The Case for Generic Edge Based Services

Diogo Jesus, Nuno Preguiça, and João Leitão

DI/FCT/NOVA

1 Introduction

With the rapid growth of devices connected to the Internet and the amount of applications running on them, traditional centralized computing models, in which data is sent to data centers for processing and storage, are becoming increasingly costly and inefficient. Edge computing allows to store and process data close to the users, usually at the network's periphery. This presents a way to mitigate some of the challenges induced by centralized computing models, such as the need to send data over the Internet to a remote data center for processing and storage, by offering a distributed computing paradigm in which data processing and storage are performed at the edge of the network, close to the clients and the sources of data [8], offering reduced latency by moving computations to the network's periphery, closer to clients (e.g., in ISP servers and 5G towers). Deploying independent services for each application in every edge location is costly and impractical, with generic services that can be used by multiple applications an alternative approach. In this work, we study how to use and implement these generic services deployed at the network's edge, in order to enhance the development of decentralized applications and explore the gains and benefits of placing computing close to the clients. In particular, we focus on services that can enhance the replication in decentralized systems, such as multiplayer games, collaborative applications, and secure blockchain-based applications.

2 Approach & Design

Several parties are expanding their infrastructures to the edge to put computation closer to the end user and allow faster response times, at the same time as the number of tools and frameworks available to deploy code on edge nodes also continue to expand. While some offer extra control to the developer in terms of deployment options in exchange of increased complexity and the need for further configuration, [6, 7] raising a set of challenges to be addressed such as reliability and security [1], others offer out-of-the-box solutions but restrain the amount of configurability available to the programmer. An example of the last case can be seen with some platforms like *AWS* and *CloudFlare* that offer services which allow one to put small computations on the periphery of the user, for instance, *Lambda@Edge Functions* and *CloudFlare Workers*, respectively.

Since in a real-world scenario it is infeasible for each application to be running its own services on every edge devices spread across the globe due to financial

costs, logistics, and security, we propose a set of generic services that can be used by different of applications and be feasible in a large deployment and in low resources computing environments. We predict that three possible deployment scenarios may arise, namely, (1) generic edge deployments at key edge locations like ISPs and 5G towers; (2) edge deployments on cloud providers infrastructures, such as Cloudflare or AWS; (3) deployments in routers at key locations.

2.1 Services

To test our proposal, we began by designing a *labeling service*, which allows an application to sign, timestamp, and verify the authenticity of an operation issued by a client , offering the possibility to guarantee that the operation wasn't tampered with by third parties. It can be a great auxiliary to replication protocols, such as causal consistency protocols that require a total order of operations to guarantee the correct synchronization between replicas [5].

Next, we designed a *get k-neighbours service* that allows a node to propose itself as a neighbour to other nodes that which to enter or participate in a determined application, as common in overlay networks.

Additionally, we designed and implemented a *randomness service* which allows a client to send arbitrary data to a randomness server that runs an oblivious pseudorandom function (OPRF) [3], which in turn allows clients to receive pseudorandom function evaluations on their data, without revealing anything about the actual contents of their actual content to the server

Finally, we present an *anonymization service* that implements the Oblivious HTTP protocol, which allows the forwarding of encrypted HTTP messages through the usage of Hybrid Public Key Encryption (HPKE) [2] by running a proxy that strips IP addresses from HTTP traffic, so that clients can make HTTP requests without revealing their identity to the destination of the request.

These last two services were later on used to implement and validate the STAR [4] protocol in an edge context, a protocol that provides distributed secret sharing for private threshold aggregation reporting.

3 Evaluation

In our evaluation we implemented and tested the mentioned services in two scenarios: an *edge scenario* where the code was deployed in the CloudFlare Workers edge nodes, thus having each client communicating with its nearest access point; and a *centralized scenario* where the code was deployed in AWS EC2 instances placed in similar distance to each client, making each client contact the single service instance. To execute the **edge scenario**, we used the Cloudflare Workers infrastructure, thus replicating the services automatically to all of Cloudflare access points spread across the globe and their respective *workers* (128MB RAM, up to 50ms of CPU time per request). On the other hand, the **centralized scenario** was executed by placing both services in London, on AWS EC2 (t3.micro,2 vcpu, 1GB RAM) servers. In both scenarios, we placed

clients in London, Madrid and Paris by running them on AWS EC2 (t2.micro,1 vcpu, 1GB RAM) instances. The aggregation server was deployed in London on an AW2 EC2 (t3.micro,2 vcpu, 1GB RAM) in both scenarios.

