FedApp: a Research Sandbox for Application Orchestration in Federated Clouds using OpenStack

Johan Ruuskanen
Department of Automatic Control,
Lund University
Lund, Sweden
johan.ruuskanen@control.lth.se

Haorui Peng
Department of Electrical and Information Technology,
Lund University
Lund, Sweden
haorui.peng@eit.lth.se

Alfred Åkesson
Department of Computer Science,
Lund University
Lund, Sweden
alfred.akesson@cs.lth.se

Lars Larsson
Department of Computing Science,
Umeå University
Umeå, Sweden
larsson@cs.umu.se

Maria Kihl
Department of Electrical and Information Technology,
Lund University
Lund, Sweden
maria.kihl@eit.lth.se

ABSTRACT
Multi-cluster federation is envisioned to be the next-generation cloud infrastructure, where it will play a vital part in the realization of concepts such as edge and fog computing. Orchestrating applications in federated environments poses new challenges to well-known research problems in various fields, such as load-balancing, auto-scaling, resource allocation and service migration. However, as access to real multi-cluster infrastructure is limited, a test-bed that provides similar characteristics to a real system is in demand. To enable researchers in associated fields to quickly setup experiments in a federated cloud environment, we have created the open-source sandbox FedApp that simplifies the process of deploying multiple virtual clusters in an OpenStack environment with the possibility of adding realistic network characteristics between sites. Each cluster comes deployed with the open-source and production-grade container orchestrator Kubernetes, complete with federation-wide monitoring using Prometheus/Grafana and simplified inter-cluster microservice communication using Istio.

KEYWORDS
Multi-cluster Federation, Edge Computing, Orchestration, Kubernetes, Sandbox

1 INTRODUCTION
It has long been known that relying on a single cloud provider is not feasible for certain applications and use-cases [10, 21]. Instead, multiple clusters can be used in a joint effort to offer increased performance. This collaboration between a group of autonomous clusters of computational resources is commonly referred to as cluster federation. Together with emerging infrastructure provider alternatives such as edge computing [23, 25], driven in part by the 5G evolution [12], it is increasingly more important to make separate clouds cooperate to jointly provide applications in a more desirable manner. Edge locations are further destined to outnumber the low number of major public cloud providers and regions, thus causing a more pressing need for software to manage underlying systems [5]. To put things into perspective, as of June 2020 Google offers cloud infrastructure in a mere 23 regions. In contrast, edge computing opens the Infrastructure-as-a-Service market to Telco providers, and thus dramatically changes the scale and nature of the infrastructure being offered to customers. In Germany alone, a single Telco provider (Deutsche Telekom) will operate 36,000 base stations by 2021. [15]. Regardless of whether all or “merely” ten percent of these will offer edge computing facilities, it is clear that methodologies and technologies supporting management and use of such infrastructure will require novel research.

Regarding these new federated environments together with recent trends in shifting application architecture from single monolithic implementations to collections of networked microservices, which in itself adds new complexities on cluster management [9, 11], new questions arise in how to automatically orchestrate applications to provide services of desirable performances. As cluster federations are by nature highly dynamic environments, where uncertain cluster-to-cluster connections with potentially non-stationary latency
on top of existing intra-cluster dynamics, application robustness will be an issue. Strategies for inter-cluster load-balancing and routing between services, or auto-scaling of individual services will need to adapt to changing dynamics in real-time to guarantee performance. Further, new problems arise in service scheduling and migration when an entire federation of clusters is considered. Objectives and constraints in end-to-end latency and network bottlenecks might exist and be sporadically violated due to the dynamics.

These types of research problems are commonly tackled using simulation or emulation models [1, 2, 20], which are often quick to set up and play an important part for initial algorithm design and evaluation, but ultimately fail to capture actual system behaviour. To complement this, researchers have the option to perform actual experiments in real environments. However, creating such an environment is in general difficult, expensive and time-consuming, and the steps required to go beyond simulations to explore ideas in real settings quickly becomes complicated when dealing with complex systems such as multi-cluster federation. To the best of our knowledge, there is currently no easy way for academic researchers to cross this gap and access a desired federation of clusters.

