Serverless Dataflows

1st Diogo Jesus
dept. name of organisation (of Aff.)
Instituto Superior Tecnico (IST), INESC-ID Lisboa
Lisbon, Portugal
diogofjesus@inesc-id.pt

2nd Luís Veiga
dept. name of organisation (of Aff.)
Instituto Superior Tecnico (IST), INESC-ID Lisboa
Lisbon, Portugal
luis.veiga@inesc-id.pt

***Abstract*—Serverless computing has become a suitable cloud paradigm for many applications, prized for its operational ease, automatic scalability, and fine-grained pay-per-use pricing model. However, executing workflows–compositions of multiple tasks–in Function-as-a-Service (FaaS) environments remains inefficient. This inefficiency stems from the stateless nature of functions, and a heavy reliance on external services for intermediate data transfers and inter-function communication.**

In this document, we introduce a decentralized DAG engine that leverages historical metadata to plan and influence task scheduling. Our solution encompasses metadata management, static workflow planning, and a worker-level scheduling strategy designed to drive workflow execution with minimal synchronization. We compare our system against WUKONG, another decentralized serverless DAG engine, and Dask Distributed, a more traditional cluster-based DAG engine. Our evaluation demonstrates that utilizing historical information significantly improves performance and reduces resource utilization for workflows running on serverless platforms.

***Index Terms*—Cloud Computing, Serverless, FaaS, Serverless Workflows, DAG, Metadata, Workflow Prediction**

I. Introduction

Serverless computing, with its Function-as-a-Service (FaaS) model, offers operational simplicity, automatic scalability, and a fine-grained pay-per-use pricing model by abstracting infrastructure management. This has led to its widespread adoption for event-driven systems, microservices, and web services on platforms like AWS Lambda[1], Azure Functions[2], and Google Cloud Functions[3].

This paradigm is also increasingly used to execute complex scientific and data processing workflows, such as the Cybershake [1] seismic hazard analysis or Montage [2], an astronomy image mosaicking workflow. These applications are structured as workflows—formally represented as Directed Acyclic Graphs (DAGs) of interdependent tasks. However, efficiently executing these complex workflows on serverless platforms remains a significant challenge. The stateless nature of serverless functions hinders direct communication, forcing data exchange between these tasks through external storage and leading to performance degradation and increased latency for data-intensive workflows.

Existing solutions, including commercial stateful functions and research prototypes like WUKONG [3], address these issues with improved orchestration and decentralized scheduling. Yet, these approaches often rely on one-step scheduling, making decisions based solely on the immediate workflow stage without considering global implications.

In this paper, we argue that leveraging historical metadata from previous workflow runs can provide critical insights into task behavior. We propose a scheduler that uses this information to make smarter scheduling decisions, aiming to reduce overall workflow makespan and increase resource efficiency on serverless platforms.

[1]https://aws.amazon.com/pt/lambda/
[2]https://azure.microsoft.com/en-us/products/functions
[3]https://cloud.google.com/functions

II. Related Work

Our work builds upon and differs from existing research in serverless workflow orchestration, which can be broadly categorised into **cloud-native stateful services**, **FaaS runtime extensions**, and **decentralized serverless schedulers**.

A. Cloud-Native Stateful Orchestration

Commercial platforms like AWS Step Functions [4], Azure Durable Functions [5], and Google Cloud Workflows [6] provide a managed solution for composing serverless functions into workflows. They handle state persistence and fault tolerance, abstracting orchestration complexity from the developer. However, they are often tightly coupled with their provider's ecosystem, can introduce vendor lock-in, and are generally designed for reliability and ease of use rather than performance, optimal resource usage or data locality.

B. FaaS Runtime Extensions for Data Locality

A body of work seeks to address the fundamental data exchange inefficiency in serverless platforms through runtime modifications. **Palette** [7] introduces locality "hints" to co-locate related function invocations on the same worker. **Faa$T** [8] provides a transparent, auto-scaling distributed cache for serverless functions. **Lambdata** [9] requires developers to declare data intents (input/output objects) to enable data-aware scheduling. These solutions demonstrate the significant performance gains possible by improving data locality, but often require modifications to the application code or the underlying FaaS platform itself, limiting their portability and adoption.

C. Decentralized Serverless Schedulers

Several frameworks have been designed to execute workflows efficiently on unmodified serverless infrastructure. **PyWren** [10] is an early orchestrator for embarrassingly parallel computations, but its simple design can be inefficient for more complex workflows. **Unum** [11] decentralizes orchestration logic by embedding it within application code, allowing portability across different cloud providers, but offering limited data locality optimizations.

The most directly comparable work to ours is **WUKONG** [3], a decentralized DAG engine that uses static scheduling and runtime optimizations to minimize data movement. WUKONG's key innovations include:

- **Decentralized Scheduling:** Eliminating the central scheduler bottleneck;
- **Task Clustering:** Forcing the execution of sequences of tasks on the same worker to reuse intermediate results and avoid using external storage;
- **Delayed I/O:** Heuristically postponing data writes to external storage in the hope that dependent tasks can be executed locally.

