Manisha Luthra, Sebastian Hennig, Pratyush Agnihotri, Lin Wang, Boris Koldehofe. *Highly Flexible Server Agnostic Complex Event Processing Operators.* To appear in the Proceedings of 20th ACM/IFIP/USENIX International Middleware Conference: Posters and Demo, December 2019, ISBN 978-1-4503-7042-4/19/12.

Poster Abstract: Highly Flexible Server Agnostic Complex Event Processing Operators

Manisha Luthra, Sebastian Hennig
TU Darmstadt
Darmstadt, Germany
manisha.luthra@kom.tu-darmstadt.de

Pratyush Agnihotri
axxessio GmbH, Darmstadt
Germany
agnihotri@axxessio.com

Lin Wang
Vrije Universiteit Amsterdam
Amsterdam, Netherlands
lin.wang@vu.nl

Boris Koldehofe
TU Darmstadt
Darmstadt, Germany
boris.koldehofe@kom.tu-darmstadt.de

ABSTRACT

Complex Event Processing (CEP) is a powerful paradigm that can derive correlations from different data sources for a wide variety of applications. CEP provides semantic units called operators e.g., filter and join, that collectively represent a complex event. In current CEP systems, operators are highly dependent on the programming language and the underlying server. This restricts the capability of provisioning user-defined operators at runtime as well as the flexibility of developing server agnostic custom operators.

In this paper, we provide a serverless CEP architecture, which offers developers the flexibility to design operators in any language and integrate them at runtime. We embed operators in the function as a service model of serverless architecture. This is very beneficial for applications such as financial fraud detection where complex machine learning operators must be integrated at runtime to avoid service disruption. We show using our preliminary evaluation that only with minimal overhead in latency, we can offer highly flexible server agnostic CEP operators.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**;

KEYWORDS

Complex Event Processing; Internet of Things; Serverless

ACM Reference format:
Manisha Luthra, Sebastian Hennig, Pratyush Agnihotri, Lin Wang, and Boris Koldehofe. 2019. Poster Abstract: Highly Flexible Server Agnostic Complex Event Processing Operators. In *Proceedings of 20th International Middleware Conference Demos and Posters, Davis, CA, USA, December 9–13, 2019 (Middleware Demos and Posters '19),* 2 pages.
https://doi.org/10.1145/3366627.3368110

1 INTRODUCTION

Credit card fraud is an ongoing issue in the financial industry. Recently, an analysis conducted by the Federal Trade Commission shows that Americans reported loss of over \$1.48 billion to credit fraud in 2018 alone [11]. With credit card fraud inflation by 38% from 2017, financial providers are heavily investing into new technologies to prevent further damages. Visa alone reported to prevent a total of \$25 billion fraud in the year 2018, while spending \$500 million in the last 5 years for the development of infrastructure and AI driven algorithms. [2]

While the detection of fraud using AI is an ongoing issue in research, the validation and training of these algorithms using data from production systems is a field with open problems. Especially in Complex Event Processing (CEP) systems [3], the deployment of new operators becomes a cumbersome task because of the following reasons. (i) The integration of an operator into an existing CEP system is not standardized and requires in-depth knowledge of the streaming system. (ii) The deployment of new operators requires a restart of the streaming system cluster which could lead to service disruptions. However, it is extremely important to perform an online update of a CEP system for applications like fraud detection.

To overcome these limitations towards complex event processing systems, we developed a unified API for CEP systems using the serverless paradigm. Serverless computing or the function as a service (*FaaS*) model offered by the cloud, provides an abstraction for users to define their custom logic without direct knowledge of the underlying compute resource. Hence, we propose CEPFaaS that allows the user to develop custom operators towards any CEP system and be able to deploy them at runtime without service disruptions. In the next sections, we will give an overview of our design, preliminary evaluation, existing work and outlook for future work.

2 THE CEPFAAS UNIFIED API

We aim for two main design goals for our unified API, namely, being independent of (i) execution environment and (ii) programming language. We establish two different entities in our design, which provide information on how a CEP system can utilize serverless operators. The first entity is the *CEPFaaS unified API*, which provides a flexible abstraction for user-defined function development over any CEP execution environment. In our design, we show with the example of two established CEP systems: Apache Flink [1] and TCEP [9], that our unified interface is highly extendable, e.g., CEP-x, can be included by only minor updates in the system.

The *CEP-FaaS unified API* acts as the single-point-of-contact for users to deploy a custom operator ω_{udf}. This is an important feature of a serverless system to contain a serverless API that acts as an interface between user and underlying execution environment. This provides the aforementioned advantages of deployment and

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Middleware Demos and Posters '19, December 9–13, 2019, Davis, CA, USA
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-7042-4/19/12…\$15.00
https://doi.org/10.1145/3366627.3368110

The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

Manisha Luthra, Sebastian Hennig, Pratyush Agnihotri, Lin Wang, Boris Koldehofe. *Highly Flexible Server Agnostic Complex Event Processing Operators.* To appear in the Proceedings of 20th ACM/IFIP/USENIX International Middleware Conference: Posters and Demo, December 2019, ISBN 978-1-4503-7042-4/19/12.

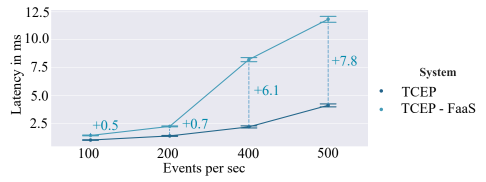

Figure 1: Latency overhead of CEPFaaS

configuration simplicity for the user while still preserving correct execution of the application on the underlying resource. Besides providing simple communication between user and the CEP system, the platform also acts as a central point of knowledge in our deployment implementation. This is because it keeps information of all custom operators submitted by the users and all operator execution requests by the CEP systems.

