Report from the Fourth Workshop on Algorithms and Systems for MapReduce and Beyond (BeyondMR'17)

Foto N. Afrati
National Technical University of Athens, Greece
afrati@softlab.ntua.gr

Jan Hidders
Vrije Universiteit Brussel, Belgium
jan.hidders@vub.be

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

Jacek Sroka
University of Warsaw, Poland
j.sroka@mimuw.edu.pl

Jeffrey Ullman
Stanford University, USA
ullman@cs.stanford.edu

ABSTRACT

This report summarizes the presentations and discussions of the fourth workshop on Algorithms and Systems for MapReduce and Beyond (BeyondMR'17). The BeyondMR workshop was held in conjunction with the 2017 SIGMOD/PODS conference in Chicago, Illinois, USA on Friday May 19, 2017. The goal of the workshop was to bring together researchers and practitioners to explore algorithms, computational models, languages and interfaces for systems that provide large-scale parallelization and fault tolerance. These include specialized programming and data-management systems based on MapReduce and extensions thereof, graph processing systems and data-intensive workflow systems.

The program featured two well-attended invited talks, the first on current and future development in big data processing by Matei Zaharia from Databricks and the University of Stanford, and the second on computational models for the analysis and development of big data processing algorithms by Ke Yi from the Hong Kong University of Science and Technology.

1. INTRODUCTION

In this edition of BeyondMR four main themes emerged: (*i*) *languages and interfaces*, which investigates new interfaces and languages for specifying data processing workflows to increase usability, which includes both high-level and declarative languages, as well as more low-level workflow evaluation plans, (*ii*) *algorithms*, which involves the design, evaluation and analysis of algorithms for large-scale data processing and (*iii*) *integration*, which concerns the integration of different types of analytics frameworks, e.g., for database-oriented analytics and for linear algebra.

Both keynotes covered the aforementioned themes. The keynote by Matei Zaharia addressed *usability*, *integration* and *hardware*, by describing recent and future developments of the *Spark* framework, and the keynote by Ke Yi gave an overview of computational models for large-scale parallel *algorithms*.

The presented papers addressed the themes as follows. Paper [13] covered the themes *languages and interfaces* and *algorithms* by presenting an approach where spreadsheets are used as the programming interface for a large-scale data-processing framework with special algorithms for executing typical spreadsheet data-manipulation operations. Paper [7] tackles the themes *languages and interfaces* and *integration* by presenting an integrated algebra for implementing and optimizing data-processing workflows with both relational algebra and linear algebra operators. In paper [10] the themes *languages and interfaces* and *algorithms* are both addressed by presenting distributed algorithms and implementations for computing the closure of certain OWL ontologies. The same themes also return in paper [12] which discusses distributed algorithms and implementations for computing navigation queries over RDF graphs formulated in a high-level query language. The theme *algorithms* was again covered in paper [2] which discussed benchmarking dataflow systems for the purpose of machine learning, by emphasizing scalability for high-dimensional but sparse data. In paper [8] the themes *languages and interfaces*, *algorithms* and *integration* were addressed by presenting a framework translating high-level specifications of data processing pipelines to different evaluation plans, making different choices concerning parallelization frameworks, acceleration frameworks and data-processing platforms. Furthermore, the theme *algorithms* was covered by paper [4] which discusses algorithms for matrix multiplication on MapReduce-like data processing platforms. The theme was also addressed by paper [6], presenting streaming algorithms for multi-way theta-joins, and by paper [3], presenting distributed algorithms for binning in big genomic datasets.

We will give a summary of the two keynotes in Section 2 followed by a description of the contributions of the presented papers further in Section 3.

2. SUMMARY OF KEYNOTES

We give a summary of the two keynotes, which contributed to visibility of the workshop.

What's Changing in Big Data?

The first keynote was presented by Matei Zaharia, Stanford University, USA, and Databricks. His presentation discussed the main changes to big-data processing in the last 10 years, as he has experienced as codeveloper of Spark and chief technologist of Databricks. It focused on three developments, namely (1) the user base moving from software engineers to data analysts, (2) the changing hardware and the computational bottleneck moving from I/O to the computation and (3) the delivery of big-data processing services through cloud computing.

