Accelerating the Performance of Data Analytics using Network-centric Processing

Bochra Boughzala
supervised by Boris Koldehofe
Bernoulli Institute, University of Groningen
Groningen, The Netherlands
b.boughzala@rug.nl

ABSTRACT

Distributed execution of real-time data analytics such as event stream processing is the key to scalability, performance and reliable detection of situation changes. Although real-time analytics is highly I/O centric, existing methods supporting the efficient execution of data analytics functions mostly rely on traditional compute models that are available in data centers, e.g., CPU or GPU based processing models, but treat the network mainly as a blackbox. However, with recent advance in software-defined networking (SDN) and the standardization of packet processing pipeline, data analytics functions can be offloaded to programmable switches and benefit from hardware acceleration in an easier and more flexible way than a decade ago. In this paper we focus on the potential of in-network processing to enhance the performance of the overall real-time data analytics application. We aim to contribute to an (i) understanding on how in-network processing can accelerate real-time data analytics and (ii) assess what models of in-network computing can accelerate which event processing functions considering the limitations of network models compared to traditional compute models. We motivate the potential and illustrate the research problems in the context of load balancing which is an important concept in the data-parallel execution of event processing systems.

CCS CONCEPTS

• **Networks** → **Programmable networks, In-network processing**; • **Software and its engineering** → *Publish-subscribe event-based architectures, Message oriented middleware.*

KEYWORDS

In-network Computing, Data Plane Programming, P4, Complex Event Processing (CEP)

ACM Reference Format:

Bochra Boughzala, supervised by Boris Koldehofe. 2021. Accelerating the Performance of Data Analytics using Network-centric Processing. In *The 15th ACM International Conference on Distributed and Event-based Systems (DEBS '21)*, June 28-July 2, 2021, Virtual Event, Italy. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3465480.3468162>

Permission to make digital or hard copies of all or part 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 components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DEBS '21, June 28-July 2, 2021, Virtual Event, Italy

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-8555-8/21/06...\$15.00

<https://doi.org/10.1145/3465480.3468162>

1 INTRODUCTION

Real-time data analytics through continuous data stream processing require scaling capabilities and high speed processing as the workload grows higher. For example, Industrial Internet of Things (IIoT) applications are leveraging 5G networks to connect hundreds of thousands of sensors [1]. They use data analytics augmented with machine learning for automation and better efficiency in the manufacturing process [11]. To support real-time data analytics at scale, application architectures have also evolved from the traditional client/server model to event-driven distributed applications using the publish/subscribe paradigm. In fact, the distributed execution, by the decoupling of event producers and data consumers, is a key feature for ensuring system scalability and adapting applications to environmental changes. In the same context, network operators have embraced *cloud computing* and new networking paradigms along with high performance specialized hardware in their data centers to scale with the increased demands. The location of data centers has impact on the latency and the network bandwidth efficiency, hence application deployment may vary from the core to the edge network, a.k.a. *edge computing*. Traditionally, data center resources have been categorized into compute, storage and network where networking resources do not participate in the compute operations but with new paradigms such as information-centric networking (ICN) and recent advance in software-defined networking (SDN) the network is moving from a hardware-centric view to a programmable platform using domain-specific programming languages [4]. As a result, compute and network resources are blending in the cloud-edge spectrum leading to a new research area called *in-network computing* [16]. For example, the Internet Research Task Force created the COIN (Compute In the Network) [12] research group to study the use cases, requirements and research challenges of this particular field. New specialized intensive I/O-capable programmable data planes [24] are suitable for data-heavy operations. However, most approaches on distributed executions of data analytics to date ignore the network altogether and at best use observable *noisy signals* [2] such as latency or packet loss without insight on the causes and the underlying infrastructure.

In this paper, we argue against network-agnostic deployments as we believe that considering the network as a blackbox, when it provides valuable insights and powerful computational units for efficiently processing and forwarding event streams, is a wastage. Even with faster and more reliable 5G networks, challenging real-time data analytics adaptation can only benefit from hardware acceleration and a tighter control over its distributed execution especially when the infrastructure is shared with other applications.

