A Research Perspective on Fog Computing

David Bermbach[1(✉)], Frank Pallas[1], David García Pérez[2], Pierluigi Plebani[3], Maya Anderson[4], Ronen Kat[4], and Stefan Tai[1]

[1] TU Berlin, Information Systems Engineering Research Group, Berlin, Germany
{db,fp,st}@ise.tu-berlin.de
[2] Atos Spain SA, Atos Research and Innovation, Barcelona, Spain
david.garciaperez@atos.net
[3] Politecnico di Milano, Milan, Italy
pierluigi.plebani@polimi.it
[4] IBM Research Haifa, Haifa, Israel
{mayaa,ronenkat}@il.ibm.com

Abstract. State-of-the-art applications are typically deployed on top of cloud services which offer the illusion of infinite resources, elastic scalability, and a simple pay-per-use billing model. While this is very convenient for developers, it also comes with relatively high access latency for end users. Future application domains such as the Internet of Things, autonomous driving, or future 5G mobile apps, however, require low latency access which is typically achieved by moving computation towards the edge of the network. This natural extension of the cloud towards the edge is typically referred to as Fog Computing and has lately found a lot of attention. However, Fog Computing as a deployment platform has not yet found widespread adoption; this, we believe, could be helped through a consistent use of the service-oriented computing paradigm for fog infrastructure services. Based on this motivation, this paper describes the concept of Fog Computing in detail, discusses the main obstacles for Fog Computing adoption, and derives open research challenges.

Keywords: Fog Computing · Cloud computing · Edge computing

1 Introduction

Today's state-of-the-art applications are typically deployed on top of cloud services, thus, leveraging cost benefits, ease-of-use, elastic scalability, and the illusion of infinite resources [12]. While cloud services come with these obvious benefits, they also have a major disadvantage: cloud data centers are centralized and, thus, typically far from the end user resulting in high access latencies. This is sufficient for many application domains such as enterprise or web applications but not for more modern application domains such as autonomous driving, Internet of Things (IoT)-based platforms, or 5G mobile applications. Therefore, these

L. Braubach et al. (Eds.): ICSOC Workshops 2017, LNCS 10797, pp. 198–210, 2018.
https://doi.org/10.1007/978-3-319-91764-1_16

applications are typically deployed on edge devices. However, while such edge devices offer low access latencies due to their physical proximity to end users, they also come with limited resources and are subject to availability problems.

In this situation, an obvious approach is to use both cloud services and edge nodes at the same time to achieve low latency while having access to scalable, infinite resources. This paradigm, which has recently emerged as a natural extension of Cloud Computing, is typically referred to as Fog Computing [4,13]. Beyond the benefits already discussed, Fog Computing also promises better privacy to end users: While current, cloud applications collect personal data need in a central place, future Fog applications can keep detailed personal data at the edge, transferring only aggregated or properly anonymized data to the cloud.

These obvious advantages of Fog Computing and the public attention it created notwithstanding, there has so far been surprisingly little adoption of the paradigm for actual applications. We believe that there are several reasons for this, chief among which we see a lack of edge service offerings: managing edge nodes manually is a gruesome task and requires massive upfront investments – the exact opposite of cloud services with their convenience focus and pay-as-you-go model. Also, it is still not completely clear what Fog Computing actually is.

This paper aims to shed some light on Fog Computing and its role in service-oriented computing (SOC). To this aim, we firstly provide a definition of Fog Computing and the involved concepts relevant to SOC. Secondly, we identify the main obstacles for widespread fog adoption and describe a number of fundamental research challenges. For this reason, this paper should, thus, be seen as a call to action and as an overview of challenges that we as researchers should tackle.

This paper is structured as follows: In Sect. 2, we give an overview of state-of-the-art Fog Computing (along with its competing definitions as well as its benefits and opportunities). Afterwards, in Sect. 3, we identify and discuss obstacles for widespread fog adoption. In Sect. 4, we describe research challenges that result from these obstacles before concluding.

2 From Cloud to Fog Computing

Fog computing has been initially introduced in the telecommunication sector [4] when researchers and practitioners realized how the role of the final users changed from *consumers* of information to *prosumers* (producers and consumers at the same time). In fact, the original paradigm on which the Web is based assumes that the core of the network is in charge of providing information that will be consumed at the edge. Prosumers with mobile devices or IoT sensors, however, generate immense data quantities at the edge of the network.

