2014 IEEE 13th International Symposium on Network Computing and Applications

Resource Management in Cloud Federation using XMPP

Maria Fazio, Antonio Celesti, Massimo Villari, Antonio Puliafito
DICIEAMA Department
University of Messina
Messina, Italy
mfazio{acelesti, mvillari, apuliafito}@unime.it

***Abstract*—This paper deals with Cloud federation issues, where Clouds are both providers and clients of virtual resource at the same time. Specifically, we have designed a solution based on a XMPP communication platform, which allows to set up federated environments and to easily manage virtual shared resources. Heterogeneous and dynamic Clouds can interact in near-real time, making available their own resources according to specific agreement policies. We are thinking to implement the proposed solution using the Hadoop framework, a well-known Cloud Map-Reduce middleware designed to offer different types of services. Here, we analyze the basic and useful elements necessary to making up federated clouds. By mean of Cloud federation, two or more Hadoop clusters can be connected in order to increase service scalability, overcoming some technical limitations of the Hadoop framework itself.**

***Keywords*-Cloud computing, federation, XMPP, Hadoop, Big Data, Cloud services.**

I. Introduction

The main aim of Cloud federation is to pursue new levels of efficiency in delivering Cloud services to customers. Let us imagine a scenario where different Clouds belonging to independent administrative domains interact each other for increasing their capabilities. They become themselves both clients and resources providers at the same time. The federation we are focusing on hereby obeys to the rule in which systems evolve according to an "Horizontal Federation", where small, medium, and large providers federate to gain economies of scale, optimize the usage of their assets, enlarge their capabilities and share resources to establish new forms of collaboration.

In this paper, we describe a new solution to support resource management in federated Clouds. To achieve high interoperability among different Cloud infrastructures, it is necessary to establish an efficient and near-real time communication platform, in order to integrate heterogeneous systems and technologies. To achieve this goal, we have exploited the well-known Extensible Messaging and Presence Protocol (XMPP) [1], which allows to fulfill Cloud federation requirements and to provide a distributed communication environment able to federate many Cloud providers.

Our solution can be employed in several different scenarios, where Cloud providers offer different types of virtual resources and services. To show the effective applicability of our federation platform, we focus the attention on storage provisioning application field. Indeed, Federation brings benefits in many scenarios. Among them, Cloud storage resource provisioning represents a very interesting topic, because it deals with the increasing storage demand of end users and applications.

Thus, we discuss how Clouds can improve the quality of service in storage provisioning by using federation. Specifically, we describe the deployment of the Cloud service for large files storage (e.g., audio and video)using the Hadoop [2] framework, and in particular Hadoop Distributed File System (HDFS).

Hadoop is widely used for storing huge amount of unstructured data through a fully distributed approach. It implements a highly fault-tolerant file system designed to be deployed on low-cost hardware, provides high throughput to application data and is suitable for applications that manage large data sets. However, it is unsuitable to face the new emerging challenges of data mining and Big Data management, since it does not allow link-based data models, such as linked-lists, trees, and graphs.

In our solution, each Cloud provider holds its own HDFS cluster to store client files. The proposed federated storage service leads several benefits, since it:

- exposes storage resources as a service, according to the NIST Cloud paradigm [3];
- federates several storage providers, offering an elastic and seamless service to end users;
- overcomes scalability and technical limitations of storage platforms.

The paper is organized as follows. In Section II, we provide a brief overview of current works on Cloud federation and scalable storage. In Section III, we introduce the technologies and tool exploited in our work to establish a real federated environment exploiting XMPP. Then, Section IV presents our solution for resource management in federated Clouds. Section V summarizes our conclusions and future work.

II. Related Work

Cloud federation refers to mesh of Clouds that are interconnected by using open standards, in order to provide a universal decentralized computing environment. Federation is raising many challenges in different research fields on

978-1-4799-5393-6/14 $31.00 © 2014 IEEE
DOI 10.1109/NCA.2014.15

Cloud computing as discussed in [4][5][6]. One of the most recent FP7 European Project focused on Cloud federation is RESERVOIR [7]. RESERVOIR introduces an abstraction layer that allows to develop a set of high level management components that are not tied to any specific environment. In RESERVOIR, several sites can share physical infrastructure resources on which service applications can be executed. A RESERVOIR Cloud federation is homogeneous (i.e., each site has to run the same middleware) and decentralized. It occurs at the IaaS level and needs a static a-priori configuration in each site. The CLEVER architecture presented hereby was born as outcome of the RESERVOIR project, and tries to overcome some issues in Cloud federated scenarios.

