D-Lambda: a fully decentralised serverless model

Divi De Lacour

D-Lambda: a fully decentralised serverless model

Divi De Lacour
Orange Innovation, IMT Atlantique, Inria
divi1.delacour@orange.com

Abstract

Functions as a Service (FaaS) is a crucial element of cloud computing, particularly for machine learning inferences. To reduce latency, cloud offloading has been proposed, positioning the FaaS provider closer to the client on the Edge-Cloud continuum. We extend this approach by proposing a multi-provider, fully decentralised model to enhance the flexibility, scalability, and resilience of FaaS. Additionally, we identify the necessary security properties and available countermeasures.

Keywords: Function as a Service, serverless, Decentralised, P2P, security

1 Introduction

Functions as a Service (FaaS) is a cloud computing approach that allows for the execution of functions without the need to manage the underlying infrastructure. This model offers a pay-per-use billing system and provides elasticity, making it particularly useful for machine learning (ML) model inference [17, 24].

Current FaaS approaches typically rely on a single cloud provider, this introduces latency issues due to the distance between the client and the data centre. In addition, this approach faces challenges in scalability and resilience due to the dependence on a single provider.

Multicloud approaches are used to call multiple providers at the same time, looking for the best performance/cost equilibrium and improve the scalability and resilience by handling multiple providers [23].

To improve FaaS latency, it has been proposed to execute loads closer to client on Edge-Cloud continuum. When Edge capacity is reached, cloud offloading transfers the load to the cloud to maintain availability, but at the cost of higher latency [13, 15].

P2P approaches have been proposed to handle cloud offloading [15] and load distribution [4, 5] in FaaS settings, although these approaches often leave security considerations underdeveloped.

Other works focus on the security of FaaS [9], especially on the use of Trusted Execution Environments (TEE) [20] to guarantee the computation integrity and data privacy.

In this paper, we propose a new model for decentralised FaaS, allowing for flexible scheduling and offloading. We identify the necessary security properties and the available countermeasures.

The paper is organised as follows. Section 2 reviews the FaaS and decentralisation approaches. Section 3 describes our proposed model. Section 4 identifies the FaaS security challenges our model faces and the possible countermeasures. Section 5 discusses the paths to be explored. Section 6 concludes.

2 Related works

Functions as a Service (FaaS) is a cloud computing model that allows developers to execute individual functions or pieces of code in response to specific events without the need to manage the underlying infrastructure. In this model, the cloud provider automatically provisions, scales, and manages the servers required to run the code [10]. Users are billed based on the actual execution time and resources consumed by their functions, rather than on pre-allocated server capacity.

2.1 FaaS challenges and solutions

Surveys such as [10, 17] have reviewed the challenges of serverless computing. We concentrate here on three primary challenges: scalability, resilience, and latency.

For scalability and resilience, two complementary approaches are possible: using multiple providers (the multicloud approach [23]) and improving FaaS scheduling. Utilizing multiple providers enhances elasticity and resilience by distributing the load across various cloud services [12]. Scheduling involves the efficient allocation of computational resources to execute functions in response to events. It optimizes performance, ensuring scalability and resilience. Effective scheduling dynamically distributes resources to handle varying workloads, maintaining system responsiveness and reliability.

Latency is another significant challenge, particularly in scenarios involving cold and warm starts. A cold start involves initializing a new execution environment, causing significant latency (can add more than 2 seconds of latency [14]), while a warm start uses an already-initialized environment for near-instant execution.

Solutions to mitigate latency include scheduling to prevent cold starts such as pre-warming (keeping execution environments warm by periodically invoking them), provisioned concurrency (asking the cloud provider to keep a specified number of execution always warm) and event event triggers (using an event trigger to pre-allocate resources). To reduce warm start latency, an approach is edge computing, placing the execution environment closer to the client in the IoT-cloud continuum.

To enhance resilience and scalability, decentralisation through peer-to-peer (P2P) approaches are the next step, as showed with IPFS [6] for file sharing. This allows multiple providers to be found and called by a client. This extension of the multicloud approach enables better distribution of providers, with some positioned closer in the continuum for improved latency. Additionally, the volunteer computing approach [3, 11] can be leveraged to exploit unused computing resources, particularly in consumer devices, further enhancing the system's scalability, resilience and latency.

