Applicances of different kind of storage systems for network traffic analysis results

Vladislav Egorov
Ivannikov Institute for System Programming of the RAS,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
Moscow, Russia
unclehook@ispras.ru

Roman Ponomarenko
Ivannikov Institute for System Programming of the RAS,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
Moscow, Russia
rerandom@ispras.ru

Aleksandr Getman
Ivannikov Institute for System Programming of the RAS,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
National Research University "Higher School of Economics",
20 Myasnitskaya ulitsa, Moscow 101000 Russia
Moscow Institute of Physics and Technology,
9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia.
Lomonosov Moscow State University,
1 Leninskie Gory, Moscow, 119991 Russia
Moscow, Russia
ever@ispras.ru

***Abstract*—Network Traffic Analysis (NTA) helps identify security threats, monitor network performance, and plan for future capacity. While real-time analysis is ideal, it can be difficult due to high data volume and complexity. Large amounts of traffic require parsing, and real-time data may miss hidden threats. Post-analysis can address these challenges. It hardly depends on choosing an effective and appropriate storage solutions. A variety of storage systems exist, each employing different approaches and formats to retain data. This article explores the applications of various storage systems for NTA results. Three different types of storage systems considered, including Greenplum, Nebula graph and OpenSearch. A comparative approach is employed, analyzing the same dataset across various storage systems.This allows to examine how different database structures and query capabilities influence the efficiency and accuracy of NTA. The resulting insights will not only provide valuable guidance for selecting the optimal storage solution for specific NTA tasks, but also serve as a foundation for future research in this area.**

***Index Terms*—Network Traffic Analysis, storage systems, database analysis, greenplum, OpenSearch, nebula graph**

I. Introduction

Network Traffic Analysis (NTA) have many appliances such as security threats identification, network performance monitoring and bottlenecks, capacity planning of network infrastructure and different application monitoring. While real time traffic analysis is very important it might be hard to achieve. Moreover, it could be unachievable is some cases. There are two main reasons why analyzing network traffic in real-time is challenging: Volume of Data and Data Complexity. Network traffic can be immense, especially in large organizations or environments with high bandwidth. Another important thing is that not all network traffic is readily interpretable. Raw data packets need to be parsed and analyzed to understand the context and identify anomalies. In other cases information from real-time data is not enough. For example, threat incidents might be identified only then it already appeared. Post-facto analysis is a viable solution to this problem. Stored network traffic analysis results have a wide range of practical applications, here are some key areas:

- Stored data allows for forensic investigation after a security breach. Analysts can examine historical traffic patterns to identify suspicious activity leading up to the incident.
- Stored traffic data can be used to train machine learning models to recognize malicious patterns. These models can then be used for real-time analysis to proactively block threats.
- Historical traffic data helps predict future network demands. This information is crucial for network administrators to plan for infrastructure upgrades or capacity expansion to ensure smooth network operation.
- By analyzing historical traffic patterns, security teams can build baselines for normal network activity. Deviations from these baselines can indicate potential threats like malware or unauthorized access attempts.

The biggest challenge for NTA systems lies in designing a robust data storage solution that is flexible, scalable, and delivers fast performance. The large volume of data generated necessitates this robust infrastructure to ensure efficient storage and analysis. Choosing the optimal database for this demanding environment presents distinct challenges, including:

978-1-7281-4387-3/19/$31.00 © 2024 IEEE

- Data Variety: Effectively accommodating the diverse data formats generated by NTA systems within a single database.
- Performance: Ensuring efficient data retrieval and analysis, especially for real-time security monitoring.
- Scalability: Adapting the database to accommodate the ever-growing volume of network traffic data.
- Security: Implementing robust security measures to safeguard sensitive network data stored within the database.