An interesting research question then arises in how a research prototype for federated cloud environments can be designed to best support application orchestration research. To this end, we have created the sandbox FedApp with the goal of providing such an environment that (a) is flexible, yet remains a close approximation of real systems, (b) is easy to both deploy and use, and (c) has an easily extendable implementation. Its key features include:

1. creation of a user-defined, multi-cluster virtual environment in OpenStack, with a centralized structure complete with tools for controlling and monitoring the entire federation, streamlining the setup and usage of the sandbox.
2. possibility of inducing network characteristics such as delay, jitter and packet loss between clusters, thus enabling faithful emulation of real-world federations.
3. application orchestration using Kubernetes complete with a multi-cluster service mesh layer using Istio, for easy handling and deployment of federation-wide applications.

The sandbox can be found here

In this section, the design choices for creating the federation environment to fulfill our goal is presented. We first discuss how our emulated federation environment is constructed, later we explain how we can induce realistic network characteristics, and finally we describe how the sandbox simplifies the deployment and management of federation-wide applications. An illustration of the complete setup can be seen in Figure 1 which can be used as an overview of the sandbox in its entirety.

2 SANDBOX DESIGN

In this section, the design choices for creating the federation environment to fulfill our goal is presented. We first discuss how our emulated federation environment is constructed, later we explain how we can induce realistic network characteristics, and finally we describe how the sandbox simplifies the deployment and management of federation-wide applications. An illustration of the complete setup can be seen in Figure 1 which can be used as an overview of the sandbox in its entirety.

2.1 Topology of the emulated federation

The sandbox will need some type of infrastructure to be deployed upon. To provide this in a both flexible and easy to use manner, we chose to supply the means of creating a suitable virtual infrastructure in OpenStack. OpenStack is an open-source software platform for providing common IaaS (Infrastructure-as-a-Service) on physical hardware, and is commonly used to operate private clouds. This platform provides both the means of creating a controlled environment in which the rest of the sandbox can be deployed in a standardized manner, and makes it flexible as there is a standardized way of scaling the size and amount of clusters. To emulate a federated cloud setup, each cluster is first represented and provisioned in OpenStack as a set of virtual machines (VM) with their own isolated internal network. To further provide a close approximation to a real cluster federation, it is vital that users are able to impose desired network characteristics between clusters. To enable this we let each internal cluster network be connected to a gateway virtual machine, which allows us to use it as a single point handler of routing and network emulation between clusters.

---

1https://github.com/JohanRuuskanen/FedApp

---

Figure 1: Illustration of the complete sandbox with 3 clusters containing 3 virtual machines each.
Given the stated goal of this project, centralizing the network through a VM in this manner is beneficial. First, it enables researchers to affect network characteristics and observe network behavior such as traffic patterns and workload classifications from a single point. Moreover, a centralized VM increases usability as it can be shipped with all the tools necessary for controlling, monitoring and deploying applications on the clusters. Finally, the choice of pooling all functionalities for commandeering the emulated federation makes the setup easy to extend, and easy to debug when something eventually breaks down.

### 2.2 Network characteristics emulation

Benefiting from the chosen centralized topology, the procedure to mimic a real-world networking environment is greatly simplified. Each cluster sees itself as a "stand-alone" cluster on a private network, where all the inter-cluster communication traffic is handled by the Gateway VM. Thus by only affecting the inter-cluster routing logic on the gateway, an arbitrary inter-cluster network profile could be emulated.

We achieve this by fully utilizing the Linux Traffic Control (TC) utility on the gateway. The TC network emulator (TC-netem) provides the possibility of adding packet loss, delay and other characteristics on the packets from a selected Network Interface Controller (NIC). By applying desired emulated characteristics on the NICs of the Gateway VM, the clusters are not aware of the network status in beforehand. Thereby, the network characteristic to a cluster can be specified arbitrarily, for example mimic the propagation and transmission delay caused by the geographic distance. Further, by deploying TC classful queuing discipline (qdisc) and filter, it is possible to control the point-to-point network characteristics between clusters. Each class can then be given its individual configuration for a desired cluster-to-cluster network characteristic without affecting important meta-communication between the Gateway VM and the clusters, such as monitoring data or control commands.

Using TC-netem, it is then possible to both emulate real-world network characteristics, and to test some customized scenarios, by applying delay and loss along with their distributions to the network connections between any desired pair of clusters.

