Research online Publishing
www.ronpub.com

Open Access

Open Journal of XXX (OJXX)
Volume X, Issue X, 20XX
http://www.ronpub.com/ojxx
ISSN XXXX-XXXX

Semantic caching framework, an application to FPGA-based application for IoT security monitoring

Laurent d'Orazio [1], Julien Lallet [2]

[1] Univ Rennes, CNRS, IRISA, Lannion, France, laurent.dorazio@univ-rennes1.fr
[2] Nokia Bell Labs, Lannion, France, julien.lallet@nokia-bell-labs.com

Abstract

Security monitoring is one subdomain of cybersecurity which aims to guarantee the safety of systems, continuously monitoring unusual events. The development of Internet Of Things leads to huge amounts of information, being heterogeneous and requiring to be efficiently managed. Cloud Computing provides software and hardware resources for large scale data management. However, performances for sequences of on-line queries on long term historical data may be not compatible with the emergency security monitoring. This work aims to address this problem by proposing a semantic caching framework and its application to acceleration hardware with FPGA for fast- and accurate-enough logs processing for various data stores and execution engines.

Type of Paper and Keywords

Visionary paper: *hardware acceleration, semantic caching, security monitoring*

1 Introduction

Security monitoring is one subdomain of cybersecurity. It aims to guarantee the safety of systems, continuously monitoring unusual events analyzing logs. A system in this context is very variable. It can actually be an information system in any institution or any devices, like a laptop, a smartphone, a smartwatch, a vehicle (car, plane, etc.), a television, etc. Thus, the data to be managed with a high Velocity, are Voluminous with a high Variety. Security monitoring for Internet of Things (IoT) can thus be seen as a concrete use case of Big Data [1].

Cloud computing [2] and Big Data have led to the emergence of new data management systems, as an alternative to Relational Database Management Systems (DBMS) : column-stores like Big Table [3] or systems based on Massively Parallel Processing (or MPP) [4], [5], [6], [7], especially those relying on MapReduce [8], Spark [9] or Flink [10]. However, the performance of a sequence of online queries on long term historical data may be low. For example, during a potential attack an administrator may be interested in retrieving logs for a given user. Such a process may require some minutes.

Semantic caching [11], [12], [13], [14], [15] allows to exploit resources in the cache and knowledge contained in the queries themselves. It enables effective reasoning (analysis and processing), delegating part of the computation process to the cache, reducing both data transfers and CPU load on servers. Nevertheless, reasoning on the cache content induces non-negligible overhead [16]. This thus raises the challenge on providing finely-tuned caches with appropriate analysis and processing capability with respect to access patterns (for instance supervisors with first coarse grain queries and then refining researchers), performances indicators (response

time, quality/freshness, energy consumption, etc.) and hardware acceleration opportunities.

Hardware acceleration has been proved as a relevant solution in data processing. During the last decades some operators have been implemented on Graphic Processing Units (GPUs) [17], [18] and more recently Field-Programmable Gate Arrays (FPGAs) have been used for sorting networks [19], [20]. These previous results have shown that efficient use of FPGAs thanks to proper computation model and related implementation. Nevertheless, existing work do not take into account the semantic of queries. For example, if an administrator after a first query on HTTP accesses wants to refine his research and retrieve logs related to a given user, he will have to wait again for the query to be processed on servers as for the first request, whereas the results are included in his previous research.

This paper aims to fill in the gap between few-expressive hardware acceleration and fine-grained knowledge expensive query processing with a direct application in security monitoring for IoT. On the one hand it aims at providing a framework and its implementation for query processing on FPGA so as to provide tunable caching, with on the shelf solutions, for large-scale data management environment. On the other hand, it positions existing works with respect to security supervision so as to list research challenges.

The remainder of this paper is organized as follows. In Section 2, we provide the background information that is used throughout the paper. Section 3 details our cache framework and describes our multi-layered implementation with FPGA acceleration. Finally, Section 4 concludes this paper as a list of research challenges.

2 BACKGROUND

This section introduces 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 for a platform dedicated to logs analysis. It then presents semantic caching as an optimization opportunity, defines FPGA and describes our motivation to enable efficient large-scale logs analysis via FPGA-based semantic caching.

