FERARI: A Prototype for Complex Event Processing over Streaming Multi-cloud Platforms[*]

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[§]
Taji Ćurin[‡]

[⋆]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
Taji.Curin@t.ht.hr

ABSTRACT

In this demo, we present FERARI, a prototype that enables real-time Complex Event Processing (CEP) for large volume event data streams over distributed topologies. Our prototype constitutes, to our knowledge, the first complete, multi-cloud based end-to-end CEP solution incorporating: a) a user-friendly, web-based query authoring tool, (b) a powerful CEP engine implemented on top of a streaming cloud platform, (c) a CEP optimizer that chooses the best query execution plan with respect to low latency and/or reduced inter-cloud communication burden, and (d) a query analytics dashboard encompassing graph and map visualization tools to provide a holistic picture with respect to the detected complex events to final stakeholders. As a proof-of-concept, we apply FERARI to enable mobile fraud detection over real, properly anonymized, telecommunication data from T-Hrvatski Telekom network in Croatia.

1. INTRODUCTION

Machine-to-Machine (M2M) synergies generate event data in high frequency in every modern Big Data system, from network health monitoring and mobile or sensor network deployments to computer clusters and smart energy grids. Besides M2M interactions, Internet-of-Things (IoT) platforms can offer advanced connectivity of devices and services that cover a variety of domains and applications, generating high volumes of event data streams and patterns of interest subjected to further study.

Complex Event Processing (CEP) Engines [3, 5] aim at processing such event data streams efficiently and immediately recognizing interesting situations in real-time. Primitive events are atomic (i.e., non-decomposable) occurrences that stream into a CEP system, while complex events (CEs) are derived by the CEP Engine based on application-defined event patterns (rules) engaging other primitive and/or CE combinations. For instance, consider a number of telecommunication antennas in a simplified mobile fraud detection scenario. A mobile device exiting the realm of an antenna triggers the report of a corresponding primitive event with the duration of the ongoing call maintained by that particular cellular device as an attribute. The CE of a potential fraud case apparition may involve summing up the overall call duration for a certain cell phone number across visited antennas and assessing whether it exceeds a given threshold within a recent time window.

In large scale distributed CEP systems, centralizing all raw events is not possible, as this would create a bottleneck at the central site/cloud. Streaming event data arriving at multiple, potentially geographically dispersed, cloud platforms should be efficiently processed in situ (if possible) and then combined to provide holistic answers to global application queries. Efficient inter-cloud CEP is tightly coupled with the requirement for reduced communication since conventional communication links interconnect sites with distinct cloud deployments. Towards that end, query plans should be generated exploiting the potential for in-situ processing.

One step further, enabling CEP at vast scale involves scaling the amount of local processing performed within each site/cloud. With the emergence of streaming cloud platforms (such as Apache Storm [1]), CEP systems need to be adapted and redesigned to exploit *intra-cloud* parallelization and elastic resource consumption.

In this demo, we present FERARI, a prototype that enables real-time CEP for large volume event data streams over distributed topologies. The key characteristics that distinguish FERARI from conventional competitors that operate over cloud platforms (such as Esper [4]), is that FERARI takes advantage of both inter-cloud CEP by distributing the load over a set of streaming cloud platforms and parallelized, intra-cloud CEP where appropriate. However, FERARI's breakthrough is not limited to the above important charac-

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

SIGMOD'16, June 26-July 01, 2016, San Francisco, CA, USA
© 2016 ACM. ISBN 978-1-4503-3531-7/16/06. . . $15.00
DOI: http://dx.doi.org/10.1145/2882903.2899395