So, what precisely is Fog Computing and how can it be distinguished from Edge Computing? Edge Computing is exclusively about computation at the edge of the network without any notion of cloud services. Depending on the source, Fog Computing is either the same as Edge Computing [20] or is defined as the amalgam of cloud, edge, and any intermediary nodes in between (this could be

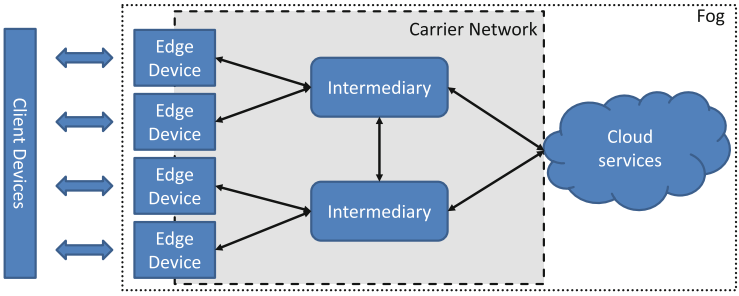

Fig. 1. Deployment overview of Fog Computing

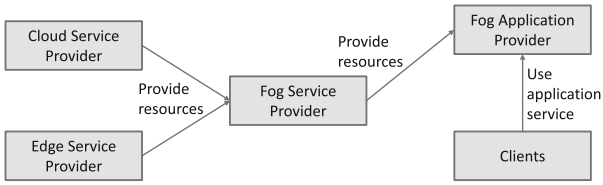

Fig. 2. Overview of roles in Fog Computing

small- to medium-sized data centers within the core network of the network provider) [21]. In this paper, we take the latter perspective. See also Fig. 1 for a high-level deployment overview of Fog Computing.

To realize its full potential, Fog Computing must be more than creating a *data center in the box*, i.e., Cloudlets [19], to bring the cloud closer to data producers. Instead, Fog Computing must be seen as a "resource layer that fits between the edge devices and the cloud data centers, with features that may resemble either." [25]. As also pointed out by the OpenFog Consortium [13], the goal of Fog Computing is to provide a set of methods and tools to create a continuum between edge and cloud. For this, Fog Computing technologies especially need to enable fluid data movement between cloud services and edge, in both directions, while satisfying application constraints in terms of quality of service (QoS). Such data movement may also be accompanied by movement of computation – both in the same way and in a complimentary fashion compared to data movement.

With the advent of Fog Computing, applications based on this paradigm can exploit the advantages of both the edge and cloud environments. With specifically tailored frameworks, e.g., [17], developers can leave it to the Fog Computing layer to automatically handle data and computation placement. Based on such dynamic data and computation movement, applications can provide new functionality or new QoS levels (e.g., significantly lower latency) and use novel ways for dynamically balancing tradeoffs. The lack of such capabilities has so far hindered the emergence of technologies such as autonomous and interconnected driving, large-scale IoT, etc., which should significantly profit from fog adoption.

While Cloud Computing is largely centered around a service concept, this is not (yet) the case for the edge as we will also discuss in the next sections. However, we firmly believe that the SOC paradigm might prove valuable for widespread fog adoption. In this context, we envision the following roles (see also Fig. 2): Already today, cloud service providers offer access to cloud infrastructure services. Edge service providers may do the same for edge resources. Fog service providers, in contrast, offer service-based access to fog infrastructure resources – in this capacity, fog service providers may act as resellers. Alternatively, two or more of these roles may be held by the same organization. Finally, Fog application providers rent resources through fog infrastructure services to offer their application to clients. We explicitly use the broad term "client" as this may include IoT devices, mobile phones, or any other end user.

However, there are several obstacles and yet unsolved research challenges that need to be faced in order to actually pave the way for a broad adoption of Fog Computing. In literature, some other papers already made the attempt to identify these challenges. For instance, even if it does not mention Fog Computing, Díaz et al. [6] analyses the challenges when considering the integration of IoT and cloud services, thus, in a scenario close to Fog Computing. Yet, [5] proposes a Fog Computing definition close to ours and also identifies some possible applications as well as research challenges mainly concerning the infrastructural level, while in this paper we are taking a broader perspective also considering the application level. Finally, in [20], even if mainly focused on the Edge Computing, a list of challenges are introduced with more emphasis on the programming models.

