A Billion Updates per Second Using 30,000 Hierarchical In-Memory D4M Databases

Jeremy Kepner[1,2,3], Vijay Gadepally[1,2], Lauren Milechin[4], Siddharth Samsi[1],
William Arcand[1], David Bestor[1], William Bergeron[1], Chansup Byun[1], Matthew Hubbell[1],
Micheal Houle[1], Micheal Jones[1], Anne Klein[1], Peter Michaleas[1],
Julie Mullen[1], Andrew Prout[1], Antonio Rosa[1], Charles Yee[1], Albert Reuther[1]
[1]MIT Lincoln Laboratory Supercomputing Center, [2]MIT Computer Science & AI Laboratory,
[3]MIT Mathematics Department, [4]MIT Department of Earth, Atmospheric and Planetary Sciences

***Abstract*—Analyzing large scale networks requires high performance streaming updates of graph representations of these data. Associative arrays are mathematical objects combining properties of spreadsheets, databases, matrices, and graphs, and are well-suited for representing and analyzing streaming network data. The Dynamic Distributed Dimensional Data Model (D4M) library implements associative arrays in a variety of languages (Python, Julia, and Matlab/Octave) and provides a lightweight in-memory database. Associative arrays are designed for block updates. Streaming updates to a large associative array requires a hierarchical implementation to optimize the performance of the memory hierarchy. Running 34,000 instances of a hierarchical D4M associative arrays on 1,100 server nodes on the MIT SuperCloud achieved a sustained update rate of 1,900,000,000 updates per second. This capability allows the MIT SuperCloud to analyze extremely large streaming network data sets.**

I. Introduction

Networks form the basis of worldwide communication and it is estimated that in 2018, there will be almost 37 Terabytes per second (TB/s) of Internet Protocol (IP) traffic. The rapid rise of sophisticated cyber threats is well documented and a growing threat to our information systems [1], [2]. Development of novel computer network traffic analytics requires: high level programming environments, massive amount of network data, and diverse data products for "at scale" algorithm pipeline development. Our team has developed a scalable network analytics platform applied to a network data using the D4M (Dynamic Distributed Dimensional Data Model) analytics environment and MIT SuperCloud interactive computing environment [3]. D4M combines the power of sparse linear algebra, associative arrays, parallel processing, and distributed databases (such as SciDB and Apache Accumulo) to provide a scalable data and computation system that addresses the big data problems associated with network analytics development. The MIT SuperCloud allows users to interactively process massive amounts of data in minutes on many thousands of cores using the software and environments most familiar to them. A key challenge for this pipeline is handling streaming updates of a large networks. This paper describes the implementation of a hierarchical approach designed to optimize the performance of the memory hierarchy.

This material is based upon work supported by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001 and National Science Foundation grants DMS-1312831 and CCF-1533644. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Assistant Secretary of Defense for Research and Engineering or the National Science Foundation.

II. Hierarchical Associative Arrays

Analyzing large scale networks requires high performance streaming updates of graph representations of these data. Associative arrays are mathematical objects combining properties of spreadsheets, databases, matrices, and graphs, and are well-suited for representing and analyzing streaming network data (see Fig. 1). In many databases, these table operations can be mapped onto well-defined mathematical operations with known mathematical properties. For example, relational (or SQL) databases [4]–[6] are described by relational algebra [7]–[9] that corresponds to the union-intersection semiring $\cup.\cap$ [10]. Triple-store databases (NoSQL) [11]–[14] and analytic databases (NewSQL) [15]–[20] follow similar mathematics [21]. The table operations of these databases are further encompassed by associative array algebra, which brings the beneficial properties of matrix mathematics and sparse linear systems theory, such as closure, commutativity, associativity, and distributivity [22]. The aforementioned mathematical properties provide strong correctness guarantees that are independent of scale and particularly helpful when trying to reason about massively parallel systems.

The D4M library implements associative arrays in a variety of languages (Python, Julia, and Matlab/Octave) and provides a lightweight in-memory database. Associative arrays are designed for block updates. Streaming updates to a large associative array requires a hierarchical implementation to optimize the performance of the memory hierarchy (see Fig. 2). Rapid updates are performed on the smallest arrays in the fastest memory. If the number of entries exceeds the threshold c_i, then $\mathbf{A}_i$ is added to $\mathbf{A}_{i+1}$ and $\mathbf{A}_i$ is cleared. Hierarchical arrays dramatically reduce the number of updates to slow memory. Upon query, all layers in the hierarchy are summed into largest array. The cut values c_i can be selected so as to optimize the performance with respect to particular applications.

