The Lannion report on Big Data and Security Monitoring Research

Laurent d'Orazio, Jalil Boukhobza, Omer Rana, Juba Agoun, Le Gruenwald, Hervé Rannou, Elisa Bertino, Mohand-Saïd Hacid, Taofik Saïdi, Georges Bossert, et al.

The Lannion report on Big Data and Security Monitoring Research

Laurent d'Orazio
Univ Rennes, CNRS, IRISA, B-Com
Lannion, France
laurent.dorazio@univ-rennes1.fr

Juba Agoun
Université de Lyon, CNRS, LIRIS
Villeurbanne, France
juba.agoun@univ-lyon1.fr

Elisa Bertino
Department of Computer Science
Purdue University
West Lafayette, United-States
bertino@purdue.edu

Georges Bossert
SEKOIA
Rennes, France
georges.bossert@sekoia.fr

Jalil Boukhobza
ENSTA Bretagne
Lab-STICC, CNRS, UMR 6285
Brest, France
jalil.boukhobza@ensta-bretagne.fr

Le Gruenwald
School of Computer Science
University of Oklahoma
Norman, United-States
ggruenwald@ou.edu

Mohand-Saïd Hacid
Université de Lyon, CNRS, LIRIS
Villeurbanne, France
mohand-said.hacid@univ-lyon1.fr

Van Long Nguyen Huu
Nokia Bell Labs, B-Com
Lannion, France
long.nguyen$_h uu@nokia.com$

Makoto Onizuka
Graduate School of
Information Science and Technology,
Osaka University
Osaka, Japan
onizuka@ist.osaka-u.ac.jp

Omer Rana
School of Computer Science and Informatics
Cardiff University
Cardiff, Wales
rana@cardiff.ac.uk

Hervé Rannou
SenX
Guipavas, France
herve.rannou@senx.io

Taofik Saïdi
Nokia
Lannion, France
taofik.saidi@nokia.com

Dimitri Tombroff
Thales Service
Thales
Rennes, France
dimitri.tombroff.e@thalesdigital.io

***Abstract*—During the last decade, big data management has attracted increasing interest from both the industrial and academic communities. In parallel, Cyber Security has become mandatory due to various and more intensive threats. In June 2022, a group of researchers has met to reflect on their community's impacts on current research challenges. In particular, they have considered four dimensions: (1) dedicated systems being data processing and analytic platforms or time series management systems; (2) graphs analytics and distributed computation; (3) privacy; and (4) new hardware.**

***Index Terms*—Cyber security, big data, security monitoring**

I. Introduction

In recent years, research in databases and distributed systems has resulted in systems allowing the processing of large volumes of data on large-scale infrastructures such as Cloud/Fog/Edge Computing (MapReduce, HBase, Spark, etc.). At the same time, data management on new hardware (GPU, FPGA for example) has been developed.

The objective of this paper is to discuss the research directions on security with big data. Specifically, the workshop aims to answer the two key questions: how can Big Data technologies be applied to security monitoring? and What are the research challenges in this domain?

In particular, we consider the following directions:

- Systems: we discuss some representative solutions within the Big Data and security monitoring ecosystems. In particular, we present open-source solutions, namely Punch and Warp10.
- Graph analytics and distributed computation: among theoretical dimensions that are covered by large-scale security monitoring, we discuss existing solutions in graphs analytics and distributed processing.
- Privacy: while providing tools to detect anomalies and

Identify applicable funding agency here. If none, delete this.

threats, it is mandatory to guarantee another important aspect of cyber security, namely privacy.
- Hardware: data analysis can be accelerated thanks to new hardware. We particularly consider three possible opportunities: NVM, GPU and FPGA.

This paper is organized as follows. Section 2 describes the motivations. Section 3 focuses on systems. Section 4 then considers graphs and distributed processing. Section 5 is dedicated to privacy. Section 6 presents new hardware. Finally, Section 7 provide conclusions.

II. Motivation

The development of Computer Science in our daily life have increased the chance for possible attacks. For example, while smart cities tend to provide energy efficient resources management, they may lead to possible black out.

In such a context, network and telecommunication are of particular interest. On one hand, 5G improves security and privacy, while on the other hand, the number of use cases and the system complexity are also increased: new radio; control and user plane split; network slicing eMBB , URLLC and MIoT; Service Based Architecture; Software Defined Networking, or even Network Functions Virtualization. More details on 5G System Security Analysis are provided in [1].

This section introduces the main concepts of this report, namely security monitoring and its link with data management in cloud computing. It describes big data technologies in cloud computing that can be used as building blocks.

A. Security monitoring

Security monitoring consists in collecting and analyzing indicators to detect potential security threats. As an example, administrators may access dashboards to analyze logs. Figure 1 presents some examples of data collected from some logs for a HTTP server.

In such a context, the monitoring tool is based on different kinds of queries. They can consist of range queries for instance to get the traffic for a given period of time (for example between 0:00:00 and 1:00:00 first of July). They may also include aggregation queries for instance to spot the machine with the biggest traffic.

B. Cloud Computing and Big Data

Security monitoring experts are directly concerned with Big Data issues. Indeed, they have to efficiently monitor various systems and their number is growing, in particular due to the always increasing amount of IoT. In addition, they are interested in collecting and storing data for long time periods to be able to detect sophisticated intrusions with events spread on large time frames.

During the last decade, many data management systems have been proposed so as to address the specific behaviors of cloud computing [2] and Big Data, in particular the scalability/elasticity, to exploit resources in large data centers. Some solutions consist in systems with a declarative language on top of distributed file systems like Pig [3], SCOPE [4], Hive [5] or Jaql [6]. Others consist in a column-store like Big Table [7] or Cassandra [8]. On top of these stores, different execution engines can be deployed such as MapReduce [9], Tez [10], Spark [11] or Flink [12]. These tools can be seen as important building blocks to manage large number of logs in security monitoring.

III. Systems

The Big Data community is dynamic and thus the ecosystem is rich. As an illustration, one can see the representation given by Matt Truck[1].

In this section, we will cite a few representative systems of classes of solutions that are currently used in security monitoring as for the reader to have an overview of tools being used in the context.

A. Security monitoring software

Dedicate security monitoring solutions enable to collect, index and correlate data to create collections. These collections are then used to generate reports and alerts, in addition to visualizations and dashboards. Splunk[2], ElastisSearch[3] and SEKOIA are representative software.

SEKOIA[4] for example, makes it possible for teams of experts in cyber security to focus on anticipating and automatising.