There are various types of storage systems, each with its own strengths and weaknesses. Traditional options include relational database management systems which follows SQL standard. They might be categorized into OnLine Analytical Processing (OLAP) and Online Transaction Processing (OLTP) systems. NoSQL databases differ from SQL by not adhering to the SQL standard. Examples include document-oriented, key-value, graph, and hierarchical databases. The selection of a storage system depends heavily on the data type, its characteristics, and how it will be used. For our experiment, three database types were chosen. Since Network Traffic Analysis (NTA) results often involve analytical queries, an OLAP database might be sufficient. Greenplum[1], a popular and advanced OLAP database with horizontal scaling and full SQL compatibility[2], is a strong contender. It also supports OLTP type of load. Document-oriented databases with full-text search capabilities are also interesting because they are schema-less and offer advanced search functionalities. Elasticsearch[3] is a well-known document-oriented database, but its license is not entirely open-source. This why, OpenSearch[4], a fork of Elasticsearch, will be considered. Given that NTA results often consist of interconnected data, a graph database could be a suitable option. While Neo4J[5], Dgraph[6], and others exist, Nebula Graph[7] was chosen for its proven horizontal autoscaling capabilities and, more importantly, its ability to handle large amounts of data[8].

Several experiments were conducted to identify the most suitable storage system for NTA results. These experiments focused on data load times, data acquisition metrics, and disk space usage. To ensure a fair comparison, all experiments used the same network traffic data set.

This paper is organized as follows. Section II describes the distinct characteristics of NTA data and the challenges associated with its storage. Then, It describes the general problem statement, structure, and unique characteristics of network traffic data. Section III includes analyzes of existing storage solutions for NTA, their strengths and weaknesses outlined. Section IV provides general design considerations and data schema used in experimental part. Section V contains evaluation results which include comparison of three database types for storing NTA results, evaluating their limitations and advantages. By leveraging a comprehensive analysis of these works and recent advancements in NTA research, we unveil a comparative evaluation of various database types. This in-depth exploration include relational, NoSQL, and graph databases, meticulously highlighting their strengths and weaknesses within the context of NTA data. Experimental results, including performance estimations, storage space requirements, and other relevant metrics, are presented. Section VI concludes article by summarizing the findings and highlighting potential areas for future research to improve storage solutions for NTA data.

II. Problem statement

For better understanding problems connected with storing NTA results, key characteristics of this type of stored data must be presented. First of all, storing and processing of network traffic falls under the concept of big data[9] due to its significant volume, variety, and speed of new data acquisition. Data can be generated at a high speed, which makes it difficult to process and store in real time. The volume of network traffic data can reach petabytes in large organizations, requiring scalable storage solutions and efficient processing methods. In article[9], suggested NewSql and NoSQL databases over relational databases, because they inappropriate on big datasets. In NTA systems time of occuring some events plays big part. This kind of systems often relies on temporal context, as patterns and anomalies can be associated with specific times, dates, or intervals. For example, time arrival of packets can be used as classification characteristic for deep learning methods [10], [11]. Time stamps and temporal metadata are crucial for effective analysis and storage of network traffic, as they help to evaluate various statistical patterns in the large volume of network traffic. For instance, a burst in network traffic[12] at unusual hours might indicate suspicious activity, while recurring spikes during business hours could point to bandwidth bottlenecks. Therefore, time stamps and precise temporal metadata are crucial for efficient analysis and storage of NTA data. They empower security professionals to not only identify potential threats but also gain valuable insights into network usage patterns and resource allocation needs. By effectively leveraging temporal data, organizations can achieve a deeper understanding of their network behavior and proactively address potential security vulnerabilities and performance issues. Network traffic analysis can be categorized by two main levels of detail[13]: packet-level and flow-level. While packet-level network data offers a highly detailed view of traffic, it necessitates powerful, specialized equipment. This approach becomes impractical for expansive networks due to scalability limitations with a growing number of devices. Additionally, storing such granular data demands significant storage capacity. Flow-level analysis, on the other hand, provides a more scalable and privacy-conscious alternative. By aggregating packets, flow-level data offers a sufficient level of detail for network monitoring in modern environments with vast amounts of traffic and numerous connected devices.

III. Related work

The challenge of storing network traffic analysis results has been around for decades. Early attempts relied on relational databases. TelegraphCQ[14], for example, leveraged PostgreSql[15] for this purpose, with extensions for handling large continuous queries over ever-changing network data streams.