2.2 Decentralised FaaS

Current works on decentralised FaaS, such as Serverledge [15], aim to offload computation to edge nodes and cloud regions. Serverledge organizes edge and cloud nodes into different regions and zones, allowing users to send requests to any edge node. When resource consumption becomes too high, scaling is handled automatically. Nodes can offload tasks vertically to the cloud or horizontally to another edge node. Another approach, DFaaS [4], uses a peer-to-peer (P2P) network for edge nodes to exchange status information and offload tasks if needed. Each 5G/WiFi station has an associated edge node, and a proxy within the node decides whether to offload tasks.

Existing approaches have limitations. They do not address security concerns. DFaaS and Serverledge require edge nodes with datacenter-like capabilities to handle numerous functions simultaneously. Serverledge also relies on a global registry, which poses resilience issue, and lacks customisability such as thematic nodes (e.g. LLM specialised nodes). Serverledge operates within a single cloud ecosystem with a single scheduler, limiting resilience and creating vendor lock-in.

3 Proposed model

In this section we present our D-Lambda architecture (figure 1). We propose to create a virtual broker on a peer-to-peer network. Clients and providers meet and negotiate the execution of functions on this P2P network.

We feature four components, separated in the client and provider infrastructures. On the client side: The **client** calls a function. The **querier** handles the search for a provider and the negotiation of the provisioning. On the provider side, the **provider** looks for potential clients and handles the negotiation. The **executor** executes the function.

This architecture allows for a client to use multiple providers at the same time for better scalability and resilience. It also allows to look for closer providers on the IoT-Cloud continuum for a lower latency.

We detail in figure 2 the execution process of our architecture. For the initialization part (cold start). The client first asks the querier to look for potential providers on the P2P network. The querier then contacts the potential providers and

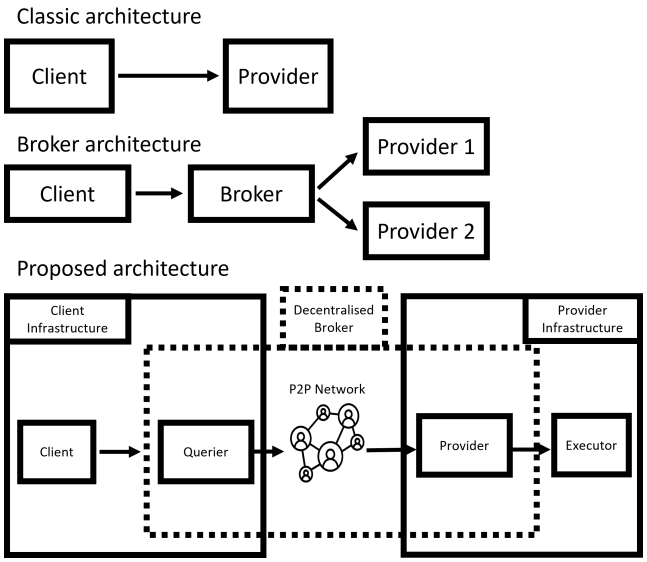

Figure 1. Decentralising the broker

negotiates the provision with them. Once the negotiation is done, the provider loads the function in the executor. The functions are then called by the client through the querier that transmits them to the providers executing them with their executor. On reception of the output, the querier sends an acknowledgement to the provider.

The search for a provider using a P2P network can be done in multiple ways. It can be done using a topical pubsub [2] or a specific key on a DHT [21]. Those rendez-vous points allow for a pre-selection of the type of provider for some topic (e.g. compute power, geographical zones). The search can be either active with the client broadcasting its requirement or passive with the providers advertising themselves.

The Querier has the responsibility to choose a reliable provider (e.g. Reputation systems [8], testing latency with a ping). The Querier verifies that the clients requirements are compatible with the providers specifications (e.g. listing 1). It is also in charge of anticipating the performance issues by offloading from the edge to the cloud or contacting multiple providers.