As already mentioned, the goal of the paper is to consider Fog Computing as an environment in which both cloud and edge resources are used as application service deployment environments. Obstacles to the adoption of Fog Computing, and the related research challenges arising when trying to overcome to those obstacles, necessarily include the design, deployment, and management of complex and distributed systems, as discussed in the following sections.

3 The Main Obstacles for Adoption of Fog Computing

Broadly, obstacles for a wide adoption of Fog Computing can be categorized as *inherent*, e.g., available technology or physical constraints, and *external* obstacles, e.g., privacy legislation. In this section, we will give an overview of both.

3.1 Inherent Obstacles

As inherent obstacles we see those that result from the very idea of using fog resources. Specifically, these can be technical constraints such as a given limit of computational power, logical constraints such as having tradeoffs in distributed systems, or simply market constraints such as the fact that there are currently no managed edge services.

O1: No Edge Services. While the cloud is conveniently usable with its service-based consumption model, there are currently no edge infrastructure services where compute or storage capacity could be provisioned on-demand. We believe that managed edge services are bound to enter the market sooner or later; possibly even as fog services in cooperation of cloud providers and network carriers. Currently, Amazon is taking first steps in that direction through their Greengrass "service"[1] which provides software for edge devices. However, since there is currently no way to really "rent" on-demand edge capacity, this means that fog application providers currently need to build, manage, and run their own physical edge "boxes" including cloud integration.

O2: Lack of Standardized Hardware. While there are some ready-to-use, off-the-shelf edge "boxes", these come in a variety of flavors, e.g., regarding compute power. Alternatively, Raspberry Pis, BeagleBoards, or custom solutions can be used. This leads to a broad heterogeneity of edge node hardware which has two effects: First, software stacks need to be adapted or may not even run everywhere. Systems that can actually run everywhere are unlikely to fully tap the potential of the respective hardware resources. Second, application architectures need to be able to deal with such diverse resource capacities. This requires highly modularized applications that can deliver some or all service features depending on the nodes where they are running.

O3: Management Effort. Since Fog Computing is mainly about bringing data and computation closer to end users, dense fog deployments will result in very high numbers of fog nodes. These all need to be managed, e.g., when scaling, installing updates, or updating configurations. As there is currently no managed fog infrastructure service, this management effort is left with application providers. In comparison small on-premise data centers, this requires immense effort.

O4: Managing QoS. Complexity of systems typically increases with the number of nodes in the system and their geographical distribution. This is due to issues and faults – ranging from simple ones such as network latencies, message reordering, or message loss to complicated faults such as network partitioning or byzantine failures. Usually, systems can cope with these issues but they still affect QoS and their tradeoffs. In large-scale geo-distributed systems, all these issues become more pronounced and faults happen more often – simply based on probabilities and the increased number of distributed nodes.

At the same time, fog application domains such as IoT or autonomous driving actually have *stronger* quality requirements on infrastructure services as the potential impact of "things going wrong" is much more severe and affects the physical world. E.g., faults in a social network may lose a private message;

[1] aws.amazon.com/greengrass.

identical faults could result in injuries and deaths if they affect safety-critical functionality of autonomous cars.

O5: No Network-Transparency. Distributed systems research has always tried to hide distribution, i.e., that applications will run as if they were running on a single machine. Over the years, more and more of these transparencies needed to be sacrificed in favor of other QoS goals. What has still remained hidden so far is the network layout; e.g, cloud services expose some basic high-level information on their setup and distribution (e.g., availability zones and regions in AWS) and heavily rely on network virtualization. However, in Fog Computing, the interconnection of nodes on a logical level can no longer be done in arbitrary ways as the underlying networks are, in fact, more or less hierarchical; e.g., two geographically near edge nodes may be connected directly, on the next higher hierarchy level, at the Internet backbone level or anywhere in between. In geo-distribution, this results in edge-to-edge latencies that are magnitudes apart. Therefore, fog application services can no longer be based on high-level abstractions describing distance notions (e.g., an AWS region). Instead, they need to be aware of actual network topologies unless they can cope with unexplainable performance variance across nodes. For manageability reasons, this calls for the introduction of novel, more fine-grained topology abstractions that can be handled inside application services.

