Future Generation Computer Systems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Editorial

New Landscapes of the Data Stream Processing in the era of Fog Computing

Valeria Cardellini [a,*], Gabriele Mencagli [b,*], Domenico Talia [c,*], Massimo Torquati [b,**]

[a] Department of Civil Engineering and Computer Science Engineering, University of Rome Tor Vergata, Italy
[b] Department of Computer Science, University of Pisa, Italy
[c] DIMES, University of Calabria, Rende (CS), Italy

ARTICLE INFO

Article history:
Available online xxxx

ABSTRACT

The "New Landscapes of the Data Stream Processing in the era of Fog Computing" special issue aims to present new research works on topics related to recent advances in Data Streaming Processing (DSP) computing paradigm in the emerging environments of Fog Computing and Internet of Things (IoT). The papers included in this special issue are relevant examples of recent research achievements in the definition of new DSP applications in the Fog Computing context, of run-time systems mechanisms and techniques targeting DPS frameworks, and also of new high-level interfaces for data streaming in highly dynamic IT environments.

1. Data stream processing and Fog computing

Data Stream Processing (DSP) is playing an important role in the area of Big Data analysis. It includes models, techniques, and systems to process data as they arrive to identify patterns, trends, and conditions within a small time period from when data are received. In particular, DSP allows people to feed data into analysis engines as soon as they are generated and find analytics results. With the large diffusion of IoT systems [1], the environments of our everyday life have started to be equipped with a multitude of interconnected devices with increasingly powerful processing and memory capabilities. Such devices are capable of producing a large number of transient flows of information concerning how people moves and behaves in the environment, reporting activities and behaviors in various conditions, and delivering updates related to their evolution and changes. A growing number of applications need efficient processing techniques to extract insights and complex knowledge from such a massive data deluge. This extends to a broad spectrum of applications with a high socio-economic impact in Social Media, Networking, Surveillance, Robotics, Finance, Cybersecurity, to mention some contexts.

DSP [2] is a computing paradigm enabling *data analytics* tasks on inputs available in the form of data streams. This paradigm is often presented as the counterpart of *batch processing*, which instead targets analytics computed over finite, permanent datasets. Starting from the '90s when the first Data Stream Management Systems (DSMSs) [3] appeared on the scene, the streaming paradi -gm has gained the attention of the scientific community, and has evolved from systems executing traditional queries into more complex programming models, supporting user-defined functions and windowing mechanisms to segment the input stream, and providing high degree of configurability to the users with various timing semantics and support to out-of-order data streams [4].

In the last years, this evolution has converged into *Stream Processing Systems* (SPSs), that are complete frameworks to develop and execute stream processing applications. These systems, such as Apache Flink [5], Apache Storm [6], Spark Streaming [7], and Apache Samza [8] to cite some of the most popular open-source ones, are characterized by several hallmark features. First, they allow the programmers to express DSP applications as a set of data transformations (*operators*) composed into data-flow graphs statically interconnected (*topologies*). Users can submit their applications to be executed on the SPS, which usually manages a complex distributed system of homogeneous machines (e.g., a cluster). These systems are in charge of deploying operators onto the available computing resources, as well as to keep the processing as fast as possible, properly balanced according to the workload, and managing fault-tolerance issues of the computation state and the sequence of messages transmitted over the data streams.

Recently, the paradigms of *Edge* and *Fog Computing* [9] have been introduced to deal with the intrinsic problems of geographically distributed systems. With the first advent of IoT environments, the computing paradigm was traditionally a *centralized* one: IoT devices were responsible for gathering and forwarding

* Corresponding authors.
** Corresponding editor.
E-mail addresses: mencagli@di.unipi.it (G. Mencagli), torquati@di.unipi.it (M. Torquati).

https://doi.org/10.1016/j.future.2019.03.027
0167-739X/

Fig. 1. Topics covered by this special issue and classification of the research papers.