The discussion of the changing user-base started with the observation that usability had been, and remains, a crucial factor in the success of systems such as Spark. The changing user base shows in the selection of programming languages for Spark, where a shift is seen from *Scala* and *Java* to *Python* and *R*. Three areas where there are currently usability problems are (a) the understandability of the API, (b) the complexity of performance predictability, and (c) the difficulty in accessing the system from smaller front-end tools such as Tableau and Excel. These problems are addressed by Spark projects such as *DataFrames* and *Spark SQL*. The first explicitly deals with structured data as it appears in *Python* and *R*, and the second introduces a high-level API for SQL-like processing with database-like query optimization. Next to that, there are also projects such as *ML Pipelines* for machine-learning pipelines, *graphFrames* for graph processing, and the introduction of streaming over DataFrames.

The changes in hardware in the last 10 years were summarized by the observations that although the size of storage and the speed of networks has increased by an order of magnitude, the speed of CPUs has not. Moreover, *GPU*s and *FPGA*s have become more ubiquitous and powerful, and so need to be better utilized. These concerns are addressed in the projects *Tungsten* [16] and *Weld* [11]. The first tries to make better use of the hardware by run-time code generation that exploits cache locality and uses off-heap memory management. The second allows the easy integration of different analytics libraries by offering a common algebra for relational algebra operations, linear algebra operations, and graph operations.

The final change discussed was that big-data processing services are increasingly provided through cloud computing. This provides new challenges as well as opportunities. New challenges are scaling up state management and stress testing, but opportunities are created by fast roll-out of new functionality and consequently rapid comprehensive feedback.

At the end of the presentation research challenges were discussed. These included new types of interfaces for new types of users, new optimization techniques to deal with new types and configurations of hardware, the integration of different processing platforms and the support of interaction between users of such platforms.

Sequential vs Parallel, Fine-grained vs Coarse-grained, Models and Algorithms

Ke Yi, Hong Kong University of Science and Technology, China, was the second keynote speaker. His talk focused on computational models for approximating the running time of distributed algorithms. He warned at the start, by quoting Gorge Box, *All models are wrong, but some are useful.* After this, the talk started with a quick introduction to *RAM* and *PRAM models*, emphasizing the *Concurrent Read / Exclusive Read* (CREW) and *Concurrent Write/ Exclusive Write* (EREW) variants. He also presented the *External Memory Model* (EM) where internal memory is limited to size M and the data is transferred between internal and external memory in blocks of size B. This part of the presentation was concluded by discussing the known relationships between those classes and with their classification according to two independent criteria, namely if the model is fine-grained or coarse-grained and if it is parallel or sequential.

Next, the *Parallel External-Memory* (PEM) model for multicore GPUs [1] was presented together followed by the *Bulk-Synchronous Parallel* (BSP) model. The latter is a shared-nothing, coarse-grained parallel model [15] that is well suited for models used in practice like MapReduce and Pregel. Simplifications and variants were presented and the models were compared. Also, methods to simulate one model by the other were discussed, as well as the relations between BSP, PRAM and EM. It was concluded that it is unlikely that optimal BSP algorithms are produced by simulating the fine-grained PRAM or the sequential EM model.

The presentation concluded with a discussion of how to design algorithms for those models. *Work-depth models* and *Brent's theorem* were recalled. Finally, the multithreaded and external memory versions of work-depth models were presented, and several problems were discussed for join algorithms, such as the nested-loop and hypercube algorithms.

3. REVIEW OF PRESENTED WORK

(short paper) Towards minimal algorithms for big data analytics with spreadsheets

The paper [13] presents research done by Jacek Sroka, Artur Leśniewski, Mirosław Kowaluk, Krzysztof Stencel and Jerzy Tyszkiewicz, from the Institute of Informatics at the University of Warsaw. The contribution of the paper is an algorithm to answer bulk range queries. The idea is based on a range tree, but adapted to work on datasets that are too big for processing on a single machine. The algorithm is organized so that the computation is perfectly balanced between the nodes in the cluster. It also allows answering queries in bulk, i.e., a large number of queries at the same time.

