Nomadic Applications Traveling in the Fog

Christoph Hochreiner[1(✉)], Michael Vögler[1], Johannes M. Schleicher[1], Christian Inzinger[2], Stefan Schulte[1], and Schahram Dustdar[1]

[1] Distributed Systems Group, TU Wien, Vienna, Austria
{c.hochreiner,m.voegler,j.schleicher,s.schulte,
s.dustdar}@infosys.tuwien.ac.at
[2] S.E.A.L, University of Zürich, Zürich, Switzerland
inzinger@ifi.uzh.ch

Abstract. The emergence of the Internet of Things introduces new challenges like network congestion or data privacy. However, it also provides opportunities, such as computational resources close to data sources, which can be pooled to realize Fogs to run software applications on the edge of the network. To foster this new type of resources, we revisit the concept of mobile agents and evolve them to so-called nomadic applications, which allow addressing vital challenges for the Internet of Things.

In this paper, we propose a system design to realize nomadic applications and discuss several challenges that need to be addressed in order to apply them to real-world scenarios.

Keywords: Fog computing · Cloud computing · Internet of Things · Mobile agents

1 Introduction

With the growing maturity of Internet of Things (IoT) concepts and technologies over the last years, we see an increase in IoT deployments. This increase fosters the integration of IoT devices and Cloud-based applications to realize information aggregation and value-added services. While the first IoT devices were only capable of emitting sensor data, today more and more IoT devices as well as network infrastructure, e.g., routers, provide computational resources to process and store data. To cope with the large volumes of data originating from IoT sensors in geographically distributed locations, IoT devices can pool their resources to create Fogs based on the Fog computing paradigm [1]. Fogs are inspired by the principles of Clouds [8], such as virtualization and pooling of resources but due to their location on the edge of the network and the limited amount of computational resources, they can not support the concepts of rapid elasticity and broad network access. However, they excel at other aspects such as low latency, location awareness, or data privacy [1].

Today's software applications are typically running in public Clouds, like Amazon EC2, or within private Clouds. Clouds provide on-demand resource scalability as well as easy software maintenance and have become the de-facto

A. Longo et al. (Eds.): IISSC 2017/CN4IoT 2017, LNICST 189, pp. 151–161, 2018.
https://doi.org/10.1007/978-3-319-67636-4_17

standard for today's software deployments. However, the Cloud computing paradigm is also confronted with several challenges, especially in the area of the IoT.

Due to the fact that applications are running in a remote data center, it is required to transfer all data to these data centers over the Internet. This approach is feasible for applications that process low volumes of data, but it becomes infeasible for IoT scenarios. In IoT scenarios, sensors produce large volumes of data, which cannot be processed by today's network infrastructure in real-time. Cloud-based applications are furthermore often in conflict with tight security restrictions since data owners want to ensure that no privacy-sensitive data leaks outside their companies' premises [9]. Therefore, it is often not feasible to transfer data to Cloud-based applications, because the data owner cannot control the data access after the data leaves the premises of the company.

To solve these challenges, it is required to refrain from deploying Cloud-based applications only in centralized remote data centers, but to consider a federated cloud architecture [2], and evolve Cloud-based applications into nomadic applications. In contrast to Cloud-based applications, which often require dedicated runtime environments, nomadic applications are self-contained software applications, which can transfer themselves autonomously among Fogs and Clouds and run directly on hypervisors due to unikernel architectures [7].

The idea is inspired by the principles of temporary workers or consultants, who move from one workplace to another to perform activities and led to the architectural design of mobile agents [5]. Mobile agents carry out operations in close proximity to the data source to reduce latency and network traffic [6]. These properties make them a perfect fit to tackle today's challenges for the IoT. Nevertheless, although mobile agents have been proposed around two decades ago, this concept never took off, mainly due to security considerations and the need for dedicated execution environments [4].

