Cloudless Resource Monitoring in a Fog Computing System Enabled by an SDN/NFV Infrastructure

Duy Thanh Le, Marcel Großmann, Udo R. Krieger
Faculty WIAI, University of Bamberg
An der Weberei 5, D-96047 Bamberg, Germany
{duy-thanh.le, marcel.grossmann}@uni-bamberg.de, udo.krieger@ieee.org

***Abstract*—Today's advanced Internet-of-Things applications raise technical challenges on cloud, edge, and fog computing. The design of an efficient, virtualized, context-aware, self-configuring orchestration system of a fog computing system constitutes a major development effort within this very innovative area of research. In this paper we describe the architecture and relevant implementation aspects of a cloudless resource monitoring system interworking with an SDN/NFV infrastructure. It realizes the basic monitoring component of the fundamental MAPE-K principles employed in autonomic computing. Here we present the hierarchical layering and functionality within the underlying fog nodes to generate a working prototype of an intelligent, self-managed orchestrator for advanced IoT applications and services. The latter system has the capability to monitor automatically various performance aspects of the resource allocation among multiple hosts of a fog computing system interconnected by SDN.**

***Index Terms*—Fog Computing, SDN/NVF, Container Virtualization, Autonomic Orchestration, Docker**

I. Introduction

Over the last decades, we have witnessed the rapid development of different computing and networking paradigms. They have been evolving from a local computing paradigm and its parallelization towards cloud-based distributed computing and from the US DARPA project towards today's advanced Internet-of-Things (IoT), [1], [2], [6]. It is a phenomenon that now affects every aspect of our daily activities like how we live, work, and socialize among each other.

Cloud computing [1] offers several advantages, e.g., scalability, a reduction of management efforts, an on-demand resource allocation, or a pay-as-you-go model. It is realized by the three primary models IaaS, PaaS, and SaaS. Although this computing paradigm is highly attractive for data-intensive processes, it is also facing a certain number of challenges, such as challenges w.r.t. bandwidth, latency, resource constraints, mobility, and security. The latter have stimulated the development of edge and fog computing as new distributed computing paradigms to support the vision of an IoT without severe restrictions [2], [9]. This concept focusses on pushing the processing and storage of data closer to the edge of a modern network infrastructure based on software-defined networks with network functions virtualization (SDN/NFV), thus, reducing the latency and improving the availability of advanced services [9], [12].

Furthermore, the adoption of lightweight virtualization techniques is drastically growing in all cloud, fog, and edge computing models due to their advantages like the reduction of costs and energy, and an immediate provisioning of services. In particular, container based virtualization is considered as a practical solution for fog computing and its SDN in a modern IoT environment as it demands less system resources compared to other virtualization technologies [4]. As traditional network architectures cannot meet the growing needs on variability, complexity, and high volume of the traffic load combined with the diverse QoS and Quality-of-Experience (QoE) requirements, flexible network architectures based on SDN and NFV technology are needed to support the fog computing paradigm for IoT environments and their services, [9], [10], [11], [12]. In particular, the concept of network functions virtualization (NFV) substantially transforms the delivery of sophisticated network services by providing network functions, recently deployed in specific hardware appliances, into commodity hardware such as industry-standard high-volume servers, switches, and storage [13].

At present, the rapidly evolving Internet-of-Things and its related services and applications are demanding an increasing effectiveness of the deployed SDN/NFV infrastructures interconnecting this vast number of heterogeneous objects and smart things [2], [6]. Considering the limited computing and data storage capabilities of the underlying IoT devices, the traditional solution is to offload services to centralized locations offering cloud services. However, some severe disadvantages arise, [14], such as

- a high latency due to the distance between the edge devices and their services running in the cloud nodes,
- a vast computational load onto the cloud systems due to the propensity of task offloading and, therefore, additional processing and queuing delays as well as bandwidth exhaustion,
- severe availability, scalability, speed, and resource management issues by the enormous number of IoT devices coalesced with the potential instability of the transmission medium (especially in wireless communication) since the computational nodes operating in the data centers may not promptly provide demanded services due to network congestion and/or connection failures.

These severe resource management and performability issues have stimulated a radical change from the cloud-centric approach towards the novel, innovative fog computing systems

that can provide lower latency and facilitate a locality awareness of services, etc. According to NIST, it supports to "decentralize applications, management, and data analytics into the network itself using a distributed and federated compute model" [15]. Its core concept is to bring more resources and computational power closer to the edge layer where data are generated and acted upon. Thus, fog computing is associated with a continuous, virtualized middleware layer between the cloud and the IoT devices [2].