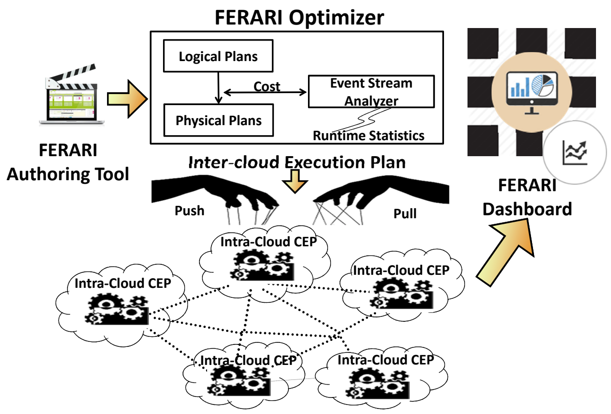


Figure 1: FERARI Architecture - The Big Picture

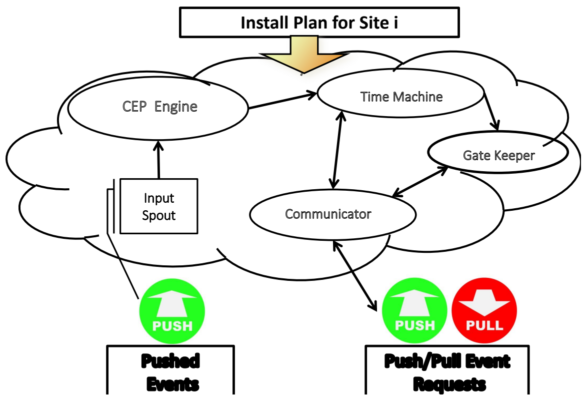


Figure 2: Intra-cloud FERARI Topology

teristic. FERARI constitutes, to our knowledge, the first complete, multi-cloud based, end-to-end CEP solution incorporating:

- a user-friendly, web-based query authoring tool,
- a powerful engine for efficient intra-cloud CEP, implemented on top of a state-of-the-art streaming cloud platform, namely Apache Storm [1],
- a CEP optimizer that orchestrates interactions among CEP Engines in different clouds, and chooses the best execution plan balancing requirements for both low latency and reduced inter-cloud communication,
- an analytics dashboard encompassing time series analysis, graph and map visualization tools to provide a holistic picture with respect to the detected CEs to final stakeholders.

As a proof-of-concept, we apply FERARI to enable mobile fraud detection using real rules and over real, anonymized, telecommunication data from T-Hrvatski Telekom network in Croatia.

2. FERARI ARCHITECTURE

The overall architecture of FERARI is illustrated is Figure 1, while Figure 2 and Figure 4 present intra-cloud processing modules in detail. Below, we present the main components of FERARI, along with our contributions.

CE Query Authoring Tool: In FERARI queries are formed using a web-based graphical user interface as shown in Figure 3. Our event query modeling approach is inspired by the CEP concepts discussed in [8, 9]. The query elements included in FERARI 's authoring tool involve [8, 9]:

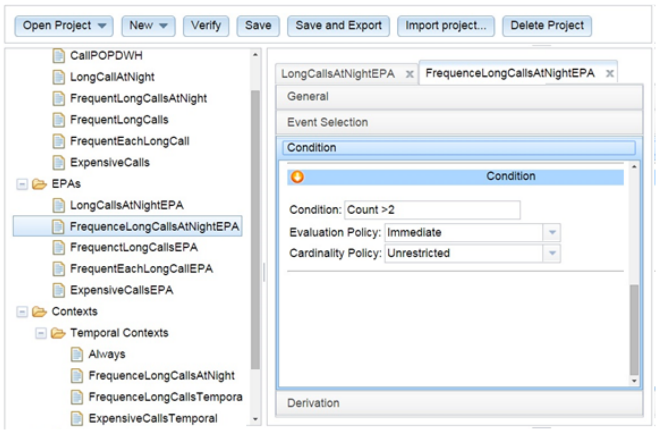
- Event types - the events that are expected to be received as input or to be generated as CEs. An event type definition includes the event name and a list of its attributes.
- Producers - event sources and the way the CEP Engine gets events from them.
- Consumers - event consumers and the way they get derived events from the CEP Engine.
- Event Processing Agents (EPAs) - patterns of incoming events with specific context (see below) that detect situations and generate derived events.
- Temporal contexts - time windows in which EPAs are active.
- Segmentation contexts - semantic groupings of events.
- Composite contexts - synthesis of different contexts.
- Complex Event Definitions - Definition of the above concepts for CEs that are produced by EPAs.

An Event Processing Network (EPN), a conceptual query model describing the event processing flow execution, is then created. An EPN comprises a collection of EPAs, event producers and consumers. The network describes the flow of events originating at event producers and flowing through various event processing agents to eventually reach event consumers. The created query (EPN) is automatically transformed into a JSON file and is forwarded to FERARI's CEP optimizer upon query submission. It can easily be observed that directly coding a proper JSON file tailoring the above concepts to the desired query is a painful task. Therefore, our web-based authoring tool facilitates and speeds up this process. Query formulation using the authoring tool is oblivious to the details of the underlying multi-cloud setup. Users pose queries in a way that abstracts the details of the network of sites running cloud deployments. The EPN produced by the authoring tool may contain annotations, which are hints provided to the query optimizer, such as the optimization metric (i.e., minimize inter-site communication (default metric), maximum detection latency, etc).

