A RESTful Approach to the Management of Cloud Infrastructure

Hyuck Han*, Shingyu Kim*, Hyungsoo Jung*, Heon Y. Yeom*,
Changho Yoon**, Jongwon Park**, Yongwoo Lee**
School of Computer Science and Engineering
Seoul National University
{hhyuck, sgkim, jhs, yeom}@dcslab.snu.ac.kr*
School of Electrical and Computer Engineering
The University of Seoul
{touch011, comics77, ywlee}@uos.ac.kr**

Abstract

Recently, REpresentational State Transfer (REST) has been proposed as an alternative architecture for Web services. In the era of Cloud and Web 2.0, many complex Web service-based systems such as e-Business and e-Government applications have adopted REST. Unfortunately, the REST approach has been applied to few cases in management systems, especially for a management system for cloud computing infrastructures. In this paper, we design and implement a RESTful Cloud Management System (CMS). Managed elements can be modelled as resources in REST and operations in existing systems can be evaluated using four methods of REST or a combination of them. We also show how components of existing management systems can be realized as REST-style Web services.

1 Introduction

After the concept of REpresentational State Transfer (REST) was introduced in 2000 [10], many toolkits, such as Restlet [15] and Rest in Python (RIP) [13], have been developed, and many research projects [19, 22, 14] and commercial services [1, 11, 12] have broadened its target areas. Outcomes of research projects and related symposia have attracted the attention of many researchers and developers, and REST has been established as a promising software development methodology for many web-service applications. However, the RESTful approach has been applied to few cases in management systems for cloud infrastructures despite active research to apply REST to many different areas.

In this paper, we present a challenging case study to apply REST to management systems, and show that a RESTful approach can be useful for building or integrating management systems. REST can enhance existing management systems since resources in REST can model managed elements, such as computing/network/storage resources, and four methods in REST can replace full operations of management systems. Based on the fact, we design and implement the prototype version of the RESTful Cloud Management System (CMS). The main contributions of this paper follow:

- We show that RESTful Web services can replace software components of existing management systems.
- We then propose a CMS architecture detail and implement it using REST guidelines.

2 Related Work

REST and Its Application.
Roy Fielding introduced REST as an architectural style for distributed hypermedia systems, describing the software engineering principles and the interaction constraints in his doctoral dissertation [10]. REST details how resources in Internet are defined and addressed. However, since REST is usually used to describe simple interfaces over Hypertext Transfer Protocol (HTTP) without an additional messaging layer such as Simple Object Access Protocol (SOAP), the term REST is used in conjunction with HTTP. The fundamental principles of REST follow.

- Each resource (such as a record or an item) is referenced using a uniform resource identifier (URI).

978-0-7695-3840-2/09 $26.00 © 2009 IEEE
DOI 10.1109/CLOUD.2009.68

- Resources are manipulated by only four major HTTP methods (*GET*, *PUT*, *DELETE* and *POST*)[1].

- Each interaction with a resource is stateless.

As explained in [17], REST and SOAP have advantages and disadvantages. SOAP has the advantages of protocol transparency and independence. Web Service Description Language (WSDL) allows service interfaces to be described in an abstract manner, and WSDL tools [9, 2] help developers build applications easily without considering the complexity of the underlying protocol or the software stack. However, SOAP and similar protocols such as XML-RPC and XML over HTTP can be misused when existing software components are turned into Web services [20]. Moreover, since SOAP headers were given more prominence after the first version, SOAP became more complex[2].

REST is thought to be simple and lightweight because RESTful Web services use well-known W3C/IETF standards. HTTP clients and servers are easily integrated and deployed in major platforms. Moreover, RESTful Web services are known to scale very well because of the web caching effect. However, many firewalls accept only two verbs (*GET* and *POST*) of the four major HTTP methods. Also, URIs larger than 4 KB cannot be encoded in the *GET* method.

