GraphBLAS on the Edge: High Performance Streaming of Network Traffic

Michael Jones\textsuperscript{1}, Jeremy Kepner\textsuperscript{1}, Daniel Andersen\textsuperscript{2}, Aydn Buluc\textsuperscript{3}, Chansup Byun\textsuperscript{1}, K Claffy\textsuperscript{2}, Timothy Davis\textsuperscript{4}, William Arcand\textsuperscript{5}, Jonathan Bernays\textsuperscript{1}, David Bestor\textsuperscript{1}, William Bergeron\textsuperscript{1}, Vijay Gadepally\textsuperscript{1}, Micheal Houle\textsuperscript{1}, Matthew Hubbell\textsuperscript{1}, Hayden Jananthan\textsuperscript{1}, Anna Klein\textsuperscript{1}, Chad Meiners\textsuperscript{1}, Lauren Milechin\textsuperscript{1}, Sandeep Pisharody\textsuperscript{1}, Andrew Prout\textsuperscript{1}, Albert Reuther\textsuperscript{1}, Antonio Rosa\textsuperscript{1}, Siddharth Samsi\textsuperscript{1}, Jon Sreekantan\textsuperscript{5}, Doug Stetson\textsuperscript{1}, Charles Yee\textsuperscript{1}, Peter Michaleas\textsuperscript{1}

\textsuperscript{1}MIT, \textsuperscript{2}CAIDA, \textsuperscript{3}LBNL, \textsuperscript{4}Texas A&M, \textsuperscript{5}Accolade 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, air, 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 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

Fig. 1. **Internet Observatories and Outposts.** Traffic matrix view of the Internet depicting selected observatories and outposts and their notional proximity to various types of network traffic [11]–[17].

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.
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 [59]–[62]. 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 [63]. 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], [63]. 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.

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 [63]. 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].

### Table I

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

![Fig. 4. Multi-temporal streaming traffic matrices. Efficient computation of network quantities on multiple time scales can be achieved by hierarchically aggregating data in different time windows.](image-url)
For example, standard gzip compression reduces file size by 40% but also reduces performance by 80%. 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 $2^{12} \times 2^{12}$ 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 then transmitted to the appropriate parties tasked for analysis.

For example, standard gzip compression reduces file size by 40% but also reduces performance by 80%.

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 $2^{12} \times 2^{12}$ 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 then transmitted to the appropriate parties tasked for analysis.

For example, standard gzip compression reduces file size by 40% but also reduces performance by 80%.

For example, standard gzip compression reduces file size by 40% but also reduces performance by 80%.
IV. IMPLEMENTATION

Effective implementation of the GraphBLAS pipeline requires that the anonymization, creation, and saving of the resulting files can keep up with the data rates of typical high bandwidth links. To measure this performance two Accolade Technology ANIC-200Kq dual port 100 gigabit flow classification and shunting adapters were installed into the PCIe slots of two HP ProLiant DL360 G9 servers (Figure 7). These servers were connected via a Mellanox 40 gigabit network connection. ANIC-200Kq cards are capable of a wide range of analysis techniques, this experiment only used their data transmission, shunting, and buffering capabilities.

The C implementation of the GraphBLAS hypersparse traffic matrix pipeline shown in Figures 5 and 6 was run on dual Intel Xeon E5-2683 processors in the receiver server. The Accolade card on the receiver server collects packets in ring buffers, the number of which can be set at initialization. Using C pthreads [70] an Accolade worker thread is assigned to each Accolade hardware ring. Within each Accolade worker thread, a block of $2^{23}$ IPv4 packets are collected and a GraphBLAS worker thread is spawned to process the block in subblocks of $2^{17}$ packets. Each subblock is anonymized using a cryptoPAN generated anonymization vector and the resulting anonymized sources and destinations are used to construct a GraphBLAS matrix. The matrix is the serialized and appended to a TAR buffer, which is saved to a file after all 64 subblocks have been processed. A more detailed outline of the code is as follows:

