Edge Clouds Control Plane and Management Data Consistency Challenges: Position Paper for IEEE International Conference on Cloud Engineering, 2019

Bohdan Dobrelia
OpenStack platform
Red Hat
Poznan, Poland
bdobreli@redhat.com

***Abstract*—Fog computing is emerging Cloud of (Edge) Clouds technology. Its control plane and deployments data synchronization is a major challenge. Autonomy requirements expect even the most distant edge sites always manageable, available for monitoring and alerting, scaling up/down, upgrading and applying security fixes. Whenever temporary disconnected sites are managed locally or centrally, some changes and data need to be eventually synchronized back to the central site(s) with having its merge-conflicts resolved for the central data hub(s). While some data needs to be pushed from the central site(s) to the Edge, which might require resolving data collisions at the remote sites as well. In this paper, we position the outstanding data synchronization problems for OpenStack platform becoming a cloud solution number one for fog computing. We define the inter-cloud operational invariants based on that Always Available autonomy requirement. We show that a causally consistent data storage is the best match for the outlined operational invariants and there is a great opportunity for designing such a solution for Edge clouds. Finally, the paper brings the vision of unified tooling to solve the data synchronization problems the same way for infrastructure owners, IaaS cloud operators and tenants running workloads for PaaS, like OpenShift or Kubernetes deployed on top of Edge clouds.**

***Index Terms*—Open source software, Edge computing, Distributed computing, System availability, Design**

I. Introduction

OpenStack is an Infrastructure-as-a-Service platform number one for private cloud computing, and it becomes being so for fog computing as well. Hybridization and Mutli-cloud trends for private clouds interconnected with public clouds and Platform-as-a-Service (PaaS) solutions, like OpenShift/Kubernetes, allow the containerization of micro-services oriented workloads to emerge in a highly portable, self-contained and the hosting cloud-agnostic way. Giving it massively distributed scale of fog computing and bringing the data it operates closer to end users, opens great opportunities for Internet of Things (IoT) and nextgen global telecommunication technologies, which first of all requires low-latency and higlhy responsive interfaces always available for end users.

Speaking of always available, back to the system administration realities, the Edge clouds control and management plane capabilities in such a perfect world shall not fall behind as well. This paper is about to position the associated data replication challenges and to bring vision of future development trends on that topic, both for OpenStack and hopefully for anything residing on top of it, e.g. PaaS and/or workloads designed for massively distributed scale and following the IoT/fog computing best practices.

II. Glossary

Aside of the established terms [3], we define a few more for the data processing and operational aspects:

A. Deployment Data

Data that represents the configuration of *cloudlets* [3], like API endpoints URI, or numbers of deployed *edge nodes* [3] in *edge clouds* [3]. That data represents the most recent state of a deployment.

B. Cloud Data

Represents the most recent[1] internal and publicly visible state of cloudlets, like cloud users or virtual routers. Cloud data also includes logs, performance and usage statistics, state of message queues and the contents of databases.

C. Control Plane

Corresponds to any operations performed via cloudlets API endpoints or command line tooling. For example, starting a virtual machine instance, or creating a cloud user. Such operations are typically initiated by cloud applications, tenants or operators.

D. Management Plane

Corresponds to administrative actions performed via configuration and lifecycle management systems. Such operations are typically targeted for cloudlets, like edge nodes, *edge data centers* [3], or edge clouds. E.g. upgrading or reconfiguring

[1]when there is unresolved data merging conflicts, the most recent state becomes the best known state

cloudlets in a *virtual data center* [3], or scaling up edge nodes. And typically initiated by cloud infrastructure owners. For some cases, like Baremetal-as-a-Service, tenants may as well initiate actions executed via the management plane. Collecting logs, performance and usage statistics for monitoring and alerting systems also represents the management plane operations, although it operates with the cloud data.

E. Always Available

The operational mode of the control and management planes that corresponds to the *sticky available* [4] causal consistency models, i.e. *Real-Time Causal* [2], or *causal+* [1].

The stickiness property imposes an additional constraint [4]: "on every non-faulty node, so long as clients only talk to the same servers, instead of switching to new ones". And the real-time constraint is keeping the system time synchronized for all cloudlets. Causal+ and Real-Time causal consistency ensures ordering of relative operations, i.e. all causally related writes can be seen in the same order by all processes (connected to the same server). All that provides the best causal consistency guarantees[2] we can get for today.

III. Analysis and Discussion

A. Autonomy Requirements

We define always available autonomy for cloudlets as the following strict requirements:

- any operations performed on cloudlets state[3] fit data consistency models that allow the involved control/management planes operating as always available, and there is no read-only or blocking access limitations.
- cloudlets data can be modified at any given moment of time, despite of inter-cloudlets network connectivity[4].
- data can be synchronized eventually across cloudlets to/from[5] its adjacent aggregation edge layer, but not horizontally[6].
- data replication conflicts can be resolved automatically or by hand, and/or queued[7] for later processing[8].

