Title: ENTRADA: a High-Performance Network Traffic Data Streaming Warehouse

Authors: Maarten Wullink, Giovane C. M. Moura, Moritz Müller, and Cristian Hesselman

Venue: IEEE/IFIP Network Operations and Management Symposium 2016 (NOMS 2016), Istanbul, Turkey

Track: Experience Session

Conference dates: April 25th to 29th, 2016.

Citation:

- Wullink, M., Moura, G. C. M., Muller, M, and Hesselman, C. "ENTRADA: a High-Performance Network Traffic Data Streaming Warehouse". In: IEEE/IFIP Network Operations and Management Symposium (NOMS 2016) – Experience Session. Istanbul, Turkey, May 2016 (to appear)
- Bibtex:

```
@inproceedings{noms-2016,
author = {{Maarten Wullink, Giovane C. M. Moura,
Muller, M, and Cristian Hesselman}},
booktitle={{Network Operations and Management
Symposium (NOMS), 2016 IEEE (to appear)}},
title={{ENTRADA: a High Performance Network
Traffic Data Streaming Warehouse}},
year={2016},
month={April},
}
```

ENTRADA: a High-Performance Network Traffic Data Streaming Warehouse

Maarten Wullink, Giovane C. M. Moura, Moritz Müller, and Cristian Hesselman
SIDN Labs
Stichting Internet Domeinregistratie Nederland (SIDN)
Arnhem, The Netherlands
Email: {firstname.lastname}@sidn.nl

***Abstract*—We present ENTRADA, a high-performance data streaming warehouse that enables researchers and operators to analyze vast amounts of network traffic and measurement data within interactive response times (seconds to few minutes), even in a small computer cluster. ENTRADA delivers such performance by employing a optimized file format and a high-performance query engine, both open-source. ENTRADA has been operational for more than 1.5 years, having ingested more than 100 TB of `pcap` files from two .nl DNS authoritative servers. As we discuss, we use this data in projects that aim at further increasing the security and stability of the .nl zone. We present in this paper our design choices, experiences, and a performance evaluation of ENTRADA. Finally, we open-source ENTRADA, which can be used "out-of-the-box" by researchers, operators, and registries to deploy their own networking analysis clusters for DNS traffic, and can be easily extended to handle any other structured data.**

I. Introduction

To cope with the increasing growth in the volume of Internet traffic, researchers have resorted to computer cluster-based solutions as a way to achieve better performance, scalability, and dependability when processing increasingly large datasets [1], [2], [3], [4], [5]. Such cluster are often based on Apache Hadoop [6] or other non-relational databases (NoSQL) [7].

Differently from traditional "only once" analysis on snapshot datasets, in this paper we focus on high-performance cluster solutions that (i) are designed to ingest and process continuous *streams* of network data and (ii) combine long term data storage, delivering *interactive* response times (seconds or few minutes) on both historical ($\geq$ years) and recent data. Such solutions are referred in the literature as *data streaming warehouses* (DSW) [1], [8], [2].

Building such DSWs is far from being an trivial task, and poor design and data management/engineering choices may lead to significantly poorer performance with the same hardware set. In this paper, we cover our experience in building ENTRADA (ENhanced Top-level domain Resilience through Advanced Data Analysis), a Hadoop-based DSW for network traffic analysis. ENTRADA was built to enable fast data analysis on the network traffic to the .nl (The Netherlands) country -code top-level domain (ccTLD) authoritative DNS servers, with the main goal of supporting applications that improve both security and stability of the .nl zone. ENTRADA is built entire with open-source tools, and we make it openly available at [9]. Moreover, ENTRADA can be extended to any file format/protocol (e.g., log files, passive and active measurement data), or any other structure data.

ENTRADA is a enabler of high-performance analysis of networking traffic data: it can be used to quickly determine statistics over large data sets, and enable quick perform hypothesis tests. As we show, it is capable to analyze the equivalent of 52TB of `pcap` data in less than 3.5 minutes, even on a small 6-nodes cluster. This level of performance would be hard to achieve using traditional networking data analysis tools. ENTRADA has been uninterruptedly operational for 1.5 years (December 2015), with more than 88 billion records (DNS query/response pairs).

