Oakestra: An Orchestration Framework for Edge Computing

Giovanni Bartolomeo Simon Bäurle Nitinder Mohan Jörg Ott
Technical University of Munich

ABSTRACT

Edge computing enables developers to deploy their services on compute resources deployed closer to the users. The abstraction requires powerful orchestration capabilities and the resolution of complex optimization problems. While edge computing is a consistently growing trend, the community (research and industry) still largely embraces adaptations and extensions of existing cloud technologies that have been proven ineffective on edge (e.g. Kubernetes). In this work, we present Oakestra, a novel hierarchical orchestration framework specifically designed for supporting service operation over heterogeneous edge infrastructures. In this demonstration, we showcase the various features and operations of Oakestra using our latency-critical augmented reality (AR) application.

CCS CONCEPTS

• **Computer systems organization** → *Distributed architectures*; **Heterogeneous (hybrid) systems**; • **Software and its engineering** → *Development frameworks and environments.*

KEYWORDS

Edge Computing, Orchestration Framework, Resource Management

ACM Reference Format:
Giovanni Bartolomeo, Simon Bäurle, Nitinder Mohan, Jörg Ott. 2022. Oakestra: An Orchestration Framework for Edge Computing. In *ACM SIGCOMM 2022 Conference (SIGCOMM '22 Demos and Posters), August 22–26, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3546037.3546056

1 INTRODUCTION

Within a decade since inception, edge computing has found a wide range of industrial and research use-cases [14, 23, 24]. In order to cope with the strict QoS requirements of latency-critical applications [11, 22], orchestration frameworks for edge computing must support efficient service management along with cloud offloading capabilities [1, 26]. Heterogeneity of the resources and fluctuations in the network makes orchestration particularly challenging [9, 10, 13, 25]. Popular approaches for orchestrating edge infrastructures adapt existing cloud-native technologies [27]. For example, solutions like Kubernetes (k8s) [6], even the lightweight distributions like k3s [8], KubeEdge [7], and microk8s [5] struggle at the edge due to their strong consistency assumptions [4, 20]. Furthermore, edge adaptations of k8s [17, 18, 20] carry-forward the burden of k8s footprint on contrained hardware resources. Other research orchestration approaches at the edge like [2, 15, 16, 19, 21, 28] also fail to (a) integrate cloud and edge workloads with minimal efforts, (b) manage geographically distributed heterogeneous hardware clusters, and (c) interconnect coordinated microservices considering diverse processing and network constraints. As a solution, with our work we bring the following contributions:

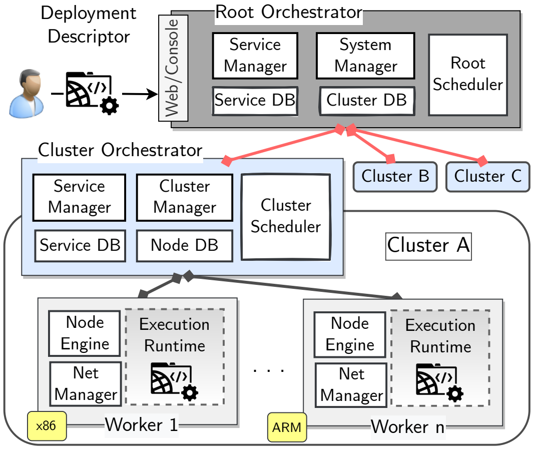

Figure 1: Oakestra Architecture

(1) We propose Oakestra, a novel hierarchical orchestration framework that supports federated infrastructures for application deployment, management, and computation offloading. Different clusters of heterogeneous resources over multiple geographical locations can be combined to build a federated cloud-to-edge continuum. We implement Oakestra as lightweight and scalable framework supporting compute-constrained edge devices in combination with powerful cloud resources with minimal overhead.

(2) Application providers can deploy services at the edge by specifying high-level SLA, such as hardware, latency, bandwidth, etc. as constraints within deployment descriptors. Oakestra's hierarchical management structure, in conjunction with the delegated service scheduling priciple, allows it to decentralize task placement and find effective deployment much faster than the state-of-the-art.

(3) Using semantic IP addresses we natively support flexible and transparent load balancing techniques. This way, we provide supsport for service mobility and hardware constraints while enabling easy portability of applications as no code adaptation is required. Moreover, the platform provides an overlay network for service-to-service communication to allow the traversal of NATs and firewalls.

