Algorithmic Aspects of Parallel Data Processing

Paraschos Koutris
University of Wisconsin-Madison
paris@cs.wisc.edu

Semih Salihoglu
University of Waterloo
semih.salihoglu@uwaterloo.ca

Dan Suciu
University of Washington
suciu@cs.washington.edu

the essence of knowledge
Boston — Delft

Foundations and Trends® in Databases

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

P. Koutris, S. Salihoglu and D. Suciu. *Algorithmic Aspects of Parallel Data Processing*. Foundations and Trends® in Databases, vol. 8, no. 4, pp. 239–370, 2016.

This Foundations and Trends® issue was typeset in LaTeX using a class file designed by Neal Parikh. Printed on acid-free paper.

ISBN: 978-1-68083-406-2

Foundations and Trends® in Databases

Volume 8, Issue 4, 2016

Editorial Scope

Topics

Foundations and Trends® in Databases covers a breadth of topics relating to the management of large volumes of data. The journal targets the full scope of issues in data management, from theoretical foundations, to languages and modeling, to algorithms, system architecture, and applications. The list of topics below illustrates some of the intended coverage, though it is by no means exhaustive:

- Data models and query languages
- Query processing and optimization
- Storage, access methods, and indexing
- Transaction management, concurrency control, and recovery
- Deductive databases
- Parallel and distributed database systems
- Database design and tuning
- Metadata management
- Object management
- Trigger processing and active databases
- Data mining and OLAP
- Approximate and interactive query processing
- Data warehousing
- Adaptive query processing
- Data stream management
- Search and query integration
- XML and semi-structured data
- Web services and middleware
- Data integration and exchange
- Private and secure data management
- Peer-to-peer, sensornet, and mobile data management
- Scientific and spatial data management
- Data brokering and publish/subscribe
- Data cleaning and information extraction
- Probabilistic data management

Information for Librarians

Foundations and Trends® in Databases, 2016, Volume 8, 4 issues. ISSN paper version 1931-7883. ISSN online version 1931-7891. Also available as a combined paper and online subscription.

Foundations and Trends® in Databases
Vol. 8, No. 4 (2016) 239–370

DOI: 10.1561/1900000055

Algorithmic Aspects of Parallel Data Processing

Paraschos Koutris
University of Wisconsin-Madison
paris@cs.wisc.edu

Semih Salihoglu
University of Waterloo
semih.salihoglu@uwaterloo.ca

Dan Suciu
University of Washington
suciu@cs.washington.edu

Contents

Abstract

In the last decade or so we have witnessed a growing interest in processing large data sets on large distributed clusters. The idea was pioneered by the MapReduce framework, and has been widely adopted by several other systems, including PigLatin, Hive, Scope, U-SQL, Dremmel, Spark and Myria. A large part of the complex data analysis performed by these systems consists of a sequence of relatively simple query operations, such as joining two or more tables. This survey discusses recent algorithmic developments for distributed data processing. It uses a theoretical model of parallel processing called the Massively Parallel Computation (MPC) model, which is a simplification of the BSP model where the only cost is given by the amount of communication and the number of communication rounds. The survey studies several algorithms for multi-join queries, for sorting, and for matrix multiplication, and discusses their relationships and common techniques applied across the different data processing tasks.

P. Koutris, S. Salihoglu and D. Suciu. *Algorithmic Aspects of Parallel Data Processing*. Foundations and Trends® in Databases, vol. 8, no. 4, pp. 239–370, 2016.
DOI: 10.1561/1900000055.

1

Introduction

In the last decade we have witnessed a huge and growing interest in processing large data sets on large distributed clusters. This trend began with the MapReduce framework [31], and has been widely adopted by several other systems, including PigLatin [69], Hive [83], Scope [24], Dremmel [65], Spark [91] and Myria [88] to name a few. While the applications of such systems are diverse (e.g., machine learning, data analytics), most involve relatively standard data processing tasks, such as identifying relevant data, cleaning, filtering, joining, grouping, transforming, extracting features, and evaluating results [25, 35].

This has generated great interest in the study of algorithms for data processing on large distributed clusters. This survey reviews some of the recent theoretical results on efficient data processing on large distributed architectures, as well as some of the relevant classical results on parallel sorting and parallel matrix multiplication.

