GraphBLAS on the Edge: Anonymized High Performance Streaming of Network Traffic

Michael Jones1, Jeremy Kepner1, Daniel Andersen2, Aydin Buluç3, Chansup Byun1, K Claffy2, Timothy Davis4, William Arcand1, Jonathan Bernays1, David Bestor1, William Bergeron1, Vijay Gadeppally1, Michéal Houle1, Matthew Hubbell1, Hayden Jananthan1, Anna Klein1, Chad Meiners1, Lauren Milechin1, Julie Mullen1, Sandeep Pisharody1, Andrew Prout1, Albert Reuther1, Antonio Rosa1, Siddharth Samsi1, Jon Sreekanth5, Doug Stetson1, Charles Yee1, Peter Michaleas1

1MIT, 2CAIDA, 3LBNL, 4Texas A&M, 5Accolade Technology

Abstract—Long range detection is a cornerstone of defense in many operating domains (land, sea, undersea, air, space, ...). In the cyber domain, long range detection requires the analysis of significant network traffic from a variety of observatories and outposts. Construction of anonymized hypersparse traffic matrices on edge network devices can be a key enabler by providing significant data compression in a rapidly analyzable format that protects privacy. GraphBLAS is ideally suited for both constructing and analyzing anonymized hypersparse traffic matrices. The performance of GraphBLAS on an Accolade Technologies edge network device is demonstrated on a near worst case traffic scenario using a continuous stream of CAIDA Telescope darknet packets. The performance for varying numbers of traffic buffers, threads, and processor cores is explored. Anonymized hypersparse traffic matrices can be constructed at a rate of over 50,000,000 packets per second; exceeding a typical 400 Gigabit network link. This performance demonstrates that anonymized hypersparse traffic matrices are readily computable on edge network devices with minimal compute resources and can be a viable data product for such devices.

Index Terms—Internet defense, packet capture, streaming graphs, hypersparse matrices

I. INTRODUCTION

The Internet has become as essential as land, sea, and space for enabling activities as diverse as commerce, education, health, and entertainment [1], [2]. Long range detection has been a cornerstone of defense in many operating domains since antiquity [3]–[10]. In the cyber domain, observatories and outposts have been constructed to gather data on Internet traffic and provide a starting point for exploring long range detection [11]–[17] (see Figure 1). The largest public Internet observatory is the Center for Applied Internet Data Analysis (CAIDA) Telescope that operates a variety of sensors including a continuous stream of packets from an unsolicited darkspace representing approximately 1/256 of the Internet [18]–[21]. In general, long range detection requires the analysis of significant network traffic from a variety of observatories and outposts [22], [23].

The data volumes, processing requirements, and privacy concerns of analyzing a significant fraction of the Internet have been prohibitive. The North American Internet generates billions of non-video Internet packets each second [1], [2]. The GraphBLAS standard provides significant performance and compression capabilities which improve the feasibility of analyzing these volumes of data [24]–[38]. Specifically, the GraphBLAS is ideally suited for both constructing and analyzing anonymized hypersparse traffic matrices. Prior work with the GraphBLAS has demonstrated rates of 75 billion packets per second (pps) [39], while achieving compressions of 1 bit per packet [17], and enabling the analysis of the largest publicly available historical archives with over 40 trillion packets [40]. Analysis of anonymized hypersparse traffic matrices from a variety of sources has revealed power-law distributions [41], [42], novel scaling relations [17], [40], and inspired new models of network traffic [43].

GraphBLAS anonymized hypersparse traffic matrices represent one set of design choices for analyzing network traffic. Specifically, the use case requiring some data on all packets (no down-sampling), high performance, high compression, matrix-based analysis, anonymization, and open standards. There are a wide range of alternative graph/network analysis technologies and many good implementations achieve performance close to the limits of the underlying computing hardware [44]–[54]. Likewise, there are many network analysis tools that focus on providing a rich interface to the full diversity of data found in network traffic [55]–[57]. Each of these technologies has appropriate use cases in the broad field of Internet traffic analysis.