B. Data processing and analytics platforms

The PunchPlatform is a big data platform with an emphasis on industrial deployment, end-to-end critical data processing, and data analytics applications. It provides a data pipeline concept implemented on top of Apache Kafka[5]. It enables users to design industrial data pipeline where they plug in their processing where they need, from simple data tagging to complex distributed machine learning algorithms.

One of its key difference with a ElasticSearch-Logstash-Kibana (ELK) setup is to let the user deploy arbitrary processing in Storm [13] and Spark engines, not just logstash filters. The PunchPlatform comes equipped with ready-to-use log parsers. These are deployed automatically in the stream of data. In addition, scripting language is available to write personal parsers on the PunchPlatform

The platform enables search (using Kibana or the native ElasticSearch APIs) as well as machine learning. It provides additional functions such as multi-tenancy, long term archiving using CEPH object storage, multi site deployment.

C. Time series database

A time series is a series of data points indexed (or listed or graphed) in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Thus it is a sequence of discrete-time data. Examples

[1]https://mattturck.com/data2020/
[2]https://www.splunk.com
[3]https://www.elastic.co
[4]https://www.sekoia.io
[5]https://kafka.apache.org

```
host    logname time    method  url     response        bytes   referer useragent
199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/ HTTP/1.0" 200 6245
unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET /shuttle/countdown/ HTTP/1.0" 200 3985
199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET /shuttle/missions/sts-73/mission-sts-73.html HTTP/1.0" 200 4085
burger.letters.com - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/countdown/liftoff.html HTTP/1.0" 304 0
199.120.110.21 - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/missions/sts-73/sts-73-patch-small.gif HTTP/1.0" 200 4179
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 304 0
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /shuttle/countdown/video/livevideo.gif HTTP/1.0" 200 0
205.212.115.106 - - [01/Jul/1995:00:00:12 -0400] "GET /shuttle/countdown/countdown.html HTTP/1.0" 200 3985
d104.aa.net - - [01/Jul/1995:00:00:13 -0400] "GET /shuttle/countdown/ HTTP/1.0" 200 3985
129.94.144.152 - - [01/Jul/1995:00:00:13 -0400] "GET / HTTP/1.0" 200 7074
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET /shuttle/countdown/count.gif HTTP/1.0" 200 40310
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 200 786
```

Fig. 1. Example of logs of a HTTP server

of time series are heights of ocean tides, counts of sunspots, and the daily closing value of the Dow Jones Industrial Average. A time series database (TSDB) is a software system that is optimized for storing and serving time series through associated pairs of time(s) and value(s). Examples of TSDB include InfluxDB[6] and Prometheus[7]. In this paper, we will focus on Warp 10.

The Warp 10[8] core platform is built to manage and simplify time series processing. It includes a Geo Time Series™ database and an analytic engine. Each one can be used separately but of course can be combined they work great together. Each measurement has a specific time, a value and optional spatial metadata such as geographic coordinates and/or elevation. Those augmented measurements are what we call Geo Time Series™ (GTS).

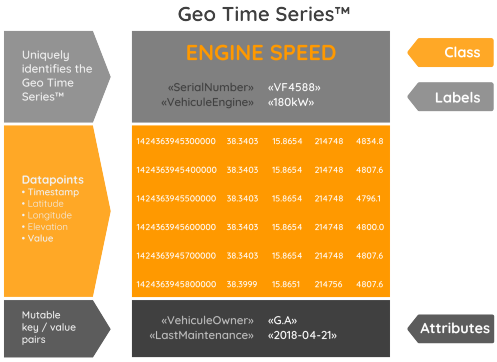

Fig. 2. Warp 10™ Geo-Time Series representation

Figure 2 illustrates a GTS. The header introduces the class (engine speed) and some labels (serial number and vehicle engines). The data points consist of a timestamp, a latitude, a longitude, an elevation and a value. The bottom part provide additional key/value pairs (vehicle owner and last maintenance).

Warp 10™ dedicated analytics language, called WarpScript™, is specifically designed for complex analysis of Time Series data at any scale. An example of Warp 10 is illustrated by Figure III-C. WarpScript™ provides more than 1000 functions ranging from simple statistics to complex algorithms such as pattern and anomaly detection.

WarpScript™ is a data flow programming language. This differs from a query language which can fetch data and perform simple computations. WarpScript™ has the power of a Turing Complete programming language, with conditionals, loops, and asynchronous transfer of control. The WarpScript™ Execution Engine is the part of the Warp 10™ Platform which runs the WarpScript™ code, simply submit code via a HTTP request, the engine will execute it close to where the data reside and will return the result as a JSON object which can be integrated into an application. The WarpScript™ language is also integrated in many existing data processing languages and environments such as R, Python, Zeppelin, Jupyter, Spark, Pig, Storm or NiFi, with the ability to use the same WarpScript™ code across all those environments. This greatly increases efficiency.

```
[
  $read_token
  'gov.noaa.storm.wind'
  {}
  '2015-02-01T00:00:00.000z'
  '2015-01-01T00:00:00.000z'
] FETCH 'fetch_wind' STORE
[ $fetch_wind bucketizer.mean NOW 1 d 0 ]
BUCKETIZE 'bucketizedWind' STORE
```

In WarpScript™, as in all concatenative programming languages, every expression is a function and juxtaposition of expressions denotes function composition, in the spirit of how Unix pipelines work. WarpScript™ manages a data pipeline with which all functions interact, getting their arguments by emptying the pipeline, and filling it with their results. The data pipeline is passed from function to function during the execution of WarpScript™ code. The Warp 10™ Analytics Engine, the WarpScript™ Execution Engine of Warp 10™ accessed via the /api/v0/exec endpoint, returns a JSON list which contains the elements of the data pipeline after the code execution. The first element of this list contains the most recent element of the pipeline.

