Anonymized Network Sensing Graph Challenge

Hayden Jananthan[1], Michael Jones[1], William Arcand[1], David Bestor[1], William Bergeron[1], Daniel Burrill[1], Aydin Buluc[2], Chansup Byun[1], Timothy Davis[3], Vijay Gadepally[1], Daniel Grant[4], Michael Houle[1], Matthew Hubbell[1], Piotr Luszczek[1,5], Peter Michaleas[1], Lauren Milechin[1], Chasen Milner[1], Guillermo Morales[1], Andrew Morris[4], Julie Mullen[1], Ritesh Patel[1], Alex Pentland[1], Sandeep Pisharody[1], Andrew Prout[1], Albert Reuther[1], Antonio Rosa[1], Gabriel Wachman[1], Charles Yee[1], Jeremy Kepner[1]

[1]MIT, [2]LBNL, [3]Texas A&M, [4]GreyNoise, [5]University of Tennessee

***Abstract*—The MIT/IEEE/Amazon GraphChallenge encourages community approaches to developing new solutions for analyzing graphs and sparse data derived from social media, sensor feeds, and scientific data to discover relationships between events as they unfold in the field. The anonymized network sensing Graph Challenge seeks to enable large, open, community-based approaches to protecting networks. Many large-scale networking problems can only be solved with community access to very broad data sets with the highest regard for privacy and strong community buy-in. Such approaches often require community-based data sharing. In the broader networking community (commercial, federal, and academia) anonymized source-to-destination traffic matrices with standard data sharing agreements have emerged as a data product that can meet many of these requirements. This challenge provides an opportunity to highlight novel approaches for optimizing the construction and analysis of anonymized traffic matrices using over 100 billion network packets derived from the largest Internet telescope in the world (CAIDA). This challenge specifies the anonymization, construction, and analysis of these traffic matrices. A GraphBLAS reference implementation is provided, but the use of GraphBLAS is not required in this Graph Challenge. As with prior Graph Challenges the goal is to provide a well-defined context for demonstrating innovation. Graph Challenge participants are free to select (with accompanying explanation) the Graph Challenge elements that are appropriate for highlighting their innovations.**

***Index Terms*—privacy preserving, Internet analysis, packet capture, streaming graphs, traffic matrices**

I. Introduction

The MIT/IEEE/Amazon GraphChallenge encourages community approaches to developing new solutions for analyzing graphs and sparse data. GraphChallenge.org provides a well-defined community venue for stimulating research and highlighting innovations in graph and sparse data analysis software, hardware, algorithms, and systems. The target audience for these challenges are any individual or team that seeks to highlight their contributions to graph and sparse data analysis software, hardware, algorithms, and/or systems. The Sparse DNN [1]–[9], Stochastic Block Partitioning [10]–[14], Subgraph Isomorphism [15]–[29], and PageRank [30]–[32] Graph Challenges have enabled a new generation of graph analysis by highlighting the benefits of novel innovations. Graph Challenge is part of the long tradition of challenges that have played a key role in advancing computation, AI, and other fields, such as, YOHO [33], MNIST [34], HPC Challenge [35], ImageNet [36] and VAST [37], [38]. More recently, major community activities, such as the NeurIPS conference and the MIT AI Accelerator [39], have prioritized the regular development of datasets, benchmarks, and challenges.

The Anonymized Network Sensing Graph Challenge seeks to enable large, open, community-based approaches to protecting networks [40]–[44]. Many large-scale networking problems can only be solved with community access to very broad data sets with the highest regard for privacy and strong community buy-in [45]–[47]. In the broader networking community (commercial, federal, and academia) anonymized source-to-destination traffic matrices with standard data sharing agreements have emerged as a data product that can meet many of these requirements (see Figure 1). This challenge provides an opportunity to highlight novel approaches for optimizing the construction and analysis of anonymized traffic matrices.

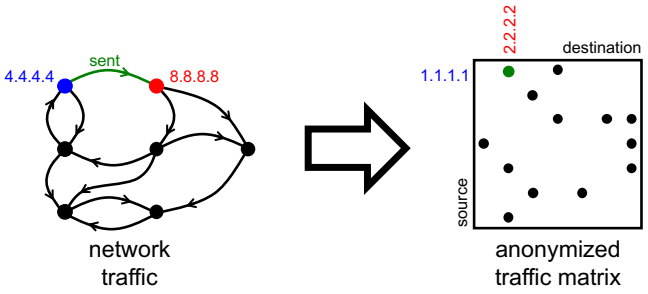

Fig. 1. **Network Traffic Messages to Anonymized Traffic Matrix.** Network traffic uses numbers to denote the source and destination addresses of messages. Network traffic messages can be aggregated and summarized into traffic matrices for analysis. These traffic matrices, when coupled with data sharing agreements, can be anonymized by relabeling source addresses (e.g., 4.4.4.4 → 1.1.1.1) and destination addresses (e.g., 8.8.8.8 → 2.2.2.2) using various anonymization schemes. The Anonymized Network Traffic Graph Challenge provides an opportunity to highlight novel approaches for optimizing the construction and analysis of anonymized traffic matrices from network traffic.

Research was sponsored by the Department of the Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

A GraphBLAS reference implementation is provided, but the use of GraphBLAS is not required in this Graph Challenge. GraphBLAS anonymized hypersparse traffic matrices represent one set of design choices for analyzing network traffic [3], [48]–[60]. Specifically, the use cases 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 [61]–[71]. Likewise, there are many network analysis tools that focus on providing a rich interface to the full diversity of data found in network traffic [72]–[74]. Each of these technologies has appropriate use cases in the broad field of Internet traffic analysis.

The outline of the rest of the paper is as follows. First, some basic network quantities are defined in terms of traffic matrices. Second, the steps of the Anonymized Network Sensing Graph Challenge and computational metrics are described. Next, the test data sets both real and random are presented. Finally, some preliminary performance results of the reference implementation are provided.

II. Anonymized Network Traffic Matrices

Network data must be handled with care. The Center for Applied Internet Data Analysis (CAIDA) based at the University of California's San Diego Supercomputer Center has pioneered trusted data sharing best practices that combine anonymizing source and destination internet addresses using CryptoPAN [75] with data sharing agreements. These data sharing best practices include the following principles [45].

- 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.