huge data flows to Cloud systems, which in turn were responsible for storage and advanced analysis by extracting the outcome of sophisticated data analytics. This solution is far from being scalable, as the network bandwidth does not scale as with the computing power, and data migration from IoT devices to Cloud systems can easily become a bottleneck. Edge and Fog Computing refer to the technologies aimed at pushing intelligence and processing capabilities down to the network edge, closer to where data streams are produced. In other words, by moving the computation to the closest suitable devices along the path from the data sources to the Cloud servers. In this sense, the distinction between Edge and Fog Computing is quite smooth but important: Edge Computing is more oriented towards pushing the intelligence in very thin devices like microcontrollers, directly connected to sensors, e.g., like in programmable automation controllers. Fog Computing, which is the primary target of this special issue, also pushes intelligence down in the architecture, typically moving computations to the Local Area Network (LAN) level, such as in micro-data centers, smart routers and gateways equipped with proper computational and architectural capabilities. So, Fog Computing is based on a hierarchical view of the distributed system, comprising several layers of data processing "along the continuum from Cloud to Things" [10]. In both cases, the two approaches permit to save network bandwidth, reduce communication latency, and allow streaming services to be performed in a delay-sensitive manner.

Despite the technology evolution, SPSs are still far from offering the flexibility required for the seamless integration with Fog Computing scenarios. Indeed, they are tailored for the execution on traditional locally distributed clusters, and this has an impact on both the programming model offered to the users and on the design of their runtime systems. At the programming model level, there is still a lack of abstractions to deal with the asymmetry of the computing and networking environment. This includes abstractions to drive the deployment and placement decisions of individual operators by providing a set of admissible locations and optionally indicating prohibited configurations (e.g., for data security reasons). At run-time system level, the integration with Fog Computing fosters a new design targeting novel, enhanced features. Examples are run-time modifications of the running applications such as topology changes, operator fission/fusion and state migration, which have to be performed among highly heterogeneous nodes. The dynamic selection of different data transfer modes assumes a key role in the scenario of networks with unreliable performances, such as micro-batched transmissions. Finally, but not less important, fault-tolerance models and supports need to be revisited with respect to the existing ones for providing novel failure compensation techniques designed to scale on large environments without centralization points.

In the rest of this editorial, we briefly describe the research contributions part of this special issue and their themes. The papers cover different research topics, all of them related to the integration of Data Stream Processing with Fog Computing.

2. Themes of the special issue

This special issue titled "New Landscapes of the Data Stream Processing in the era of Fog Computing" contains selected research papers focused on the integration of Data Stream Processing techniques and programming models in Fog Computing environments. We received 26 submissions. 9 papers out of 26 manuscripts were extended versions of papers presented during the first edition of the Auto-DaSP workshop [11] (Autonomic Solutions for Parallel and Distributed Data Stream Processing), held in conjunction with Euro-Par 2017 (23rd International European Conference on Parallel and Distributed Computing) in Santiago de Compostela, Spain. A technical committee has carefully reviewed all the submissions, with up to three rounds of extensive reviews. At the end of the process, 10 submissions were selected to be included in this special issue.

The contribution of the 10 accepted papers is organized in three main sub-topics: (i) *new DSP applications in Fog Computing scenarios*, (ii) *Fog-oriented run-time systems for DSP frameworks*, and (iii) *high-level interfaces for stream processing in the Fog*. Fig. 1 shows a schematic view of the three sub-topics and classifies the contribution of each paper in the special issue based on the main sub-topic covered by that research. In the rest of this section, the three sub-topics are described in detail, providing a summary of the main scientific contributions of the accepted papers together with references to them.

2.1. New DSP applications in the Fog

In the paper titled "*Low-Power Portable Devices for Metagenomics Analysis: Fog Computing Makes Bioinformatics Ready for the Internet of Things*" [12], the authors study a practical scenario of the use of Fog Computing technologies in the context of Machine Learning applications. In particular, the focus is on portable genome sequencing machines as the Oxford Nanopore MinION devices [13], which can be used to monitor the bacterial communities in different environments by providing useful information about their interactions with human activities. The applicability of this technology is vast and with high socio-economic impacts. This use case is a significant application of the streaming paradigm in a Fog scenario: Oxford Nanopore devices are configured to start the sequencing process immediately in a pure streaming fashion. The proposed approach allows immediate analysis of the data enabling sequence mapping as quickly as

Please cite this article as: V. Cardellini, G. Mencagli, D. Talia et al., New Landscapes of the Data Stream Processing in the era of Fog Computing, Future Generation Computer Systems (2019), https://doi.org/10.1016/j.future.2019.03.027.

the results become available. Furthermore, such devices are used to process data in the field, and only already processed results are stored in Cloud-based systems. Therefore, the application design perfectly merges advanced stream processing techniques with highly-distributed and pervasive computing environments.

