Towards a Framework for Data Stream Processing in the Fog

Thomas Hießl · Christoph Hochreiner
Stefan Schulte

Introduction

The advent of the Internet of Things (IoT) has contributed to the ever-growing equipment of personal, public, and business spaces with ubiquitous devices which generate, process, and consume data [1, 3]. Examples of the rising advent of the IoT can be found in areas like smart factories [15], smart cities [16, 21], or smart healthcare [7]. In all these areas, a potentially very large number of IoT devices might be used to sense and actuate, leading to very large volumes of data being produced. Very often, IoT data is generated as a series of data items, i. e., data streams, making it necessary to provide software solutions which are able to process these data streams in (near) real-time instead of processing the data in batches [20].

Since the IoT and smart systems are inherently volatile, with entities leaving or entering a system at all times, or the data volumes produced by the single devices changing frequently, one particular challenge in data stream processing is the allocation of an appropriate amount of computational resources (e. g., CPU, memory, storage resources) in an on-demand fashion [12]. Today, this is mostly solved by using cloud-based computational resources in the form of virtual machines (VMs) or containers [26].

As an alternative, the exploitation of already existing computational resources in the IoT (e. g., provided by networking equipment, cyber-physical systems (CPS), or powerful sensor nodes) to deploy software services has recently gained much attention by the research community and the industry. The combination of IoT-based computational resources at the edge of the network with cloud-based data processing is known as *fog computing* [4, 8], which is defined as "a system-level horizontal architecture that distributes resources and services of computing, storage, control and networking anywhere along the continuum from cloud to things" [18]. Accordingly, computational resources at the edge of the network include IoT devices as well as devices in field area networks [4].

In short, fog computing provides a conceptual approach for virtualizing and orchestrating computational, network, and storage capabilities in the IoT and in the cloud. It does so by providing IoT-based computational resources in a similar manner as physical resources are offered as VMs or containers in the cloud [8]. Fog computing is seen as a basic building block in future smart systems where data (pre-)processing or data filtering should be done in the vicinity of data sources, while big data tasks and tasks which are in general too compute intensive to be executed on IoT devices are offloaded to the cloud [25].

The benefits arising from the enactment of stream processing on fog devices are manifold, especially compared to stream processing in the public cloud. Example use cases for fog-based stream processing are, e. g., urban surveillance [6], industrial automation, and mobile crowdsensing [26]. In the following, we will briefly discuss the benefits of fog-based stream processing using these example use case scenarios.

https://doi.org/10.1007/s00287-019-01192-z

Thomas Hießl · Christoph Hochreiner · Stefan Schulte
Distributed Systems Group, TU Wien,
Vienna, Austria
E-Mail: s.schulte@infosys.tuwien.ac.at

Abstract

In volatile data streams as encountered in the Internet of Things (IoT), the data volume to be processed changes permanently. Hence, to ensure timely data processing, there is a need to reconfigure the computational resources used for processing data streams. Up to now, mostly cloud-based computational resources have been utilized for this. However, cloud data centers are usually located far away from IoT data sources, which leads to an increase in latency since data needs to be sent from the data sources to the cloud and back. With the advent of fog computing, it is possible to perform data processing in the cloud as well as at the edge of the network, i. e., by exploiting the computational resources offered by networked devices. This leads to decreased latency and a lower communication overhead. Despite this, there is currently a lack of approaches to data stream processing which explicitly exploit the computational resources available in the fog.

Within this paper, we consider the usage of fog-based computational resources for the purposes of data stream processing in the IoT. For this, we introduce a representative application scenario in the field of Industry 4.0 and present a framework for stream processing in the fog.

First, by processing data streams (partially) at the edge of the network, it is possible to decrease the latency and communication overhead, since data items do not have to be sent to centralized cloud data centers [8, 28]. This is an important feature in areas where data needs to be processed very quickly, e. g., as part of online monitoring for decision-making purposes in industrial automation. Second, communication efforts in general are reduced, leading to lower bandwidth consumption, since data does not have to be sent to the large data centers that constitute the public cloud. For instance, in mobile participatory sensing settings where a very large number of mobile clients are involved, this may lead to less load on the telecommunication network. Third, fog devices are very often located on the premises of the data owners. Thus, if applying fog computing, data does not have to leave the premises of the data owner, or the amount of data to be sent to the cloud is at least reduced. This leads to improvements in data privacy and data protection. However, even in more public scenarios, fog computing can lead to improved data privacy. For instance, in the case of urban traffic surveillance, car plates may be analyzed in the fog, and only if a sought-after car is identified its data is forwarded to the authorities, while data about other cars is discarded after initial analysis. Fourth, the utilization of already existing hardware at the edge of the network may even lead to reduced cost, since there is no need to obtain resources from a public cloud or to set up a private cloud. This is especially an option in scenarios where unused computational resources are already available, e. g., in CPS on a factory shop floor.