FERARI Query Optimizer & Inter-cloud CEP: In our multi-cloud, distributed setup, different sites may observe only a subset of the event types that participate in the posed query. Therefore, the EPN received by the FERARI optimizer must be broken down into pieces involving different portions of the query. These parts of the EPN should be installed in, potentially overlapping, subsets of sites where relevant event types arrive. Furthermore, in order to enable the detection of CE pattern matches that encompass information across various sites, primitive or complex events need to be exchanged among the sites. For each EPA of the EPN that should be installed in a subset of sites, a sole site must be picked as the special one, responsible for synthesizing event information from other sites in the same subset to determine matches of event patterns.

In this context, our optimization approach uses the push/pull paradigm discussed in [6], but significantly extends these concepts by allowing the potential for EPAs to be installed at different clouds and by loosening the restricting assumption of having all sites in the network observe exactly the same event types. The main optimization goal is to produce Pareto optimal plans with respect to communication and latency. An optimal plan appropriately 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 event detection latency due to these postponed transmissions.

As shown in the middle top of Figure 1, the optimizer initially produces a set of logical plans. During logical plan generation, query rewriting procedures are applied on the posed query and subsets of sites where portions of the posed query need to be installed



Condition

Condition: Count >2

Evaluation Policy: Immediate

Cardinality Policy: Unrestricted

Figure 3: FERARI 's Authoring Tool

are determined. Furthermore, topological orderings of event types are created to later represent all possible sequences of push/pull activity during pattern matching checks. Based on the gathered statistics, physical plans are generated which include the best, in terms of Pareto optimality, EPA placement choice and the best order by which event tuples will be pulled from sites for each query portion and site subset. As soon as our optimizer comes up with the best possible plan, respective information is disseminated and properly transformed portions of the EPN are installed in each site running respective cloud platform deployments. To our knowledge, the FERARI optimizer is the only one incorporating the above advanced features for efficient plans in inter-cloud CEP processes.

FERARI Intra-cloud CEP: FERARI 's intra-cloud processing is built on top of a state-of-the-art streaming cloud platform, namely Apache Storm [1]. In other words, each site is assumed to run a Storm installation on which FERARI intra-cloud topology is created. A site's Storm topology, shown in Figure 2, is comprised of the following components [7]:

- Input Spout: A Storm spout where streaming tuples arrive or pushed events from other sites are fed into the CEP Engine.
- CEP Engine: Receives the input events from the Input Spout and having processed them according to the locally installed EPAs, emits derived events towards the Time Machine component. Details of our CEP Engine follow shortly.
- Time Machine: A Storm Bolt that buffers derived events from the CEP Engine.
- Gatekeeper: A Storm Bolt responsible for advanced calculations and distributed CE resolution procedures.
- Communicator: A Storm Bolt responsible for the push/pull based communication to/from sites in the same subsets, according to the parts of the query plan that are processed.

The CEP Engine module of our prototype, namely ProtonOnStorm [3], was built by IBM Research - Haifa within the scope of the FERARI project [2] and is an open source CEP Engine that extends the IBM Proactive Technology Online (Proton) standalone platform [3]. ProtonOnStorm's architecture is distributed across a number of Storm bolts (see Fig. 4) which allows for different degrees of parallelization in different modules. In that, ProtonOnStorm is a more elastic intra-cloud processing solution maximizing the potential for efficient intra-cloud CEP. Upon the reception of an incoming event, multiple independent parallel instances of the routing bolt of ProtonOnStorm, determine the metadata that should be assigned to the received event, the EPA name and the context name, which are added to the event tuple. ProtonOnStorm uses