Sending large volumes of raw Internet traffic to a central
location to construct anonymized hypersparse traffic matrices is prohibitive. To meet the goal of providing some data on all packets without down-sampling requires constructing the anonymized hypersparse traffic matrices in the network itself in order to realize the full data compression benefits. The goal of this paper is to explore the viability of this approach by measuring the performance of GraphBLAS on an on edge network device. The performance is measured on a near worse case traffic scenario using a continuous stream of CAIDA Telescope darknet packets (mostly botnets and scanners) which have an irregular distribution and almost no data payload (i.e., all header).

The outline of the rest of the paper is as follows. First, the CAIDA Telescope test data and some basic network quantities are defined in terms of traffic matrices. Next, the anonymized hypersparse traffic matrix pipeline is described followed by a description of the experimental setup and implementation. Finally, the results, conclusions, and directions for further work are presented.

II. TEST DATA AND TRAFFIC MATRICES

The test data is drawn from the CAIDA Telescope darknet packets (mostly botnets and scanners) and is a near worse case with a highly irregular distribution and almost no data payload (i.e., all header). The CAIDA Telescope monitors the traffic into and out of a set of network addresses providing a natural observation point of network traffic. These data can be viewed as a traffic matrix where each row is a source and each column is a destination. The CAIDA Telescope traffic matrix can be partitioned into four quadrants (see Figure 2). These quadrants represent different flows between nodes internal and external to the set of monitored addresses. Because the CAIDA Telescope network addresses are a darkspace, only the upper left (external → internal) quadrant will have data.

Internet data must be handled with care, and CAIDA has pioneered trusted data sharing best practices that combine anonymizing source and destinations using CryptoPAN [58] with data sharing agreements. These data sharing best practices are the basis of the architecture presented here and include the following principles [22]

- Data is made available in curated repositories
- Using standard anonymization methods where needed: hashing, sampling, and/or simulation
- Registration with a repository and demonstration of legitimate research need
- Recipients legally agree to neither repost a corpus nor deanonymize data
- Recipients can publish analysis and data examples necessary to review research
- Recipients agree to cite the repository and provide publications back to the repository
- Repositories can curate enriched products developed by researchers

A primary benefit of constructing anonymized hypersparse

Fig. 2. Network traffic matrix. The traffic matrix can be divided into quadrants separating internal and external traffic. The CAIDA Telescope monitors a darkspace, so only the upper left (external → internal) quadrant will have data.
TABLE I

NETWORK QUANTITIES FROM TRAFFIC MATRICES

Formulas for computing network quantities from traffic matrix $A_t$ at time $t$ in both summation and matrix notation. $1$ is a column vector of all $1$’s, $^T$ is the transpose operation, and $|\cdot|_0$ is the zero-norm that sets each nonzero value of its argument to $1$ [59]. These formulas are unaffected by matrix permutations and will work on anonymized data.

| Aggregate Property | Summation Notation | Matrix Notation |
|--------------------|---------------------|-----------------|
| Valid packets $N_V$ | $\sum_i \sum_j A_{i,j}$ | $^T A_t 1$ |
| Unique links | $\sum_i |A_{i,j}|_0 1^T |A_t|_0$ |
| Link packets from $i$ to $j$ | $A_{i,j}$ | $A_t$ |
| Max link packets ($d_{max}$) | $\max_{i,j} A_{i,j} (\text{max}(A_t))$ |
| Unique sources | $\sum_i |A_{i,(j)}|_0 1^T |A_t|_0$ |
| Packets from source $i$ | $\sum_j A_{i,j}$ | $A_t$ |
| Max source packets ($d_{max}$) | $\max_{i,j} A_{i,j} (\text{max}(A_t))$ |
| Source fan-out from $i$ | $\sum_j |A_{i,j}|_0 |A_t|_0$ |
| Max source fan-out ($d_{max}$) | $\max_{i,j} |A_{i,j}|_0 (\text{max}(A_t))$ |
| Unique destinations | $\sum_j |A_{(i,j)}|_0 1^T |A_t|_0$ |
| Destination packets to $j$ | $\sum_i A_{i,j}$ | $A_t$ |
| Max destination packets ($d_{max}$) | $\max_{i,j} A_{i,j} (\text{max}(A_t))$ |
| Destination fan-in to $j$ | $\sum_i |A_{i,j}|_0 1^T A_t$ |
| Max destination fan-in ($d_{max}$) | $\max_{i,j} |A_{i,j}|_0 (\text{max}(1^T A_t))$ |

