COMET: Distributed Metadata Service for Multi-cloud Experiments

Komal Thareja, Cong Wang, Paul Ruth, Anirban Mandal, Ilya Baldin, Michael Stealey
Renaissance Computing Institute (RENCI)
{kthare10, cwang, pruth, anirban, ibaldin, stealey}@renci.org

***Abstract*—A majority of today's cloud services are independently operated by individual cloud service providers. In this approach, the locations of cloud resources are strictly constrained by the distribution of cloud service providers' sites. As the popularity and scale of cloud services increase, we believe this traditional paradigm is about to change toward further federated services, a.k.a., multi-cloud, due to the improved performance, reduced cost of compute, storage and network resources, as well as increased user demands. In this paper, we present COMET, a lightweight, distributed storage system for managing metadata on large scale, federated cloud infrastructure providers, end users, and their applications (e.g. HTCondor Cluster or Hadoop Cluster). We showcase use case from NSF's, Chameleon, ExoGENI and JetStream research cloud testbeds to show the effectiveness of COMET design and deployment.**

I. Introduction

The emergence of cloud computing has substantially affected how scientists and engineers think about Internet architecture. Cloud computing provides a shared pool of re-configurable computing resources, such as computing servers, storage, application engines, or network links that can be deployed and destroyed based on user needs.

This provides great flexibility for a wide spectrum of users to rapidly obtain resources from a few "Internet-scale" providers like Google, Microsoft and Amazon, whose resources are concentrated in a relatively small number of data centers strategically located around the globe. However, today's cloud application developers can only enjoy the conveniences from these cloud providers when they deploy applications on a specific cloud. There are significant difficulties for developers to develop, automate and manage distributed applications spanning multiple clouds. We believe this traditional, individually managed cloud architectural pendulum is about to change towards the direction of open, unified services. This change motivates researchers to rethink today's centralized network-cloud architecture as a widely distributed multi-provider, a.k.a.,multi-cloud, marketplace that allows end-users to choose providers based on cost, distance, jurisdiction, capabilities, and specific tasks they need to perform within their application stacks. This paper focuses on metadata management services for a multi-cloud architecture. Metadata services store information about compute, networking, or storage resources provisioned for the tenants, such as host names, public keys, list of IP and MAC addresses, instance status, network VLAN tags, etc.

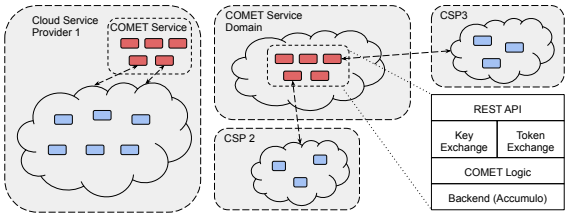

Figure 1. COMET Metadata service architecture

Many public and private cloud implementations include metadata services, such as AWS metadata service [1], OpenStack metadata service [2] or AWS SSM [3], which are essential for bootstrapping the tenant software stacks. A major drawback is that, the current metadata services are constrained to a specific cloud provider domain or tied to a specific provider authorization infrastructure. To launch applications (e.g. HTCondor or Hadoop Cluster) across multiple cloud domains, the tenants must be able to post and retrieve metadata from resources deployed across multiple cloud providers in a secure way.

In this paper, we propose a lightweight metadata management framework—COMET [4]—to support distributed, key-value oriented metadata for both the cloud infrastructures and user applications. We also demonstrate the use cases of COMET. The COMET Service [5] is currently hosted at RENCI and accessible from Headnode (`comet-hn1.exogeni.net`) and used by the DyNamo Mobius [6] platform and ExoGENI users.

II. Design

COMET is designed to provide a global, universal metadata service that stores data from any node that was created by one or more cloud orchestration services. Fig. 1 depicts the COMET structure for two cases: COMET can be deployed inside a single cloud service (Cloud Service Provider 1); it can also be deployed on a third-party service domain to coordinate multiple cloud services. COMET is RESTful and stateless, which makes it scale up well.

III. Data Security

The COMET entry structure is depicted in Table I. COMET extends Accumulo by storing a serialized object as Accumulo's value entry. The object contains the `WriteToken`, which allows the client to perform write or delete operations

Table I
COMET TABLE SCHEMA

Key				
Row ID	Column			Time Stamp
	Family	Qualifier	Read Token	
Value				
ifDeleted	Write Token	Context Value	COMET Version	Delete TS

only if the correct token is presented. `ifDeleted` keeps track of deleted entries: when a metadata entry is deleted, COMET flags the data as "deleted" and keep records of deletion timestamps for debugging purposes.

