platforms on top of Spark or Hadoop but they are commercial and closed source solutions.

The work presented in [18] deals with hierarchical partitioning, optimizing the query plan to prune processing only to required data partitions. For each data dimension there is a different level in the partitioning tree structure. It involves changing the optimizer module by adding extra features to decide on the chain of joins performed. The main focus of the authors is on traditional RDBMSs while they claim that their approach can be extended to parallel databases. Our approach differs in the fact that we use a flat partitioning scheme dictated by the K-d Tree structure and our system design is tailored to using NoSQL storage and distributed system techniques. Furthermore, our partitioning scheme focuses on splitting the volume of data appropriately to fit in a mapper's memory to perform efficient map-side joins.

In [20], the authors present TidalRace which builds on data streaming applications and show how to optimize partition-based operations. Datix focuses on log processing (batch operations) and overall optimization of query execution in various cases. In contrast, TidalRace supports incremental updates to partitioning information, partition re-organization, and partition-wise optimizations.

DBStream [10] is a Data Stream Warehouse solution for Network Traffic Monitoring and Analysis applications. The queries we execute could also be deployed on this system by extending the functionality of DBStream to support the import of various *meta-datasets* like Datix and then evaluate the resulting performance. DBStream supports real-time data analysis and incremental queries apart from batch processing jobs. However, it is deployed in a centralized manner over a traditional RDBMS (PostgreSQL) while our system is fully decentralized, designed to be able to scale to a large number of nodes and gain extra performance when more resources become available. TicketDB [9], which is the predecessor of DBStream, was compared to vanilla MapReduce jobs performing reduce-side joins, but it does not use a partitioning scheme similar to our K-d Tree approach to enable efficient execution of map-side joins.

6. CONCLUSION

In this paper we introduced a novel network analytics system that depends on distributed processing techniques and is able to effectively execute filtering queries over state-of-the-art distributed processing engines. We introduced a smart pre-partitioning scheme to speed up the execution time of filtering queries (i.e., over a particular time period or set of IP addresses) and we integrated this functionality into an SQL compliant system by using custom-made user-defined functions that are aware of the data format and implement a custom variation of map-join algorithm. Our approach reduced query execution time compared to the basic Hive and Shark implementation by nearly 70%, while efficiently answering queries that took over a day to be processed with existed Python-based code. In this work, we used sFlows as a log dataset from which various information was recovered. Extensions of our work include the assessment of the system's performance when sFlow records are being streamed rather than stored in advance.

7. ACKNOWLEDGMENTS

Nikolaos Papailiou was supported by IKY fellowships of excellence for postgraduate studies in Greece - SIEMENS program. Georgios Smaragdakis was supported by the EU Marie Curie Fellowship CDN-H and EU project BigFoot.

8. REFERENCES

[1] Bro. http://www.bro.org/.
[2] Cisco Visual Networking Index: Forecast and Methodology, 2013 – 2018. Available at http://www.cisco.com.
[3] GeoLite. http://dev.maxmind.com/geoip/legacy/geolite/.
[4] Internet Scans. http://scans.io/.
[5] NetFlow. http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html.
[6] sFlow. http://www.sflow.org/.
[7] TreeMap. http://docs.oracle.com/javase/7/docs/api/java/util/TreeMap.html.
[8] F. N. Afrati and J. D. Ullman. Optimizing Joins in a Map-Reduce Environment. In *EDBT*, 2010.
[9] A. Baer, A. Barbuzzi, P. Michiardi, and F. Ricciato. Two Parallel Approaches to Network Data Analysis. In *LADIS*, 2011.
[10] A. Bar, P. Casas, L. Golab, and A. Finamore. DBStream: an Online Aggregation, Filtering and Processing System for Network Traffic Monitoring. In *IWCMC*, 2014.
[11] S. Blanas, J. M. Patel, V. Ercegovac, J. Rao, E. J. Shekita, and Y. Tian. A Comparison of Join Algorithms for Log Processing in MapReduce. In *ACM SIGMOD*, 2010.
[12] V. K. Bumgardner and V. W. Marek. Scalable Hybrid Stream and Hadoop Network Analysis System. In *ACM SPEC*, 2014.
[13] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *ACM TOCS 26(2)*, 2008.
[14] N. Chatzis, G. Smaragdakis, A. Feldmann, and W. Willinger. There is More to IXPs than Meets the Eye. *CCR 45(5), 2013.*
[15] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Comm. of the ACM 51(1)*, 2008.
[16] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-Wide Scanning and its Security Applications. In *USENIX Security Symposium*, 2013.
[17] V. Gopalkrishnan, Q. Li, and K. Karlapalem. Star/Snow-Flake Schema Driven Object-Relational Data Warehouse Design and Query Processing Strategies. In *DaWaK*. 1999.
[18] H. Herodotou, N. Borisov, and S. Babu. Query Optimization Techniques for Partitioned Tables. In *ACM SIGMOD*, 2011.
[19] Y. Huai, A. Chauhan, A. Gates, G. Hagleitner, E. Hanson, O. O'Malley, J. Pandey, Y. Yuan, R. Lee, and X. Zhang. Major Technical Advancements in Apache Hive. In *ACM SIGMOD*, 2014.
[20] T. Johnson and V. Shkapenyuk. Data Stream Warehousing in Tidalrace. In *CIDR*, 2015.
[21] V. Koukis, C. Venetsanopoulos, and N. Koziris. ∼okeanos: Building a Cloud, Cluster by Cluster. *IEEE Internet Computing 17(3)*, 2013.
[22] Y. Lee and Y. Lee. Toward Scalable Internet Traffic Measurement and Analysis with Hadoop. *CCR 43(1), 2013.*
[23] B. Li, M. H. Gunes, G. Bebis, and J. Springer. A Supervised Machine Learning Approach to Classify Host Roles On Line Using sFlow. In *ACM HPDC*, 2013.
[24] B. J. Louis. Multidimensional Binary Search Trees Used for Associative Searching. *Comm. of the ACM 18(9)*, 1975.
[25] A. Okcan and M. Riedewald. Processing Theta-Joins using MapReduce. In *ACM SIGMOD*, 2011.
[26] N. Papailiou, I. Konstantinou, D. Tsoumakos, P. Karras, and N. Koziris. H2RDF+: High-performance Distributed Joins over Large-scale RDF Graphs. In *IEEE BigData*, 2013.
[27] I. Poese, B. Frank, B. Ager, G. Smaragdakis, and A. Feldmann. Improving Content Delivery using Provider-aided Distance Information. In *IMC*, 2010.
[28] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *IEEE MSST*, 2010.
[29] L. Tang and N. Jain. Join Strategies in Hive. *Hive Summit.*
[30] A. Thusoo et al. Hive: A Warehousing Solution over a Map-Reduce Framework. *VLDB*, 2009.
[31] H.-c. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker. Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters. In *ACM SIGMOD*, 2007.
[32] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster Computing with Working Sets. In *USENIX conference on Hot topics in cloud computing*, 2010.