Our implementations of parallel TC algorithms are inspired by previous studies on parallel TC computation [6]–[9]. The idea of implementing SEMINAIVE and SMART using hash-based parallel relational algebra operators is attributed to Valduriez and Khoshafian [6]. Agrawal and Jagadish [7] and Wolfson et al. [8] proposed to partition the computation by the source vertices so that each core applies SEMINAIVE on a set of source vertices. The idea is similar to that of SSC1 except SSC1 applies SEMINAIVE on one source vertex one at the time. Cacace et al. [9] provided a survey on parallel TC algorithms. Previous studies use theoretical models to analyse the performance of algorithms, whereas our study focuses on experimental evaluation. In another experimental study [22] that includes the parallel Floyd algorithm [7], we showed that the Floyd algorithm achieves competitive performance for small dense graphs. But its memory requirement is impractical for large sparse graphs. Moreover, it is outperformed by SSC12 in the experiments.

VI. CONCLUSION AND FURTHER WORK

In this paper, we compared several recursive query evaluation algorithms on a modern multicore machine. A clear conclusion emerging from these experiments is that, for multicore machines, the simple SSC algorithms perform better than other algorithms in terms of speed and significantly better in terms of memory utilization. We thus introduced an algorithm, called SSC12, which combines the strengths of SSC1 and SSC2, and thus provides the obvious target algorithm for the compiler of our Datalog system DeAL [28] on multicore machines. However, our experiments also confirmed that performance of SSC12 (and other algorithms) on multicore machines will always be limited by the memory bandwidth bottleneck. Higher level of scalability through parallelism are however achievable on multi-node clusters. For multi-node clusters, the SMART algorithm has been shown to be optimal [1] — a conclusion that is confirmed by our recent investigation. The objective of this ongoing investigation is understanding the cost-performance tradeoffs on different multicore and multi-node systems for TC-like applications. From this study we seek to derive simple criteria for deciding which system, out of the many available on the cloud, can be most cost-effective for the application at hand. The ability of our Datalog compiler [28] to retarget recursive queries for different platforms is based on its ability to transform linear recursive rules into non-linear ones, which was described through simple examples in Section II. Many important algorithms that use TC-like rules in conjunction with monotonic aggregates [19] are amenable to such platform-driven porting and optimization.

ACKNOWLEDGMENT

This work was supported by NSF grants IIS 1218471 and IIS 1118107. We would like to thank the reviewers, Tyson Condie, Kai Zeng, Shi Gao and Alexander Shkapsky for their comments. We would like to thank the authors of [29], [30] for making their source code available online.

REFERENCES

[1] F. N. Afrati, V. Borkar, M. Carey *et al.*, "Map-reduce extensions and recursive queries," in *EDBT*. ACM, 2011, pp. 1–8.
[2] F. N. Afrati and J. D. Ullman, "Transitive closure and recursive datalog implemented on clusters," in *EDBT*. ACM, 2012, pp. 132–143.
[3] M. Shaw, P. Koutris, B. Howe *et al.*, "Optimizing large-scale semi-naive datalog evaluation in hadoop," in *Datalog in Academia and Industry*. Springer, 2012, pp. 165–176.
[4] Y. E. Ioannidis, "On the computation of the transitive closure of relational operators," in *VLDB*, vol. 86, 1986, pp. 403–411.
[5] P. Valduriez and H. Boral, "Evaluation of recursive queries using join indices," in *Expert Database Conf.*, 1986, pp. 271–293.
[6] P. Valduriez and S. Khoshfian, "Parallel evaluation of the transitive closure of a database relation," *International Journal of Parallel Programming*, vol. 17, no. 1, pp. 19–42, 1988.
[7] R. Agrawal and H. Jagadish, "Multiprocessor transitive closure algorithms," in *Proceedings of the first international symposium on Databases in parallel and distributed systems*. IEEE, 1988, pp. 56–66.
[8] O. Wolfson, W. Zhang, H. Butani *et al.*, "Parallel processing of graph reachability in databases," *International Journal of Parallel Programming*, vol. 21, no. 4, pp. 269–302, 1992.
[9] F. Cacace, S. Ceri, and M. Houtsma, "A survey of parallel execution strategies for transitive closure and logic programs," *Distributed and Parallel Databases*, vol. 1, no. 4, pp. 337–382, 1993.
[10] S. Warshall, "A theorem on boolean matrices," *JACM*, vol. 9, no. 1, pp. 11–12, 1962.
[11] H. S. Warren Jr, "A modification of warshall's algorithm for the transitive closure of binary relations," *CACM*, vol. 18, no. 4, pp. 218–220, 1975.
[12] R. Agrawal and H. Jagadish, "Direct algorithms for computing the transitive closure of database relations." in *VLDB*, vol. 87, 1987, pp. 1–4.
[13] Y. E. Ioannidis and R. Ramakrishnan, "Efficient transitive closure algorithms." in *VLDB*, vol. 88, 1988, pp. 382–394.
[14] R. Agrawal and H. Jagadish, "Hybrid transitive closure algorithms." in *VLDB*, 1990, pp. 326–334.
[15] H. Jakobsson, "Mixed-approach algorithms for transitive closure," in *PODS*. ACM, 1991, pp. 199–205.
[16] R. Kabler, Y. E. Ioannidis, and M. J. Carey, "Performance evaluation of algorithms for transitive closure," *Information Systems*, vol. 17, no. 5, pp. 415–441, 1992.
[17] S. Dar and R. Ramakrishnan, "A performance study of transitive closure algorithms," in *ACM SIGMOD Record*, vol. 23, no. 2. ACM, 1994, pp. 454–465.
[18] C. Zaniolo, S. Ceri, C. Faloutsos *et al.*, "Advanced database systems," 1997.
[19] A. Shkapsky, N. Lu, and C. Zaniolo, "Towards more powerful datalog systems: The implementation and optimization of recursive queries with monotonic aggregates in deals," UCLA, Tech. Rep. 140004, 2014.
[20] C. Kim, T. Kaldewey, V. W. Lee *et al.*, "Sort vs. hash revisited: fast join implementation on modern multi-core cpus," *PVLDB*, vol. 2, no. 2, pp. 1378–1389, 2009.
[21] S. Manegold, P. Boncz, and M. Kersten, "Optimizing main-memory join on modern hardware," *TKDE*, vol. 14, no. 4, pp. 709–730, 2002.
[22] M. Yang and C. Zaniolo, "Main memory evaluation of recursive queries on multicore machines," UCLA, Tech. Rep. 140014, 2014.
[23] "Graphstream," http://graphstream-project.org.
[24] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *SIGKDD*. ACM, 2005, pp. 177–187.
[25] "Wikipedia," http://www.wikipedia.org.
[26] "Eastern usa road network," http://www.dis.uniroma1.it/challenge9/data/USA-road-d/USA-road-d.E.gr.gz.
[27] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Mathematics*, vol. 6, no. 1, pp. 29–123, 2009.
[28] A. Shkapsky, K. Zeng, and C. Zaniolo, "Graph queries in a next-generation datalog system," *PVLDB*, vol. 6, no. 12, pp. 1258–1261, 2013.
[29] S. Blanas, Y. Li, and J. M. Patel, "Design and evaluation of main memory hash join algorithms for multi-core cpus," in *SIGMOD*. ACM, 2011, pp. 37–48.
[30] C. Balkesen, J. Teubner, G. Alonso *et al.*, "Main-memory hash joins on multi-core cpus: Tuning to the underlying hardware," in *ICDE*, 2013, pp. 362–373.