This sketched development illustrates that there is a strong need to develop an effective container orchestration tool for fog computing. In our approach its design is governed by an autonomic computing framework of self-management, self-configuration, and self-optimization, [3]. Regarding such a fog computing system in an advanced IoT scenario and following a container based virtualization approach, it can promptly and functionally manage containers running in multiple, large-scale geographically distributed fog nodes. It can also collaborate with an SDN/NFV infrastructure to control in a real-time fashion the network behavior based on the knowledge provided by the monitored system and service data as well as the realized application demands.

Regarding the integration of effective, self-managed resource management and autonomic orchestration schemes into fog computing systems, it is the primary goal of our research to generate a first fundamental component of the underlying MAPE-K model in terms of a fully container-virtualized prototype interworking with an SDN/NFV infrastructure, see also [7], [8].

In this paper we present a working prototype of such a Cloudless Resource Monitoring System (CRMS) consisting of various software components:

- Two multi-processes systems, called *Collector* and *Worker*, can automatically manage, monitor, and report the resource consumption of all virtualized hosts within an SDN/NFV infrastructure. The employed monitoring framework used in CRMS is based on Docker [5] and uses the multi-architecture monitoring framework of Großmann and Schenk [8] as a starting point.
- A custom-built ONOS plugin, called *CRMA*, aggregates the resource usage data generated by the underlying *Collector* system into a time-series database realized by the real-time database `Influxdb`.
- Finally, multi-panel dashboards provide bird-eye views of the current resource consumption within the underlying fog computing system.

The paper is organized as follows. In the next section we describe the architecture of the proposed Cloudless Resource Monitoring System for advanced fog computing in an IoT scenario that is enabled by an underlying SDN/NFV infrastructure. In particular, we discuss the three basic layers and functions of the CRMS architecture that is governed by the MAPE-K concept of an employed autonomic orchestration approach. Finally, we present some conclusions and a perspective on further software development of CRMS.

II. An Architecture of Cloudless Resource Monitoring for Fog Computing Systems

In this section we present the general software architecture of the proposed Cloudless Resource Monitoring System (CRMS) for a fog computing environment and its interworking with a software-defined network incorporating virtualized network functions. The CRMS applies a container virtualization technique based on Docker [4], [5] and follows the MAPE-K approach of autonomic computing to realize the monitoring step of an underlying autonomic orchestration model, [3], [7]. The proposed CRMS consists of three separate logical layers as shown in Figure 1, namely, a generation, accumulation, and presentation layer. Subsequently, we explain the incorporated functionalities and the interworking of the realized software components in a bottom-to-top fashion following the direction of the information flows in our proposed system.

A. The Generation Layer

The lowest *Generation* layer incorporates an arbitrary number of SDN switches, e.g., OpenFlow compliant switches or P4 switches. They are connected to multiple heterogeneous computing nodes, e.g., SBCs, desktops, workstations, etc. These logical *Worker* nodes of the fog layer comprise the virtualized network and computing functionalities. They are supplemented by a *Collector* node. Residing in the lowest logical layer of the fog computing model, these *Collector* and *Worker* nodes constitute the main computational power of the CRMS. Their main functions as well as internal software components are working as follows:

- It is the primary task of the *Collector* node to identify potential *Worker* nodes, to manage them and to report their resource usage to the next higher layer. It can achieve these tasks by utilizing four daemons, namely, a `coordinatord`, `architectured`, `collectord`, and `queryd` daemon. Each background process has certain responsibilities. The `coordinatord` manages the communication between the *Collector* and multiple *Workers* with the help of specialized messages. It also synchronizes the workflows of the *Collector* using multiple intra- and inter-processes communication channels (i.e., queues). The `architectured` remotely "terraforms" the registered *Workers* by invoking their customized API endpoints. The `collectord` and `queryd` daemon measure together various machine resources and their metrics, e.g., CPU, disk, memory utilization, generate corresponding reports and transfer the latter to specialized upstream applications for further usage. To bootstrap those complicated daemons, the *Collector* makes use of a Docker [5] specification file *config.yml* to define required configurations, e.g., opening port numbers, API endpoint, Docker options, etc.
- A *Worker* node is the basic computational entity of the fog computing system. As the name implies, this type of computing node does the heavy lifting. It has two daemons, the `workerd` and `builderd`. Similar