TelegraphCQ focuses on handling large streams of continuous queries over variable data streams. Later, timemachine[16] emerged, offering a cost-effective solution that utilized commodity hardware to buffer high-volume traffic for several days. The core principle behind this time machine lies in the "heavy-tailed" distribution of network traffic. This allows for capturing most connections in their entirety, while strategically skipping less critical data, all within a configurable per-connection byte limit. Some of the suggested approaches[17] based on special data format of network data, such as NetFlow[18] and IpFIX[19]. This kind of solutions facilitate the cost-effective monitoring of high-bandwidth links, using off-the-shelf hardware capabilities. Further this approach improved by NetMemex[20], provide network flow data with full packet payload. More advanced solution[21] separate constraints of static data analysis, exploiting high bandwidth cluster solutions which gain immediate insights into dynamic environments through seamless data acquisition and analysis. Unveiling historical context with rapid responsiveness, empowering data-driven decision-making at every level. These comprehensive solution, often referred to as streaming data warehouses, represent a paradigm shift in network data analysis, offering unparalleled flexibility and agility. The researchers highlight a key advantage of their solution by contrasting it with existing systems like those[22] based on Apache Hadoop. While Hadoop offers valuable functionalities, it's limited to analyzing data captured at specific points in time (snapshots). This can be a disadvantage when real-time insights are crucial. Many of mentioned solutions didn't exist in free access nowadays. In the next paragraphs, several most recent and interesting technologies introduced.

PcapDB[23] is a distributed, open-source, and search-optimized packet capture system. It is designed to replace expensive commercial appliances with off-the-shelf hardware and a free, easy-to-manage software system. Captured packets are reorganized during capture by flow (an indefinite length sequence of packets with the same src/dst IPs/ports and transport protocol), indexed by flow, and searched (again) by flow. The indexes for the captured packets are relatively tiny (typically less than 1% the size of the captured data). Data captured in Pcap format is stored on-site at each Capture Node, minimizing network traffic. This setup enables cyber incident responders and analysts to swiftly search through indexed data instead of raw Pcap files, significantly cutting down query times, and allowing for searches across various capture locations. PcapDB indexes build on top of the PostgreSql but developers didn't describe them in detail. While indexes in PcapDB allows to fast search networks it strugle of luck of functionality. It allows only to search information through network and transport level of TCP/IP stack.

In 2017 group of researchers presented Moloch[24] an open source, large scale, full packet capturing, indexing, and database system. It build on top of the Elasticsearch[3] database system. Later, developers of Moloch evolved it to Arkime[25]. Arkime group network packets by logical flows of data, grouping by network addresses information. Arkime store all possible data in one index. Every document in this index may consists of any of supported network protocol records. A key benefit of this approach is the ability to perform full-text searches across captured network traffic. This facilitates easy exploration of large datasets. However, using a full-text search engine like Elasticsearch can introduce performance overhead, especially for structured data like network traffic. Elasticsearch offers horizontal scaling by distributing data across nodes (sharding). While this enhances scalability, it can increase query complexity and impact performance. Replication ensures data redundancy but comes at the cost of higher storage consumption and write latency. Striking a balance between these features can be challenging. Additionally, large and frequently updated datasets can strain the indexing process, affecting search performance.

GRANEF[26] — Graph-based network forensics is a new approach to analyzing network traffic data. GRANEF toolkit utilizes Dgraph database for storing and querying data. Main advantages in links between various network members. It allows analysts to easily navigate and visually identify interesting network traffic. Dmitry Larin's 2023 master's thesis[27] proposed a novel approach for describing network topology using a multilevel graph model. This model leverages data from OSPF and BGP communication between routers to represent the network topology across different layers of the TCP/IP protocol suite. The approach facilitates analysis of the network's state at each level. The storage architecture utilizes two key components: Nebula Graph and Clickhouse. Nebula Graph is a graph database optimized for efficient graph queries, which is particularly well-suited for the type of data being analyzed. ClickHouse[28], an OnLine Analytical Processing (OLAP) database, serves as the storage for time-series events. The proposed composite data storage approach leverages two types of databases to gain the advantages of each.