### 2.3 Managing federation-wide applications

Cloud applications are increasingly being packaged and deployed as collections of networked services using containerization technologies. To allow such applications to be efficiently managed, the sandbox fits each cluster with the well-known container orchestrator Kubernetes, which creates an abstraction layer between the raw infrastructure and applications. This provides both a vital platform for building, deploying and handling federation-wide applications in a standardized manner, and allows researchers to experiment in a popular and well-maintained open-source setting. To enable non-trivial collaboration in service-graphs spanning multiple clusters, the sandbox deploys Istio on top of Kubernetes, which greatly simplifies handling inter-cluster service-to-service networking. The sandbox further supplies kubectl at the Gateway VM, enabling full control of the Kubernetes clusters via a CLI. Finally, to simplify the management of more advanced deployments, the Gateway VM is supplied with the Kubernetes package manager Helm.

To provide insight on the behavior of the emulated federation and its deployed applications, the sandbox supplies federation-wide monitoring in the form of data-collection using Prometheus, and a Grafana GUI deployed on the Gateway VM. With this setup, it is possible to inspect important metrics such as CPU and memory utilization and traffic patterns. Further, the Istio service mesh makes it possible to retrieve the timestamp of every request entering/exiting any microservice without tampering with the code of the service. It is also possible to collect tracing information of all requests, by performing the minimal code change of forwarding http headers in each service.

As we set out to make the sandbox flexible, easy to use and extendable, its deployment is heavily scripted and requires little input from the user to create the multiple Kubernetes clusters with Istio and monitoring as showcased in Figure 1. To setup, the only things needed are access to an OpenStack cluster and a local computer with Terraform and Ansible installed. The deployment itself further capitalizes on the natural modularity between the different tools, which makes it easy to change or debug one part of the deployment without affecting the other parts of the sandbox.

### 3 FUNCTIONALITIES

The sandbox was created with the goal of providing a federated cloud environment for application orchestration research. In this section we discuss its functionalities that could specifically contribute to this point. Via our design, researchers can set up an user-defined federated Kubernetes environment, with a single centralized point for collecting all tools and scripts needed for researchers to deploy, monitor and control federation-wide applications.

The decision to provision the infrastructure with OpenStack together with TC-netem for network emulation enables researchers to experiment in a wide range of federated cloud or fog computing settings. Further, the usage of Kubernetes and Istio makes it possible to easily deploy and

---

2https://www.linux.org/docs/man8/tc-netem.html
control federation-wide applications in these settings. Finally, together with the Istio service mesh, the inclusion of Prometheus makes it possible to effortlessly monitor both the resource utilization for every service and VM, and further also the timestamp and tracing for every service. Because of the centralized structure and our emphasis on extendability, new tools or software that are not supplied out-of-the-box can easily be incorporated into the sandbox.

The possible use cases of FedApp is numerous, and many of the open issues in cloud computing as discussed in [3] could readily be explored in it. Below follows some key research problems that can directly take advantage of experimental investigation and validation in FedApp.

1. **Resource management:** Auto-scaling and load-balancing in large scale services and multi-cluster settings is an open issue. As both Kubernetes and Istio natively supports these methods, the sandbox gives a great entrypoint for researchers to test their own solutions.

2. **Scheduling:** Where to schedule what microservice in a service graph, the effects of multi-tenancy and over-provisioning, and how to tackle migration problems in a dynamic multi-cloud setting can readily be investigated using the available control and monitoring solutions in the sandbox.

3. **Reliability:** The sandbox provides a natural platform for developing detection and recovering solutions for single-point fail-over in federated cloud environments. For example, TC-netem can be used to impose a connection drop to a cluster without affecting the meta-communication between the Gateway VM. Further, the fine grained tracing information makes it possible to evaluate performance models of e.g. the quality-of-service in applications.

4. **Networking:** Analysis of network traffic in large scale distributed systems can readily be performed at the Gateway VM, as it handles all cluster-to-cluster communication. Further, TC-netem offers the opportunity to customize various network characterizes to evaluate systems and applications under different types of networking scenarios.

5. **Security:** The sandbox supports the study and analysis of security and privacy on cloud applications in a federated setting. For example, Istio provides a natural way of imposing encryption on service-to-service communication.