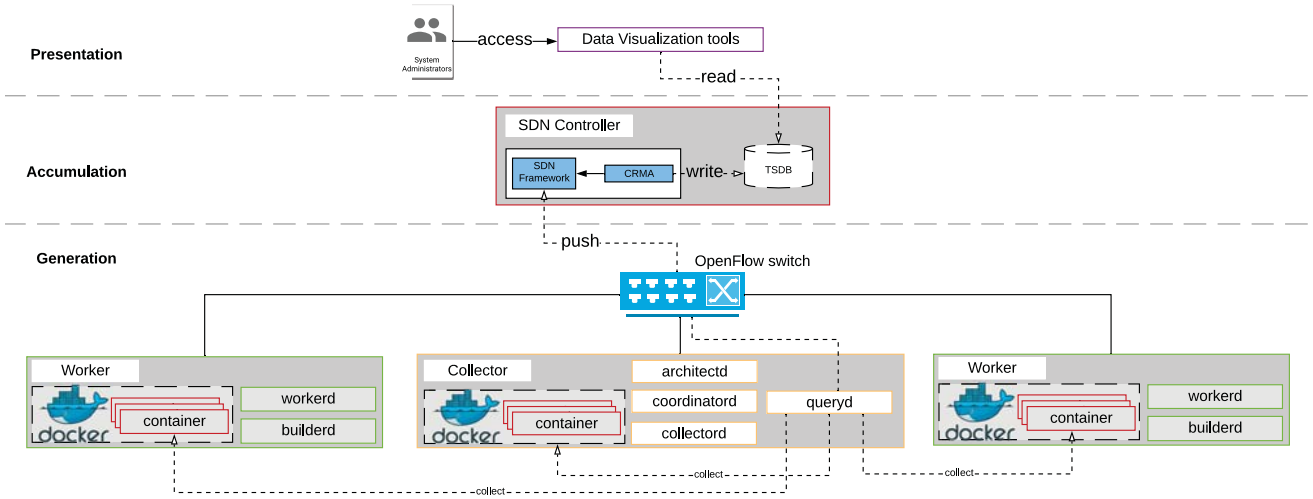

Fig. 1. Software architecture of the realized Cloudless Resource Monitoring System

to the *Collector*, each daemon has separated tasks. The `workerd` communicates with the `coordinatord`, and registers itself to the monitoring system, whereas the `builderd` handles the HTTP requests made by the `architectured` daemon of the *Collector*. Then it executes various operations to alter the local Docker environment, [5]. The *Worker* also employs a *config.yml* file to store the configurations required by its two main daemons.

B. The Accumulation Layer

The *Accumulation* layer contains two other important components of the proposed CRMS, the CRM application *CRMA* and an NoSQL database (DB) such as a time series DB (*TSDB*) like `Influxdb`. The CRM application is an SDN networking application designed with explicit objectives, namely, to intercept certain kinds of packets emitted by the *Collector* node in the lower layer, to perform modifications, if required, to those captured data payloads, and then finally to stockpile them into a NoSQL database system, particularly a time-series DB, for further usages such as data analytics and visualization. The rationale behind favoring a time-series database system in this scenario is conspicuous. A relational database is useful when presenting the current state of data, but the CRMS is required to exhibit the history of the resource consumptions in the fog computing system over time, rather than at a fixed moment. In this respect these two services of this *Accumulation* layer materialize the *Monitoring* function. The latter is one of the four pillars of an MAPE-K architecture which is the foundation of our research proposal. Here the Java application *CRMA* is the main virtualized engine of our approach. It is specifically developed to integrate itself into an *ONOS* control subsystem. Due to the nature of this proposed system in which data are mostly utilized to track changes of multiple types of the resource usage in all computing nodes over a period of time the other service is given by the *TSDB*. It is more pertinent than a relational database in this situation. Several *TSDB* candidates, such as `Influxdb`, `TimescaleDB`[1], or `QuestDB`[2] present a lot of potential. After multiple exhaustive evaluations, `Influxdb`, the de-facto leader of such a *TSDB*, has been chosen as our solution since it can handle high data velocity. It also provides a unique query language called InfluxQL, that allows users to query, extract, and aggregate useful information quickly and precisely.

C. The Presentation Layer

Lastly, the *Presentation* layer resides on top of the two mentioned layers and materializes a data visualization tool for the graphical representation of the resource usage information as final component of our CRMS. Using visual elements such as charts or graphs, system administrators can quickly observe and discover the latest trends or patterns of the resource usage occurring among multiple fog nodes in the underlying infrastructure. They can also identify outliers which are realized in the underlying fog computing system. There are several available open-source data visualization tools, e.g., Kibana[3], `Grafana`, or Graphite[4]. After carefully considering all the pros and cons of those tools, `Grafana` serves on the first place of our implementation approach. Generally, it allows users to generate comprehensive dashboards utilizing multiple visual elements, e.g., line graph, table, singlestat, combined with many customized options, being completely versatile. Furthermore, `Grafana` can collaborate with multiple types of a database system supported by built-in data-source plugins, such as `Influxdb`, PostgreSQL, Elasticsearch, Prometheus, and so forth.

