J. Parallel Distrib. Comput. 71 (2011) 343–353

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Deadline-sensitive workflow orchestration without explicit resource control☆

Lavanya Ramakrishnan[a,*], Jeffrey S. Chase[b], Dennis Gannon[c], Daniel Nurmi[d], Rich Wolski[d]

[a] Lawrence Berkeley National Lab, Berkeley, CA, United States
[b] Duke University, Durham, NC, United States
[c] Microsoft Research, Redmond, WA, United States
[d] University of California, Santa Barbara, CA, United States

ARTICLE INFO

Article history:
Received 8 June 2009
Received in revised form
18 November 2010
Accepted 18 November 2010
Available online 8 December 2010

Keywords:
Grid computing
Cloud computing
Workflow scheduling
Orchestration
Deadline-sensitive workflows

ABSTRACT

Deadline-sensitive workflows require careful coordination of user constraints with resource availability. Current distributed resource access models provide varying degrees of resource control: from limited or none in grid batch systems to explicit in cloud systems. Additionally applications experience variability due to competing user loads, performance variations, failures, etc. These variations impact the quality of service (QoS) that goes unaccounted for in planning strategies. In this paper we propose **W**orkflow **OR**chestrator for **D**istributed **S**ystems (WORDS) architecture based on a least common denominator resource model that abstracts the differences and captures the QoS properties provided by grid and cloud systems. We investigate algorithms for effective orchestration (i.e., resource procurement and task mapping) for deadline-sensitive workflows atop the resource abstraction provided in WORDS. Our evaluation compares orchestration methodologies over TeraGrid and Amazon EC2 systems. Experimental results show that WORDS enables effective orchestration possible at reasonable costs on batch queue grid and cloud systems with or without explicit resource control.

1. Introduction

Large scale computations from various scientific endeavors such as drug discovery, weather modeling, and other applications are composed as a sequence of dependent operations or workflows. A number of these workflows have user constraints associated with them including deadline and budget. In addition, these workflows often access shared resources or data and run computations on grid or cloud systems. For example, a weather prediction workflow is triggered by streaming sensor atmospheric data and consists of a number of data-processing steps that use distributed data and resources [8]. This workflow must complete in a timely manner to generate appropriate forecasts and initiate any emergency management measures that might be necessary. Thus deadline-sensitive workflows require careful coordination of workflow tasks with underlying resource behavior to ensure timely completion.

Resource mechanisms and protocols are available today to coordinate grid resources and ensure quality of service (QoS) [5,6,27]. There are tools for workflow planning using performance models [3,10,18] and execution systems or workflow engines for managing runtime environment of workflows [4,15]. Today's planning techniques can provide a "yes" or "no" answer to the question of whether a workflow will meet its constraints (e.g., deadline) on a set of resources. However this information alone is insufficient for deadline-sensitive applications such as weather prediction, given the underlying uncertainty in resources. Users are willing to run the workflow so long as the odds of completion are "reasonable". Users are often willing to pay extra or trade-off application requirements to ensure timely workflow completion. Current systems do not allow these trade-offs or speculative scheduling based on QoS properties of the resources.

Grid and cloud systems provide varying degrees of resource control to an end user. Users interact with grid systems by submitting jobs to a batch queue, which executes the job on the user's behalf once enough resources become available. Cloud systems, unlike batch systems, enable *explicit resource control*, i.e., users request specific quantities and types of resources at specific times. Yet users of both these systems cannot expect strong QoS assurances due to both availability and reliability variations of underlying hardware and software services. Additionally, resource systems lack standardized interfaces and workflow tools interact with these systems using ad hoc mechanisms and comparison of QoS capabilities is extremely difficult.

In this paper, we use the term *workflow orchestration* to describe the holistic, coordinated, dynamic and adaptive approach

☆ The work was performed at Indiana University, Bloomington, IN.

* Corresponding author.
E-mail address: LRamakrishnan@lbl.gov (L. Ramakrishnan).

0743-7315/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2010.11.010

to workflow planning that works with user requirements and variable resource characteristics while being agnostic to specific resource policy or systems. A fundamental research question this paper attempts to address in the context of the WORDS architecture is *how much explicit knowledge of and control over resources is necessary for effective workflow orchestration over grid and cloud systems?* To answer this question, we develop a lowest common denominator resource model that is powerful enough to implement workflow orchestration for deadline-sensitive workflows over systems like batch queue and cloud systems with or without explicit resource control. Specifically, we make the following contributions in this paper:

- We develop the **W**orkflow **OR**chestrator for **D**istributed **S**ystems (WORDS) architecture that facilitates the separation of concerns between resource and application layers in distributed resource environments.
- In the context of WORDS we present the design of a resource abstraction that consists of a standard set of interfaces and mechanisms required at the resource layer in grid and cloud systems to implement effective and predictable QoS for end users.
- We develop a probabilistic QoS model in WORDS to account for the uncertainty that comes from the resource layer characteristics.
- We evaluate a number of workflow orchestration strategies on top of the resource abstraction in WORDS for deadline-sensitive workflows.