### 4 PROOF-OF-CONCEPT

To showcase some of the functionalities and to demonstrate how the sandbox can be used, we in this section present a fog computing scenario with a federated example application deployed on top of the sandbox. In the scenario we will consider a face detection application that in order to save bandwidth, has had its image pre-processing off-loaded to two smaller edge clusters $C_1, C_2$ from the actual classification at two larger data centers $C_3, C_4$.

#### 4.1 Deploying the scenario

Without loss of generality, each virtual edge cluster and data center were given the same size of four VMs each with four vCPUs and 8 Gbytes of RAM. The Gateway VM was further deployed with four vCPUs and 16 Gbytes of RAM. All of the VMs were given Ubuntu 18.04 as the instance image. In order to emulate a mock geographic distance, a delay was introduced using TC-netem between the edge clusters $C_1, C_2$ and the backend cluster $C_3$ with the following one-way delay matrix.

$$D = \begin{pmatrix}
0 & 0 & 25 & 0 \\
0 & 0 & 25 & 0 \\
25 & 25 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix} \text{ms}$$

where $D_{ij}$ gives the one way delay between clusters $C_i$ and $C_j$. This means that the roundtrip-time (RTT) between $C_1, C_2$ and $C_3$ should be 50 ms.

For the example application, frontend image pre-processing and backend classification where delivered in two standalone microservices. The frontend microservice reads a RGB image from an API call and reduces it to a grayscale image. The smaller grayscale image is sent to the backend microservice, where the face detection is performed using a standard Haar-cascade classifier in OpenCV. If a face is detected, the corners of the bounding box is returned to the frontend, which adds the box to the original RGB image and sends it back as a response to the original API call. The frontend microservices were then deployed with two replicas on each edge cluster, while the backend microservices used three replicas on each data center. Further, in order to balance requests between the multiple edge clusters, an NGINX load balancer was launched in a docker container on the Gateway VM. An overview of the scenario setup can be seen in Figure 2.

In order to put any real stress on the system, a load generator was further created which sends images at a given rate to the NGINX load balancer at the Gateway VM. For this demonstrative scenario, the necessary images where provided by the UMass face detection data set and benchmark[14].

#### 4.2 Experimental evaluation

To see whether the system behaves as expected, a handful of experiments were performed on the example scenario. Initially, we examined how well TC-netem managed to create our desired network characteristics. This was done by measuring the RTT by running ping between every cluster...
simultaneously with an inter-sample time of 0.2 s over 600 s. We then examined if the cluster software stacks managed to properly handle the requests despite various disturbances, by observing the application vCPU usage at the different clusters when putting stress on the system. Thanks to the federation-wide monitoring, metrics such as vCPU for all clusters is easily observed in Grafana.

The impact of two different inter-cluster load balancers in Istio, Round Robin (RR) and Least Connection (LC), under different loads was first examined. The RR load balancer distributes the requests evenly amongst the backends, which makes it largely unaffected by delay, while LC on the other hand prioritizes sending requests to the backend which has the least amount of active requests, making it sensitive to our added delay. Images were sent at both 10 images/s and 20 images/s for a time of 600 seconds per rate. If all is working properly, we would expect to see a vCPU usage lower for the edge clusters $C_1, C_2$ than for the data centers $C_3, C_4$ as these perform a more computationally heavy task. We further expect to see a doubling of vCPU usage when the load generation rate is doubled. Further, we expect to see roughly equal vCPU usage on $C_3, C_4$ using the RR policy, while using LC policy would prioritize $C_4$.

We then examined how well the emulation could reproduce sudden cluster link failure. The introduced delay was focused on bridging technological gaps between competing cloud infrastructure providers with possibly heterogeneous underlying technology stacks [21]. The vision of inter-cloud compatibility was embodied by shared or compatible APIs on the infrastructure level [7, 8, 19]. With more than a decade of hindsight, it is clear that these early attempts failed to convincingly demand industry adoption among the major cloud providers.

Recent work shows that a Kubernetes-based edge deployment is promising for real-life applications, and offers improvements with regard to e.g. latency [6, 13, 24]. In addition to reying on Kubernetes to provide a common platform abstraction, cloud-native applications can leverage service meshes such as Istio to simplify networking code [22].