Network traffic data can be viewed as a traffic matrix where each row is a source and each column is a destination (see Figure 1). A primary benefit of constructing anonymized traffic matrices is the efficient computation of a wide range of network quantities via matrix mathematics. Figure 2 illustrates essential quantities found in all streaming dynamic networks. These quantities are all computable from anonymized traffic matrices created from the source and destination addresses found in Internet packet headers [79]–[82]. It is common to filter the Internet Protocol (IP) 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 [78]. At a given time t, N_V consecutive valid packets are aggregated from the network traffic into a matrix $\mathbf{A}_t$, where $\mathbf{A}_t(i,j)$ is the number of valid packets between the source i and destination j. The sum of all the entries in $\mathbf{A}_t$ is equal to N_V:

$$\sum_{i,j} \mathbf{A}_t(i,j) = N_V$$

Constant packet, variable time samples simplify the statistical analysis of the heavy-tail distributions commonly found in network traffic quantities [76], [83], [84]. All the network quantities depicted in Figure 2 can be readily computed from $\mathbf{A}_t$ 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

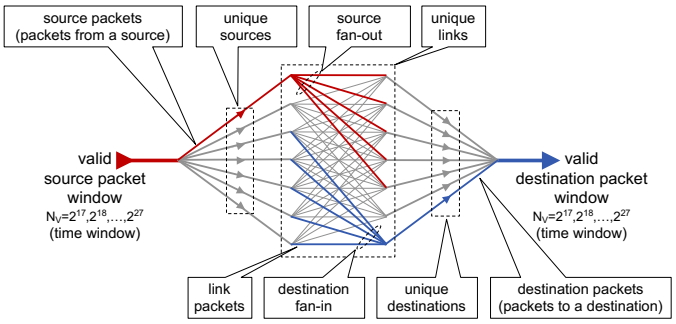

Fig. 2. **Streaming Network Traffic Quantities.** Internet traffic streams of N_V valid packets are divided into a variety of quantities for analysis: source packets, source fan-out, unique source-destination pair packets (or links), destination fan-in, and destination packets. Figure adapted from [76].

TABLE I
Network Quantities from Traffic Matrices

Formulas for computing network quantities from a traffic matrix $\mathbf{A}_t$ at time t in both summation and matrix notation. $\mathbf{1}$ is a column vector of all 1's, $^\mathsf{T}$ is the transpose operation, and $|\ |_0$ is the zero-norm that sets each nonzero value of its argument to 1 [77]. These formulas are unaffected by matrix permutations and work on anonymized data. Underlined quantities are those specified in the anonymized network sensing Graph Challenge. Table adapted from [78].

Aggregate Property	**Summation Notation**	**Matrix Notation**
Valid packets N_V	$\sum_i \sum_j \mathbf{A}_t(i,j)$	$\mathbf{1}^\mathsf{T}\mathbf{A}_t\mathbf{1}$
Unique links	$\sum_i \sum_j \lvert\mathbf{A}_t(i,j)\rvert_0$	$\mathbf{1}^\mathsf{T}\lvert\mathbf{A}_t\rvert_0\mathbf{1}$
Link packets from i to j	$\mathbf{A}_t(i,j)$	$\mathbf{A}_t$
Max link packets ($d_{\max}$)	$\max_{ij} \mathbf{A}_t(i,j)$	$\max(\mathbf{A}_t)$
Unique sources	$\sum_i \lvert\sum_j \mathbf{A}_t(i,j)\rvert_0$	$\mathbf{1}^\mathsf{T}\lvert\mathbf{A}_t\mathbf{1}\rvert_0$
Packets from source i	$\sum_j \mathbf{A}_t(i,j)$	$\mathbf{A}_t\mathbf{1}$
Max source packets ($d_{\max}$)	$\max_i \sum_j \mathbf{A}_t(i,j)$	$\max(\mathbf{A}_t\mathbf{1})$
Source fan-out from i	$\sum_j \lvert\mathbf{A}_t(i,j)\rvert_0$	$\lvert\mathbf{A}_t\rvert_0\mathbf{1}$
Max source fan-out ($d_{\max}$)	$\max_i \sum_j \lvert\mathbf{A}_t(i,j)\rvert_0$	$\max(\lvert\mathbf{A}_t\rvert_0\mathbf{1})$
Unique destinations	$\sum_j \lvert\sum_i \mathbf{A}_t(i,j)\rvert_0$	$\lvert\mathbf{1}^\mathsf{T}\mathbf{A}_t\rvert_0\mathbf{1}$
Destination packets to j	$\sum_i \mathbf{A}_t(i,j)$	$\mathbf{1}^\mathsf{T}\lvert\mathbf{A}_t\rvert_0$
Max destination packets ($d_{\max}$)	$\max_j \sum_i \mathbf{A}_t(i,j)$	$\max(\mathbf{1}^\mathsf{T}\lvert\mathbf{A}_t\rvert_0)$
Destination fan-in to j	$\sum_i \lvert\mathbf{A}_t(i,j)\rvert_0$	$\mathbf{1}^\mathsf{T}\ \mathbf{A}_t$
Max destination fan-in ($d_{\max}$)	$\max_j \sum_i \lvert\mathbf{A}_t(i,j)\rvert_0$	$\max(\mathbf{1}^\mathsf{T}\ \mathbf{A}_t)$

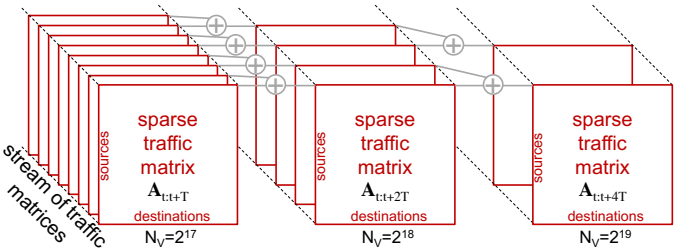

Fig. 3. **Binary Summation of Traffic Matrices.** Summing traffic matrices as binary pairs can result in more efficient memory access and more efficient analysis of matrices at each intermediate level. Figure adapted from [78].

from anonymized data. Furthermore, the anonymized data can be analyzed by source and destination subranges (subsets when anonymized) using simple matrix multiplication. For a given subrange represented by an anonymized diagonal matrix $\mathbf{A}_r$, where $\mathbf{A}_r(i,i) = 1$ implies source/destination i is in the range, the traffic within the subrange can be computed via: $\mathbf{A}_r\mathbf{A}_t\mathbf{A}_r$. Likewise, for additional privacy guarantees that can be implemented at collection, the same method can be used to exclude a range of data from the traffic matrix:

$$\mathbf{A}_t - \mathbf{A}_r\mathbf{A}_t\mathbf{A}_r$$

Efficient computation of network quantities on multiple time scales can be achieved by hierarchically aggregating data in different time windows [78]. Figure 3 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.