IV. Graph Analytics and Distributed Computation

A. Graph analytics

Graph is a fundamental data structure consisting of nodes and edges. Graphs are ubiquitous in various domains, such as

[6]https://www.influxdata.com/
[7]https://prometheus.io
[8]https://www.warp10.io

web graph [14], social networks [15], computer vision [16], and gene expressions [17]. The typical tasks for graph analytics are graph clustering, graph classification, and link prediction. Graph/node embeddings and graph neural networks (GNNs) are widely used for such tasks to achieve high quality of analysis results. Node embedding is a technique to embed each node in a graph to a multi-dimensional space that captures the node proximity in the graph. The proximity of a node is defined based on its node features and its topology of neighbors. That is, two persons in a social network should have similar embeddings if their features (such as hobbies) are similar and they are close friends. Such proximity is known to be effective for the above graph analytic tasks. In particular, there have been various algorithms that leverage different types of proximities, 1st-order/2nd-order proximities (Microscopic aspect) and higher-order proximities (Macroscopic aspect). As an example, Spectral clustering [18] leverages the 1st-order proximity (1-hop distance) to compute graph clusters. SCAN [19] is another type of density-based clustering technique that utilizes the 2nd-order proximity (2-hops distance). The recent GCNs (graph convolution networks) usually utilize 1st- and 2nd-order proximities, called two-layer GCNs, for semi-supervised node classification. GCNs gave a large influence over a large number of following papers, however, they are known to have two drawbacks when they increase the number of convolution layers in order to effectively utilize a small number of labels: over-fitting and over-smoothing. Many-layer GCNs tend to be overfitting because their number of parameters is large. Also, they tend to be over-smoothing: many times convolution operations cause the embedding of all nodes to be similar, which makes it difficult to decide the class boundary of classification tasks.

To tackle these problems, we propose Adaptive Node Embedding Propagation Network (ANEPN) [20] that utilizes high-order proximities by adaptively increasing the number of propagation hops. The novelty of ANEPN is that it keeps the number of layers at two (which avoids overfitting) and it learns models by minimizing the combined loss of the proximity loss and anti-proximity loss (which successfully separates the number of convolution operations from keeping proximities in order to avoid over-smoothing).

We also work on benchmarking various GCNs [21] in order to clarify their pros/cons using synthetic graph generator [22]. By generating various synthetic graphs, we reveal that 1) GNNs including the state-of-the-art methods suffer from a class imbalance problem that typically deteriorates the performance of multi-class classification, and 2) GNNs generalizing to graphs with few edges within each class (we call heterophilic graphs) provide marginal classification performance gains in a heterophily setting from a graph-agnostic classifier, such as MLP (multi-layer perception).

B. Distributed graph analytics

Cloud computing is a promising approach for making graph analytics on large-scale data. There are two fundamental issues with graph analytics using distributed servers in the cloud. First is a graph partitioning problem. The distributed graph engines execute analytic processes after partitioning input graph data and assigning them to distributed computers, so the quality of graph partitioning largely affects the communication cost and load balance among computers during the analysis process. We developed an effective graph partitioning technique [23] that achieves low communication cost and good load balance among computers at the same time; It produces balanced clusters by extending an efficient modularity-based graph clustering [24]. We implemented our technique on top of distributed graph engine, PowerGraph, and made intensive experiments. The results show that our partitioning technique reduces the communication cost so it improves the response time of graph analysis patterns.

Second, graph analytic algorithms or machine learning techniques are iterative by its nature, however, major distributed frameworks, such as MapReduce or Spark are not optimized for iterative processing. We propose OptIQ [25], a query optimization approach for iterative queries in a distributed environment. OptIQ removes redundant computations among different iterations by extending the traditional techniques of view materialization and incremental view evaluation. We verify the effectiveness of OptIQ through the queries of PageRank and k-means clustering on real datasets. The results show that OptIQ achieves high efficiency, up to five times faster than is possible without removing the redundant computations among iterations.

V. Privacy and security aware data sharing

Technological advances, such as IoT devices, cyber-physical systems, smart mobile devices, cloud systems, data analytics, social networks and increased communication capabilities, are making possible to capture, and to quickly process and analyze huge amounts of data from which to extract information critical for many critical tasks, such as healthcare security and cyber security. In the area of cyber security, such tasks include user authentication, access control, anomaly detection, user monitoring, and protection from insider threat. By collecting and mining data concerning user travels, contacts and disease outbreaks one can predict disease spreading across geographical areas. And those are just a few examples.

A. Privacy

The use of data for those tasks raises however major privacy concerns. Collected data, even if anonymized by removing identifiers such as names or social security numbers, when linked with other data may lead to re-identify the individuals to which specific data items are related to. Also, as organizations, such as governmental agencies, often need to collaborate on security tasks, data sets are exchanged across different organizations, resulting in these data sets being available to many different parties. Privacy breaches may occur at many different layers (e.g., networks, hosts, applications) and components in our interconnected systems. An example of a privacy attack in the context of cellular network is the ToRPEDO side-channel attack that exploits the paging protocol to track users [26].

On the other hand if we consider mobile applications, these applications have vulnerabilities, such as the ones related to authentication [27], [28], resulting in lack of security, which then undermines privacy. It is important to mention that security and privacy are two different requirements. However security is a pre-requisite for privacy. The use of machine learning techniques further threatens privacy because of attacks such as the inversion ones by which a party can infer the sensitive contents of the data samples used for training. Finally, the increased adoption of wearable devices and continuous data streaming from these devices allows a party to collect fine-grained geo-temporal data about individuals.

So it would seem that achieving privacy today would be impossible. However many different privacy techniques have been proposed over the years, including privacy-preserving data linkage techniques, protection against ML inversion attacks, network anonymizers, secure multiparty computation (SMC) techniques, homomorphic encryption, privacy-preserving digital identity management, including pseudonym systems, access control (AC) punctuations for streaming data [29], and anonymous "mode" for mobile applications [30]. So the main question is: *what is needed to achieve privacy?* What is needed is to combine those approaches for "privacy protection in depth" by developing holistic privacy-preserving environments. Indeed users consider privacy important but they often feel that privacy is complex to manage. However a key question is "personal privacy versus collective safety"? More specifically: (i) How can we make possible for people to make their choices about this question? (ii) How can we make possible to reconcile those two seemingly opposing goals? We believe that answering these questions is challenging and requires approaches making easier for the users to understand risks/benefits of releasing some of their personal data as well mechanisms by which users can be made aware of how their data is used and can participate to the related processes. Data transparency [31] and policy-based use of data are two key elements relevant to these issues.

B. Data sharing aware of access control policies

With the advent of recent information technologies and the increase in the amount of data, more and more companies and organisations are cooperating through data sharing and exchange for learning and research purposes. To effectively ensure security and privacy, data owners attach a set of rules defined as an access control policy [32]. However, when data is shared between several sources, there may be overlapping data. These redundancies can be a threat when records from the same entity are not considered at the same level of confidentiality [33]. Toward this situation, appropriate filtering of responses to a query must be introduced. Therefore, to ensure data security and confidentiality, each source, having been built independently of the others, defines its own access control policy. The latter provides information that is considered sensitive and therefore not to be disclosed.

