Autonomic Management of Internet Services: Experience with Self-Optimization

Sara Bouchenak
Université Joseph Fourier
Grenoble, France
Sara.Bouchenak@inria.fr

Noël De Palma
Institut National Polytechnique de Grenoble
Grenoble, France
Noel.Depalma@inria.fr

Daniel Hagimont
Institut National Polytechnique de Toulouse
Toulouse, France
Daniel.Hagimont@enseeiht.fr

Sacha Krakowiak
Université Joseph Fourier
Grenoble, France
Sacha.Krakowia@inria.fr

Christophe Taton
Institut National Polytechnique de Grenoble
Grenoble, France
Christophe.Taton@inria.fr

KEY WORDS: Autonomic management, Legacy systems, Self-optimization, Internet services, Multi-tier architecture, J2EE

1 Introduction

A common architectural organization for Internet applications is a multi-tier platform, in which each tier is a different legacy system that provides a specific function, e.g. a web tier which stores HTML pages, a business tier which executes the application business logic, and a database tier which stores non-ephemeral data. To ensure availability and high performance, these platforms are usually replicated on clusters of machines. The administration of such installations is an increasingly complex and error-prone task [3]. Autonomic computing, i.e. self-management in the face of evolving load conditions, hardware and software failures, and various forms of attacks, is an approach that aims at improving this situation [2]. Indeed, it provides a high-level support for deploying and (re) configuring applications, which reduces configuration errors and human administrators' efforts. Several challenges remain open in autonomic management, among which: (i) the management of legacy systems, (ii) the management of systems with complex architectures.

Legacy systems. By a legacy system, we mean a black-box with an immutable interface, on which no assumptions may be made, e.g. a web server software, or an email server software.

Complex architecture. The architecture of distributed systems, such as Internet services, is becoming more and more complex. With such an architecture, providing self-recovery upon a server failure for instance, does not simply imply restarting the failing server as an independent element, but also rebuiling the connections of that server with its front-end and back-end servers if any.

We have designed and implemented Jade, an environment that provides autonomic management capabilities for distributed applications. Jade provides a way to build control loops using an architectural framework. In this paper, we focus on the self-optimization aspect of autonomic management, through dynamic resource provising to face workload variations.

2 Design Principles and Architecture

Autonomic computing relies on the provision of control loops, in which an Autonomic Manager regulates parts of a system, called Managed Elements [2]. In order to be controllable, a Managed Element needs to be equipped with a management interface, which provides entry points for an Autonomic Manager. This interface should allow the manager to observe and to change the state of the element; the element may be an assembly of components, in which case the manager may modify the structure of this assembly.

We propose to use a software component model as a base for the design and implementation of Managed Elements. The component model that we use is Fractal [1], which has the following benefits: (i) it provides a uniform, adaptable control interface that allows introspection and dynamic binding of components; and (ii) it defines a hierarchical composition model for components, allowing a sub-component to be shared between enclosing components, at any level of granularity.

We use these properties to define a generic form of managed elements, providing a uniform management interface. This approach provides a management layer to a set of (legacy) hardware/software elements. An element, or set of elements, is wrapped in a Fractal component. This provides a means to:

- Managing legacy entities using a uniform model (the Fractal control interface), instead of relying on element-specific, hand-managed, configuration files.
- Managing complex environments with an architectural points of view.
- Adding a control behavior to the encapsulated legacy entities (e.g. monitoring, interception and reconfiguration).

In the management layer, all components provide the same (uniform) management interface for the encapsulated elements, and the corresponding implementation is specific to each element (e.g. in the case of J2EE, Apache web server, Tomcat Servlet server, MySQL database server, etc.). The interface allows managing the element's attributes, bindings and lifecycle.

Relying on this management layer, sophisticated administration programs can be implemented, without having to deal with complex, proprietary configuration interfaces, which are hidden in the wrappers. The management layer provides all the facilities required to implement such administration programs:

Introspection. The framework provides an introspection interface that allows observing managed elements (ME). For instance, an administration program can inspect an Apache web server ME (encapsulating the Apache server) to discover that this server runs on node1:port 80 and is bound to a Tomcat Servlet server running on node2:port 66. It can also inspect the overall J2EE infrastructure, considered as a single ME, to discover that it is composed of two Apache servers interconnected with two Tomcat servers connected to the same MySQL database server.

Reconfiguration. The framework provides a reconfiguration interface that allows control over the component architecture. In particular, this control interface allows changing component attributes or bindings between components. These configuration changes are reflected onto the legacy layer. For instance, an administration program can add or remove an Apache replica in the J2EE infrastructure to adapt to workload variations.

3 Conclusion

We have designed and implemented Jade, an environment for autonomic management of legacy systems. The main contribution of this paper is the definition of a set of architectural frameworks for constructing flexible and efficient autonomic management facilities for such systems. To prove the validity of our approach, we have applied these frameworks to the self-optimization of J2EE applications in the face of the wide load variations observed in Internet applications

Jade consists of two main frameworks: a framework for administrable components, which provides the administered system's components with a uniform management interface; and a framework for autonomic managers, which allows regulating a set of managed components for a specific management aspect.

The construction of these frameworks relies on the Component-Based Management approach (CBM), using a simple and generic architectural model to manage any hardware or legacy software infrastructure. The underlying component model provides the introspection and dynamic control facilities needed for the provision of a management interface to administrable elements. Autonomic managers use the component abstraction to reconfigure the underlying infrastructure and can discover the overall system infrastructure.

References

[1] E. Bruneton, T. Coupaye, and J. B. Stefani. Recursive and Dynamic Software Composition with Sharing. In *International Workshop on Component-Oriented Programming (WCOP-02)*, Malaga, Spain, June 2002. http://fractal.objectweb.org.

[2] J. O. Kephart and D. M. Chess. The Vision of Autonomic Computing. *IEEE Computer Magazine*, 36(1), 2003.

[3] K. Nagaraja, F. Oliveira, R. Bianchini, R. P. Martin, and T. D. Nguyen. Understanding and Dealing with Operator Mistakes in Internet Services. In *6th Symposium on Operating System Design and Implementation (OSDI-2004)*, San Francisco, CA, Dec. 2004.