Data Management Systems Research at TU Berlin

Ziawasch Abedjan, Sebastian Breß, Volker Markl, Tilmann Rabl, Juan Soto
Technische Universität Berlin and DFKI Berlin
{ziawasch.abedjan,sebastian.bress, volker.markl, rabl, juan.soto}@tu-berlin.de

ABSTRACT

Data management systems research at TU Berlin is spearheaded by the Database Systems and Information Management (DIMA) Group, the Big Data Management (BigDaMa) Group, as well as the affiliated Intelligent Analytics for Massive Data (IAM) Research Group at the German Research Center for Artificial Intelligence (DFKI). Jointly, our research activities encompass a wide variety of database topics, including benchmarking, data integration, modern hardware, and scalable data processing.

As of Fall 2018, the current team is comprised of three university professors, thirteen senior and postdoc researchers, twenty PhD students, and several research assistants. Among our notable accomplishments is the DFG-funded Stratosphere Research Unit, which laid the groundwork for what would later become Apache Flink. DIMA has also been leading the Berlin Big Data Center, one of only two BMBF-funded Big Data Competence Centers in Germany since 2014. In addition, DIMA is co-directing the Berlin Center for Machine Learning, one of four BMBF-funded Machine Learning Competence Centers in Germany.

1. INTRODUCTION

Modern applications have to cope with large, fast, and heterogeneous data, bridging the worlds of advanced data analysis and machine learning with data management. Naturally, this poses numerous research challenges for the design and usage of data analytics systems. Fortunately, novel advances in hardware technologies, data flow architectures, and machine learning techniques are making it possible to build efficient and user-friendly data processing systems. With a team of thirty database researchers, comprised of doctoral students, senior researchers, and university professors, TU Berlin is well positioned to address key challenges.

Given the significant importance of data flow systems, at its onset DIMA[1] embarked on the development of a next generation big data analytics platform. Initially, known as Stratosphere [6], over the course of several years, it would later go on to become Apache Flink, an open-source stream processing framework for parallel dataflow analysis.

Today, ongoing research focuses on meeting the requirement needs of novel Internet of Things infrastructures and increasing their ease of use. We have developed a declarative programming interface to enable data scientists to primarily focus their attention on analysis. Other key research topics that are underway include research on modern hardware, systems benchmarking, end-to-end machine learning pipelines, responsible data management, data analysis infrastructures, and information marketplaces. Moreover, the recently established BigDaMa Group[2] is actively conducting research (e.g., building end-to-end data preparation systems) to address data heterogeneity challenges.

In the following sections, we further motivate several of the aforementioned research topics and highlight key contributions from our database systems researchers. We will conclude with an overview of our existing grants and collaboration activities across our research projects.

2. SCALABLE DATA PROCESSING

Many specialized data processing systems have been developed, in order to analyze high *volume*, *velocity*, and *variety* data, efficiently and effectively. To meet these demands, systems often exploit specially optimized libraries. For example, to perform numerical linear algebra operations, conduct natural language processing, or execute graph analytics. Moreover, system building commonly employs both modern storage, such as non-uniform memory access (NUMA) designs and processing architectures, such as heterogeneous CPUs, to achieve higher performance. It is the wide diversity of systems and hardware solutions that greatly improve functionality and reduce the execution time for many data

[1] https://www.dima.tu-berlin.de/

[2] https://www.bigdama.tu-berlin.de

analytics-dependent applications.

Next, we discuss our research contributions in the area of scalable data processing, which includes Stratosphere, Apache Flink, stream processing, and declarative programming.

2.1 Stratosphere and Apache Flink

The official Apache Flink project website [1, 18] declares that *"Apache Flink is an open source platform for distributed stream and batch data processing.* [At its] *core* [it] *is a streaming data flow engine that provides data distribution, communication, and fault tolerance for distributed computations over data streams."*

Many of the original concepts in Stratosphere inspired or were carried over to Apache Flink [8, 23], such as the query optimizer, the streaming dataflow runtime, and the support for iterations. While others remained experimental features, such as the optimistic fault tolerance [32, 48]).