IV. General design

A. Data schema

Main investigated data schema is based on [29] dissertation work. This data model offers a method for analyzing network interactions independent of underlying protocols. It consider network communication as a set of logic connections, there packets transfered between logical entities residing at the same level within the network architecture. This approach allow to don't get attached to specific protocol stacks, allowing for the flexible representation of diverse network interactions. Logic connection described by concrete network protocol type is called Context. Distinct instances of a defined context can be efficiently differentiated by their unique key. Key is usually presented in protocol headers and usually describes some address information. In general, context key is presented in binary format, but may be serialized in different formats. At this moment it serialized into json[30]. Json bring more flexibility in further processing and exporting of data.

Set of instances of Context present a tree structure. In this tree each Context connected with it parent. Parent symbolize

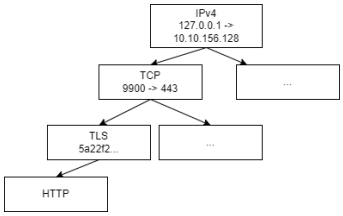

Figure 1. Context tree

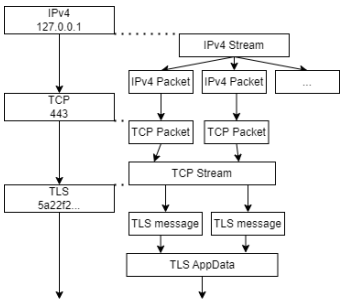

Figure 2. Block tree

more lower layer in network stack. Example of context tree presented on Figure 1.

While context describe logical connections, payload of this connections either network streams or packets with header fields, presented by blocks. Block is universal structure which describes sequence of bytes, with some characteristics. Each block is assigned a parse type. Parse type is similar to Context type, and connected with type of network stream, name of network protocol packet or type of field of packet. Proposed data schema semantically separate blocks by two subtypes: stream block and fragment block. The stream block usually acts as a data source for another blocks. It may define dedicated network stream or original network data. Fragment block more simple thing, it just define network packet or protocol field. Every fragment block must have offset and size which define range of bytes in data source related to this block. Each block possesses a semantic connection to the specific context in which it appears.

Proposed data schema utilizes a three-tiered approach to represent network connections Figure 2:

1) Top Level provides metadata describing the overall logical network connection across various network stack layers.
2) Second Level described by stream blocks represent unique network flows, each encapsulating a specific communication stream and it payload.
3) Last level delve into the most granular details, describing individual network packets or specific fields within those packets by Fragment block

B. Sql database (greenplum)

Greenplum [1] is an open-source data warehouse software built on top of PostgreSQL[15]. It is designed for handling large datasets and complex queries. Greenplum's MPP architecture distributes data and workloads across multiple servers, enabling efficient processing of large-scale data warehouses and analytics tasks. It also supports some specific data types like net-types[31] which can be useful for network data analysis. Greenplum scales horizontally by adding more servers, this allows to seamlessly grow storage and processing capacity as overall data volumes increase. This advantage is very important for storing always increasing amount of network data. Furthermore, it is fully sql compatible and inherited most of PostgreSql functions. Main feature of greenplum is support of OLAP data storage format. As work with NTA results gets into description of big data there is no big need for OLTP operations like update or delete. OLAP databases are optimized for performing aggregations and complex queries on large datasets. This enables network analysts to quickly identify trends, pinpoint peak traffic periods, and analyze traffic patterns across different user groups or applications. Greenplum allows to recursively fetch rows by recursive Common Table Expression (CTE)[32]. This feature helps to easily work with context and block tree of supposed data schema. One of inherited features of PostgreSql is support of JSONB[33] data format. This kind of data can also be stored as raw text, but jsonb add aditional features like: more compact storage format, support of specialized functions for search. JSONB also supports indexing, which can be a significant advantage. All this features allows to easily search through Context keys of experimental data schema.