The special issue also covers the design of advanced applications working in Fog Computing scenarios and needing online processing of massive data sequences generated with high frequency. The paper titled "*A Resilient and Distributed Near Real-time Traffic Forecasting Application for Fog computing Environments*" [14] proposes an architecture for a city-wide traffic modeling and prediction service based on the Fog Computing paradi -gm. The authors analyze and discuss combinations of data distribution algorithms and propose a solution aimed at being resilient to back-haul connectivity issues that affect geographically distributed computing environments. The experimental part of the paper represents an interesting description of a real-world road-assistance service fleet located in the city of Barcelona. The results confirm that the proposed solution exhibits a better behavior than existing counterparts, especially when connectivity issues occur and force data to be delivered out-of-order to the Cloud.

As a consequence of the emergence of the IoT paradigm, vast amounts of scientific data from increasingly powerful instruments and devices are made available to the scientific community. Hence, emerging extreme-scale scientific workflows are becoming widespread and so is the need to efficiently automate their deployment on a variety of platforms such as high-performance computers, dedicated clusters, and cloud environments. This is the goal of the approach described in the paper titled "*Detecting Performance Anomalies in Scientific Workflows Using Hierarchical Temporal Memory*" [15]. In the paper, the authors study performance anomalies that can considerably affect the execution of IoT applications, caused by failures and resource contention for example. They propose the use of *Hierarchical Temporal Memory* (HTM) [16] to detect such anomalies in real-time by continuously monitoring the resource consumption of executing workflow tasks and by successfully adapting to changes in the underlying statistics of the data. They apply the proposed approach to two real scientific workflows deployed in the Microsoft Azure's cloud infrastructure and compare its effectiveness and benefits against other online anomaly detection algorithms presented in the past.

2.2. Fog-oriented DSP run-time systems

The paper titled "*Optimizing Distributed Data Stream Processing by Tracing*" [17] tackles the issues related to the run-time system design of DSP frameworks in Fog Computing scenarios. The authors propose a tracing framework for heterogeneous interconnected distributed data stream processing systems. *Traces* are per-record information that the run-time collects to capture causality relations between past, present, and future records at specific points. From the traces, it is possible to build a *lineage* describing the origins and the processing history of an output record. Tracing is especially difficult in distributed environments. In the paper, the authors propose and compare two alternatives. In the first solution, fine-grained events for a sample of records are directly reported into an external distributed message queue and key-value store. In the second solution, the system piggybacks sample records by the execution trace. The authors compare the two approaches by using an Apache Spark prototype to experimentally show that both solutions are capable of tracing with low impact, both concerning coding and execution time.

The authors of the paper titled "*T3-Scheduler: A Topology and Traffic Aware Two-Level Scheduler for Stream Processing Systems in a Heterogeneous Cluster*" [18] propose an efficient task allocation policy targeting highly heterogeneous computing environments. They start from the analysis of existing heuristics schedulers, where the standard practice is to find tasks that communicate intensively with each other and to place them into the same computing node. The authors point out the flaws of this idea, since such schedulers usually inspect each communicating pair of tasks or groups of tasks in isolation, which can result in sub-optimal task assignment decisions. To address the above issues, they propose T3-Scheduler, a new scheduler able to find highly communicating tasks and to assign them to the same compute node in a heterogeneous system. This prototype opens a new design approach of task schedulers for Fog Computing scenarios, where next-generation schedulers must be *topology aware*, as they consider the shape and connectivity of the topology graph when finding highly communicating tasks, and *traffic aware*, as they continuously monitor the streaming application execution to find the data transfer rates between tasks.

Adaptiveness and autonomicity are well-known features in Grid and Cloud computing environments, and their interest is renewed in Fog Computing. In fact, DSP applications urgently need to adapt the settings of their run-time system (e.g., concerning operator placement and use of computational resources) to variable working conditions and workloads, and this is emphasized in the Fog. This problem needs to be studied from different perspectives: not only reliable strategies and efficient run-time mechanisms are required, but a further effort needs to be spent in the design of new programming models incorporating adaptiveness and its features in an easy and user-friendly manner. This is the primary aim of the paper titled "*Simplifying Self-Adaptive and Power-Aware Computing with Nornir*" [19]. The authors propose `Nornir`, a C++ customizable framework for integrating existing stream processing applications with autonomic/adaptive features. The model can be used to enforce specific performance and power consumption requirements by using some self-adaptive strategies already provided by the framework. Besides, `Nornir` can be customized by adding new self-adaptive algorithms. To show the flexibility of the tool, it has been used to implement some state-of-the-art self-adaptive algorithms and to test them over the instrumented PARSEC benchmarks. The novelty of this solution and its high degree of configurability is promising and fosters further research efforts in this direction.

