.

In the next year, PCS will also be used to support a new JPL-led NASA mission, the Soil Moisture Active Passive (SMAP) mission. The science computing facility designs on OCO and NPP have been used to create an algorithm testbed for SMAP scientists early in the design phase of the mission so that software integration risks can be mitigated during mission development [13].

5 Conclusions and Future Work

While the norm for earth science missions has been for each mission to develop their own one-off science data system from scratch, the continual decrease in mission funding combined with the exponential increase in mission complexity (data volume and processing throughput) over the last decade has made this approach passé and risky. It was clear that the need for a new approach was eminent.

To this end, we have developed a standards-based software framework to provide common science data system services that yields the benefits of reuse while remaining adaptable to address the requirements that are unique to the customer. This reusable software is centered around the most basic science data system functions that support file and metadata management, workflow management, and resource management. Additional frameworks augment the core capabilities to provide automation for remote data acquisition, data ingestion and standard pipeline processing. This reusable software framework is the Process Control System (PCS) we have described in this paper.

While the PCS has successfully supported the Orbiting Carbon Observatory (OCO) and NPP Sounder PEATE missions, upcoming missions in NASAs Decadal Survey present additional challenges. The JPL-led Soil Moisture Active Passive (SMAP) Mission (currently in formulation phase) will be using the PCS not only for operations, but also for the algorithm testbed and the science computing facility. Providing the operational infrastructure to the algorithm team early in the mission lifecycle will greatly reduce the cost and risk of development-to-operations for the most costly and risky aspect of most earth science data systems, the algorithms. However, this also means that easy integration of algorithms and dynamic workflow specification are our current focus for extending the PCS capabilities. Not far behind SMAP is another JPL-led mission, Deformation, Ecosystem Structure and Dynamics of Ice (DESDynI) Mission. The challenges of DESDynI are requiring us to consider the deployment of PCS components to support a grid architecture, supporting distributed file management and processing capabilities supported by centralized access to a virtual science data system.

Acknowledgements

This effort was supported by the Jet Propulsion Laboratory, managed by the California Institute of Technology under a contract with the National Aeronautics and Space Administration.

References

[1] Open archives initiative, http://www.openarchives.org.
[2] Open grid forum, http://www.ogf.org.
[3] S4pa, http://daac.gsfc.nasa.gov/techlab/s4pa/index.shtml.
[4] Dublin core metadata element set, 1999.
[5] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifiers (uri): Generic syntax. Technical Report RFC 2396, 1998.
[6] D. Crichton, S. Kelly, C. Mattmann, Q. Xiao, J. S. Hughes, J. Oh, M. Thornquist, D. Johnsey, S. Srivastava, L. Essermann, and W. Bigbee. A distributed information services architecture to support biomarker discovery in early detection of cancer. In *e-Science*, page 44, 2006.
[7] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, and M. Livny. *Pegasus: Mapping Scientific Workflows onto the Grid*. 2004.
[8] I. Foster. The anatomy of the grid: Enabling scalable virtual organizations. pages 6–7, 2001.
[9] I. Foster. Globus toolkit version 4: Software for service-oriented systems. pages 2–13. 2005.
[10] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke. Condor-g: A computation management agent for multi-institutional grids. *Cluster Computing*, 5(3):237–246, July 2002.
[11] M. J. Litzkow, M. Livny, and M. W. Mutka. Condor-a hunter of idle workstations. pages 104–111, 1988.
[12] C. Mattmann, D. J. Crichton, N. Medvidovic, and S. Hughes. A software architecture-based framework for highly distributed and data intensive scientific applications. In *ICSE*, pages 721–730, 2006.
[13] D. Woollard, O. ig Kwoun, T. Bicknell, S. Dunbar, and K. Leung. A science data system approach for the smap mission. In *IEEE Radar*, 2009.
[14] D. Woollard, N. Medvidovic, Y. Gil, and C. A. Mattmann. Scientific software as workflows: From discovery to distribution. *Software, IEEE*, 25(4):37–43, 2008.
[15] J. Yu and R. Buyya. A taxonomy of workflow management systems for grid computing, Apr 2005.