The discussed benefits of fog computing also apply to the field of data analytics, and therefore to data stream processing: Stream processing applications deployed at the edge of the network are suitable to achieve low latency or real-time data analysis, while cloud-based computational resources are well suited for longer-running data analysis batch processes [4].

Despite this conceptual work on the application of fog computing in data stream processing, there is to the best of our knowledge still a lack of concrete technical solutions. Therefore, within this paper, we discuss how fog computing can be applied in data stream processing. For this, we first give a general introduction to data stream processing in the fog in the next section and discuss the current state of the art. Afterwards, we briefly present a framework for fog-based distributed stream processing in industrial settings.

Data stream processing in the fog

In order to discuss the core entities in data stream processing in the fog, we apply the simplified scenario shown in Fig. 1. The figure shows the different entities in a smart system, e. g., a smart factory. In IoT-based data stream processing scenarios, the data streams are generated by single sensors or groups of sensors, as indicated at the bottom of the figure. These data streams consist of continuously generated data items [2].

Data stream processing

Stream processing operators (SPOs) are software entities that consume one or more data streams as data inputs, transform the data, and gener-

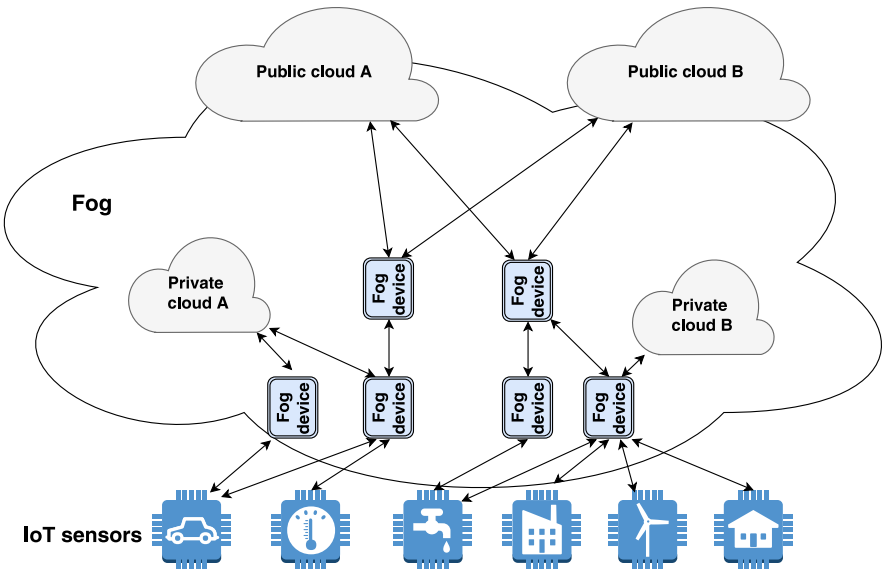

Fig. 1 Example hierarchy of cloud and edge resources in an IoT/fog scenario

ate one or more output streams. Together, the SPOs in a system and the connections between the SPOs constitute a stream processing topology, with the connections defining the data flow between the SPOs. In general, SPOs are able to process *structured* data, which follow a well-defined data schema, *semi-structured* data, for which no schema are available, but for which the data can be extracted from context data or metadata, and *unstructured* data, for which this is not possible, e. g., video or audio streams, binary-encoded data, etc. [2]. The software managing the deployment and execution of SPOs is named stream processing engine (SPE). For instance, Apache Storm[1] or Heron[2] are well-known SPEs.

Deployment of stream processing operators in the fog

Usually, cloud-based computational resources are leased and released to host SPO instances. The amount of cloud-based resources to be leased should be based on the actual resource demand [10]. Thereby, SPOs may be operated both on private as well as on public cloud hosts. The deployment of SPOs on a set of hosts is also known as operator placement. The decision where to place SPOs can be made during the design time of a stream processing topology, but also during runtime [13].

[1] https://storm.apache.org
[2] https://apache.github.io/incubator-heron/

Figure 1 also shows that – apart from the public cloud or a potentially available private cloud – in a typical IoT scenario, further computational resources exist at the edge of the network, provided by so-called *fog devices*. Fog devices are all devices which are located within the network or at the edge of the network and could become part of the fog. From a hardware perspective, fog devices can be very heterogeneous. Typical examples are powerful sensor nodes, routers, switches, single-board computers like Raspberry Pis, or even resource-rich computers providing cloudlet capabilities (e. g., Amazon's AWS Snowball Edge or TTTech's Nerve) [28]. With regard to data stream processing, the most important capability of these fog devices is the provisioning of virtualized computational resources, which can then be used to host arbitrary software (here: SPOs). In the example scenario depicted in Fig. 1, fog devices could be used in order to pre-filter or pre-process data, thus decreasing the amount of data that needs to be processed using the private or public cloud resources, or even completely avoiding that data needs to be sent to the cloud.

Current state of the art

Despite the potential benefits discussed above, the number of approaches explicitly aiming at data stream processing in the fog is still quite limited, i. e., the computational resources offered by fog devices are mostly neglected today. Nevertheless, there are some examples where

the exploitation of IoT-inherent computational resources is proposed.