2.1 Running Example on security monitoring

The proposal will be illustrated through a security monitoring example. In this context, administrators access dashboards to analyze logs. Figure 1 presents some examples of data, that is to say 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 instance between 0:00:00 and 1:00:00 first of July). They also may be interested in aggregation queries, for instance to spot the machine with the biggest traffic.

Security monitoring experts are directly concerned by Big Data issues. Indeed, they have to efficiently monitor various systems and their number is growing, in particular due to always more 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.

2.2 Cloud Computing and Big Data

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 [4], SCOPE [5], Hive [6] or Jaql [7]. Others consist in a column-store like Big Table [3] or Cassandra [21]. On top of these stores different execution engines can be deployed such as MapReduce [8], Tez [22], Spark [9] or Flink [10]. These tools can be seen as interesting building blocks to manage large number of logs in security monitoring.

2.3 Semantic caching

Semantic caching [11] [12] allows to exploit resources in the cache and knowledge contained in the queries themselves. As a consequence, it enables effective reasoning, delegating part of the computation process to the cache, reducing both data transfer and the load on servers.

When a query is submitted to a cache, it is split into two disjoint pieces: (1) a probe query, which retrieves the portion of the result available in the local cache, and (2) a remainder query, which retrieves any missing tuples in the answer from the server. If the remainder query exists then it is sent to the server for processing.

Example 1 *Consider a semantic cache storing an entry* $e\ date <' 07/01/1995 - 0:00:00'$ *and a posed query* $q\ ip =' 131.254.254.30'$. *In that case a probe query* $probe\ is\ date <' 07/01/1995 - 0:00:00' \wedge ip =' 131.254.254.30'$ *can be processed in the cache while a remainder query* $remainder\ date \neq' 07/01/1995 - 0:00:00' \wedge ip =' 131.254.254.30'$ *has to be sent to the server.*

```
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
```

Figure 1: Example of logs of a HTTP server

Semantic caching has been successfully used in various contexts: distributed databases [11] [12], mobile environments [13], data warehouses [14], Web [15], Peer-to-Peer systems [14], etc.

2.4 FPGA

Field-Programmable Gate Array (FPGA) is an integrated circuit designed as a programmable device. FPGAs can be configured to process specific applications, as this is the case for Application Specific Integrated Circuits (ASIC), except that FPGAs can be reprogrammed at runtime to modify the functionality of the hardware circuitry.

FPGAs are mainly composed of configurable logic blocks (CLBs) and configurable Input/outputs (IOs). The CLBs are the basic elements of the processing resources of FPGAs and are made of two basic components: flip-flops and lookup tables (LUTs). The LUTs can be seen as small memories where preprocessed results of functions are stored. The inputs of the LUT are the function variables. The exhaustive input possible sets are first processed off-line (synthesis) and stored in the LUT so that the right "address" of the LUT is selected at runtime by the LUT inputs. If needed, flip-flops placed after the LUTs will make the function synchronous. Additionally, application specific functions or memories can be added in the FPGA in order to extend and enhanced the performances. Finally, CLBs are joined through programmable interconnections and constitute the whole algorithm to compute.

The well spread use of FPGAs in data centers nowadays allows us to consider these processing resources as standard solutions for acceleration. Using FPGAs in cloud infrastructures allows to fill the gap between the processing flexibility offered by GPPs and the efficiency which can be achieved by ASIC.

2.5 Motivation

Big Data technologies are promising for large-scale log analytics. On one hand they can be used for long-term storage. On the other hand they can handle data processing on large clusters. However, they do not address fine-grained reusability. In particular, they may induce unnecessary query re-executions.

Middleware- or application-level semantic caching have been successfully used to improve query execution. A semantic cache can store results and support an interface to answer query using the cache content. Unfortunately, the expressiveness gain is the result of non-negligible overhead [16], especially due to the complexity of query rewriting [23].

Hardware acceleration with FPGA opens new research directions, with large scale logs analysis in security monitoring as an interesting use case. Indeed, caching is relevant (in terms of response time, energy consumption or quality of results) only if it is finely tuned with respect to its environment. This first makes in mandatory to provide a semantic caching framework for FPGA as well as the associated implementation. Then it requires to address challenges with respect to the infrastructure environment (rewriting and optimizing queries on FPGA) and the application (validating the solution for security monitoring).

