Rudder: A Rule-based Multi-agent Infrastructure for Supporting Autonomic Grid Applications [*]

Zhen Li and Manish Parashar
The Applied Software Systems Laboratory
Rutgers University
Piscataway, NJ, 08854, USA
{zhljenny,parashar}@caip.rutgers.edu

1. Introduction

The emerging pervasive information and computational Grid is enabling a new generation of autonomic applications that continuously interact with their environments and with each other to monitor, adapt and optimize their execution. However, supporting these context-aware self-managing applications in widely distributed, heterogeneous and highly dynamic Grid environments requires suitable and effective coordination middleware services. This poster presents the architecture and operation of Rudder, a rule-based adaptive multi-agent infrastructure for supporting autonomic applications in pervasive Grid environments. Specifically, Rudder enables dynamic composition and coordination of autonomic components to manage changing application requirements and system context. The design builds on three basic concepts: (1) software agents that dynamically define, deploy and execute rules to achieve self-* behaviors, operating within flexible organizational structures while being situated in the dynamic pervasive Grid environments [1]; (2) composition rules dynamically enabling application workflows, which can be executed using autonomic components; (3) a decentralized reactive tuple-space can scalably support agent coordination, execution of composition rules and creation of application workflows in Grid environments. The operation of Rudder is illustrated using an autonomic oil reservoir optimization application.

2. Architecture

Rudder is a key component of project AutoMate [2] and provides the coordination infrastructure for Accord programming framework [3]. A conceptual overview of the architecture is shown in Figure 1. In this figure, a hierarchy of context-aware software agents coordinate with each other using the decentralized reactive tuple-space, and enable self-management and dynamic composition of autonomic components.

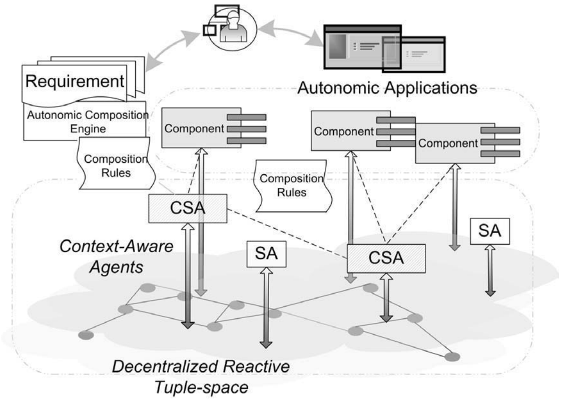

Figure 1. A conceptual overview of Rudder architecture.

Rudder agent framework consists of three types of context-aware peer agents: Component Agents (CA), System Agents (SA), and Composition Agents (CSA). CAs and SAs exist as system services while CSAs are transient and generated to satisfy specific application requirements. CAs dynamically configure and deploy autonomic components, manage their life cycles, and provide them with uniform access to middleware services (e.g., transaction, security, and persistence). SAs monitor, schedule and adaptively optimize physical resource utilization, such as CPU and disk. They are embedded within resource units (e.g., computer, cluster, data archive). CSAs define workflow-selection and component-selection rules based on application requirements. They use workflow-selection rules to choose a

[*] The research presented in this paper is supported in part by the National Science Foundation via grants numbers ACI 9984357 (CAREERS), EIA 0103674 (NGS), EIA-0120934 (ITR), ANI-0335244 (NRT), CNS-0305495 (NGS) and by DOE ASCI/ASAP (Caltech) via grant number 82-1052856.

workflow plan to enact, derive task-ordering rules and insert them as reactive tuples into the tuple-space. Using component-selection rules, CSAs discover registered autonomic components, and then select the appropriate ones.

Agents interact with each other using a decentralized reactive tuple-space, in which the space behaviors can be programmed by means of reactions to standard communication events. Agents communicate, interact, and control the behavior of the tuple-space by associatively reading, writing or removing tuples. The reactive tuple-space provides powerful abstractions, since agent coordination laws can be embedded in it as programs for coordinating agent interactions. It also provides mechanisms for dynamically defining, modifying, and forwarding rules to relevant peer agents.

