Self-Reconfiguration of Service-Based System for Service Level Agreements and Resource Optimization

Ying Li, IBM China Research Laboratory, China
Kewei Sun, IBM China Research Laboratory, China
Jie Qiu, IBM China Research Laboratory, China
Ying Chen, IBM China Research Laboratory, China

ABSTRACT

The configuration of a service-based system has a significant impact on the non-functional requirements of the system as a whole. However, finding the best configuration is very challenging and sometimes impossible for administrators because so many factors have to be considered. More importantly, a service-based system has to be frequently reconfigured to adapt to rapid and continuous changes in user requirements and runtime environments. In this paper we propose an autonomic computing approach to the problem of reconfiguration, that is, enabling the service-based system to configure itself by means of a loop of monitoring, analyzing, planning and executing. We begin by formalizing the definition of reconfiguration. Then, we describe how to implement the autonomic computing mechanisms for reconfiguring service-based systems to satisfy Service Level Agreements with minimal resource consumption. The approach is demonstrated on a resilient service provisioning environment. Finally, the preliminary experiments are evaluated to determine the effectiveness of proposed approach.

Keywords: autonomic computing; distributed systems; reconfiguration; service-based system

INTRODUCTION

Service Oriented Architecture (SOA) is considered as a promising approach to constructing large-scale enterprise information systems that span the Internet. In such service-based systems, a service can run on many machines and single machine can host many services. From the perspective of resource management, the execution of a service request results in the consumption of resources on the machine hosting the service. If the machine cannot provide adequate resources, all services on this machine are likely to execute with degraded performance and may even become un-
usable. Obviously, different services consume different types and quantities of resources (memory, CPU, I/O) while different machines provide varying types and quantities of resources. As a result, the configuration of a service-based system, defining the services that run on a given machine, may significantly impact the non-functional requirements (performance, reliability, cost) of individual services and the system as a whole. Yet despite the best efforts of the administrator, the configuration may not be good enough. More importantly, due to the open and dynamic nature of SOA, a configuration that is good enough may actually become worse due to changing factors that influence the metrics of non-functional requirements. This implies that the configuration of a service-based system should be continually optimized to cope with rapid and continuously changing factors such as user requirements or runtime environments.

The act of changing a configuration, called reconfiguration in this paper, is challenging. First, it is hard for administrators to monitor individual services or the whole system to determine whether the system requires reconfiguration. Second, it is very complex to determine an optimal configuration that satisfies all the desired requirements and that copes with the ever-changing factors. In the most general case the complexity of selecting the optimal configuration is $2^{mn}$, where $m$ is the number of services and $n$ is the number of machines. Third, the reconfiguration should be executed as soon as possible to limit the negative impact of a sub-optimal configuration. Fourth, there are few platforms or mechanisms that support the reconfiguration of a service-based system sufficiently and/or efficiently. Finally, the reconfiguration action, including monitoring, selecting, planning, and effecting the configuration change, has to be repeated again and again in spite of ever-changing factors, hence making the reconfiguration a nightmare for administrators.

We argue that executing the reconfiguration automatically is a promising way to address the challenges just mentioned. In this paper we propose an autonomic computing approach to the problem of reconfiguration, that is, to enabling a service-based system to continue adjusting its configuration by means of a loop that monitors, analyzes, plans, and executes re-configuration changes as work is serviced to satisfy the non-functional requirement coping with ever-changing factors — we call this approach self-reconfiguration in this paper. To begin, we formalize the definition of configuration and reconfiguration. Then, we describe how autonomic computing mechanisms may be implemented to perform self-reconfiguration for service-based systems to satisfy two common metrics of non-functional requirements: Service Level Agreements with minimal resource consumption. This approach has been demonstrated by our previous work that constructed a resilient service provisioning environment — Service Ecosystem (Li, 2004), a system that provides mechanisms to dynamically change the location of services on machines as they are executing service request.

The remainder of the paper is organized as follows. The second section gives an overview of autonomic computing concepts with a formulated self-reconfiguration definition. The third section presents the demonstration with the implemented automatic computing mechanisms for self-reconfiguration on Service Ecosystem. The fourth section evaluates the approach with experiment tests. The fifth section introduces the related work. The last section concludes this paper and identifies future work.

**APPROACH OVERVIEW**

According to Hofmeister (1993), for single system, there are three kinds of possible configuration changes: (1) component replacement, which means the component compositing the service may be changed; (2) structural change, which means the system’s logical structure (also called either the modular structure or the topology) may be changed; and (3) geometrical change, which means the logical appli-
Integration of Business Event and Rule Management With the Web Services Model
www.igi-global.com/article/integration-business-event-rule-management/3036?camid=4v1a