[2]for *one-way convergent* [2] systems

[3]despite the cloudlets aliveness or failure conditions

[4]for disconnected/partitioned cloudlets, data can be modified via local control/management plane, if exists and not failed. Despite the adjacent *aggregation edge layer* [3] global view and/or quorum requirements

[5]bi-directional data replication is not a strict requirement though. For some cases, it is acceptable to have state replicated only from aggregation edge layer to cloudlet(s) under its control/management, like virtual machine or hardware provisioning images data. Or otherwise, logs, performance and metering statistics can flow from cloudlets to aggregation edge layer. Real-Time Causal

[6]also implies there is no horizontal data replication across neighbor aggregation edge layers. This corresponds to the acyclic graph (tree) topology

[7]depending on the numbers and allowed isolation periods of cloudlets under control/management, the disc and memory requirements for aggregation edge layers may vary drastically

[8]COPS [1] supports a similar failure mode for a datacenter: "the system administrator has two options: allow the queues to grow if the partition is likely to heal soon, or reconfigure COPS to no longer use the failed datacenter", except that the datacenter may be either failed or operating isolated long-time, and that we can have multiple such datacenters

B. Operational Invariants

To be always available as we defined it, control and management planes of cloudlets should provide the following operational capabilities (*invariants* hereafter):

- TBD (see ./ICFC-2019/challenges.md)

C. Data Consistency Requirements

The operational invariants dictate inevitable presense of shared state and sophisticated data replication mechanisms[9] among cloudlets. We know [4] that *causally consistent* [1] [2] data accessing mechanisms is the best fit for the declared invariants of the always available control and management planes.

OpenStack and OpenShift/Kubernetes, have yet causally consistent data backends[10] supported. The vision of the unified architecture based on such an always available data storage dictates us to not consider different backends for control and deployment data. Although generic version control systems, like Git, might fit all for the deployment data versioning, replicating and manual conflicts resolving, that would break the unified design approach for cloud data processing.

OpenStack cloud data is normally stored in databases via transactions based on stronger than causal *unavailable* [4] data consistency models, e.g. *serializable* [4], or *repeatable read* [4].

The weaker than causal+ and Real-Time Causal *total available* [4] consistency models may be considered as an alternative. Transactional global databases [5], may technically support it[11]. A weaker consistency model provides a really poor alternative though as it brings increased implementaion complexity, like corner cases handling for either the storage replicas, or client sided, or both, associated with relaxed constraints. E.g. *monotonic atomic view* [4] does not impose any real-time constraints, while Real-Time Causal does, which somewhat simplifies the end system design. Additionally, monotonic atomic view would require sophisticated handling of *fuzzy reads* [4], *phantoms* [4], discarded write-only transactions, empty state returned for any reads. All that makes that the strongest option for totally available data backends less preferable than causally (sticky) consistent ones.

Kubernetes clusters state is backed with Etcd, which only supports the stronger than Real-Time Causal consistency models.

D. Vision of a Unified Control/Cloud Data Storage Design

The definition we made for always available distributed systems self-explains why the causally (sticky) consistent storage backends is the best match for the cloudlets autonomy requirements and operational invariants as we defined those.

COPS formally proves implementation of a client library and highly scalable tooling for causal+ data operations. By

[9]when we refer to just *data* or *state*, we do not differentiate either that is deployment or control data

[10]that is, for control/deployment data only

[11]not Galera/MariaDB cluster though, as it has a strict quorum requierements for database writes

design, it does not impose any real-time constraints and supports a single Edge data center failure. The real tooling made off that base, may be operating on top of the nonshared local cloudlets databases, or key value storages (KVS), that provide the stronger consistency guarantees by the costs of reduced local availability[12]. That would work as weaker consistency guarantees work well, when built on top of the stronger ones, and provide an always available global view of cloudlets for the adjacent aggregation edge layer that controls/manages these cloudlets as the next hop connection. Replicating the state changes via causally related operations and conflicts resolving via custom handlers is that COPS covers as well.

The open questions are:

- Does COPS retains operations causal+ related when executed over multiple datacenters failure events (or extended time of being network partitioned)? Given the operational invariants, an aggregation edge layer cloudlet should allow all of its managed/controlled cloudlets running fully autonomous long-time, having all the outgoing operations queued and either eventually applied with conflicts resolved, or dropped/expired[13].
- Does COPS support two-way convergent systems, in terms of [2], for bi-directional causal+ replications?

TODO: find a use for Real-Time Causal and [6] alternatives to form more options for vision. Finally, make preferences for causal databases vs KVS, if possible?

IV. Conclusion

TBD

References

[1] W. Lloyd, M. J. Freedman, M. Kaminsky, and D. G. Andersen, “Dont settle for eventual: Scalable causal consistency for wide-area storage with COPS,” Proc. 23rd ACM Symposium on Operating Systems Principles (SOSP 11), Cascais, Portugal, October 2011.

[2] P. Mahajan, L. Alvisi, and M. Dahlin. “Consistency, availability, and convergence,” Technical Report TR-11-22, Univ. Texas at Austin, Dept. Comp. Sci., 2011.

[3] The Linux Foundation, “Open Glossary of Edge Computing,” [Online]. Available: https://github.com/State-of-the-Edge/glossary

[4] K. Kingsbury, “Consistency Models,” [Online]. Available: https://jepsen.io/consistency

[5] M. Bayer, “Global Galera Database,” [Online]. Available: https://review.openstack.org/600555

[6] M. M. Elbushra, J. Lindstrom, “Causal Consistent Databases”, Open Journal of Databases (OJDB), Volume 2, Issue 1, 2015.

[12]that is, the local view for a cloudlet and have no impact onto global views

[13]the global view of fully autonomous cloudlets may be represented for the agregation edge layer with the state marks, like “unknown/autonomous”, “synchronizing”, “connected”, “failed/disconnected/fenced”, if and only if it is confirmed as failed, or manually disconnected, or fenced automatically