The rest of this paper is organized as follows. We discuss the WORDS architecture and associated resource abstraction in Sections 2 and 3. We explore some *workflow orchestration* approaches for deadline-sensitive workflows atop the WORDS architecture in Section 4. We expand the orchestration approaches to schedule a workflow set with deadline and accuracy constraints in Section 5. Finally, we compare and contrast various workflow orchestration approaches in the context of scientific workflows over grid and cloud computing systems (Section 6).

2. Overview

In this section, we first define the specific problem associated with deadline-sensitive workflows and detail the WORDS architecture.

2.1. Problem description

A *deadline-sensitive workflow* W_x is a Directed Acyclic Graph (DAG) that must complete by a deadline D for the results of the computation to be useful. For example, meteorological workflows need to complete by a certain deadline since they influence emergency response measures.

The workflows have access to resources $R = \{R_1, \ldots, R_w\}$ across distributed sites. In addition for each task Task_k its execution on resource R_j is given by $[n, T]$ where n is the number of processors required for the task and T denotes execution time of the application.

Resources are available through either grid or cloud mechanisms. Resource procurement is implicit in grid systems. Users submit a job description to a queue managed by batch queueing software such as Maui/PBS. The job waits its turn to acquire resources. When the requested resources become available and the job is at the head of the queue, the job starts executing. The job is killed if it exceeds the requested wall clock time. The users in this environment do not know when exactly their job might start, though more recently there are services [18] that provide the methodology for predicting bounds on the amount of queue wait times. Cloud providers provide stronger guarantees and immediate access to resources through *explicit resource control* since the requests are typically bounded in both time and space. The most prevalent example of a cloud system in operation today is Amazon's EC2 system. In Amazon EC2, resources are accessible to the user almost instantly, with startup time of the instance and image imposing the only delays. In addition there is a diversity between the cost models provided by these systems. However both these systems experience hardware and software failures that make it hard if not impossible to make strong guarantees as those required by deadline-sensitive workflows.

The goal of the workflow orchestration is to find a schedule for workflow W_x on available resource set R such as to meet the deadline D over diverse resource platforms such as grid and cloud systems. In this paper, we present and evaluate an architecture over grid and cloud resources that provides an effective workflow orchestration for deadline-sensitive workflows. The WORDS architecture enables us to quantify the effectiveness of the schedule in meeting the deadline. Next, we describe in this architecture in detail.

2.2. WORDS

Fig. 1 shows the WORDS architecture that introduces a clean separation between the resource and application layers. WORDS receives a specification of a workflow as a directed acyclic graph (DAG) and user constraints (e.g., deadline). The workflow planner communicates user requirements to the resource coordinator which initiates resource procurement. The resource coordinator interacts with both grid and cloud sites through conventional scheduling mechanisms and interfaces.

The resource coordinator interacts with various site-specific resource control mechanisms and returns a Gantt chart to the application layer. The Gantt chart consists of a set of resource slots from different sites and its associated properties. A *resource slot* is an abstract representation of a resource set on a site that has been assigned to the application or user by the resource layer. A resource slot has defined width (i.e., number of processors) and length (i.e., duration). The *resource slot* is central to our resource abstraction. A slot can be resources allocated to a job through the batch queue system or to a user in cloud systems or through advanced reservation or probabilistic mechanisms (more in Section 3.1).

The workflow planner determines a schedule by assigning tasks on the slots using criteria such as computational time, data transfers, success probabilities, cost, etc. (more in Section 4). This process of resource acquisition and task mapping might be iterative with the goal of enhancing the schedule for some or all tasks in the workflow.

The execution system (bottom of Fig. 1), consisting of the workflow engine and web services, is largely orthogonal to the orchestration components. The slot execution manager consults the orchestration system for resource related decisions (i.e., where and when should a task run). The workflow planner cannot anticipate all runtime failures that might occur. The WORDS architecture provides resistance to runtime failures through the execution system that is responsible for detecting deviations from the original schedule or other failures. Further discussion on the execution system and handling runtime failures is beyond the scope of this paper.

The WORDS architecture provides a dynamic, adaptive resource abstraction that the higher-level workflow orchestration can use for planning workflows to meet user constraints. Next, we discuss the resource abstraction provided by WORDS in greater detail.

3. Resource abstraction

Fundamentally grid and cloud computing systems have different access models and policies. However there are also