traffic matrices with the GraphBLAS is the efficient computation of a wide range of network quantities via matrix mathematics. Figure 3 illustrates essential quantities found in all streaming dynamic networks. These quantities are all computable from anonymized traffic matrices created from the source and destinations found in Internet packet headers.

The network quantities depicted in Figure 3 are computable from anonymized origin-destination traffic matrices that are widely used to represent network traffic [60]–[63]. It is common to filter the packets down to a valid set for any particular analysis. Such filters may limit particular sources, destinations, protocols, and time windows. To reduce statistical fluctuations, the streaming data should be partitioned so that for any chosen time window all data sets have the same number of valid packets [64]. At a given time $t$, $N_V$ consecutive valid packets are aggregated from the traffic into a hypersparse matrix $A_t$, where $A_{t}(i,j)$ is the number of valid packets between the source $i$ and destination $j$. The sum of all the entries in $A_t$ is equal to $N_V$.

$$\sum_{i,j} A_{t}(i,j) = N_V$$

Constant packet, variable time samples simplify the statistical analysis of the heavy-tail distributions commonly found in network traffic quantities [41], [42], [65]. All the network quantities depicted in Figure 3 can be readily computed from $A_t$ using the formulas listed in Table I. Because matrix operations are generally invariant to permutation (reordering of the rows and columns), these quantities can readily be computed from anonymized data. Furthermore, the anonymized data can be analyzed by subranges of IPs using simple matrix multiplication. For a given subrange represented by an anonymized hypersparse diagonal matrix $A_r$, where $A_r(i,i) = 1$ implies source/destination $i$ is in the range, the traffic within the subrange can be computed via: $A_r A_t A_r$. Likewise, for additional privacy guarantees that can be implemented at the edge, the same method can be used to exclude a range of data from the traffic matrix $A_t - A_r A_t A_r$.

The contiguous nature of these data allows the exploration of a wide range of packet windows from $N_V = 2^{17}$ (sub-second) to $N_V = 2^{27}$ (minutes), providing a unique view into how network quantities depend upon time. These observations provide new insights into normal network background traffic that could be used for anomaly detection, AI feature engineering, polystore index learning, and testing theoretical models of streaming networks [66]–[68].

Network traffic is dynamic and exhibits varying behavior on a wide range of time scales. A given packet window size $N_V$ will be sensitive to phenomena on its corresponding timescale. Determining how network quantities scale with $N_V$ provides insight into the temporal behavior of network traffic. Efficient computation of network quantities on multiple time scales can be achieved by hierarchically aggregating data in different time windows [64]. Figure 4 illustrates a binary aggregation of different streaming traffic matrices. Computing each quantity at each hierarchy level eliminates redundant computations that would be performed if each packet window was computed separately. Hierarchy also ensures that most computations are performed on smaller matrices residing in faster memory. Correlations among the matrices mean that adding two matrices each with $N_V$ entries results in a matrix with fewer than $2N_V$ entries, reducing the relative number of operations as the matrices grow.