3.2 External Obstacles

Beyond the inherent obstacles already discussed, there are also influence factors resulting from external entities such as government agencies or attackers.

O6: Physical Security. Traditional data centers can employ arbitrarily complex measures for physical access control including but not limited to onsite security staff. Typically, this is subject to audit and certification to assert that their physical security measures are up to par. For fog nodes widely distributed across a basically hostile environment, this is impossible. Here, physical security may mean to attach an edge box to the top of a street light's pole (instead of at eye level) or surrounding it with a fire-resistant coating to counter vandalism. What is less obvious, though, is that potential attackers gain a physical attack vector into a software system which enables an entire set of so far theoretic attacks. Even though there are existing approaches to comparable problems from other fields – e.g., for tamper-proofing smart energy meters or for protecting modern cars against software manipulations – these are typically addressing highly specific risks that are well-known in advance. Also, such approaches have only been used in specialized embedded systems yet. They can, thus, not be directly applied to heterogeneous, multi-purpose fog nodes. Nonetheless, they might provide valuable inspiration and direction for addressing physical security in Fog Computing; e.g, one could design fog nodes so that they can detect tampering to then shut down the node, move and wipe all data, and notify the authorities. Application services, in turn, need to be able to cope with such behavior.

O7: Legal and Regulatory Requirements. In many application domains, e.g., in health care, there are legal and regulatory requirements that mandate data to be held in certain physical locations. Also, storage and processing facilities must demonstrably meet certain requirements. While cloud data centers can be certified if they exist in the respective region, this seems impractical for edge nodes. Furthermore, data privacy regulation such as the EU's GDPR [9] also include aspects like transparency, i.e., the data subjects' right to know where their personal data is stored. In liquid fog-based applications, this is challenging to prove as the application may not even know the data location due to virtualization.

4 Open Research Challenges in Fog Computing

While we as a research community cannot start a managed edge infrastructure service, we can support fog adoption through research efforts on the research challenges presented in this section. Even though there are interdependencies between these challenges, they can be broadly categorized regarding the role that they mainly affect: fog infrastructure service providers or fog application providers. In a third group are management challenges such as failure handling or security and privacy aspects which fog application providers are primarily but not exclusively responsible for. In this section, we will present this non-conclusive list of main challenges following the above-mentioned ordering structure; where possible we also point out possible avenues for addressing them.

4.1 Fog Service Providers

While our research is unlikely to directly result in fog infrastructure services, there are still a number of open research challenges that potential fog service providers need to face once they decide to enter the market.

RC1: New Abstractions. One of the main goals of using a SOC model is to hide implementation or runtime details. In the case of fog resources, geo-distribution, physical locations, and sometimes the node type actually matter. Still, fog application providers seem ill equipped to deal with services that expose individual edge locations and capacities. What we need is some middle ground that exposes enough details on distribution and physical locations so that applications can work with it but are not overwhelmed by it. We believe that the event-driven programming model of typical (serverless) Function as a Service (FaaS) [18] offerings naturally lends itself to this: Applications can use policies to define properties on various levels, e.g., for the case of location, based on density for a given region, everywhere within a given region, or based on concrete coordinates, which are then used by the fog FaaS offering to deploy functions accordingly. Of course, other programming models are also an option – however, a serverless approach seems like an easy-to-use abstraction that allows control over deployments while not forcing unnecessary details on application developers.

RC2: Capacity Management. One of the main properties of cloud services is the illusion of infinite resources [12]; for edge nodes, in contrast, upholding this illusion becomes impossible. Essentially, in an edge node there is a limited set of resources that is available to fog service consumers. In the cloud, resource pooling across a large customer base and inside gigantic data centers helps cloud providers even out natural demand variability. In Edge Computing, the limited capacity of nodes does not allow this: edge nodes will alternate between states of idleness and periods with more demand than they can handle. However, offloading processing and data to adjacent nodes or intermediary nodes is bound to impact QoS. This leads to the following three main questions for fog service providers:

1. How shall scarce capacity be distributed across applications in periods of high demand?
2. Shall fog service providers deny requests that cannot be fulfilled or shall they fulfill them partially, e.g., on other nodes, resulting in poorer QoS behavior?
3. How shall fog service providers size their edge and intermediary nodes if demand is unknown before deployment?