In a nutshell, to achieve high-performance in networking data analysis in DSWs (where the same data is queried and analyzed multiple times), we demonstrate the benefits of (i) converting standard `pcap` files to query-optimized format – Apache Parquet [10] – and using (ii) Impala [11] as query engine, instead of standard MapReduce jobs or other engines. For more complex queries or analysis not supported by Impala, other engines, such as Apache Spark [12], can be easily used on the same data in Parquet format.

The remainder of this paper is divided as follows: in Section II we present background information on the measurement data we store in ENTRADA. Then, in Section III, we lay out our requirements and present in Section IV an assessment of the design choices with regards to these requirements. Data engineering and management plays a key role in performance, which we cover in Section V. We evaluate the performance of ENTRADA using datasets from one of the .nl authoritative servers in Section VI. Finally, related work and conclusions are covered in Sections VII and VIII, respectively.

II. Background

In order to clarify why we need such DSW, we first have to explain our business. *Stichting Internet Domeinregistratie Nederland* (SIDN) [13] is the registry for .nl, which is the country code top-level domain (ccTLD) of the Netherlands. As part of this registry role, SIDN manages the authoritative DNS servers for .nl as well as the database of all 5.5 million currently registered .nl domain names. At SIDN Labs, the research arm of SIDN, we built ENTRADA as an enabling

978-1-5090-0223-8/16/$31.00 © 2016 IEEE

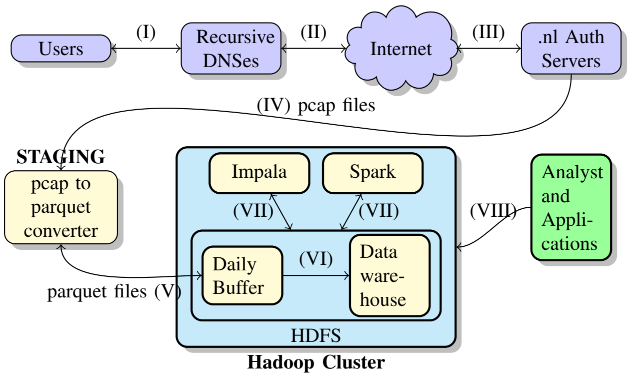

Fig. 1. ENTRADA data sequence flow

platform for applications that aim at further increasing the security and stability of .nl zone.

Authoritative DNS data and domain resolution: currently, we store in ENTRADA passive measurement data collected at two of the .nl authoritative servers, i.e., servers that are responsible for the .nl zone.

To understand this data, we have to understand the domain resolution process, which we simplify here in an example – we refer the reader to [14] for a more detailed explanation. Consider the case in which a user tries to connect to a website (e.g.: www.example.nl), the user's computer needs to resolve the domain name into ultimately an IP address. The user's computer runs small stub DNS resolver, which, connects to a recursive DNS servers (I in Figure 1), usually the ISPs' DNS recursive server. This server, in turn, will start a recursive process [14] on behalf of the user, asking the root servers (the "." zone, II in Fig. 1, not shown) for the addresses of the .nl TLD authoritative servers.

The recursive DNS server will then ask one of the .nl servers for `example.nl` (III in the same figure). The .nl name server will refer the recursive server to the name server of example.nl, which knows the IP address of `example.nl`and returns it to the recursive server. The recursive server will ultimately send the IP address to the user, whose browser will then be able to reach `example.nl`.

Partial view: to improve performance, recursive servers employ local caches that store queried domains responses [15]. As a consequence, cache hits in the recursive DNS server avoid it from issuing queries to the .nl authoritative servers. Therefore, the requests that reach a DNS authoritative server are only a *subset* of all the queries for the zone the name server is authoritative. This subset of queries is the data we store in ENTRADA (as well as their respective IP and UDP/TCP headers, and other metadadata – see Sec. V), from two authoritative servers. We will add data from other servers in the near future.

