A RESTful Approach for a Cloud Gateway

Chang Ho Yun, Jong Won Park, Hae Sun Jung, Yong Woo LEE
School of Electrical & Computer Engineering
The Ubiquitous-City (Smart City) Consortium, the University of Seoul
Seoul, South Korea
{touch011, comics77, banyasun, ywlee}@uos.ac.kr

Haeng Jin Jang
Korea Institute of Science and Technology Information
Seoul, South Korea
hjjang@kisti.re.kr

***Abstract*— Polar research is active nowadays since it gives us many kinds of information about global climate change so that we can respond to it more properly. We found that the research can have much benefit by using a data farm approach, which gives high performance computing power without limit. Here, we are interested in providing more convenient and useful interface to use the high performance computing power in the polar research. This paper presents a cloud gateway, that is, a science research gateway which supports cloud and grid computing in a unique REST architecture. It provides facilities and interfaces which enable polar researchers to do computer supported remote collaborative work as well as to use data farms.**

Keywords-Cloud Computing; Grid Computig; REST; Web Service; Science Gateway; Polar Research.

I. INTRODUCTION

Recently, the change of global climates has emerged as a global agenda [1]. It has attracted much attention so far and would do so more and more in future. Korea has been doing polar research to cope with the problem as well as many other useful issues which are global issues nowadays. Cloud computing and Grid computing can significantly help the polar research experts. We develop a scientific research gateway, called as Cloud Gateway, which enables the polar research experts to use the technologies with easiness and efficiency.

The Cloud Gateway also provides a collaboration environment for the various kind of polar research. Polar researchers can do computer supported cooperative work and share data among interesting research groups beyond geographical gaps and regardless of different working times. The management of polar metadata can be easy and efficient with it.

The Cloud Gateway consists of three tiers - Infrastructure Tier, Processing Tier and Presentation Tier - to support the distributed environment. It uses RESTful Web services [2] for the data transmission and service request between each tier. Also, it collects, processes and provides many kinds of information such as Portable Batch System (PBS) accounting information, information of file system, CPU information and slave node information to users.

The Cloud Gateway was designed to meet the following two requirements. Firstly, it should support geographically scattered multiple computing facilities such as clusters, web servers, databases, etc. through integrated service. Secondly, the service should be provided in a user-transparent way. That is, it should enable polar researchers to use the computing resources without pushing them to know any detailed knowledge of the underlying technologies of the Cloud Gateway.

The Cloud Gateway also has the following three distinctive factors. Firstly, it has three-tier architectures as explained before. Secondly, it uses REST technologies so that users can access geographically scattered multiple computing facilities through a single interface as explained before. Thirdly, the web portal is used as the user access point to the Cloud Gateway in order to meet the user transparency requirement.

The outline of the paper is organized as follows: Section 2 investigates related works. Section 3 outlines the design of the Cloud Gateway. Section 4 explains how it was implemented. Finally, Section 5 gives conclusions and our plan to the future works.

II. RELATED WORKDS

Currently existing science gateways usually provide high end resources to a community of users, scientists, and engineers through web-based graphical interface [3]. A common approach in the previous generation was to adopt the JSR 168 portlet component model and WSDL/SOAP style web services.

The TeraGrid User Portal serves as a launch pad for new users and a control panel for current users by integrating TeraGrid Resource Provider, services, and information into a single web interface serving a national community of computational researchers [4][5].

The Linked Environments for Atmospheric Discovery (LEAD) Portal is a science application portal which was designed to enable effective use of Grid resources in exploring mesoscale meteorological phenomena [6].

WLCG provides graduate and accurate verification of performance of hardware resources such as CPU, storage, and network. It also provides the middleware services for

 ISBN: 978-1-61208-271-4

Grid projects and the LHC experiment-specific software applications [7].

PolarGrid portal added current social networking techniques to a typical science gateway model to enable a scientific collaboration [8]. It uses a RESTful Web-service and Web 2.0 technologies. However, it just uses them for user interface, not for managing computing resources.

World Wide Web was usually chosen as preferred infrastructure. Thus, most initiatives adopted Web technologies such as CORBA (Common Object Request Broker Architecture) [9], OLE/DCOM [10], SOAP (Simple Object Access Protocol), etc. Especially, SOAP is the de facto standard in current science gateways.