Apache Flink offers numerous additional features. For example, it: (i) provides consistent, exactly once guarantees for event time processing, (ii) is fault-tolerant, even for stateful operations, and (iii) is highly scalable, able to run on thousands of nodes with high throughput and low latency.

Additionally, Apache Flink offers several APIs. For example, the DataStream API applies transformations, such as filtering and aggregation on data streams. The Table API supports the composition of queries from relational operators, such as selection, filter, and join. Furthermore, Flink provides numerous libraries, such as the: CEP library for complex event-processing, FlinkML library for machine learning, and Gelly library for graph-processing. Using Flink's APIs and libraries, software developers are empowered to build and execute applications that run on Flink. Flink is increasingly gaining traction around the world. According to Alibaba [33], it is Flink's distinguishing technological capabilities that make it the *"most advanced stream processing engine today."*

2.2 Stream Processing

In recent years, our researchers have investigated how to enhance Apache Flink and related systems. For example, we devised a novel technique to address performance challenges faced when conducting aggregate sharing in data stream windows. Subsequently, we developed a prototype in Flink and demonstrated that our technique outperforms the state-of-the-art [19, 28].

Moreover, we developed optimizations to improve both the sharing of windows and computation for highly distributed setups [50], interactivity in streaming visualizations [51], and surveyed state management [49]. Currently, we are conducting research on the management of large-state for analytics that are beyond the capabilities of today's batch and stream processing engines.

2.3 Managing the Data Science Process

Reducing the entry barrier and cost of analyzing large amounts of data at scale requires the simplification of the data analysis process, which is today a grand challenge in data management research [42]. Addressing this demand, will require the development of a novel approach to automate the implementation decisions that data scientists routinely make, such as the decisions about the heterogeneous computing environments to employ. Particularly, since they are founded on a broad spectrum of theories, systems, and hardware solutions. Automated optimization, parallelization, and hardware-adaptation is a holy-grail of data science.

This grand challenge can be met, if we combine existing data processing technologies currently available in the scientific and systems community. The major obstacle to achieving automation is the absence of a principled model for scalable data science systems, akin to relational algebra in database systems. The key is to provide a declarative, algebraic, and optimizable representation for the entire data analysis process. To solve this problem, we need to integrate disparate hardware and software components present in today's data analysis architectures into a unified mosaic of systems, hardware devices, and theories that view analytics as graphs, matrices, or relations.

As a first step towards solving the automation problem, our researchers developed Emma [2, 7], a Scala DSL that enables holistic optimizations of data flow programs for scalable data analysis on Apache Flink and Apache Spark. As a result, developers can disregard the details of a platform-specific API, which reduces both program development and execution time. In 2015, Emma garnered an ACM SIGMOD Research Highlight Award.

Our researchers also introduced Lara, a deeply embedded language in Scala that enables developers to exploit optimizing transformations across linear and relational algebra operators [38] and physical operators, such as Blockjoin [39] to bridge relational and matrix representations and write scalable programs. Additionally, we devised a novel approach called ScootR [40] that significantly improves the performance of R programs executed in data flow systems, by establishing bidirectional ac-

cess between native user-defined functions and Flink's data structures. We also developed novel data handling methods, to better cope with large data sets and more efficiently yield visualizations by exploiting data aggregation approaches [34, 35]. The research conducted in this area received a VLDB best paper award in 2014.

Currently, we are investigating how to optimize the entire iterative data science process, from data source selection over information extraction and integration to data analysis, model building, model application, and visualization. Moreover, we are embarking on novel research in the areas of large-scale data analysis infrastructures, data management for the Internet of Things, data processing in the fog, end-to-end machine learning, information marketplaces, and technological enablers for responsible data management.