The survey begins in Chapter 2 with a review of parallel models used to analyze algorithms on large distributed clusters. Modern data analytics run on large, shared-nothing clusters, where the cost of communication during data reshuffling can dominate the running time. For example, individual jobs in Cosmos, Microsoft's distributed file system, often execute on over 10k nodes [72]. We introduce a very simple model of parallel computation, called the Massively Parallel Computation model (MPC) where the cost of a distributed algorithm is measured in the amount of communication per processor and the number of communication rounds. This model is a simplification of Valiant's *Bulk Synchronous Parallel (BSP)* model [84], and allows us to separate the computation cost from the communication cost, and to focus solely on the latter. In this chapter we introduce the MPC model, then review several important classical models of parallel computation, and discuss their connection to the MPC model.

In Chapter 3 we present and analyze two different approaches for computing in parallel the join of two large relations. Join operations are the bread and butter of most database processing tasks, and the support of efficient join algorithms is a top priority for all major big data systems. We discuss Parallel Hash join, and Parallel Sort Join. The preferred algorithm in practice is the Parallel Hash join, because on most datasets this algorithm is very effective and scales up linearly with the number of processors. However, the Parallel Hash join performs poorly on skewed data, when a large number of records have the same value of the join attribute and, thus, are hashed to the same processor. We discuss in detail how to handle skewed data. In contrast, Parallel Sort join is simpler and less sensitive to skew, but requires extra communication rounds to do the actual sorting.

Next, we consider multi-join queries, and discuss a variety of hash-based algorithms in Chapter 4. In the standard architecture of a database system, a multi-join query is first converted into a query plan, which is then optimized, and finally the plan is executed. The plan consists of simple operators like join, selection, duplicate elimination, and each operator creates an intermediate result that, in distributed query processing, needs to be materialized and re-shuffled for the next operator. Afrati and Ullman [4] pioneered an alternative approach for computing a multi-join query on a distributed system, which computes the query using a single reshuffle operation. Their algorithm, initially described for the MapReduce system, organizes the processors (which correspond to *reducers* in a MapReduce job) in a multidimensional cube, then partitions each input relation in a sub-cube. The theoretical aspects of the algorithm have been studied in [17], where the algorithm was called *HyperCube*, while extensions to skewed data and to multiple rounds of communication were further discussed in [18, 57]; these will be reviewed in this chapter. While these algorithms are appealing because of their strong theoretical guarantees, modern database systems compute multi-join queries in traditional ways, by converting the query into a join plan. We continue the chapter by discussing the theoretical aspects of join plans, which have a long history in database theory. We review Yannakakis' algorithm for computing acyclic queries [90], the concept of hypertree decomposition [42], and various notions of tree-width [43, 55], and describe how these have been put together in the GYM algorithm [3].

In Chapter 5 we discuss a few traditional aspects of parallel sorting algorithms. Similar to hashing, sorting is a core technique in database query processing, both in the sequential and in the parallel setting. Sort-based techniques suffer less than hash-based techniques from skew in the data. For example, recently Hu, Tao, and Yi [45] have shown how to use sorting to design a simple join algorithm that is provably optimal for any input data (reviewed in Chapter 3). In this chapter we review some fundamental lower bounds for sorting on a distributed system, and also review three classic parallel sorting algorithms: Batcher's odd-even sort [16], Cole's algorithm [27], and Goodrich's algorithm [40].

Finally, in Chapter 6 we discuss classic parallel algorithms for matrix multiplication. We focus on multiplication of dense square matrices and adopt the relational view of matrix multiplication as a join of two tables followed by a group-by-and aggregate computation. Using techniques similar to those used in proving lower bounds in sorting and multi-join queries, we review the communication and round lower bounds for matrix multiplication of square and dense matrices. Then, we review existing algorithms that match these lower bounds. The chapter ends with a very brief overview of other known results in linear algebra, such as multiplication of non-square and sparse matrices, or *LU* and Cholesky matrix factorization.

Table 1.1 summarizes the notations used in the survey.

Table 1.1: Notations Used Throughout the Survey.

