

Gridflow Description, Query, and Execution at SCEC using the SDSC Matrix

Jonathan Weinberg, Arun Jagatheesan, Allen Ding, Marcio Faerman and Yuanfang Hu
{jonw | arun | alding | mfaerman | yhu } @sdsc.edu
San Diego Supercomputer Center
University of California, San Diego
La Jolla, CA 92093

Abstract

While conventional workflow systems have been around for many years, the deployment of analogous systems onto a grid infrastructure introduces a number of unique questions and challenges. Innovative approaches to grid workflow (gridflow) are needed to leverage the heterogeneity, autonomy, dynamic behavior, and wide-area distribution that characterize grid resources. The Matrix Project carries out research and development to deliver the language descriptions and protocols necessary to build collaborative gridflow management systems for the emerging grid infrastructures. We describe here our activities to date including development of the Data Grid Language (DGL) and the usage of the Matrix gridflow management system by the Southern California Earthquake Center (SCEC) to manage its gridflows.

1. Introduction

Like workflow, grid workflow (gridflow) is the automation of a business process whereby data and tasks are passed from one participant to another according to some set of procedural rules. Unlike conventional workflow, gridflows are executed on grid infrastructures formed by inter-organizational collaborations. These organizations make autonomous resource-sharing decisions according to dynamic policies. Hence, a Gridflow Management System (GfMS) must have the ability to dynamically discover and utilize the distributed grid resources available to it as well as collaborate with other inter-organizational GfMS's.

Since the physical resources that would be used in a gridflow are decided only at runtime (*late binding*), the description of gridflows must have only abstract references to the underlying infrastructure. A new language to describe and query gridflows that can be

executed by GfMS's is required and is essential to applying workflow technologies to the grid.

2. Gridflow Language Requirements

A single expressive language that can describe, control, and query gridflows in a highly distributed setting is needed. Some requirements are:

- Provide sufficient gridflow control patterns
- Abstract description of grid infrastructure dependencies (without physically defining the resources)
- Describe both process-based and data-based gridflow dependencies
- Support dynamic definition and update of gridflow metadata and annotations
- Support gridflow querying
- Support runtime management of a gridflow
- Arbitrarily extensible schema

3. Data Grid Language

We have addressed the need for a gridflow description language by proposing the *Data Grid Language* (DGL), an XML schema for expressing and executing gridflows. We provide several facilities to facilitate end-user development of DGL programs, including a Java API and a graphical IDE. Further, as part of the SDSC Matrix Project [1], we have implemented a GfMS called Matrix that accepts and executes DGL requests.

There are two basic types of gridflow descriptions that can be written in DGL and sent to a GfMS: flow requests and status requests. A flow request describes a gridflow that the sender wishes to execute along with some authentication information. When a user or application sends a flow request, the GfMS returns an acknowledgement in DGL containing a unique identifier for that transaction. At any point thereafter, the GfMS may be queried using DGL status requests,

to reveal information such as which steps had completed properly, what execution path the gridflow followed, and the general results of any step in the gridflow.

A flow request describes gridflows through recursive control structures called “flows” and concrete operations called “steps”. A step is simply an atomic operation that accepts parameters and performs a set operation using those parameters. Users can currently create and execute arbitrary steps by specifying a URI where the GfMS can find the executable or by writing their own step implementations.

Flows are recursive control structures that can describe how to execute steps. Each flow is a certain predefined gridflow pattern that dictates how its contents should be executed. Examples of gridflow patterns used in flows are “sequential”, “parallel”, “while loop”, “for-each loop”, “switch-case” etc., which are very similar to any modern programming language. Using these control structures, users can create arbitrarily complicated gridflow descriptions that are ready to be executed on the grid.

In order to operate these control structures, DGL supports dynamic creation and update of metadata variables and rules. A user can declare variables simply for the purpose of maintaining meta-data (the GfMS can later be queried for these values) or to control state-based branching.

4. Matrix in SCEC Project

The NSF SCEC project in support the Southern California Earthquake Center [7] has successfully used DGL and the Matrix GfMS to ingest collections of files into the Storage Resource Broker Data Grid Management System (SRB DGMS) [2]. In this simple example of gridflow, members of the SCEC project used DGL to describe the extraction of files from remote storage systems, the derivation of logical names and metadata concerning each of those files, and the subsequent ingestion of both into the SRB DGMS. The Data Grid Language was used to describe the gridflow and query its status during and after execution.

5. Future Work

The focus of our current gridflow research is on defining the underlying peer-to-peer protocols by which GfMS's can coordinate with one another over the WAN. There is a need to define a set of powerful

and minimally verbose protocols by which GfMS's can carry out such collaboration. Aggravating this challenge is the difficulty of distributed coordination over wide area networks (e.g. atomic commit, rollback, etc.) Once these underlying protocols are built, any type of scheduling strategy could be implemented on top of these gridflow protocols.

We are currently working on extensions to Matrix that will enable us to deploy it more readily in production settings. We are addressing a need in Matrix for a persistent storage mechanism as well as more comprehensive interfaces to existing remote execution systems such as Condor-G [3].

As part of the drive to make DGL an accepted standard for gridflow description and execution, we are looking to integrate or merge DGL with BPEL [4], the emergent standard for conventional workflow description. Also, we will try to integrate the DGL and P2P Gridflow concepts with the Chimera Virtual Data System [5] and Pegasus planner [8], both used in the NSF GriPhyN Project [6]

6. References

- [1] SDSC Matrix Project
<http://www.npaci.edu/dice/srb/matrix/>
- [2] Moore, R.W., Jagatheesan, A., Rajasekar, A., Wan, M. and Schroeder, W., “Data Grid Management Systems,” *Proceedings of the 21st IEEE/NASA Conference on Mass Storage Systems and Technologies*, 2004, Maryland.
- [3] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC 10)*, San Francisco, California, August 7-9, 2001.
- [4] Business Process Execution Language (BPEL) for Web Services Version 1.0," BEA, IBM and Microsoft, Aug-2002
- [5] Foster, I., Voeckler, J., Wilde, M. and Zhao, Y., “Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation”. In *Scientific and Statistical Database Management*, (2002).
- [6] GriPhyN: Grid Physics Network, <http://www.griphyn.org>
- [7] SCEC: Southern California Earthquake Center
<http://www.scec.org>
- [8] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. Su, K. Vahi and M. Livny , “Pegasus: Mapping scientific workflows onto the grid,” *Across Grids Conference 2004*, Nicosia, Cyprus.