III. ENTRADA Requirements

We defined the following requirements for ENTRADA:

Performance: the system should be able to support interactive data exploration with short response times (seconds or few minutes), even over large time windows ($\geq$ years).

Scalability: we started by storing all DNS queries from one .nl server ($\sim$ 85GB/day in `pcap` format, per authoritative server). ENTRADA must scale, i.e., support both storage while not compromising the performance, as data from more .nl servers is stored.

Usability: users should be able to access and query data without the need to use complex APIs or writing low-level code, enabling fast data exploration and prototyping. We determine that SQL-like query support is a must, enabling data analysts and engineers to easily explore the data [16].

Extensibility: the data format should be compatible with various data analytics engines. Therefore, we should avoid any form of vendor and/or software lock-in.

Security: data flows from and to the ENTRADA should always be encrypted, by default, with strict access-control mechanisms in place.

Dependability: Failure of any cluster component must not result in the loss of data or the cluster becoming unavailable. Performance degradation caused by component outage is acceptable.

Privacy: We store passive DNS data from two .nl authoritative name servers. Under Dutch law, parts of this data may qualify as personal identifiable information, even though most of them come from recursive servers at ISPs and not end users [14]. We do not cover this in this paper, but we have developed, together with our legal department, a publicly available data privacy framework [17] that conforms to both EU and Dutch laws. This framework has been implemented, including a privacy board that oversees SIDN Labs research.

IV. Cluster Design Choices

In this section, we discuss and motivate our design choices.

A. Data Query Engines

Whenever building a DSW, an engineer is faced with multiple choices for query engines, i.e., the software that processes the data. The performance of such engines may vary significantly according to use cases. For example, in one study [11], Impala outperformed by 6.7 times other Hadoop-based SQL-engines, such as SparkSQL, Presto, and Hive.

For ENTRADA, we have compared qualitatively the following SQL and NoSQL query engines against our requirements available when we started this project: Apache HBase[1], Elasticsearch[2], MongoDB[3], PostgreSQL[4], Hadoop HDFS+Map/Reduce [18] and Impala combined with Apache Parquet/HDFS, the more popular choices when we started this project in 2013. Since then other engines, such as Spark [12], have been developed and/or improved (ENTRADA also supports Spark).

[1] https://hbase.apache.org/
[2] https://www.elastic.co/
[3] https://www.mongodb.org/
[4] http://www.postgresql.org/

Engine	Usab.	Exten.	Perf.	Scal.	Dep.
HBase(HDFS)	0	0	1	1	0
Elasticsearch	0	0	1	1	1
MongoDB	0	0	1	1	1
Hadoop+MapReduce(HDFS)	1	0	0	1	1
PostgreSQL	1	0	0	1	0
Impala+Parquet(HDFS)	1	1	1	1	1

TABLE I
COMPARISON OF DATA QUERY ENGINES (1 = MATCHES OUR REQUIREMENTS, 0 DOES NOT MATCH)

Table I shows our ranks for each of these engines with regards to our requirements. We omit security and privacy from the table, since privacy was handled separately, and all of them provided the security features we needed.

We ruled out HBase, MongoDB and Elasticsearch from the very beginning since they do not provide SQL-compatible interface, our usability requirement. HBase since then has included a SQL interface.

Nevertheless, we tested all the engines/frameworks on a dataset of 1 billion records (1 billion DNS packets), and only Hadoop/MapReduce and PostgreSQL did not perform as we expected for our sample queries. MapReduce uses a slow batch oriented model, which incurs additional latency. For example, a simple count of DNS packets from a certain country would take roughly 1 minute to set up the batch process, plus the complexity required of writing MapReduce code, which impairs interactive analysis. PostgreSQL performance, on the other hand, decreased significantly after adding hundreds of millions of rows. We did not try to optimize it at this point, and instead we eliminated these two engines from our test, moving to our last option: Impala in combination with Parquet (HDFS) (we started with Impala around its first releases, even before an performance comparison was made available in [11]).