REST is widely used because of its advantages. McFaddin *et al.* [19] proposed RESTful data service for mobile environments. Völkel [22] proposed a RESTful wiki architecture. Mazurek *et al.* [14] proposed distributed digital libraries using REST. The Amazon Simple Storage Service (Amazon S3), [1] which is an outstanding storage service, provides a REST application programming interface (API) to its users. To the best of our knowledge, the REST approach has been applied to few cases of management systems of cloud infrastructures. Thus, in this work we build such a management system using RESTful Web services.

Management Systems. There is much research that focuses on management systems such as NMS and SMS. A management system is composed of three components, namely, a GUI, manager and managed elements (agents). Typically, a manager uses a centralized database or a federation of databases to manipulate management information.

[1]*POST* **C**reats a new resource, *GET* **R**etrieves the current state of a resource, *PUT* **U**pdates the status of a resource and *DELETE* **D**eletes a resource. The four methods are abbreviated as **CRUD**.

[2]Some bloggers say that "Simple Object Access Protocol" (SOAP) is not "simple" any more.

Common Object Request Broker Architecture (CORBA) [16], SOAP, XML-RPC, Enterprise JavaBeans [21], HTTP and user-defined protocols can be used as protocols between a user interface (or external systems) and a manager, while SNMP [5, 18], CMIP [24], XML over HTTP [6], XML-RPC [3] and CIM over HTTP [7, 8] can be used as protocols between a manager and managed elements. Even if many management systems adopt standard specifications, sharing information among management systems is difficult because there are a variety of management information models and communication protocols in manager applications.

We believe that replacing both protocols with REST would mitigate the difficulty of sharing information. Moreover, REST also allows management systems to be easily decentralized because management information can modelled as a resource, which is identified by a URI. Thus, it is very important to investigate whether the simple interfaces of REST can replace the two protocols. We therefore design and implement a management system using RESTful Web services based on our analysis.

3 CMS Architecture

In this section, we present CMS architectures. As illustrated in Figure 1, CMS is composed of a GUI (or external systems), a REST-based manager and a REST-based agent (in the managed element side).

3.1 Design of the REST-based Manager

The basic components of manager systems are the *Frontend Interface*, *Asynchronous Event Handler*, *Management Client* and *Management Module*. The *Frontend Interface* is used to provide users or external systems with a management user interface. The *Frontend Interface* receives requests from user interfaces, passes them to the *Management Module*, and returns responses.

The *Asynchronous Event Handler* receives notifications from managed elements, stores notifications in databases and sends a meaningful notification to the *Management Module*. The *Management Client* collects information from managed elements periodically and stores information in databases. It also sends a control message to managed elements.

The *Management Module* manages the configuration parameters for managed elements, handles logical or physical topologies of managed elements, obtains monitoring information from databases or the *Management*