Since the initial proposal of mobile agents, the technological landscape has advanced and the Cloud computing paradigm has fostered the technical foundation for the execution of arbitrary applications on virtualized and pooled resources. Given the technical requirements for nomadic applications, Fogs provide a controlled and secure execution environment, which is managed by the owner of the data. Therefore, nomadic applications allow for a more efficient data transfer among IoT devices and processing applications due to the elimination of the transfer over the Internet. Furthermore, due to the fact that Fogs are often fueled by IoT devices who are already within the control of the data owner, it is also feasible to enforce strict privacy policies.

Although nomadic applications are the most promising approach to process privacy-sensitive data on already existing computational resources on the edge of the network, there are still a number of challenges, like data transfer or data recovery, which need to be resolved as discussed in Sect. 5.

The remainder of this paper is structured as follows: First, we provide a short discussion on the related work in Sect. 2. Then, we provide a motivational scenario in Sect. 3. Based on the motivational scenario we identify several

requirements and present the foundation of our system design in Sect. 4. Furthermore, we discuss open research challenges in Sect. 5 before we conclude the paper in Sect. 6.

2 Related Work

The only recent manifestation of Fog computing, which is often also considered as edge computing, leads to a plethora of definitions (e.g., [1,3,11,12]). Literature as well as the OpenFog Consortium[1] consider Fog computing as an evolution of Cloud computing, which extends established data centers with computational resources that are located at the edge of the network. This enables Fog computing to bridge the currently existing geographic gap between IoT devices and the Cloud by providing different deployment locations to cater for the different requirements of data processing applications, e.g., low latency or cost efficiency [1].

Bonomi et al. [1] propose a layered model, which categorizes the different computing platforms ranging from embedded sensors from IoT devices over field area networks to data centers, where each layer provides a distinct level of quality of service. Furthermore, Dastjerdi et al. [3] provide a reference architecture for Fog computing that can be used to leverage the computational resources at the edge of the network and outline several research challenges, such as security and reliability. Another challenge for the realization of Fogs is the heterogeneity of devices that are used to build Fogs [12]. Nevertheless, there is already some preliminary work, such as the LEONORE framework proposed as part of our previous work, which accommodates the diversity for the deployment of applications on IoT devices [13].

3 Motivational Scenario from the Manufacturing Domain

Our motivational scenario originates from the manufacturing domain, which is one of the most advanced areas regarding the realization of the IoT. Today's manufacturing companies operate specialized manufacturing machines in different geographic locations [10], as depicted in Fig. 1. These machines are continuously monitored by sensors to assess their status as well as the quality of the manufactured products. The product quality often decreases over time, because expendable parts wear off and the machines need to be recalibrated from time to time. This recalibration requires the use of a dedicated application that analyzes sensor data and calculates the optimal configuration for each machine. Furthermore, the application maintains a knowledge base that is continuously updated to improve recalibration results based on reinforced learning. Now, there are two approaches on how to execute the recalibration application.

The first approach is to run the application within the factory's premises. This approach is optimal in terms of data privacy since the data never leaves the

[1] https://www.openfogconsortium.org.

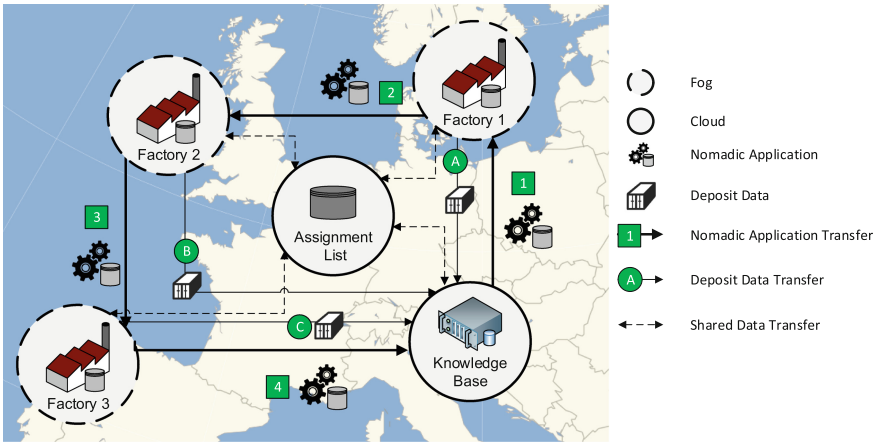

Fig. 1. Motivational scenario

company. However, this approach requires dedicated computational resources to perform the recalibration. To reduce the cost for the recalibration, it is also possible to operate the recalibration application in a Software-as-a-Service manner on Cloud resources in a remote data center.