COMET provides two level data security. First, it verifies a client's SSL certificate to ensure that the data entries are created only by the applications or users presenting a certificate that is signed by a Trusted Authority. Second, it uses a combination of read and write tokens to ensure each data entry is managed by the right user.

IV. Why COMET?

Without a global meta data service like COMET, automatic deployment and configuration of applications like Hadoop cluster or HT Condor cluster across multiple clouds requires either manual intervention or complex scripting. We describe the problem with the example of HT Condor Cluster(used by DyNamo [6]) configuration below.

HT Condor cluster has 3 kinds of nodes - master, submit and worker nodes. A user requires the following information to configure a condor cluster:

- Master node's hostname - must be shared with submit and worker nodes
- Exchange public keys for root user (assuming condor is running as root) between all nodes
- Specify the network subnet in condor config file on each node
- Every node in the cluster must know hostname and IP address of rest of the nodes in the cluster
- Add or Delete the newly added/deleted nodes in `/etc/hosts` file and authorized keys

COMET provides an easy, secure and extensible mechanism to automate deployment and configuration of applications like HT Condor Cluster across multiple clouds.

V. Experimental Use Case

COMET is used by the Mobius [7] platform in DyNamo [6] as depicted in Fig. 2 to spawn a HTCondor cluster across ExoGENI, Chameleon Cloud and JetStream testbeds. All the resources within a cluster are identified as workflow. A user or experimenter could add or delete resources dynamically to a workflow. Mobius has a Python based provisioning client, which provisions resources on various clouds for a workflow. It also creates a context in COMET protected by read and write tokens via a COMET client [8]. It creates two entries shown below within the context for each Compute Node. Read and Write tokens are passed to each Compute Node. All compute nodes share same read token but have individual write tokens.

- "family": "hostsall", "key": "hostname", "value": ""
- "family": "pubkeysall", "key": "hostname", "value": ""

Each of the Compute Node has a HostKey Daemon [9] running on it. The HostKey Daemon does the following using the COMET Client:

- Pushes nodes's IP address and Public Key to the COMET in hostsall and pubkeysall family respectively
- Periodically pulls all the entries from hostsall family and updates nodes's `/etc/hosts` to reflect the current cluster topology
- Periodically pulls all the entries from pubkeysall family and updates nodes's authorized keys

COMET makes the exchange of host names, IP addresses and public keys seamless and enables easy automation of the HT Condor Cluster configuration.

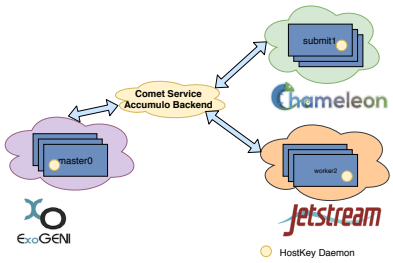

Figure 2. Hostname and public key sharing across CI via COMET.

VI. Conclusions

In this paper, we discussed the COMET metadata management service that focuses on security and flexibility for multi-cloud applications. We discussed the design, implementation and experimental use of such a service by DyNamo to automate configuration of applications running across CIs.

References

[1] AWS Metadata Service, https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html.

[2] OpenStack Metadata Service, https://docs.openstack.org/nova/latest/user/metadata-service.html.

[3] AWS SSM Service, https://docs.aws.amazon.com/systems-manager/latest/userguide/ssm-agent.html.

[4] C. Wang, K. Thareja, M. Stealey, P. Ruth, and I. Baldin, "Comet: A distributed metadata service for federated cloud infrastructures," in *High Performance Extreme Computing Conference (HPEC)*, Sep 2019, to appear.

[5] COMET github repository, https://github.com/RENCI-NRIG/COMET-Accumulo.

[6] E. Lyons, G. Papadimitriou, C. Wang, K. Thareja, P. Ruth, J. Villalobos, I. Rodero, E. Deelman, M. Zink, and A. Mandal, "Toward a dynamic network-centric distributed cloud platform for scientific workflows: A case study for adaptive weather sensing," in *15th IEEE eScience Conference*, 2019, to appear.

[7] Mobius github repository, https://github.com/RENCI-NRIG/Mobius.

[8] COMET Client github repository, https://github.com/RENCI-NRIG/COMET-Client.

[9] Host-Key Daemon github repository, https://github.com/RENCI-NRIG/host-key-tools.