One of the important capabilities of the SuiteSparse GraphBLAS library is direct support of hypersparse matrices where the number of nonzero entries is significantly less than either dimensions of the matrix. If the packet source and destination identifiers are drawn from a large numeric range, such as those used in the Internet protocol, then a hypersparse representation of $A_t$ eliminates the need to keep track of additional indices and can significantly accelerate the computations [39].
III. HYPSPARSE MATRIX PIPELINE

The aforementioned analysis goals set the requirements for the GraphBLAS hypersparse traffic matrix pipeline. Specifically, the compression benefits are maximized if the GraphBLAS hypersparse traffic matrix is constructed in the network as close to the network traffic as possible as this minimizes the amount of data that needs to be sent over the network. In addition, collection at the network source allows the data owner to construct and own the anonymization scheme and only share anonymized data under trusted data sharing agreements with the parties tasked with analyzing the data [69].

The first step in the GraphBLAS hypersparse traffic matrix pipeline is to capture a packet, discard the data payload, and extract the source and destination Internet Protocol (IP) addresses (Figure 5). For the purposes of the current performance testing, only IPv4 packets are used which are stored as 32 bit unsigned integers. Collections of $N_V = 2^{17}$ consecutive packets are then each anonymized using a cryptoPAN generated anonymization table. The resulting anonymized source and destination IPs are then used to construct a $2^{32} \times 2^{32}$ hypersparse GraphBLAS matrix. 64 consecutive hypersparse GraphBLAS matrices are each serialized in compressed sparse rows (CSR) format with LZ4 compression and saved to a UNIX TAR file (Figure 6). The TAR files can be further compressed using other compressing methods (if desired) and saved to a file after all 64 subblocks have been processed. A more detailed outline of the code is as follows:

V. RESULTS

Using the experimental setup shown in Figure 7, a number of performance experiments were conducted. In these experiments the sender server would load $100 \times 2^{23}$ CAIDA Telescope packets into the sender Accolade card and sends the packets at defined rate over the network to the receiver server where the receiver Accolade card loads the packets into its hardware rings. Likewise, on the receiver server the GraphBLAS hypersparse traffic matrix pipeline described in the previous section was executed. The cryptoPAN anonymization table was created offline, stored, and loaded at startup. The reason for this is that the single core cryptoPAN performance is approximately 700,000 IP address per second. The $2^{32} \times 2^{32}$ entry cryptoPAN anonymization lookup table dramatically speeds up the performance. Generation of the table is readily run in parallel and can be generated in a few seconds (if desired).
**Fig. 5. Network Packet Description.** A network packet consists of a header and a payload. To avoid downsampling and minimize privacy concerns only the source and destination are selected.

**Fig. 6. Anonymized Hypersparse Matrix Pipeline.** Continuous sequences of $N_{V} = 2^{17}$ packets are extracted from packet headers, anonymized, formed into a GraphBLAS hypersparse matrix, serialized, and saved in groups of 64 GraphBLAS matrices to a UNIX TAR file.

**Fig. 7. Experimental Setup.** Test system consisted of two compute nodes each with an Accolade card connected over a network. The sender node reads CAIDA Telescope packet data into memory and then the Accolode card sends the data directly from memory over the network to the receiver node. The Accolade card on the receiver takes the data off the network, places the data in a hardware ring buffer, and makes the data available to be processed by the receiver processor.

**Fig. 8. Performance Results.** Packets per second processed versus the number of hardware rings used. The packet performance can be converted into an estimated equivalent bandwidth using a representative packets size of 10,000 bits per packet (see right vertical axis). The rate of packets being sent was adjusted to achieve the maximum rate without dropped packets, which indicated that the GraphBLAS hypersparse traffic matrix pipeline was able to keep up with the incoming traffic. The number of hardware rings were varied which increase the number of threads/cores.
CAIDA Telescope data are a near worse case scenario because almost all of the packets have no payload and very few packets use the same source and destination connection so the resulting hypersparse traffic matrices have very few entries that are greater than 1. An advantage of the few payloads in the CAIDA Telescope data is that it allows the emulation of packet streams that are representative of much higher bandwidth networks than the current experimental setup is capable of.