We believe that auctioning-based approaches (comparable to virtual machine spot markets [14]) can help us tackle the first two questions. E.g., an application could submit different price bids for specific groups of nodes (e.g., a specific edge node compared to an intermediary node with slightly higher latencies). Whenever demand exceeds capacity, this allows fog service providers to auction off free capacity to the highest paying bidder. If bids reflect individual utilities, this results in an (economically) optimal distribution of capacity where optimizations can be decided locally. Of course, our community could develop more complex mechanisms, e.g., distributions based on QoS requirements etc.

Regarding the third question, we believe that fog service providers should strive to design modular edge nodes where additional capacity can be added or removed incrementally, e.g., a rack of Raspberry Pis.

4.2 Fog Application Providers

There is a number of key differences between cloud-native and fog-native applications. Beyond dealing with abstractions offered by (potential) fog service providers, application services need to be able to run in a widely distributed setup (resulting in new modularization challenges), must be able to dynamically adapt to new environments (meaning a unique "fluidity" challenge), and must consider – on an architectural level – how to integrate various fog nodes ranging from high performance cloud machines to Raspberry Pi-based edge nodes.

RC3: Modularization. State-of-the-art modularization of applications typically follows a microservice-based approach [11] where each microservice represents a vertical cut through the application stack. This means that a microservice will usually have its own storage layer and may even use its own technology

stack. This modularization is largely driven by organizational decisions, i.e., a company organized into small DevOps teams will assign part of the functionality to each of the teams who will then be responsible for developing and running that microservice. Service-oriented architectures, in contrast, were mainly driven by technical design goals such as reusability, less by organizational factors. Now, in future fog applications, it may make sense to still follow a microservice-based approach (or it may not – depending on the organizational structure of the developer group), but within a microservice not all parts of the stack may run on all nodes, also considering that replication in the deployment suddenly becomes widely distributed. What we envision is that the vertical cuts of microservices are cut horizontally into smaller slices that can be distributed. For instance, an application service may decide to keep some basic presentation and application logic plus a caching layer at the edge and persistently store data on cloud nodes. For this, we need further research on how to slice microservices into even smaller pieces that can be replicated, shuffled around, and distributed widely.

RC4: Fluidity. Novel fog applications will have to be designed in a way that openly embraces a notion of fluidity: application modules will be interrupted frequently, will move between fog nodes, and maybe cloned at any time for such reasons as limited edge capacity, moving client devices, cost optimizations, or changing QoS requirements. Part of fluidity is also the ability to deal with varying resource capacity in nodes, e.g., when moving from a high performance cloud node to a small edge node. A convenient way to address this fluidity challenge is to strictly separate stateful and stateless components – an approach also (originally) taken with containers such as Docker [7]. The same paradigm can also be found in serverless FaaS deployments where applications are broken down into short-lived, stateless tasks that interact with a stateful storage tier. Research should focus on application designs that can cope with such degrees of fluidity.

4.3 Cross-Cutting Concerns

Failure handling primarily falls into the domain of application developers; however, depending on the actions of prospective fog service providers, failure handling activities can be severely affected. E.g., applications that use bare resources will have to employ fundamentally different mechanisms than applications that run on top a fog-based FaaS offering. Finally, as in Cloud Computing [10], security and privacy aspects are obvious candidates for further research challenges.

RC5: Graceful Degradation and Fault-Tolerance. Fault-tolerance is already challenging in the cloud – fog services with more nodes, wider distribution, and more complex and dynamic interdependencies between components make this even harder. At the same time, fog application domains such as autonomous driving or IoT tend to have more stringent QoS requirements as unhandled failures are bound to have physical effects. Based on this, it is

of utmost importance to handle and hide failures where possible and to support graceful degradation, e.g., in the form of a "safe mode", when not. How this can be achieved in fog applications or infrastructure services is still unclear. Suitable safeguards that monitor necessary requirements and notify in case of violations might be alternatives as well as compensation and data interpolation mechanisms. E.g., in autonomous driving, a fog storage service should aim to underestimate distances between cars when forced to in the presence of missing data as an overestimating interpolation may result in accidents. Here, research should work on identifying failure detectors or use case-driven compensation mechanisms.