Compared to other approaches that aims to enable research for cloud systems, the sandbox differentiates itself in a number of ways. Its flexibility and openness puts it in stark contrast to other conceptually similar approaches, such as [16]. Further, in regards to the numerous Fog/Edge emulators, see Section 4.14.1 in [26], cloud simulators such as CloudSim [4], or direct modeling of Kubernetes-based platforms, such as [17, 18], the sandbox enables researchers to utilize the entire cluster software stack. We believe that due

4.2.1 Results. For the RTT experiment, the resulting means and variances of the measured RTT for every link was

$$E[2D] \approx \begin{bmatrix} 0.05 & 1.48 & 51.7 & 1.56 \\ 1.46 & 0.04 & 51.8 & 1.46 \\ 51.7 & 51.8 & 0.05 & 1.59 \\ 1.47 & 1.45 & 1.58 & 0.03 \end{bmatrix} \text{ms}$$

As can be seen, the results corresponds well to the desired values. A small bias can be observed due to the internal OpenStack network. For our load generation experiments, the results can be seen in Figure 3 and Figure 4. For all graphs, the results looks as expected indicating that the sandbox works as intended. The slight difference in RR CPU usage could be contributed to the underlying VM allocations. As the measurement constitutes the rate of change in container CPU usage in seconds, a faster clock speed would lead to lower CPU usage. We can however be fairly certain that the RR difference is not due to the added delay, as the results in Figure 4 which has no delay experiences similar differences as the results displayed in Figure 3a. Further, the apparent noisiness could be attributed to the different sizes of images in the dataset in conjunction with the inherent uncertainty in execution time on the cloud. To let researchers try out this example by themselves, and to demo how a federation-wide application can be implemented and deployed in FedApp, the example application is included in the sandbox.

5 RELATED WORK

In older literature, the matter of multi-cluster collaboration was focused on bridging technological gaps between competing cloud infrastructure providers with possibly heterogeneous underlying technology stacks [21]. The vision of inter-cloud compatibility was embodied by shared or compatible APIs on the infrastructure level [7, 8, 19]. With more than a decade of hindsight, it is clear that these early attempts failed to convincingly demand industry adoption among the major cloud providers.

Figure 2: Emulated edge clusters and data centers hosting the two frontend and three backend microservices for the example federated face detection application.
to the rapid pace of development in cloud orchestration tools such as Kubernetes and Istio, a practical sandbox that lets researchers deploy the actual software and not models thereof, have a greater impact and thus provide more valuable future insights.

6 DISCUSSION

The way the emulated federation in our sandbox is constructed is motivated by our focus to support research in application orchestration. It is worth mentioning that our implementation choices are not the only way to create such an environment, especially with another focus in mind a different approach with different tools and topologies might be more suitable. For example, if one desires to test e.g. application scaling performance over a very large amount of clusters, the Gateway VM will be a potential issue.

This is because from a scalability perspective, the centralized connection constitutes a bottleneck. But scalability to support hundreds or more clusters is not part of our goal, and interesting problems to study as discussed in Section 3 does not need such large realizations. Moreover, the current implementation seems to have a maximum number of possible network interfaces for an OpenStack VM that would need to be circumvented for such a scenario. On our specific OpenStack instance we managed to allocate up to 22 clusters before running into this interface limit on our Gateway VM. A different OpenStack implementation or instance image might have a different interface limit. For these 22 clusters, we however experienced only negligible effects on network performance in a low traffic scenario, which hints that the sandbox can handle much larger experiments than our demonstrative scenario in Section 4. Ultimately, it would have been a nice feature to support an even larger amount of clusters but it is simply not the type of research that we have aimed to support in the current iteration of our work.

Besides our aims, there are other use-cases which the sandbox could support, or easily extended to support in its current form. For example, FedApp has the potential to
deliver an environment for both end-to-end and performance testing of specific multi-cluster software and tools under different scenarios. Kubernetes own federated deployment, KubeFed which is currently in its alpha comes to mind.

Looking forward, apart from our simple example it would be of benefit to out-of-the-box include other open source benchmark applications such as the ones introduced by Gan et al. [11]. Further, implementing and deploying e.g. auto-scaling algorithms in Kubernetes is not a trivial matter for the inexperienced researcher. A benefit to usability would be to streamline this process as well, either via providing concrete examples or by some toolkit.

7 CONCLUSION

We have presented FedApp, a tool for providing an emulated cloud federation environment in OpenStack with user-defined network characteristics between clusters. The goal has been to give researchers a way to quickly set up federated environments for experimentation. Upon publication, the sandbox will be fully available as open source.

