Operation continuity in smart cities with an Edge-Cloud framework

Carmine Colarusso
Dept. of Engineering
University of Sannio
Benevento, Italy
ccolarusso@unisannio.it

Ida Falco
Dept. of Engineering
University of Sannio
Benevento, Italy
i.falco@studenti.unisannio.it

Eugenio Zimeo
Dept. of Engineering
University of Sannio and CINI
Benevento, Italy
zimeo@unisannio.it

***Abstract*—This extended abstract introduces an Edge-Cloud framework designed to migrate Cloud applications towards Edge-Cloud with the aim of reducing latency and ensuring operation continuity even during network interruptions. The approach has been evaluated using an IoT application that simulates intensive tracking activity from multiple retail shops.**

***Index Terms*—IoT, operation continuity, consistency, Edge-Cloud continuum.**

I. Introduction

Cloud computing is a very effective paradigm for easily and consistently sharing the state of a client/server application among numerous geographically distributed clients. However, its usage can cause high communication latency, bandwidth consumption, and operational discontinuity in the event of network failures. This is particularly felt in the context of Internet of Things (IoT) data-intensive applications, where devices generate vast amounts of data that require timely processing and analysis. In smart cities, this need is amplified as numerous interconnected devices continuously monitor and manage urban infrastructure, traffic, public safety, and environmental conditions. Network interruptions in such scenarios can severely impact real-time decisions and seamless operation of city services. Therefore, ensuring robust data processing and analysis despite network disruptions is crucial for maintaining the efficiency and reliability of smart city applications.

While Edge computing can reduce latency and bandwidth consumption, its naive adoption is unsuitable for ensuring state consistency, especially when a significant number of Edge nodes hosting a replicated portion of an application database are deployed. To ensure consistency, state modifications on one node must be promptly propagated to all other nodes. This requires sophisticated consistency models and synchronization protocols that balance real-time data access with network reliability and performance limitations.

Fig. 1 shows possible interaction scenarios with the application logic. The first case depicts a network failure in a full Cloud application that does not adopt failover mechanisms. In this case, network failures can lead to delays, packet loss, and increased latency, resulting in the unavailability of the whole system. The second case shows the adoption of an emergency network. However, the backup network could exhibit lower quality due to cost constraints, impacting the system's performance until the primary is restored. The last case shows a solution based on the Edge-Cloud continuum [1]. This solution reduces latency and protects the platform from network communication discontinuities. In this case, the system degrades consistency semantics until communication to the Cloud is restored, as for PACELC theorem [2].

This work has been partially supported by the SISMA (SolutIonS for engineering Microservices Applications) project funded by the Italian Ministry of University and Research [grant number 201752ENYB_002].

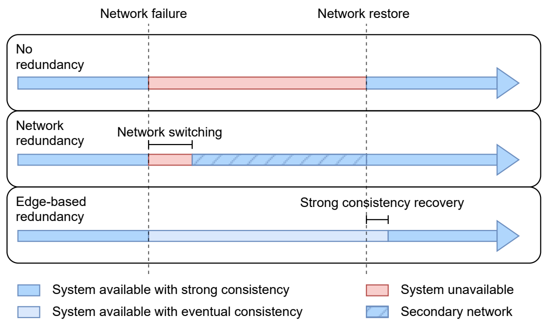

Fig. 1. Network failure countermeasures

According to the third scenario, we propose a framework to reorganize Cloud applications in a distributed and decentralized context, such as a smart city. The framework exploits the write-ahead logging pattern and event-driven communication to ensure that the state of the different Edge nodes is consistent so that any replicas can be used without loss of consistency.

II. Edge-Cloud continuum framework

By mirroring the application logic and data of an IoT application, already deployed in the Cloud, to the Edge, we expand the scope of Edge computing. Through data replication, we ensure that each Edge node can seamlessly access the needed data through an application logic proxy.

However, data replication may cause inconsistencies due to decentralized and concurrent transactions. To avoid this problem, the framework provides a reconciliation layer that lies in the Edge-Cloud continuum [3].

It is composed of: *Reconciliation Agent*, present on each Edge node, and a *Reconciliation Manager* in the Cloud, as