For instance, Sajjad et al. [24] discuss the utilization of cloud-based and near-the-edge data centers to decrease latency and bandwidth consumption in data stream processing. For this, the authors apply the notion of data centers, while fog computing allows to also address computational resources on a more fine-grained level. A similar approach is presented by Renart et al. [23]. Here, the authors make use of an overlay network to orchestrate geographically distributed computational resources. Yassine et al. [27] discuss the usage of fog-based data analytics (which includes data stream processing) for smart homes. While in the work at hand, we focus on a framework to conduct data stream processing in the fog, Yassine et al. focus on particular data analytics methods. In a very recent approach, Dautov et al. discuss the clusterization of edge devices for data stream processing [9]. While the authors focus on rather powerful edge devices, this is nevertheless the approach that comes closest to the work at hand. Finally, Cardellini et al. [5] present a fine-grained approach to allocate computational resources for stream processing in the fog in an elastic manner.

Also, some approaches applying complex event processing (CEP) in the fog have been proposed, e. g., [19]. In general, CEP is similar to stream processing with regard to the need to provide results without undue delays and the processing of input data streams [2, 28]. However, CEP systems usually provide only limited support for the handling of unstructured data and apply a rule-based programming model different from the exploratory programming model applied in stream processing [2]. Despite this, there are some similarities regarding the utilization of fog-based computational resources in stream processing and CEP which should be taken into account.

There is – to the best of our knowledge – no framework for data stream processing in the fog which explicitly facilitates the integration of fine-grained computational resources at the edge of the network. Such a framework allows us to not only apply basic data stream processing capabilities in the fog, but could also serve as the foundation for pursuing advanced research questions. These questions cover the already established research field of optimal SPO placement across different geographic locations, but also the development of complex key performance indicators (KPIs), which could serve as a foundation for SPO placement instead of the usually applied low-level metrics like CPU or memory utilization. Also, fog computing allows taking into account data ownership and data privacy. By defining data models or a domain-specific language which allow to detail if individual data items or data types should be processed in the cloud or at particular computational resources at the edge of the network, data privacy could be enforced on a fine-grained level. In addition, there is currently a lack of complex approaches to fault tolerance in stream processing. Such approaches need to go further than the usually applied notion of active replication on the level of single SPOs. Instead, the relationships between the single operators in a topology need to be taken into account in order to avoid backpressure and ripple effects.

The Vienna platform for elastic stream processing

IoT manufacturing scenario

In order to further exemplify the discussion from the last section, we consider a simplified scenario from the field of Industry 4.0 in Fig. 2. This scenario is based on our work in EU H2020 RIA CREMA[3]. Notably, this smart factory scenario is used for illustration purposes only and does not limit the discussion in this paper to the field of Industry 4.0 or the entities depicted in the figure. Instead, fog-based data stream processing could be also applied in other scenarios where data is generated continuously, e. g., smart cities, smart healthcare, or smart grids, as discussed above.

As can be seen in the figure, we consider a manufacturer with two smart factories, which are located in different countries. For the purpose of our simplified scenario, we assume that *Smart Factory 1* operates two CPS in terms of manufacturing assets. These CPS offer sensor and actor capabilities, computational capabilities, and are connected to a network, in order to provide monitoring, coordination, control, and integration of the operations offered by the CPS [22]. One particular functionality of the CPS is to provide a (real-time) feedback loop for its operations [17]. *Smart Factory 2* operates one CPS, which is again a manufacturing asset. Notably,