Although this second approach reduces the cost, it also has its disadvantages. For this approach, the data needs to be transferred to a data center over the Internet. This data transfer potentially conflicts with legal or organizational policies, which may forbid any data transfer to remote locations or it simply may not be feasible to transfer a huge volume of sensor data over the Internet.

Fortunately, the concept of nomadic applications running in Fogs combines the benefits of the two previously described approaches. Whenever a machine is in maintenance mode to perform the recalibration, some of its computational resources are not used and can be contributed to a local Fog to host applications, such as the nomadic recalibration application. Another benefit of hosting the application on the Fog is that the recalibration is performed within the companies' premises and the data owner can control all activities performed by the application. The high trust level facilitated due to the tight control, entitles the recalibration application to also access internal interfaces of the machine, which is not possible for Cloud-based applications (due to security and data privacy reasons).

Figure 1 shows an exemplary order of events for a nomadic application, which travels among three factories to recalibrate machines. At the beginning of the journey, the application is residing at the knowledge base, since it is not required for any recalibration. Here, the application updates its configuration, i.e., becomes a stateful application, for future calibrations based on historic data.

Then, the application relocates immediately, as depicted by number 1, from the knowledge base to its first working assignment at Factory 1. At Factory 1, the application is running on the Fog of Factory 1, where it can access the sensor

data as well as any other factory-specific configuration data. This data is then processed using the Fog, i.e., pooled and virtualized computational resources provided by different IoT devices. After completing its task, the application is ready to relocate to the next working assignment based on the information from a remote assignment list.

Due to the fact that the recalibration operation at Factory 1 generated new insights, which can be used to refine the recalibration operation, it is required to transfer these insights to the knowledge base. This transfer may contain a large amount of non-privacy-sensitive data, which does not exhibit any time-related transportation constraints and can be carried out independently from the travel route of the nomadic application as shown by migration A in Fig. 1.

These two operations are repeated twice, to also recalibrate the machines for Factory 2 respectively Factory 3 and collect the obtained insights in the knowledge base for future refinement. At the end of the journey, the nomadic application returns to the knowledge base to update its configuration and to be ready for future recalibration activities.

4 System Design

Our system design to realize a nomadic application infrastructure encompasses Clouds and Fogs, which are connected by a network, i.e., the Internet. To address the requirements, we propose the Nomadic Application Infrastructure, as depicted in Fig. 2. For our discussion, we distinguish between application management aspects, which are designed to run in the Cloud, and data management-oriented ones because these two management topics expose different requirements.

4.1 Application Management

Requirements. The requirements for the application management ensure that nomadic applications can travel from Fog to Fog over the Internet and cover all aspects of the nomadic application lifecycle. The first requirement is that applications need a location where they can reside, whenever they do not run on any Fog. Furthermore, the application management also needs to provide a mechanism to upgrade applications, e.g., to fix faulty applications or improve existing ones.

While some applications carry application data, i.e., are stateful, and can only exist once, the majority of nomadic applications is stateless and can be replicated to operate in multiple locations at the same time. Here, the application management needs to ensure that stateful applications are never replicated, i.e., the management needs to keep track of all running applications and the number of instances thereof. Furthermore, the application management needs to provide a recovery mechanism when one application crashes or the Fog is permanently disconnected from the Internet.