Impala is massively-parallel SQL query engine that utilizes the underlying Hadoop infrastructure, including HDFS. Impala implements its own services and daemons on each data node in the cluster to improve performance, and therefore does not run any MapReduce jobs on the cluster. It supports Apache Parquet file format, and both in combination enable interactive queries over very large datasets.

By performing a series of data management optimizations, Impala outperforms significantly other query engines [11], such as SparkSQL[5] (SQL module for Spark) and Presto. We therefore chose the combination of Impala and Apache Parquet for ENTRADA since it matched our requirements (Section III): it performs well [11], provides a SQL-engine that allow fast exploratory analysis with little overhead in comparison to MapReduce, and the data format it uses (Parquet) can also be used by other engines, such as Spark.

We developed the workflow showed in Figure 1: we convert the incoming `pcap` files to Parquet format (V in Fig. 1), store them on the HDFS (VI) and query using Impala (or any Parquet-compatible engines, such as Spark, VII in the same figure). We cover the data conversion and management in Section V.

[5]https://spark.apache.org/sql/

B. Cluster Architecture

We use Cloudera Hadoop Distribution for ENTRADA, since it supports Impala and has user-friendly management interfaces. As any other Hadoop Linux distribution, it provides the basic Hadoop functionalities, which includes HDFS [19], the default distributed file system that manages the data and distributes it across the data nodes. We employ triple data redundancy – data replicated on three different data nodes – to improve data dependability and performance.

We dimensioned ENTRADA to be able to store 2 years of data from one .nl authoritative server, and more nodes can be added on demand. ENTRADA consists initially of 6 nodes: one Hadoop name node, whose function is not to store data but to keep "the directory tree of all files in the file system, and tracks where across the cluster the file data is kept[6]". It is a central part of the HDFS that is used to located data stored in the cluster. The name node has a Intel Xeon 1.9 GHz 6-core processor, with 96GB of RAM and two 1Gbps line, plus 3 TB of storage in SAS. The data node is also used as the management and staging node (Fig. 1).

We have 4 data nodes, i.e., nodes in which the data is actually stored across the HDFS. They have the same specification as the name node, except for having 6TB of disk. Finally, we have a metadata node, which is responsible to store the configuration parameters of the cluster (PostgreSQL database), as well as Impala's metadata storage. This server has a 6-core 2GHz Xeon processor, with 32GB of RAM, 1Gbps line, and 4TB storage. Finally, we have just acquired two new data nodes, but they we were not active at the writing of this paper.

V. DATA MANAGEMENT

In this section we show how ENTRADA ingests the `pcap` files from the DNS authoritative servers and convert this data to a a high-performance format.

A. Data Format

Networking traffic in `pcap` format is not optimized for interactive aggregation queries For example, counting unique IPs addresses in a 100TB `pcap` file would require to read and parse all 100TB data – a very CPU/IO intensive task.

To deal with this problem over other types of datasets, Google developed Dremel [16], a query system for analysis of read-only nested data that delivers aggregation queries (e.g., averages) of over trillion-row tables in seconds. Dremel combines multi-level execution trees and columnar data storage [20]. Columnar storage [20] is an efficient format for data storage, which differs from traditional row-based storage (such as `pcap`) in the way the data elements are ordered. In row-based storage, the individual columns for a row are written sequentially to file. With columnar storage, in turn, all values for a column are written sequentially to file. This has several advantages: it reduces data latency for queries that only need

[6]https://wiki.apache.org/hadoop/NameNode

specific columns – it only reads data related to the requested columns. In the aforementioned example, instead of reading every single packet/field, it would only read the data related to source IP addresses.