C. NoSQL database(OpenSearch)

OpenSearch[4] is open source fork of elasticsearch popular full-text search database. OpenSearch excels at horizontal scaling by adding more nodes to the cluster. This allows to handle growing volumes of network traffic data efficiently without sacrificing performance. Advanced full-text search engine allows to easily find and filter some information across various network traffic fields (IP addresses, URLs, protocols) for in-depth analysis. It also support of aggregation and extended analytic. OpenSearch's aggregation framework allows to analyze network traffic data from various angles. Moreover, it provides insightful visualizations, which can be used to identify trends, patterns, and anomalies in traffic flows. Unlike traditional databases with rigid schemas, OpenSearch offers a schema-less design. This provides flexibility to store and analyze network traffic data with diverse formats and structures. While Elasticsearch is great for near real-time analysis, for very long-term historical data analysis, other solutions like data lakes might be more efficient. Index in OpenSearch defined by mappings. Each mapping consists of fields with any of the supported types. One of the supported types is Object[34]. An object field type contains a json object. A value in a json object may be another json object. This kind of fields automatically indexed by search engine and their containment may be used in queries. Big disadvantages of OpenSearch is hard working with related data. There is no some kind of sql join or reference. OpenSearch provide basic

relations only between documents of one index by mechanism of parent/child relationship[35]. This approach didn't allow to represent tree like structures.

D. Graph database(Nebula graph)

Nebula graph[7] excels at modeling and querying relationships between data points. This could be beneficial for network traffic analysis if you want to understand how different devices, users, or IP addresses interact with each other. For instance, tracking connections within a malware outbreak or visualizing communication patterns within a network. Nebula boasts impressive query performance for interconnected data, allowing for efficient retrieval of specific network traffic flows based on relationships. Nebula can scale horizontally to handle growing volumes of network traffic data. While Nebula Graph Database offers a variety of data types, it doesn't natively support storing and indexing JSON data directly. However, it provides a workaround: JSON data can be dynamically converted (casted) into a specific data type called a "Map"[36]. This "Map" type essentially acts as a representation of a JSON object within Nebula Graph and can be used in complex queries for flexible data manipulation.

E. Binary data

Network traffic analysis results come in a variety of formats, and understanding these formats is crucial for effective storage and analysis. One key aspect to consider is that this data can often be presented in binary format. Common storage formats for network traffic captures include pcap (packet capture) and its successor, pcapng. These formats preserve the raw network packets, allowing for in-depth forensic analysis. Additionally, network traffic analysis can involve extracting network flow data, which summarizes conversations between devices rather than individual packets. Furthermore, network traffic analysis may involve processing captured data through specific data changing alghorithms like decompression and decryption, which result must be stored in binary format. The only storage which supports raw bytes data is greenplum. It allows to store raw bytes in specific format called binary large objects(BLOB[37]). While Elasticsearch doesn't natively support raw bytes, it can accommodate binary data encoded in base64 format[38]. However, the process of encoding and decoding can introduce significant performance overhead, especially for large datasets. Moreover, binary data in OpenSearch is not searchable. Unlike Greenplum and Elasticsearch, Nebula currently doesn't offer direct support for storing raw byte data types. Binary data often presents indexing challenges for traditional databases. Storing it literally within a database might not be the most performant or scalable solution. Best practices for storing this kind of data lay in usage of external object storages, like CEPH, HDFS and etc. In this research raw bytes data will be ignored.

V. Evaluation

A. Experiment design

To ensure a fair comparison of storage systems for NTA results, a single network data set was used throughout the experiments for each of considered databases. This ensured a controlled environment for evaluating performance and facilitated a direct comparison of the systems' capabilities for loading and retrieving data. The first set of experiments investigated data loading times. Minimizing this metric is crucial for NTA due to the large volumes of data typically collected and the limited storage capacity of collectors. Following successful data loading, storage space usage was analyzed. Subsequent experiments focused on data acquisition performance for loaded data. Here, the specific tree-based data schema, as described in Section IV, played a critical role. The goal was to efficiently extract data with minimal connections between elements. For metrics collection Opentelemetry[39] used. OpenTelemetry uses tracing to capture the performance of requests or tasks. Percentiles can then be used to summarize the distribution of these execution times. In this context, p50 (50th percentile) would likely represent the duration at which 50% of requests took less time to complete, and p90 (90th percentile) would represent the duration at which 90% of requests took less time to complete.

