Chapter 25
An Interoperable GridWorkflow Management System

Maria Mirto, Marco Passante, Italo Epicoco, Giovanni Aloisio

University of Salento, Italy

Abstract

A WorkFlow Management System (WFMS) is a fundamental component enabling to integrate data, applications and a wide set of project resources. Although a number of scientific WFMSs support this task, many analysis pipelines require large-scale Grid computing infrastructures to cope with their high compute and storage requirements. Such scientific workflows complicate the management of resources, especially in cases where they are offered by several resource providers, managed by different Grid middleware, since resource access must be synchronised in advance to allow reliable workflow execution. Different types of Grid middleware such as gLite, Unicore and Globus are used around the world and may cause interoperability issues if applications involve two or more of them. In this paper we describe the ProGenGrid Workflow Management System which the main goal is to provide interoperability among these different grid middleware when executing workflows. It allows the composition of batch; parameter sweep and MPI based jobs. The ProGenGrid engine implements the logic to execute such jobs by using a standard language OGF compliant such as JSDL that has been extended for this purpose. Currently, we are testing our system on some bioinformatics case studies in the International Laboratory of Bioinformatics (LIBI) Project (www.libi.it).

1. Introduction

Workflows are commonly used to execute data - and compute - intensive analysis pipelines. Such workflows are typically composed of analysis software components interconnected through data dependencies. Although
a number of scientific WFMSs support this task on homogeneous Grids, many analysis pipelines in different domains, such as bioinformatics, require large-scale, heterogeneous Grid computing infrastructures to cope with their high compute and storage requirements. Moreover as the Grid is becoming the day-to-day infrastructure for e-Science and e-Business, support is required for more complex and demanding application scenarios in which not just a single resource is required, but a set of resources with certain dependencies. Such workflows complicate the management of resources, especially in cases where they are offered by several different resource providers, since resource access must be synchronised in advance to allow reliable workflow execution. Different types of middleware such as gLite [21] (http://www.glite.org), Unicore [4] (http://www.unicore.eu) and Globus [14] (http://www.globus.org), etc. are used around the world and may cause interoperability issues if applications (i.e. workflows) involve two or more.

Usually a Grid job cannot use resources that are integrated by different middleware, because these middleware have various architectures and use divergence methods.

Establishing interoperability between Grids is vital in order to bridge these differences and enable virtual organizations to access resources at the institutions, that is independent of the Grid project’s affiliation. Without Grid interoperability, collaboration would be artificially limited to one Grid or the collaboration would have to create multiple virtual organizations and manage the diversity itself.

In this paper we focus on the implementation of the ProGenGridWorkFlowManagement System (WFMS) that solves several interoperability issues such as the submission and monitoring of jobs among three different grid middleware: gLite, Globus and Unicore. Our WFMS is composed by an editor (WF editor), that allows the composition of the applications and the input of data through an arbitrary graph that can contain cycles, and by an engine (WF engine), based on a meta scheduler named GMS (Grid Meta Scheduler), that retrieves the references to involved executables and data files as well as specific resources, chooses the opportune resource broker on the basis of the selected application in the workflow, and submits and monitors the jobs on it. The design of the meta scheduler was driven mainly by the definition of a modular architecture which can be easily extended through a plug-in based approach. In order for the meta scheduler to support different grid middleware a specific adaptor for the submission and monitoring of jobs on target middleware has been devel-