REST is widely used because of its simplicity and lightweight [11]. McFaddin et. al. [12] and Christensen [13] proposed RESTful service for mobile environments. Volkel [14] proposed RESTful wiki architecture. Twitter [15], Flickr [16], Amazon Simple Storage Service (Amazon S3) [17], Amazon Elastic Compute Cloud (Amazon EC2) [18], and others provide a REST application programming interface (API) to their users. However, the RESTful approach has been seldom applied to the management of science gateways. Contrastingly, our Cloud Gateway provides RESTful Web services to manage it.

III. ARCHITECTURE

Our Cloud Gateway uses the three tier Architecture and RESTful Web service technologies in order to support a distributed computing environment. There, technologies of Java platform was used in order to make the Cloud Gateway be independent of computing platform.

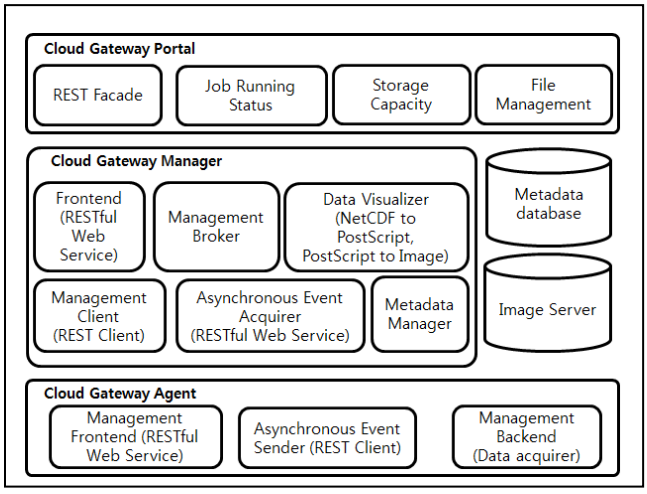

Figure 1. The architecture of the Cloud Gateway.

Figure 1 shows the architecture of the Cloud Gateway. The tier 1 of the Cloud Gateway is the Cloud Gateway Agent. The Cloud Gateway Agent is installed on PBS Clusters. It's role is to communicates with the Cloud Gateway Manager. It collects information of CPU, file system, accounting, etc. in the PBS system and passes an asynchronous message to the Cloud Gateway Manager when accounting information is updated. The tier 2 of the Cloud Gateway is the Cloud Gateway Manager. It does logical processing. It determines whether the data exist in the metadata database or not. If the data does not exist, then it collects the data from the Cloud Gateway Agent when the Cloud Gateway Portal requests information of accounting or file system. The Cloud Gateway Portal shows information of accounting and system resources in the PBS clusters through graph or table. It provides interface to manage the file system of each node. Because the Cloud Gateway uses 3 tier architecture, users can easily manage the scatted resources in cloud computing environments and/or grid computing environments. Figure 2 shows the operational concept of our three-tier architecture that can manage multiple clusters.

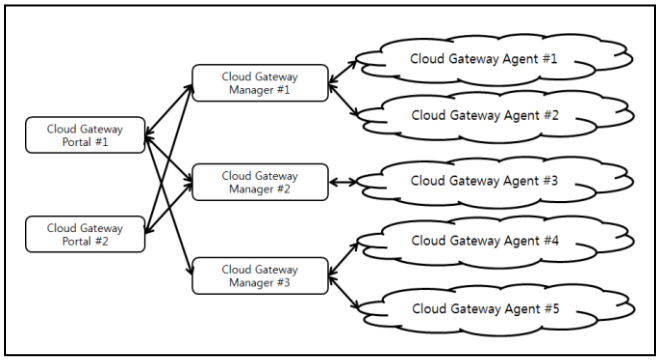

Figure 2. The multiple cluster management of the Cloud Gateway.

A. *Cloud Gateway Agent*

The Cloud Gateway Agent gives RESTful web service and is installed on the master node of the PBS cluster system. The components of the Cloud Gateway Agent are the Management Backend, the Management Agent and the Asynchronous Event Sender.

The Management Backend provides a management interface to the Management Agent and the Asynchronous Event Sender. It returns proper responses such as the CPU information, the file system information, the accounting information and the detail information of slave nodes of the PBS system to the Management Agent according to requests from the Management Agent. The Asynchronous Event Sender receives an event from the Management Backend and sends a notification of the event to the Asynchronous Event Acquirer of the Cloud Gateway Manager. And, it also sends the request message when the Cloud Gateway Agent is added to the Cloud Gateway Manager in a first time. The Management Frontend returns the response according to requests of the Management Client of the Cloud Gateway Manager through RESTful web service interface. The resource managed by the Cloud Gateway Agent can be accessed through HTTP methods such as GET, POST, PUT, and DELETE.