ACKNOWLEDGMENTS

This work is partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. Also, the work in this paper is partially supported by the SEC4FACTORY project, funded by the Swedish Foundation for Strategic Research (SSF), and the 5G-PERFECTA Celtic Next project funded by Sweden’s Innovation Agency (VINNOVA). Johan Ruuskanen & Maria Kihl is part of the Excellence Center at Linköping-Lund on Information Technology (ELLIIT).

REFERENCES

[1] David Perez Abreu, Karima Velasquez, Marilia Curado, and Edmundo Monteiro. 2019. A comparative analysis of simulators for the Cloud to Fog continuum. Simulation Modelling Practice and Theory (Nov. 2019), 102029. https://doi.org/10.1016/j.simpat.2019.102029

[2] A. Ahmed and A. S. Sabyasachi. 2014. Cloud computing simulators: A detailed survey and future direction. In 2014 IEEE International Advance Computing Conference (IACC). 866–872. https://doi.org/10.1109/IADCC.2014.6779436

[3] Rajkumar Buuya, Satish Narayana Sirama, Giuliano Casale, Rodrigo Calheiros, Yogesh Simhnan, Blesson Varghese, Erol Gelenbe, Bahman Javadi, Luis Miguel Vaquero, Marco A. S. Netto, Adel Nadjaran Toosi, Maria Alejandra Rodriguez, Ignacio M. Llorente, Sabrina De Capitani Di Vimercati, Pierangela Samarati, Dejan Milojicic, Carlos Varela, Rami Bahsoon, Marcos Dias De Assuncao, Omer Rana, Wanlei Zhou, Hai Jin, Wolfgang Gentzsch, Albert Y. Zomaya, and Haiying Shen. 2018. A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade. ACM Comput. Surv. 51, 5, Article 105 (Nov. 2018), 38 pages. https://doi.org/10.1145/3241737

[4] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buuya. 2010. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience 41, 1 (Aug. 2010), 23–50. https://doi.org/10.1002/spe.995

[5] Ronan-Alexandre Cherrueau, Adrien Lebre, Dimitri Pertin, Fetha Wuhib, and João Monteiro Soares. 2018. Edge Computing Resource Management System: a Critical Building Block! Initiating the debate via OpenStack. In USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18). USENIX Association, Boston, MA. https://www.usenix.org/conference/hotedge18/presentation/cherrueau

[6] Franco Cicirelli, Antonio Guarneri, Giandomenico Spezzano, and Andrea Vinci. 2017. An edge-based platform for dynamic Smart City applications. Future Generation Computer Systems 76 (2017), 106 – 118. https://doi.org/10.1016/j.future.2017.05.034

[7] Andy Edmonds, Thijs Metsch, Alexander Papaspyrou, and Alexis Richardson. 2012. Toward an open cloud standard. IEEE Internet Computing 16, 4 (2012), 15–25.

[8] E. Elmroth and L. Larsson. 2009. Interfaces for Placement, Migration, and Monitoring of Virtual Machines in Federated Clouds. In 2009 Eighth International Conference on Grid and Cooperative Computing. 253–260. https://doi.org/10.1109/GCC.2009.36

[9] M. Fazio, A. Celesti, R. Ranjan, C. Liu, L. Chen, and M. Villari. 2016. Open Issues in Scheduling Microservices in the Cloud. IEEE Cloud Computing 3, 5 (Sep. 2016), 81–88. https://doi.org/10.1109/MCC.2016.112

[10] Ana Juan Ferrer, Francisco Hernández, Johan Tordsson, Erik Elmroth, Ahmed Ali-Eldin, Csilla Zsagri, Raul Sirvent, Jordi Guitart, Rosa M. Badia, Karim Djemame, Wolfgang Ziegler, Theo Dimitrakos, Srijith K. Nair, George Kousiouris, Kleopatra Konstanteli, Theodora Varvarigou, Benoit Hudzia, Alexander Kipp, Stefan Wesner, Marcelo Corrales, NikolausForgó, Tabassum Sharif, and Craig Sheridan. 2012. OPTIMIS: A holistic approach to cloud service provisioning. Future Generation Computer Systems 28, 1 (2012), 66 – 77. https://doi.org/10.1016/j.future.2011.05.022

[11] Yu Gan, Yanqi Zhang, Da llin Cheng, Aniktha Shetty, Priyai Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Rithchen, Brendon Jackson, and et al. 2019. An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPL ’19). Association for Computing Machinery, New York, NY, USA, 3–18. https://doi.org/10.1145/3297858.3304013

[12] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. 2015. Mobile edge computing—A key technology towards 5G. ETSI white paper 11, 1 (2015), 1–16.

