Eventually Consistent Configuration Management in Fog Systems with CRDTs

Nick Stender, Tobias Pfandzelter, David Bermbach
Technische Universität Berlin & Einstein Center Digital Future
Mobile Cloud Computing Research Group
{nst,tp,db}@mcc.tu-berlin.de

***Abstract*—Fog systems rely on centralized and strongly consistent services for configuration management originally designed for cloud systems. Considering geo-distribution, such systems can exhibit high communication latency or become unavailable in case of network partition. In this paper, we examine the drawbacks of strong consistency for fog configuration management and propose an alternative based on CRDTs. We prototypically implement our approach for the FReD fog data management platform with promising early results.**

I. Introduction

Fog computing combines geo-distributed servers at the edge, in the cloud, and in the core network to support novel application domains such as the IoT [1], [2]. Fog platforms, e.g., *FogStore* [3] and *FReD* [4], use centralized configuration management systems with strong consistency. While desirable for easier configuration of replicas and availability clusters, this comes with an inherent performance penalty [5], [6] that is exacerbated in fog systems, which are highly geo-distributed with connections over the unreliable Internet.

Eventual consistency could enable distributed configuration management with low latency and high availability [5]. In this paper, we explore the QoS benefits of this approach and show drawbacks of eventual consistency in fog configuration management. We develop an alternative distributed configuration management system with eventual consistency for the fog data management platform FReD. We convert existing methods and data fields to use conflict-free replicated data types (CRDTs) that allow resolving consistency conflicts after they occur due to network partitions or delay [7], [8].

We make the following contributions:[1]

- We design an eventually consistent configuration management architecture for the FReD fog data management platform based on CRDTs (Section II).
- We prototypically implement this design and evaluate it experimentally (Section III).

II. CRDT-Based Configuration in FReD

In FReD, clients read and write data to *keygroups*, logically coherent data tables that can be accessed by application with a key/value interface and replicated to geographically diverse locations. As a central source of truth about the available FReD locations, replication instructions, and user authentication, the FReD *naming service* runs a centralized `etcd` cluster in the cloud. We propose replacing this centralized naming service with a decentralized CRDT-based approach with the goal of improving client access latency and network partition tolerance. This is especially relevant as reading configuration data is on the hot path of a client request to FReD: When a client reads data from a FReD node, the node has to check that the client is allowed to perform this read. Similarly, when an update request occurs, the FReD node has to read keygroup configuration that specifies to which other nodes in the fog network data should be replicated.

We use *Last-Write-Wins element set* (LWW), a state-based CRDT [7], to hold configuration data in our eventually consistent configuration service. Specifically, we use one set each for node information, keygroup configuration, system permission, and FReD node organization. We use a distributed bootstrapping approach where new nodes are informed of one existing node to create an overlay network. We use a gossip-style message dissemination where nodes periodically call other nodes to update their view of the network and discover unavailable nodes. We convert the following functionality of the FReD naming service:

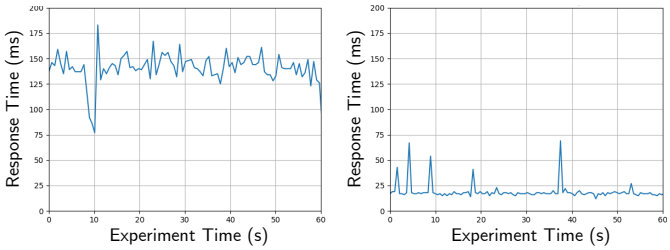
Node Registration: Instead of registering a new node with a central orchestrator, node identifier and address are sent to the bootstrapping node. As node creation happens infrequently and identifiers can easily be made unique, this is unlikely to lead to incorrect behavior. In case of a restart after failure LWW ensures that outdated information about a node is overwritten.

User Permission Changes: When an administrator makes permission changes for a user at the user's node, this node will immediately apply those changes. If message dissemination is slower than user movement, data staleness could lead to user permissions being outdated when switching nodes. The correlation between physical locations of users and nodes, and the data dissemination latency, however, makes this unlikely. Partitioned nodes are a challenge, as updated permission information cannot reach them. The only alternative to stale information is unavailability of the node, e.g., by disabling access for users when the partition is detected.

Keygroup Modification: Identifier uniqueness is paramount in creating keygroups. Concurrent creation of keygroups with identical names at two different nodes will lead to conflicts in LWW, yet the large identifier space makes this unlikely.

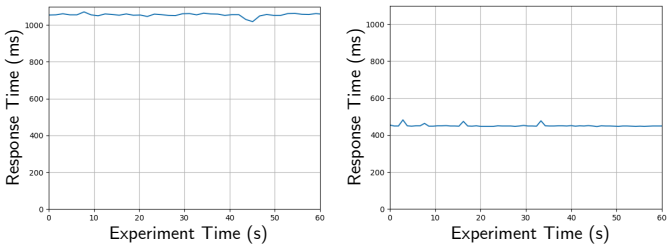
Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 415899119.

