Chapter 1.24
Domain-Specific Language for Describing Grid Applications

Enis Afgan
University of Alabama at Birmingham, USA

Purushotham Bangalore
University of Alabama at Birmingham, USA

Jeff Gray
University of Alabama at Birmingham, USA

ABSTRACT

Grid computing environments are dynamic and heterogeneous in nature. In order to realize application-specific Quality of Service agreements within a grid, specifications at the level of an application are required. This chapter introduces an XML-based schema language (called the Application Specification Language, ASL) and a corresponding modeling tool that can be used to describe applications in grid computing environments. Such application descriptions allow derivation of guided and autonomic service developments for installation and invocation routines throughout the grid. In order to promote the language and ease the application description process, a domain-specific tool is also introduced. Based on our experience, the ASL in combination with higher level models improves, simplifies and promotes the grid application deployment process while simultaneously minimizing tedious and error-prone tasks such as manual application description composition.

INTRODUCTION

Grid computing (Foster, Kesselman, & Tuecke, 2001) has gained popularity as the emerging architecture for next-generation high performance distributed computing. Grid computing provides ubiquitous access to distributed high performance computing (HPC) resources that are shared between multiple organizations through virtualization and aggregation. This is realized through a layer of software (e.g., grid middleware), thus

Copyright © 2009, IGI Global, distributing in print or electronic forms without written permission of IGI Global is prohibited.

making grid applications extremely software-intensive systems where the value of software is equivalent to the value of the underlying infrastructure. Grid middleware provides a standard set of services for authentication, authorization, resource allocation and management, job scheduling, submission, monitoring, and data transfer and management (Berman, Hey, & Fox, 2003b). Software packages and tools based on open-source/open-standard approaches such as Globus Toolkit (Foster & Kesselman, 1997) have enabled the deployment of "production quality" computational grids. Several domain-specific grids are currently operational, for example, Grid Physics Network (GriPhyN) (GriPhyN, 2006), Network for Earthquake Engineering Simulation (NEES-Grid) (NEES, 2006), International Virtual Data Grid Laboratory (IVDGL) (IVDGL, 2006), Open Science Grid (Grid, 2007), and Particle Physics Data Grid Collaboratory Pilot (Grid2003) (PPDG, 2006). Grid computing has offered researchers enormous computing and data storage capabilities by providing seamless access to geographically distributed resources through the creation of virtual organizations (VOs).

Despite the many benefits of grid computing, the grid itself does not provide a novel programming paradigm for developing new applications. Furthermore, no formal methodology exists for porting existing legacy applications to the grid. Most of the applications developed for the grid are based on traditional HPC or distributed computing principles. Typical HPC applications are developed using implicit parallel programming techniques (e.g., compiler-based automatic parallelization and directive-based parallelization) or explicit parallel programming techniques (e.g., threads and message-passing). After an HPC application is developed and tested on local resources, it is then deployed; that is, the entire application (source code, dataset, scripts) is transferred to a remote site, compiled on a remote host, and made available for execution.

Deploying an application on the grid requires additional steps that involve user intervention, great insight into the internal structure of the application, as well as familiarity with the various grid computing technologies and toolkits. In addition, the progressive steps of application execution and job submission may involve many additional steps required from the end-user, not necessarily found in a typical application. Due to this inherent complexity and difficulty using the grid, several approaches have attempted to simplify grid deployment and configuration by developing technologies such as Web portals (Gannon et al., 2003), workflow systems (Aalst & Hee, 2002), and component assembly (Armstrong et al., 1999). The ultimate goal of such efforts is to enable the adoption of grid technologies and applications to a wider group of end-users who are not familiar with programming languages and the lower level grid infrastructure. The potential impact for improving grid accessibility to such users is significant (e.g., applied science researchers, distributed organizations, and organizations with variable computational requirements).

To tailor the complexities of a user's view of the vast amount of grid software by addressing the goals of end-user accessibility, there is a need to standardize and simplify the process of application deployment on the grid. As part of the contribution of this chapter, we present a new language called the Application Specification Language (ASL) (Afgan & Bangalore, 2007) that can be used by application developers and end-users to describe details of a given application. The ASL allows an application to be represented in the heterogeneous world of the grid by capturing its functionality, options and differences as compared with other applications found in the grid. Through the use of ASL, application descriptions can be made available for immediate use or further advancements among applications such as job schedulers, automated interface generators and application-specific on-demand help provisioning. The ASL

36 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/domain-specific-language-describing-grid/29396

Related Content

Text-Dependent and Text-Independent Writer Identification Approaches: Challenges and Future Directions
Rajandeep Kaur, Rajneesh Raniand Roop Pahuja (2022). *International Journal of Software Innovation (pp. 1-23).*
www.irma-international.org/article/text-dependent-and-text-independent-writer-identification-approaches/297514

Autonomic Business-Driven Dynamic Adaptation of Service-Oriented Systems and the WS-Policy4MASC Support for Such Adaptation
Vladimir Tosic (2012). *Theoretical and Analytical Service-Focused Systems Design and Development (pp. 140-156).*
www.irma-international.org/chapter/autonomic-business-driven-dynamic-adaptation/66797

Language Engineering for Mobile Software
Engineer Bainomugisha, Alfredo Cádiz, Pascal Costanza, Wolfgang De Meuter, Sebastián González, Kim Mens, Jorge Vallejosand Tom Van Cutsem (2012). *Handbook of Research on Mobile Software Engineering: Design, Implementation, and Emergent Applications (pp. 150-166).*
www.irma-international.org/chapter/language-engineering-mobile-software/66466

Katana: Towards Patching as a Runtime Part of the Compiler-Linker-Loader Toolchain
Sergey Bratus, James Oakley, Ashwin Ramaswamy, Sean W. Smithand Michael E. Locasto (2010). *International Journal of Secure Software Engineering (pp. 1-17).*
www.irma-international.org/article/katana-towards-patching-runtime-part/46149

Reliability Modeling and Assessment for Open Source Cloud Software: A Stochastic Approach
Yoshinobu Tamuraand Shigeru Yamada (2014). *Handbook of Research on Architectural Trends in Service-Driven Computing (pp. 718-742).*
www.irma-international.org/chapter/reliability-modeling-and-assessment-for-open-source-cloud-software/115451