B. Dataset

Basic experimental dataset it is a network trace in pcap format. Table I summarizes key statistics about the trace content. The trace serves as input for a network traffic analysis system. After processing, the results are loaded into one of the storage systems under consideration.

Table I
DATASET STATISTICS

Statistic	Value
Overall size, Mbytes	5893.71
Number of unique contexts	150355
Number of unique streams	19117
Number of IP packets	6067525
Number of TCP packets	2463840
Number of HTTP streams	2548

C. Hardware

The experiments were conducted using servers with specific hardware configurations. These servers included Intel(R) Xeon(R) Silver 4314 CPUs clocked at 2.40 GHz, 256 GB of memory, and Dell EMC VD SSDs. One important property for NTA storage is horizontal scaling. This type of scaling, often achieved through distributed databases, which allows to distribute the data load across multiple nodes. To leverage this benefit, all the considered databases were deployed in a multi-node configuration, with each database running on two nodes.

D. Data load

Different databases handle concurrent access in varying ways. For instance, Greenplum utilizes a centralized approach. Client requests are received by the master node, which then forwards them to the appropriate data node for processing. In contrast, OpenSearch employs a distributed client-based load balancing strategy. Here, any node within the cluster can

Table II
LOAD STATISTICS

	Greenplum		OpenSearch		Nebula	
	C	S	C	S	C	S
Time, s	1087		770		1012	
p50, ms	5.9	5.9	4.2	3.5	5.6	5.7
p90, ms	7.1	7	4.6	3.8	6.1	6.2
p99, ms	7.5	7.5	5.1	4.9	6.4	6.5
min, ms	3.4	4.07	2.74	2.48	3.18	4.65
max, ms	54.25	29.4	217.5	153.4	212.1	8.44
SD, ms	0.73	0.83	0.69	2.06	0.76	0.44
spans/min	7160	910	9400	1190	7520	954

Table III
SPACE STATISTICS

Space stats	
Greenplum, MB	181
Nebula, MB	115.7
OpenSearch, MB	20.7

potentially receive and handle client requests. It's important to note that in this experiment, data loading is performed synchronously within a single thread. Statistics of loading data into databases presented in Table II. In Table II, C refers to Context and S refers to Stream.

Important characteristic for consider is size of stored data. Table III provides an overview of the storage space statistic for all types of databases.

E. Data query

For each database had executed queries which return same data. It is set of queries that return address information about HTTP streams by defined source IP address. In case of OpenSearch it is impossible to get connected data in one query. Acquisition of required data might be achieved only be series of queries. Total number of executed queries for each database is equal 330. Results of HTTP streams acquisition presented in Table IV.

It's worth noting that queries to OpenSearch performed in several requests, yet it still achieved impressive read speeds. The exceptional read speed of OpenSearch might be attributed to its handling of the "context key", the most valuable field for search within the considered data schema. Unlike Greenplum and Nebula, which required special type casts and functions for searching this JSON field, OpenSearch natively stored and indexed it in its original format. This native handling likely contributed to the superior read performance.

Table IV
FETCH STATISTICS

	Greenplum	OpenSearch	Nebula
p50, ms	80.4	18.9	170
p90, ms	83.1	140.4	176.6
p99, ms	92.8	264.9	181.5
min, ms	68.9	15.5	155.5
max, ms	140.9	348.5	198.7
SD, ms	5.05	62.2	5.7