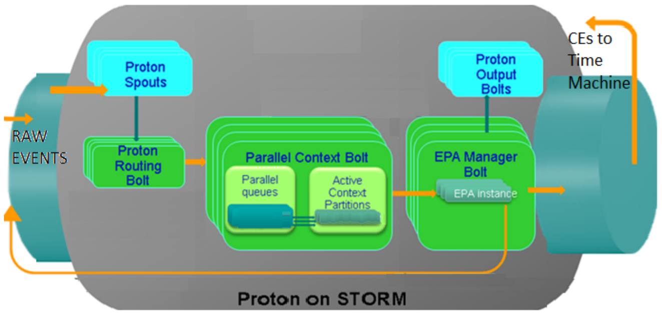


Figure 4: Intra-cloud FERARI - The CEP Engine

the STORM field grouping option on the metadata routing fields - the agent name and the context name- to route the information to the context processing bolt (see Fig. 4). After queuing the event instance in order to solve issues of out of order reception and parallelize processing of the same instance where possible by different EPAs, the event tuple is processed by the context service and the relevant context partition id is added to it. At this point, ProtonOnStorm uses the field grouping on context partition and agent name fields to route the event to specific instances of the relevant EPA, this way performing data segmentation. If a CE is detected that needs to be transmitted to a remote cloud, it will be routed to the Time Machine component (Fig. 2) and it will be pushed to the site responsible for synthesizing events from other sites in the same subset upon request, according to the inter-cloud execution plan.

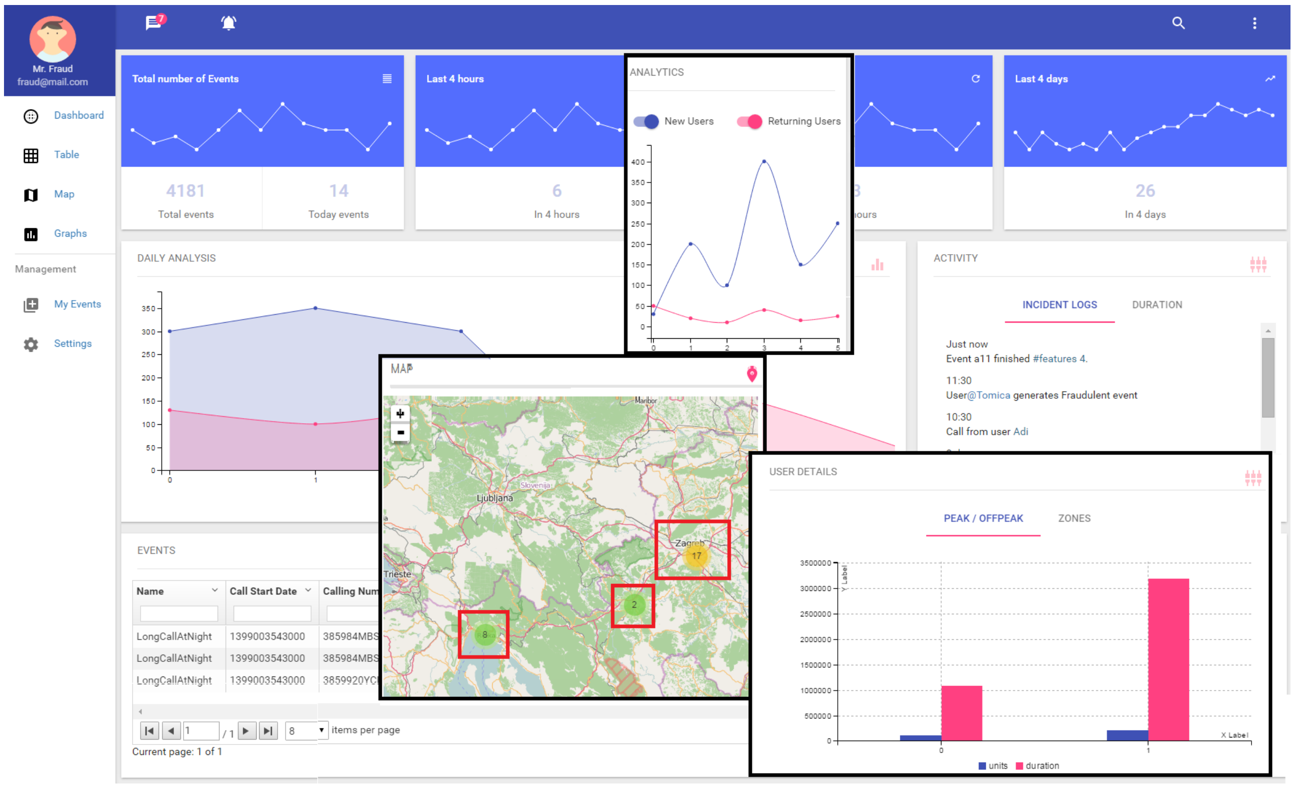
FERARI Dashboard: The web-based dashboard of our prototype is generic enough to produce a wide variety of reports and analytic results that are useful from an application viewpoint, irrespectively of the actual details of the CEs, context or EPAs. It simply needs to be defined as an event consumer using a RESTful API [3]. FERARI Analytics come both in numerical, tabular format and graphical representations, while map visualization combines aggregate event statistics with their spatial reference, i.e., regarding the site of the multi-cloud platform where interesting situations occur.