Main Thread
- Set number of Accolade hardware rings
- Load $2^{32}$ entry IPv4 anonymization table
- For each Accolade hardware ring
  - Launch Accolade Worker thread

Accolade Worker
- Create libpcap handle to Accolade device
- Allocate 64MB buffer for packet processing
- For each packet
  - Retrieve packet from Accolade device buffers
  - Append source and destination IP addresses to buffer
  - If buffer has $2^{23}$ packets
    - Launch GraphBLAS Worker thread with pointer to packet buffer
    - Allocate new 64MB buffer for packet processing

GraphBLAS Worker
- Initialize GraphBLAS
- For each subblock of $2^{17}$ packets
  - Create new GraphBLAS matrix
  - Create row, column and value vectors
  - For each packet in subblock
    - Lookup in anonymization table
    - Insert into row, column and value vectors
  - Build GraphBLAS matrix from row, column and value vectors; summing duplicate entries
  - Serialize and compress GraphBLAS matrix
  - Append to TAR buffer
- Write TAR buffer to file

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 $2^{32}$ 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). 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; these are approximately proportional to the number of threads/cores being used.

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 capable of constructing anonymized hypersparse traffic matrices at a rate above that corresponding to a typical 400 Gigabit network link.

VI. CONCLUSIONS AND FUTURE WORK
For many operating domains (land, sea, undersea, air, space, ...) long range detection is a cornerstone of defense. Long range detection in the cyber domain requires significant network traffic to be analyzed 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. Constructing and analyzing anonymized hypersparse traffic matrices are operations ideally suited to the GraphBLAS high performance library. Using an Accolade Technologies edge network device the performance of the GraphBLAS is demonstrated on a near worse case traffic scenario using a continuous stream of CAIDA Telescope darknet packets. The performance was explored by varying the number of traffic ring buffers, which are proportional to the number of threads and processor cores used. Rates of over 50,000,000 packets per second for constructing anonymized hypersparse traffic matrices were achieved which exceeds 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. This work suggests a variety of future directions that could be pursued: (1) exploring additional network cards; (2) developing the appropriate key management architecture for multiple observatories and outposts; (3) analysis of spatial temporal patterns in anonymized traffic matrices to identify adversarial activities; (4) cross-correlation of data from different observatories and outposts; (5) development of AI algorithms for classification of background traffic; (6) creation of underlying models of traffic.