VI. CONCLUSION

In this paper, appliance of different storage systems for NTA results was considered. The investigation focused on their suitability for storing and facilitating analysis of complex network data. This article evaluated the suitability of three storage systems (OpenSearch, Nebula Graph, Greenplum) for storing and analyzing complex network traffic data (NTA results). While OpenSearch offers the fastest write speeds and space efficiency, it lacks the ability to represent intricate relationships within the data. Nebula Graph excels at modeling these relationships but may not be ideal for very high data volumes. Greenplum provides a traditional relational model, allowing for flexible analysis but potentially consumes more storage space. The key takeaway is that there's no one-size-fits-all solution. The optimal storage system depends on specific network needs: High Data Volume: All evaluated databases can horizontally scale to accommodate significant data growth. Intricate Relationships: Nebula Graph excels at modeling complex network connections, making it ideal for scenarios where understanding these relationships is crucial. Loosely Connected Data: Greenplum's relational model with SQL support allows efficient storage of data points with fewer connections. Schema Flexibility: OpenSearch offers the most flexibility for storing various NTA data structures due to its schema-less architecture. The optimal storage solution depends on the specific requirements of the network environment and analysis needs. OpenSearch performed well in write benchmark tests, achieving the lowest time in load experiments. Moreover, OpenSearch impressed with its space efficiency. Impressively, in 50% of fetch experiments, this database outperformed all others in speed. However, it did experience occasional performance spikes. Greenplum showed average results in read tests and more important it had lowest deviation of values. Nebula performs well on load tests but it had some problems with fetching the data. Most inefficient space consumption demonstrate Greenplum. Future research could involve conducting controlled experiments to quantify the performance and scalability of each storage system under varying network traffic loads. Additionally, investigating the integration of these systems into a comprehensive NTA framework could offer valuable insights for network security professionals. By carefully considering the volume, structure, and desired analysis depth of their network data, organizations can select the most suitable storage system for their NTA efforts.

REFERENCES

[1] "Greenplum database." (2024), [Online]. Available: https://greenplum.org (visited on 03/20/2024).

[2] Z. Lyu, H. H. Zhang, G. Xiong, *et al.*, "Greenplum: A hybrid database for transactional and analytical workloads," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2530–2542.

[3] "Elasticsearch." (2024), (visited on 03/22/2024).

[4] "Opensearch." (2024), [Online]. Available: https : / / opensearch.org (visited on 03/20/2024).

[5] "Neo4j graph database & analytics — graph database management system." (2024), [Online]. Available: https://neo4j.com (visited on 03/22/2024).

[6] "Dgraph — graphql cloud platform, distributed graph engine." (2024), [Online]. Available: https://dgraph.io (visited on 03/22/2024).

[7] "Open source distributed graph database — nebula-graph." (2024), [Online]. Available: https://www.nebula-graph.io (visited on 03/22/2024).

[8] M. Wu, X. Yi, H. Yu, Y. Liu, and Y. Wang, "Nebula graph: An open source distributed graph database," *arXiv preprint arXiv:2206.07278*, 2022.

[9] A. D'Alconzo, I. Drago, A. Morichetta, M. Mellia, and P. Casas, "A survey on big data for network traffic monitoring and analysis," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 800–813, 2019.

[10] A. I. GETMAN and M. K. Ikonnikova, "A survey of network traffic classification," *Proceedings of the Institute for System Programming of the RAS (Proceedings of ISP RAS)*, vol. 32, no. 6, pp. 137–154, 2021.

[11] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE communications magazine*, vol. 57, no. 5, pp. 76–81, 2019.

[12] D. Wei, F. Shi, and S. Dhelim, "A self-supervised learning model for unknown internet traffic identification based on surge period," *Future Internet*, vol. 14, no. 10, p. 289, 2022.

[13] M. Piskozub, R. Spolaor, and I. Martinovic, "Compactflow: A hybrid binary format for network flow data," in *Information Security Theory and Practice: 13th IFIP WG 11.2 International Conference, WISTP 2019, Paris, France, December 11–12, 2019, Proceedings 13*, Springer, 2020, pp. 185–201.

[14] S. Chandrasekaran, O. Cooper, A. Deshpande, *et al.*, "Telegraphcq: Continuous dataflow processing," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003, pp. 668–668.

[15] "Postgresql: The world's most advanced open source database." (2024), [Online]. Available: https://www.postgresql.org/ (visited on 03/20/2024).

[16] S. Kornexl, V. Paxson, H. Dreger, A. Feldmann, and R. Sommer, "Building a time machine for efficient recording and retrieval of high-volume network traffic," in *5th Internet Measurement Conference*, USENIX Association, 2005, pp. 267–272.

[17] A. Bär, P. Casas, L. Golab, and A. Finamore, "Dbstream: An online aggregation, filtering and processing system for network traffic monitoring," in *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, IEEE, 2014, pp. 611–616.

[18] B. Claise, "Cisco systems netflow services export version 9," RFC Editor, RFC 3954, Oct. 2004. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3954.txt.

