Dynamic Management of Distributed Stream Processing Pipelines in Fog Computing Infrastructures

Patrick Wiener
FZI Research Center for Information Technology
Karlsruhe, Germany
wiener@fzi.de

Abstract

While todays' stream processing applications are typically deployed in the cloud, newly arising use cases in the context of Internet of Things (IoT) often require low-latency analytics to derive time-sensitive actions. A common approach, referred to as *fog computing*, shifts the focus away from the cloud by offloading specific parts of the analytical pipelines in closer proximity of the spatially distributed devices at the edge. However, this requires mechanisms for context-aware deployment, scale, or monitoring both in the cloud as well as the fog landscape. This work explores the challenges with respect to dynamically managing distributed stream processing pipelines in heterogeneous fog computing infrastructures.

CCS Concepts • **Computer systems organization** → *Fog computing*;

Keywords fog computing; stream processing pipelines

1 Introduction

The real value of the IoT in domains, such as smart city, manufacturing automation, or transport service management, is not derived from data, but from insights that facilitate real-time actions that increase asset efficiency, reliability and utilization. But, to actually save time and money with IoT, data insights have to come from somewhere – typically centralized, scalable cloud computing platforms tailored for the hardware, connectivity and data management needs of these domains. Thereby, the dataflow approach is a popular programming paradigm to model these specific analyses [5, 8] as pipelines composed of several individual processing elements that are typically interconnected by a publish-subscribe messaging system. However, for scenarios that require very low latency, are limited in available bandwidth, or where privacy is a limiting factor, there are some inevitable downsides to solely deploying these pipelines in the cloud. Consequently, an obvious approach is to extend the cloud to move certain computation to the edge to overcome these issues, while still having access to scalable, infinite resources, which is commonly referred to as *fog computing* [3]. This decentralized computing paradigm shifts the focus away from the cloud by offloading specific processing elements in closer proximity of the spatially distributed devices, analogous to the "code to data" principle in big data. Certainly, leveraging software container technology (e.g. Docker[1]) and the surrounding ecosystem for managing container-based clusters (e.g. Kubernetes[2]), provide a natural fit for (to some extent) resource-limited fog computing infrastructures (FCI) due to the minimal overhead [1, 6]. Still, there is little adoption of the fog computing paradigm in practice [2]. One reason might be the shortcoming of managing these hierarchical and heterogeneous FCI. Thereby, it is crucial to have context-aware mechanisms that enable a dynamic deployment of stream processing pipelines (*SPP*) [8] based on varying capabilities of dedicated compute resources as well as changing overall pipeline requirements.

The aim of this work is to examine the following central research questions: **(RQ1)** How can FCI and their entities be modelled? **(RQ2)** How can dynamic (context-aware) deployments of *SPP* be achieved? **(RQ3)** How can individual processing elements of *SPP* be moved (inter-, intra-layer) or scaled-up/down (intra-node) when the context (capabilities, requirements) changes?

2 Motivating Use Case

The constant growth of the e-commerce business further increases the traffic density in urban areas caused by frequent transport service provider order delivery, thus, introducing various issues for cities, e.g., due to noise, emissions, or general traffic level. To address these problems, electrified vehicular delivery robots equipped with various sensors (e.g. laser scanner) and actuators (e.g. motors) can be used to pick up packages at defined packing stations in order to autonomously deliver them to customers. On the one hand, a centralized approach is necessary for global monitoring and to perform big data analytics. On the other hand, a decentralized decision-making in terms of edge analytics is helpful to increase the overall responsiveness and robustness in case of latency-sensitive analyses, network outages or privacy implications. Consequently, fog computing can be beneficial aiming to roll-out defined distributed *SPP* on a hierarchical infrastructure (cloud, intermediary, edge) to perform dedicated monitoring and analytical tasks.

3 Approach

Generally, a FCI consists of a multi-layer hierarchical infrastructure (see Figure 1) of heterogeneous compute nodes along the cloud-edge continuum typically implemented leveraging container technologies, where there exists an intermediary layer (often referred to as fog), that provides additional processing power in closer proximity to the edge as compared to the cloud [2]. At the edge, nodes are having access to sensors and actuators to retrieve data or place actions with respect to the real-world. The cloud functions as a centralized scalable backbone for all sublayers, e.g., to store analyzed and aggregated data, to run big data analytics or to provide

Middleware'18, Rennes, France

DOI: 10.1145/nnnnnnn.nnnnnnn

[1] https://www.docker.com/
[2] https://kubernetes.io/

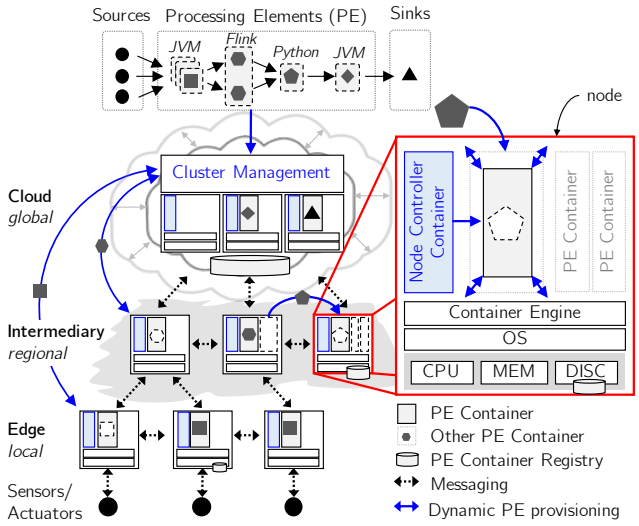

Figure 1. Dynamic management of distributed stream processing pipelines (*SPP*) in fog computing infrastructures

adequate compute resources for certain processing elements of the *SPP*. Individual processing elements can be realized in a multitude of programming languages and frameworks (e.g. Java, Apache Flink[3], Python), each performing a dedicated task and exchanging their results in a publish-subscribe based messaging approach. The semantic description of each element is used in order to perform reasoning on the typically user-defined *SPP* identifying suitable connections. Additionally, each element and the overall pipeline specifies infrastructural requirements, e.g., hardware related (CPU, Mem, Disc) as well as various other extensible user-defined requirements, e.g., privacy-zones. Other than that, the infrastructure itself is semantically described thus providing useful information about node capabilities. Both, requirements and capabilities can be extended and adapted in a flexible manner during runtime according to the given context (**RQ1**).