III. The Graph Challenge

This challenge provides an opportunity to highlight novel approaches for optimizing the construction and analysis of anonymized traffic matrices. This paper describes the anonymization, construction, and analysis of these traffic matrices. The overall steps of the challenge are depicted in Figure 4. The Anonymized Network Traffic Graph Challenge consist of several timed steps, each of which can be important to optimize in a real system. The complete process for performing the challenge consists of the following steps

1) Timed: Read/stream each network packet capture (PCAP) file containing 2^{30} packets.
2) Timed: Extract the source IP address and destination IP address from each packet header.
3) Timed: Anonymize the source IP and destination IP. Anonymization should be consistent over all files so all the data can be meaningfully further aggregated. Assume that any pair in the $2^{32}\times2^{32}$ IPv4 traffic space is possible. Novel approaches that also handle 128-bit IPv6 are encouraged. Anonymization can be done at different levels as long as it is explicitly stated: [trivial] no anonymization, [reference implementation] trusted sharing employing anonymization (e.g., CryptoPAN) that assumes the existence of an agreement prohibiting deanonymization, [advanced research] semantically secure anonymization.
4) Timed: Construct sequential traffic matrices with $N_V = 2^{17}$ packets (this size is large enough for meaningful statistics but small enough to preserve enough temporal information for statistical analysis given that Internet packets can arrive in any order). Matrices should be aligned with the mathematical definition of a matrix and can be read directly into an available matrix analysis environment. Avoid internal redundancy and store each (i,j) pair only once. Valid matrix formats include, but are not limited to, compressed sparse rows (CSR), compressed sparse columns (CSC), and sorted triples. Proprietary binary formats are allowed.
5) Timed: Save the traffic matrices to files. Valid file formants include, but are not limited to, comma separated values (CSV), tab separated values (TSV), SuiteSparse GraphBLAS [reference implementation], HDF, CDF, and NetCDF. The number of files and number of traffic matrices per file is up to the implementor and range from 2^{13} files each containing 1 traffic matrix to 1 file with 2^{13} traffic matrices. The reference implementation saves 2^7 .tar files each containing 2^6 SuiteSparse GraphBLAS traffic matrices.
6) Timed: Read in the 2^{13} traffic matrices associated with 2^{30} packets, sum the traffic matrices into a single large traffic matrix $\mathbf{A}_t$ (see Figure 3), and perform the analysis highlighted in Table I.

Reference serial implementations in various programming languages are available at GraphChallenge.org. The pseudocode for the reference implementation is shown in Figure 5. For a given implementation of the Graph Challenge an implementor should keep the following guidance in mind.

Do

- Use an implementation that could work on real-world data.
- Distribute inputs and run in data parallel mode to achieve higher performance (this may require storing traffic matrices on every processor and increase the memory footprint).
- Split up steps and run in a pipeline parallel mode to achieve higher performance (this saves memory, but requires communicating results after each group of steps).
- Use other reasonable optimizations that would work on real-world data.

Avoid

- Using optimizations that would not work on real-world data.
- Unnecessarily pre-computing quantities for a subsequent step in a previous step.

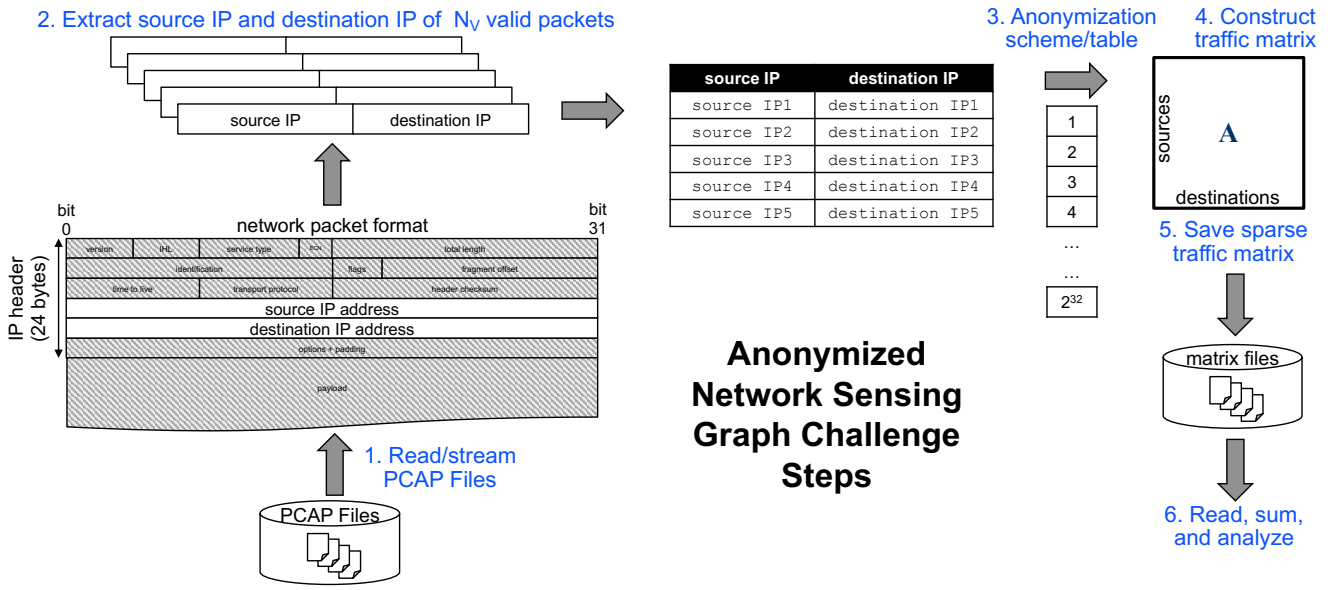

Fig. 4. **Anonymized Network Traffic Graph Challenge Steps**. (1) Read/stream each network packet capture (PCAP) file. (2) Extract the source IP and destination IP addresses from the packet headers and buffer N_V valid packets. (3) Anonymize the source IP and destination IP. Anonymization should be consistent over all files so all the data can be meaningfully further aggregated. Assume that any pair in the $2^{32} \times 2^{32}$ IPv4 traffic space is possible. (4) Construct sequential traffic matrices from N_V valid packets. Matrices should align with the mathematical definition of a matrix. (5) Save the traffic matrices to files (granularity is up to the implementor). (6) Read in the traffic matrix files, sum the traffic matrices associated with a PCAP file into one large traffic matrix $\mathbf{A}_t$ (see Figure 3), and perform the analysis highlighted in Table I.