[3] http://www.crema-project.eu

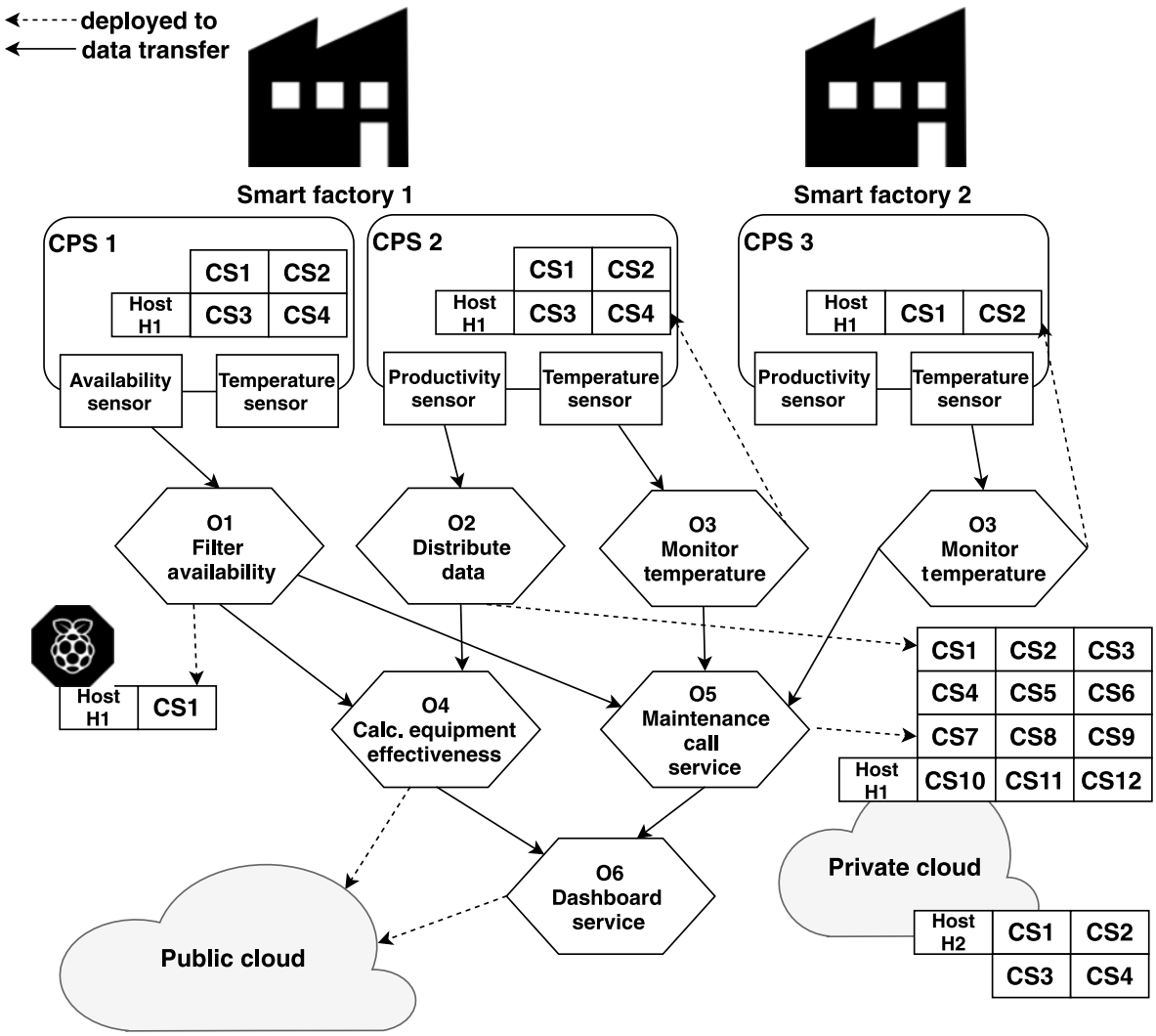

Fig. 2 Example distributed stream processing in smart factories

the "system of cyber-physical systems" depicted in Fig. 2 is itself a CPS.

Each CPS features a number of sensors (e. g., availability, temperature, and frequency sensors), which provide important input for decision support. The CPS sensors emit data items regularly and with a high frequency, i. e., the CPS are data stream sources. Therefore, it is necessary to process the data using the capabilities of an SPE. We assume that a data schema is available for all data streams, i. e., that the SPOs process structured data.

Figure 2 shows the SPOs *O*1–*O*6 which are used for data stream processing. In brief, the SPOs are orchestrated to derive several KPIs of production processes from the different sensors of CPS 1–3. If irregularities are detected, maintenance actions are initiated to avoid serious production problems. This is also done via an SPO.

The company is able to deploy SPOs on a variety of different computational resources. In the figure, hosts provide these resources in terms of software container slots (CS). It should be noted that for now, we assume that all slots provide the same amount of computational resources. Of course, this is in reality not the case, since different SPOs running in containers need different amounts of computational resources. We will discuss this in more detail below.

CPS 1–3 possess the capability to host SPOs, however, only to a certain extent, since the computational resources offered by the CPS are rather

limited. Therefore, some SPOs may be hosted on the CPS themselves (in the figure: two instances of *O*3), while others need to be hosted in the cloud. In addition to the CPS and cloud resources, both factories possess further computational resources, e. g., single-board computers, as depicted for Smart Factory 1 (hosting *O*1). CPS-inherent computational resources and single-board computers are fog devices as depicted in Fig. 1.

The company also owns a private cloud, which can be used to host SPOs. This private cloud is not located in one of the two smart factories but still within the premises of the company. Therefore, hosting SPOs in the private cloud may lead to higher communication delays and is, therefore, preferable for SPOs that are not very delay sensitive. In Fig. 2, the private cloud hosts *O*2 and *O*5, but the delay sensitive *O*1 and both *O*3 instances are hosted on the CPS and on a single-board computer close to the CPS, respectively. Last but not least, the company could also deploy SPOs in the public cloud, e. g., if the amount of computational resources is otherwise not sufficient, if the fog devices or the private cloud are not available, or if SPOs are shared with external partners. In Fig. 2, *O*4 and *O*6 are hosted in the public cloud.