3 FPGA-BASED SEMANTIC CACHING

The proposed platform for historical logs analysis is based on semantic caching deployed on FPGA and configured for a given data store and execution environment. The coarse grain vision is the one illustrated by figure 2. Log data are stored for a long period, generating large volume of information on a Big Data storage system deployed over commodity hardware. This data store can be a distributed file system-based solution (for instance Hive), a column-store like HBase or Cassandra, a key-value store like MongoDB, etc. This level thus consists in large cold data with low response time.

Logs analysis may rely on different execution environments like Dryad, MapReduce, Tez or Spark. After logs processing, accessed data will be copied on high-performant, yet limited, hardware namely FPGA, used as a caching-level enabling efficient processing of hot data.

The proposal consists in two main contributions; (1)

a cache framework to provide adaptability for not only various kinds of storage or processing solutions but also to define appropriate caching strategies in terms of management, replacement or resolution; and (2) multi-layered implementation with FPGA acceleration consisting of operators for query processing (select, join, etc.).

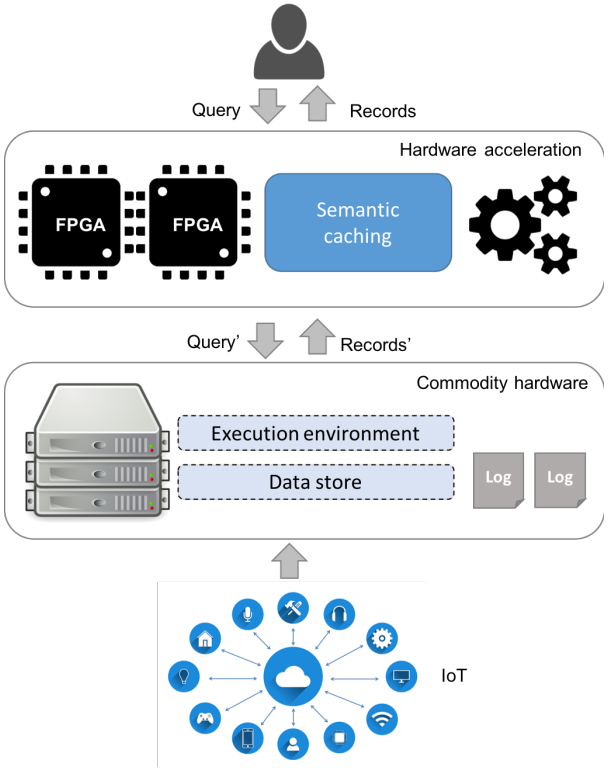

Figure 2: Overview of the architecture

3.1 Cache framework

The main objective of the proposed cache framework is to make available an abstract cache that will then be finely tuned to match the requirements of the considered application. Especially, it will enable to instantiate semantic caches on a distributed environment for various kinds of systems.

The proposed cache framework is based on the Separation of Concerns (SoC) design principle [24]. It consists of three components, Cache Manager, Replacement Manager and Resolution Manager, for the main features to take into account which are: (1) cache management, (2) replacement and (3) resolution.

3.1.1 Cache Management

Cache management mainly consists in two aspects: addressing and searching. Addressing can be identity-based, when a unique identifier is associated to a cache item (for example for page caching). Addressing can also introduce some knowledge, when a collection of elements can be accessed through a query like in a semantic cache. Searching is in charge of looking for a given object in the cache. Since this is a frequent operation it has to be efficient. Different searching schemes can be adopted [25]: direct search, sorted table, hash table, etc.

3.1.2 Replacement

Replacement is necessary when items have to be added in the cache while the amount of storage allocated to it is full. A policy is then used to determine elements to evict. Many replacement policies exist: random, Last Recently Used (LRU), First In First Out (FIFO), Least Frequently Used (LFU), etc. Choosing a policy mainly depends on the access patterns of the considered application.