IV. Computational Metrics

Submissions to the Anonymized Network Sensing Graph Challenge will be evaluated on the overall innovations highlighted by the implementation and two metrics: correctness and performance.

A. Correctness

Correctness is evaluated by comparing the reported Table I quantities for each 2^{30} packet PCAP file with the ground truth provided.

B. Performance

The performance of the algorithm implementation should be reported in terms of the following metrics.

- Total number of packets processed: this measures the amount of data processed.
- Execution time: total time required to perform the Graph Challenge.
- Rate: measures the throughput of the implementation as the ratio of the number of packets processed divided by the execution time.
- Processor: number and type of processors used in the computation.

Graph Challenge participants are free to select (with accompanying explanation) the elements of any of the Graph Challenges that are appropriate for highlighting their innovations. Reporting the performance of individual steps in the Graph Challenge are encouraged. It is often the case that a particular innovation is focused on improving a single step.

V. Anonymized Test Data

The test data consists of two types (1) randomized and (2) anonymized real data derived the from the CAIDA telescope. The randomized data consists of a single freely available 2^{30} packet PCAP file with source and destination IP addresses generated with 2×2^{30} calls of the C PCG32 (Permuted Congruential Generator 32 bit) pseudo random number generator function [85]. The CAIDA darknet telescope is a significant portion of a globally routed /8 network carrying essentially no legitimate traffic since it is an Internet darkspace, providing an ideal vantage point by which to observe and study unsolicited anomalous traffic. The CAIDA network traffic is collected and anonymized into traffic matrices in a process similar to steps 1-5 shown in Figure 4 [86]. Subsets of these traffic matrices representing 2^{30} contiguous packets were selected around noon and midnight from many days around the first quarter of 2022 (see Table II) [87]. These traffic matrices were then converted back into 2^{30} packet PCAP files. For both the random and CAIDA data, the other fields in the PCAP header are populated using the values or methods shown in Figure 6.

An additional enrichment data set is also included that looks up sources found in the CAIDA telescope data in the GreyNoise honeyfarm database [87]–[89]. The GreyNoise honeyfarm is made up of thousands of servers carrying out conversations with sources scanning the Internet; based on these conversations GreyNoise can associate various metadata with those sources to collectively build a refined picture of the malicious sources regularly scanning the Internet and the techniques they employ. The GreyNoise enrichment of CAIDA data uses anonymized IP addresses throughout and is provided

```
AnonNetSensingGraphChallenge(
  PCAPfile, # name of PCAP file
  Np, # packets in file (2^30)
  Nv, # packets per matrix (2^17)
  NmatPerFile, # matrices per output file (2^6)
  anonKey # anonymization key
);
PCAPbuffer = read(PCAPfile);
p = 0;
for i = 0 to (Np/(NmatPerFile*Nv))-1 # (2^7 - 1)
  for j = 0 to NmatPerFile-1
    for k = 0 to Nv-1
      [srcIP(k) destIP(k)] = readPCAPheader(PCAPbuffer,p);
      srcIPanon(k) = anonymize(srcIP,anonKey);
      destIPanon(k) = anonymize(destIP,anonKey);
      p++;
    end
    A[j] = constructMatrix(srcIPanon,destIPanon);
  end
saveMatrices(A,i);
end
A_t(:,:) = 0;
for i = 0 to (Np/(NmatPerFile*Nv))-1
  A = readMatrices(i);
  for j = 0 to NmatPerFile-1
    A_t += A[j];
  end
end
# perform the analysis on A_t listed in Table I
end
```

Fig. 5. **Anonymized Network Traffic Graph Challenge Pseudocode**. Code begins by reading a 2^{30} packet PCAP file in groups of $N_V = 2^{17}$ valid packets. The source and destination IP addresses of the packets are anonymized and then used to populate the entries of a traffic matrix $\mathbf{A}[j]$. 2^6 of these traffic matrices are then saved as individual files within a .tar file. 2^7 .tar files are saved per PCAP files. After all the traffic matrices are constructed and saved, the .tar files are then read again and all the traffic matrices are summed into one traffic matrix $\mathbf{A}_t$. The analysis highlighted in Table I are then performed on $\mathbf{A}_t$ and reported.

TABLE II
ANONYMIZED DATA SETS

Characteristics of CAIDA Telescope derived anonymized PCAP files and corresponding GreyNoise enrichment data.

Month	CAIDA 2^{30} packet sets	CAIDA data size compressed	GreyNoise data size
2022-01	25 (noon); 24 (midnight)	375 GB	3.6 GB
2022-02	17 (noon); 18 (midnight)	271 GB	3.6 GB
2022-03	24 (noon); 26 (midnight)	432 GB	3.6 GB
2022-04	13 (noon); 14 (midnight)	242 GB	

as a point of departure for future investigations (see [87], [88] for details on the enriched fields provided).

VI. PERFORMANCE MEASUREMENTS

Performance measurements of the reference C traffic matrix constructor code and the Python traffic matrix sum and analysis code are shown in Figure 7 and provide one example for reporting results. Parallel implementations of the reference code were developed and tested using dual 2.4

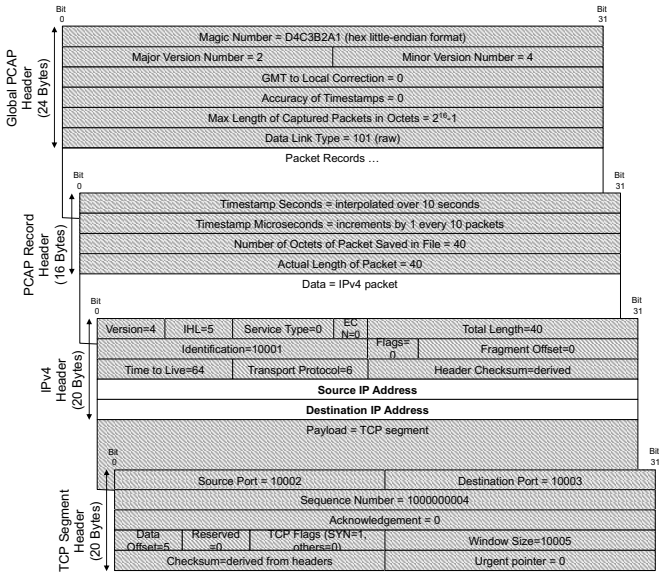

Fig. 6. **Packet Capture (PCAP) File Format.** Each of 2^{30} packets are stored in a PCAP file in the above format. PCAP files begin with a global header followed by a sequence of PCAP records. Each PCAP record consists of a header and data containing an Internet Protocol version 4 (IPv4) packet. An IPv4 packet consists of a header containing the source address and the destination address fields. For completeness, the IPv4 payload contains a Transmission Control Protocol (TCP) segment header. The specific values or method used to populate the fields are listed in the figure. [Note: setting data link type to 101 indicates this a raw data and there is no Ethernet frame header between the PCAP record header and the IPv4 header.]