The authors in [8] propose a decentralized technique using a structured peer-to-peer network supporting discovery, deployment and output data collection of a (PaaS) middleware (Aneka). The system is organized as a set of Aneka coordinator peers deployed in several Clouds, offering discovery and coordination mechanisms for federation. In our previous work [9] we describe an architectural solution for federation by means of the Cross-Cloud Federation Manager (CCFM), a software component in charge of executing three main functionalities: i) discovery, which allows to exchange information on federated Clouds, ii) match-making, which performs the best choice of the provider that can loan its resources, and iii) authentication, to create a secure communication channel among federated Clouds. In [10], the authors propose a quite complex model for federate systems. It is composed of three entities: i) Cloud Coordinator, which manages a specific Cloud and acts as interface for the external Clouds by exposing well-defined Cloud interfaces; ii) Cloud Exchange Component, which stores all the necessary information on federated Cloud providers together with their demands and offers for computational resources; iii) Cloud Broker, which interacts with the Cloud Exchange Component to find most suitable Cloud provider, and with the Cloud Coordinator to perform resource allocation according to specific QoS requirements.

The solution we propose in this paper differs a lot from the solutions in literature, because it is based on a fully distributed approach, where there are not additional components responsible for federation management.

The Hadoop framework transparently provides both reliability and data transfer. Some solutions in literature aim to improve specific features of the Hadoop framework. The Apache Hadoop on Demand (HOD) [11] provides virtual Hadoop clusters over a large physical cluster. It uses the Torque [12] resource manager to perform node allocation. Other solutions try to improve the functionalities of Hadoop in federated environments. In [13], the authors propose the Optimis Data Manager and Distributed File System (ODFS) solution based on HDFS. This paper shows how to interact with HDFS using RESTful interfaces and SSHfs for external interactions. Respect to our approach, HDFS instances are located into different data centers and expose common interfaces, which are used to accomplish real federation. Another interesting work that tries to overcome the issue of scalability in large federated datacenter is described in [14]. The idea behind this work is to implement a Hadoop Filesystem Agnostic API (HFAA), which aim to integrate Hadoop with any distributed file system over TCP sockets.

III. Cloud Federation and XMPP

Cloud federation implies the establishment of a trust context between several Cloud platforms acting within different administrative domains and located at different places. The main advantage of setting up a federation is that Clouds can set inter-domain communications to benefit of new business opportunities, such as the enlargement of their virtual resources capability. The concept of federation we use in this work entails that if a Cloud provider needs of additional resources, it can ask to federated Clouds to loan their own. This approach aims to dynamically resize the set of available virtual resources and improve performance of each Cloud, without affecting their business management. The system presented hereby can be used for managing two independent Hadoop frameworks deployed on different administration domains.

In a federated environment, specific mechanisms for dynamic identification and service discovery have to be employed. We believe that the best way to accomplish the federation features is in using strategic communication technologies, able to interconnect many different distributed entities and to provide an integrated platform. To this aim, we make use of the XMPP protocol, because it natively supports federation capabilities. In the next sections we present the main features of XMPP and we explain how it can be employed to carry out the management of resources in federation.

XMPP is an XML-based protocol used for near-real time and extensible instant messaging, presence information and contacts list maintenance. It remains the core protocol of the Jabber Instant Messaging and Presence technology. The "Jabber" technology leverages open standards to provide a highly scalable architecture that supports the aggregation of presence information across different devices, users and applications. XMPP applications beyond instant messaging include network management, content syndication, collaboration tools, file sharing, gaming, remote systems monitoring and even applications in Cloud scenarios.