The placement of the SPOs is the duty of an SPE which is able to take into account the heterogeneous computational resources offered. The SPE also makes sure that the data flow between different SPOs is realized and SPO instances are replicated, if necessary. Different constraints may play a role during SPO placement, since the deployment locations of the SPOs may vary according to the current situation in the computational infrastructure (e. g., latency, availability) as well as other constraints, e. g., data privacy demands – we will discuss these in more detail below. For instance, as can be seen in Fig. 2, two different instances of *O*3 are hosted on CPS 2 and CPS 3, respectively, since latency plays a large role when monitoring the temperature, e. g., in order to be able to shut down a CPS very quickly. However, the data is also used to derive more sophisticated KPIs and, therefore, needs to be forwarded to further SPOs. Importantly, the optimal placement of SPOs on heterogeneous fog resources may change over time, e. g., since the data volume or the location of data sources may change in volatile IoT systems, or because further SPOs are added to a stream processing topology [11]. Hence, an SPE needs to be able to find an initial placement *and* to replan during the runtime of a topology. In the next section, we will present the Vienna Platform for Stream Processing (VISP), which meets these requirements.

Fog-based data stream processing with the Vienna Platform for Elastic Stream Processing

VISP is a fully-fledged research SPE for the fog. Most importantly, the framework is able to place SPOs on fog devices as well as in the cloud[4].

As can be seen in Fig. 3, there are two major components in the VISP Ecosystem, namely the *VISP Runtime* and the *Computational Resources*. The VISP Runtime facilitates the core SPE functionalities of VISP, i. e., it provides the means for instantiation, execution, and monitoring of stream processing topologies down to the level of single SPOs.

For this, the VISP Runtime is able to pull software images from a repository (not depicted in Fig. 3) in order to instantiate new SPOs. Furthermore, the VISP Runtime has an *Elasticity* component which provides the means for monitoring the usage of computational resources. Based on this monitored data and information about the topology, VISP's *Reasoner* analyzes whether the system should scale in or out, i. e., deploy SPOs on additional fog resources or to undeploy existing SPOs since they are not needed any longer, should utilize further computational resources, or should remain unchanged. Last but not least, the VISP Runtime provides the means for the management of SPO instances: VISP is able to place SPO instances on arbitrary computational resources, as long as the according hosts are able to host SPOs which are offered as containerized software, e. g., Docker containers. As can be seen in the figure, the SPO instances themselves are hosted on the Computational Resources. SPOs provide the actual processing logic which is applied to the incoming data streams, as well as system and failure monitors to track the non-functional behavior of the respective SPO instance. Also, each SPO instance is provided with a configuration API. For a more detailed discussion of VISP, we refer to [14]. A first approach to achieve optimal placement of SPOs using VISP is presented in [11].

[4] Further details on VISP as well as the software are available at https://github.com/visp-streaming.

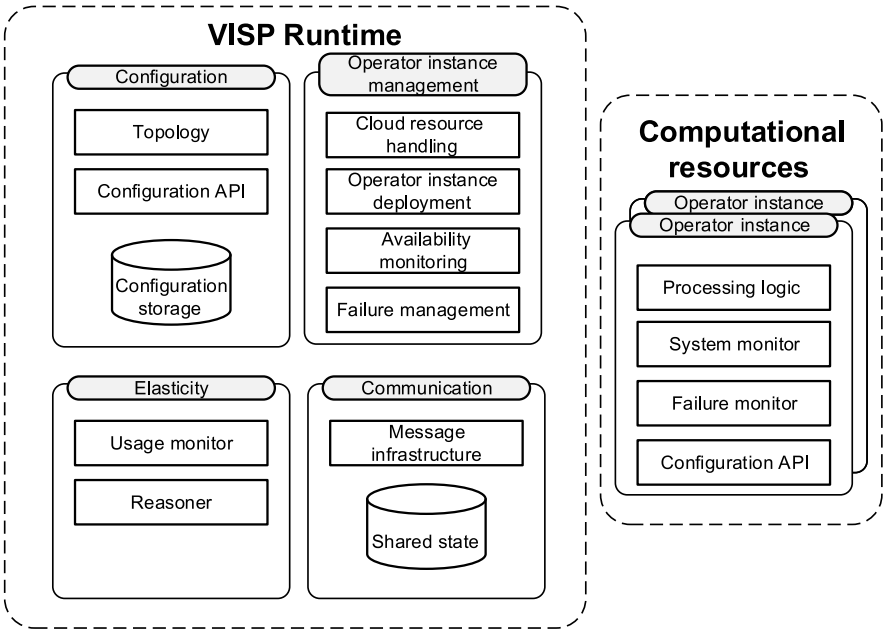

Fig. 3 VISP Ecosystem

In order to be able to allocate computational resources to particular SPOs, VISP considers *resource pools*. Figure 4 provides an example of the mapping from resource pools to computational resources, which can then be used by VISP for the deployment of SPO instances. In the figure, we consider a simplified version of the scenario discussed above. For this, we consider CPS 1, CPS 3, and the private cloud from Fig. 2, which become resource pools of different sizes (indicated by the size of the respective boxes).