Figure 3 shows RESTful web service interface of the Cloud Gateway Agent. It consists of the ci (Common Information) and the pi (PBS Information).

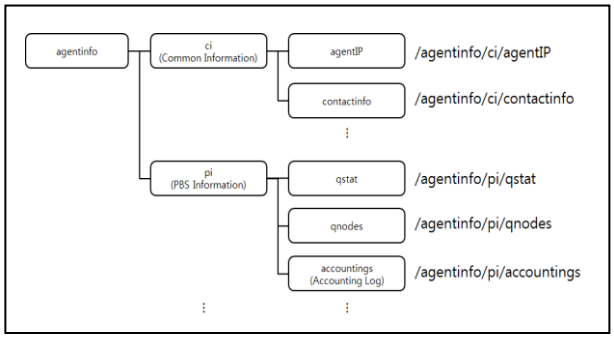

Figure 3. RESTful web service interface of the Cloud Gateway Agent.

B. *Cloud Gateway Manager*

The components of the Cloud Gateway Manager are the Frontend, the Management Broker, the Management Client, The Asynchronous Event Acquirer, the Metadata Manager, the Data Visualizer, the Metadata Database and the Image Server.

The Frontend provides a RESTful web service-based interface for the Cloud Gateway Portal. It receives requests from the Cloud Gateway Portal, passes them to the Management Broker, and returns responses. Figure 4 shows the RESTful web service interface of the Frontend. It can also be accessed through HTTP methods such as GET, POST, PUT, and DELETE.

The Management Broker is accessible by the Frontend and the Asynchronous Event Acquirer and provides them with the response according to the request from them. It manages the agents, obtains monitoring information from the Metadata Manager and the Management Client. The Management Broker cannot access other tiers and the Meta database and uses the Management Client to access other tiers. The Metadata Manager accesses the Meta database.

The Management Client and the Asynchronous Event Acquirer communicate with the Cloud Gateway Agent. They do not do any logical behaviors and pass the event to the Management Broker. The Management Client requests the necessary data such as the status of PBS slave nodes to the Cloud Gateway Agent which uses them to perform the service of the Management Broker.

The Asynchronous Event Acquirer receives the asynchronous event from the Asynchronous Event Sender of the Cloud Gateway Agent. The Metadata Manager can only access the Metadata Database. The Data Visualizer processes images for collaboration among polar researchers. These images are stored in the Image server. In request of the Cloud Gateway Portal, they are provided through the Frontend.

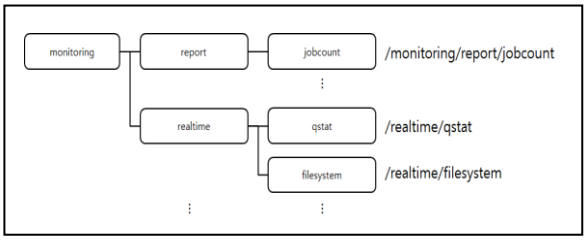

Figure 4. REST web service interface of the Cloud Gateway Manager.

C. *Cloud Gateway Portal*

The Cloud Gateway Portal is a user transparent web portal. Its components are the REST Facade, the Job Running Status, the Storage Capacity and the File Management. The Cloud Gateway Portal uses Springframework and Ajax/Javascript.

The REST Façade manages the requests and the responses from the Cloud Gateway Portal to the Cloud Gateway Manager. The Job Running Status requests accounting information of PBS cluster using the REST Façade, translates the response to contents such as the chart and the table and shows them. The Storage Capacity component requests the storage and the file system information of PBS cluster using the REST Façade, translates the response to contents such as the chart and the table and shows them. The File Management requests the information of the file systems in PBS cluster using the REST Façade, help download and upload and modify the files.

IV. IMPLEMENTATION

We implemented the Cloud Gateway on a Linux system using Java language. However, the Cloud Gateway Manager and the Cloud Gateway Portal can be installed on any operating system with Java and Tomcat. MySQL was used as the Metadata database. We used the Restlet package [19] to build the RESTful Web services. Jnotify package [20] was used to monitor PBS accounting as a tool [21] for the Cloud Gateway Agent.