GHz Intel Xeon Platinum 8260 processor compute nodes on the MIT SuperCloud TX-Green supercomputer [90]. These results demonstrate that traffic matrix construction can be done in a streaming fashion with modest memory enabling large numbers of PCAP files to be processed simultaneously on a single compute node. Reading the PCAP files can take significant time as shown by the rate increase achieved by caching the files in memory. Likewise, in-line anonymization has the opposite effect. Prior work shows that anonymization time can be effectively eliminated by using look-up tables [58]. Sum and analysis of the traffic matrices requires a larger memory footprint which can be accelerated with threads. In both cases, multiple files can be processed simultaneously and the performance scales linearly with nodes.

VII. CONCLUSIONS

The anonymized network sensing Graph Challenge seeks to enable large, open, community-based approaches to protecting networks. Community access to very broad data sets with the highest regard for privacy is essential for solving many large-scale networking problems. Anonymized source-to-destination traffic matrices with standard data sharing agreements have emerged in the broader networking community as a data product that can meet many of these requirements. Using over 100 billion network packets derived from the largest Internet telescope in the world (CAIDA) the anonymized network sensing Graph Challenge provides an opportunity to highlight

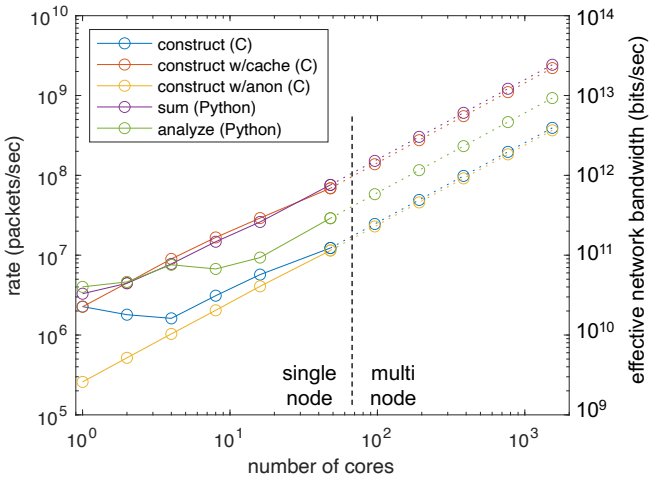

Fig. 7. **Reference Performance.** Average performance measurements of the reference C traffic matrix constructor code (with anonymization and with file caching) and the Python traffic matrix sum and analyze code. Effective bandwidth is computed assuming 10,000 bits/packet for real packet data on a real network. The constructor code has a small memory footprint and 48 distinct instances each processing a separate PCAP file can be run on a 48 core node. The sum and analyze code have a larger memory footprint and 3 distinct instances each with 16 OpenMP threads each processing a separate PCAP file can be run on a 48 core node with 192 GB of RAM. The multi-node performance scales linearly over distinct PCAP files.

novel approaches for optimizing the construction and analysis of anonymized traffic matrices. A GraphBLAS reference implementation is provided, but the use of GraphBLAS is not required in this Graph Challenge. As with prior Graph Challenges the goal is to provide a well-defined context for demonstrating innovation. Graph Challenge participants are free to select (with accompanying explanation) the elements of any of the Graph Challenges that are appropriate for highlighting their innovations.

ACKNOWLEDGMENTS

The authors wish to acknowledge the following individuals for their contributions and support: Daniel Andersen, LaToya Anderson, Sean Atkins, David Bader, Chris Birardi, Bob Bond, Alex Bonn, Koley Borchard, Stephen Buckley, Aydin Buluc, K Claffy, Cary Conrad, Chris Demchak, Phil Dykstra, Alan Edelman, Peter Fisher, Garry Floyd, Jeff Gottschalk, Dhruv Gupta, Oded Green, Thomas Hardjono, Chris Hill, Miriam Leeser, Charles Leiserson, Chris Long, Kirsten Malvey, Sanjeev Mohindra, Roger Pearce, Heidi Perry, Ali Pinar, Christian Prothmann, Steve Rejto, Josh Rountree, Daniela Rus, Siddharth Samsi, Mark Sherman, Scott Weed, Michael Wright, Marc Zissman.

REFERENCES

[1] J. Kepner, S. Alford, V. Gadepally, M. Jones, L. Milechin, R. Robinett, and S. Samsi, "Sparse deep neural network graph challenge," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2019.

[2] M. Bisson and M. Fatica, "A gpu implementation of the sparse deep neural network graph challenge," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–8, 2019.

[3] T. A. Davis, M. Aznaveh, and S. Kolodziej, "Write quick, run fast: Sparse deep neural network in 20 minutes of development time via suitesparse:graphblas," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–6, 2019.

[4] D.-L. Lin and T.-W. Huang, "A novel inference algorithm for large sparse neural network using task graph parallelism," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2020.

[5] M. Hidayetoglu, C. Pearson, V. S. Mailthody, E. Ebrahimi, J. Xiong, R. Nagi, and W.-m. Hwu, "At-scale sparse deep neural network inference with efficient gpu implementation," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2020.

[6] J. Xin, X. Ye, L. Zheng, Q. Wang, Y. Huang, P. Yao, L. Yu, X. Liao, and H. Jin, "Fast sparse deep neural network inference with flexible spmm optimization space exploration," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2021.

[7] Y. Sun, L. Zheng, Q. Wang, X. Ye, Y. Huang, P. Yao, X. Liao, and H. Jin, "Accelerating sparse deep neural network inference using gpu tensor cores," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2022.

[8] S. Xu, M. Wu, L. Zheng, Z. Shao, X. Ye, X. Liao, and H. Jin, "Towards fast gpu-based sparse dnn inference: A hybrid compute model," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2022.

[9] M. Dun, X. Zhang, H. Cao, Y. Zhang, J. Huang, and X. Ye, "Adaptive sparse deep neural network inference on resource-constrained cost-efficient gpus," in *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2023.

[10] E. Kao, V. Gadepally, M. Hurley, M. Jones, J. Kepner, S. Mohindra, P. Monticciolo, A. Reuther, S. Samsi, W. Song, D. Staheli, and S. Smith, "Streaming graph challenge: Stochastic block partition," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–12, 2017.