The focus of our research is to design and implement a framework that allows secure data sharing between two sources [34]. Data sharing is based on the establishment of mappings between entities of two sources. We are interested in using entity matching rules [35] between instances in order to augment the result of queries while ensuring the enforcement of security policies. Furthermore, we seek to bridge the security gap that emerges when two records, from different sources, that represent the same real-world entity are not considered with the same degree of sensitivity.

We first study the problem of data publishing in the presence of access control rules. We consider the context where a data source is described by a set of publication views and access restriction rules. A view is a table representing a query result intended to be published. The objective is to detect views that leak sensitive information and rather than neutralizing them, we propose a view revision [36]. Our approach uses the necessary and sufficient conditions for a view to comply with a policy request. We formulate a preliminary work that consists of a data-independent method to review views that do not preserve privacy. The goal of this revision process is to strike a balance between data restricting access and data availability.

Subsequently, we propose an entity matching-oriented and policy-oriented methodology to provide a secure data sharing framework [37], [38]. We present an algorithm for translating a query submitted against one schema into an augmented query for the other schema to capture concerning tuples, based on entity matching rules. Then, we provide a methodology to answer queries while maximizing sharing and preserving local access control policies by avoiding any inference leakage that could result from entity matching.

VI. Hardware

A. New memory and storage technologies

New memories are emerging and heterogeneity of storage systems is increasing. In a data centric context, applications need more processing power for managing large amounts of data. Many-core systems were adopted to increase the processing power. However, this further puts pressure on the memory subsystems to perform in an efficient way. Fortunately, the revolutions of chip manufacturing and hardware design make several emerging ultra-low latency and ultra-high capacity non-volatile memory/storage devices (e.g., Intel Optane [39] and OCSSD [40] already or almost on-the-market). These new devices can blur or even eliminate the boundary between the processing and I/O layer in the Von-Neumann architecture. Nevertheless, the design of legacy software stack (e.g., redundant buffering from devices to system and applications) leads to a serious burden and hinders data-intensive applications system performance enhancement by taking full advantage of emerging memory/storage devices.

Those new emerging memories are disruptive in a sense that they push the scientific community to revise the way algorithms and applications are designed. Flash memory is already considered as a mature technology deployed as a high performance storage media in embedded systems, Cloud and High Performance Computing (HPC) infrastructures, or

even in common laptops. Several traditional algorithms and applications were revised to take full advantage of the Flash memory properties [41], such as embedded systems [42], databases [43], Cloud storage systems [44], or HPC [45].

Other Non-Volatile-Memories (NVMs) are under study to complement the traditional memory hierarchy [46]. Some examples can be found in embedded systems where MRAM (Magnetoresistive RAM [47]) could take profit of the low leakage power to enhance the energy efficiency for several applications. FeRAM (Ferroelectric RAM) is already used in several DSP based platforms or in automotive equipment for data recording, it is a good candidate for wearable electronics [48]. Phase-Change Memory (PCM) [49], based on the resistivity of chalcogenide alloy to represent bits, is already widely deployed in intel Optane SSDs thanks to its interesting performance figures and energy properties. Resistive memory (ReRAM), being compatible with conventional semi-conductor fabrication processes, belongs to the class of memristors [50], [51] and is also under study for more than a decade.

In addition to those new technologies that optimize the latency, transfer rate and response time of accessing data, some new research paths consist to avoid such transfers by whether deploying computation power near storage or computing near memory. Computational storage devices make it possible to run software within the storage device, offloading the CPU, memory and busses from such a burden [52]. In-memory computing makes it possible to have computing power within the memory array [53] avoiding data transfers between memory and CPU.

The rise of those new memory and storage technologies makes the data supervision and security necessarily more challenging due to the heterogeneity of storage and memory technologies, the heterogeneity of the software stack deployed on top of them, and the disparity of applications used.

B. New processing

1) Semantic caching framework towards FPGA acceleration: With the emergence of new data management systems (DMS) in context of *big data and cloud computing*, caching data has become important since it can reduce unnecessary query execution. Given the relatively high latency in communication between the compute and the storage layer of the DMS, we consider that caching data at the compute layer now has become more important. Thus, we require a caching framework as a cache management system (CMS) in the middleware layer of DMS, which facilitates the construction process of cache services in a variety of applications.

However, the frameworks of cache services have been studied, most of them are presented with respect to traditional caching mechanisms (page or block cache), such as Adaptable Cache Services (ACS) [54], Amazon Elastic Cache [55], and in-memory cache Redis [56]. As an alternative approach, semantic caching (SC) allows to exploit the resources in the cache and knowledge contained in the queries [57]. Thus, SC can be seen as a candidate to be exploited in middleware layer of DMS. Nevertheless, the complexity of query rewriting in SC (i.e., NP-complete problem [58]–[60]) can induce a high overhead because of its excessive computations. Noticeably, beside the algorithm of query rewriting, the capability of the infrastructure plays an important role to solve the presented issue in SC. Moreover, none of prior studies, such as [61]–[71] exhibits SC as cache management system (CMS) in the middleware layer of DMS.

To overcome the problem of query rewriting in SC, there are two approaches in terms of scaling-up (vertical scaling) with hardware resources. In particular, upgrading Central Processing Units (CPUs) with multiple-cores technology or using more of them as the first approach. Since CPU has "power wall" limitation (Moore's law), as an alternative approach, replacing or accelerating with high computing specialized hardware, in particular, Field Programmable Gate Arrays (FPGAs) have been proposed. FPGAs have been noted to be good candidates for their high parallelism of multi-tasks, reconfigurability, low power consumption, and can be attached to the CPU as an IO device accelerator [72]. Additionally, FPGAs have been accepted gradually in many studies including accelerations of database analytic, Massive Parallel Processing (MPP) or even commercial products ([73]–[79]). Nonetheless, Field Programmable Gate Arrays (FPGAs) have not been considered yet to handle the excessive computations of query rewriting in SC.