3. MODERN HARDWARE

The modern hardware landscape is rapidly evolving. Today, there are massively parallel processors with anywhere from hundreds to thousands of cores in graphic processing units (GPU) and the many integrated core (MIC) architecture, whose series of microarchitectures integrate many physical cores onto a single integrated circuit. Additionally, main memory costs have continued to drop, enabling a database to be stored in main memory. Network technologies, such as Infiniband and remote direct memory access (RDMA) provide low latency communication and low network bandwidth on the same order of magnitude as main memory bandwidth, as discussed by Binnig et al. in [10].

These novel technologies can accelerate data management by orders of magnitude, decrease computing costs by scaling-down cluster resources, and reduce the data to knowledge time. We conduct research in the modern hardware space to discover new ways to exploit these technologies.

3.1 Hardware Tailored Query Compilation

The power wall is arguably the defining limit of modern processor performance. Thus, vendors develop processor cores that are specialized to particular tasks, such as *ARM big.LITTLE*, a heterogeneous computing architecture. Alternatively, they develop processors that adhere to a new processor architecture, which is fundamentally different than classical CPUs, as discussed by Borkar and Chien [14] and Esmaeilzadeh et al. [22]. However, for data management systems it is difficult to exploit these heterogeneous processors. Instead, costly experts are required to re-implement and optimize query processors for new processor architectures.

To overcome this challenge, we launched the Hawk Project[3], in order to automatically exploit heterogeneous processors and increase performance in data management systems. The key problem is how to support many heterogeneous processors efficiently without having to rewrite code, for each new processor release. A problem that commonly arises with data management operators. Naturally, such an effort is both costly and error prone. Our aim is to enable data management systems to rewrite their code until they run optimally on a single processor.

W have developed a hardware-tailored code generator called Hawk [15] based on the CoGaDB system [16]. Hawk utilizes advanced query compilation strategies to produce custom code variants for each processor and query. By automatically exploring code variants, Hawk can tune generated code for each processor avoiding manual tuning and sidestepping the need for expensive experts.

3.2 Data Processing on Modern Processors

Over the past few years, we have been investigating alternative ways to leverage heterogeneous processor capabilities. For example, we explored code variants for selections and aggregations using an approach akin to micro adaptivity [46, 47]. We also implemented vectorized hashing primitives (e.g., gather, scatter, selective load, selective store) in OpenCL, to reduce code complexity and enable portability for both CPUs and MICs [9].

In addition, we devised a new approach to execute the k-means [41] algorithm more efficiently on GPUs and achieve a higher throughput (up to 20x over state-of-the-art approaches). Furthermore, we discovered how we can use the GPU memory hierarchy efficiently and developed compilation-based query processing strategies for massively parallel processors [27]. We also experimentally evaluated design aspects of current stream processing systems on modern hardware and found that the throughput of streaming systems on a single node can be improved by up to two orders of magnitude [53]. Our system Ocelot [31] is an OpenCL-based execution engine for the MonetDB main-memory database, which assesses efficiency in systems that completely rely on a hardware-oblivious code base. Finally, we investigated how to accelerate query optimization using massively parallel processors [29, 30, 37].

4. BENCHMARKING

Today's big data systems are designed to be scalable and meant to be run on a large number of

[3] `https://www.dfg-spp2037.de/ma4662-5`

nodes, in order to distribute workloads and speedup processing. Once these systems come to exist, benchmarking them is of paramount importance, to measure their performance under varying real-world scenarios. Historically, the Transaction Processing Performance Council's (TPC) benchmarks have enabled database researchers to meet their optimization goals. With the sheer-diversity of data processing systems under development at an ever-increasing pace, new benchmarking scenarios as well as novel tooling are required to properly assess system performance.

Our research in benchmarking data management systems includes designing, developing, and conducting performance surveys, devising novel measurement techniques and building software tools that implement them, and contributing to standardization efforts. Our objectives include identifying a system's capabilities and limitations as well as providing insight into functional areas, where additional investigation is required. Next, we present our benchmarking tools for big data, stream processing, and machine learning systems.