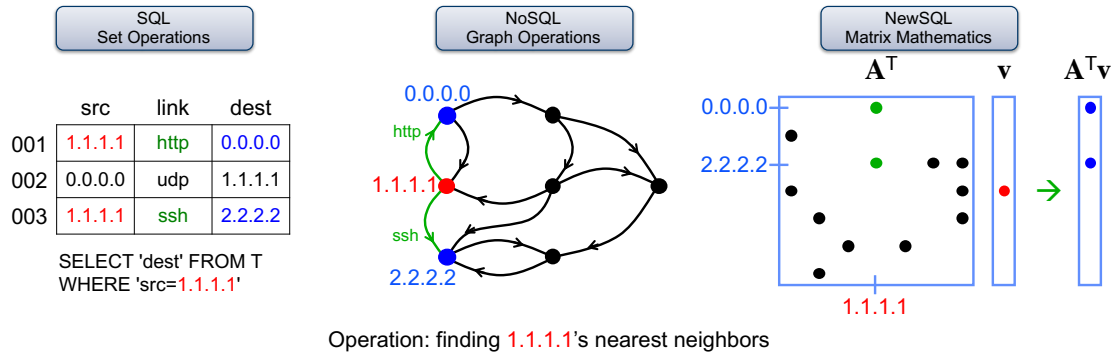

Fig. 1. Associative arrays combine the properties of databases, graphs, and matrices and provide common mathematics that span SQL, NoSQL, and NewSQL databases, and are ideal for analyzing networks. The diagram shows the graph operation of finding the neighbors of 1.1.1.1 in each representation.

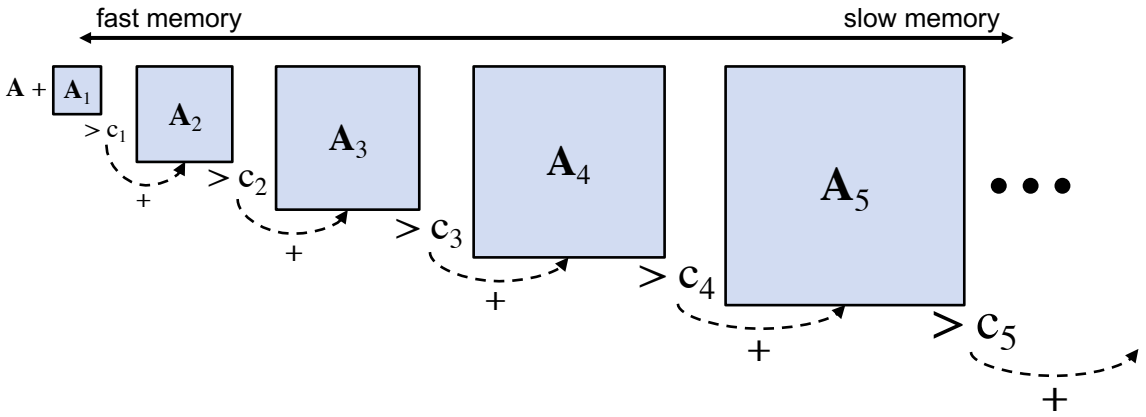

Fig. 2. Hierarchical associative arrays store increasing numbers of non-zero entries in each layer. If layer $\mathbf{A}_i$ surpasses the non-zero threshold c_i it is added to $\mathbf{A}_{i+1}$ and cleared. Hierarchical arrays ensure that the majority of updates are performed in fast memory.

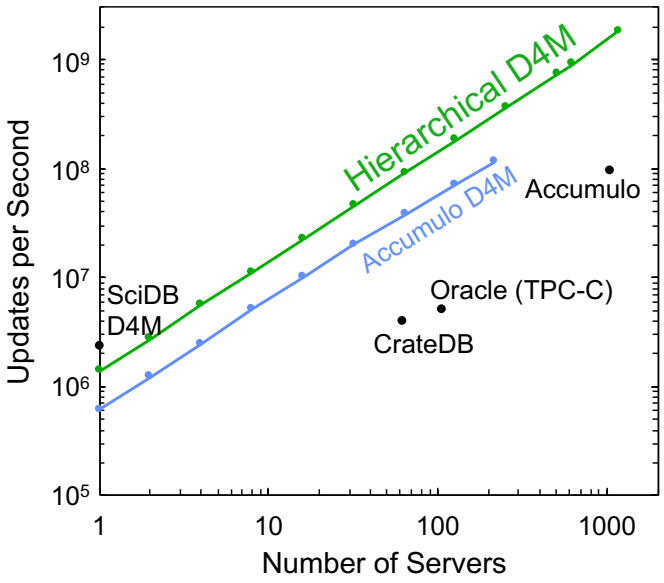

Fig. 3. Update rate as a function of number of servers for hierarchical D4M associative arrays and other previous published work: Accumulo D4M [23], SciDB D4M [24], Accumulo [25], Oracle TPC-C benchmark, and CrateDB [26]

III. Performance Results

The performance of associative arrays are benchmarked using a power-law graph of 100,000,000 entries divided up into 1,000 sets of 100,000 entries. These data were then simultaneously loaded and updated using varying numbers processes on varying number of nodes on the MIT SuperCloud. This experiment mimics thousands of processors each creating many different graphs of 100,000,000 edges each. In a real analysis application, each process would also compute various network statistics on each of the streams as they are updated. The update rate as function of severs nodes is shown on Fig. 3. The achieved update rate of 1,900,000,000 updates per second is significantly larger than prior published results. This capability allows the MIT SuperCloud to analyze extremely large streaming network data sets.

