Demo: Complex Event Processing over Streaming Multi-Cloud Platforms - The FERARI Approach*

Ioannis Flouris⋆ Vasiliki Manikaki⋆ Nikos Giatrakos⋆ Antonios Deligiannakis⋆
Minos Garofalakis⋆ Michael Mock◇ Sebastian Bothe◇ Inna Skarbovsky♯
Fabiana Fournier♯ Marko Štajcer† Tomislav Križan† Jonathan Yom-Tov§
Marijo Volarević‡

⋆Technical University of Crete
{gflouris, manikaki, ngiatrakos, adeli, minos}@softnet.tuc.gr

♯IBM Research - Haifa
{fabiana, inna}@il.ibm.com

◇Fraunhofer IAIS
{michael.mock, sebastian.bothe}@iais.fraunhofer.de

§Technion, Israel Institute of Technology
jonyomtov@cs.technion.ac.il

†Poslovna Inteligencija
{marko.stajcer, tomislav.krizan}@inteligencija.com

‡T-Hrvatski Telekom
Marijo.Volarevic@t.ht.hr

ABSTRACT

We present FERARI, a prototype for processing voluminous event streams over multi-cloud platforms. At its core, FERARI both exploits the potential for in-situ (intra-cloud) processing and orchestrates inter-cloud complex event detection in a communication-efficient way. At the application level, it includes a user-friendly query authoring tool and an analytics dashboard providing granular reports about detected events. In that, FERARI constitutes, to our knowledge, the first complete end-to-end solution of its kind. In this demo, we apply the FERARI approach on a real scenario from the telecommunication domain.

CCS Concepts

•Information systems → Data streams; •Applied computing → Event-driven architectures; •Computer systems organization → Distributed architectures;

Keywords

Complex Event Processing, Cloud Computing, Distributed Streams

1. INTRODUCTION

Many modern Big Data technologies such as Machine-to-Machine (M2M) or Internet-of-Things (IoT) platforms create real-time data streams that are structured in the form of series of interaction event occurrences. Complex Event Processing (CEP) systems encompass the ability to process and query such data so as to detect complex patterns. Complex Event (CE) patterns, involve predefined rules that match incoming event notifications on the basis of their content and on some ordering relationships on them [7].

With the emergence of streaming cloud platforms that allow for CEP at large scale, the design principles of respective systems need to be adapted to exploit intra-cloud parallelization and elastic resource consumption. One step further, in vast scale distributed CEP systems, centralizing voluminous raw events first and then processing them is infeasible, as this would cause a bottleneck at a central site/cloud. Streaming event data arriving at multiple, potentially geographically dispersed, cloud platforms should be efficiently processed in-situ and then wisely combined to provide holistic answers to global application queries. Efficient inter-cloud CEP calls for reduced communication to avoid congested links among sites with distinct cloud deployments.

Motivated by the above, in this demo we present FERARI [6], a prototype that enables real-time CEP for large-volume event data streams over distributed topologies. At its core, FERARI exploits both the potential for intra-cloud and orchestrated inter-cloud CEP in a communication-efficient way that simultaneously allows for timely event detection. At the application level, it provides an event query authoring tool together with an analytics dashboard that presents a holistic picture with respect to the detected complex events to final stakeholders. In that, FERARI constitutes, to our knowledge, the first complete end-to-end solution of its kind. As a proof of concept, we apply FERARI on a mobile fraud detection scenario using real rules and real, anonymized, telecommunication data from T-Hrvatski Telekom network in Croatia.

2. ARCHITECTURE

The architectural modules of FERARI are illustrated in Figure 1 and are discussed below along with our contributions.

CE Query Authoring Tool: As shown at the left top of Figure 1, application queries are posed via a web-based user interface. Our query design approach complies with the CEP concepts discussed in [7, 8]. Hence, the basic building blocks for designing application queries are Event Processing Agents (EPAs). An EPA handles a pattern operator that receives as input certain types of events of in-

*A full version of this paper appears in [6].

DEBS '16 June 20-24, 2016, Irvine, CA, USA

ACM ISBN 978-1-4503-4021-2/16/06.

DOI: http://dx.doi.org/10.1145/2933267.2933289

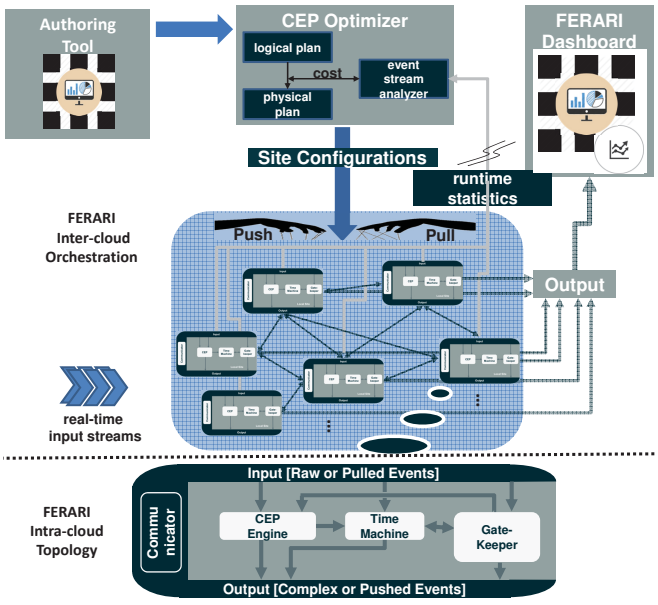

Figure 1: FERARI Architecture at Inter- & Intra-cloud levels

terest and outputs detected CEs. The whole query is an Event Processing Network (EPN) describing the event flows among EPAs for the detection of the final CEs. Manually coding all the application rules and their specifications can be a painful task. Our authoring tool facilitates and speeds up this process.