2 OAKESTRA: A PRIMER

Oakestra is a lightweight orchestration framework natively designed to support the many constraints of edge environments. The key innovation lies in the two-tier logical hierarchical orchestration of compute resources which, unlike flat architectures of k8s-inspired frameworks, decomposes the control plane management

SIGCOMM '22 Demos and Posters, August 22–26, 2022, Amsterdam, Netherlands
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9434-5/22/08.
https://doi.org/10.1145/3546037.3546056

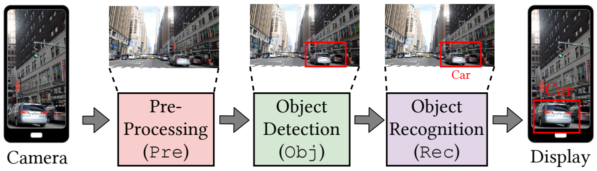

Figure 2: AR Application Pipeline

into many clusters. As shown in fig. 1, each cluster is managed by its local *cluster orchestrator* which is only responsible for resources within its cluster. The cluster orchestrator coordinates with the parent *root orchestrator* and sends it aggregated resource utilization of the cluster (such as CPU/GPU cores, memory, and storage) and current service operation statistics (e.g. resources consumed, location, latency, etc.). Application developers only interact with the root orchestrator for deploying their services at the edge – submitting the *deployment descriptor* containing the SLA requirements along with the service code via API/CLI/Web UI. The SLA may include latency thresholds or geographical area constraints, resource requirements (bandwidth in/out, vCPU/vGPU/vTPU cores, memory, and disk size), and convergence time for service scheduling operations. Thanks to its hierarchical structure and *delegated scheduling mechanism*, Oakestra can quickly resolve service scheduling requests at the edge, which is a well-known NP-hard problem [12]. The root orchestrator first calculates the best-fit clusters for every service request by broadly mapping the requirements to aggregated statistics it received from the cluster orchestrators. Further, it offloads the service request to the cluster scheduler, which finds the optimal deployment of the resources within the cluster. Since the problem space is greatly reduced, frequent service (re)scheduling is no longer costly using Oakestra.

We also design and implement Oakestra such that multiple edge (or cloud) operators can contribute their resources (as different clusters) to shared infrastructure and retain administrative control. For instance, we allow each cluster operator to incorporate different scheduling policies within their clusters, such as latency, fairness, etc. To support service interactions over resources across different clusters (possibly behind NATs/firewalls), *Net Manager* in the worker utilizes a novel *semantic addressing* scheme which can dynamically (and transparently) adjust communication endpoints in response to infrastructure changes, e.g., service migrations, resource failures, etc., ensuring uninterrupted service interactions. Moreover, Oakestra is extremely lightweight which allows it to operate effectively over highly compute-constrained resources with minimal overhead. Our experiments over real edge infrastructures reveals that Oakestra outperforms state-of-the-art solutions like k8s and k3s. In comparison, we achieved a ≈ 10× reduction on computing resources footprint and 10% application performance improvement measured in terms of max FPS achieved by the same AR pipeline (not shown due to space restrictions).

3 DEMONSTRATION

This demonstration will showcase the capabilities of Oakestra in an edge-cloud infrastructure using a latency-critical AR application.
AR Application. The AR pipeline (see fig. 2) is composed of three networked microservices [3]. Pre is the *pre processing* service that takes live camera stream and scales down the image as per the model specification of rest of the pipeline. Obj performs *object detection* on each frame and outputs the bounding boxes coordinates. Rec performs *object recognition* on the bounding boxes found by Obj and labels them. It sends the output to the display which scales up the output to original resolution. Each component of the pipeline can be replicated in multiple instances and is GPU accelerated.