In general, a resource pool could be a fog device, a private cloud, or a public cloud, as shown in Fig. 2. This allows VISP to apply a technology-agnostic approach with regard to the computational resources, as long as the resources are virtualized and able to host a container. The size of a resource pool in Fig. 2 refers to the resource capacities available, like number of CPU cores, memory (RAM), or storage.

Figure 4 shows the two steps of mapping a resource pool: After the first step, each pool is partitioned into equally-sized slots *CS*. In order to better differentiate between these slots, we have added the ID of the resource pool to the name of the slot, i. e., $CS_{1,1}$ identifies the first slot within Resource Pool 1 (CPS 1), etc. Each slot is considered to host one container, which in turn runs one particular SPO instance. For this, each slot at this stage is allocated with the capacity of a specified reference host, e. g., 0.2 cores, 1 GB RAM, and 300 MB storage. The concrete numbers are based on the available resources. For instance, in the scenario used here, CPS 1 and CPS 3 only provide rather limited computational resources, as can be seen by the limited number of slots offered by these machines, while the private cloud provides a larger number of slots.

Due to the fact that VISP places SPOs with different resource demands in these slots, the provided resource capacity of one slot might not suffice for an SPO instance that has to bear higher loads. Therefore, slots with more computational power and storage are required. This can be achieved by mapping the equally partitioned slots in a second step to larger partitions, as depicted at the bottom of Fig. 4. For instance, in the example depicted in the figure, slots of *small*, *medium*, and *large* size are provided, whereas each category uses a fixed portion of the overall capacity. Again, this is just an example, since additional classes of slots might be introduced, based on the demands of the stream processing system and the available computational resources in the fog.

It should be noted that the depicted resource pool only provides an abstracted and very simplified view on the computational resources available in the fog. In fact, the diversity of available devices in the fog makes it necessary to provide means to benchmark the performance and to evaluate the