The design of runtime systems targeting stream processing applications is traditionally a complex problem, driven by performance and energy requirements that are becoming ever more stringent and difficult to achieve in highly heterogeneous environments. The paper titled "*Viper: A Module for Communication-Layer Determinism and Scaling in Low-Latency Stream Processing*" [20] moves in this direction. In this paper, the authors study the problem of *determinism* in stream processing applications, which ensures consistent results independently of the way the computation is parallelized. To prevent costly coordination among parallel replicas to enforce determinism, the authors propose `Viper`, a communication module that can be integrated with the communication layer of SPSs. The module boosts the coordination of parallel threads during the data analysis task and represents an efficient and transparent solution to enforce determinism in existing systems.

Elasticity mechanisms have been studied from a large-scale viewpoint in the paper titled "*Decentralized Self-Adaptation for Elastic Data Stream Processing*" [21]. The authors mainly focus their attention on how to support operator elasticity in distributed data stream processing systems, featuring nodes with different networking capabilities and computing power. The proposed solution is a two-level hierarchical control approach, where a centralized per-application component coordinates the run-time adaptation of sub-ordinated distributed components, which,

Please cite this article as: V. Cardellini, G. Mencagli, D. Talia et al., New Landscapes of the Data Stream Processing in the era of Fog Computing, Future Generation Computer Systems (2019), https://doi.org/10.1016/j.future.2019.03.027.

in turn, locally control the adaptation of single operators within the running applications. The authors carefully analyze and compare different strategies to drive elastic decisions in that environment, from classic threshold-based solutions to novel algorithms based on Reinforcement Learning. Promising results have been obtained on real-world applications from the Grand Challenge contest of the DEBS conference, demonstrating the practical usage of the approach in concrete scenarios.

2.3. High-level programming interfaces

The paper titled "*Paving the Way Towards High-Level Parallel Pattern Interfaces for Data Stream Processing*" [22] focuses in a very detailed way on the programmability issues affecting DSP applications. The need for high-level interfaces, close to the final users and easy-to-use, is of crucial importance to increase code productivity and to reduce the time-to-solution of streaming applications in IoT environments. In particular, the paper presents GrPPI, a C++ generic high-level library that acts as a layer between developers and existing parallel programming frameworks, such as C++ threads, OpenMP and Intel TBB. In the paper, the authors complement their interface with patterns targeting streaming scenarios, like Split–Join and Window, and the advanced parallel patterns Stream-Pool, Windowed-Farm and Stream-Iterator. All the parallel patterns expose a general interface usable with different backends. Interestingly, the switching between different backends requires minimum modifications to the source code, allowing for the easy testing of different parallel frameworks to appreciate their scalability and overhead differences in various execution settings.

Along the same research direction, the paper titled "*PiCo: High-Performance Data Analytics Pipelines in Modern C++*" [23] introduces a new framework for data stream analytics entirely written in C++. The programming model is based on a fluent interface and, similarly to Apache Flink, unifies batch and stream data access models. `PiCo` offers a model based on pipelines and operators that are polymorphic with respect to data types, in the sense that it is possible to reuse the same algorithms and pipelines on different data models such as lists, streams, sets and so forth. Preliminary results compared to Spark and Flink demonstrate that better performance can be achieved in terms of execution time with a significantly lower memory utilization, making this new framework particularly suitable for devices with limited resources like the ones available in Fog Computing infrastructures.

3. Conclusions

Data streams are more and more pervasive and DSP is playing an important role in many application domains. Efficient techniques and algorithms are vital for managing and analyzing large data streams. Fog Computing solutions must be exploited for reducing the communication latency and increasing performance and dependability of DSP solutions. The papers presented in this special issue provide significant examples of recent advances of the Data Stream Processing paradigm in the context of Fog Computing.