Listing 1. Example of provider specifications

```
"cpu": {"cores": 2},
"memory": {"size": "512MB"},
"gpu": {"vram": "2GB"},
"tee": {"enabled": true}
```

Once the Querier has found a possible provider, it negotiates a contract for the execution of the function. It is an agreement signed by both parties that specifies the conditions of execution (e.g. number of calls, cost, specific technologies like TEE, quality of Service).

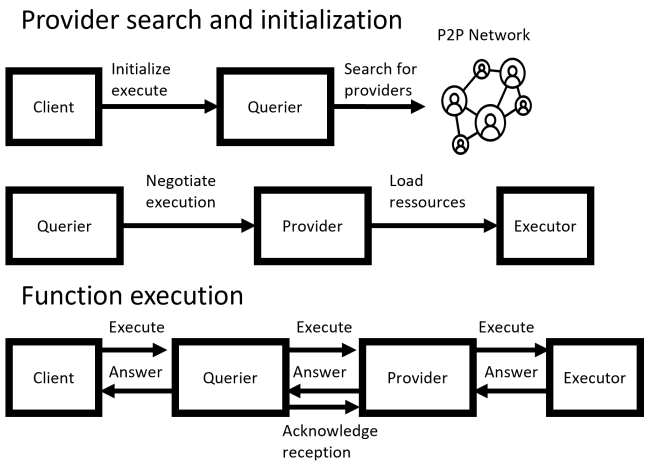

Figure 2. D-Lambda execution process

In case of conflict between the client and provider on the contract execution, the arbitrage can be done using a trusted third party or a smart contract on a blockchain.

4 Security

In section 3, we presented our model of decentralised FaaS. In this section we delve in the security of our model. We study how each actor (clients and providers) can protect themselves from malicious behaviors. We consider that components of the same infrastructure trust each other. We first identify the security properties to guarantee from the clients and providers point of view. We then identify the possible countermeasures.

4.1 Security properties to guarantee

From the *client point of view*, **Input/Output privacy** is the main privacy challenge: the system should prevent the provider from having access to the functions inputs and outputs. It may also require **Function Privacy** where the provider does not have access to the function executed on its infrastructure.
The client requires infrastructure integrity and availability, the **Integrity** of the function output is needed, the FaaS provider should also guarantee the **Availability** of executors to limit the impact on clients of failures and scaling.
The client also expects **Billing Honesty** from the provider where only the resources used are billed and **Behavior Privacy** where the provider and members of the network do not track abusively the client behavior in time and number of requests.
From the *provider point of view*, the main security challenges are **Execution Isolation** and **Billing Honesty**. Execution Isolation ensures that requests are isolated from the rest of the infrastructure. This means that they gain no reading or writing access to other functions being executed or to the provider's underlying infrastructure. Billing Honesty ensures that clients pay accurately for the execution of their requests.

4.2 Security countermeasures

We distinguish the following countermeasures to provide the security guarantees. There is no countermeasure covering the full set of protection requirements. They must be combined to provide the required security properties. However, their usage may induce overheads in compute and latency.

Execution Isolation – TEE/ Hardware isolation: TEEs like Intel TDX, ADM-SEV, ARM Trustzone are hardware isolated execution environments with encrypted memory that guarantees integrity of the execution and privacy of its data (Function and Input/Ouput Privacy). They feature remote attestation schemes that allows to remotely verify that a program is being executed in a TEE [20]. In our proposed model, the querier establishes an encrypted connection with the executor's TEE and remotely attests that it is connected to a TEE. It sends the workload (code and data) directly to the TEE and receives the output using the encrypted connection.
Software isolation include containers or WASM, which isolate the workload from the OS and other processes [18], preventing malicious workloads from interfering with other workloads.
Cryptographic countermeasures – Homomorphic encryption (HE) allows to compute on encrypted data, this guarantees the privacy of the Input and Output data [1]. It can be added to our model by encapsulating the function to be called. Current schemes are too costly to be applied.
Consensus techniques like blockchains or trusted third parties can act as arbitrators by reviewing the execution results to guarantee the integrity of execution. They can also review the exchanged messages to arbitrate the billing in case of conflict [22]. They are added in the querier and provider components. A reputation system [8] can be featured in the querier and provider to encourage availability, execution integrity of providers and billing honesty of both providers and clients by signaling bad behaviors.
Networking Counter-measures – Distributed Hash tables (DHT) [21] help by maintain the availability by providing more possible providers to the client in case of failures or scaling. Networking anonymity techniques like onion routing [19] and Mixnets [16] can help keep the client's behavior private by hiding its identity and its requests behaviour [7].