3. DEMO SPECIFICATIONS

For the purposes of the demonstration, we will use a real, anonymized, dataset of call data coming from HT's network. HT data include around 2 TB of Call Detail Records (CDR), 600 GB of Postpaid usage and more than 1.5 TB of Prepaid CDR. In a simulated multi-cloud deployment, each site will be fed with a distinct local stream that comprises a set of calls recorded by a group of HT's telecommunication antennas, where groups are non-overlapping i.e., they share no antennas. During the demo, users will be able to interact with all the components of FERARI architecture in a real use case scenario involving mobile fraud detection.

CE Query Formation: we target at enabling both quick exploratory interaction with FERARI, as well as provide the potential for allowing users dig further into our prototype's querying and CEP capabilities. To achieve that, we use real (properly masked to comply with company policies) fraud detection rules from HT to a priori build a set of EPAs. In the pre-built scenario (see Fig. 3) users will be able to express their queries by combining the already available EPAs, defining event producers and consumers and visualizing the created EPN in FERARI 's authoring tool. Event patterns expressing masked HT's rules for mobile fraud detection involve:

- *LongCallAtNight:* Report long calls (defined as longer than X minutes) to premium locations during night hours (limited by a pair of timestamps).



Just now
Event a11 finished #features 4.
11:30
User@Tomica generates Fraudulent event
10:30
Call from user Adi

Name	Call Start Date	Calling Num
LongCallAtNight	1399003543000	385984MBS
LongCallAtNight	1399003543000	385984MBS
LongCallAtNight	1399003543000	3859920YC

Current page: 1 of 1

Figure 5: Screenshots of FERARI Dashboard

- *FrequentLongCallsAtNight:* Raise alarm upon the occurrence of at least Y calls made to premium locations during night hours, lasting longer than X minutes per calling number.
- *FrequentLongCalls:* Notify when at least Z calls made to a premium location sum up to at least X minutes duration in a day.
- *FrequentEachLongCall:* Notify when at least Z long, at least X minutes each, calls are made to a premium location in a day.
- *ExpensiveCall:* Every X minutes provide notifications in case calls dialed to premium locations sum up to more than a predefined cost per calling number.

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 modifying existing or define from scratch CEs, Contexts (temporal, segmentation or composite), EPAs and finalize their queries by graphically exploring the EPN that will be submitted to FERARI.

Assessing Inter- and Intra-cloud Internals: In order 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, which includes:

- FERARI optimizer's steps in choosing the best query plan as well as logical (including query rewriting in terms of the produced JSON file), physical plans and corresponding execution costs of other alternatives,
- real-time logs of inter-cloud push/pull communication according to the plan that has been prescribed by FERARI 's optimizer,
- real-time logs of intra-cloud activity via ProtonOnStorm's routing and EPA Manager bolts (Fig. 4).

CE Analytics: Real-time CE notifications will be provided via FERARI 's analytics platform. Nonetheless, the functionality of our prototype's dashboard includes much more than mere notifications of fraud event reports with detailed feature descriptions. Users will be able to access reports of numerical and graphical results about detected mobile fraud CEs as shown in Figure 5. More precisely, aggregate statistics of suspicious calls and user records will be provided at different temporal granularities in an online fashion (Fig 5). Furthermore, map visualization tools will account for the spatial dimension of the analysis, regarding the position of HT 's telecommunication antennas where fraud incidents occur or sets of antennas where suspicious users span as they commute.

4. REFERENCES

[1] Apache Storm project homepage. http://storm.apache.org/.
[2] Flexible event pRocessing for big dAta aRchItectures. the FERARI project. http://www.ferari-project.eu/.
[3] IBM Proactive Technology Online on STORM. https://github.com/ishkin/Proton.
[4] Storm-Esper. https://github.com/tomdz/storm-esper.
[5] J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman. Efficient pattern matching over event streams. In *SIGMOD*, June 2008.
[6] M. Akdere, U. Çetintemel, and N. Tatbul. Plan-based complex event detection across distributed sources. In *PVLDB*, 2008.
[7] 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.
[8] O. Etzion and P. Niblett. *Event Processing in Action*. Manning Publications Company, 2010.
[9] 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.