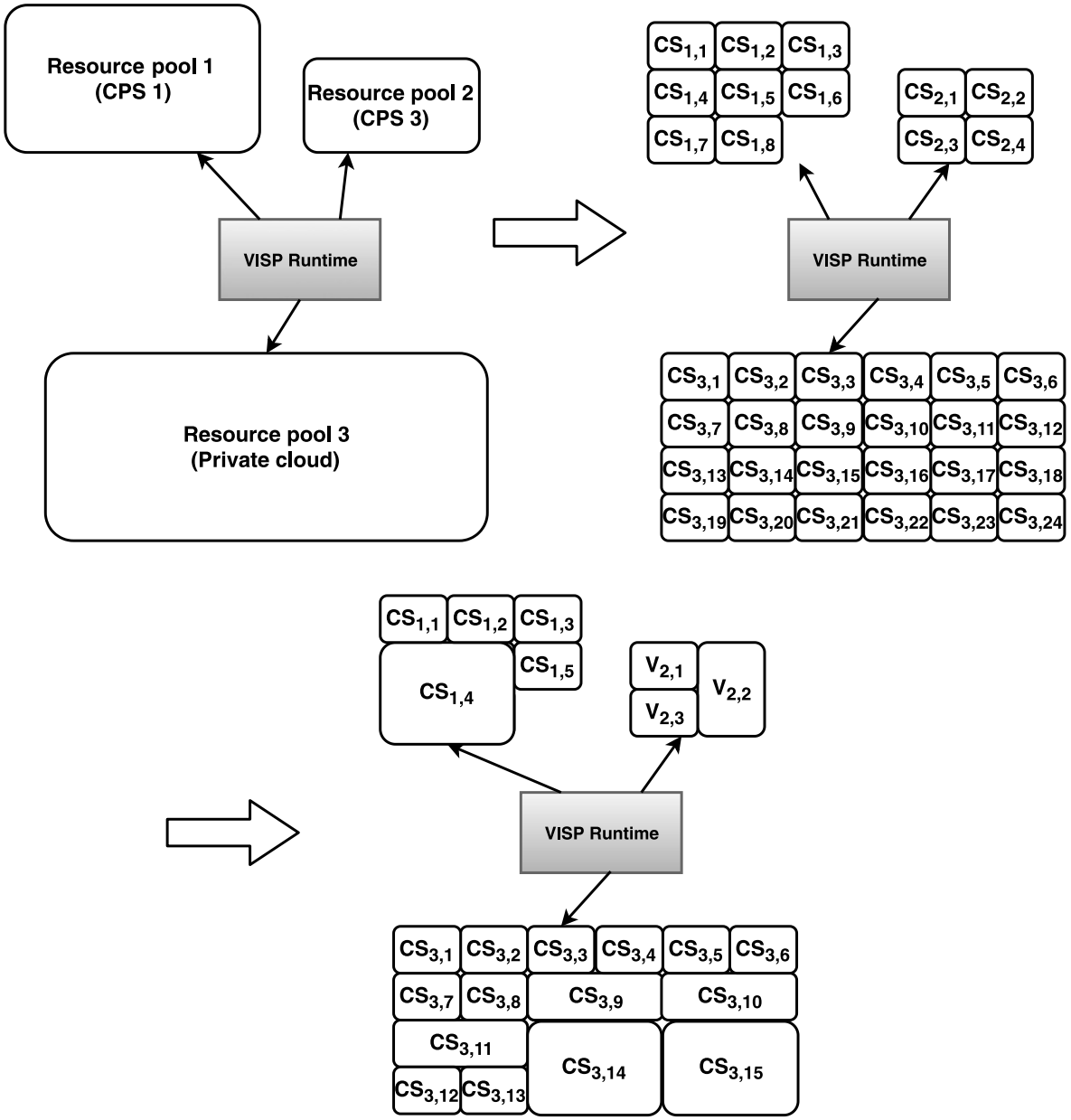

Fig. 4 Resource Graph in VISP

capabilities of fog devices, which is currently still an open issue.

Obviously, one central functionality that needs to be provided by the VISP Reasoner is to find an optimal placement of SPO instances on computational resources, based on the resource demands of the SPOs and the quality of service demands of the data stakeholders. Examples for the attributes that need to be taken into account during computation of a placement are, e. g.:

– **Operator and network latency**
This attribute influences the processing duration within stream processing topologies. Hence, by optimizing placements such that SPOs are executed on resources with appropriate computational power, the processing performance can be maximized. Furthermore, the response time can be improved by avoiding long network distances, which cause delays when data is streamed between the SPOs and from/to data sources and data sinks.

– **Data privacy**
  In order to improve data privacy, data stream processing should allow defining which data items should be processed or stored on private and public resources. This obviously influences the SPO and network latency as well as the required types of computational resources, and therefore also becomes a factor when finding an optimal SPO placement.
– **Execution cost of using the underlying infrastructure**
  SPOs can be deployed to decentralized hosts that provide their computing power at different cost or even with different cost models. While for privately-owned resources sunk cost and energy cost may have to be taken into account, public cloud providers offer different cost models which have to be regarded.
– **Resource availability**
  SPOs can be placed at different locations and hosts with varying availability. Especially in a heterogeneous environment like the IoT, the differences in availability can be significant. Monitoring the online status of the involved cloud and fog resources is the basis for making optimal decisions to ensure the availability of the overall topology. Therefore, this data has to be considered during the computation of an optimal placement.
– **Overhead of operator replacement**
  In order to optimize the criteria above, replacements in terms of deploying, redeploying, and undeploying SPOs have to be executed. These replacements lead to overheads and cost for, e. g., temporary downtimes and data transfer. This has to be considered to limit the extra expenditure and save overall cost.

Conclusion

This paper provides a general introduction to data stream processing in the fog. To this end, we presented basic concepts of data stream processing, discussed an exemplary scenario from the manufacturing domain, and presented our work on an SPE for distributed stream processing in the fog, i. e., the Vienna Platform for Elastic Stream Processing (VISP).

While VISP is a fully-fledged research SPE, we primarily consider it as a starting point for future research in the field of fog-based data stream processing, aiming at the research questions mentioned in the discussion of the related work. Notably, VISP's software components as shown in Fig. 3 provide open interfaces and are loosely coupled. This makes it possible to easily replace the underlying functionalities of the framework. Hence, researchers can use VISP in order to evaluate novel approaches to fault tolerance, monitoring based on KPIs, to take into account data ownership, and for the evaluation of other functionalities needed in fog-based data stream processing.

Acknowledgements

This work is partially funded by COMET K1, FFG – Austrian Research Promotion Agency, within the Austrian Center for Digital Production and by the Commission of the European Union within the CREMA H2020-RIA project (Grant agreement no. 637066).

Funding. Open access funding provided by TU Wien (TUW).

References

1. Al-Fuqaha AI, Guizani M, Mohammadi M, Mohammed Aledhari M, Ayyash M (2015) Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Commun Surv Tutor 17(4):2347–2376
2. Andrade H, Gedik B, Turaga D (2014) Fundamentals of Stream Processing. Cambridge University Press
3. Atzori L, Iera A, Morabito G (2010) The internet of things: A survey. Comput Networks 54:2787–2805
4. Bonomi F, Milito R, Natarajan P, Zhu J: Fog computing: A platform for internet of things and analytics. In: Bessis N, Dobre C (eds) Big Data and Internet of Things: A Roadmap for Smart Environments, Studies in Computational Intelligence, vol 546. Springer, pp 169–186
5. Cardellini V, Lo Presti F, Nardelli M, Russo Russo G (2018) Decentralized self-adaptation for elastic data stream processing. Future Gener Comp Sy 87:171–185
6. Chen N, Chen Y, You Y, Ling H, Liang P, Zimmermann R: Dynamic Urban Surveillance Video Stream Processing Using Fog Computing. In: 2016 IEEE Second International Conference on Multimedia Big Data. IEEE, pp 105–112
7. Cortés R, Bonnaire X, Marin O, Sens P (2015) Stream Processing of Healthcare Sensor Data: Studying User Traces to Identify Challenges from a Big Data Perspective. In: 4th International Workshop on Body Area Sensor Networks, Procedia Computer Science, vol 52. Elsevier, pp 1004–1009
8. Dastjerdi AV, Gupta H, Calheiros RN, Ghosh SK, Buyya R (2016) Fog computing: Principles, architectures, and applications. In: Buyya R, Dastjerdi AV (eds) Internet of Things: Principles and Paradigms, chap 4. Morgan Kaufmann, pp 61–75

