End User Data Scheduling with PDMS

Abhishek Gaurav, Nayden Markatchev, Philip Rizk and Rob Simmonds
Grid Research Centre, University of Calgary.
http://grid.ucalgary.ca

1 Introduction

The Proactive Data Management System (PDMS) [13] is a tool designed to manage large datasets in grid environments. It can be used by other higher level software tools in a secure fashion to manage files based on metadata describing the contents of the files. This document describes how PDMS can be used with tools that schedule the movement of data to and from the resources that use or produce the data.

The rest of the document is laid out as follows. Section 2 describes the PDMS interfaces. Section 3 explains the workflows and meta-schedulers and shows how PDMS can be integrated with the workflow managers. Section 4 explains the Virtual Data Toolkit and integration of PDMS with it. Finally, the document is summarized in Section 5.

2 PDMS Interfaces

This section describes the interfaces provided by PDMS; namely the command line tools and a Web services API. These interfaces can be used by other tools to allow secure access to a PDMS server. The interfaces employ GSI for secure communication between the clients and the PDMS server, as described in User Management Issues in PDMS [14].

2.1 Command Line Tools

A PDMS deployment includes a command line client that allows interaction with a remote PDMS server using Unix style commands. Specifically, the client enables registration of a dataset to the PDMS server, submission of replication requests and monitoring progress of replication requests. The following examples show the use of the command line tool.

- Create a logical collection:
  PDMSClient create collection ⟨*collection_name*⟩
  A collection is an aggregation or group of similar data items or other logical collections that aid in group authorization of files rather than individual files.

- Register a dataset with a collection:
  PDMSClient register ⟨*file*⟩
  The registration of a dataset with the PDMS system provides information about datafiles to be managed by PDMS, namely the metatdata attributes for the data and the physical locations of the data. The metatdata attributes and the physical locations are provided in the ⟨*file*⟩.

- Replicate the registered data to a destination site:
  PDMSClient replicate ⟨*dest_host*⟩ ⟨*dest_path*⟩ ⟨*options*⟩
  This command starts the replication process. The ⟨*options*⟩ contain the metadata information that identifies certain data items. PDMS uses MCS to find logical files that match the metadata query and then uses RLS to find the physical locations of the data items. PDMS then starts the data transfers using the RFT service.

- Get the status of the replication jobs:
  PDMSClient repstatus alljobs|⟨*job_id*⟩
  This command monitors and reports the status of the replication requests. Status reports can be obtained either for a particular replication request or for all the replication requests for which the user is authorized to obtain information.

- Get the listing of objects:
  PDMSClient list collections — **lfns** This command is used to list the collections or the logical file names that exist within the system.

- Deletion of Objects:
  PDMSClient delete ⟨*options*⟩
  Delete operation removes objects, *i.e.*, collections, lfns or files from the PDMS. Delete operation within PDMS can have different semantics and can happen at multiple levels. For example, when deleting a collection should the underlying actual files be also deleted or just left at storage site with only the corresponding meta-data removed; when deleting files should all the replicas be deleted or should at least one copy be retained, etc. The ⟨*options*⟩ in the deletion command is used to specify the actual objects that needs to be deleted.

- Modify Access Control Lists:
  PDMSClient grantPermission ⟨*options*⟩
  The PDMS server can be configured to control access to the objects that it handles. By default, the user creating an object gets the read, write and manage permissions on that object. This command is used to grant and/or revoke access permissions to other users. The ⟨*options*⟩ is used to specify the target user(s) and the permissions being distributed.

- Modfiy Metadata Attributes:
  PDMSClient modify-atts ⟨*options*⟩
  This command is used to add, remove or modify the metadata attributes of the objects

within PDMS. It is useful for the situations when new attributes need to be associted and/or the existing attributes need to be removed or modified to the objects.

A detailed explanation of the commands explained above can be found in the PDMS Design Document [12].

2.2 Web Services API

The second type of interface that PDMS provides is a Web services application programming interface (API). Using this API it is possible to write custom clients that can interact directly with the PDMS server. PDMS uses the Web Services Description Language (WSDL) to define its API interface. The Web services interface enables PDMS to talk with other Web services in a unified way.