We believe that with software-defined networking and in-network computing, the network should no longer be seen as a dumb pipe system with the only responsibility of carrying packets from a source to a destination but it can be part of the application deployment solution. Among the benefits of in-network processing is the reduced latency and better bandwidth efficiency as shown in [27], in-network data aggregation results in traffic reduction ratio of 89% with a decrease of computation time on the server side. As appealing these initial results are, there is yet a gap in the understanding of what in-network processing models can support which kind of data analytics functions. In the proposed research, we aim to work on how in-network processing can accelerate real-time data analytics. A starting point for addressing these research questions is to illustrate the problem of supporting data-parallel execution in the context of complex-event processing (CEP).

The remainder of the paper is structured as follows. Section 2 presents the related works. Section 3 illustrates the research problems. Section 4 describes the methodology and finally Section 5 is the conclusion.

2 RELATED WORK

In the context of event-based systems, InetCEP [20] is a new communication model and a query language for integrating complex-event processing (CEP) architecture over ICN for enhanced performance while maintaining extensibility. SDN has been also gaining interest for example through the use of programmable TCAMs (Ternary Content-Addressable Memory) to implement in-network content-based filtering. In [3], the authors propose to offload publish/subscribe middleware filtering operations to SDN switches whose *flow-table* entries can be configured via the OpenFlow standard [21] and thereby benefit from line-rate performance and the flexibility of adapting to constantly changing interests of producers and consumers thanks to the centralized view of SDN controller. While in [19], the P4 language [4] was used for moving the *Broker* function of publish/subscribe system from an overlay network to the underlay P4-based switches. Finally, P4CEP [18] is a toolchain for executing in-network complex-event processing on smartNICs. It consists of a language to define complex-event specification rules using high-level semantics, such as *window* operator, and a compiler to generate P4 code.

In domains other than event processing, in-network caching solution NetCache [14] shows that by efficiently processing key-value items in the data plane, leveraging the switch on-chip memory, the throughput of in-memory key-value stores is improved and the latency is reduced. Another example is NetChain [13] which demonstrates that programmable switches can be used to bypass coordination servers. By processing the data entirely within the network, NetChain eliminates coordination latency overhead to a half of an RTT (round-trip time) and achieves higher throughput compared to a server-based solution. Consensus and trust systems depict another type of distributed applications for applying in-network computation as presented in NetPaxos [8] and [7]. Lastly, in [28], the authors describe an attempt to run a neural network model in a switch and highlight the challenges of applying in-network computing to the domain of artificial intelligence.

These works confirm the potential of offloading functions of distributed applications to the network yet there is a lack of understanding and guidelines on how and what network models can support and accelerate which functions. In the context of data analytics, our work aims at investigating models of in-network processing to enable a new generation of *network-aware* event stream processing systems.

3 PROBLEM STATEMENT

Figure 1: Complex event processing over in-network computing.

In complex-event processing (CEP) examples of data sources (producers) can be Internet user-clicks on a web browser or IoT devices (i.e., sensors) generating *basic-events* (e.g., temperature or location change). The incoming events are transported from the event producers to data consumers via a publish/subscribe messaging service for downstream processing where application-specific rules are executed by a rule engine (*Operator*) leading to the transformation of a set of incoming streams to a set of outgoing streams of inferred *complex-events* (Figure 1). The results may be used to trigger actions such as email push notifications or alarms. Data producers publish events based on topics while data consumers receive only events of interest based on their subscriptions. The event subscriptions can be (1) *topic-based* (coarse-grained subscription) where messages are delivered via logical channels defined by the publishers or (2) *content-based* (fine-grained subscription) using constraints expressed in classification rules defined by the subscribers. Message filtering in event-based system is ensured by a message *Broker*. *Publishers*, *Subscribers* as well as CEP *Operators* may join or leave the system dynamically.