The operations of the Cloud Gateway can be one of the following three types. First, the response result for request from the Cloud Gateway Portal exists in the Metadata Database of the Cloud Gateway Manager. Second, it does not exist in the Cloud Gateway Manager, so the Cloud Gateway Manager queries the request to the Cloud Gateway Agent. Third, the Cloud Gateway Agent sends the notification of changing the status of the PBS cluster to the Cloud Gateway Manager.

 ISBN: 978-1-61208-271-4

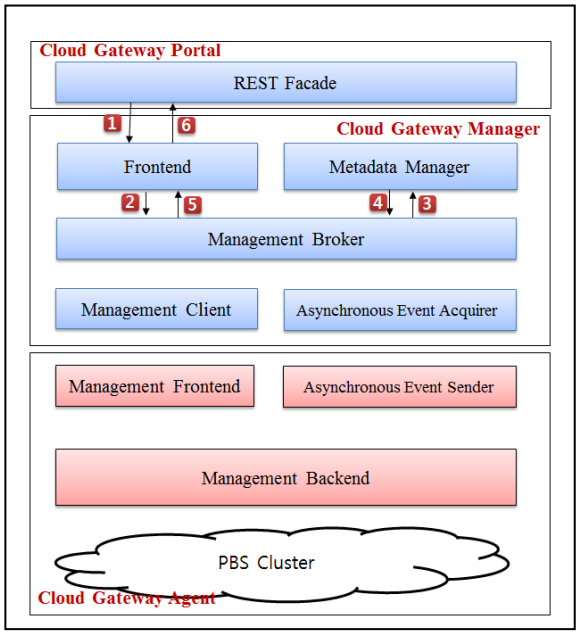

Figure 5. The processing of Job accounting.

An example of the first type is shown in Figure 5. When the user requests job accounting information, the REST Façade of Cloud Gateway Portal requests URL of "/monitoring/report/jobcount" to the RESTful web service interface of the Frontend of the Cloud Gateway Manager. Then, the Frontend sends the message to the Management Broker and the Management Broker analyses the request. Because the Management Broker can find response result in the Metadata Database, the Management Broker collects accounting information by using the Metadata Manager and returns the result to the REST Façade through Frontend.

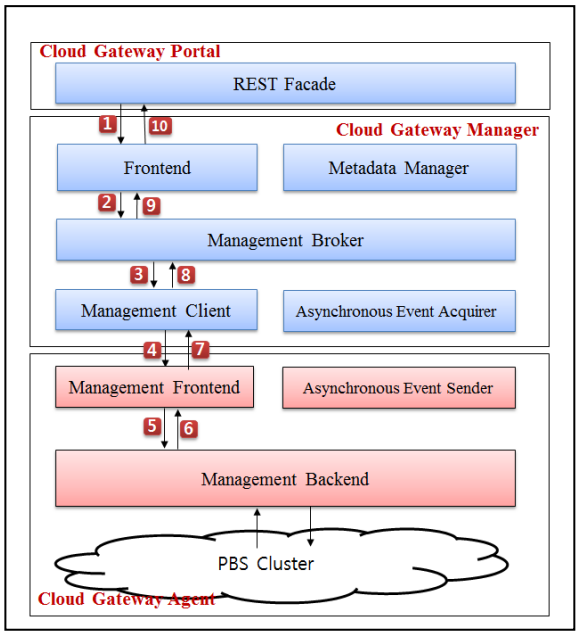

Figure 6. The processing of qnodes request.

An example of the second type is shown in Figure 6. When the user requests the information of slave nodes in PBS cluster, the REST Façade requests the URL of "/realtime/qnodes" to the RESTful web service of the Frontend of the Cloud Gateway Manager. The Frontend sends the request to the Management Broker and the Management Broker analyses it. Because the data are not in the metadata database, the Management Broker requests the URL of "/agnetinfo/pi/qnodes" to the Management Frontend of the Cloud Gateway Agent through the Management Client. The Management Frontend sends the received request to the Management Backend and the Management Backend queries the request to PBS cluster. The result of the query is returned to the Management Client through the Management Frontend. The Management Client sends it to the Management Broker. The Management Broker returns the result to the REST Façade of the Cloud Gateway Portal.