While WUKONG represents a significant advance, its scheduling and optimizations are based solely on the structure of the *current* workflow DAG. It employs a **one-step scheduling** policy, making decisions using immediate runtime information without leveraging knowledge it has about the entire workflow. Besides that, WUKONG also uses optimizations based on heuristics, which can lead to suboptimal performance when workflow behavior deviates from expected patterns.

D. Discussion

The Function-as-a-Service (FaaS) model offers a transformative approach to cloud computing by abstracting infrastructure management and enabling developers to focus on business logic. Despite current platform limitations, Serverless architectures have been widely adopted for event-driven applications, microservices, IoT processing, and web services.

While researching, we explored some modern cloud-native solutions that seamlessly integrate with cloud environments. We also found innovative extensions to the FaaS runtime that aim to improve **data locality**, a technique that can enhance performance and resource efficiency of serverless workflows by reducing data transfer overheads and removing the need to use external services for synchronization.

Finally, we compared serverless workflow execution orchestrators and schedulers, from which we found WUKONG to be the most interesting, innovative and promising approach for exploiting the most out of the serverless computing paradigm. WUKONG achieves **fast scale-out times** by delegating part of the worker instancing to an external component, **scalability** with its distributed scheduling approach, and **data locality** with its optimizations that try to run related tasks on the same worker, minimizing data transfers. Despite its advantages, we also found that WUKONG's heuristic optimizations could become inefficient in some scenarios. We believe that WUKONG scheduling decisions are optimized for workflows with short and uniform tasks.

By analyzing related works, we didn't find solutions that tried to make scheduling decisions based on the entire workflow nor that used historical information to make scheduling decisions. We saw this as a research opportunity and decided to explore it with the goal of reducing makespan of workflows running on top of FaaS while also using fewer resources, thereby improving cost efficiency and enabling a wider variety of workflows to run on top of FaaS.

III. Architecture

IV. Evaluation and Analysis

V. Conclusion

References

[1] R. Graves, T. H. Jordan, S. Callaghan, E. Deelman, E. Field, G. Juve, C. Kesselman, P. Maechling, G. Mehta, K. Milner, D. Okaya, P. Small, and K. Vahi, "Cybershake: A physics-based seismic hazard model for southern california," *Pure and Applied Geophysics*, vol. 168, no. 3, pp. 367–381, 2011. [Online]. Available: https://doi.org/10.1007/s00024-010-0161-6

[2] J. C. Jacob, D. S. Katz, G. B. Berriman, J. C. Good, A. Laity, E. Deelman, C. Kesselman, G. Singh, M.-H. Su, T. Prince, and R. Williams, "Montage: A grid portal and software toolkit for science-grade astronomical image mosaicking," *International Journal of Computational Science and Engineering*, vol. 4, no. 2, pp. 73–87, 2009. [Online]. Available: https://doi.org/10.1504/IJCSE.2009.026999

[3] B. Carver, J. Zhang, A. Wang, A. Anwar, P. Wu, and Y. Cheng, "Wukong: A scalable and locality-enhanced framework for serverless parallel computing," in *Proceedings of the 11th ACM symposium on cloud computing*, 2020, pp. 1–15.

[4] Aws step functions. [Online]. Available: https://aws.amazon.com/en/step-functions/

[5] Azure durable functions. [Online]. Available: https://learn.microsoft.com/en-us/azure/azure-functions/durable/durable-functions-overview

[6] Google cloud workflows. [Online]. Available: https://cloud.google.com/workflows

[7] M. Abdi, S. Ginzburg, C. Lin, J. M. Faleiro, I. Goiri, G. I. Chaudhry, R. Bianchini, D. S. Berger, and R. Fonseca, "Palette load balancing: Locality hints for serverless functions," in *EuroSys*. ACM, May 2023. [Online]. Available: https://www.microsoft.com/en-us/research/publication/palette-load-balancing-locality-hints-for-serverless-functions/

[8] F. Romero, G. I. Chaudhry, I. n. Goiri, P. Gopa, P. Batum, N. J. Yadwadkar, R. Fonseca, C. Kozyrakis, and R. Bianchini, "Faa$t: A transparent auto-scaling cache for serverless applications," in *Proceedings of the ACM Symposium on Cloud Computing*, ser. SoCC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 122–137. [Online]. Available: https://doi.org/10.1145/3472883.3486974

[9] Y. Tang and J. Yang, "Lambdata: Optimizing serverless computing by making data intents explicit," in *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*. IEEE, 2020, pp. 294–303.

[10] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, and B. Recht, "Occupy the cloud: Distributed computing for the 99%," in *Proceedings of the 2017 symposium on cloud computing*, 2017, pp. 445–451.

[11] D. H. Liu, A. Levy, S. Noghabi, and S. Burckhardt, "Doing more with less: Orchestrating serverless applications without an orchestrator," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 1505–1519.