The Web services interface can be used to write clients that calls PDMS method using SOAP messages. Such clients would need to generate a stub that serializes and deserializes the SOAP messages. The SOAP messages could be bound with the popular network protocols like HTTP, FTP, SMTP,etc.

The interface, written using WSDL, is described in the PDMS WSDL Document [15]. It defines operations for creating collections, registering files, replicating data, retrieving a job's status, etc., along with the various messages and data types to be used while calling those operations from the client code.

3 Workflow and Meta-scheduler Integration

A workflow in a grid environment is automation and orchestration of the various processes of a grid-wide application. Many applications are composed of a set of other services and applications. For example, the input data stage-ins and output data stage-outs along with the actual computations at remote sites, compose an application that involves two services, the data transfer and job execution services. Workflow managers are the software tools that track and manage the temporal dependencies between the individual tasks in a workflow.

A meta-scheduler is a software tool that sits on top of the individual schedulers and manages the spatial dependencies of the tasks of a workflow. Meta-schedulers track the resource availability and accordingly schedules the tasks to the suitable resources.

Condor-G [2], DAGMan [1] and Stork [5] are the popular meta-schedulers and workflow managers from the Condor project at University of Wisconsin. These tools can be integrated with PDMS to manage the data management portions of the workflows and applications in a sense PDMS would be used by those tools.

3.1 Condor-G, Dagman and Stork

Condor-G is a job manager that uses Condor software and the Globus Toolkit [11] to provide secure and cross-organizational meta-scheduling functionality.

Condor-G manages the matching, and submission of a local queue of jobs to an arbitrary number of resources that comply with the Globus job submission protocol. This includes basic functionality to stage data in and out as well as management of credential.

DAGMan (Directed Acyclic Graph Manager) is the workflow manager part of the Condor group of projects. It allows users to specify a directed acyclic graph (DAG) of tasks to be executed with complex relationships and dependencies. DAGMan then efficiently schedules the ordering of task executions and uses Condor-G to get the individual tasks executed on the available resources. DAGMan is fault-tolerant in a sense that it has ability to retry failed tasks at a later time.

Stork is a data placement scheduler from the Condor project that enables efficient and reliable data transfers. Similar to how Condor-G maintains a queue of computational jobs, Stork maintains a queue of data placement jobs, like storage space allocation and de-allocation, finding location of data items and the actual data transfers themselves. Stork interacts with the several data storage protocols like NeST, SRM, SRB, UniTree and DiskRouter and data transfer protocols like FTP, HTTP and GridFTP [7]. Stork provisions for storage space allocation and de-allocation and data movement operations. It interacts with DAGMan to include these operations in a workflow. DAGMan submits the computational jobs to Condor-G and data placement jobs to Stork.

3.2 Integrating PDMS

PDMS can be integrated with the workflow managers and the meta-schedulers. Condor-G scripts can be generated to execute the PDMS command line client tools. These scripts will be submitted to the Condor-G where they will be scheduled to run on a particular resource. The PDMS client would need to be installed on the potential resources where the scripts containing PDMS commands will be executed. In many cases, this will only be the server condor is running on. the PDMS client does not have to be on every resource. The PDMS server can be present at any remote machine and will be invoked with SOAP messages.

Similarly, the tools generating or the users hand-writing the DAGMan scripts can write the scripts to execute the PDMS command line client tool. The scripts to execute the PDMS client form one of the jobs of the workflow and can be orchestrated within the DAGMan scripts according to the dependencies with other jobs of the workflow.