4.1 Benchmarking Big Data Systems

Our researchers have been instrumental in the development and standardization of several application-level benchmarks, including TPCx-BB [17], TPCx-IOT [44], and TPC-DS [45]. Typically, benchmark projects require a large number of configuration and data collection steps for the many experiments that need to be conducted. To simplify this tedious process, we have developed the Peel [11] framework. It automates the setup and deployment of big data frameworks, conducts benchmark experiments, gathers all system and performance data, and stores them in versioned repositories.

4.2 Benchmarking Stream Processing Systems

In recent years, there has been a surge in the number of novel stream processing systems (SPS). This poses numerous challenges for benchmarking due to the many subcomponents involved. In particular, since these may be outside of the boundary of the system under test and can easily become the predominant bottleneck. Recently, we demonstrated that all of the earlier benchmarking studies for SPS violated assumptions about the system setup. Consequently, since system's do not have control over incoming data streams, our experiments demonstrated [36] that the measurements reported in these studies both overestimated throughput and underestimated latency.

4.3 Benchmarking Machine Learning Systems

Machine learning has become ubiquitous for many data-driven applications. Since there is a natural trade-off between accuracy and performance in machine learning models, solely benchmarking the performance of machine learning systems is insufficient. Thus, we are currently conducting research to develop comprehensive benchmarks and build benchmarking tools that ensure reproducibility for machine learning systems.

The advent of big data processing systems, such as Hadoop and next generation systems, such as Apache Spark have quickly spurred interest in implementing more complex analytics jobs (beyond simple indexing or sorting) on these scalable systems. Among these complex analytics are Apache Mahout and SparkML. Although these libraries similarly feature weak-scaling capabilities as simple processing jobs, they do not scale in other ways (e.g., in terms of model dimensionality [13]). Furthermore, many evaluations today compare their machine learning systems against weak baselines, such as simple and highly-inefficient Hadoop implementations that are easily outperformed by state of the art single-node machine learning libraries [12].

5. DATA INTEGRATION

Increasingly, organizations want to obtain value from their disparate datasets. To do so, they inject all of their data into *data lakes*, to make the data available for analysis. Although this approach solves the data access problem, another challenge for effective data analysis remains: metadata about datasets is often missing or poorly documented and data scientists rarely possess comprehensive knowledge about the data lake. Today, data discovery, data integration, and data cleaning are factually the most time-consuming and least enjoyable tasks for data scientists [20]. In addition to small contributions to the field of entity resolution [21, 43], our research in the area of data integration tackles two general challenges, i.e., data discovery in data lakes and iterative data preparation.

5.1 Data Lake Management

Despite the presence of data lakes, discovering the data of actual interest is still very challenging [25]. Due to the abundance of data without a central owner, a holistic organization of these datasets via ETL is infeasible. As a compromise, a proposed solution is to generate easy to obtain metadata [5] from datasets in the data lake and infer relationships, such as foreign/primary key relationships and

other types of inter-column similarities.

Jointly, with colleagues from Massachusetts Institute of Technology (MIT), Qatar Computing Research Institute (QCRI), and the University of Waterloo, we built *Data Civilizer* [20, 26], an end-to-end big data management system. Data Civilizer incorporates a data discovery component called *Aurum* [24] that achieves the aforementioned functionalities and efficiently identifies column similarities and overlaps. However, these types of heuristics can lead to the generation of many false positives. In particular, numerical columns, such as ID columns are often quite similar. Currently, we aim to solve this problem, by developing new heuristics that disambiguate those columns more accurately.

5.2 Iterative Data Cleaning

While there has been a huge body of work in the area of data cleaning, most data practitioners resort to custom data wrangling scripts. The main reason behind this is that there is still no one-size-fits-all system for data cleaning [4]. Algorithms tackle very specific types of errors, such as rule or pattern violations and outliers. As a result, the data scientist will undergo an iterative try-and-error procedure, which is time-consuming. Additionally, most of these algorithms require some sort of hyperparameter or a set of given patterns/rules, which may be unavailable. To bridge this gap, we treat data cleaning as an iterative process and preserve the history of previously performed cleaning tasks, to minimize the overall user-effort and identify the right set of cleaning routines and their respective configurations for the task at hand. Furthermore, have studied several aggregation methods to combine the effectiveness of varying error detection strategies [52].