5 Discussion

The proposed model has multiple paths to be explored to improve performance. The first path to explore is the combination of security countermeasures and their impact on performance. Another approach is function storage, by hosting and sharing functions on IPFS for better resilience and

scalability. Scheduling improvements can be made by having providers detect the most demanded functions and preload them in anticipation, as well as by clients broadcast their anticipated load. Performance can also be enhanced by providers offering multiple functions simultaneously, as seen in models like tinyfaas. Additionally, "private networks" can be established to offer better security and performance guarantees, with their own reputation systems and search mechanisms. Finally, this FaaS model could also be extended to microservices, allowing for stateful loads.

6 Conclusion

We have presented the base architecture for a fully decentralised FaaS, designed to enhance scalability, resilience and latency. Additionally, we have identified the security aspects of this model and the associated countermeasures with their place in the architecture. Future work will focus on implementation and benchmarking, with particular attention to evaluating latency during both hot and cold starts.

References

[1] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. 2019. A Survey on Homomorphic Encryption Schemes: Theory and Implementation. *Comput. Surveys* 51, 4 (July 2019), 1–35. https://doi.org/10.1145/3214303

[2] Pedro Agostinho, David Dias, and Luís Veiga. 2022. SmartPubSub: Content-based Pub-Sub on IPFS. In *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. 327–330. https://doi.org/10.1109/LCN53696.2022.9843795 ISSN: 0742-1303.

[3] David P. Anderson. 2020. BOINC: A Platform for Volunteer Computing. *Journal of Grid Computing* 18, 1 (March 2020), 99–122. https://doi.org/10.1007/s10723-019-09497-9

[4] Michele Ciavotta, Davide Motterlini, Marco Savi, and Alessandro Tundo. 2021. DFaaS: Decentralized Function-as-a-Service for Federated Edge Computing. In *2021 IEEE 10th International Conference on Cloud Networking (CloudNet)*. 1–4. https://doi.org/10.1109/CloudNet53349.2021.9657141

[5] Claudio Cicconetti, Marco Conti, and Andrea Passarella. 2021. A Decentralized Framework for Serverless Edge Computing in the Internet of Things. *IEEE Transactions on Network and Service Management* 18, 2 (June 2021), 2166–2180. https://doi.org/10.1109/TNSM.2020.3023305 arXiv:2110.10974 [cs].

[6] Erik Daniel and Florian Tschorsch. 2022. IPFS and Friends: A Qualitative Comparison of Next Generation Peer-to-Peer Data Networks. *IEEE Communications Surveys & Tutorials* 24, 1 (2022), 31–52. https://doi.org/10.1109/COMST.2022.3143147 Conference Name: IEEE Communications Surveys & Tutorials.

[7] Benjamin Greschbach, Gunnar Kreitz, and Sonja Buchegger. 2012. The devil is in the metadata — New privacy challenges in Decentralised Online Social Networks. In *2012 IEEE International Conference on Pervasive Computing and Communications Workshops*. 333–339. https://doi.org/10.1109/PerComW.2012.6197506

[8] Ferry Hendrikx, Kris Bubendorfer, and Ryan Chard. 2015. Reputation systems: A survey and taxonomy. *J. Parallel and Distrib. Comput.* 75 (Jan. 2015), 184–197. https://doi.org/10.1016/j.jpdc.2014.08.004

[9] Xing Li, Xue Leng, and Yan Chen. 2023. Securing Serverless Computing: Challenges, Solutions, and Opportunities. *IEEE Network* 37, 2 (March 2023), 166–173. https://doi.org/10.1109/MNET.005.2100335 Conference Name: IEEE Network.