The XMPP network uses a client-server architecture, but a strong decentralization is achieved because anyone can run his own XMPP server without the existence of a central master system. The XMPP specification supports both the Simple Authentication and Security Layer (SASL) Mechanism and the Transport Layer Security (TLS) technologies for the authentication and encryption of communication channels. Moreover, customized features can be built on top

of XMPP. To maintain interoperability, common extensions are managed by the XMPP Software Foundation.

Each user/application on the network is identified by a *Jabber ID* (JID). The utilization of a central server which maintains a list of IDs can be avoided by structuring each JID as e-mail addresses, with *username* and *domain* names. The former identifies a single entity, the latter refers to the domain the entity belongs to; the two fields are separated by @, such as `username@exampledomain.com`.

The XMPP server acts as an intelligent abstraction layer for XMPP communications. Its primary responsibilities are: 1) to manage connections among entities, in the form of XML streams; 2) to route appropriately-addressed XML stanzas among such entities over XML streams. An XML stanza is the basic unit of meaning in XMPP. An entity may publish a presence state to indicate its current communication status and to inform others about its willingness to communicate.

Each communication room is identified by a *room JID* ⟨roomname@service⟩ (e.g., `<jdev@conference.jabber.org>`), where *roomname* is the name of the room (or *room ID*) and *service* is the name of the host where the multi-user chat service is running. A user enters a room by sending a presence stanza. Then, it can exchange stanzas with other users in the same room. Messages sent within Multi User Chat (MUC) rooms are addressed to the room itself (`roomname@service`) and then forwarded to all the users.

An advantage of XMPP is that the server can listen on the HTTP port to establish a connection with its clients. This feature is very useful in inter-domain environment interconnected with more distributed systems. Typically, systems are protected from external malicious attacks by a firewall that blocks the unwanted traffic on several ports. However, usually the HTTP port is always opened and traffic can flow through. These features make XMPP strategic for dynamic interactions among Cloud systems, where entities join/leave the inter-domain environment, because it is not required to change the configuration of the firewall.

IV. Federation setting up

Cloud entities establish a federated communication environment by creating a MUC used to exchange information and directives for resources management. Due to the presence feature of XMPP, the federated environment can be established on fly and it can change during the time. Entities can join and leave the MUC room without causing neither disruption in resource management nor side effects in service provisioning.

Each entity involved in the federation has a unique JID and can join the MUC. Let us consider three hosts belonging to three different administrations, e.g. three different Cloud providers, identified by their respective JID. Also, the XMPP server necessary to deploy the communication system (`jabber@domD`) is hosted in a fourth domain. The entity `HostA@DomA` needs to enter the federation and, hence, it has to register to the MUC. To this aim, it sends a request for registration to the XMPP server `jabber@domD`, which verifies its credentials and, if the registration succeeds, authorizes the requesting entity to access the MUC. Moreover, in the reply stanza it sends the list of entities already logged in. Then, `HostA@DomA` gets through to connected entities specifying its *username* and the *domain* it belongs to. When an entity joins the federation, it can exploit the MUC to send messages to other components.

XMPP has been designed to offer high interoperability and extensibility. Using the power of XML, we can build customized functionalities on top of the core protocols. Thus, we have designed new types of XMPP messages, which hold command invocations in the body, in order to execute commands in remote hosts on different domains. Whenever an entity needs resource loan, it sends a stanza with the tag `TYPE=REQUEST`. Although XMPP communications are asynchronous in nature, each entity sending a REQUEST message waits for a REPLY to know if the command has been executed, aborted or generated exceptions.

Let us assume an example in which given the `HostA`, that is a node in the domain `DomA`trying to invoke a command into the `HostB` in the domain `DomB`. Specifically, `HostA`may run two threads. The `ThreadA` executes a specific task according to the design of the `Agent` running in the host. It sends a stanza over the XMPP channel with the request for remote command invocation. The stanza also includes an ID (`ID1` in the example) to identify the request of the host. To maintain asynchronous communications among hosts, information on each request generated by `HostA` are stored into a hashtable structure. When the reply message for request `ID1` will be received at the `HostA`, `ThreadE` will manage it. Each host runs a thread to listen messages in the MUC. Thus, the request sent by `ThreadA` is picked up by `ThreadB` in `HostB` through the `FederationListenerAgent`. As soon as `ThreadB` realizes that the received message is a request for `HostB`, creates a new thread, that is `ThreadC`, to mange the message. The `FederationRequestExecutor` component in `ThreadC` pushes the command wrapped in the message to the `Agent` responsible for local resource management. Then, the thread waits for results on the command execution, in order to arrange a reply message for `HostA`, thus specifying if the command execution generated exceptions or not. Information on the incoming request and the outgoing reply are stored in a local hashtable structure. `ThreadD` is charged with the task of sending the reply message over the MUC. The behavior of `ThreadE` is very similar to the `ThreadB` one. It listens stanzas in the MUC and picks up messages for `HostA`. When it realizes that the