Relation	R_j
Number of relations	ℓ
Variable	x_i
Number of variables	k
Query	q
Input size	IN or N
Output size	OUT
Number of processors	p
Number of communication rounds	r
Load (incoming communication per processor)	L
Memory per processor	M
Total communication	C
Fractional edge cover or edge packing	u_j
Fractional vertex cover or vertex packing	v_i
Fractional edge packing number	τ^*
Fractional edge covering number	ρ^*
Quasi-packing number	ψ^*

References

[1] Christopher R. Aberger, Susan Tu, Kunle Olukotun, and Christopher Ré. EmptyHeaded: A Relational Engine for Graph Processing. In *SIGMOD*, 2016.

[2] F. N. Afrati, A. D. Sarma, S. Salihoglu, and J. D. Ullman. Upper and Lower Bounds on the Cost of a Map-Reduce Computation. *PVLDB*, 6(4), 2013.

[3] Foto N. Afrati, Manas R. Joglekar, Christopher Ré, Semih Salihoglu, and Jeffrey D. Ullman. GYM: A Multiround Distributed Join Algorithm. In *ICDT*, 2017.

[4] Foto N. Afrati and Jeffrey D. Ullman. Optimizing multiway joins in a map-reduce environment. *IEEE Transactions on Knowledge and Data Engineering*, 23(9), 2011.

[5] R. C. Agarwal, S. M. Balle, F. G. Gustavson, M. Joshi, and P. Palkar. A Three-dimensional Approach to Parallel Matrix Multiplication. *IBM Journal of Research and Development*, 39(5), 1995.

[6] Alok Aggarwal, Ashok K. Chandra, and Marc Snir. Communication Complexity of PRAMs. *Theoretical Computer Science*, 71(1), 1990.

[7] Alok Aggarwal and S. Vitter, Jeffrey. The Input/Output Complexity of Sorting and Related Problems. *Communications of the ACM*, 31(9), 1988.

[8] Miklós Ajtai, János Komlós, and Endre Szemerédi. Sorting in $c \log n$ Parallel Sets. *Combinatorica*, 3(1), 1983.

[9] Albert Atserias, Martin Grohe, and Dániel Marx. Size Bounds and Query Plans for Relational Joins. *SIAM Journal on Computing*, 42(4), 2013.

[10] G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, and O. Schwartz. Strong Scaling of Matrix Multiplication Algorithms and Memory-Independent Communication Lower Bounds. Technical report, EECS Department, University of California, Berkeley, March 2012.

[11] Grey Ballard, Aydin Buluç, James Demmel, Laura Grigori, Benjamin Lipshitz, Oded Schwartz, and Sivan Toledo. Communication Optimal Parallel Multiplication of Sparse Random Matrices. In *SPAA*, 2013.

[12] Grey Ballard, James Demmel, Olga Holtz, Benjamin Lipshitz, and Oded Schwartz. Graph Expansion Analysis for Communication Costs of Fast Rectangular Matrix Multiplication. In *MedAlg*, 2012.

[13] Grey Ballard, James Demmel, Olga Holtz, and Oded Schwartz. Minimizing Communication in Numerical Linear Algebra. *SIAM Journal of Matrix Analysis Applications*, 32(3), 2011.

[14] Grey Ballard, James Demmel, Olga Holtz, and Oded Schwartz. Graph Expansion and Communication Costs of Fast Matrix Multiplication. *Journal of the ACM*, 59(6), 2013.

[15] Pablo Barceló, Georg Gottlob, and Andreas Pieris. Semantic Acyclicity Under Constraints. In *PODS*, 2016.

[16] Kenneth E. Batcher. Sorting Networks and Their Applications. In *AFIPS*, 1968.

[17] Paul Beame, Paraschos Koutris, and Dan Suciu. Communication Steps for Parallel Query Processing. In *PODS*, 2013.

[18] Paul Beame, Paraschos Koutris, and Dan Suciu. Skew in Parallel Query Processing. In *PODS*, 2014.

[19] Paul Beame, Paraschos Koutris, and Dan Suciu. Communication Cost in Parallel Query Processing. *CoRR*, abs/1602.06236, 2016.

[20] Guy E. Blelloch and Bruce M. Maggs. Parallel Algorithms. In *Algorithms and Theory of Computation Handbook*, chapter 25. Chapman & Hall/CRC, 2010.

[21] A. Borodin and J. E. Hopcroft. Routing, Merging, and Sorting on Parallel Models of Computation. *Journal of Computer and System Sciences*, 30(1), 1985.