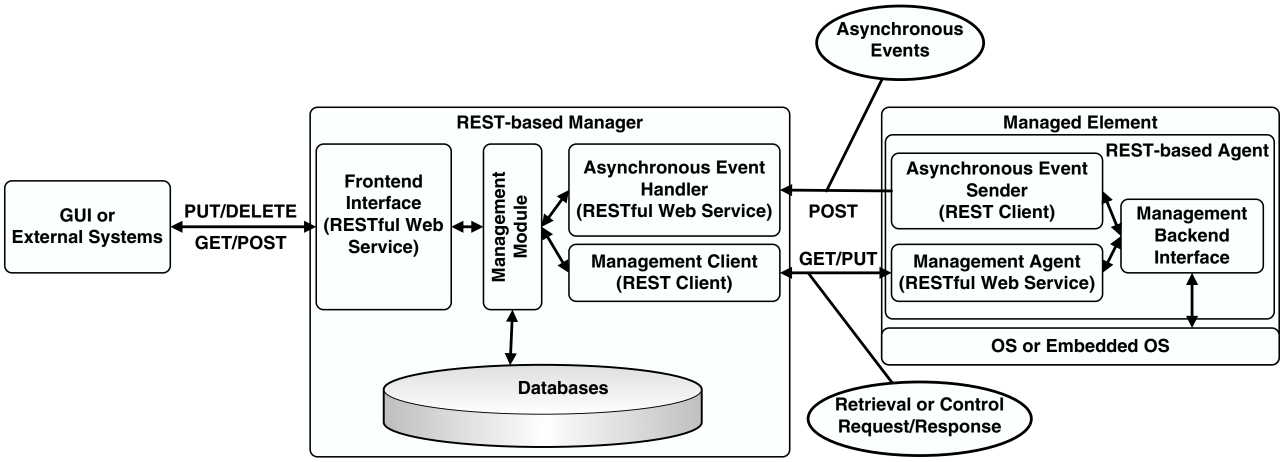
Asynchronous Events
REST-based Manager
Managed Element
REST-based Agent
GUI or External Systems
PUT/DELETE
GET/POST
Frontend Interface (RESTful Web Service)
Management Module
Asynchronous Event Handler (RESTful Web Service)
Management Client (REST Client)
POST
GET/PUT
Asynchronous Event Sender (REST Client)
Management Agent (RESTful Web Service)
Management Backend Interface
OS or Embedded OS
Databases
Retrieval or Control Request/Response

Figure 1. CMS Architecture

Client, logs the necessary data and manages notification data from the *Asynchronous Event Handler*.

3.2 Design of the REST-based Agent

The basic components of agent systems are the *Management Agent*, *Asynchronous Event Sender* and *Management Backend Interface*. The *Management Backend Interface* is used to provide the *Management Agent* and *Asynchronous Event Sender* with a management interface. If the *Management Backend Interface* detects an asynchronous event, such as *High System Temperature* or *CPU Overload*, it sends the event to the *Asynchronous Event Sender*. It also returns appropriate responses to the *Management Agent* according to requests of the *Management Agent*, such as retrieving the status of a specified network interface or applying a new configuration to a *access control list (ACL)*.

The *Asynchronous Event Sender* receives an event from the *Management Backend Interface* and sends a notification to the *Asynchronous Event Handler*. The *Management Agent* receives control commands or information requests from the *Management Client* and sends responses to the *Management Client* through the REST interface.

3.3 Management Information Model

Every piece of information in a RESTful Web service is represented by a URI, as explained in Section 2. Thus, not only information about each managed element, such as CPU and memory information, but also user-created information, such as group information or topological information, should be described as appropriate URIs.

Most management information models (MIMs) are conceptually organized as trees. Internal nodes of the tree represent subdivision by organization or function. Each variable value is stored in the corresponding leaf of the tree. Thus, every distinct variable value corresponds to a unique path from the root of the tree. The children of a node are numbered sequentially from left to right, starting at 1, so that every node in the tree has a unique name.

Each node in the tree for the MIM of an agent represents management information. The management information of an agent is divided into three parts: common information, standard management information and private management information. The common information consists of the device configuration, the administrator list, the contact information for the managed element, and so on. Standard management information consists of information that is defined in standard specifications. If a new standard information model is adopted by agents, we convert the structure of the information model into a tree and translate every piece of management information of the tree-structured model to a unique URI string.

For example, if the *Management Client* wants to retrieve the administrator list and the system description of "$cloud01.snu.ac.kr$", it sends HTTP requests, the target URIs are "http://cloud01.snu.ac.kr/deviceInfo/ci/AdminList" and "http://cloud01.snu.ac.kr/deviceInfo/smi/mib/iso/org/dod/internet/mgmt/mib/system/sysDesc" via the HTTP *GET* operation.