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-summaries/annual-internet-report/white-paper-c11-741490.html.
[3] W. P. Delaney, Perspectives on Defense Systems Analysis. MIT Press, 2015.
[4] S. Topouzi, A. Sarris, Y. Pikoulas, S. Mertikas, X. Frantzis, and A. Giourou, “Ancient mantinea’s defence network reconsidered through a gis approach,” BAR INTERNATIONAL SERIES, vol. 1016, pp. 559–566, 2002.
[5] Y. Shi and Y. He, “Research on the historical and cultural value of and protection strategy for rammed earth watchtower houses in chongqing, china,” Built Heritage, vol. 5, no. 1, pp. 1–16, 2021.
[6] R. Cacciotti, “The ‘guardian of the pontifical state’: structural assessment of a damaged coastal watchtower in south lazio,” Master’s thesis, Universidade do Minho, 2010.
[7] R. A. Watt-Watson, Three Steps to Victory; A Personal Account by Radar’s Greatest Pioneer. London: Odhams Press, 1957.
[8] W. P. Delaney, “Air defense of the united states: Strategic missions and modern technology,” International Security, vol. 15, no. 1, pp. 181–211, 1990.
[9] J. Geul, E. Mooij, and R. Noomen, “Modelling and assessment of the current and future space surveillance network.” 7th Ecsd, 2017.
[10] K. W. O’Haver, C. K. Barker, G. D. Dockery, and J. D. Hufskedar, “Radar development for air and missile defense,” Johns Hopkins APL Tech. Digest, vol. 34, no. 2, pp. 140–153, 2018.
[11] “CAIDA Anonymized Internet Traces Dataset (April 2008 - January 2019),” https://www.caida.org/catalog/datasets/passive_dasets/.
[12] “UCSD Network Telescope,” https://www.caida.org/projects/network_telescope/.
[13] “Global Cyber Alliance,” https://www.globalcyberalliance.org/.
[14] “Greynoise,” https://greynoise.io/.
[15] “MAWI Working Group Traffic Archive,” http://maawi.wide.ad.jp/maawi/.
[16] “Shadowserver Foundation,” https://www.shadowserver.org/.
[17] J. Kepner, C. Meiners, C. Byun, S. McGuire, T. Davis, W. Arcand, J. Bernays, D. Bestor, W. Bergeron, V. Gadeppally, R. Haranas, 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.
[18] K. Claffy, “Measuring the internet,” IEEE Internet Computing, vol. 4, no. 1, pp. 73–78, 2000.
[19] B. Li, J. Springer, G. Babis, and M. H. Gunes, “A survey of network flow applications,” Journal of Network and Computer Applications, vol. 36, no. 2, pp. 567–581, 2013.
[20] M. Rabinovich and M. Allman, “Measuring the internet,” IEEE Internet Computing, vol. 20, no. 4, pp. 6–8, 2016.
[21] K. Claffy and D. Clark, “Workshop on internet economics (wie 2019) report,” SIGCOMM Comput. Commun. Rev., vol. 50, pp. 53–59, May 2020.
[22] J. Kepner, J. Bernays, S. Buckley, K. Cho, C. Conrad, L. Daigle, K. Erhardt, V. Gadepally, B. Greene, M. Jones, R. Knafe, B. Mags, P. Michaleas, C. Meiners, A. Morris, A. Pentland, S. Pisharody, S. Povazek, A. Prout, P. Reiner, K. Suzuki, K. Takeda, T. Tauber, L. Walker, and D. Stetson, “Zero hotnets: An observe-pursue-counter approach.” Belfer Center Reports, 6 2021.
[23] S. Weed, “Beyond zero trust: Reclaiming blue cyberspace,” Master’s thesis, United States Army War College, 2022.
[24] J. Kepner and J. Gilbert, Graph algorithms in the language of linear algebra. SIAM, 2011.
[25] J. Kepner, D. Bader, A. Bulu•, J. Gilbert, T. Mattson, and H. Meyerhenke, “Graphs, matrices, and the graphblas: Seven good reasons,” Procedia Computer Science, vol. 51, pp. 2453–2462, 2015.
[26] J. Kepner, P. Aaltenon, D. Bader, A. Bulu•, F. Franchetti, J. Gilbert, D. Hutchison, M. Kamar, 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.
[64] J. Nair, A. Wierman, and B. Zwart, “The fundamentals of heavy tails: Properties, emergence, and estimation,” Preprint, California Institute of Technology, 2020.

[65] 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.

[66] A. J. Elmore, J. Duggan, M. Stonebraker, M. Balazinska, U. Cetintemel, V. Gadeapally, J. Heer, B. Howe, J. Kepner, T. Kraska, et al., “A demonstration of the bigdawg polystore system,” Proceedings of the VLDB Endowment, vol. 8, no. 12, p. 1908, 2015.

[67] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis, “The case for learned index structures,” in Proceedings of the 2018 International Conference on Management of Data, SIGMOD 18, (New York, NY, USA), pp. 489–504, Association for Computing Machinery, 2018.

[68] E. H. Do and V. N. Gadepally, “Classifying anomalies for network security,” in ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2907–2911, 2020.

[69] 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.

[70] B. Nichols, D. Buttlar, and J. P. Farrell, Pthreads programming. O’Reilly & Associates, Inc., 1996.