6. GRANTS, ALLIANCES, AND SERVICE

Our research activities are funded through grants obtained from varying national, international, and industry sources, including the German Federal Ministry of Education and Research (BMBF), the German Federal Ministry for Economic Affairs and Energy (BMWi), the German Federal Ministry of Transport and Digital Infrastructure (BMVI), the German Research Foundation (DFG), and the European Union, among others. Most notably, we are coordinating two German flagship big data projects, the Berlin Big Data Center[4] (BBDC) and the Smart Data Forum[5] as well as co-directing the Berlin Center for Machine Learning (BZML).

We have transferred our research into numerous commercial products and open-source systems. We closely collaborate with many leading information management companies and have created several startups based on our research. We also contribute to our governmental and scientific communities. For example, we serve as grant reviewers and on expert panels at national and international funding agencies and provide expert advice to government agencies in Germany and the EU.

Our researchers support the database community on various levels. In 2013, Volker participated in the Beckman Database Research Self-Assessment Meeting to discuss the state of database research and offer perspectives on key directions for future research [3]. Since 2018, Volker is President of the VLDB Endowment. Furthermore, we have hosted an EDBT conference and served as conference officers and program committees for VLDB, SIGMOD, and ICDE, among others.

[4] `https://www.bbdc.berlin`

[5] `https://smartdataforum.de`

7. REFERENCES

[1] Official apache flink website. `https://flink.apache.org/`.

[2] Official website of the emma language. `https://github.com/emmalanguage/emma`.

[3] D. Abadi, R. Agrawal, et al. The beckman report on database research. *Communications of the ACM*, 59(2):92–99, Jan. 2016.

[4] Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang. Detecting data errors: Where are we and what needs to be done? *PVLDB*, 9(12):993–1004, Aug. 2016.

[5] Z. Abedjan, L. Golab, and F. Naumann. Profiling relational data: a survey. *VLDBJ*, 24(4):557–581, 2015.

[6] A. Alexandrov et al. The stratosphere platform for big data analytics. *VLDBJ*, 23(6):939–964, 2014.

[7] A. Alexandrov, A. Kunft, A. Katsifodimos, F. Schüler, L. Thamsen, O. Kao, T. Herb, and V. Markl. Implicit parallelism through deep language embedding. In *SIGMOD*, pages 47–61, 2015.

[8] D. Battré, S. Ewen, F. Hueske, O. Kao, V. Markl, and D. Warneke. Nephele/pacts: a programming model and execution framework for web-scale analytical processing. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC)*, pages 119–130, 2010.

[9] T. Behrens, V. Rosenfeld, J. Traub, S. Breß, and V. Markl. Efficient SIMD vectorization for hashing in opencl. In *EDBT*, pages 489–492, 2018.

[10] C. Binnig, A. Crotty, A. Galakatos, T. Kraska, and E. Zamanian. The end of slow networks: It's time for a redesign. *PVLDB*, 9(7):528–539, 2016.

[11] C. Boden, A. Alexandrov, A. Kunft, T. Rabl, and V. Markl. PEEL: A framework for benchmarking distributed systems and algorithms. In *TPCTC*, pages 9–24, 2017.

[12] C. Boden, T. Rabl, and V. Markl. Distributed machine learning - but at what cost? In *Workshop on ML Systems at NIPS*, 2017.

[13] C. Boden, A. Spina, T. Rabl, and V. Markl. Benchmarking data flow systems for scalable machine learning. In *Proceedings of the 4th ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and Beyond, BeyondMR@SIGMOD 2017, Chicago, IL, USA, May 19, 2017*, pages 5:1–5:10, 2017.