RC6: Benchmarking and Testing. Realistic benchmarking [3] and testing are already challenging for cloud services: Both can easily be executed under controlled conditions (much work in benchmarking is dedicated to creating controlled environments) whereas interesting behavior, e.g., [1,2,16] may only be observable in realistic conditions, i.e., at scale and in a regular deployment while running a production workload. For fog-based applications, additional influence factors like heterogeneous hardware (potentially under control by different legal entities), wide geo-distribution, and only occasionally (re)occurring failure situations (possibly resulting from unforeseeable events in the "physical world") are added to the melee of challenges. In short, it becomes completely impossible to accurately emulate a production system for testing purposes since the cost of running such a setup is prohibitive and a second set of comparable resources may not even exist. Future research in this area should focus more on continuous benchmarking [24] and testing where both are executed during a build process: starting from a small isolated test setup to test basic QoS and functionality, gradually rolling out the software to the target environment. In short, the differences between benchmarking and testing on the one side and monitoring on the other side will gradually be blurring. Research should try to identify how this can be done in practice without impairing production systems. Furthermore, more research needs to be directed towards benchmarking fault-tolerance and related qualities.

RC7: Attack-Agnostic Services. Fog Computing also raises a number of security challenges, mainly how to provide appropriate security in an essentially hostile environment. Besides technical mechanisms, this might also require novel views to the weighing between acceptable risks and the costs of countermeasures. Especially with regard to the physical attack vectors omnipresent in the context of edge nodes, it seems highly worthwhile to put more effort on *coping* with attacks instead of *avoiding* them. We, thus, see the design of "attack-agnostic" services that accept certain attacks as unpreventable and implement measures for detecting and reacting to them (particularly to safeguard data) as a promising strand of research. Integrating such approaches into established security approaches for distributed service systems is another research question. Finally, the limited capabilities of edge nodes may also restrict the applicability

of security mechanisms widely adopted today: Encryption overheads – all too often already mistaken as virtually nonexistent [15] – may suddenly become restricting factors in the context of constrained edge nodes and. Also, the use of well-established concepts for access management and enforcement may prove inappropriate due to the common unreachability of centralized components, calling for novel schemes tailored to the specific givens in fog scenarios [23].

RC8: Compliance with Privacy Legislation. For privacy, Fog Computing provides both opportunities and challenges. Fulfilling data privacy requirements that give data subjects the right to know and to control where their personal data resides is particularly challenging in such a widely distributed, ever-changing network. Other regulations, e.g., regarding auditability or the physical placement of data are also difficult to address – possibly through information flow control, e.g., [22], if it can be implemented with little overhead.

On the other hand, Fog Computing also provides a number of opportunities, e.g., data could be aggregated or anonymized on edge nodes before forwarding it to a "central" cloud service, i.e., sensitive, detailed data may not even leave the direct vicinity of the data subject [8]. Similarly, the processing of sensitive personal data may be relocated away from centralized data lakes to the place of data collection, serving privacy goals like data minimization and purpose limitation.

5 Conclusion

While Cloud Computing can today be considered well established, modern application domains such as IoT, autonomous driving, or even mobile applications trying to tap the full potential of future 5G networks require an extension of the cloud towards the edge, thus, naturally leading to the new Fog Computing paradigm. In Fog Computing, application services run on both edge nodes (with low latency access but very limited resource capabilities) and in the cloud (with higher access latency but practically unlimited resources) as well as on possible intermediary nodes. Fog Computing as a new paradigm is a yet virtually unexplored field that offers a number of open research challenges.

In this paper, we started by describing what Fog Computing actually is – also in comparison to Edge Computing and Cloud Computing. Afterwards, we described the main obstacles for widespread fog adoption, distinguishing inherent and external challenges. An example for inherent challenges is the current lack of an edge service model which leads to a "build your own infrastructure" trend with all its undesirable implications; examples for external challenges are physical security or regulations. Building on these identified obstacles, we then described open research questions that arise for either (potential) fog service providers or fog applications running on top of such services. We also identified a number of cross-cutting concerns which fall into the domain of fog service consumers and application providers but are strongly affected by fog service provider actions, e.g., fault-tolerance, security, privacy, but also benchmarking and testing.

All in all, this paper aims to identify a research agenda in Fog Computing where we, as service-oriented computing researchers, as well as other research communities can contribute. Where feasible, we also pointed out specific directions that we believe might prove useful for overcoming the named obstacles or for addressing the resulting research questions.