Interactive response times are a must for any DSW. Therefore, we convert all incoming `pcap` files to more query-efficient formats. In our case, we convert it to Apache Parquet [11] format, which is based on Dremel. Besides enabling fast aggregation query response times, Parquet employs encoding algorithms such as run length, dictionary and delta encoding on entire columns, since they have same-type values, reducing storage requirements. By default, Impala uses Snappy[7], a compression algorithm that is a good compromise between compression and decompression speed. From the ∼85 GB of daily `pcap` data per authoritative server, Parquet and Snappy compress it to ∼6GB (after also filtering some fields, as we discuss in the next session).

B. *Data Pre-processing and Partitioning*

As we show in Fig. 1 (step IV), we first obtain `pcap` files that contain traffic to and from the .nl authoritative servers. This data, in turn, must be converted the Parquet format, so Impala and other data query engines can easily and efficiently read it.

To achieve this, we have developed a Parquet converter that is based on RIPE's NCC Hadoop `pcap` reader [21]. This converter reads incoming `pcap` files and decodes the IP, TCP, UDP, DNS and ICMP packets. For DNS packets, we perform three steps: (i) matching, (ii) filtering, and (iii) enrichment, which we describe next.

First of all, we (i) match every DNS request to its respective response[8], and store both of them as a single record, avoiding this to be done in the analyst phase. After that, we store the IP, UDP or TCP, and DNS header information, along with the DNS tuple. We (ii) filter the data, by removing the answer part of the response packet, i.e., storing the header but removing the answer sections of the DNS format. We remove this since we already store this information in another database that contains authoritative data for the .nl zone file (a list of all fields can be found at [22]).

Then, for data enrichment (iii), we add the country code of the resolver IP address to the data (using Maxmind's GeoIP database), as well as its respective autonomous system number (ASN).

We partition the data by year, month, day and authoritative server. This allows query engines to perform partition pruning, i.e., avoiding look ups in partitions that do not match the requested queries, improving performance on query response times. Finally, we write it to the Parquet format (see next subsection) using the KiteSDK library[9], in a tree structure according to the data partitions we previously defined.

[7]https://github.com/google/snappy

[8]There is an exception: rate-limited requests are not responded, leaving the response fields empty – which we also store.

[9]http://kitesdk.org/docs/current/

C. *Data Storage on HDFS*

The generated Parquet files are uploaded from the staging server to a "daily buffer" directory on the HDFS (V in Fig. 1). To reduce the number of Parquet files on the HDFS, we store all files from the "current" day in this directory, and later merge into larger files and persist into the data warehouse directory. We do this to improve lookup times, since it is preferable to have fewer larger files than many small ones.

The staging server receives `pcap` files from the .nl TLD servers every five minutes. Currently, it takes less than a minute to convert a `pcap` file to Parquet format and upload it to the daily buffer. After that, the data is automatically made available to Impala (or any other tool Parquet-compatible).

D. *Data Analytics*

A number of interfaces can be used to analyze the stored data. For interactive ad-hoc querying the user-friendly Hue[10] web-based interface is good for initial data analysis, since it allow SQL queries to be directly executed from the browser. Integration with other tooling or applications can easily be achieved with the use of Impala-shell (command line query interface) or with the Impyla Python library[11], as well as other engines, such as Spark [12].

E. *Data Conversion Monitoring*

To guarantee the data conversion process occurs without issues, we collect various metrics during the conversion (e.g.: time series of number of imported packets, DNS opcodes, etc.), and export them. Then, we persist this data in Graphite[12], a highly scalable real-time graphing system. We then use Grafana[13] to create real-time dashboards with the metric data stored in Graphite. These dashboards show the current state and historical state of the conversion module. They are also used to give an overview of the data that ENTRADA has processed.

VI. Performance Evaluation

A. *Setup and Datasets*

In the last 1.5 years, ENTRADA has accumulated 88 billion records – each record corresponding to a DNS request/response. This is equivalent to ∼100 TB in `pcap` format. After the data pre-processing described in Section V-B, the volume is reduced to ∼3.8 TB. We store the data on HDFS with a replication factor of 3, the total storage requirements are therefore 11.4 TB (December 2015).