[22] Aydin Buluç and John R. Gilbert. Challenges and Advances in Parallel Sparse Matrix-Matrix Multiplication. In *ICPP*, 2008.

[23] Lynn Elliot Cannon. *A Cellular Computer to Implement the Kalman Filter Algorithm.* PhD thesis, Montana State University, 1969.

[24] Ronnie Chaiken, Bob Jenkins, Per-Åke Larson, Bill Ramsey, Darren Shakib, Simon Weaver, and Jingren Zhou. SCOPE: easy and efficient parallel processing of massive data sets. *PVLDB*, 1(2), 2008.

[25] Surajit Chaudhuri. What Next?: A Half-dozen Data Management Research Goals for Big Data and the Cloud. In *PODS*, 2012.

[26] Shumo Chu, Magdalena Balazinska, and Dan Suciu. From Theory to Practice: Efficient Join Query Evaluation in a Parallel Database System. In *SIGMOD*, 2015.

[27] Cole, Richard. Parallel Merge Sort. *SIAM Journal on Computing*, 17(4), 1988.

[28] Michael Conley, Amin Vahdat, and George Porter. TritonSort 2014. `http://sortbenchmark.org/TritonSort2014.pdf`.

[29] Stephen A. Cook, Cynthia Dwork, and Rüdiger Reischuk. Upper and Lower Time Bounds for Parallel Random Access Machines without Simultaneous Writes. *SIAM Journal on Computing*, 15(1), 1986.

[30] David E. Culler, Richard M. Karp, David A. Patterson, Abhijit Sahay, Klaus E. Schauser, Eunice E. Santos, Ramesh Subramonian, and Thorsten von Eicken. LogP: Towards a Realistic Model of Parallel Computation. In *PPOPP*, 1993.

[31] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI*, 2004.

[32] Eliezer Dekel, David Nassimi, and Sartaj Sahni. Parallel Matrix and Graph Algorithms. *SIAM Journal on Computing*, 16(3), 1984.

[33] James Demmel, David Eliahu, Armando Fox, Shoaib Kamil, Benjamin Lipshitz, Oded Schwartz, and Omer Spillinger. Communication-Optimal Parallel Recursive Rectangular Matrix Multiplication. In *IPDPS*, 2013.

[34] David J. DeWitt and Jim Gray. Parallel Database Systems: The Future of High Performance Database Systems. *Communications of the ACM*, 35(6), 1992.

[35] EMC Corporation. Data Science Revealed: A Data-Driven Glimpse into the Burgeoning New Field. `http://www.emc.com/collateral/about/news/emc-data-science-study-wp.pdf`.

[36] Jon Feldman, S. Muthukrishnan, Anastasios Sidiropoulos, Clifford Stein, and Zoya Svitkina. On Distributing Symmetric Streaming Computations. *ACM Transactions on Algorithms*, 6(4), 2010.

[37] Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1), 1984.

[38] Sumit Ganguly, Abraham Silberschatz, and Shalom Tsur. Parallel Bottom-Up Processing of Datalog Queries. *Journal of Logic Programming*, 14(1&2), 1992.

[39] Alan Gibbons and Wojciech Rytter. *Efficient Parallel Algorithms*. Cambridge University Press, 1988.

[40] Michael T. Goodrich. Communication-Efficient Parallel Sorting. *SIAM Journal on Computing*, 29(2), 1999.

[41] Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, Searching, and Simulation in the Mapreduce Framework. In *ISAAC*, 2011.

[42] Georg Gottlob, Gianluigi Greco, Nicola Leone, and Francesco Scarcello. Hypertree Decompositions: Questions and Answers. In *PODS*, 2016.

[43] Martin Grohe and Dániel Marx. Constraint Solving via Fractional Edge Covers. *ACM Transactions on Algorithms*, 11(1), 2014.

[44] D. Halperin, V. Teixeira de Almeida, L. Choo, S. Chu, P. Koutris, D. Moritz, J. Ortiz, V. Ruamviboonsuk, J. Wang, A. Whitaker, S. Xu, M. Balazinska, B. Howe, and D. Suciu. Demonstration of the Myria Big Data Management Service. In *SIGMOD*, 2014.

[45] Xiao Hu, Yufei Tao, and Ke Yi. Output-optimal Parallel Algorithms for Similarity Joins. In *PODS*, 2017.