Therefore, we aim to combine three aspects: CMS framework, SC and FPGA-based database acceleration together to accelerate range query processing in the domain of massive distributed data. According to this goal, we achieve the following contributions: 1) ModulAr Semantic CAching fRAmework (MASCARA) in the middleware layer of DMS [80]. MASCARA divides and regroups the functionalities, computations and procedures of SC into modules and stages. More precisely, this work can be done by defining templates, data structures and interfaces. Thus, the main contribution of this architecture is about the flexibility, scalability and adaptability to different environments, infrastructures and requirements. 2) A coalescing heuristic with a new replacement value function in terms of cache management of MASCARA [81]. In particular, the heuristic can decide when to coalesce data regions based on the recency of usage (temporal locality) and percentage of response contribution (spatial locality) that are presented through a new replacement function. It strikes a good balance in different aspects, such as, response time, hit ratio and cache space usage compared to the conventional approaches. 3) Multi-view processing to handle select-project-join query in MASCARA. In particular, this procedure brakes down an original (inner) join query into (select-project) subqueries that belong to different joined relations or views. 4) A cooperative model, called MASCARA-FPGA (as can be shown in Figure 3, where query processing is accelerated regarding query rewriting and part of query execution [82].

As it can be seen, query rewriting of MASCA, called Query Trimming, now is offloaded and accelerated on FPG within the bottom-to-top pipeline execution. In detail, the two sub-stages,

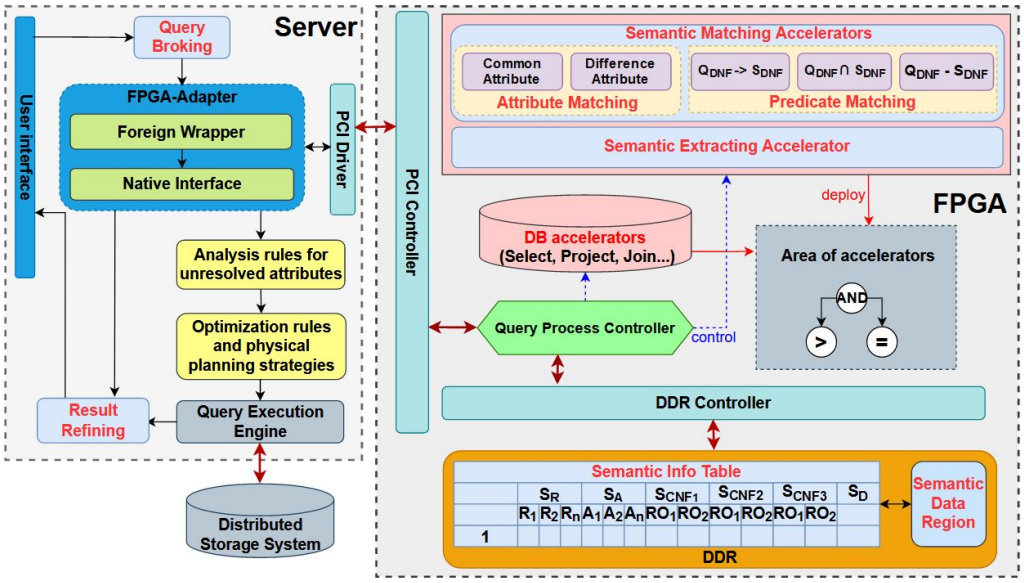

Fig. 3. Cooperative model MASCARA-FPGA with its major components.

Semantic Matching and Semantic Extracting are converted to corresponding accelerators. For example, Semantic Matching category can consist of the accelerators (also called kernels) for two main functions: Attribute Matching and Predicate Matching. It is worth to note that a kernel can be divided into smaller accelerated engines to increase the level of task parallelism on FPGA. In particular, the kernel Predicate Matching can be split to three different functions as engines: *Intersection* ($Q_{DNF} \wedge S_{DNF}$), *Implication* ($Q_{DNF} \rightarrow S_{DNF}$) and *Difference* ($Q_{DNF} - S_{DNF}$). Other stages of MASCARA (i.e., Query Broking, Result Refining) do not cause any computing overhead, they thus do not need to be accelerated on FPGA. Besides query rewriting, MASCARA-FPGA also executes in parallel a list of generated probe queries thanks to the database (DB) operators on FPGA, such as filter, project and sort-merge-join. As complement of the accelerators, we also organize and manage SC on off-chip memory (i.e., Dynamic Random Access Memory DRAM) of FPGA since it has sufficient space for big data applications (e.g., more than 64GB) and provides a high bandwidth connection to kernels. Additionally, MASCARA-FPGA consists of a FPGA-Adapter which can encapsulate or wrap the native functionalities into high-level interfaces. More precisely, FPGA-Adapter bridges the gap between high-level application of MASCARA and low-level accelerators on FPGA by leveraging on Java Native Interface (JNI). Finally, a small component called Query Process Controller (QPC) is responsible for the management of the returned signals, statuses and results within the workflow of Query Trimming and execution of probe query on FPGA.

2) Real-Time Outlier Explanations for Data Streams on GPUs: In many monitoring applications, such as sensor networks for biological, earth, and environmental data collection and analysis, and power usage monitoring, data are in the form of streams. A data stream is a sequence of data points with timestamps that has many special characteristics including infinite size, transiency, uncertainty, dynamic distribution, and multi-dimensionality. For applications involving multiple related streams, additional characteristics exist, such as asynchronous data arrival, dynamic relationships among streams, and schema heterogeneity. Although data streams are different from (regular) non-stream data in many respects, they are not free of outliers. An outlier is a data point that is significantly different from other data points in the same dataset. Practically outliers are inevitable in any data acquisition process as they can be introduced because of numerous reasons, such as malicious activity or instrument error. Thus, outlier detection is an important part of the data analysis process. Instead of making outlier detection as a black box, for outlier detection to be beneficial, explanations about the outliers discovered need to be provided to the user.

There are three major types of outlier explanations [83]: outlier causality, outlying attributes, and outlier ranking. Outlier causality explains what causes a data object/event to be an outlier. This can be derived from examining the causal interactions among outliers as an outlier can cause another outlier to occur as well as the causal interactions between inliers and outliers as an inlier can also cause another outlier to occur. Outlying attributes refer to the attributes that are responsible for the abnormality of outliers. The explanation in this case can be given as a set, each member of which consists of an attribute and a score indicating the attribute's contribution to the outlierness of the detected outlier. Outlier ranking reveals the importance level of each outlier in a set of detected outliers. Most of the existing outlier explanation algorithms provide one type of explanations in isolation, such as causal interactions among outliers in [84], [85], outlying attributes in [86], [87], and outlier ranking in [88], [89]. None

of the algorithms provides all three types of outlier explanations [83]. In addition, the number of existing algorithms that explain outliers in data streams is very limited and they focus mostly on outlying attributes only [88], [90], [91], [92].