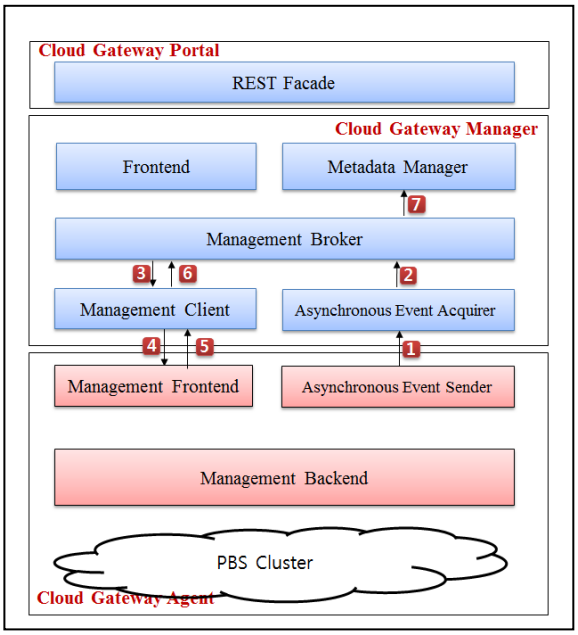

Figure 7. The processing of asynchronous event.

An example of the third type is the asynchronous event in the Management Backend. The Management Backend of the Cloud Gateway Agent monitors the accounting logs of PBS cluster. If the logs are found to be changed, then the Management Backend sends the asynchronous event to the Asynchronous Event Acquirer through the Asynchronous Sender. The Asynchronous Event Acquirer sends the event to the Management Broker. The Management Broker analyses it and checks the need to update the Metadata database. If the Metadata Database is needed to be updated, then the Management Broker requests the data to the Management Frontend of the Cloud Gateway Agent through the Management Client. The Management Frontend returns the result that is acquired from the Management Backend. Then the Management Client sends the result to the Management Broker. Now the Management Broker updates

 ISBN: 978-1-61208-271-4

the Metadata Database using them through the Metadata Management.

Figure 8, 9, and 10 show the snapshot of job running status, the snapshot of job status, the snapshot of file management respectively.

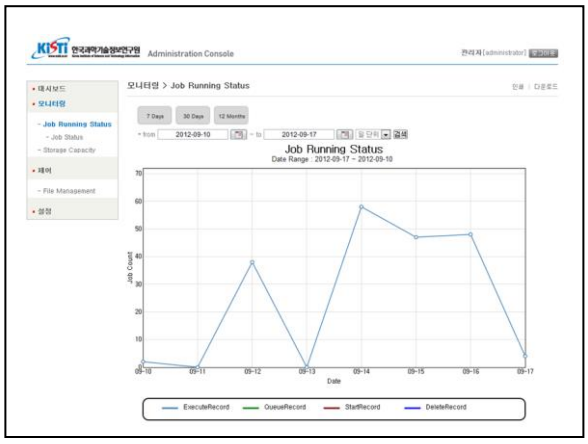

Figure 8. The snapshot of job running status.

Figure 9. The snapshot of job status.

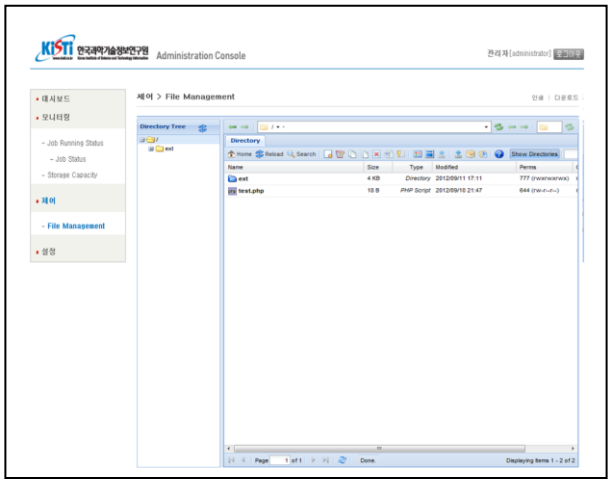

Figure 10. The snapshot of file management.

V. CONCLUSION

This paper proposed our cloud gateway model and its RESTful approach in order to support cloud computing, Grid computing, computer supported collaboration, etc. for the polar research. The Cloud Gateway uses the three tier architecture to provide the RESTful web service. Therefore, users can access geographically scattered multiple computing facilities such as clusters, web servers and databases through a single interface easily, efficiently and user-transparently. The future works are planned to add analysis tools for geospatial query components and visualization components.

