Chapter 6.5
Error Recovery for SLA-Based Workflows within the Business Grid

Dang Minh Quan
International University in Germany, Germany

Jörn Altmann
Seoul National University, South Korea

Laurence T. Yang
St. Francis Xavier University, Canada

ABSTRACT

This chapter describes the error recovery mechanisms in the system handling the Grid-based workflow within the Service Level Agreement (SLA) context. It classifies the errors into two main categories. The first is the large-scale errors when one or several Grid sites are detached from the Grid system at a time. The second is the small-scale errors which may happen inside an RMS. For each type of error, the chapter introduces a recovery mechanism with the SLA context imposing the goal to the mechanisms. The authors believe that it is very useful to have an error recovery framework to avoid or eliminate the negative effects of the errors.

INTRODUCTION

In the Grid Computing environment, many users need the results of their calculations within a specific period of time. Examples of those users are meteorologists running weather forecasting workflows, automobile producers running dynamic fluid simulation workflow (Lovas et al., 2004). Those users are willing to pay for getting their work completed on time. However, this requirement must be agreed on by both, the users and the Grid provider, before the application is executed. This agreement is kept in the Service Level Agreement (SLA) (Sahai et al., 2003). In general, SLAs are defined as an explicit statement of expectations and obligations in a business relationship between service providers and customers. SLAs specify the a-priori negotiated resource

DOI: 10.4018/978-1-4666-0879-5.ch6.5

requirements, the quality of service (QoS), and costs. The application of such an SLA represents a legally binding contract. This is a mandatory prerequisite for the Next Generation Grids. The basic concepts of a system handling the Grid-based workflow within an SLA context are described in the following sections.

Grid-Based Workflow Model

Workflows received enormous attention in the databases and information systems research and development community (Georgakopoulos et al., 1995). According to the definition from the Workflow Management Coalition (WfMC) (Fischer, 2004), a workflow is "*The automation of a business process, in whole or parts, where documents, information or tasks are passed from one participant to another to be processed, according to a set of procedural rules.*" Although business workflows have great influence on research, another class of workflows emerged in sophisticated scientific problem-solving environments, which is called Grid-based workflows. A Grid-based workflow differs slightly from the WfMC definition as it concentrates on intensive computation and data analyzing but not the business process. A Grid-based workflow is characterized by the following features (Singh et al., 1997):

- A Grid-based workflow usually includes many sub-jobs (i.e. applications), which perform data analysis tasks. However, those sub-jobs are not executed freely but in a strict sequence.
- A sub-job in a Grid-based workflow depends tightly on the output data from previous sub-jobs. With incorrect input data, a sub-job will produce wrong results and damage the result of the whole workflow.
- Sub-jobs in the Grid-based workflow are usually computationally intensive. They can be sequential or parallel programs and require a long runtime.
- Grid-based workflows usually require powerful computing facilities (e.g. super-computers or clusters) to run on.

Most of existing Grid-based workflows (Ludtke et al., 1999, Berriman et al., 2003, Lovas et al., 2004) can be presented under Directed Acyclic Graph (DAG) form so only the DAG workflow is considered in this chapter. The user specifies the required resources needed to run each sub-job, the data transfer between sub-jobs, the estimated runtime of each sub-job, and the expected runtime of the whole workflow.

In this chapter, we assume that time is split into slots. Each slot equals a specific period of real time, from 3 to 5 minutes. We use the time slot concept in order to limit the number of possible start-times and end-times of sub-jobs. More over, delaying 3 minutes also has little impact with the customer. It is noted that the data to be transferred between sub-jobs can be very large.

Grid Service Model

The computational Grid includes many High Performance Computing Centers (HPCCs). The resources of each HPCC are managed by a software called local Resource Management System (RMS)[1]. Each RMS has its own unique resource configuration. A resource configuration comprises the number of CPUs, the amount of memory, the storage capacity, the software, the number of experts, and the service price. To ensure that the sub-job can be executed within a dedicated time period, the RMS must support advance resource reservation such as CCS (Hovestadt, 2003). In our model, we reserve three main types of resources: CPU, storage, and expert. The addition of further resources is straightforward.

If two output-input-dependent sub-jobs are executed on the same RMS, it is assumed that the time required for the data transfer equals zero. This can be assumed since all compute nodes in a cluster usually use a shared storage system like