[10] Yongkang Li, Yanying Lin, Yang Wang, Kejiang Ye, and Chengzhong Xu. 2023. Serverless Computing: State-of-the-Art, Challenges and Opportunities. *IEEE Transactions on Services Computing* 16, 2 (March 2023), 1522–1539. https://doi.org/10.1109/TSC.2022.3166553

[11] T. M. Mengistu and D. Che. 2019. Survey and Taxonomy of Volunteer Computing. *ACM Comput. Surv.* 52, 3 (2019).

[12] Fawaz Paraiso, Philippe Merle, and Lionel Seinturier. 2013. Managing elasticity across multiple cloud providers. In *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds*. ACM, Prague Czech Republic, 53–60. https://doi.org/10.1145/2462326.2462338

[13] Tobias Pfandzelter and David Bermbach. 2020. tinyFaaS: A Lightweight FaaS Platform for Edge Environments. In *2020 IEEE International Conference on Fog Computing (ICFC)*. 17–24. https://doi.org/10.1109/ICFC49376.2020.00011

[14] Sashko Ristov, Christian Hollaus, and Mika Hautz. 2022. Colder Than the Warm Start and Warmer Than the Cold Start! Experience the Spawn Start in FaaS Providers. In *Proceedings of the 2022 Workshop on Advanced tools, programming languages, and PLatforms for Implementing and Evaluating algorithms for Distributed systems*. ACM, Salerno Italy, 35–39. https://doi.org/10.1145/3524053.3542751

[15] Gabriele Russo Russo, Tiziana Mannucci, Valeria Cardellini, and Francesco Lo Presti. 2023. Serverledge: Decentralized Function-as-a-Service for the Edge-Cloud Continuum. In *2023 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 131–140. https://doi.org/10.1109/PERCOM56429.2023.10099372 ISSN: 2474-249X.

[16] K. Sampigethaya and R. Poovendran. 2006. A Survey on Mix Networks and Their Secure Applications. *Proc. IEEE* 94, 12 (Dec. 2006), 2142–2181. https://doi.org/10.1109/JPROC.2006.889687

[17] Hossein Shafiei, Ahmad Khonsari, and Payam Mousavi. 2022. Serverless Computing: A Survey of Opportunities, Challenges, and Applications. *Comput. Surveys* 54, 11s (Nov. 2022), 239:1–239:32. https://doi.org/10.1145/3510611

[18] Simon Shillaker and Peter Pietzuch. 2020. Faasm: Lightweight Isolation for Efficient Stateful Serverless Computing. 419–433. https://www.usenix.org/conference/atc20/presentation/shillaker

[19] Paul Syverson, Roger Dingledine, and Nick Mathewson. 2004. Tor: The secondgeneration onion router. In *Usenix Security*. USENIX Association Berkeley, CA, 303–320.

[20] Bohdan Trach, Oleksii Oleksenko, Franz Gregor, Pramod Bhatotia, and Christof Fetzer. 2019. Clemmys: towards secure remote execution in FaaS. In *Proceedings of the 12th ACM International Conference on Systems and Storage*. ACM, Haifa Israel, 44–54. https://doi.org/10.1145/3319647.3325835

[21] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. 2011. A survey of DHT security techniques. *Comput. Surveys* 43, 2 (Feb. 2011), 8:1–8:49. https://doi.org/10.1145/1883612.1883615

[22] Yang Xiao, Ning Zhang, Wenjing Lou, and Y. Thomas Hou. 2020. A Survey of Distributed Consensus Protocols for Blockchain Networks. *IEEE Communications Surveys & Tutorials* 22, 2 (2020), 1432–1465. https://doi.org/10.1109/COMST.2020.2969706 arXiv: 1904.04098.

[23] Haidong Zhao, Zakaria Benomar, Tobias Pfandzelter, and Nikolaos Georgantas. 2022. Supporting Multi-Cloud in Serverless Computing. In *2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*. 285–290. https://doi.org/10.1109/UCC56403.2022.00051

[24] Ming Zhao, Kritshekhar Jha, and Sungho Hong. 2023. GPU-enabled Function-as-a-Service for Machine Learning Inference. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 918–928. https://doi.org/10.1109/IPDPS54959.2023.00096 ISSN: 1530-2075.