Inter-cloud CEP Optimization: To evaluate the submitted EPN in our distributed setup, events arriving at distinct sites/clouds need to be combined (after any possible in-situ processing). Each EPA will be placed at some site, which will be responsible for synthesizing information from other sites producing input events to this EPA.

To accomplish this task in a communication-efficient way, the FERARI optimizer (middle top of Figure 1) greatly extends the push/pull paradigm of [4] to allow for heterogeneous site inputs and multiple choices of where to place each EPA. Based on available event statistics, the FERARI optimizer picks the best plan including both the determination of the placement for each EPA, as well as the order by which relevant events will be pulled from/pushed to it. An optimal plan best balances reduced communication - achieved by postponing push/pull activity regarding higher frequency events participating in a CE pattern until the occurrence of lower frequency ones - and detection latency due to the postponed transmissions. Based on the chosen query plan, site configurations are generated and disseminated to each site.

FERARI Intra-cloud CEP: Each site runs an Apache Storm [2] topology, shown at the bottom of Figure 1, comprised of [5]:

- Input Spout: where streaming tuples arrive or pushed events from other sites are fed into the CEP Engine.
- CEP Engine: details of our CEP Engine follow shortly.
- Time Machine Bolt: buffers & chronically orders derived events.
- Gatekeeper Bolt: responsible for advanced calculations and distributed CE resolution procedures.
- Communicator Bolt: responsible for the push/pull based communication between different sites.

The CEP Engine module of our prototype, namely ProtonOnStorm [3], is an open-source platform that extends IBM Proactive Technology Online (Proton) standalone. ProtonOnStorm is itself distributed across a number of Storm bolts allowing for different degrees of parallelization in different modules [3]. The CEP Engine receives events from the Input Spout and having processed them in-situ, emits derived events towards the Time Machine. If a CE is detected that needs to be transmitted to a remote cloud, it will be then routed to the Communicator (Figure 1) to be pushed to the proper remote site, according to the inter-cloud execution plan.

FERARI Dashboard: The web-based dashboard of our prototype is generic enough to produce a wide variety of reports and analytic results that are useful from an application viewpoint. As shown in [1], it comprises of the following widgets: (1) the complex event grid, which depicts detected events and relevant information, (2) the event statistics widgets with cumulative statistics over different time windows, (3) peek/offpeek statistics regarding events in different periods of the day, (4) a widget displaying the most frequent CEs aggregates, (5) an interactive map of site/cloud locations and detected events, and (6) a navigation panel.

3. DEMONSTRATION DETAILS

For the purposes of the demonstration, we will use a real, anonymized, dataset of call data coming from HT's network and real (properly masked) fraud detection rules from HT to a priori build a set of EPAs. Using the pre-built EPAs as a guide, users will also be able via the authoring tool to build their own, fraud detection related, queries. From an application viewpoint, users will be able - having posed their fraud related queries - to experience real-time CE notifications via FERARI's analytics platform specializing the widgets discussed in Section 2 (also see [1]). To fully assess the details of FERARI's deployment and the function of individual components beyond the application's viewpoint, interested users will be able to have access to logging information presented in a user-friendly way, including: (a) the inter-cloud execution alternatives (logical and physical plans) and the procedure FERARI's optimizer followed to produce and choose the best plan, (b) logs of inter-cloud (push/pull) and intra-cloud (i.e., per site) activity.

4. ACKNOWLEDGMENTS

This work was supported by the European Commission under ICT-FP7-FERARI-619491 (Flexible Event pRocessing for big dAta aRchItectures).

5. REFERENCES

[1] FERARI Dashboard Preview. http://recordit.co/4yi6Qpnlo6.
[2] Apache Storm Project Homepage. http://storm.apache.org/.
[3] IBM Proactive Technology Online on STORM. https://github.com/ishkin/Proton.
[4] M. Akdere, U. Çetintemel and N. Tatbul. Plan-based complex event detection across distributed sources. In *PVLDB*, 2008.
[5] S. Bothe, V. Manikaki, A. Deligiannakis and M. Mock. Towards flexible event processing in distributed data streams. In *Proc. of the Workshops of the EDBT/ICDT*, 2015.
[6] I. Flouris, V. Manikaki, N. Giatrakos, A. Deligiannakis, M. Garofalakis, M. Mock, S. Bothe, I. Skarbovsky, F. Fournier, M. Štajcer, T. Križan, J. Yom-Tov, T. Ćurin. FERARI: A Prototype for Complex Event Processing over Streaming Multi-cloud Platforms. In *SIGMOD*, 2016.
[7] O. Etzion and P. Niblett. *Event Processing in Action*. Manning Publications Company, 2010.
[8] C. Moxey, M. Edwards, O. Etzion, M. Ibrahim, S. Iyer, H. Lalanne, M. Monze, M. Peters, Y. Rabinovich, G. Sharon and K. Stewart. *A Conceptual Model for Event Processing Systems*. IBM Redguide publication, 2010.