[46] M. Husain, J. McGlothlin, M. M. Masud, L. Khan, and B. M. Thuraisingham. Heuristics-Based Query Processing for Large RDF Graphs Using Cloud Computing. *IEEE Transactions on Knowledge and Data Engineering*, 23(9), 2011.

[47] Dror Irony, Sivan Toledo, and Alexander Tiskin. Communication Lower Bounds for Distributed-memory Matrix Multiplication. *Journal of Parallel and Distributed Computing*, 64(9), 2004.

[48] Hong Jia-Wei and H. T. Kung. I/O Complexity: The Red-blue Pebble Game. In *STOC*, 1981.

[49] Jie Jiang, Lixiong Zheng, Junfeng Pu, Xiong Cheng, Chongqing Zhao, Mark R. Nutter, and Jeremy D. Schaub. Tencent Sort. `http://sortbenchmark.org/TencentSort2016.pdf`.

[50] Manas Joglekar and Christopher Ré. It's All a Matter of Degree: Using Degree Information to Optimize Multiway Joins. In *ICDT*, 2016.

[51] S. Lennart Johnsson. Minimizing the Communication Time for Matrix Multiplication on Multiprocessors. *Parallel Computing*, 19(11), 1993.

[52] Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and Combinatorics*. Springer, 2012.

[53] Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. A Model of Computation for MapReduce. In *SODA*, 2010.

[54] Bas Ketsman and Dan Suciu. A Worst-Case Optimal Multi-Round Algorithm for Parallel Computation of Conjunctive Queries. In *PODS*, 2017.

[55] Mahmoud Abo Khamis, Hung Q. Ngo, and Atri Rudra. FAQ: Questions Asked Frequently. In *PODS*, 2016.

[56] Marcel Kornacker, Alexander Behm, Victor Bittorf, Taras Bobrovytsky, Alan Choi, Justin Erickson, Martin Grund, Daniel Hecht, Matthew Jacobs, Ishaan Joshi, Lenni Kuff, Dileep Kumar, Alex Leblang, Nong Li, Henry Robinson, David Rorke, Silvius Rus, John Russell, Dimitris Tsirogiannis, Skye Wanderman-milne, and Michael Yoder. Impala: A Modern, Open-Source SQL Engine for Hadoop. In *CIDR*, 2015.

[57] Paraschos Koutris, Paul Beame, and Dan Suciu. Worst-Case Optimal Algorithms for Parallel Query Processing. In *ICDT*, 2016.

[58] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[59] Longbin Lai, Lu Qin, Xuemin Lin, and Lijun Chang. Scalable subgraph enumeration in mapreduce: A cost-oriented approach. *The VLDB Journal*, 26(3), 2017.

[60] Leonid Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[61] Longbin Lai and Lu Qin and Xuemin Lin and Ying Zhang and Lijun Chang. Scalable distributed subgraph enumeration. *PVLDB*, 10(3), 2016.

[62] L. H. Loomis and H. Whitney. An Inequality Related to the Isoperimetric Inequality. *Bulletin of the American Mathematical Society*, 55(10), 1949.

[63] W. F. McColl and A. Tiskin. Memory-Efficient Matrix Multiplication in the BSP Model. *Algorithmica*, 24(3), 1999.

[64] A. C. McKellar and E. G. Coffman, Jr. Organizing Matrices and Matrix Operations for Paged Memory Systems. *Communications of the ACM*, 12(3), 1969.

[65] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis. Dremel: Interactive Analysis of Web-Scale Datasets. *PVLDB*, 3(1), 2010.

[66] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[67] Thomas Neumann and Gerhard Weikum. The RDF-3X Engine for Scalable Management of RDF Data. *VLDB Journal*, 19(1), 2010.

[68] H. Ngo, C. Ré, and A. Rudra. Skew Strikes Back: New Developments in the Theory of Join Algorithms. *SIGMOD Record*, 42(4), 2014.

[69] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig Latin: A Not-So-Foreign Language for Data Processing. In *SIGMOD*, 2008.

[70] Andrea Pietracaprina, Geppino Pucci, Matteo Riondato, Francesco Silvestri, and Eli Upfal. Space-round Tradeoffs for MapReduce Computations. In *ICS*, 2012.

[71] Raghu Ramakrishnan and Johannes Gehrke. *Database management systems (3rd edition)*. McGraw-Hill, 2003.