For the performance evaluation, we use parts of this datasaset. We divided it into three time windows (day, month, and year), from one name server only, in order to assess the scalability of ENTRADA. In Parquet format, a day of DNS traffic averages 6 GB of data, a month to 240 GB, and a year 2.2 TB.

[10]http://gethue.com/category/impala/

[11]https://github.com/cloudera/impyla

[12]http://graphite.wikidot.com/

[13]http://grafana.org/

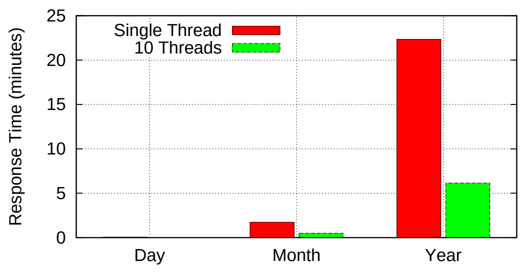

Fig. 2. Parallel query – response time

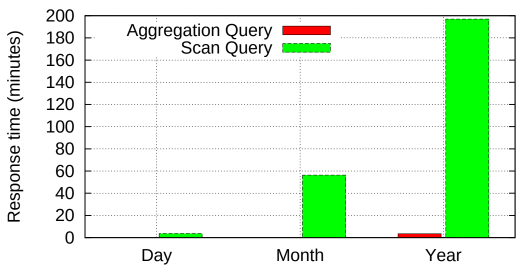

Fig. 3. Response times for aggregation and scan queries

We ran our queries from our Hadoop name node (Section IV-B), using the `impala-shell` client. The four data nodes are located in two different locations ($<$ 100km apart, two per data center), connected via 2Gbps lines. The management node is located on a third datacenter, also with the same line speed.

The name node connects to the data nodes and orchestrates the query execution, retrieving and aggregating the results. We also disabled caching at Impala (cold cache) for this evaluation.

B. Parallelizing Queries

Before measuring ENTRADA's performance, we first asses the impact of the both dataset size and number of parallel threads in the execution of an aggregation query. Since each Impala aggregation query is mapped to a single-threaded process on the data nodes, by sending only one query, we underutilize the 6 available cores on each data done. Therefore, to avoid this, we ran 10 parallel threads.

To accomplish this, we first have to split a query according to the partitions we have chosen. For example, a query that encompasses a time window of one year can be broken down in 365 queries that cover 1 day only, and later aggregated.

Figure 2 shows the response time for this cases for our test query: `select concat_ws('-',day,month,year), count(1) from dns.queries where ipv=4 and year = 'X' and month = 'Y' and day='Z' group by concat_ws('-',day,month,year).`

This query counts the number of IPv4 request/responses per day. We can observe that the response time grows linearly with the time window, and 10 threads in parallel ran 3.47 and 3.63 times faster than only one, respectively. Overall, for the one year 10-thread query, we were able to count the number of IPv4 queries for 2.2 TB of data (equivalent to 52TB in `pcap`) in less than 7 minutes, and less than a minute for a month period. This would have been unfeasible had the files been in `pcap` format using the same hardware.

C. Evaluation Queries and Results

We measure ENTRADA's response time to queries of two types: aggregation and scan. Aggregation type queries perform an operation of a number of rows, and return a single number. For example, the average packet size. A scan type returns a large number of records, e.g.: selecting all unique IP addresses from a single day.

To achieve better performance, we employ 10 parallel threads, as discussed in the previous section. We used the following queries:

- Aggregation query: `SELECT avg(reqSize), avg(respSize), sum(reqSize), sum(respSize) FROM dns.queries WHERE ipv = 6 and year = 'X' and month = 'Y' and day='Z'`
- Scan query: `SELECT ipv from dns.queries WHERE ipv = 6 AND year = 'X' and month = 'Y' and day='Z'`

The aggregation query calculates both the average and the sum of DNS requests and responses using IPv6. In the scan query, on the other hand, we retrieve all IP version column values from only IPv6 packets. Notice that we explicitly specify year, day, and month, to allow Impala to perform partition pruning (Section V-B), avoiding unnecessary IO operations.