[14] S. Borkar and A. Chien. The future of microprocessors. *Communications of the ACM*, 54(5):67–77, 2011.

[15] S. Breß et al. Generating custom code for efficient query execution on heterogeneous processors. *VLDBJ*, 2018.

[16] S. Breß, H. Funke, and J. Teubner. Robust query processing in co-processor-accelerated databases. In *SIGMOD*, pages 1891–1906, 2016.

[17] P. Cao, B. Gowda, S. Lakshmi, C. Narasimhadevara, P. Nguyen, J. Poelman, M. Poess, and T. Rabl. From bigbench to tpcx-bb: Standardization of a big data benchmark. In *Performance Evaluation and Benchmarking. Traditional - Big Data - Interest of Things - 8th TPC Technology Conference, TPCTC 2016, New Delhi, India, September 5-9, 2016, Revised Selected Papers*, pages 24–44, 2016.

[18] P. Carbone et al. Apache flink^{TM}: Stream and batch processing in a single engine. *IEEE Data Eng. Bull.*, 38(4):28–38, 2015.

[19] P. Carbone, J. Traub, A. Katsifodimos, S. Haridi, and V. Markl. Cutty: Aggregate sharing for user-defined windows. In *CIKM*, pages 1201–1210, 2016.

[20] D. Deng, R. C. Fernandez, Z. Abedjan, S. Wang, M. Stonebraker, A. Elmagarmid, I. Ilyas, S. Madden, M. Ouzzani, and N. Tang. The data civilizer system. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2017.

[21] D. Deng, W. Tao, Z. Abedjan, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. Unsupervised string transformation learning for entity consolidation. In *ICDE*, 2019.

[22] Esmaeilzadeh et al. Dark silicon and the end of multicore scaling. In *ISCA*, pages 365–376, 2011.

[23] S. Ewen, K. Tzoumas, M. Kaufmann, and V. Markl. Spinning fast iterative data flows. *PVLDB*, 5(11):1268–1279, 2012.

[24] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker. Aurum: A data discovery system. In *ICDE*, 2018.

[25] R. C. Fernandez, Z. Abedjan, S. Madden, and M. Stonebraker. Towards large-scale data discovery: position paper. In *International Workshop on Exploratory Search in Databases and the Web (ExploreDB)*, pages 3–5, 2016.

[26] R. C. Fernandez, D. Deng, E. Mansour, A. A. Qahtan, W. Tao, Z. Abedjan, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. A demo of the data civilizer system. In *SIGMOD*, pages 1639–1642, 2017.

[27] H. Funke, S. Breß, S. Noll, V. Markl, and J. Teubner. Pipelined query processing in coprocessor environments. In *SIGMOD*, 2018.

[28] P. M. Grulich, R. Saitenmacher, J. Traub, S. Breß, T. Rabl, and V. Markl. Scalable detection of concept drifts on data streams with parallel adaptive windowing. In *EDBT*, pages 477–480, 2018.

[29] M. Heimel, M. Kiefer, and V. Markl. Self-tuning, gpu-accelerated kernel density models for multidimensional selectivity estimation. In *SIGMOD*, pages 1477–1492, 2015.

[30] M. Heimel and V. Markl. A first step towards gpu-assisted query optimization. In *ADMS@VLDB*, pages 33–44, 2012.

[31] M. Heimel, M. Saecker, H. Pirk, S. Manegold, and V. Markl. Hardware-oblivious parallelism for in-memory column-stores. *PVLDB*, 6(9):709–720, 2013.

[32] F. Hueske, M. Peters, M. Sax, A. Rheinländer, R. Bergmann, A. Krettek, and K. Tzoumas. Opening the black boxes in data flow optimization. *PVLDB*, 5(11):1256–1267, 2012.

[33] X. Jiang. Unified engine for data processing and ai. `https://berlin-2018.flink-forward.org/conference-program/#unified-engine-for-data-processing-and-ai`. Alibaba, Flink Forward, Berlin, 2018.