3.1.3 Resolution

The resolution protocol [26] consists in the process to be executed to retrieve missing elements. It first manages communications with the server side. As a consequence, different protocols enable to consider various servers/big data management systems.

The resolution protocol can also be used to leverage the load on sibling or father caches. Both of them try to look for the requested elements in their own content. The main difference between them is that a father cache must provide the element, even it does not possess it (leading to another resolution), while a sibling cache does not have to.

In addition, different resolution protocols can be used for sibling caches, for example with flooding like in the Internet Cache Protocol (ICP) [27] or with a catalog [28].

3.2 Multi-layered implementation with FPGA acceleration

The foreseen cache framework will have to support the implementation of multi-layered application. The first layer, which will be the front-end interface for queries, will be a standard software layer. Depending on the status of the cache management, it will have the possibility to answer the queries as done in legacy systems, or, if possible, benefit of the acceleration processing offered by the second layer. The latter is the FPGA layer and will provide efficient hardware acceleration for the software layer. In this prospective view, software layers may run in virtual machines or containers according to current function implementation in data centers. The multi-layered architecture is described figure 3. On the right side of the picture, a server is used to host the software

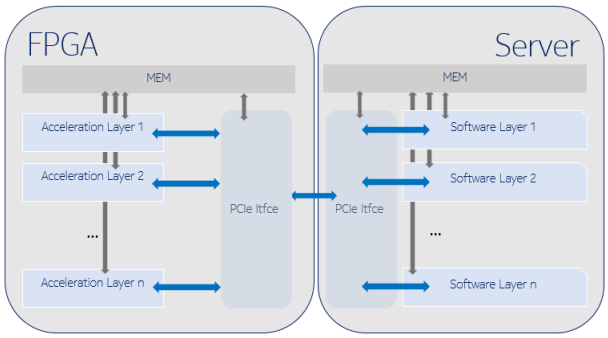

Figure 3: Two layers server infrastructure

layer, while, on the left side, a FPGA board, connected through PCIe interface will host the second layer. To ensure the efficiency of implemented algorithms, fast and low latency communications link will be proposed inspired from high frequency trading FPGA infrastructures [29]. Furthermore, accelerators sharing solutions will be designed to enhanced the flexible and parallel use of the available FPGA resources among queries.

4 Research Challenges

This paper presents a FPGA-based Semantic Caching for big logs analysis. It has first introduced an overview of the considered architecture. It has then described a proposal of cache framework so as to instantiate finely tuned cache instances. After that it has discussed a multi-layered implementation with FPGA acceleration.

The proposed context covers domains such as database management, parallel and distributed systems, High-Performances Computing (HPC), security or Human-Computer Interaction (HCI). As a consequence, it consists in a relevant use case for many research directions. By way of conclusion, this section lists some of them from a data management and parallel systems perspective.

4.1 Query rewriting

Query rewriting has been extensively studied during decades in the data base community (interesting readers can refer to this representative survey [23]). However, big data and client caching make it possible to tackle the problem from a new perspective [30]. Indeed, query processing may be so expensive that overhead due to NP-complete problem such as satisfiability may be beneficial. In this context, it is necessary to provide a formal framework for semantic caching considering Select Project Join (SPJ) queries with aggregation.

4.2 Multi-objective query processing

Multi-objective query processing has recently gained increasing attention [31], [32], [33], [34] especially due to the tradeoff between efficiency and expensiveness of scalable processing in a pay-as-you-go environment. FPGA-semantic caching makes it possible to extend the research problem increasing heterogeneity and adding quality as another objective, when response time may be reduced scarifying quality (consistency) of results.

4.3 Benchmarking

Performance analysis is a well-known domain in databases [35], [36], [37], [38]. Several benchmarks have been developed, especially with corporation such as TPC providing benchmarks for OLTP, integration or Big Data systems[1]. Unfortunately, these benchmarks do not take into account the specificities of cybersecurity, being in terms of data and queries. For example some intrusions may be subtil, with sparse events on a long period of time. This makes it necessary to extend current solutions to take into account these behaviors and provide data sets and queries. This will thus provide a reproductible environment to compare solutions in this field.

Acknowledgements

We would like to thank members of SHAMAN team at IRISA, Nokia Bell Labs and VLIOT reviewers for insightful comments.