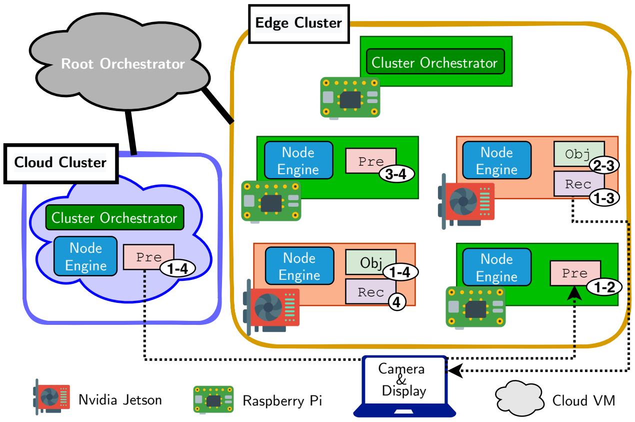

Figure 3: Demonstration Setup Steps

Oakestra Setup. We will create a two cluster infrastructure deployment managed by Oakestra as shown in fig. 3. The *edge cluster* will be composed of heterogeneous resources (Raspberry Pi's and Jetsons) on the demo table while the *cloud cluster* will have VMs in a cloud datacenter. The *root orchestrator* will also be in the cloud, albeit from a different cloud operator than cloud cluster to highlight the multi-cloud operation. We will demonstrate (i) how to deploy applications at the edge using Oakestra's APIs, (ii) how Oakestra handles sudden spikes in application load via reschedules/replications, and (iii) how Oakestra transparently handles resource and service failures at the edge. Simultaneously, audience can observe the live performance of the application and the operational load of the infrastructure.
Demonstration Design. In step ①, we will deploy the AR application using Oakestra's web interface which will schedule its microservices on edge resources. Additionally, we will deploy another instance of Pre in the cloud such that both instances are independentally operational in the infrastructure. In step ② we will scale the number of clients, thereby increasing the load on the pipeline. As a consequence, Oakestra will scale-up the bottleneck Obj instances in real-time – which will be evident from latency reduction in the pipeline output. In step ③ we will demostrate resilience of Oakestra by injecting node failures in edge cluster – abruptly killing the Pre service, which will seamlessly re-route client traffic to the instance in cloud cluster. In meantime, Oakestra will re-deploy the failed Pre instance on an unused resource in the edge cluster and will resume service operation at the edge. Finally, in step ④, we will kill the node hosting the only Rec service instance which requires GPU for operation – resulting in loss of labels in pipeline output. As a result, Oakestra will reschedule the failed instance of Rec to the only available GPU node in edge cluster, restoring optimal pipeline operation with minimal downtime.

Acknowledgements. This work is supported by EU Celtic project Piccolo (C2019/2-2), Federal Ministry of Education and Research of Germany (BMBF) project 6G-Life (16KISK002) and Bavarian Ministry of Economic Affair project 6G Future Lab Bavaria.

REFERENCES

[1] Hadeel Abdah, Joaão Paulo Barraca, and Rui L. Aguiar. 2019. QoS-Aware Service Continuity in the Virtualized Edge. *IEEE Access* 7 (2019), 51570–51588. https://doi.org/10.1109/ACCESS.2019.2907457

[2] Brian Amento, Bharath Balasubramanian, Robert J. Hall, Kaustubh Joshi, Gueyoung Jung, and K. Hal Purdy. 2016. FocusStack: Orchestrating Edge Clouds Using Location-Based Focus of Attention. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*. 179–191. https://doi.org/10.1109/SEC.2016.22

[3] Simon Bäurle and Nitinder Mohan. 2022. ComB: A Flexible, Application-Oriented Benchmark for Edge Computing. In *Proceedings of the 5th International Workshop on Edge Systems, Analytics and Networking* (Rennes, France) *(EdgeSys '22)*. Association for Computing Machinery, New York, NY, USA, 19–24. https://doi.org/10.1145/3517206.3526269

[4] Sebastian Böhm and Guido Wirtz. 2021. Profiling Lightweight Container Platforms: MicroK8s and K3s in Comparison to Kubernetes.. In *ZEUS*.

[5] Canonical. 2018. MicroK8s. Retrieved May 24, 2022 from https://microk8s.io

[6] CNCF. 2015. Kubernetes - Production-Grade Container Orchestration. Retrieved May 24, 2022 from https://kubernetes.io

[7] CNCF. 2017. KubeEdge. Retrieved May 24, 2022 from https://kubeedge.io/en/

[8] CNCF. 2019. Lightweight Kubernetes | K3S. Retrieved May 24, 2022 from https://k3s.io

[9] Lorenzo Corneo, Maximilian Eder, Nitinder Mohan, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, Per Gunningberg, Jussi Kangasharju, and Jörg Ott. 2021. Surrounded by the Clouds: A Comprehensive Cloud Reachability Study. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 295–304. https://doi.org/10.1145/3442381.3449854

[10] Lorenzo Corneo, Nitinder Mohan, Aleksandr Zavodovski, Walter Wong, Christian Rohner, Per Gunningberg, and Jussi Kangasharju. 2021. (How Much) Can Edge Computing Change Network Latency?. In *2021 IFIP Networking Conference (IFIP Networking)*. 1–9. https://doi.org/10.23919/IFIPNetworking52078.2021.9472847

[11] Vittorio Cozzolino, Leonardo Tonetto, Nitinder Mohan, Aaron Yi Ding, and Jorg Ott. 2022. Nimbus: Towards Latency-Energy Efficient Task Offloading for AR Services. *IEEE Transactions on Cloud Computing* (2022), 1–1. https://doi.org/10.1109/TCC.2022.3146615

[12] Hongyan Cui, Yang Li, Xiaofei Liu, Nirwan Ansari, and Yunjie Liu. 2017. Cloud service reliability modelling and optimal task scheduling. *Iet Communications* (2017).