[11] M. Halappanavar, H. Lu, A. Kalyanaraman, and A. Tumeo, "Scalable static and dynamic community detection using grappolo," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–6, 2017.

[12] B. W. Priest, A. Dunton, and G. Sanders, "Scaling graph clustering with distributed sketches," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2020.

[13] A. J. Uppal, J. Choi, T. B. Rolinger, and H. Howie Huang, "Faster stochastic block partition using aggressive initial merging, compressed representation, and parallelism control," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2021.

[14] F. Wanye, V. Gleyzer, E. Kao, and W.-c. Feng, "An integrated approach for accelerating stochastic block partitioning," in *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2023.

[15] S. Samsi, V. Gadepally, M. Hurley, M. Jones, E. Kao, S. Mohindra, P. Monticciolo, A. Reuther, S. Smith, W. Song, D. Staheli, and J. Kepner, "Static graph challenge: Subgraph isomorphism," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–6, 2017.

[16] M. M. Wolf, M. Deveci, J. W. Berry, S. D. Hammond, and S. Rajamanickam, "Fast linear algebra-based triangle counting with kokkoskernels," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2017.

[17] R. Pearce, "Triangle counting for scale-free graphs at scale in distributed memory," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–4, 2017.

[18] C. Voegele, Y.-S. Lu, S. Pai, and K. Pingali, "Parallel triangle counting and k-truss identification using graph-centric methods," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2017.

[19] M. Bisson and M. Fatica, "Static graph challenge on gpu," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–8, 2017.

[20] Y. Hu, H. Liu, and H. H. Huang, "High-performance triangle counting on gpus," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pp. 1–5, 2018.

[21] M. Bisson and M. Fatica, "Update on static graph challenge on gpu," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pp. 1–8, 2018.

[22] A. Yasar, S. Rajamanickam, M. Wolf, J. Berry, and U. V. Çatalyurek, "Fast triangle counting using cilk," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pp. 1–7, 2018.

[23] R. Pearce and G. Sanders, "K-truss decomposition for scale-free graphs at scale in distributed memory," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pp. 1–6, 2018.

[24] S. Pandey, X. S. Li, A. Buluc, J. Xu, and H. Liu, "H-index: Hash-indexing for parallel triangle counting on gpus," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2019.

[25] M. P. Blanco, T. M. Low, and K. Kim, "Exploration of fine-grained parallelism for load balancing eager k-truss on gpu and cpu," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2019.

[26] R. Pearce, T. Steil, B. W. Priest, and G. Sanders, "One quadrillion triangles queried on one million processors," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–5, 2019.

[27] S. Samsi, J. Kepner, V. Gadepally, M. Hurley, M. Jones, E. Kao, S. Mohindra, A. Reuther, S. Smith, W. Song, D. Staheli, and P. Monticciolo, "Graphchallenge.org triangle counting performance," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–9, 2020.

[28] S. Ghosh and M. Halappanavar, "Tric: Distributed-memory triangle counting by exploiting the graph structure," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–6, 2020.

[29] Z. Wang, Z. Meng, X. Li, X. Lin, L. Zheng, C. Tian, and S. Zhong, "Smog: Accelerating subgraph matching on gpus," in *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2023.

[30] P. Dreher, C. Byun, C. Hill, V. Gadepally, B. Kuszmaul, and J. Kepner, "Pagerank pipeline benchmark: Proposal for a holistic system benchmark for big-data platforms," in *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 929–937, 2016.

[31] S. Zhou, K. Lakhotia, S. G. Singapura, H. Zeng, R. Kannan, V. K. Prasanna, J. Fox, E. Kim, O. Green, and D. A. Bader, "Design and implementation of parallel pagerank on multicore platforms," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–6, 2017.

[32] F. Sadi, J. Sweeney, S. McMillan, T. M. Low, J. C. Hoe, L. Pileggi, and F. Franchetti, "Pagerank acceleration for large graphs with scalable hardware and two-step spmv," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pp. 1–7, 2018.

[33] J. P. Campbell, "Testing with the yoho cd-rom voice verification corpus," in *1995 International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 341–344 vol.1, May 1995.

[34] C. C. Y. LeCun and C. J. Burges, "The MNIST Database." http://www.hpcchallenge.org, 2017. [Online; accessed 01-January-2017].

[35] J. Dongarra and P. Luszczek, "Hpc challenge: design, history, and implementation highlights," in *Contemporary High Performance Computing*, pp. 13–30, Chapman and Hall/CRC, 2017.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[37] K. A. Cook, G. Grinstein, and M. A. Whiting, "The VAST Challenge: History, Scope, and Outcomes: An introduction to the Special Issue," *Information Visualization, 13(4):301-312*, Oct 2014.

[38] J. Scholtz, M. A. Whiting, C. Plaisant, and G. Grinstein, "A Reflection on Seven Years of the VAST Challenge," in *Proceedings of the 2012 BELIV Workshop: Beyond Time and Errors - Novel Evaluation Methods for Visualization*, BELIV '12, pp. 13:1–13:8, ACM, 2012.

[39] V. Gadepally, G. Angelides, A. Barbu, A. Bowne, L. J. Brattain, T. Broderick, A. Cabrera, G. Carl, R. Carter, M. Cha, E. Cowen, J. Cummings, B. Freeman, J. Glass, S. Goldberg, M. Hamilton, T. Heldt, K. W. Huang, P. Isola, B. Katz, J. Koerner, Y.-C. Lin, D. Mayo, K. McAlpin, T. Perron, J. Piou, H. M. Rao, H. Reynolds, K. Samuel, S. Samsi, M. Schmidt, L. Shing, O. Simek, B. Swenson, V. Sze, J. Taylor, P. Tylkin, M. Veillette, M. L. Weiss, A. Wollaber, S. Yuditskaya, and J. Kepner, "Developing a series of ai challenges for the united states department of the air force," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2022.

[40] S. Atkins and C. Lawson, "An improvised patchwork: success and failure in cybersecurity policy for critical infrastructure," *Public Administration Review*, vol. 81, no. 5, pp. 847–861, 2021.

[41] S. Atkins and C. Lawson, "Cooperation amidst competition: cybersecurity partnership in the us financial services sector," *Journal of Cybersecurity*, vol. 7, no. 1, 2021.

[42] C. Demchak, "Achieving systemic resilience in a great systems conflict era," *The Cyber Defense Review*, vol. 6, no. 2, pp. 51–70, 2021.

[43] S. Weed, "Beyond zero trust: Reclaiming blue cyberspace," Master's thesis, United States Army War College, 2022.