[34] U. Jugel, Z. Jerzak, G. Hackenbroich, and V. Markl. M4: A visualization-oriented time series data aggregation. *PVLDB*, 7(10):797–808, 2014.

[35] U. Jugel, Z. Jerzak, G. Hackenbroich, and V. Markl. Vdda: Automatic visualization-driven data aggregation in relational databases. *VLDBJ*, 25(1):53–77, 2016.

[36] J. Karimov, T. Rabl, A. Katsifodimos, R. Samarev, H. Heiskanen, and V. Markl. Benchmarking distributed stream processing engines. In *ICDE*, 2018.

[37] M. Kiefer, M. Heimel, S. Breß, and V. Markl. Estimating join selectivities using bandwidth-optimized kernel density models. *PVLDB*, 10(13):2085–2096, 2017.

[38] A. Kunft, A. Alexandrov, A. Katsifodimos, and V. Markl. Bridging the gap: towards optimization across linear and relational algebra. In *BeyondMR@SIGMOD 2016*, page 1, 2016.

[39] A. Kunft, A. Katsifodimos, S. Schelter, T. Rabl, and V. Markl. Blockjoin: Efficient matrix partitioning through joins. *PVLDB*, 10(13):2061–2072, 2017.

[40] A. Kunft, L. Stadler, D. Bonetta, C. Basca, J. Meiners, S. Breß, T. Rabl, J. J. Fumero, and V. Markl. Scootr: Scaling R dataframes on dataflow systems. In *SOCC*, pages 288–300, 2018.

[41] C. Lutz, S. Breß, T. Rabl, S. Zeuch, and V. Markl. Efficient and scalable k-means on GPUs. *Datenbank-Spektrum*, 2018.

[42] V. Markl. Breaking the chains: On declarative data analysis and data independence in the big data era. *PVLDB*, 7(13):1730–1733, Aug. 2014.

[43] Öykü Özlem Çakal, M. Mahdavi, and Z. Abedjan. Clrl: Feature engineering forcross-language record linkage. In *EDBT*, 2019.

[44] M. Poess, R. Nambiar, C. Narasimhadevara, K. Kulkarni, T. Rabl, and H.-A. Jacobsen. Analysis of tpcx-iot: The first industry standard benchmark for iot gateway systems. In *ICDE*, 2018.

[45] M. Poess, T. Rabl, and H. Jacobsen. Analysis of TPC-DS: the first standard benchmark for sql-based big data systems. In *SOCC*, pages 573–585, 2017.

[46] V. Rosenfeld, M. Heimel, C. Viebig, and V. Markl. The operator variant selection problem on heterogeneous hardware. In *ADMS@VLDB*. VLDB Endowment, 2015.

[47] B. Răducanu et al. Micro adaptivity in Vectorwise. In *SIGMOD*, pages 1231–1242, 2013.

[48] S. Schelter, S. Ewen, K. Tzoumas, and V. Markl. "all roads lead to rome": optimistic recovery for distributed iterative data processing. In *CIKM*, pages 1919–1928, 2013.

[49] Q.-C. To, J. Soto, and V. Markl. A survey of state management in big data processing systems. *VLDBJ*, Aug 2018.

[50] J. Traub, S. Breß, T. Rabl, A. Katsifodimos, and V. Markl. Optimized on-demand data streaming from sensor nodes. In *SoCC*, pages 586–597, 2017.

[51] J. Traub, N. Steenbergen, P. Grulich, T. Rabl, and V. Markl. I^2: Interactive real-time visualization for streaming data. In *EDBT*, pages 526–529, 2017.

[52] L. Visengeriyeva and Z. Abedjan. Metadata-driven error detection. In *SSDBM*, 2018.

[53] S. Zeuch, B. Del Monte, J. Karimov, C. Lutz, M. Renz, J. Traub, S. Breß, T. Rabl, and V. Markl. Analyzing efficient stream processing on modern hardware. *PVLDB*, 12(5):516–530, 2018.