PDMS can also be integrated with Stork in a sense that the data migration tasks within PDMS can be submitted to and handled by Stork. Using Stork for data movement purposes would replace the RFT and GridFTP component in the current design of PDMS. As Stork supports a variety of data transfer protocols, replacing RFT with Stork would help PDMS in interacting with several heterogeneous systems. PDMS, in its current design, lacks the capability of provisioning a guaranteed storage space for the data that it manages and keeps track of across various sites. PDMS has no control and does not have the ability to affect the local site's storage space allocation policies. Stork interacts with storage manager systems like, NeST, SRB, UniTree, etc. to provide storage space allocation and de-allocation. This feature of Stork would further benefit PDMS by allowing the PDMS managed data to have guaranteed storage space. From an implementation perspective, instead of using the RFT

API for the actual data transfers, PDMS would have to use the Stork API for the transfers and for negotiating with the storage resource managers for the space allocation. It is to be noted that the PDMS interface to its client would remain unaffected by using Stork instead of RFT and GridFTP.

It should be noted that all the meta-schedulers and workflow managers, discussed in this section, do not deal with the concept of meta-data whereas the central concept behind PDMS is data management using meta-data. This fact could be safely worked around and the meta-schedulers and workflow managers can be made to implicitly handle meta-data. The commands for registration of data using meta-data can be included in the scripts that are generated for data transfers, along with the replication commands. So, the sequence of data registration and replication commands would be treated as one atomic job by the above mentioned workflow managers and the meta-schedulers.

4 Virtual Data Toolkit

The Virtual Data Toolkit (VDT) [6] is a software package that includes the Globus Toolkit version 2.4 and a large collection of tools that extend it. VDT uses the Pacman packaging tool [4] for an automated and easy installation and configuration of the system. VDT includes versions of Condor-G and Dagman. It also includes some higher level tools, Chimera and Pegasus, that will be described in this section.

The Chimera [8] and Pegasus [10] components of the VDT, together also known as the Virtual Data System (VDS), are used for expressing, executing and tracking workflows. VDS has been particularly motivated by the concept of virtual data at the GriPhyn project [3].

Note that, at the time of writing, VDT does not provide many of the tools and services required by PDMS. It uses the older GT2.4 version of Globus and the Condor-G version provided will not inter operate with GT4 which is used by PDMS.

4.1 Chimera

A large amount of data in some scientific communities is often derived from some base data using computations rather than being measured. The GriPhyn project terms such data as virtual data. Chimera [8] is a system designed to capture and track the generation of virtual data. Chimera uses a *virtual data catalog* (VDC) to record virtual data information. The information stored in VDC can be used to create and re-create the data. Chimera also consists of a *virtual data language* (VDL) that captures and formalizes abstract descriptions of how a computation can be invoked, what parameters it needs, what files it takes as input, what environment is required and so forth. Such descriptions are called *transformations* in VDS terminology. An invocation of a transformation with a specific set of input parameters and files is called a *derivation*. Each invocation of a transformation is recorded to a central database for a large scientific collaboration. The virtual data language also allows for query functions that allow a user or application to search the virtual data catalog for a transformation or derivation.

Chimera is also known as the abstract workflow planner component of VDT. For a given request for a virtual data object, the Chimera planner generates a directed acyclic graph (DAG) for use by DAGMan, that manages the computation of the data object. The Chimera planner first finds which derivation contains the requested file as output, then for each input of the associated transformation finds the derivation that contains it as output, iterating until all the dependencies are resolved. Such a DAG, produced by Chimera, is an abstract workflow because it does not contain any information as to what resources to use, where the actual data is located, where the executables are located, etc.

4.2 Pegasus

Pegasus [10] is the concrete workflow planner component of VDT. It takes in a description of the abstract workflow, produced by Chimera, and produces a concrete workflow, mapping it onto the Grid environment. Mapping an abstract workflow description to a concrete executable workflow involves finding the resources that are available and can perform computations, finding the locations of the data used in the workflow and the necessary executables.

Pegasus uses RLS, MDS and a VDT specific Transformation Catalog (a catalog of the executables) to map the logical entities of the abstract workflow to the actual physical entities. RLS is used to find out the physical locations of the files involved in the workflow. MDS is used to locate the resources where the computations can take place and the Transformation Catalog is used to locate the executables at the compute sites and the right environment variables needed to make executions happen.

In addition to generating concrete workflows, Pegasus also performs workflow reduction. If certain data objects within abstract workflow already exist, Pegasus can reuse them and thus reduce the complexity of the concrete workflow. In doing so, Pegasus assumes that it is more costly to execute a transformation than to access the results of that transformation.