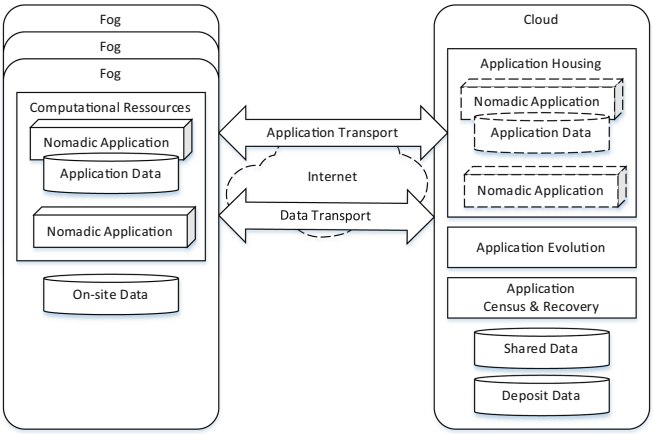

Fig. 2. Nomadic application infrastructure

Besides the infrastructural aspects, it is also required to support a fast and secure application transfer. Here, it is crucial to ensure the integrity of the application at any time so that the applications cannot be abused for malicious purposes, e.g., as attack vectors for sabotage operations or data leakage.

Architecture. Although nomadic applications are designed to operate in a decentralized manner, it is required to provide a centralized component to support the application management whenever they are not deployed on any Fog.

Application Housing and Transport. The housing component stores a shadow copy of each application to enable replication for stateless applications. Implementation-wise, we propose a similar infrastructure as the Docker Registry to cater for the required functionalities. Originating from the housing infrastructure, the applications can start their journey towards the Fogs over the Internet. These movement operations are supported by a dedicated application transport layer, which ensures a fast and reliable transport. Besides the basic transport aspects, this layer also ensures that applications are not modified or tampered with at any point in time, by validating the application's integrity based on cryptographic hash functions after each transport. The application housing also allows applications to update their application data, i.e., the state, before they continue their journey towards the privacy-sensitive Fogs.

Application Evolution. To comply with the need for continuous evolution for nomadic applications, the housing component furthermore needs to provide the possibility to update existing applications at any time and to inform applications that are running on Fogs about the update. These updates can be only applied when the applications have finished their tasks they must not be altered

while processing data. To solve this challenge, we require the application housing component to support versioning of applications to facilitate the update of applications in the housing component. The applications are required to regularly check whether a new version is available. If this is the case, stateful applications need to return to the application housing to transfer their application data to the evolved application and then they are able to continue their work.

For stateless applications, i.e., applications without application data attached, the housing component updates the shadow copy. The stateless application checks each time, before it moves to another location, whether the application has evolved. If this is the case, the application returns to the housing component to apply the updates or is discarded and new instances of the application are spawned from the updated shadow copy.

Application Census and Recovery. The application census and recovery component keeps track of all nomadic applications that are currently running on Fogs. This census is required to ensure that stateful applications are not replicated at any time. Besides the census functionality, this component is also required to provide a recovery mechanism for stateful applications. This recovery mechanism needs to decide whether a stateful operation will or will not return from the Fog, either due to an application crash or due to the fact that the Fog is permanently disconnected from the network. In this situation, the recovery mechanism restores the application alongside with its application data from a previous point in time and appoints the reconstructed application as the new sole instance of the nomadic application.

4.2 Data Management

Requirements. The data management is responsible for storing and transferring data of different data types that are used by nomadic applications.

The first data type is the application data, which represents information that is stored within the application. This type is rather small, e.g., configuration settings or initialization values, does not contain any privacy-sensitive data, and remains the same for every Fog location. This application data only contains static information to avoid any data leakage, e.g., by embedding privacy sensitive information within the application data when the application is running in a Fog. Therefore, it must be ensured that this state can be only updated within a specific location that does not contain any privacy-sensitive information, e.g. the application housing.

The second type represents information that comprises both static data, which is stored in the Fog, and dynamic data, such as streaming data, e.g., sensor data, which cannot be stored due to its large volume. Although these two data types have entirely different requirements towards data management, both data types contain privacy-sensitive information, which must be processed according to the owner's policies. Due to the geographic colocation of this data and the application at runtime, this data is not moved outside the companies premises, which mitigates a potential data leakage.