References

[1] D. Abadi, R. Agrawal, A. Ailamaki, M. Balazinska, P. A. Bernstein, M. J. Carey, S. Chaudhuri, J. Dean, A. Doan, M. J. Franklin, J. Gehrke, L. M. Haas, A. Y. Halevy, J. M. Hellerstein, Y. E. Ioannidis, H. V. Jagadish, D. Kossmann, S. Madden, S. Mehrotra, T. Milo, J. F. Naughton, R. Ramakrishnan, V. Markl, C. Olston, B. C. Ooi, C. Ré, D. Suciu, M. Stonebraker, T. Walter, and J. Widom, "The Beckman report on database research," *Communications of the ACM*, vol. 59, no. 2, pp. 92–99, 2016.

[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, pp. 50–58, 2010.

[1] www.tpc.org

[3] 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 TOCS*, vol. 26, no. 2, pp. 4:1–4:26, 2008.

[4] 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, pp. 1099–1110.

[5] 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, pp. 1265–1276, 2008.

[6] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Anthony, H. Liu, and R. Murthy, "Hive - a petabyte scale data warehouse using Hadoop," in *ICDE*, Long Beach, California, USA, 2010, pp. 996–1005.

[7] 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, pp. 1272–1283, 2011.

[8] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[9] 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 *SIGMOD*, Melbourne, Victoria, Australia, 2015, pp. 1383–1394.

[10] 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, pp. 28–38, 2015.

[11] A. M. Keller and J. Basu, "A Predicate-based Caching Scheme for Client-Server Database Architectures," *VLDBJ*, vol. 5, no. 1, pp. 35–47, 1996.

[12] S. Dar, M. J. Franklin, B. b. Jónsson, D. Srivastava, and M. Tan, "Semantic Data Caching and Replacement," in *VLDB*, Mumbai (Bombay), India, 1996, pp. 330–341.

[13] K. C. K. Lee, H. V. Leong, and A. Si, "Semantic query caching in a mobile environment," *Mobile Computing and Communications Review*, vol. 3, no. 2, pp. 28–36, 1999.

[14] P. Kalnis, W. S. Ng, B. C. Ooi, D. Papadias, and K.-L. Tan, "An adaptive peer-to-peer network for distributed caching of OLAP results," in *SIGMOD*, Madison, Wisconsin, USA, 2002, pp. 25–36.

[15] B. Chidlovskii and U. M. Borghoff, "Semantic Caching of Web Queries," *VLDBJ*, vol. 9, no. 1, pp. 2–17, 2000.

[16] B. b. Jónsson, M. Arinbjarnar, B. \THórsson, M. J. Franklin, and D. Srivastava, "Performance and overhead of semantic cache management," *ACM TIT*, vol. 6, no. 3, pp. 302–331, 2006.

[17] G. Teodoro, E. Valle, N. Mariano, R. da Silva Torres, W. M. Jr., and J. H. Saltz, "Approximate similarity search for online multimedia services on distributed CPU-GPU platforms," *VLDBJ*, vol. 23, no. 3, pp. 427–448, 2014.

[18] T. Karnagel, R. Müller, and G. M. Lohman, "Optimizing GPU-accelerated Group-By and Aggregation," in *ADMS@VLDB*, Kohala Coast, Hawaii, USA, 2015, pp. 13–24.

[19] R. Müller, J. Teubner, and G. Alonso, "Sorting networks on FPGAs," *VLDBJ*, vol. 21, no. 1, pp. 1–23, 2012.

[20] L. Woods, Z. István, and G. Alonso, "Ibex - an intelligent storage engine with support for advanced {sql} off-loading," *PVLDB*, vol. 7, no. 11, pp. 963–974, 2014.

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

[22] 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, pp. 1357–1369.

[23] A. Y. Halevy, "Answering queries using views: A survey," *VLDBJ*, vol. 10, no. 4, pp. 270–294, 2001.

[24] E. W. Dijkstra, *Selected Writings on Computing: A Personal Perspective*. Springer-Verlag, 1982.

[25] W. Effelsberg and T. Härder, "Principles of Database Buffer Management," *ACM TODS*, vol. 9, no. 4, pp. 560–595, 1984.