received message is a reply message, it uses information in the hashtable structure to know the `Agent`, and hence the thread engaged in the communication and forwards the body of the message to it.

V. Conclusion and future works

In this paper, we have presented a new solution to actualize federation among Cloud providers. The future idea its to exploit Hadoop, a Cloud Map-Reduce middleware that might make extensively use of the XMPP technology. XMPP connections, in order to benefit of scalable, near-real time and asynchronous communications among federated components. The proposed approach guarantees high lever of elasticity and dynamism to the Cloud domains, therefore possible misbehaviors in a domain do not affect the service reliability in the other ones.

Federation brings benefits in many scenarios. Among them, we have focused the attention on the computation services. The proposed service uses XMPP MUC to achieve storage resources in federated domains due the increasing interest in Big Data.

Acknowledgment

This work was partially supported by Projects SIMONE and SIGMA, Italian National Operative Program (PON) 2007-2013.

References

[1] P. Saint-Andre, "Extensible messaging and presence protocol (xmpp): Core," Internet RFC 3920, October 2004.

[2] HADOOP, 2012, The Hadoop Distributed File System: Architecture and Design, http://hadoop.apache.org/docs/r0.16.4/hdfs_design.html.

[3] NIST, Cloud Computing Reference Architecture http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505 December 2011.

[4] F. Tusa, A. Celesti, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation," in *Proceedings of IEEE CLOUD '10*. IEEE, July 2010, pp. 337–345.

[5] S. Kiani, A. Anjum, N. Bessis, and R. Hill, "Large-scale context provisioning: A use-case for homogenous cloud federation," in *Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on*, july 2012, pp. 241–248.

[6] I. Goiri, J. Guitart, and J. Torres, "Characterizing cloud federation for enhancing providers' profit," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, july 2010, pp. 123 –130.

[7] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, E. Levy, A. Maraschini, P. Massonet, H. Munoz, and G. Toffetti, "Reservoir - when one cloud is not enough," *Computer*, vol. 44, pp. 44–51, 2011.

[8] R. Buyya and R. Ranjan, "Special section: Federated resource management in grid and cloud computing systems," *Future Generation Comp. Syst.*, vol. 26, no. 8, pp. 1189–1191, 2010.

[9] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "Three-phase cross-cloud federation model: The cloud sso authentication," in *Proceedings of the 2010 Second International Conference on Advances in Future Internet*, ser. AFIN '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 94–101.

[10] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010*. Springer, 2010, pp. 21–23.

[11] L. Wang, J. Tao, R. Ranjan, H. Marten, A. Streit, J. Chen, and D. Chen, "G-hadoop: Mapreduce across distributed data centers for data-intensive computing," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 739–750, March 2013.

[12] Torque. http://www.adaptivecomputing.com/products/torque.php year=2013.

[13] G. Kousiouris, G. Vafiadis, and T. Varvarigou, "A front-end, hadoop-based data management service for efficient federated clouds," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, Nov 2011, pp. 511–516.

[14] A. Yee and J. Shafer, "Hfaa: A generic socket api for hadoop file systems," in *Proceedings of the 2Nd Workshop on Architectures and Systems for Big Data*, ser. ASBD '12. New York, NY, USA: ACM, 2012, pp. 15–20.

[15] "The open source, open standards cloud, innovative, open source cloud computing software for building reliable cloud infrastructure. http://openstack.org/ jan 2014." [Online]. Available: http://openstack.org/

[16] OpenStack Inter Cloud Resource Federation. https://wiki.openstack.org/wiki/InterCloudResource-Federation, 2014.