ACKNOWLEDGMENT

This study was supported by the Ministry of Education, Science and Technology in Korea and KISTI (Korea Institute of Science and Technology Information). This study was also supported by the Seoul Research and Business Development Program, Smart (Ubiquitous) City Consortium (10561) and Seoul Grid Center. This work was also supported by the 2011 research fund of the University of Seoul (Yong Woo LEE : the corresponding author).

REFERENCES

[1] M. Kok, W. Vermeulen, A. Faaij, and D. Jager, Global Warming and Social Innovation: The Challenge of a Climate Neutral Society, Earthscan Publications Ltd, 2002.

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

[3] N. Wilkins-Diehr, D. Gannon, G. Klimeck, S. Oster, and S. Pamidighantam, "TeraGrid Science Gateways and Their Impact on Science," IEEE Computer, vol. 41, Nov. 2008, pp. 32-41, doi: 10.1109/MC.2008.470.

[4] M. Dahan, E. Roberts, and J. Boisseau, "TeraGrid User Portal v1.0: Architecture, Design, and Technologies," Proc. International Workshop on Grid Computing Environments, Nov. 2006.

[5] J. Basney, V. Welch, and N. Wilkins-Diehr, "TeraGrid Science Gateway AAAA Model: implementation and lessons learned," Proc. The 2010 TeraGrid Conference, Aug. 2010, pp. 1-6, doi: 10.1145/1838574.1838576.

[6] M. Christie and S. Marru, "The LEAD Portal: a TeraGrid gateway and application service architecture," Concurrency and Computation: Practice and Experience, vol. 19, Apr. 2007, pp. 767-781, doi: 10.1002/cpe.1084.

[7] D. Bonacorsi and T. Ferrari, "WLCG Service Challenges and Tiered architecture in the LHC era," IFAE 2006, pp. 365-368, doi:10.1007/978-88-470-0530-3_68.

[8] Z. (G.) Guo, R. Singh, and M. Pierce, "Building the PolarGrid Portal Using Web 2.0 and OpenSocial," Proc. The fifth Grid Computing Environments Workshop, 2009, article no.5, doi: 10.1145/1658260.1658267.

[9] Object Management Group, Inc, Cobra [retrieved: Mar. 2013], http://www.corba.org/

[10] Microsoft, DCOM [retrieved: Mar. 2013], http://www.microsoft .com/COM/

[11] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. "big" web services: making the right architectural decision," Proc. The 17th international conference on World Wide Web (WWW 08), 2008, pp. 805-814, doi: 10.1145/1367497.1367606.

 ISBN: 978-1-61208-271-4

[12] S. McFaddin, D. Coffman, J. H. Han, H. K. Jang, J. H. Kim, J. K. Lee, M. C. Lee, Y. S. Moon, C. Narayanaswami, Y. S. Paik, J. W. Park, and D. Soroker, "Modeling and Managing Mobile Commerce Spaces Using RESTful Data Services," Proc. The Ninth International Conference on Mobile Data Management (MDM 08), Apr. 2008, pp. 81-89, doi: 10.1109/MDM.2008.38.

[13] J. H. Christensen, "Using RESTful web-services and cloud computing to create next generation mobile applications," Proc. The 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications (OOPSLA 09), 2009, pp. 627-634, doi:10.1145/1639950.1639958.

[14] M. Volkel, "Semwiki: a restful distributed wiki architecture," Proc. The 2006 international symposium on Wikis (WikiSym 06), 2006, pp. 141-142, doi:10.1145/1149453.1149486.

[15] Twitter, Twitter REST API [retrieved: Mar. 2013], https://dev.twitter.com/docs/api

[16] Yahoo, flickr REST API [retrieved: Mar. 2013], http://www.flickr.com/services/api/response.rest.html

[17] Amazon, Amazon Simple Storage Service REST API [retrieved: Mar. 2013], http://docs.amazonwebservices.com/AmazonS3/latest/API/APIRhest.html

[18] Amazon, Amazon Elastic Compute Cloud REST API [retrieved: Mar. 2013], http://docs.amazonwebservices.com/AWSEC2/latest/APIReference/Welcome.html

[19] Noelios Technologies, Restlet [retrieved: Mar. 2013], http://www.restlet.org.

[20] O. Yadan, jnotify [retrieved: Mar. 2013], http://jnotify.sourceforge.net/

[21] R. Mach, PBS XML Accounting Toolkit [retrieved: Mar. 2013], http://pbsaccounting.sourceforge.net/

 ISBN: 978-1-61208-271-4