[26] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, "A Hierarchical Internet Object Cache," in *Proceedings of the USENIX Technical Conference*, San Diego, California, USA, 1996, pp. 153–164.

[27] D. Wessels and K. C. Claffy, "Icp and the squid web cache," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 3, pp. 345–357, 1998.

[28] L. Fan, P. Cao, J. M. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache

sharing protocol," *IEEE/ACM TON*, vol. 8, no. 3, pp. 281–293, 2000.

[29] C. Leber, B. Geib, and H. Litz, "High frequency trading acceleration using fpgas," in *International Conference on Field Programmable Logic and Applications*, 2011.

[30] I. Elghandour and A. Aboulnaga, "ReStore: reusing results of MapReduce jobs in pig," in *SIGMOD*, Scottsdale, AZ, USA, 2012, pp. 701–704.

[31] I. Trummer and C. Koch, "Multi-objective parametric query optimization," *VLDBJ*, vol. 26, no. 1, pp. 107–124, 2017.

[32] ——, "A Fast Randomized Algorithm for Multi-Objective Query Optimization," in *SIGMOD*, San Francisco, CA, USA, 2016, pp. 1737–1752.

[33] ——, "An Incremental Anytime Algorithm for Multi-Objective Query Optimization," in *SIGMOD*, Melbourne, Victoria, Australia, 2015, pp. 1941–1953.

[34] H. Kllapi, E. Sitaridi, M. M. Tsangaris, and Y. E. Ioannidis, "Schedule optimization for data processing flows on the cloud," in *SIGMOD*, Athens, Greece, 2011, pp. 289–300.

[35] C. Düntgen, T. Behr, and R. H. Güting, "BerlinMOD: a benchmark for moving object databases," *VLDBJ*, vol. 18, no. 6, pp. 1335–1368, 2009.

[36] O. Erling, A. Averbuch, J.-L. Larriba-Pey, H. Chafi, A. Gubichev, A. Prat-Pérez, M.-D. Pham, and P. A. Boncz, "The LDBC Social Network Benchmark: Interactive Workload," in *SIGMOD*, Melbourne, Victoria, Australia, 2015, pp. 619–630.

[37] L. Gu, M. Zhou, Z. Zhang, M.-C. Shan, A. Zhou, and M. Winslett, "Chronos: An elastic parallel framework for stream benchmark generation and simulation," in *ICDE*, Seoul, South Korea, 2015, pp. 101–112.

[38] R. Ulbricht, C. Hartmann, M. Hahmann, H. Donker, and W. Lehner, "Web-based Benchmarks for Forecasting Systems: The ECAST Platform," in *SIGMOD*, San Francisco, CA, USA, 2016, pp. 2169–2172.

AUTHOR BIOGRAPHIES

Laurent d'Orazio has been a Professor at Univ Rennes, CNRS, IRISA since 2016. He received his PhD degree in computer science from Grenoble National Polytechnic Institute in 2007. He was a Associate Professor at Blaise Pascal University and LIMOS CNRS, Clermont-Ferrand from 2008 to 2016. His research interests include (big) data algorithms and architectures, distributed and parallel databases. He has published papers in Information Systems, Sigmod Record, Concurrency and Computation Practice and Experience. He served in Program Committees in BPM, workshops affiliated to VLDB, EDBT, etc. and Reviewing Commitees in Transactions on Parallel and Distributed Systems, Concurrency and Computation: Practice and Experience. He is or has been involved (sometimes as a coordinator) in research projects such as the NSF MOCCAD project (since 2013), the ANR SYSEO project (797 000 euros funding, 2010-2015) and the STIC ASIA GOD project (30 000 euros funding, 2013-2015).

Julien Lallet joined the company Alcatel-Lucent in 2011 and is currently a research engineer at Nokia Bell-Labs since 2016. He received his PhD degree in electrical engineering from the University of Rennes in 2008. He was a Post-doctoral fellow at the University of Bielefeld in Germany from 2009 to 2010. His research interests include efficient processing in the context of cloud computing and hardware acceleration on FPGA. He has published papers in computing architectures and FPGA systems.