3. Rudder Operation

The operation of Rudder is illustrated using an autonomic oil reservoir optimization application (AORO) [4] (shown in Figure 2). The goal of AORO is to maximize revenue from an oil field by optimizing the placement and configurations of oil wells. In this application, autonomic composition engine (ACE) [5] generates application workflows to satisfy the application objectives. The following workflows are dynamically defined: (1) the optimization service provides the IPARS reservoir simulator with an initial guess of well parameters based on the configuration of the oil field; (2) IPARS uses the well parameters along with current market parameters to periodically compute the current revenue using an Economic Model (EM) service; and (3) IPARS iteratively interacts with the optimization service to optimize well parameters for the maximum profit. Based on these workflows, three composition agents are instantiated for the EM, Optimizer, and IPARS respectively. The CSAs dynamically discover the appropriate autonomic components with desired functionality and cost/performance properties using AutoMate discovery service. The CSAs then coordinate with the CAs which are associated with these components via the decentralized tuple-space to accomplish oil reservoir optimization.

Autonomic optimization in the application is achieved through the agent autonomous behaviors. The agents adaptively configure and compose components, tune system parameters to achieve application objectives. Each CA monitors and manages the execution of its component, while the CSAs proactively search and select available components and resources to satisfy current application objectives. For example, the Optimizer CSA selects the most appropriate one from a suite of available optimization components (currently VFSA and SPSA) to optimize the application according to the current objective of the application. Similarly, the SAs monitor the runtime utilization of the resource and dynamically balance the workload.

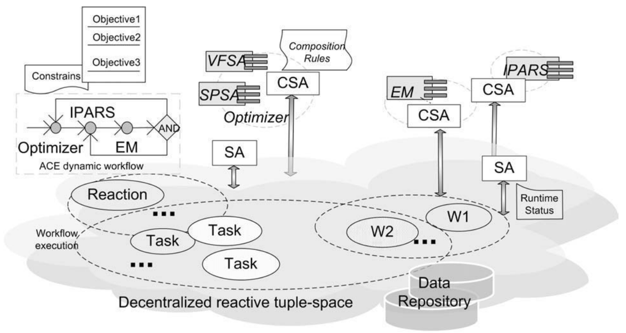

Figure 2. Autonomic composition and execution of AORO.

4. Conclusion

Rudder provides a rule-based multi-agent infrastructure to support autonomic applications in Grid environments. Distributed peer agents provide the mechanism to enable dynamic composition by defining and enabling workflows at runtime. A scalable decentralized reactive tuple-space infrastructure provides the coordination layer for agent interactions. Rudder is currently in a prototype stage and is used to enable autonomic oil reservoir optimization.

References

[1] N. Jennings, "Agent-Oriented Software Engineering," *Proceedings of the 9th European Workshop on Modeling Autonomous Agents in a Multi-Agent World: Multi-Agent System Engineering (MAAMAW-99)*, 1999.

[2] M. Agarwal, V. Bhat, Z. Li, H. Liu, V. Matossian,V. Putty, C. Schmidt, G. Zhang, M. Parashar, B. Khargharia, and S. Hariri., "AutoMate: Enabling Autonomic Applications on the Grid," *Proceedings of Autonomic Computing Workshop, 5th Annual International Active Middleware Services Workshop(AMS 2003)*, pp 365-375, Seattle, WA, June 25 2003.

[3] H. Liu and M. Parashar, "A Component Based Programming Framework for Autonomic Applications," *Proceedings of the 1st International Conference on Autonomic Computing (ICAC 2004)*, New York, NY, May 17-18 2004.

[4] V. Matossian and M. Parashar, "Autonomic Optimization of an Oil Reservoir using Decentralized Services," *Proceedings of the 1st International Workshop on Heterogeneous and Adaptive Computing– Challenges for Large Applications in Distributed Environments (CLADE 2003)*, Seattle, WA, USA, Computer Society Press, pp 2-9, June 2003.

[5] M.Agarwal and M. Parashar, "Enabling Autonomic Compositions in Grid Environments, *Proceedings of the 4th International Workshop on Grid Computing (Grid 2003)*, Phoenix, AZ, USA, IEEE Computer Society Press, pp 34 - 41, November 2003.