We ran the aggregation query above specified for one year period – or 2.2 TB of parquet data (52 TB of `pcap`). It took *less than 3.5 minutes* to retrieve the results, and *16.1 seconds* for a month of data (240GB in Parquet), as can be seen in Figure 3. Combined with the Parquet format and partition pruning, Impala is able to query only the necessary files and delivers interactive-speed results, even in such a small cluster. The response time of such queries can be further reduced by simply adding more nodes to the cluster.

Even though ENTRADA delivers better performance with aggregation queries – DSWs and related solutions are more frequently queried using aggregation type queries [16] – some scenarios demand scan queries, in which large volumes of data are pulled from ENTRADA. Figure 3 shows also the results for the scan query. As can be seen, it did not enable interactive analysis. Regardless, even for the scan data case, we can see that ENTRADA performs very well when retrieving all this data – and can be easily improved by adding more nodes.

D. ENTRADA Applications

We use ENTRADA to enable a series of applications. Due to space constrains, we briefly cover three of them.

The Resolver Reputation (ResRep)[14] is used to single out IP addresses of DNS resolvers that query .nl authoritative servers with a suspicious behavior (e.g.: high percentage of invalid MX queries), which may be indicator of botnet activity. IP

[14]https://www.sidnlabs.nl/downloads/presentations/A_Resolver_Reputation_System.pdf

addresses geo-located to the Netherlands are then submitted as a feed to AbuseHUB[15], a Dutch joint initiative formed by major ISPs to share incident data in order to mitigate botnets, so their respective ISPs can take action based on our data.

We also use ENTRADA to detect maliciously registered domains, in a project called new Domains Early-Warning System (nDEWS). By analyzing their initial query pattern[16], we single out suspicious domains that may be involved in phishing, spam, fraud, and malware distribution, among others. We then notify their respective registrars (currently in a pilot phase), so they can act on it.

Finally, as part of our open data initiative, we use ENTRADA to make DNS datasets openly available on our statistics website [23], in addition to a series of analysis on .nl zone and traffic.

VII. Related Work

To the best of our knowledge, this is the first work that uses off-the-shelf open-source software to implement a data streaming warehouse. Other solutions, such as DateDepot [1], provide similar features, however only using customized close source software. DBStream [2], which is also an open-source DSW, uses PostgreSQL as a query engine. ENTRADA, on the other hand, uses off-the shelf Impala query engine and Parquet file format based on Google's Dremel [16] to achieve high performance. It is not clear, however, if DBStream is capable to deliver the same performance as ENTRADA: with 10 nodes in a cluster analyzing several aggregating queries (J1) over a 640GB raw dataset, it took the DBStream more than 50 minutes. It took ENTRADA less than 3.5 minutes to analyze almost 4 times more compressed data (2.2 TB) datasets in a 6-node cluster. However, a precise benchmark is still necessary to assess the performance.

Other works, such as [3], [4], [5], focus of "off-line" analysis in Hadoop clusters, i.e., data analysis on snapshot data that does not need long term storage. ENTRADA, however, is designed to be append-only – i.e., new data is continuous appended but not updated, and delivers near real-time responses.

VIII. Conclusions

We presented ENTRADA, a data streaming warehouse designed to deliver interactive response times on large datasets, via SQL-like queries. This is achieved by employing both an optimized file format and a query engine, which allows researchers and operators to easily and quickly investigate their hypothesis. It can be extended to store not only networking-related data, but any sort of structured data.

ENTRADA has been operational for more than 1.5 years (Dec. 2015), storing DNS traffic from two of the .nl authoritative servers. We use ENTRADA as a enabler in projects that aim at improving both security and stability of the .nl zone.

The lessons here presented can be used as guidelines for other research teams and DNS operators to deploy their own data analysis clusters – which can also be done by using cloud providers. To help in this process, we open-source the modules we have developed for ENTRADA and make it available at [9].