References

[1] N. Kshetri, "Positive externality, increasing returns, and the rise in cybercrimes," *Communications of the ACM*, vol. 52, no. 12, pp. 141–144, 2009.

[2] C. Hale, "Cybercrime: Facts & figures concerning this global dilemma," *Crime and Justice International*, vol. 18, no. 65, pp. 5–6, 2002.

[3] V. Gadepally, J. Kepner, L. Milechin, W. Arcand, D. Bestor, B. Bergeron, C. Byun, M. Hubbell, M. Houle, M. Jones, *et al.*, "Hyperscaling internet graph analysis with d4m on the mit supercloud," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pp. 1–6, IEEE, 2018.

[4] M. Stonebraker, G. Held, E. Wong, and P. Kreps, "The design and implementation of ingres," *ACM Transactions on Database Systems (TODS)*, vol. 1, no. 3, pp. 189–222, 1976.

[5] C. J. Date and H. Darwen, *A guide to the SQL Standard: a user's guide to the standard relational language SQL*. Addison-Wesley, 1989.

[6] R. Elmasri and S. Navathe, *Fundamentals of database systems*. Addison-Wesley Publishing Company, 2010.

[7] E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, 1970.

[8] D. Maier, *The theory of relational databases*, vol. 11. Computer science press Rockville, 1983.

[9] S. Abiteboul, R. Hull, and V. Vianu, eds., *Foundations of Databases: The Logical Level*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed., 1995.

[10] H. Jananthan, Z. Zhou, V. Gadepally, D. Hutchison, S. Kim, and J. Kepner, "Polystore mathematics of relational algebra," in *Big Data Workshop on Methods to Manage Heterogeneous Big Data and Polystore Databases*, IEEE, 2017.

[11] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," *ACM SIGOPS operating systems review*, vol. 41, no. 6, pp. 205–220, 2007.

[12] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.

[13] L. George, *HBase: the definitive guide: random access to your planet-size data*. " O'Reilly Media, Inc.", 2011.

[14] A. Cordova, B. Rinaldi, and M. Wall, *Accumulo: Application Development, Table Design, and Best Practices*. " O'Reilly Media, Inc.", 2015.

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

[16] R. Kallman, H. Kimura, J. Natkins, A. Pavlo, A. Rasin, S. Zdonik, E. P. Jones, S. Madden, M. Stonebraker, Y. Zhang, J. Hugg, and D. Abadi, "H-store: A high-performance, distributed main memory transaction processing system," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1496–1499, 2008.

[17] M. Balazinska, J. Becla, D. Heath, D. Maier, M. Stonebraker, and S. Zdonik, "A demonstration of scidb: A science-oriented dbms," *Cell*, vol. 1, no. a2, 2009.

[18] M. Stonebraker and A. Weisberg, "The voltdb main memory dbms.," *IEEE Data Eng. Bull.*, vol. 36, no. 2, pp. 21–27, 2013.

[19] D. Hutchison, J. Kepner, V. Gadepally, and A. Fuchs, "Graphulo implementation of server-side sparse matrix multiply in the accumulo database," in *High Performance Extreme Computing Conference (HPEC)*, IEEE, 2015.

[20] V. Gadepally, J. Bolewski, D. Hook, D. Hutchison, B. Miller, and J. Kepner, "Graphulo: Linear algebra graph kernels for nosql databases," in *Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International*, pp. 822–830, IEEE, 2015.

[21] J. Kepner, V. Gadepally, D. Hutchison, H. Jananthan, T. Mattson, S. Samsi, and A. Reuther, "Associative array model of sql, nosql, and newsql databases," in *High Performance Extreme Computing Conference (HPEC)*, IEEE, 2016.

[22] J. Kepner and H. Jananthan, *Mathematics of Big Data*. MIT Press, 2018.

[23] J. Kepner, W. Arcand, D. Bestor, B. Bergeron, C. Byun, V. Gadepally, M. Hubbell, P. Michaleas, J. Mullen, A. Prout, A. Reuther, A. Rosa, and C. Yee, "Achieving 100,000,000 database inserts per second using accumulo and d4m," in *High Performance Extreme Computing Conference (HPEC)*, IEEE, 2014.

[24] S. Samsi, L. Brattain, W. Arcand, D. Bestor, B. Bergeron, C. Byun, V. Gadepally, M. Hubbell, M. Jones, A. Klein, *et al.*, "Benchmarking scidb data import on hpc systems," in *High Performance Extreme Computing Conference (HPEC)*, pp. 1–5, IEEE, 2016.

[25] R. Sen, A. Farris, and P. Guerra, "Benchmarking apache accumulo bigdata distributed table store using its continuous test suite," in *Big Data (BigData Congress), 2013 IEEE International Congress on*, pp. 334–341, IEEE, 2013.

[26] CrateDB, "Big Bite: Ingesting Performance of Large Clusters." https://crate.io/a/big-cluster-insights-ingesting/, 2016. [Online; accessed 01-December-2018].