Typically, production-grade CEP solutions are implemented fully in software (e.g. Google Cloud Platform (GCP) and Apache Flink) where the network is completely agnostic of the messaging service and it is only used to forward event streams between data sources and data sinks. The problem with software-based CEP implementation is the added latency of event storage and the overhead of *Operator* execution on servers. This approach results in high delays for time-sensitive IoT applications known for low latency and jitter requirements. Besides, in order to scale with dynamic and increasing workloads more workers are deployed resulting in higher cost (more servers and power usage) and complexity issues (orchestration of micro-services). Instead, one can leverage in-network processing resources to benefit from the performance of high-speed

and programmable data planes [5]. Offloading parts of distributed applications onto the network has been done so far in an experimental manner using workarounds to bypass inherent problems with existing network models. One of the major roadblocks for in-network event processing is the stateless nature of traditional packet processing [10] while data analytics are very state-centric. For this reason, some efforts towards statefulness like FlowBlaze [25] offer to complement the Match-Action Table (MAT) abstraction with a Finite State Machine (FSM) to have a better handling of the state in the data plane.

In the proposed thesis, our ultimate research goal is to build a new understanding of in-network programming models that can support event processing. We envision that network-centric processing will play a big role for the efficient execution of distributed real-time data analytics applications.

Research Goals : We define the following more specific research goals : (RG1) Assess the limitations of in-network processing such as the lack of a *unified* programming model that support both stateless and stateful operations. (RG2) Identify real-time data analytics functions that can be good candidates for in-network performance acceleration. (RG3) Propose customized Quality of Service (QoS) strategy to ensure *fairness* in a heterogeneous network setting where time-sensitive event-driven applications share network resources with other services.

Hypothesis : If we apply in-network processing to offload functions of real-time data analytics to programmable data planes then the overall application performance will be enhanced and the latency will be reduced. This hypothesis goes against the *end-to-end principle* [26] which argues that most of the times it is better to keep advanced functions at the end-hosts. This is because the network nodes were too complex and expensive to change. With today's programmable data planes and their new standard packet programming language we have the flexibility to develop application-specific optimizations in the network with no additional cost.

Research Questions : The following research questions will be investigated. (RQ1) How in-network processing can improve the performance of real-time data analytics and what models of in-network computing can accelerate which functions ? (RQ2) What are the challenges and limitations of offloading for example a stateful event stream load-balancing function to programmable forwarding plane, knowing that contrary to conventional compute models who are theoretically Turing-complete, some network models (e.g., P4, OpenFlow) are not ? (RQ3) How to ensure QoS guarantees (such as bounded delay, lossless) when time-sensitive event-based applications are deployed using in-network processing and therefore have to coexist with other background services and share and compete over the same network resources ?

4 METHODOLOGY

To address the research question (RQ1–3) we take a close look into the state of the art networking models. We motivate the potential of in-network processing and illustrate the related research problems in the context of *load-balancing* which has many important implications in the data-parallel execution of event-processing systems. To work towards a solution we aim for the following contributions.

4.1 In-network programming models for real-time data analytics

One of our goals is to identify in-network programming models which can accelerate functionalities of complex event processing. Let's note that such models have also the potential to benefit other use cases, e.g., neural networks, consensus systems.

One of the most popular network programming models is P4 [4] which offers multiple advantages among which (1) the **flexibility** to define new customized protocols and packet headers with the corresponding match-action pipeline, (2) the **portability** to run on various platforms ranging from software switches to hardware targets, e.g., SmartNICs, FPGAs and programmable ASICs and (3) the access to **high-performance** programmable packet processors such as Tofino2 [24] that has a processing capacity of 12.8 Tbps.

Other models are also widely used, for instance DPDK [17] offers multi-core scalability and other frameworks built on top of it offer higher semantics such as packet vector-oriented (VPP) and graph-oriented (BESS) processing models. While DPDK-based programs reside in the user space, XDP (eXpress Data Path) [9] does not require a full kernel bypass with the benefit to *hook* user programs (eBPF) in conjunction with the Linux networking stack. The choice of the execution environment will imply that either a run-to-completion model or a pipeline model will be employed. For example the P4 programming model is based on a pipeline model which does not support loops and has limited support of the state.

CEP applications have their own specific abstractions that are not inherently supported in current programmable data planes. Therefore, we need a model for event processing that not only supports stateful computation but also captures and exposes new semantics such as *sliding window*.