[13] The Khang Dang, Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Jörg Ott, and Jussi Kangasharju. 2021. Cloudy with a Chance of Short RTTs: Analyzing Cloud Connectivity in the Internet. In *Proceedings of the 21st ACM Internet Measurement Conference* (Virtual Event) *(IMC '21)*. Association for Computing Machinery, New York, NY, USA, 62–79. https://doi.org/10.1145/3487552.3487854

[14] Aaron Yi Ding, Ella Peltonen, Tobias Meuser, Atakan Aral, Christian Becker, Schahram Dustdar, Thomas Hiessl, Dieter Kranzlmüller, Madhusanka Liyanage, Setareh Maghsudi, Nitinder Mohan, Jörg Ott, Jan S. Rellermeyer, Stefan Schulte, Henning Schulzrinne, Gürkan Solmaz, Sasu Tarkoma, Blesson Varghese, and Lars Wolf. 2022. Roadmap for Edge AI: A Dagstuhl Perspective. *SIGCOMM Comput. Commun. Rev.* 52, 1 (mar 2022), 28–33. https://doi.org/10.1145/3523230.3523235

[15] Eclipse. 2019. fog05 - The End-to-End Compute, Storage and Networking Virtualisation solution. Retrieved May 24, 2022 from https://fog05.io/

[16] Eclipse. 2019. ioFog - Bring your own edge. Retrieved May 24, 2022 from https://iofog.org/

[17] Kubermatic GmbH. 2016. Kubermatic. Retrieved May 24, 2022 from https://www.kubermatic.com/

[18] David Haja, Mark Szalay, Balazs Sonkoly, Gergely Pongracz, and Laszlo Toka. 2019. Sharpening kubernetes for the edge. In *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos*. 136–137.

[19] Minsung Jang, Karsten Schwan, Ketan Bhardwaj, Ada Gavrilovska, and Adhyas Avasthi. 2014. Personal clouds: Sharing and integrating networked resources to enhance end user experiences. In *IEEE INFOCOM 2014*.

[20] Andrew Jeffery, Heidi Howard, and Richard Mortier. 2021. Rearchitecting Kubernetes for the Edge. *4th ACM EdgeSys* (2021).

[21] Peng Liu, Dale Willis, and Suman Banerjee. 2016. ParaDrop: Enabling Lightweight Multi-tenancy at the Network's Extreme Edge. In *IEEE/ACM SEC*.

[22] Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. 2020. Pruning Edge Research with Latency Shears. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks* (Virtual Event, USA) *(HotNets '20)*. Association for Computing Machinery, New York, NY, USA, 182–189. https://doi.org/10.1145/3422604.3425943

[23] Nitinder Mohan and Jussi Kangasharju. 2016. Edge-Fog cloud: A distributed cloud for Internet of Things computations. In *2016 Cloudification of the Internet of Things (CIoT)*. 1–6. https://doi.org/10.1109/CIOT.2016.7872914

[24] Nitinder Mohan and Jussi Kangasharju. 2018. Placing it right!: optimizing energy, processing, and transport in Edge-Fog clouds. *Annals of Telecommunications* 73, 7 (2018), 463–474. https://doi.org/10.1007/s12243-018-0649-0

[25] Nitinder Mohan, Aleksandr Zavodovski, Pengyuan Zhou, and Jussi Kangasharju. 2018. Anveshak: Placing Edge Servers In The Wild. In *Proceedings of the 2018 Workshop on Mobile Edge Communications* (Budapest, Hungary) *(MECOMM'18)*. Association for Computing Machinery, New York, NY, USA, 7–12. https://doi.org/10.1145/3229556.3229560

[26] Jinke Ren, Guanding Yu, Yinghui He, and Geoffrey Ye Li. 2019. Collaborative cloud and edge computing for latency minimization. *IEEE Transactions on Vehicular Technology* 68, 5 (2019), 5031–5044.

[27] Sergej Svorobej, Malika Bendechache, Frank Griesinger, and Jörg Domaschka. 2020. Orchestration from the Cloud to the Edge. *The Cloud-to-Thing Continuum* (2020), 61–77.

[28] Aleksandr Zavodovski, Nitinder Mohan, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. 2018. ICON: Intelligent Container Overlays. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks* (Redmond, WA, USA) *(HotNets '18)*. Association for Computing Machinery, New York, NY, USA, 15–21. https://doi.org/10.1145/3286062.3286065