Apart from the actual application level transformations of the workflow, Pegasus also includes the jobs that handle the data transfers before, in between and after the actual transformations. The transfer protocol that should be used to manage the data transfers is specified in the transformation catalog, for example, GridFtp or RFT. Pegasus generates a condor submit script for each of the jobs in the workflow and submits the concrete workflow to Condor Dagman for execution.

4.3 Integrating PDMS

PDMS can be used within the virtual data system to manage the data transfers during generation of a "virtual" data object. Often, due to the nature of the virtual data generation applications, the transformations are executed at multiple remote sites. Input data needs to be transferred from storage sites to compute sites, intermediate data transferred in-between different compute sites and final output data transferred to the archival sites. As mentioned in Section 4.2, the tool used to perform the data transfers is specified in the transformation catalog, the entry for which contains the location of the executable and a description of the

environment variables needed to execute the data transfer program. PDMS can be specified in the transformation catalog, along with the necessary environment variables.

It is to be noted that in using PDMS in the above mentioned way, only the PDMS client needs to be available at the site where DAGMan scripts are generated and executed. The PDMS server can be present at a remote site, the information about which should be available from MDS that Pegasus uses to locate the resources. The remote PDMS server would perform the transfers to and from archiving sites and in between compute sites.

As mentioned earlier, at the time of writing, VDT includes Globus Toolkit version 2.4, which is not what PDMS uses. A separate Globus Toolkit version 4.0 would be required to be installed at the site where DAGMan scripts are generated and executed.

Pegasus can be used in a different configuration that does not use the abstract workflows produced by Chimera [9]. In this configuration, Pegasus uses the application level metadata to automatically construct an abstract workflow, and thus replacing Chimera's functionality. It is also claimed in [9] that this configuration has been used successfully in the LIGO pulsar search project. However, the version of Pegasus that can be used in this configuration has not been publicly released at the time this document was written. PDMS can be integrated into this scenario however it would require the Pegasus code to be changed so that is uses PDMS for the data management instead of its own way of managing the data during workflow execution.

5 Summary

This document has described the various systems that schedule the data movement and explained how PDMS can be used in the context of those systems. It surveyed the workflow systems and meta-schedulers like, Condor, Stork and Virtual Data Toolkit. While integrating PDMS with systems like Condor has already been made possible, making PDMS work with VDT still appear to be a challenging task.

References

[1] Condor dagman web site. http://www.cs.wisc.edu/condor/dagman.

[2] Condor-g project web site. http://www.cs.wisc.edu/condor/condorg.

[3] Grid physics network. http://www.griphyn.org.

[4] Pacman web site. http://physics.bu.edu/ youssef/pacman.

[5] Stork project web site. http://www.cs.wisc.edu/condor/stork.

[6] Virtual data toolkit web site. http://vdt.cs.wisc.edu.

[7] W. Allcock, J. Bester, J. Bresnahan, S. Meder, P. Plaszczak, and S. Tuecke. GridFTP: protocol extensions to FTP for the Grid. Proposed Recommendation GFD-R-P.020, Global Grid Forum, April 2003.

[8] J. Annis, Y. Zhao, J. Voeckler, M. Wilde, S. Kent, and I. Foster. Applying chimera virtual data concepts to cluster finding in the sloan sky survey, 2002.

[9] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, S. Koranda, A. Lazzarini, G. Mehta, M. A. Papa, and K. Vahi. Pegasus and the pulsar search: From metadata to execution on the grid. In *PPAM*, pages 821–830, 2003.

[10] 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. In *European Across Grids Conference*, pages 11–20, 2004.

[11] I. Foster. Globus Toolkit version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing*, pages 2–13, 2005.

[12] U. of Calgary Grid Research Centre. PDMS design document. GRC Internal Report.

[13] U. of Calgary Grid Research Centre. PDMS project web site.

[14] U. of Calgary Grid Research Centre. PDMS user management document. GRC Internal report.

[15] U. of Calgary Grid Research Centre. PDMS WSDL document. GRC Internal Report.