Besides the application data and the owner's data, applications also require remote data repositories. Here, the data storage infrastructure needs to support two different scenarios. For the first scenario, the applications must be able to have a shared data storage, which can be used to either communicate with Fog providers, to retrieve new next job assignments or to share non-privacy-sensitive information with all replicas of an application.

This shared repository is required to be ACID-compliant and to support different access policies, e.g., a permissive one for the job assignments and a restricted one to restrict the access to a specific application. In addition to the shared repository, the application also requires a storage location, where it can deposit any arbitrary information that is generated while operating in one of the Fogs. This storage also needs to be partitioned into different segments, which can be only accessed by the assigned application. In contrast to the shared repository, this data storage only requires eventual consistency and any data transfer to this storage does not need to comply with any time-based restrictions. Nevertheless, the data needs to be persisted and provided at any point in time. The Fog operator needs to be able to check, whether the data transferred to the shared repository and the data deposit complies with the owner's policies and does not contain any privacy-sensitive information.

Architecture. To address the identified requirements for the different data types, we propose different data management techniques, as illustrated in Fig. 2.

Application Data. To realize tight coupling between the nomadic application and its application data, we propose to employ unikernels [7]. This tight coupling ensures that the data is always available for the application and the integrity check after the transfer over the network can be applied to the application and the application data at the same time.

On-site Data. Although the on-site data can be either static or dynamic, both data types must not leave the Fog, i.e., the premises of the data owner. Therefore, the data needs to be protected by methods originating from information rights management. These methods ensure that the data can be only read within the Fog and any data leakage outside the Fog renders the data useless due to the information rights management restrictions.

Shared Data. For the realization of the shared data, we propose a storage infrastructure similar to Amazon S3, where each application and potential replicas are able to read and write from a dedicated storage bucket. This storage infrastructure furthermore provides a high and instant availability regarding read and write operations, which distinguish the shared data from the deposit data.

Deposit Data. The deposit data follows the principles from Amazon Glacier. This makes it possible to store arbitrary information at any point in time, but it may

take some time until the information is persisted within the storage respectively accessible for the application to be read. Furthermore, it is required to implement a data transport layer, which ensures both the integrity of the data as well as the compliance with the policy of the data owner at any time.

5 Discussion

Even though nomadic applications build on established principles and concepts, we have identified four research challenges, which are essential to realize them:

Scalable Privacy Protection for On-Site Data. One of the most important challenges is the development of a scalable privacy protection mechanism for dynamic data. While there are already solutions to enforce fine-grained access policies for static data, like office documents, there is still a lack of solutions for dynamic data. This is mainly due to the large volume of data that needs to be processed in real-time. Furthermore, it needs to be ensured that malicious applications cannot store any of this privacy-sensitive information within their own application data and transfer the data out of the owner's premises. Here, the data transport layer needs to ensure that only non-privacy-sensitive data, which is permitted to be sent to the deposit data, is transferred out of the Fog.

Target Scheduling for Stateful Applications. While stateless applications can be replicated on demand to be deployed in arbitrarily many Fogs, it is more complex for stateful applications to select their next assignment because they can exist only once and may be required at several locations at the same time. A first solution approach is the implementation of the first-come, first-served principle, but it is more desirable to implement a more flexible scheduling mechanism to avoid that one Fog operator occupies a specific application by spamming the waiting queue. To solve this issue it is desirable to implement an auctioning mechanism, which ensures a fair assignment of applications across all Fogs. This auctioning mechanism needs to be implemented on top of the transport layer in a decentralized manner to avoid any single point of failure.

Support Different Speeds in the Transport Layer. Due to the fact that the network transfer capabilities among the Fogs, application infrastructures, and storage facilities are limited, it is required to design a transport scheduling algorithm that ensures that each entity, i.e., a nomadic application or data, is transferred as efficiently as possible. Therefore it is required to design a transport protocol, which allows the different entities in the system to negotiate the transportation capabilities. This protocol should allow delaying those entities, which are not required to be transferred in a time critical manner, such as nomadic applications without any further job assignments returning to the housing component or deposit data. Nevertheless, it must be guaranteed that no entity suffers starvation in terms of transportation capabilities and all entities reach their destination eventually.