4.2 Stateful and scalable event stream partitioning and load-balancing

Figure 2: CEP event window partitioning and load balancing.

For data-parallel execution of event processing, a load-balancing function can create a bottleneck while partitioning the event streams to the pool of available operators. Therefore, we propose to accelerate the events window splitting logic by leveraging for example high-performance P4 programmable switches (Figure 2). There are some works on stateful and scalable load-balancing at line-rate using P4 such as Hula [15] and SilkRoad [22] but these solutions are designed for connection-oriented communications (TCP flows). In the context of in-network complex-event processing, a stateful event stream load-balancer must ensure the following properties. (1) **Correctness**. The window semantic (time-based, count-based

or marker-based) must be properly executed otherwise the downstream processing may result in false negatives (a complex event should have been detected but it was not) or duplicates of complex event detection. The *per-window consistency* property ensures that the result of parallel processing will be the same as sequential processing. (2) **Adaptability**. Due to the dynamic nature of the system, a CEP load-balancer must react fast to link failures and changes to the set of available operators as they might be added or removed at runtime. and (3) **Even Distribution**. Finally, while keeping the property of *per-window consistency* event windows must be evenly distributed. One could think of Round-Robin scheduling or Weighted Round Robin if the different workers don't have the same processing capacity. These properties can be invalidated when events are *out-of-order*. For the load-balancer to behave correctly we assume that packets arrive in order which might require a pre-processing of the event streams before arriving at an in-network processing unit and therefore can reduce the performance.

4.3 Evaluation

To validate the effectiveness of in-network programming models with event stream processing we can implement different CEP operator algorithms and evaluate how helpful or challenging the available abstractions are. For the load-balancing function, the *sliding window* operator needs to be implemented to detect the start of new windows and send events within a window to the same operator. We will evaluate the load-balancer first at the behavioral level with focus on correctness and even event distribution. For that purpose, Mininet [23] can be used to verify the functionality. We will need to develop an event stream generator where we can use one of the open datasets of DEBS grand challenge [6] that are available (e.g., nyc taxi trips dataset). Next, we want to experiment with representative in-network devices as we plan to work on a hardware implementation to conduct a performance evaluation of the load-balancer as well as other data analytics functions. The throughput and the quality i.e., accuracy, of the results is among the evaluation metrics that will be considered.

5 CONCLUSION

This paper argues that with today's network deep programmability, which was not as flexible a decade ago, we can enable novel methods for in-network real time data analytics. We understand that network-centric processing has not yet been paid a lot of attention in the area of distributed event-based systems. With the proposed research goals we hope we contribute to understanding of network accelerated stream processing. We also hope that our proposed research will raise more interest and awareness in the potentials of in-network processing.

ACKNOWLEDGMENTS

We would like to thank the reviewers for their valuable feedback.