[19] B. Claise, B. Trammell, and P. Aitken, "Specification of the ip flow information export (ipfix) protocol for the exchange of flow information," RFC Editor, RFC 7011, Sep. 2013. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7011.txt.

[20] H. Lim, V. Sekar, Y. Abe, and D. G. Andersen, "Netmemex: Providing full-fidelity traffic archival," *arXiv preprint arXiv:1603.04387*, 2016.

[21] M. Wullink, G. C. Moura, M. Müller, and C. Hesselman, "Entrada: A high-performance network traffic data streaming warehouse," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2016, pp. 913–918.

[22] Y. Lee and Y. Lee, "Toward scalable internet traffic measurement and analysis with hadoop," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 1, pp. 5–13, 2012.

[23] S. I. Steinfadt and P. S. Ferrell, "Packet capture solutions: Pcapdb benchmark for high-bandwidth capture, storage, and searching," Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), Tech. Rep., 2017.

[24] J. Uramová, P. Segeč, M. Moravčík, J. Papán, T. Mokoš, and M. Brodec, "Packet capture infrastructure based on moloch," in *2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, IEEE, 2017, pp. 1–7.

[25] "Arkime." (2024), [Online]. Available: https://arkime.com (visited on 03/20/2024).

[26] M. Cermak and D. Sramkova, "Granef: Utilization of a graph database for network forensics.," in *SECRYPT*, 2021, pp. 785–790.

[27] D. Larin, "Razrabotka i primenenie modeli opisanija mnogourovnevyh setevyh topologij dlja reshenija zadachi monitoringa i modelirovanija setevoj infrastruktury," Master's thesis, Moscow Institute of Physics and Technology (National Research University), Moscow, Jun. 2023.

[28] "Fast open-source olap dbms - clickhouse." (2024), [Online]. Available: https://clickhouse.com (visited on 03/22/2024).

[29] Y. Markin, "Metody i sredstva uglublennogo analiza setevogo trafika," Ph.D. dissertation, ISP RAN, Moscow, 2017.

[30] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format," RFC 7159, Mar. 2014, 16 pp. [Online]. Available: https://www.rfc-editor.org/info/rfc7159.

[31] "Postgresql: Documentation: 16: 8.9. network address types." (2024), [Online]. Available: https://www.postgresql.org/docs/current/datatype-net-types.html (visited on 03/22/2024).

[32] "Greenplum: With queries (common table expressions)." (2024), [Online]. Available: https://docs.vmware.com/en/VMware-Greenplum/7/greenplum-database/admin_guide-query-topics-CTE-query.html (visited on 03/22/2024).

[33] “Postgresql: Documentation: 12: 8.14. json types.” (2024), [Online]. Available: https://www.postgresql.org/ (visited on 03/21/2024).
[34] “Object — opensearch documentation.” (2024), [Online]. Available: https://opensearch.org/docs/latest/field-types/supported-field-types/object/ (visited on 03/21/2024).
[35] “Join — opensearch documentation.” (2024), [Online]. Available: https://opensearch.org/docs/latest/field-types/supported-field-types/join/ (visited on 03/22/2024).
[36] “Map — nebulagraph database manual.” (2024), [Online]. Available: https://docs.nebula-graph.io/3.6.0/3.ngql-guide/3.data-types/8.map/ (visited on 03/20/2024).
[37] “Postgresql: Documentation: 12: 8.4. binary data types.” (2024), [Online]. Available: https://www.postgresql.org/docs/12/datatype-binary.html (visited on 03/20/2024).
[38] “Binary — opensearch documentation.” (2024), [Online]. Available: https://opensearch.org/docs/latest/field-types/supported-field-types/binary/ (visited on 03/20/2024).
[39] “Opentelemetry.” (2024), [Online]. Available: https://opentelemetry.io (visited on 03/22/2024).

APPENDIX

ACRONYMS

CTE Common Table Expression. 4

NTA Network Traffic Analysis. 1, 2, 4–6

OLAP OnLine Analytical Processing. 2–4

OLTP Online Transaction Processing. 2, 4