Outlier explanation for data streams is more difficult than for regular data because of data streams' characteristics and their Big Data challenges, which call for online and scalable parallel algorithms to explain outliers in real time, as close in time as possible to the arrival of data, so that timely actions can be taken for applications. However, existing work in outlier explanations addresses only some characteristics of data streams in isolation. In addition, while commodity parallel hardware such as GPUs which offers high computational performance has become popular and affordable, no existing work on data stream outlier explanations is designed for GPUs. The GPU's research challenges for Big Data include low throughput of the host to the GPU interface, small memory space of GPUs, low global memory bandwidth with respect to the number of threads, and load balancing. What is needed in order to advance the state of the art in this area is an integrative outlier explanation algorithm that can address all the data stream characteristics, take advantage of scalable commodity parallel hardware, and at the same time, provide all three types of explanations for the discovered outliers. The algorithm needs to address both the data streams' and the GPU's research challenges for Big Data.

VII. Conclusion

Cyber Security can be seen as a concrete use case of Big Data. As a consequence, various directions can be considered from the perspective of scalability in data management. In this paper, we have first presented some systems related to security monitoring, namely SEKOIA, Punch and Warp10. We have then discussed research directions from the perspective of (distributed) graph analytics. The problem of privacy and access control has then been covered. The last dimension that this paper has taken into account is the hardware acceleration with the possibility to integrate GPU, FPGA or NVM.

Obviously, other directions are worth investigating. To cite a few can be mentioned: knowledge representation, machine learning, artificial intelligence, anomaly and threat detection, spatial-temporal data analysis.

Acknowledgment

This material is based upon work within the PEC (Pôle d'Excellence Cyber) supported by the INRIA and DGA.

References

[1] G. Holtrup, W. Lacube, D. P. David, A. Mermoud, G. Bovet, and V. Lenders, "5g system security analysis," *CoRR*, vol. abs/2108.08700, 2021. [Online]. Available: https://arxiv.org/abs/2108.08700

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, 2010.

[3] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin: a not-so-foreign language for data processing," in *SIGMOD*, Vancouver, BC, Canada, 2008.

[4] R. Chaiken, B. Jenkins, P.-Å. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou, "SCOPE: easy and efficient parallel processing of massive data sets," *PVLDB*, vol. 1, no. 2, 2008.

[5] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Z. 0002, S. Anthony, H. Liu, and R. Murthy, "Hive - a petabyte scale data warehouse using Hadoop," in *ICDE*, Long Beach, California, USA, 2010.

[6] K. S. Beyer, V. Ercegovac, R. Gemulla, A. Balmin, M. Y. Eltabakh, C.-C. Kanne, F. Özcan, and E. J. Shekita, "Jaql: A Scripting Language for Large Scale Semistructured Data Analysis," *PVLDB*, vol. 4, no. 12, 2011.

[7] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A Distributed Storage System for Structured Data," *ACM Transactions on Computer Systems (TOCS)*, vol. 26, no. 2, 2008.

[8] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.

[9] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, 2008.

[10] B. Saha, H. Shah, S. Seth, G. Vijayaraghavan, A. C. Murthy, and C. Curino, "Apache Tez: A Unifying Framework for Modeling and Building Data Processing Applications," in *SIGMOD*, Melbourne, Victoria, Australia, 2015.

[11] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, and M. Zaharia, "Spark SQL: Relational Data Processing in Spark," in *Proceedings of the SIGMOD International Conference on Management of Data*, Melbourne, Victoria, Australia, 2015.

[12] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache Flink™: Stream and Batch Processing in a Single Engine," *IEEE Data Engineering Bulletin*, vol. 38, no. 4, 2015.

[13] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, S. Mittal, and D. V. Ryaboy, "Storm@twitter," in *SIGMOD*, 2014.

[14] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee, "Self-organization and identification of web communities," *Computer*, vol. 35, no. 3, pp. 66–71, 2002.

[15] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[16] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-RNN: Deep learning on spatio-temporal graphs," in *Proceedings of CVPR*, 2016, pp. 5308–5317.

[17] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-supervised graph clustering: a kernel approach," *Machine learning*, vol. 74, no. 1, pp. 1–22, 2009.

[18] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *NIPS*. MIT Press, 2001, pp. 849–856.

[19] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, "SCAN: a structural clustering algorithm for networks," in *KDD*. ACM, 2007, pp. 824–833.

[20] Y. Ogawa, S. Maekawa, Y. Sasaki, Y. Fujiwara, and M. Onizuka, "Adaptive node embedding propagation for semi-supervised classification," in *ECML/PKDD*, ser. Lecture Notes in Computer Science, vol. 12976. Springer, 2021, pp. 417–433.

[21] S. Maekawa, K. Noda, Y. Sasaki, and M. Onizuka, "Beyond real-world benchmark datasets: An empirical study of node classification with GNNs," in *NeurIPS*, 2022.

[22] S. Maekawa, Y. Sasaki, G. Fletcher, and M. Onizuka, "GenCAT: Generating attributed graphs with controlled relationships between classes, attributes, and topology," *CoRR*, vol. abs/2109.04639, 2021.

[23] M. Onizuka, T. Fujimori, and H. Shiokawa, "Graph partitioning for distributed graph processing," *Data Sci. Eng.*, vol. 2, no. 1, pp. 94–105, 2017.

[24] H. Shiokawa, Y. Fujiwara, and M. Onizuka, "Fast algorithm for modularity-based graph clustering," in *AAAI*. AAAI Press, 2013.

[25] M. Onizuka, H. Kato, S. Hidaka, K. Nakano, and Z. Hu, "Optimization for iterative queries on mapreduce," *Proc. VLDB Endow.*, vol. 7, no. 4, pp. 241–252, 2013. [Online]. Available: http://www.vldb.org/pvldb/vol7/p241-onizuka.pdf

[26] S. R. Hussain, M. Echeverria, O. Chowdhury, N. Li, and E. Bertino, "Privacy attacks to the 4g and 5g cellular paging protocols using side channel information," in *NDSS*, 2019.

[27] S. Ma, E. Bertino, S. Nepal, J. Li, D. Ostry, R. H. Deng, and S. Jha, "Finding flaws from password authentication code in android apps," in *ESORICS*, K. Sako, S. A. Schneider, and P. Y. A. Ryan, Eds., 2019.

[28] S. Ma, J. Li, H. Kim, E. Bertino, S. Nepal, D. Ostry, and C. Sun, "Fine with "1234"? an analysis of SMS one-time password randomness in android apps," in *ICSE*, 2021.