Figure 8 shows the performance in terms packets per second versus the number of hardware rings used. GraphBLAS matrix construction is highly cache sensitive due to the underlying index sorting required. The performance of a single ring using a few processing threads/cores is over 50M packets per second. The performance increases significantly using two rings, and then drops with 3 rings because of cache effects. Using 16 rings makes up for the cache effects with increased parallelism and the performance reaches the maximum number of packets the sender server can send to the receiver (approximately 88M packets per second). The packet performance can be converted into an estimated equivalent bandwidth using a representative packet size of 10,000 bits per packet (see Figure 8 right vertical axis). The performance measurements indicate that a standard server is a capable of constructing anonymized hypersparse traffic matrices at a rate above that corresponding to a typical 400 Gigabit network link.

Figure 9 shows the processors used as computed by the max load of the Linux top command, average load of the Linux perf command, and the max load of the Linux ps command. Processors Used

As determined by the max load of the Linux top command, average load of the Linux perf command, and the max load of the Linux ps command.

CAIDA Telescope data are a near worse case scenario because almost all of the packets have no payload and very few packets use the same source and destination connection so the resulting hypersparse traffic matrices have very few entries that are greater than 1. An advantage of the few payloads in the CAIDA Telescope data is that it allows the emulation of packet streams that are representative of much higher bandwidth networks than the current experimental setup is capable of.

Figure 8 shows the performance in terms packets per second versus the number of hardware rings used. GraphBLAS matrix construction is highly cache sensitive due to the underlying index sorting required. The performance of a single ring using a few processing threads/cores is over 50M packets per second. The performance increases significantly using two rings, and then drops with 3 rings because of cache effects. Using 16 rings makes up for the cache effects with increased parallelism and the performance reaches the maximum number of packets the sender server can send to the receiver (approximately 88M packets per second). The packet performance can be converted into an estimated equivalent bandwidth using a representative packet size of 10,000 bits per packet (see Figure 8 right vertical axis). The performance measurements indicate that a standard server is a capable of constructing anonymized hypersparse traffic matrices at a rate above that corresponding to a typical 400 Gigabit network link.

The performance increases significantly using two rings, and then drops with 3 rings because of cache effects. Using 16 rings makes up for the cache effects with increased parallelism and the performance reaches the maximum number of packets the sender server can send to the receiver (approximately 88M packets per second). The packet performance can be converted into an estimated equivalent bandwidth using a representative packet size of 10,000 bits per packet (see Figure 8 right vertical axis). The performance measurements indicate that a standard server is a capable of constructing anonymized hypersparse traffic matrices at a rate above that corresponding to a typical 400 Gigabit network link.

Table 1: Processors Used

| Number of Rings | Processors Used |
|-----------------|-----------------|
| 0               | 5               |
| 5               | 10              |
| 10              | 15              |
| 15              | 20              |
| 20              | 25              |
| 25              | 30              |

ACKNOWLEDGMENTS

The authors wish to acknowledge the following individuals for their contributions and support: Bob Bond, Stephen Buckley, Ronisha Carter, Cary Conrad, Alan Edelman, Tucker Hamilton, Jeff Gottschalk, Nathan Frey, Chris Hill, Mike Kanaan, Tim Kraska, Andrew Morris, Charles Leiserson, Dave Martinez, Mimi McClure, Joseph McDonald, Sandy Pentland, Christian Prothmann, John Radovan, Steve Rejto, Daniela Rus, Matthew Weiss, Marc Zissman.

REFERENCES

[1] “Cisco Visual Networking Index: Forecast and Trends.” https://newsroom.cisco.com/press-release-content?articleId=1955935.
[2] “Cisco Visual Networking Index: Forecast and Trends, 2018–2023.” https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.
[3] W. P. Delaney, Perspectives on Defense Systems Analysis. MIT Press, 2015.