On this basis, we derive the concept of a node controller that runs on every node in the cluster and is responsible (1) for watching for context-changes, e.g., topological changes due to intermediary node failure or preemption policies, (2) for triggering the re-deployment of a processing element if central cluster manager is not available due to connectivity issues, and (3) for resizing a specific processing element. Thus, every compute layer provides access to relevant snapshots of the central container registry, that is used for caching images to reduce network traffic (**RQ2**).

Lastly, to move a processing element either horizontally (intra-layer) or vertically (inter-layer) the respective node controller needs to checkpoint and restore the containers state on another matching node. Also, pre-deployed containers can be scaled-up/down (intra-node), e.g., when hardware requirements or other contextual information change during runtime as shown in the projection of the intermediary node in Figure 1 (**RQ3**).

[3]https://flink.apache.org/

4 Related Work

In [5] a distributed dataflow programming paradigm is introduced and implemented by extending Node-RED[4]. Thereby, processing pipelines are executed on FCI based on static node capabilities and defined constraints. In [9] Kubernetes label feature is used to provide static node metadata (e.g. hardware resources, location) in order to appropriately deploy containers on suitable nodes. In [10] a framework for dynamic resource provisioning and automated container-based application deployment is proposed, presenting a more fine-grained description of requirements including prioritization to enable preemption as well as privacy constraints in terms of the actual placement in the infrastructure hierarchy. A container-based architecture for supporting autonomic data stream processing applications on FCI is shown in [4], yet only exploiting native Docker features to scale and migrate application containers.

Overall, the abovementioned approaches are not context-aware such that they do not account for dynamism in changes of the environment. Additionally, they neglect relevant dependencies and requirements of the holistic *SPP*, and do not allow for higher level reasoning. The closest related work is [7] where a context-aware software framework for management of resources and service provisioning based on topology changes in FCI is proposed, mainly focusing on failover and handover management.

5 Conclusion

The aim of this work is to examine how to dynamically manage distributed stream processing pipelines that support context-awareness in heterogeneous fog computing infrastructures. The results will be implemented and intensively evaluated in the presented use case in the course of a collaborative research project.

References

[1] Paolo Bellavista and Alessandro Zanni. 2017. Feasibility of Fog Computing Deployment based on Docker Containerization over RaspberryPi. In *Proc. of the 18th Int. Conf. on Distributed Computing and Networking - ICDCN '17*. 1–10.
[2] David Bermbach, Frank Pallas, David Garc, Ronen Kat, and Stefan Tai. 2017. A Research Perspective on Fog Computing. *Research Paper* (2017).
[3] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog Computing and Its Role in the Internet of Things. *Proc. of the first edition of the MCC workshop on Mobile cloud computing* (2012), 13–16.
[4] Antonio Brogi, Gabriele Mencagli, Davide Neri, Jacopo Soldani, and Massimo Torquati. 2018. Container-Based Support for Autonomic Data Stream Processing Through the Fog. *Euro-Par 2017: Parallel Processing Workshops* (2018), 17—-28.
[5] Nam Ky Giang, Michael Blackstock, Rodger Lea, and Victor C.M. Leung. 2015. Developing IoT applications in the Fog: A Distributed Dataflow approach. In *Proc. - 2015 5th Int. Conf. on the Internet of Things, IoT 2015*. IEEE, 155–162.
[6] Bukhary Ikhwan Ismail, Ehsan Mostajeran Goortani, Mohd Bazli Ab Karim, Wong Ming Tat, Sharipah Setapa, Jing Yuan Luke, and Ong Hong Hoe. 2016. Evaluation of Docker as Edge computing platform. In *ICOS 2015 - 2015 IEEE Conf. on Open Systems*. IEEE, 130–135.
[7] Saša Pešić, Milenko Tošić, Ognjen Iković, Mirjana Ivanović, Miloš Radovanović, and Dragan Bošković. 2017. Context aware resource and service provisioning management in fog computing systems. *Studies in Computational Intelligence* 737 (Oct 2017), 213–223. http://link.springer.com/10.1007/978-3-319-66379-1
[8] Dominik Riemer, Ljiljana Stojanovic, and Nenad Stojanovic. 2014. SEPP: Semantics-based management of fast data streams. In *Proc. - IEEE 7th Int. Conf. on Service-Oriented Computing and Applications, SOCA 2014*. IEEE, 113–118.
[9] Cecil Wöbker, Andreas Seitz, Harald Mueller, and Bernd Bruegge. 2018. Fogernetes: Deployment and management of fog computing applications. In *IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018*. IEEE, 1–7.
[10] Emre Yigitoglu, Mohamed Mohamed, Ling Liu, and Heiko Ludwig. 2017. Foggy: A Framework for Continuous Automated IoT Application Deployment in Fog Computing. In *Proc. - 2017 IEEE 6th Int. Conf. on AI and Mobile Services, AIMS 2017*. 38–45.

[4]https://nodered.org/