[1] An extended version of this paper is available as technical report [9].



(a) `etcd` (b) CRDT

Fig. 1: FReD response times without network delays between configuration service machines using `etcd` and CRDTs.



(a) `etcd` (b) CRDT

Fig. 2: FReD response times with 10ms delays between configuration service machines using `etcd` and CRDTs.

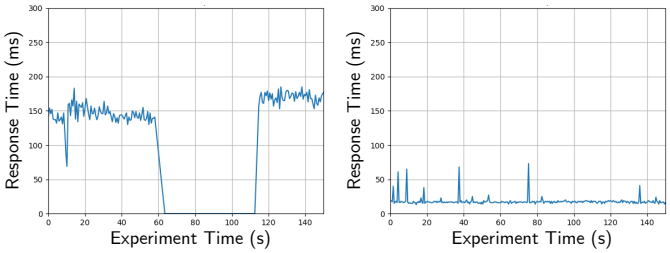
Keygroup Membership: Administrators and application can join and remove nodes from keygroups to specify data replication. Conflicts in eventually consistent configuration could occur only for changes made to the same keygroup, as memberships to keygroups is independent. If keygroup membership for a single node is modified concurrently, one of these changes is overwritten by LWW. This is unlikely, however, as each keygroup is managed by a single application.

III. Evaluation

We implement our CRDT-based service using *Go* and *gRPC*, making it compatible with FReD. In our experiments, we start FReD nodes as containers and compare the original `etcd` naming service implementation and our new CRDT-based system. Each naming service is distributed over at least three machines. We inject artificial network delays between nodes using `tc-netem`. A load generator connected to a FReD node measures request completion times.

Baseline: As a baseline, we compare configuration management approaches without network delay. To invoke write access to the naming service call the `createKeygroup` API of FReD to create keygroups from our load generator. The results in Fig. 1 show a higher delay for the `etcd` naming service. Although we expect this improvement to be caused mainly by the switch to a CRDT-based approach, we cannot rule out that our prototypical implementation is otherwise more efficient than production-ready `etcd`.

With Network Delays: Using an artificial network delay of 10ms, we evaluate the impact of communication delay between naming service machines. As shown in Fig. 2, this small communication delay increases FReD response times for both implementations. However, the total impact is more noticeable for the `etcd` naming service.



(a) `etcd` (b) CRDT

Fig. 3: FReD response times with configuration service machines using `etcd` and CRDTs. After 45s experiment time, we partition the network between configuration service machines.

Network Partitions: After running the experiment for 45 seconds, we introduce a network partition between the naming service machine used by our FReD node and the two others. We re-enable the network link after a further 35 seconds. As shown in Fig. 3, the partition impacts only the strongly consistent `etcd` implementation, where all requests fail during the partition (shown as a 0ms response time). Note also that it takes an additional 20 seconds after the network links are re-enabled for the system to recover. The CRDT-based implementation remains unaffected by this partition.

IV. Future Work

Future work will include a more comprehensive evaluation of the drawbacks of using eventually consistent configuration management in the fog. We also plan to explore the combination of strong consistency for some configuration data and eventual consistency for others. While complex, such a hybrid approach would allow for more efficient data dissemination without impacting application logic.

References

[1] F. Bonomi *et al.*, "Fog computing and its role in the internet of things," in *MCC '12*, Aug. 2012, pp. 13–16.
[2] D. Bermbach *et al.*, "A research perspective on fog computing," in *ISYCC 2017*, Nov. 2017, pp. 198–210.
[3] H. Gupta and U. Ramachandran, "Fogstore: A geo-distributed key-value store guaranteeing low latency for strongly consistent access," in *DEBS '18*, Jun. 2018, pp. 148–159.
[4] T. Pfandzelter *et al.*, "Managing data replication and distribution in the fog with fred," *Software: Practice and Experience*, 2023.
[5] T. Pfandzelter, T. Schirmer, and D. Bermbach, "Towards distributed coordination for fog platforms," in *CCGrid 2021*, May 2022, pp. 760–762.
[6] W. Vogels, "Eventually consistent," *Communications ACM*, vol. 52, no. 1, pp. 40–44, 2009.
[7] M. Shapiro *et al.*, "A comprehensive study of convergent and commutative replicated data types," INRIA, Tech. Rep. RR-7506, Jan. 2011.
[8] A. Jeffery, H. Howard, and R. Mortier, "Rearchitecting kubernetes for the edge," in *EdgeSys '21*, Apr. 2021, pp. 7–12.
[9] N. Stender, T. Pfandzelter, and D. Bermbach, "Fog system configuration management with crdts," TU Berlin & ECDF, Mobile Cloud Computing Research Group, Tech. Rep. MCC.2023.1, Jul. 2023.