[29] R. V. Nehme, H. Lim, and E. Bertino, "FENCE: continuous access control enforcement in dynamic data stream environments," in *ICDE*, 2010.

[30] B. Shebaro, O. Oluwatimi, D. Midi, and E. Bertino, "Identidroid: Android can finally wear its anonymous suit," *Transactions on Data Privacy*, vol. 7, no. 1, 2014.

[31] E. Bertino, S. Merrill, A. Nesen, and C. Utz, "Redefining data transparency: A multidimensional approach," *Computer*, vol. 52, no. 1, 2019.

[32] E. Ferrari, *Access Control in Data Management Systems*, ser. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

[33] H. Elmeleegy, M. Ouzzani, A. K. Elmagarmid, and A. M. Abusalah, "Preserving privacy and fairness in peer-to-peer data integration," in *SIGMOD*, A. K. Elmagarmid and D. Agrawal, Eds., 2010.

[34] A. Kementsietsidis and M. Arenas, "Data sharing through query translation in autonomous sources," in *VLDB*, 2004.

[35] R. Singh, V. V. Meduri, A. K. Elmagarmid, S. Madden, P. Papotti, J. Quiané-Ruiz, A. Solar-Lezama, and N. Tang, "Synthesizing entity matching rules by examples," *PVLDB*, vol. 11, no. 2, 2017.

[36] J. Agoun and M.-S. Hacid, "Data publishing: Availability of data under security policies," in *International Symposium on Methodologies for Intelligent Systems*. Springer, 2020, pp. 277–286.

[37] J. Agoun and M. Hacid, "Data sharing in presence of access control policies," in *CoopIS*, 2019.

[38] ——, "Access control based on entity matching for secure data sharing," *Service Oriented Computing and Applications*, vol. 16, no. 1, 2022.

[39] J. Izraelevitz, J. Yang, L. Zhang, J. Kim, X. Liu, A. S. Memaripour, Y. J. Soh, Z. Wang, Y. Xu, S. R. Dulloor, J. Zhao, and S. Swanson, "Basic performance measurements of the intel optane DC persistent memory module," *CoRR*, vol. abs/1903.05714, 2019. [Online]. Available: http://arxiv.org/abs/1903.05714

[40] A. S. Arka Sharma, Amit Kumar, "Open channel ssd," *FreeBSD Journal*, 2021.

[41] J. Boukhobza and P. Olivier, *Flash Memory Integration: Performance and Energy Issues, 1st Edition*. ISTE Press - Elsevier, 2017. [Online]. Available: https://www.sciencedirect.com/science/book/9781785481246

[42] P. Olivier, J. Boukhobza, E. Senn, and H. Ouarnoughi, "A methodology for estimating performance and power consumption of embedded flash file systems," *ACM Trans. Embed. Comput. Syst.*, vol. 15, no. 4, aug 2016. [Online]. Available: https://doi.org/10.1145/2903139

[43] A. Laga, J. Boukhobza, F. Singhoff, and M. Koskas, "Montres : Merge on-the-run external sorting algorithm for large data volumes on ssd based storage systems," *IEEE Transactions on Computers*, vol. 66, no. 10, pp. 1689–1702, 2017.

[44] D. Boukhelef, J. Boukhobza, K. Boukhalfa, H. Ouarnoughi, and L. Lemarchand, "Optimizing the cost of dbaas object placement in hybrid storage systems," *Future Generation Computer Systems*, vol. 93, pp. 176–187, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17325785

[45] L.-M. Nicolas, L. Thomas, Y. Hadjadj-Aoul, and J. Boukhobza, "Slrl: A simple least remaining lifetime file evicition policy for hpc multi-tier storage systems," in *Proceedings of the Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems*, ser. CHEOPS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 33–39. [Online]. Available: https://doi.org/10.1145/3503646.3524297

[46] J. Boukhobza, S. Rubini, R. Chen, and Z. Shao, "Emerging nvm: A survey on architectural integration and research challenges," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 23, no. 2, nov 2017. [Online]. Available: https://doi.org/10.1145/3131848

[47] E. Kültürsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating stt-ram as an energy-efficient main memory alternative," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013, pp. 256–267.

[48] Y. Fujisaki, "Review of emerging new solid-state non-volatile memories," *Japanese Journal of Applied Physics*, vol. 52, no. 4R, p. 040001, apr 2013. [Online]. Available: https://doi.org/10.7567/jjap.52.040001

[49] "A survey of phase change memory systems," *Journal of Computer Science and Technology*, vol. 30, no. 1, pp. 121–144, 2015.

[50] J. J. Yang and R. S. Williams, "Memristive devices in computing system: Promises and challenges," *J. Emerg. Technol. Comput. Syst.*, vol. 9, no. 2, may 2013. [Online]. Available: https://doi.org/10.1145/2463585.2463587

[51] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[52] A. Barbalace and J. Do, "Computational storage: Where are we today?" Jan. 2021, conference on Innovative Data Systems Research 2020, CIDR 2020 ; Conference date: 11-01-2021 Through 15-01-2021. [Online]. Available: http://cidrdb.org/cidr2021/index.html

[53] S. Mittal, G. Verma, B. Kaushik, and F. A. Khanday, "A survey of sram-based in-memory computing techniques and applications," *Journal of Systems Architecture*, vol. 119, p. 102276, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762121001909

[54] L. d'Orazio, F. Jouanot, C. Labbé, and C. Roncancio, "Building adaptable cache services," in *MGC@Middleware*, 2005.

[55] Amazon, "Amazon elasticache." [Online]. Available: https://aws.amazon.com/elasticache/

[56] Redis, "Redis enterprise cache services." [Online]. Available: https://redis.com/solutions/use-cases/caching/

[57] S. Dar, M. J. Franklin, B. T. Jónsson, D. Srivastava, and M. Tan, "Semantic data caching and replacement," in *Proceedings of the 22th International Conference on Very Large Data Bases*, ser. VLDB '96. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996, p. 330–341.

[58] D. J. Rosenkrantz and H. B. Hunt III, "Processing conjunctive predicates and queries," in *Proceedings of the Sixth International Conference on Very Large Data Bases - Volume 6*, ser. VLDB '80. VLDB Endowment, 1980, p. 64–72.

[59] S. Guo, W. Sun, and M. A. Weiss, "Solving satisfiability and implication problems in database systems," *ACM Trans. Database Syst.*, vol. 21, no. 2, p. 270–293, jun 1996.