[13] Yuzhou Huang, Kaiyu cai, Ran Zong, and Yugang Mao. 2019. Design and Implementation of an Edge Computing Platform Architecture Using Docker and Kubernetes for Machine Learning. In Proceedings of the 3rd International Conference on High Performance Compilation, Computing and Communications (HPCC’19). Association for Computing Machinery, New York, NY, USA, 29–32. https://doi.org/10.1145/3318265.3318288

[14] Vaidit Jain and Erik Learned-Miller. 2010. FDDB: A Benchmark for Face Detection in Unconstrained Settings. Technical Report UM-CS-2010-009. University of Massachusetts, Amherst.

[15] Pascal Kiel-Koslowski. [n. d.]. Deutsche Telekom commission over 300 new LTE mobile base stations. (n. d.), https://www.telekom.com/en/media/media-information/archive/300-new-lte-mobile-base-stations-574106

[16] Endah Kristiani, Chao-Tung Yang, Yuan Ting Wang, and Chin-Yin Huang. 2019. Implementation of an Edge Computing Architecture Using OpenStack and Kubernetesp. In Information Science and Applications 2018, Kuinam J. Kim and Nakhoon Baek (Eds.). Springer Singapore, Singapore, 675–685.
[17] V. Medel, O. Rana, J. Á. Bañares, and U. Arronategui. 2016. Adaptive Application Scheduling under Interference in Kubernetes. In 2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC). 426–427.

[18] Víctor Medel, Rafael Tolosana-Calasanz, José Ángel Bañares, Unai Arronategui, and Omer F. Rana. 2018. Characterising resource management performance in Kubernetes. Computers & Electrical Engineering 68 (2018), 286 – 297. https://doi.org/10.1016/j.compeleceng.2018.03.041

[19] Boris Parák, Zdenek Sustr, Florian Feldhaus, Piotr Kasprzak, and Maik Srba. 2014. The rOCCI project: providing cloud interoperability with OCCI 1.1. In International Symposium on Grids and Clouds (ISGC) 2014, Vol. 210. SISSA Medialab, 014.

[20] B. Ramprasad, M. Fokaefs, J. Mukherjee, and M. Litoiu. 2019. EMU-IoT - A Virtual Internet of Things Lab. In 2019 IEEE International Conference on Autonomic Computing (ICAC). 73–83.

[21] B. Rochwerger, A. Galis, E. Levy, J. A. Caceres, D. Breitgand, Y. Wolfthal, I. M. Llorente, M. Wusthoff, R. S. Montero, and E. Elmeroth. 2009. RESERVOIR: Management technologies and requirements for next generation Service Oriented Infrastructures. In 2009 IFIP/IEEE International Symposium on Integrated Network Management. https://doi.org/10.1109/INM.2009.5188828

[22] Ozair Sheikh, Serjik Dikaleh, Dharmesh Mistry, Darren Pape, and Chris Felix. 2018. Modernize Digital Applications with Microservices Management Using the Istio Service Mesh. In Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering (ICCSW '18). IBM Corp., USA, 359–360.

[23] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge computing: Vision and challenges. IEEE internet of things journal 3, 5 (2016), 637–646.

[24] P. Tsai, H. Hong, A. Cheng, and C. Hsu. 2017. Distributed analytics in fog computing platforms using tensorflow and kubernetes. In 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS). 145–150. https://doi.org/10.1109/APNOMS.2017.8094194

[25] M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan. 2016. Osmotic Computing: A New Paradigm for Edge/Cloud Integration. IEEE Cloud Computing 3, 6 (Nov 2016), 76–83. https://doi.org/10.1109/MCC.2016.124

[26] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlaliji, Jian Kong, and Jason P. Jue. 2019. All one needs to know about fog computing and related edge computing paradigms: A complete survey. Journal of Systems Architecture 98 (Sept. 2019), 289–330. https://doi.org/10.1016/j.sysarc.2019.02.009