Acknowledgments. This work has been supported by the European Commission through the Horizon 2020 Research and Innovation program under contract 731945 (DITAS project).

References

1. Bermbach, D., Tai, S.: Benchmarking eventual consistency: Lessons learned from long-term experimental studies. In: Proceedings of IC2E. IEEE (2014)
2. Bermbach, D., Wittern, E.: Benchmarking web API quality. In: Bozzon, A., Cudre-Maroux, P., Pautasso, C. (eds.) ICWE 2016. LNCS, vol. 9671, pp. 188–206. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-38791-8_11
3. Bermbach, D., Wittern, E., Tai, S.: Cloud Service Benchmarking. Measuring Quality of Cloud Services from a Client Perspective. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55483-9_1
4. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of MCC. ACM (2012)
5. Dastjerdi, A.V., Buyya, R.: Fog computing: helping the internet of things realize its potential. Computer **49**(8), 112–116 (2016)
6. Díaz, M., Martín, C., Rubio, B.: State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing. J. Netw. Comput. Appl. **67**, 99–117 (2016)
7. Ernst, D., Bermbach, D., Tai, S.: Understanding the container ecosystem: A taxonomy of building blocks for container lifecycle and cluster management. In: Proceedings of WoC. IEEE (2016)
8. Esposito, C., Castiglione, A., Pop, F., Choo, K.K.R.: Challenges of connecting edge and cloud computing: a security and forensic perspective. IEEE Cloud Comput. **4**(2), 13–17 (2017)
9. European Parliament and European Council: Regulation 2016/679 - general data protection regulation - GDPR (2016)
10. Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I.: Above the clouds: A berkeley view of cloud computing. University of California, Berkeley, Technical report (2009)
11. Lewis, J., Fowler, M.: Microservices: a definition of this new architectural term. ThoughtWorks (2014). Accessed 1 Jun 2017. http://martinfowler.com/articles/microservices.html
12. Mell, P., Grance, T.: The NIST definition of cloud computing. NIST Special Publication 800-145 (2011)
13. OpenFog Consortium Architecture Working Group: OpenFog Architecture Overview (2016). Accessed 8 Aug 2017. http://www.openfogconsortium.org/ra
14. Ouyang, X., Irwin, D., Shenoy, P.: Spotlight: An information service for the cloud. In: Proceedings of ICDCS. IEEE (2016)
15. Pallas, F., Bermbach, D., Müller, S., Tai, S.: Evidence-based security configurations for cloud datastores. In: Proceedings of SAC. ACM (2017)

16. Pallas, F., Günther, J., Bermbach, D.: Pick your choice in hbase: Security or performance. In: Proceedings of BigData. IEEE (2016)
17. Plebani, P., Garcia-Perez, D., Anderson, M., Bermbach, D., Cappiello, C., Kat, R.I., Pallas, F., Pernici, B., Tai, S., Vitali, M.: Information Logistics and Fog Computing: The DITAS Approach. In: Proceedings of CAISE Forum. CEUR (2017)
18. Roberts, M.: Serverless architectures (2016). Accessed 1 Jun 2017. http://martinfowler.com/articles/serverless.html
19. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput. **8**(4), 14–23 (2009)
20. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
21. Shi, W., Dustdar, S.: The promise of edge computing. Computer **49**(5), 78–81 (2016)
22. Singh, J., Pasquier, T.F.J.M., Bacon, J., Diaconu, R., Powles, J., Eyers, D.: Big Ideas paper: Policy-driven middleware for a legally-compliant Internet of Things. In: Proceedings of MIDDLEWARE. ACM (2016)
23. Stojmenovic, I., Wen, S., Huang, X., Luan, H.: An overview of fog computing and its security issues. Concurr. Comput. Pract. Exp. **28**(10), 2991–3005 (2016)
24. Tai, S.: Continuous, trustless, and fair: changing priorities in services computing. In: Lazovik, A., Schulte, S. (eds.) ESOCC 2016. CCIS, vol. 707, pp. 205–210. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-72125-5_16
25. Varshney, P., Simmhan, Y.: Demystifying fog computing: Characterizing architectures, applications and abstractions. CoRR (2017). http://arxiv.org/abs/1702.06331