Recovery Mechanism. The final challenge is the development of a recovery mechanism which allows the reconstruction of application data. This reconstruction is required, when an application is unable to return to the housing component due to an application crash, a network disruption between the Fog and the network, or simply a failure of a single IoT device, which contributes to the Fog. Therefore, it is required to design a lightweight and efficient solution to track application states, e.g., data synchronization points, to be able to reconstruct applications. Furthermore, it is necessary to implement an algorithm to decide whether an application is lost forever and needs to be recovered or whether the application just needs longer than expected to finish its job. This functionality is crucial to avoid instances of a stateful application at any future point in time.

6 Conclusion

The growing use of IoT devices enables the emergence of the Fog computing paradigm which represents an evolution of the Cloud-based deployment model. In this paper, we identify opportunities of Fogs and introduce the concept of nomadic applications. These nomadic applications promise to solve both the challenges regarding the constantly growing volume of data and to enable a tight control of the data's privacy. In our future work, we will further develop the infrastructure for nomadic applications and apply them to real world scenarios.

Acknowledgements. This paper is supported by TU Wien research funds and by the Commission of the European Union within the CREMA H2020-RIA project (Grant agreement no. 637066).

References

1. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the Internet of Things. In: Proceeding of the MCC workshop on Mobile Cloud Computing, 1st edn., pp. 13–16. ACM (2012)
2. Celesti, A., Fazio, M., Giacobbe, M., Puliafito, A., Villari, M.: Characterizing cloud federation in IoT. In: 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 93–98. IEEE (2016)
3. Dastjerdi, A.V., Gupta, H., Calheiros, R.N., Ghosh, S.K., Buyya, R.: Fog Computing: principles, architectures, and applications. In: Buyya, R., Dastjerdi, A.V. (eds.) Internet of Things, pp. 61–75. Morgan Kaufmann (2016)
4. Kotz, D., Gray, R.S.: Mobile agents and the future of the internet. Oper. Syst. Rev. **33**(3), 7–13 (1999)
5. Lange, D.B., Mitsuru, O.: Programming and Deploying Java Mobile Agents Aglets. Addison-Wesley Longman Publishing Co., Inc., Boston (1998)
6. Lange, D.B., Mitsuru, O.: Seven good reasons for mobile agents. Commun. ACM **42**(3), 88–89 (1999)
7. Madhavapeddy, A., Mortier, R., Rotsos, C., Scott, D., Singh, B., Gazagnaire, T., Smith, S., Hand, S., Crowcroft, J.: Unikernels: library operating systems for the cloud. ACM SIGPLAN Not. **48**(4), 461–472 (2013)

8. Mell, P., Grance, T.: The NIST definition of cloud computing (2011)
9. Schleicher, J.M., Vögler, M., Inzinger, C., Hummer, W., Dustdar, S.: Nomads-enabling distributed analytical service environments for the smart city domain. In: International Conference on Web Services (ICWS), pp. 679–685. IEEE (2015)
10. Schulte, S., Hoenisch, P., Hochreiner, C., Dustdar, S., Klusch, M., Schuller, D.: Towards process support for cloud manufacturing. In: 18th International Enterprise Distributed Object Computing Conference (EDOC), pp. 142–149. IEEE (2014)
11. Shi, W., Dustdar, S.: The promise of edge computing. Computer **49**(5), 78–81 (2016)
12. Vaquero, L.M., Rodero-Merino, L.: Finding your way in the fog: towards a comprehensive definition of fog computing. ACM SIGCOMM Comput. Commun. Rev. **44**(5), 27–32 (2014)
13. Vögler, M., Schleicher, J.M., Inzinger, C., Dustdar, S.: A scalable framework for provisioning large-scale IOT deployments. Trans. Internet Technol. (TOIT) **16**(2), 11 (2016)