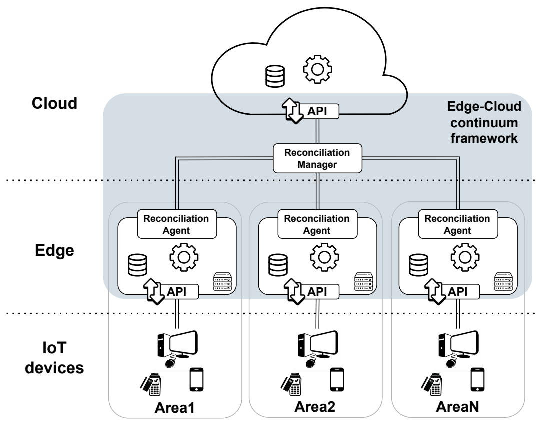

Fig. 2. Methodology overview

shown in Fig. 2. Every change operation that is performed on an Edge node must be intercepted and replicated on every other node to maintain a consistent state everywhere. When a client performs an operation on an Edge node, the *Reconciliation Agent* promotes write operations locally, leveraging event-driven log writing and propagation techniques. In this way, the operation will be visible to the *Reconciliation Manager* in the Cloud, who receives log updates and arbitrates transactions. At the same time, the *Reconciliation Agents* of the other nodes are responsible for retrieving the log entries in the Cloud, to synchronize their status. The other node could accept the new status if it does not conflict with his copy and send an outcome message to the *Reconciliation Manager*. This, in turn, will check the messages received from the different nodes and will be responsible for updating the authoritative statement in the Cloud and informing the nodes of the possibility of closing the transaction (accepting it or rolling it back).

During the reconciliation phase, the state of the different Edge nodes is marked as transient, but it is accessible with a relaxed level of consistency. Finally, strong consistency will occur during the last reconciliation phase, when the transaction is closed by the *Reconciliation Manager*.

III. Framework Evaluation

We tested our framework by exploiting a cluster of server machines managed and virtualized by OpenStack. With this IaaS software, we created a cluster of VMs. One VM is devoted to all the microservices and data stores that constitute the Cloud and the Reconciliation Manager. Each Edge node has a dedicated VM to host the replicated logic, data, and microservices that compose the Reconciliation Agent. The application logic simulates the tracking activity of a retail shop with massive data production and communication.

A. Test Results

Among various tests, we simulated the disconnection of one of the Edge nodes from the Cloud to observe the operational continuity perceived by users when transient failures occur. In this scenario, the system can operate locally on the data, which will be reconciled as soon as the network is re-established, thus ensuring only eventual consistency during connection failure. Upon the connection re-establishment, we assessed the time needed for the reconciliation mechanism to restore consistency throughout the system.

We performed a single-user test with 3 Edge nodes and a sequence of 125000 requests. After 90 seconds of load testing, we simulated a network failure by injecting a fault. The **R**ead operations could be carried out with a guarantee of eventual consistency, while **C**reate, **U**pdate, and **D**elete operations could be carried out locally and queued to be propagated as soon as the network is restored (this happens after 180 seconds from the beginning of the test). We used a network with 30 ms of latency (RTT) for this test. With this configuration, the time to obtain strong consistency is about 120 ms, while the time for the local read and write operations is about 10 ms. The response time for a full Cloud application is about 35 ms (due to network latency), which shows the solution's benefits.

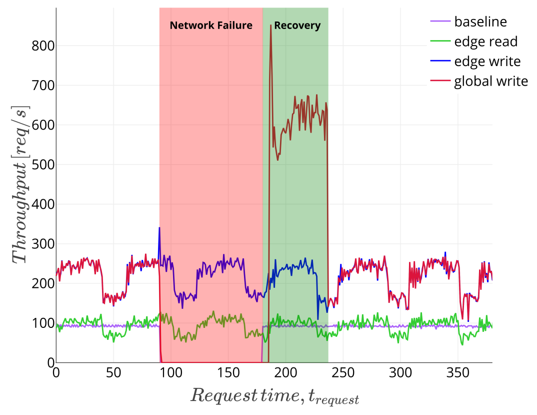

Fig. 3. Operation continuity results

Fig. 3 shows the throughput in different working conditions. The baseline throughput (purple line) observable with the full Cloud solution during the network failure drops to 0, causing unavailability. The figure also shows the throughput for the read (green line) and write (blue line) operations in the Edge-Cloud solution; these remain unchanged during the network disconnection because they are performed locally. To achieve strong consistency, the Cloud needs to close the transaction on each Edge node with an authoritative decision; the throughput of this global write (red line) drops to 0 when the connection with the *Reconciliation Manager* in the Cloud is lost. Upon network restoration, a recovery period starts, during which this throughput is higher than eventual consistency because of the enqueued messages on the Edge node. After 57 seconds from the network restoration, all the enqueued messages have been processed, and the system returns to normal operation.

The tests show that this approach outperforms full Cloud architecture in cases where a relaxed consistency is tolerated. Edge-Cloud continuum techniques allow for improving performances by reducing the response time of network-intensive applications and ensuring operation continuity in distributed environments with unstable connections.

References

[1] L. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira, M. Curado, L. Villas, L. DaSilva, C. Lee, and O. Rana, "The internet of things, fog and cloud continuum: Integration and challenges," *Internet of Things*, vol. 3-4, pp. 134–155, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660518300635

[2] D. Abadi, "Consistency tradeoffs in modern distributed database system design: Cap is only part of the story," *Computer*, vol. 45, no. 2, pp. 37–42, 2012.

[3] C. Colarusso, I. Falco, and E. Zimeo, "Towards business continuity with edge-cloud continuum," in *2024 11th International Conference on Future Internet of Things and Cloud (FiCloud)*, 2024.