Many resources are required for cloud computing and they can be divided into several groups. Each group can be divided into several subgroups, and the topology structure of the cloud infrastructure appears as a hierarchy. We convert the hierarchy of groups into a tree strcutre. Then, we map each node of the tree to a corresponding URI. For example, the URI string of "$WebServer\#1$" is "$Group\#1/Subgroup\#1/WebServer\#1$". Thus, if the hostname of the manager is "*cms.snu.ac.kr*" and a user wants to see the administrator list of "$WebServer\#1$" using the user interface, the user interface sends HTTP requests of which the target URI is "http://cms.snu.ac.kr/Group#1/Subgroup#1/WebServer#1/deviceInfo/ci/AdminList".

4 Conclusion

In this paper, we propose the RESTful Cloud Management System (CMS). Our CMS fully utilizes fundamental Web technologies, such as HTTP and URIs, to perform infrastructure management. We also developed a REST-based manager and agent to achieve a pure and simple REST-based management system. We verified the functionality of the CMS by applying it to general computing resources for cloud computing. Our future work is to improve the scalability and extensibility of the CMS using simple web caching techniques [23, 4]. We will then validate both properties using very large computing infrastructures.

Acknowledgement

This study was supported by the Seoul R&BD Program (10561), Seoul, Korea. The ICT at Seoul National University provides research facilities for this study.

References

[1] Amazon. *S3 service*. https://s3.amazonaws.com/.

[2] Apache. *Axis2*. http://ws.apache.org/axis2.

[3] Avaya Labs. *XML based Management Interface for SNMP Enabled Devices*. http://www.research.avayalabs.com.

[4] G. Barish and K. Obraczke. World wide web caching: trends and techniques. *Communications Magazine, IEEE*, 2000.

[5] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A Simple Network Management Protocol (SNMP). *RFC 1098*, 1990.

[6] M.-J. Choi, J. W. Hong, and H.-T. Ju. XML-Based Network Management for IP Networks. *ETRI Journal*, 2003.

[7] DMTF. Common Information Model (CIM).

[8] DMTF. Specification for CIM Operations over HTTP Version 1.0.

[9] Eclipse. *Eclipse Homepage*. http://www.eclipse.org/.

[10] R. T. Fielding. Architectural Styles and the Design of Network-based Software Architectures. *doctoral dissertation*, 2000.

[11] flickr. *Flickr Service API*. http://www.flickr.com/services/api/.

[12] Google. *Google Search API*. http://code.google.com/apis/ajaxsearch/documentation.

[13] Luke Arno. *REST in Python Homepage*. http://lukearno.com/projects/rip.

[14] C. Mazurek, T. Parkola, and M. Werla. Building federation of digital libraries basing on concept of atomic services. In *JCDL '08: Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, 2008.

[15] Noelios Technologies. *Restlet*. http://www.restlet.org.

[16] OMG. *Common Object Request Broker Architecture (CORBA)*. http://www.corba.org.

[17] C. Pautasso, O. Zimmermann, and F. Leymann. Restful web services vs. "big"' web services: making the right architectural decision. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, 2008.

[18] R. Presuhn, J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. Management Information Base (MIB) for the Simple Network Management Protocol (SNMP). *RFC 1907*, 2002.

[19] S. McFaddin et al. Modeling and Managing Mobile Commerce Spaces Using RESTful Data Services. In *Proceedings of the Ninth International Conference on Mobile Data Management*, 2008.

[20] R. Sessions. Fuzzy boundaries: Objects, components, and Web services. *ACM Queue*, 2004.

[21] Sun. *Enterprise JavaBeans Technology*. java.sun.com/products/ejb.

[22] M. Völkel. Semwiki: a restful distributed wiki architecture. In *WikiSym '06: Proceedings of the 2006 international symposium on Wikis*, 2006.

[23] J. Wang. A survey of web caching schemes for the internet. *SIGCOMM Comput. Commun. Rev.*, 1999.

[24] U. Warrier, L. Besaw, L. LaBarre, and B. Handspicker. The Common Management Information Services and Protocols for the Internet (CMOT and CMIP). *RFC 1189*, 1990.