[72] Raghu Ramakrishnan, Baskar Sridharan, John R. Douceur, Pavan Kasturi, Balaji Krishnamachari-Sampath, Karthick Krishnamoorthy, Peng Li, Mitica Manu, Spiro Michaylov, Rogério Ramos, Neil Sharman, Zee Xu, Youssef Barakat, Chris Douglas, Richard Draves, Shrikant S. Naidu, Shankar Shastry, Atul Sikaria, Simon Sun, and Ramarathnam Venkatesan. Azure Data Lake Store: A Hyperscale Distributed File Service for Big Data Analytics. In *SIGMOD*, 2017.

[73] Alexander Rasmussen, George Porter, Michael Conley, Harsha V. Madhyastha, Radhika Niranjan Mysore, Alexander Pucher, and Amin Vahdat. TritonSort: A Balanced Large-scale Sorting System. In *NSDI*, 2011.

[74] Tim Roughgarden, Sergei Vassilvitskii, and Joshua R. Wang. Shuffles and Circuits: (On Lower Bounds for Modern Parallel Computation). In *SPAA*, 2016.

[75] Hanmao Shi and Jonathan Schaeffer. Parallel Sorting by Regular Sampling. *Journal of Parallel and Distributed Computing*, 14(4), 1992.

[76] Sort Benchmark Home Page. `http://sortbenchmark.org/`.

[77] Spark SQL. `https://spark.apache.org/sql/`.

[78] SPARQL Query Language for RDF. `https://www.w3.org/TR/rdf-sparql-query/`.

[79] Dan Suciu and Val Tannen. A Query Language for NC. *Journal of Computer and System Sciences*, 55(2), 1997.

[80] Zhao Sun, Hongzhi Wang, Haixun Wang, Bin Shao, and Jianzhong Li. Efficient Subgraph Matching on Billion Node Graphs. *PVLDB*, 5(9), 2012.

[81] Siddharth Suri and Sergei Vassilvitskii. Counting triangles and the curse of the last reducer. In *WWW*, 2011.

[82] Graves Thomas. GraySort and MinuteSort at Yahoo on Hadoop 0.23. `http://sortbenchmark.org/Yahoo2013Sort.pdf`.

[83] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive - A Warehousing Solution Over a Map-Reduce Framework. *PVLDB*, 2(2), 2009.

[84] Leslie G. Valiant. A Bridging Model for Parallel Computation. *Communications of the ACM*, August 1990.

[85] Todd L. Veldhuizen. Leapfrog Triejoin: A Simple, Worst-Case Optimal Join Algorithm. In *ICDT*, 2014.

[86] Jeffrey Scott Vitter. Algorithms and Data Structures for External Memory. *Foundations and Trends in Theoretical Computer Science*, 2(4), 2006.

[87] Jiamang Wang, Yongjun Wu, Hua Cai, Zhipeng Tang, Zhiqiang Lv, Bin Lu, Yangyu Tao, Chao Li, Jingren Zhou, and Hong Tang. FuxiSort. `http://sortbenchmark.org/FuxiSort2015.pdf`.

[88] Jingjing Wang, Tobin Baker, Magdalena Balazinska, Daniel Halperin, Brandon Haynes, Bill Howe, Dylan Hutchison, Shrainik Jain, Ryan Maas, Parmita Mehta, Dominik Moritz, Brandon Myers, Jennifer Ortiz, Dan Suciu, Andrew Whitaker, and Shengliang Xu. The Myria Big Data Management and Analytics System and Cloud Services. In *CIDR*, 2017.

[89] Reynold Xin, Parviz Deyhim, Ali Ghodsi, Xiangrui Meng, and Matei Zaharia. GraySort on Apache Spark by Databricks. `http://sortbenchmark.org/ApacheSpark2014.pdf`.

[90] Mihalis Yannakakis. Algorithms for Acyclic Database Schemes. In *VLDB*, 1981.

[91] Zaharia, M. and Chowdhury, M. and Franklin, M. J. and Shenker, S. and Stoica, I. Spark: Cluster Computing with Working Sets. In *HotCloud*, 2010.

[92] Zeng, Kai and Yang, Jiacheng and Wang, Haixun and Shao, Bin and Wang, Zhongyuan. A Distributed Graph Engine for Web Scale RDF Data. *VLDB*, 6(4), 2013.