The contributions of the manuscripts included in the "New Landscapes of the Data Stream Processing in the era of Fog Computing" special issue have been organized in three main topics characterizing three important areas of research, namely: *new DSP applications in Fog Computing scenarios*, *Fog-oriented Run-time systems for DSP frameworks*, and *High-level interfaces for stream processing in the Fog*. The guest editors of this special issue hope that readers may benefit from the perspective presented in this editorial, and most of all, from the scientific results contained in the research papers included in the special issue. We also hope that the contributions of the papers do foster new research activities to address key challenges in this research area.

Acknowledgments

The Guest Editors would like to express their special thanks to the Future Generation Computer Systems (FGCS) Editor in Chief Prof. Peter Sloot for the opportunity he gave us to organize this special issue. We are also grateful to all of the authors who submitted their research work and for their substantial efforts in considering reviewers comments and suggestions. We sincerely thank all reviewers for their time, dedication, and cooperation without which this special issue would not have been possible. Finally, a special thanks to Hilda Xu for helping us to deal with all management issues.

References

[1] O. Vermesan, P. Friess (Eds.), Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems, in: River Publishers Series in Communication, River, Aalborg, 2013, URL http://www.internet-of-things-research.eu/pdf/Converging_Technologies_for_Smart_Environments_and_Integrated_Ecosystems_IERC_Book_Open_Access_2013.pdf.

[2] H.C.M. Andrade, B. Gedik, D.S. Turaga, Fundamentals of Stream Processing: Application Design, Systems, and Analytics, first ed., Cambridge University Press, New York, NY, USA, 2014.

[3] S. Geisler, Data stream management systems, in: P.G. Kolaitis, M. Lenzerini, N. Schweikardt (Eds.), Data Exchange, Integration, and Streams, in: Dagstuhl Follow-Ups, vol. 5, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2013, pp. 275–304, http://dx.doi.org/10.4230/DFU.Vol5.10452.275, URL http://drops.dagstuhl.de/opus/volltexte/2013/4297.

[4] J. Li, K. Tufte, V. Shkapenyuk, V. Papadimos, T. Johnson, D. Maier, Out-of-order processing: A new architecture for high-performance stream systems, Proc. VLDB Endowment 1 (1) (2008) 274–288, http://dx.doi.org/10.14778/1453856.1453890.

[5] E. Friedman, K. Tzoumas, Introduction to Apache Flink: Stream Processing for Real Time and Beyond, first ed., O'Reilly Media, Inc., 2016.

[6] J. Leibiusky, G. Eisbruch, D. Simonassi, Getting Started with Storm, O'Reilly Media, Inc., 2012.

[7] Z. Nabi, Pro Spark Streaming: The Zen of Real-Time Analytics Using Apache Spark, first ed., Apress, Berkely, CA, USA, 2016.

[8] S.A. Noghabi, K. Paramasivam, Y. Pan, N. Ramesh, J. Bringhurst, I. Gupta, R.H. Campbell, Samza: Stateful scalable stream processing at linkedin, Proc. VLDB Endowment 10 (12) (2017) 1634–1645, http://dx.doi.org/10.14778/3137765.3137770.

[9] W. Shi, S. Dustdar, The promise of edge computing, IEEE Comput. 49 (5) (2016) 78–81, http://dx.doi.org/10.1109/MC.2016.145.

[10] OpenFog Consortium, Openfog reference architecture for fog computing, 2017, URL https://www.openfogconsortium.org/ra/.

[11] D.B. Heras, L. Bougé, G. Mencagli, E. Jeannot, R. Sakellariou, R.M. Badia, J.G. Barbosa, L. Ricci, S.L. Scott, S. Lankes, J. Weidendorfer (Eds.), Euro-Par 2017: Parallel Processing Workshops - Euro-Par 2017 International Workshops, Santiago de Compostela, Spain, August 28-29, 2017, Revised Selected Papers, in: Lecture Notes in Computer Science, vol. 10659, Springer, 2018, http://dx.doi.org/10.1007/978-3-319-75178-8.

[12] I. Merelli, L. Morganti, E. Corni, C. Pellegrino, D. Cesini, L. Roverelli, G. Zereik, D. D'Agostino, Low-power portable devices for metagenomics analysis: Fog computing makes bioinformatics ready for the Internet of Things, Future Gener. Comput. Syst. 88 (2018) 467–478, http://dx.doi.org/10.1016/j.future.2018.05.010, URL http://www.sciencedirect.com/science/article/pii/S0167739X17324123.