[44] S. Atkins and C. Lawson, "Beyond zero trust: Reclaiming blue cyberspace with ai," *Cyber Defense Review*, vol. 7, no. 1, 2023.

[45] J. Kepner, J. Bernays, S. Buckley, K. Cho, C. Conrad, L. Daigle, K. Erhardt, V. Gadepally, B. Greene, M. Jones, R. Knake, B. Maggs, P. Michaleas, C. Meiners, A. Morris, A. Pentland, S. Pisharody, S. Powazek, A. Prout, P. Reiner, K. Suzuki, K. Takhashi, T. Tauber, L. Walker, and D. Stetson, "Zero botnets: An observe-pursue-counter approach." Belfer Center Reports, 6 2021.

[46] S. Pisharody, J. Bernays, V. Gadepally, M. Jones, J. Kepner, C. Meiners, P. Michaleas, A. Tse, and D. Stetson, "Realizing forward defense in the cyber domain," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, IEEE, 2021.

[47] A. Pentland, "Building a new economy: data, ai, and web3," *Communications of the ACM*, vol. 65, no. 12, pp. 27–29, 2022.

[48] J. Kepner, P. Aaltonen, D. Bader, A. Buluç, F. Franchetti, J. Gilbert, D. Hutchison, M. Kumar, A. Lumsdaine, H. Meyerhenke, S. McMillan, C. Yang, J. D. Owens, M. Zalewski, T. Mattson, and J. Moreira, "Mathematical foundations of the graphblas," in *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–9, 2016.

[49] A. Buluç, T. Mattson, S. McMillan, J. Moreira, and C. Yang, "Design of the graphblas api for c," in *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 643–652, 2017.

[50] C. Yang, A. Buluç, and J. D. Owens, "Implementing push-pull efficiently in graphblas," in *Proceedings of the 47th International Conference on Parallel Processing*, pp. 1–11, 2018.

[51] J. Kepner and H. Jananthan, *Mathematics of big data: Spreadsheets, databases, matrices, and graphs*. MIT Press, 2018.

[52] T. A. Davis, "Algorithm 1000: Suitesparse: Graphblas: Graph algorithms in the language of sparse linear algebra," *ACM Transactions on Mathematical Software (TOMS)*, vol. 45, no. 4, pp. 1–25, 2019.

[53] T. Mattson, T. A. Davis, M. Kumar, A. Buluc, S. McMillan, J. Moreira, and C. Yang, "Lagraph: A community effort to collect graph algorithms built on top of the graphblas," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 276–284, IEEE, 2019.

[54] P. Cailliau, T. Davis, V. Gadepally, J. Kepner, R. Lipman, J. Lovitz, and K. Ouaknine, "Redisgraph graphblas enabled graph database," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 285–286, IEEE, 2019.

[55] M. Aznaveh, J. Chen, T. A. Davis, B. Hegyi, S. P. Kolodziej, T. G. Mattson, and G. Szárnyas, "Parallel graphblas with openmp," in *2020 Proceedings of the SIAM Workshop on Combinatorial Scientific Computing*, pp. 138–148, SIAM, 2020.

[56] B. Brock, A. Buluç, T. G. Mattson, S. McMillan, and J. E. Moreira, "Introduction to graphblas 2.0," in *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 253–262, IEEE, 2021.

[57] M. Pelletier, W. Kimmerer, T. A. Davis, and T. G. Mattson, "The graphblas in julia and python: the pagerank and triangle centralities," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2021.

[58] M. Jones, J. Kepner, D. Andersen, A. Buluç, C. Byun, K. Claffy, T. Davis, W. Arcand, J. Bernays, D. Bestor, W. Bergeron, V. Gadepally, M. Houle, M. Hubbell, H. Jananthan, A. Klein, C. Meiners, L. Milechin, J. Mullen, S. Pisharody, A. Prout, A. Reuther, A. Rosa, S. Samsi, J. Sreekanth, D. Stetson, C. Yee, and P. Michaleas, "Graphblas on the edge: Anonymized high performance streaming of network traffic," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–8, 2022.

[59] T. Trigg, C. Meiners, S. Pisharody, H. Jananthan, M. Jones, A. Michaleas, T. Davis, E. Welch, W. Arcand, D. Bestor, W. Bergeron, C. Byun, V. Gadepally, M. Houle, M. Hubbell, A. Klein, P. Michaleas, L. Milechin, J. Mullen, A. Prout, A. Reuther, A. Rosa, S. Samsi, D. Stetson, C. Yee, and J. Kepner, "Hypersparse network flow analysis of packets with graphblas," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2022.

[60] T. A. Davis, "Algorithm 1037: Suitesparse: graphblas: Parallel graph algorithms in the language of sparse linear algebra," *ACM Transactions on Mathematical Software*, vol. 49, no. 3, pp. 1–30, 2023.

[61] A. Tumeo, O. Villa, and D. Sciuto, "Efficient pattern matching on gpus for intrusion detection systems," in *Proceedings of the 7th ACM International Conference on Computing Frontiers*, CF '10, (New York, NY, USA), p. 87–88, Association for Computing Machinery, 2010.

[62] M. Kumar, W. P. Horn, J. Kepner, J. E. Moreira, and P. Pattnaik, "Ibm power9 and cognitive computing," *IBM Journal of Research and Development*, vol. 62, no. 4/5, pp. 10–1, 2018.

[63] J. Ezick, T. Henretty, M. Baskaran, R. Lethin, J. Feo, T.-C. Tuan, C. Coley, L. Leonard, R. Agrawal, B. Parsons, and W. Glodek, "Combining tensor decompositions and graph analytics to provide cyber situational awareness at hpc scale," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2019.

[64] P. Gera, H. Kim, P. Sao, H. Kim, and D. Bader, "Traversing large graphs on gpus with unified memory," *Proceedings of the VLDB Endowment*, vol. 13, no. 7, pp. 1119–1133, 2020.

[65] A. Azad, M. M. Aznaveh, S. Beamer, M. Blanco, J. Chen, L. D'Alessandro, R. Dathathri, T. Davis, K. Deweese, J. Firoz, H. A. Gabb, G. Gill, B. Hegyi, S. Kolodziej, T. M. Low, A. Lumsdaine, T. Manlaibaatar, T. G. Mattson, S. McMillan, R. Peri, K. Pingali, U. Sridhar, G. Szarnyas, Y. Zhang, and Y. Zhang, "Evaluation of graph analytics frameworks using the gap benchmark suite," in *2020 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 216–227, 2020.