9. Dautov R, Distefano S, Bruneo D, Longo F, Merlino G, Puliafito A (2018) Data processing in cyber-physical-social systems through edge computing. IEEE Access 6:29822–29835
10. Heinze T, Roediger L, Meister A, Ji Y, Jerzak Z, Fetzer C (2015) Online parameter optimization for elastic data stream processing. In: Sixth ACM Symposium on Cloud Computing. ACM, pp 276–287
11. Hießl T, Karagiannis V, Hochreiner C, Schulte S, Nardelli M (2019) Optimal placement of stream processing operators in the fog (forthcoming). In: 3rd IEEE International Conference on Fog and Edge Computing. IEEE
12. Hochreiner C, Schulte S, Dustdar S, Lécué F (2015) Elastic stream processing for distributed environments. IEEE Internet Comput 19:54–59
13. Hochreiner C, Vögler M, Schulte S, Dustdar S (2017) Cost-efficient enactment of stream processing topologies. PeerJ Comput Sci 3:e141
14. Hochreiner C, Vögler M, Waibel P, Dustdar S (2016) VISP: An Ecosystem for Elastic Data Stream Processing for the Internet of Things. In: 20th International Enterprise Distributed Object Computing Conference. IEEE, pp 19–29
15. Jeschke S, Brecher C, Meisen T, Özdemir D, Eschert T (2017) Industrial internet of things and cyber manufacturing systems. In: Jeschke S, Brecher C, Song H, Rawat DB (eds) Industrial Internet of Things: Cybermanufacturing Systems. Springer, pp 3–19
16. Kolozali S, Bermúdez-Edo M, Puschmann D, Ganz F, Barnaghi PM (2014) A Knowledge-Based Approach for Real-Time IoT Data Stream Annotation and Processing. In: 2014 IEEE International Conference on Internet of Things. IEEE, pp 215–222
17. Lee EA (2010) CPS Foundations. In: 47th Design Automation Conference. IEEE, pp 737–742
18. OpenFog Consortium (2018) IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing. IEEE Std 1934-2018
19. Ottenwälder B, Koldehofe B, Rothermel K, Ramachandran U (2013) MigCEP: Operator Migration for Mobility Driven Distributed Complex Event Processing. In: 7th ACM International Conference on Distributed Event-Based Systems. ACM, pp 183–194
20. Perera C, Zaslavsky AB, Christen P, Georgakopoulos D (2014) Context aware computing for the internet of things: A survey. IEEE Commun Surv Tutor 16(1):414–454
21. Puiu D, Barnaghi PM, Toenjes R, Kuemper D, Ali MI, Mileo A, Parreira JX, Fischer M, Kolozali S, Farajidavar N, Gao F, Iggena T, Pham T, Nechifor C, Puschmann D, Fernandes J (2016) CityPulse: Large scale data analytics framework for smart cities. IEEE Access 4:1086–1108
22. Rajkumar R, Lee I, Sha L, Stankovic J (2010) Cyber-Physical Systems: The Next Computing Revolution. In: 47th Design Automation Conference. IEEE, pp 731–736
23. Renart E, Diaz-Montes J, Parahsar M (2017) Data-driven Stream Processing at the Edge. In: IEEE 1st International Conference on Fog and Edge Computing. IEEE, pp 31–40
24. Sajjad HP, Danniswara K, Al-Shishtawy A, Vlassov V (2016) SpanEdge: Towards Unifying Stream Processing over Central and Near-the-Edge Data Centers. In: IEEE/ACM Symposium on Edge Computing. IEEE, pp 168–178
25. Stojmenovic I, Wen S (2014) The Fog Computing Paradigm: Scenarios and Security Issues. In: 2014 Federated Conference on Computer Science and Information Systems. IEEE, pp 1–8
26. Yang S (2017) IoT stream processing and analytics in the fog. IEEE Commun Mag 55:21–27
27. Yassine A, Singh S, Hossain MS, Muhammad G (2019) IoT big data analytics for smart homes with fog and cloud computing. Future Gener Comp Sy 91:563–573
28. Yi S, Li C, Li Q (2015) A Survey of Fog Computing: Concepts, Applications and Issues. In: Workshop on Mobile Big Data. ACM, pp 37–42