[13] M. Jain, H.E. Olsen, B. Paten, M. Akeson, The oxford nanopore minION: Delivery of nanopore sequencing to the genomics community, Genome Biol. 17 (1) (2016) 239, http://dx.doi.org/10.1186/s13059-016-1103-0.

[14] J.L. Pérez, A. Gutierrez-Torre, J.L. Berral, D. Carrera, A resilient and distributed near real-time traffic forecasting application for Fog computing environments, Future Gener. Comput. Syst. 87 (2018) 198–212, http://dx.doi.org/10.1016/j.future.2018.05.013, URL http://www.sciencedirect.com/science/article/pii/S0167739X1732678X.

[15] M.A. Rodriguez, R. Kotagiri, R. Buyya, Detecting performance anomalies in scientific workflows using hierarchical temporal memory, Future Gener. Comput. Syst. 88 (2018) 624–635, http://dx.doi.org/10.1016/j.future.2018.05.014, URL http://www.sciencedirect.com/science/article/pii/S0167739X17327292.

[16] M. Putic, A.J. Varshneya, M.R. Stan, Hierarchical temporal memory on the automata processor, IEEE Micro 37 (1) (2017) 52–59, http://dx.doi.org/10.1109/MM.2017.6.

[17] Z. Zvara, P.G. Szabó, B. Balázs, A. Benczúr, Optimizing distributed data stream processing by tracing, Future Gener. Comput. Syst. 90 (2019) 578–591, http://dx.doi.org/10.1016/j.future.2018.06.047, URL http://www.sciencedirect.com/science/article/pii/S0167739X17325141.

[18] L. Eskandari, J. Mair, Z. Huang, D. Eyers, T3-scheduler: A topology and traffic aware two-level scheduler for stream processing systems in a heterogeneous cluster, Future Gener. Comput. Syst. 89 (2018) 617–632, http://dx.doi.org/10.1016/j.future.2018.07.011, URL http://www.sciencedirect.com/science/article/pii/S0167739X17326432.

[19] D. De Sensi, T. De Matteis, M. Danelutto, Simplifying self-adaptive and power-aware computing with Nornir, Future Gener. Comput. Syst. 87 (2018) 136–151, http://dx.doi.org/10.1016/j.future.2018.05.012, URL http://www.sciencedirect.com/science/article/pii/S0167739X17326699.

[20] I. Walulya, D. Palyvos-Giannas, Y. Nikolakopoulos, V. Gulisano, M. Papatriantafilou, P. Tsigas, Viper: A module for communication-layer determinism and scaling in low-latency stream processing, Future Gener. Comput. Syst. 88 (2018) 297–308, http://dx.doi.org/10.1016/j.future.2018.05.067, URL http://www.sciencedirect.com/science/article/pii/S0167739X17326791.

[21] V. Cardellini, F. Lo Presti, M. Nardelli, G. Russo Russo, Decentralized self-adaptation for elastic data stream processing, Future Gener. Comput. Syst. 87 (2018) 171–185, http://dx.doi.org/10.1016/j.future.2018.05.025, URL http://www.sciencedirect.com/science/article/pii/S0167739X17326821.

[22] D. del Rio Astorga, M.F. Dolz, J. Fernández, J.D. García, Paving the way towards high-level parallel pattern interfaces for data stream processing, Future Gener. Comput. Syst. 87 (2018) 228–241, http://dx.doi.org/10.1016/j.future.2018.05.011, URL http://www.sciencedirect.com/science/article/pii/S0167739X17324524.

[23] C. Misale, M. Drocco, G. Tremblay, A.R. Martinelli, M. Aldinucci, Pico: High-performance data analytics pipelines in modern C++, Future Gener. Comput. Syst. 87 (2018) 392–403, http://dx.doi.org/10.1016/j.future.2018.05.030, URL http://www.sciencedirect.com/science/article/pii/S0167739X1732681X.

Please cite this article as: V. Cardellini, G. Mencagli, D. Talia et al., New Landscapes of the Data Stream Processing in the era of Fog Computing, Future Generation Computer Systems (2019), https://doi.org/10.1016/j.future.2019.03.027.