[1]https://www.timescale.com/
[2]https://questdb.io/
[3]https://www.elastic.co/kibana
[4]https://graphiteapp.org/

III. Conclusions

In this paper we have discussed the design and development of a virtualized, context-aware, self-configuring orchestration system interworking with an SDN/NFV environment of a fog computing system. In particular, we have described the hierarchical layering and functionality within the underlying fog nodes to generate the working prototype of an intelligent, lightweight self-managed orchestrator for advanced IoT applications and services. The latter Cloudless Resource Monitoring System (CRMS) has the capability of automatically monitoring various performance aspects with respect to the resource allocation among multiple hosts of a fog computing system that are interconnected by SDN. As intended, it is realizing the basic element "M" of the MAPE-K principles employed in autonomic computing, see also [3], [7].

Some computational aspects of the proposed CRMS will be enhanced in the near future to improve the performance of the invoked communication patterns among the involved software components of the interconnected fog nodes.

Regarding the planning and performance management of fog computing systems such as the optimal placement of virtualized network functions or an intelligent, state-dependent scheduling of virtualized functions in smart IoT applications, it is possible to easily integrate the collected performance data which are stored in the time series data base into those more advanced planning procedures due to the layered structure of the Cloudless Resource Monitoring System. In this respect, CRMS offers a high potential for additional, more complex analysis and planing phases based on machine learning and advanced data analytics techniques such as an incremental tensor decomposition scheme. Related investigations employing federated learning schemes for geographically distributed, automatically monitored fog computing nodes are a subject of our current research. They can reveal the full potential of the adopted MAPE-K model.

Finally, we want to stress that a virtualized demonstrator of CRMS is available under Containernet[5] on demand.

References

[1] S. Patidar, D. Rane, and P. Jain, "A survey paper on cloud computing." In: 2012 Second International Conference on Advanced Computing Communication Technologies, pp. 394–398, 2012.

[2] F. Bonomi and R. Milito, "Fog computing and its role in the internet of things." In: Proceedings of the MCC Workshop on Mobile Cloud Computing, pp. 13–16, Aug. 2012.

[3] IBM, "An architectural blueprint for autonomic computing." Autonomic Computing, White Paper, Third Edition, June 2005. URL: https://www-03.ibm.com/autonomic/pdfs/ACBlueprintWhitePaperV7.pdf

[4] M. Eder, "Hypervisor- vs. container-based virtualization." Technical University of Munich, 2016.

[5] Docker: "Reference documentation." URL: https://docs.docker.com/reference/, https://docs.docker.com/

[6] R. Buyya, S. Narayana Srirama, "Internet of Things (IoT) and New Computing Paradigms." In: R. Buyya, S. Narayana Srirama (eds.), Fog and Edge Computing: Principles and Paradigms. John Wiley & Sons, Ltd, Chap. 1, pp. 1–23, 2019.

[7] E. Casalicchio, "Container Orchestration: A Survey." In: A. Puliafito, K. Trivedi (eds.), Systems Modeling: Methodologies and Tools. EAI/Springer Innovations in Communication and Computing. Springer, Cham, 2019.

[8] M. Großmann and C. Schenk, "A Comparison of Monitoring Approaches for Virtualized Services at the Network Edge." In: 2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC), pp. 85–90, 2018.

[9] W. Stallings, "Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud." Addison-Wesley Professional, 2016.

[10] A. Feldmann, "Internet clean-slate design: what and why?" ACM SIGCOMM Computer Communication Review, vol. 37, no. 3, pp. 59–64, July 2007.

[11] J. Rexford, C. Dovrolis, "Future internet architecture: clean-slate versus evolutionary research." Commun. ACM, vol. 53(9), pp. 36–40, Sep. 2010.

[12] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey." Proceedings of the IEEE, vol. 103, no. 1, pp. 14–76, 2015.

[13] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations." IEEE Communications Magazine, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[14] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog Orchestration for Internet of Things Services." IEEE Internet Computing, vol. 21, no. 2, pp. 16–24, Mar.–Apr. 2017.

[15] M. Iorga, L. Feldman, R. Barton, M.J. Martin, N.S. Goren, and C. Mahmoudi, "Fog Computing Conceptual Model." NIST, Special Publication (NIST SP) - 500–325, March 2018.

[5] https://github.com/containernet/containernet