[15] https://www.abuseinformationexchange.nl/english

[16] http://iepg.org/2015-11-01-ietf94/iepg-moura.pdf

Acknowledgments: The authors would like to thank Marco Davids and Jelte Jansen for their valuable comments.

References

[1] L. Golab, T. Johnson, J. S. Seidel, and V. Shkapenyuk, "Stream Warehousing with DataDepot," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '09. New York, NY, USA: ACM, 2009, pp. 847–854.

[2] A. Bar, A. Finamore, P. Casas, L. Golab, and M. Mellia, "Large-scale network traffic monitoring with DBStream, a system for rolling big data analysis," in *Big Data (Big Data), 2014 IEEE International Conference on*, Oct 2014, pp. 165–170.

[3] T. Vanhove, G. Van Seghbroeck, T. Wauters, F. De Turck, B. Vermeulen, and P. Demeester, "Tengu: An experimentation platform for big data applications," in *Distributed Computing Systems Workshops (ICDCSW), 2015 IEEE 35th International Conference on*, June 2015, pp. 42–47.

[4] J. Liu, F. Liu, and N. Ansari, "Monitoring and analyzing big traffic data of a large-scale cellular network with Hadoop," *Network, IEEE*, vol. 28, no. 4, pp. 32–39, July 2014.

[5] Y. Lee and Y. Lee, "Toward Scalable Internet Traffic Measurement and Analysis with Hadoop," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 1, pp. 5–13, Jan. 2012.

[6] T. White, *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 2009.

[7] N. Leavitt, "Will NoSQL Databases Live Up to Their Promise?" *Computer*, vol. 43, no. 2, pp. 12–14, Feb 2010.

[8] E. Liarou, S. Idreos, S. Manegold, and M. Kersten, "Monetdb/datacell: online analytics in a streaming column-store," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1910–1913, 2012.

[9] SIDN Labs, "ENTRADA homepage," http://entrada.sidnlabs.nl/, 2015.

[10] Apache, "Apache Parquet," https://parquet.apache.org/, 2015.

[11] M. Kornacker, A. Behm, V. Bittorf, T. Bobrovytsky, C. Ching, A. Choi, J. Erickson, M. Grund, D. Hecht, M. Jacobs *et al.*, "Impala: A modern, open-source SQL engine for Hadoop," in *Proceedings of the Conference on Innovative Data Systems Research (CIDR'15)*, 2015.

[12] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, 2010.

[13] SIDN, "SIDN: the company behind the .nl," http://sidn.nl/en, 2015.

[14] P. Mockapetris, *RFC 1034 Domain Names - Concepts and Facilities*, Internet Engineering Task Force, 1987.

[15] M. Andrews, "Negative Caching of DNS Queries (DNS NCACHE)," RFC 2308, Internet Engineering Task Force, Mar. 1998.

[16] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis, "Dremel: Interactive Analysis of Web-scale Datasets," *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 330–339, Sep. 2010.

[17] C. Hesselman, J. Jansen, M. Wullink, K. Vink, and M. Simon, "A privacy framework for DNS big data applications," Tech. Rep., 2015. [Online]. Available: https://www.sidnlabs.nl/uploads/tx_sidnpublications/SIDN_Labs_Privacyraamwerk_Position_Paper_V1.4_ENG.pdf

[18] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[19] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, May 2010, pp. 1–10.

[20] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, P. O'Neil, A. Rasin, N. Tran, and S. Zdonik, "C-store: A column-oriented dbms," in *Proceedings of the 31st International Conference on Very Large Data Bases*, ser. VLDB '05. VLDB Endowment, 2005, pp. 553–564.

[21] RIPE NCC, "Hadoop PCAP library," 2015. [Online]. Available: https://github.com/RIPE-NCC/hadoop-pcap

[22] P. Mockapetris, "Domain names - implementation and specification," RFC 1035, Internet Engineering Task Force, Nov. 1987.

[23] SIDN Labs, ".nl stats and data: Insight into the use of .nl," http://stats.sidnlabs.nl/, 2015.