[66] Z. Du, O. A. Rodriguez, J. Patchett, and D. A. Bader, "Interactive graph stream analytics in arkouda," *Algorithms*, vol. 14, no. 8, p. 221, 2021.

[67] S. Acer, A. Azad, E. G. Boman, A. Buluç, K. D. Devine, S. Ferdous, N. Gawande, S. Ghosh, M. Halappanavar, A. Kalyanaraman, A. Khan, M. Minutoli, A. Pothen, S. Rajamanickam, O. Selvitopi, N. R. Tallent, and A. Tumeo, "Exagraph: Graph and combinatorial methods for enabling exascale applications," *The International Journal of High Performance Computing Applications*, vol. 35, no. 6, pp. 553–571, 2021.

[68] M. P. Blanco, S. McMillan, and T. M. Low, "Delayed asynchronous iterative graph algorithms," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, IEEE, 2021.

[69] N. K. Ahmed, N. Duffield, and R. A. Rossi, "Online sampling of temporal networks," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 4, pp. 1–27, 2021.

[70] A. Azad, O. Selvitopi, M. T. Hussain, J. R. Gilbert, and A. Buluç, "Combinatorial blas 2.0: Scaling combinatorial algorithms on distributed-memory systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 989–1001, 2021.

[71] D. Koutra, "The power of summarization in graph mining and learning: smaller data, faster methods, more interpretability," *Proceedings of the VLDB Endowment*, vol. 14, no. 13, pp. 3416–3416, 2021.

[72] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[73] R. Sommer, "Bro: An open source network intrusion detection system," *Security, E-learning, E-Services, 17. DFN-Arbeitstagung über Kommunikationsnetze*, 2003.

[74] P. Lucente, "pmacct: steps forward interface counters," *Tech. Rep.*, 2008.

[75] J. Fan, J. Xu, M. H. Ammar, and S. B. Moon, "Prefix-preserving ip address anonymization: measurement-based security evaluation and a new cryptography-based scheme," *Computer Networks*, vol. 46, no. 2, pp. 253–272, 2004.

[76] J. Kepner, K. Cho, K. Claffy, V. Gadepally, P. Michaleas, and L. Milechin, "Hypersparse neural network analysis of large-scale internet traffic," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–11, 2019.

[77] J. Karvanen and A. Cichocki, "Measuring sparseness of noisy signals," in *4th International Symposium on Independent Component Analysis and Blind Signal Separation*, pp. 125–130, 2003.

[78] J. Kepner, C. Meiners, C. Byun, S. McGuire, T. Davis, W. Arcand, J. Bernays, D. Bestor, W. Bergeron, V. Gadepally, R. Harnasch, M. Hubbell, M. Houle, M. Jones, A. Kirby, A. Klein, L. Milechin, J. Mullen, A. Prout, A. Reuther, A. Rosa, S. Samsi, D. Stetson, A. Tse, C. Yee, and P. Michaleas, "Multi-temporal analysis and scaling relations of 100,000,000,000 network packets," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–6, 2020.

[79] A. Soule, A. Nucci, R. Cruz, E. Leonardi, and N. Taft, "How to identify and estimate the largest traffic matrix elements in a dynamic environment," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 32, pp. 73–84, ACM, 2004.

[80] Y. Zhang, M. Roughan, C. Lund, and D. L. Donoho, "Estimating point-to-point and point-to-multipoint traffic matrices: an information-theoretic approach," *IEEE/ACM Transactions on Networking (TON)*, vol. 13, no. 5, pp. 947–960, 2005.

[81] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks," *science*, vol. 328, no. 5980, pp. 876–878, 2010.

[82] P. Tune, M. Roughan, H. Haddadi, and O. Bonaventure, "Internet traffic matrices: A primer," *Recent Advances in Networking*, vol. 1, pp. 1–56, 2013.

[83] J. Nair, A. Wierman, and B. Zwart, "The fundamentals of heavy tails: Properties, emergence, and estimation," *Preprint, California Institute of Technology*, 2020.

[84] J. Kepner, K. Cho, K. Claffy, V. Gadepally, S. McGuire, L. Milechin, W. Arcand, D. Bestor, W. Bergeron, C. Byun, M. Hubbell, M. Houle, M. Jones, A. Prout, A. Reuther, A. Rosa, S. Samsi, C. Yee, and P. Michaleas, "New phenomena in large-scale internet traffic," in *Massive Graph Analytics* (D. Bader, ed.), pp. 1–53, Chapman and Hall/CRC, 2022.

[85] M. E. O'Neill, "Pcg: A family of simple fast space-efficient statistically good algorithms for random number generation," Tech. Rep. HMC-CS-2014-0905, Harvey Mudd College, Claremont, CA, Sept. 2014.

[86] M. Jones, J. Kepner, A. Prout, T. Davis, W. Arcand, D. Bestor, W. Bergeron, C. Byun, V. Gadepally, M. Houle, M. Hubbell, H. Jananthan, A. Klein, L. Milechin, G. Morales, J. Mullen, R. Patel, S. Pisharody, A. Reuther, A. Rosa, S. Samsi, C. Yee, and P. Michaleas, "Deployment of real-time network traffic analysis using graphblas hypersparse matrices and d4m associative arrays," in *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–8, 2023.

[87] H. Jananthan, J. Kepner, M. Jones, W. Arcand, D. Bestor, W. Bergeron, C. Byun, T. Davis, V. Gadepally, D. Grant, M. Houle, M. Hubbell, A. Klein, L. Milechin, G. Morales, A. Morris, J. Mullen, R. Patel, A. Pentl, S. Pisharody, A. Prout, A. Reuther, A. Rosa, S. Samsi, T. Trigg, G. Wachman, C. Yee, and P. Michaleas, "Mapping of internet "coastlines" via large scale anonymized network source correlations," in *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–9, 2023.

[88] I. Kawaminami, A. Estrada, Y. Elsakkary, H. Jananthan, A. Buluç, T. Davis, D. Grant, M. Jones, C. Meiners, A. Morris, S. Pisharody, and J. Kepner, "Large scale enrichment and statistical cyber characterization of network traffic," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2022.

[89] "*Greynoise*." https://greynoise.io/.

[90] A. Reuther, J. Kepner, C. Byun, S. Samsi, W. Arcand, D. Bestor, B. Bergeron, V. Gadepally, M. Houle, M. Hubbell, M. Jones, A. Klein, L. Milechin, J. Mullen, A. Prout, A. Rosa, C. Yee, and P. Michaleas, "Interactive supercomputing on 40,000 cores for machine learning and data analysis," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pp. 1–6, 2018.