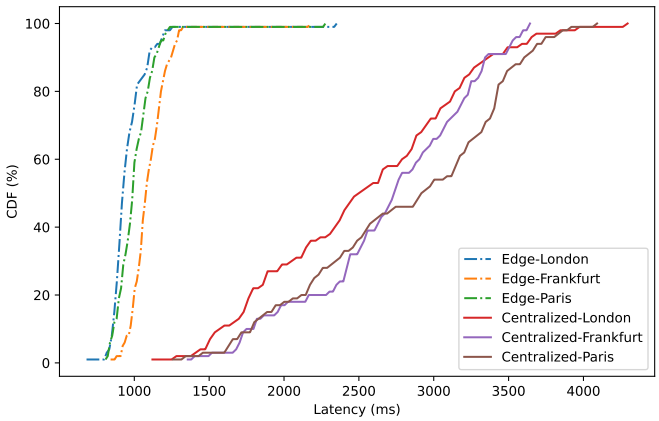

Fig. 1: Latencies on different settings

Our evaluation consisted of simulating an application that submits personal measurements of people to an aggregation server running the STAR protocol with the use of the aforementioned services. Our results showed that the edge scenario outperforms by far the centralized one, as depicted in Figure 1, mainly due to the lower latency introduced by having a client communicating with each service running in its periphery. Regarding clients running on the same scenario, we can see that their response times don't differ that much from each other, although London clients present an overall lower latency due to their proximity to the aggregation server, in both scenarios. Additionally, due to the distributed nature of the edge service, the load can be spread among nodes, thus preventing a bottleneck on the overall performance of the system. Since these services can be run independently from each other, thus preventing any need for further synchronization with each other (apart from cryptographic key rotations, that can be done in isolation), it shows an interesting usage scenario for these types of generic services.

4 Conclusions

In this work, we proposed a set of generic services deployed at the edge, as a means of bringing computation closer to the user and offering key services that can be used by a variety of applications, by comparing their implementation with a traditional centralized one.

The obtained results in our evaluation follow the line of what was expected, namely the reduced overall latency of the requests in the edge scenario, while not showing any downsides in terms of cost and functionality in having the deployment of the application in an edge context. Nevertheless, the distributed nature of these deployments offers a great set of benefits, such as proximity to the user and added robustness, that makes them a great fit for our proposal.

As future work, we plan to perform an extended testing of our proposed solution, namely using an open source FaaS platform [6], to test the edge scenario with a more bare-bones deployment, and, the evaluation of the aforementioned services for labeling and obtaining k-neighbours, to enhance the replication of distributed systems, namely secure causal consistency replication protocols.

Acknowledgements. This work was partially supported by NOVA LINCS (UIDB/04516/2020) with the financial support of FCT.IP

References

1. Balakrishnan, H., Banerjee, S., Cidon, I., Culler, D., Estrin, D., Katz-Bassett, E., Krishnamurthy, A., McCauley, M., McKeown, N., Panda, A., Ratnasamy, S., Rexford, J., Schapira, M., Shenker, S., Stoica, I., Tennenhouse, D., Vahdat, A., Zegura, E.: Revitalizing the public internet by making it extensible. SIGCOMM Comput. Commun. Rev. **51**(2), 18–24 (may 2021). https://doi.org/10.1145/3464994.3464998, https://doi.org/10.1145/3464994.3464998
2. Barnes, R., Bhargavan, K., Lipp, B., Wood, C.: Rfc 9180: Hybrid public key encryption (2022)
3. Davidson, A.: https://www.ietf.org/archive/id/draft-irtf-cfrg-voprf-06.txt
4. Davidson, A., Snyder, P., Quirk, E.B., Genereux, J., Livshits, B., Haddadi, H.: Star: Secret sharing for private threshold aggregation reporting. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. p. 697–710. CCS '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3548606.3560631, https://doi.org/10.1145/3548606.3560631
5. van der Linde, A., Leitão, J.a., Preguiça, N.: Practical client-side replication: Weak consistency semantics for insecure settings. Proc. VLDB Endow. **13**(12), 2590–2605 (jul 2020). https://doi.org/10.14778/3407790.3407847, https://doi.org/10.14778/3407790.3407847
6. OpenFaaS: Openfaas, https://www.openfaas.com/
7. Pfandzelter, T., Bermbach, D.: tinyfaas: A lightweight faas platform for edge environments. pp. 17–24 (04 2020). https://doi.org/10.1109/ICFC49376.2020.00011
8. Zhang, T., Li, Y., Philip Chen, C.: Edge computing and its role in industrial internet: Methodologies, applications, and future directions. Information Sciences **557**, 34–65 (2021). https://doi.org/https://doi.org/10.1016/j.ins.2020.12.021, https://www.sciencedirect.com/science/article/pii/S0020025520311865