The second entity is the *Custom Interface*, which is a direct implementation of a communication interface in a CEP system, e.g., for developing a user defined operator ω_{udf}. It is responsible for managing the communication between the CEP system and a custom operator in execution. As not all CEP systems are written in the same programming language and the system design for routing the event stream through an operator is also not standardized, we developed an interface that allows communication between an operator and any CEP system. Because of the largely different implementations of CEP systems, our interface can be customized for every engine that wants to use serverless operators. This interface is responsible for providing a standardized communication protocol for sending and receiving events to and from custom operators. We decided to use an in-memory database, Redis, for providing the exchange of events with a low latency overhead and maximum throughput.

In our preliminary evaluation (cf. Figure 1), we show that our approach induces a minimal overhead ($< 1.5ms$) for smaller event rates in comparison to a conventional CEP system TCEP [9].

3 EXISTING WORK

We mainly review the existing works in two directions (i) serverless computing and (ii) complex event processing systems providing FaaS model.

Serverless Computing. Serverless computing architectures like AWS Lambda [12] and Google Cloud Functions [6] allow users to issue function execution without worrying about the execution environment and the scale of the application. Nastic et al. [10] utilized serverless real-time data analytics to develop user-defined functions, which are abstracted away from the business logic of the underlying streaming system. An open-source alternative to AWS Lambda is Kubeless [7], which provides multiple different runtime environments supporting different languages, but also the ability to provide a custom runtime using a custom image. The aforementioned works show the significance of serverless computing in today's applications, however, either they are limited in providing real-time data processing or the ability to provide language independence in the execution of these systems.

Complex Event Processing. Current CEP systems provide multiple ways of enabling custom logic as operators in streaming applications. For instance, with Apache Heron/Storm [8] the developer can build custom operators in the form of JVMs, which are managed by the framework and enable a greater flexibility in expressing complex queries. Apache Beam [5] provides a means to deal with multiple CEP systems by providing a unified interface. Frameworks like Apache Flink [4] gives the user an ability to define logic using user-defined functions which are easier to express and do not require major code knowledge. However, while all frameworks provide an ability to define custom logic in different ways, they are restricted in the following aspects: (i) submitting functions at runtime, which makes the system more dynamic in terms of operator updates or initial deployment and (ii) abstractions to allow the user to develop the function only one time in a specific language and being able to use it in multiple different CEP systems.

4 CONCLUSION

We provide a serverless unified API for CEP systems such that user-defined functions can be implemented independent of the underlying execution environment as well the programming language. We showed in a preliminary evaluation that in comparison to a conventional CEP system our serverless architecture induces only a minimal overhead while providing the flexibility of being server and language independent.

ACKNOWLEDGEMENT

This work has been co-funded by the German Research Foundation (DFG) as part of the project C2 within the Collaborative Research Center (CRC) 1053 – MAKI.

REFERENCES

[1] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. 2015. Apache Flink: Stream and Batch Processing in a Single Engine. *IEEE Data Engineering Bulletin* 38 (2015).
[2] Sara Castellanos. 2019. The Wall Street Journal: Visa to Test Advanced AI to Prevent Fraud. https://www.wsj.com/articles/visa-to-test-advanced-ai-to-prevent-fraud-11565205158. (2019). [Accessed on 5.9.2019].
[3] Gianpaolo Cugola and Alessandro Margara. 2012. Processing Flows of Information: From Data Stream to Complex Event Processing. *Comput. Surveys* 44, 3 (2012), 1–62.
[4] Apache Software Foundation. 2019. Apache Flink. https://flink.apache.org/. (2019). [Accessed on 5.9.2019].
[5] The Apache Software Foundation. 2019. Beam. https://beam.apache.org/. (2019). [Accessed on 5.9.2019].
[6] Google. 2019. Google Cloud Functions. https://cloud.google.com/functions/. (2019). [Accessed on 5.9.2019].
[7] Kubeless. 2019. Kubeless. https://kubeless.io/. (2019). [Accessed on 5.9.2019].
[8] Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailesh Mittal, Jignesh M Patel, Karthik Ramasamy, and Siddarth Taneja. 2015. Twitter heron: Stream processing at scale. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 239–250.
[9] Manisha Luthra, Boris Koldehofe, Pascal Weisenburger, Guido Salvaneschi, and Raheel Arif. 2018. TCEP: Adapting to Dynamic User Environments by Enabling Transitions Between Operator Placement Mechanisms. In *Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems (DEBS '18)*. ACM, 136–147. https://doi.org/10.1145/3210284.3210292
[10] Stefan Nastic, Thomas Rausch, Ognjen Scekic, Schahram Dustdar, Marjan Gusev, Bojana Koteska, Magdalena Kostoska, Boro Jakimovski, Sasko Ristov, and Radu Prodan. 2017. A serverless real-time data analytics platform for edge computing. *IEEE Internet Computing* 21, 4 (2017), 64–71.
[11] FTC Paul Witt. 2019. Federal Trade Commission: Top frauds of 2018. https://www.ftc.gov/news-events/blogs/business-blog/2019/02/top-frauds-2018?mod=article_inline. (2019). [Accessed on 5.9.2019].
[12] Amazon Web Services. 2019. AWS Lambda. https://aws.amazon.com/lambda/. (2019). [Accessed on 5.9.2019].

The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.