REFERENCES

- [1] 5G and the promise of futureproof factories. 2021. <https://www.ericsson.com/en/blog/2021/3/5g-futureproof-factories>
- [2] Serhat Arslan and Nick McKeown. 2019. Switches Know the Exact Amount of Congestion. In *Proceedings of the 2019 Workshop on Buffer Sizing*. 1–6.
- [3] Sukanya Bhowmik, Muhammad Adnan Tariq, Boris Koldehofe, Frank Dürr, Thomas Kohler, and Kurt Rothermel. 2016. High performance publish/subscribe middleware in software-defined networks. *IEEE/ACM Transactions on Networking* 25, 3 (2016), 1501–1516.
- [4] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. 2014. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 87–95.
- [5] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. 2013. Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN. *ACM SIGCOMM Computer Communication Review* 43, 4 (2013), 99–110.
- [6] Grand Challenges. 2021. <https://debs.org/grand-challenges/>
- [7] Huynh Tu Dang, Marco Canini, Fernando Pedone, and Robert Soulé. 2016. Paxos made switch-y. *ACM SIGCOMM Computer Communication Review* (2016), 18–24.
- [8] Huynh Tu Dang, Daniele Sciascia, Marco Canini, Fernando Pedone, and Robert Soulé. 2015. Netpaxos: Consensus at network speed. In *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*. 1–7.
- [9] eXpress Data Path. 2021. <https://www.iovisor.org/technology/xdp>
- [10] Nadeen Gebara, Alberto Lerner, Mingran Yang, Minlan Yu, Paolo Costa, and Many Ghobadi. 2020. Challenging the Stateless Quo of Programmable Switches. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*. 153–159.
- [11] Artificial intelligence and machine learning in next-generation systems. 2021. <https://www.ericsson.com/en/reports-and-papers/white-papers/machine-intelligence>
- [12] IRTF. 2021. Computation in the Network Research Group. <https://irtf.org/coingr>
- [13] Xin Jin, Xiaozhou Li, Haoyu Zhang, Nate Foster, Jeongkeun Lee, Robert Soulé, Changhoon Kim, and Ion Stoica. 2018. Netchain: Scale-free sub-rtt coordination. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*. 35–49.
- [14] Xin Jin, Xiaozhou Li, Haoyu Zhang, Robert Soulé, Jeongkeun Lee, Nate Foster, Changhoon Kim, and Ion Stoica. 2017. Netcache: Balancing key-value stores with fast in-network caching. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 121–136.
- [15] Naga Katta, Mukesh Hira, Changhoon Kim, Anirudh Sivaraman, and Jennifer Rexford. 2016. Hula: Scalable load balancing using programmable data planes. In *Proceedings of the Symposium on SDN Research*. 1–12.
- [16] Daehyeok Kim, Ankush Jain, Zaoxing Liu, George Amvrosiadis, Damian Hazen, Bradley Settlemyer, and Vyas Sekar. 2020. Unleashing In-network Computing on Scientific Workloads. *arXiv preprint arXiv:2009.02457* (2020).
- [17] Data Plane Development Kit. 2021. <https://www.dpdk.org/>
- [18] Thomas Kohler, Ruben Mayer, Frank Dürr, Marius Maaß, Sukanya Bhowmik, and Kurt Rothermel. 2018. P4CEP: Towards in-network complex event processing. In *Proceedings of the 2018 Morning Workshop on In-Network Computing*. 33–38.
- [19] Ralf Kundel, Christoph Gärtner, Manisha Luthra, Sukanya Bhowmik, and Boris Koldehofe. 2020. Flexible Content-based Publish/Subscribe over Programmable Data Planes. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–5.
- [20] Manisha Luthra, Boris Koldehofe, Jonas Höchst, Patrick Lampe, Ali Haider Rizvi, Ralf Kundel, and Bernd Freisleben. 2019. Inetcep: In-network complex event processing for information-centric networking. In *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. IEEE, 1–13.
- [21] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM computer communication review* 38, 2 (2008), 69–74.
- [22] Rui Miao, Hongyi Zeng, Changhoon Kim, Jeongkeun Lee, and Minlan Yu. 2017. Silkroad: Making stateful layer-4 load balancing fast and cheap using switching ASICs. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 15–28.
- [23] Mininet. 2021. <http://mininet.org/>
- [24] Barefoot Networks. 2021. Tofino2: Second-generation of World's fastest P4-programmable Ethernet switch ASICs. <https://www.barefootnetworks.com/products/brief-tofino-2/>
- [25] Salvatore Pontarelli, Roberto Bifulco, Marco Bonola, Carmelo Cascone, Marco Spaziani, Valerio Bruschi, Davide Sanvito, Giuseppe Siracusano, Antonio Capone, Michio Honda, et al. 2019. Flowblaze: Stateful packet processing in hardware. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*. 531–548.
- [26] Jerome H Saltzer, David P Reed, and David D Clark. 1984. End-to-end arguments in system design. *ACM Transactions on Computer Systems (TOCS)* 2, 4 (1984), 277–288.
- [27] Amedeo Sapio, Ibrahim Abdelaziz, Abdulla Aldilajan, Marco Canini, and Panos Kalnis. 2017. In-network computation is a dumb idea whose time has come. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. 150–156.
- [28] Giuseppe Siracusano and Roberto Bifulco. 2018. In-network neural networks. *arXiv preprint arXiv:1801.05731* (2018).