[60] S. Guo, W. Sun, and M. Weiss, "On satisfiability, equivalence, and implication problems involving conjunctive queries in database systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 4, pp. 604–616, 1996.

[61] M. Stonebraker, A. Jhingran, J. Goh, and S. Potamianos, "On rules, procedure, caching and views in data base systems," in *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '90. New York, NY, USA: Association for Computing Machinery, 1990, p. 281–290.

[62] P. M. Deshpande, K. Ramasamy, A. Shukla, and J. F. Naughton, "Caching multidimensional queries using chunks," *SIGMOD Rec.*, vol. 27, no. 2, p. 259–270, jun 1998.

[63] Q. Ren, M. H. Dunham, and V. Kumar, "Semantic caching and query processing," vol. 15, no. 1, p. 192–210, jan 2003.

[64] B. T. Jónsson, M. Arinbjarnar, B. Þórsson, M. J. Franklin, and D. Srivastava, "Performance and overhead of semantic cache management," vol. 6, no. 3, p. 302–331, aug 2006.

[65] A. Keller and J. Basu, "A predicate-based caching scheme for client-server database architectures," in *Proceedings of 3rd International Conference on Parallel and Distributed Information Systems*, 1994, pp. 229–238.

[66] P. Godfrey and J. Gryz, "Answering queries by semantic caches," in *Database and Expert Systems Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 485–498.

[67] D. Lee and W. W. Chu, "Semantic caching via query matching for web sources," in *Proceedings of the Eighth International Conference on Information and Knowledge Management*, ser. CIKM '99. New York, NY, USA: Association for Computing Machinery, 1999, p. 77–85.

[68] N. H. Ryeng, J. O. Hauglid, and K. Nørvåg, "Site-autonomous distributed semantic caching," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, ser. SAC '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 1015–1021.

[69] L. d'Orazio, C. Roncancio, C. Labbé, and F. Jouanot, "Semantic caching in large scale querying systems," *Revista Colombiana de Computación*, vol. 9, 06 2008.

[70] L. d'Orazio and M. K. Traoré, "Semantic caching for pervasive grids," in *Proceedings of the 2009 International Database Engineering and Applications Symposium*, ser. IDEAS '09, 2009, p. 227–233.

[71] A. Vancea and B. Stiller, "Coopsc: A cooperative database caching architecture," in *WETICE*, 2010, pp. 223–228.

[72] J. Fang, Y. T. B. Mulder, J. Hidders, J. Lee, and H. P. Hofstee, "In-memory database acceleration on fpgas: a survey," *The VLDB Journal*, vol. 29, pp. 33–59, 2019.

[73] B. Salami, G. A. Malazgirt, O. Arcas-Abella, A. Yurdakul, and N. Sonmez, "Axledb: A novel programmable query processing platform on fpga," *Microprocessors and Microsystems*, vol. 51, pp. 142–164, 2017.

[74] D. Sidler, Z. Istvan, M. Owaida, K. Kara, and G. Alonso, "Doppiodb: A hardware accelerated database," in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017, p. 1659–1662.

[75] L. Woods, Z. István, and G. Alonso, "Ibex: An intelligent storage engine with support for advanced sql offloading," *Proc. VLDB Endow.*, vol. 7, p. 963–974, 2014.

[76] M. Owaida, D. Sidler, K. Kara, and G. Alonso, "Centaur: A framework for hybrid cpu-fpga databases," in *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines*, 2017, pp. 211–218.

[77] B. Sukhwani, H. Min, M. Thoennes, P. Dube, B. Brezzo, S. Asaad, and D. E. Dillenberger, "Database analytics: A reconfigurable-computing approach," *IEEE Micro*, vol. 34, no. 1, pp. 19–29, 2014.

[78] D. Chiou, "The microsoft catapult project," in *2017 IEEE International Symposium on Workload Characterization (IISWC)*, 2017, pp. 124–124.

[79] Amazon, "Amazon ec2." [Online]. Available: https://aws.amazon.com/ec2

[80] V. L. N. Huu, J. Lallet, E. Casseau, and L. d'Orazio, "MASCARA (modular semantic caching framework) towards FPGA acceleration for iot security monitoring," *Open Journal of Internet of Things*, vol. 6, no. 1, 2020.

[81] V. L. N. Huu, L. d'Orazio, E. Casseau, and J. Lallet, "Cache management in MASCARA-FPGA: from coalescing heuristic to replacement policy," in *DaMoN@SIGMOD*, 2022.

[82] ——, "MASCARA-FPGA cooperation model: Query trimming through accelerators," in *SSDBM*, 2021.

[83] E. Panjei, L. Gruenwald, E. Leal, C. Nguyen, and S. Silvia, "A survey on outlier explanations," *VLDBJ*, vol. 31, no. 5, 2022.

[84] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xie, "Discovering spatio-temporal causal interactions in traffic data streams," in *SIGKDD*, C. Apté, J. Ghosh, and P. Smyth, Eds., 2011.

[85] L. Xing, W. Wang, G. Xue, H. Yu, X. Chi, and W. Dai, "Discovering traffic outlier causal relationship based on anomalous DAG," in *ICSI*, 2015.

[86] X. Dang, I. Assent, R. T. Ng, A. Zimek, and E. Schubert, "Discriminative features for identifying and interpreting outliers," in *ICDE*, 2014.

[87] N. Gupta, D. Eswaran, N. Shah, L. Akoglu, and C. Faloutsos, "Beyond outlier detection: Lookout for pictorial explanation," in *ECML PKDD*, 2018.

[88] K. Viswanathan, C. Lakshminarayan, V. Talwar, C. Wang, G. Macdonald, and W. Satterfield, "Ranking anomalies in data centers," in *NOMS*, 2012.

[89] M. A. Siddiqui, A. Fern, T. G. Dietterich, R. Wright, A. Theriault, and D. W. Archer, "Feedback-guided anomaly discovery via online optimization," in *SIGKDD*, 2018.

[90] H. Zhang, Y. Diao, and A. Meliou, "Exstream: Explaining anomalies in event stream monitoring," in *EDBT*, 2017.

[91] F. Song, Y. Diao, J. Read, A. Stiegler, and A. Bifet, "EXAD: A system for explainable anomaly detection on big data traces," in *ICDM Workshops*, 2018.

[92] E. Panjei, L. Gruenwald, E. Leal, and C. Nguyen, "Micro-clusters-based outlier explanations for data streams," in *ANDEA@KDD*, 2021.