The motivation for the algorithm is to significantly lower the technological barrier, which currently prevents average users from analysing big datasets available as CSV files. The authors propose creation of a tool, which translates spreadsheets of regular structure into semantically equivalent MapReduce or Spark computations. Such a tool would enable users to prepare spreadsheets on a small sample of the data and then automatically translate them into MapReduce or Spark to be executed on the full dataset. As it is estimated that spreadsheet programmers outnumber the programmers of all other languages together, achieving that goal would allow a huge number of new users to benefit from big-data processing.

(full paper) LaraDB: A Minimalist Kernel for Linear and Relational Algebra Computation

The paper [7] was produced by Dylan Hutchison, Bill Howe and Dan Suciu, from the Department of Computer Science and Engineering at the University of Washington. It presents *Lara*, an algebra consisting of three operators that expresses *Relational Algebra* (RA) and *Linear Algebra* (LA), as well as optimization rules for a certain programming model. A proof is provided that the algebra is more explicit then MapReduce, but also more general then RA and LA. The practicality and efficiency of the algebra is demonstrated by implementing its operators using range scans over partitioned, sorted maps, a primitive that is available in a wide range of back-end engines. Concretely, the operators were implemented as range iterators in *Apache Accumulo*, an implementation of Google's *BigTable*. This implementation is shown to outperform the *Accumulo*'s native MapReduce integration for mixed-abstraction workflows involving joins and aggregations in the form of matrix multiplication.

(full paper) SPOWL: Spark-based OWL 2 Reasoning Materialisation

The paper [10] was the result of work by Yu Liu and Peter Mcbrien, from the Department of Computing at Imperial College London. It presents *SPOWL*, which uses *Spark* to implement OWL inferencing over large ontologies. This system compiles T-Box axioms to *Spark* programs, which are iteratively executed to compute and materialize the closure of the ontology. The result can, for example, be used to compute SPARQL queries. The system leverages the advantages of *Spark* by caching results in distributed memory and parallelizing jobs in a more flexible manner. It also optimises the number of necessary iterations by analysing the dependencies in the T-Box hierarchy and compiling the axioms correspondingly. The performance of *SPOWL* is evaluated against *WepPIE* on *LUBM* datasets, and is shown to be generally comparable or better.

(full paper) Querying Semantic Knowledge Bases with SQL-on-Hadoop

The paper [12] describes research done by Martin Przyjaciel-Zablocki, Alexander Schätzle and Georg Lausen, in the Databases and Information Systems group, at the Institut für Informatik, in the Albert-Ludwigs-Universität Freiburg. The paper presents a new processor for *Trial-QL*, an SQL-like query language for RDF that is based on the *Triple Algebra with Recursion* (TriAL*) [9] and therefore especially suited to express navigational queries. The presented *Trial-QL* processor is built on top of *Impala* (a massive parallel SQL Engine on *Hadoop*) and *Spark*, using one unified data storage system based on *Parquet* and *HDFS*. The paper studies and compares different evaluation algorithms and storage strategies, and compares the performance to other RDF management systems such as *Sempala*, *Neo4j*, *PigSPARQL*, *H2RDF+* and *SHARD*. Interactive and competitive response times are shown on datasets of more than a billion triples, and even results with more than 25 billion triples could be produced in minutes.

(full paper) Benchmarking Data Flow Systems for Scalable Machine Learning

The paper [2] presents the contribution by Christoph Boden, Andrea Spina, Tilmann Rabl and Volker Markl, in the Database Systems and Information Management Group at the TU Berlin. The paper investigates the scalability of data-flow systems such as *Apache Flink* and *Apache Spark* for typical machine-learning workflows. The workloads are assumed to differ from conventional workloads as are found in existing benchmarks, which are often based on tasks such as *WordCount*, *Grep* or *Sort*. The main differences are that the data tends to be more sparse but at the same high dimensional, so the scalability in the number of dimensions becomes more important. The paper therefore presents a representative set of distributed machine-learning algorithms suitable for large-scale distributed settings that have close resemblance to industry-relevant applications and provide generalizable insights into system performance. These algorithms were implemented in *Apache Spark* and *Apache Flink*, and after tuning, the two systems were compared in scalability in both the size of the data and the dimensionality of the data. Measurements were done with datasets containing up to 4 billion data points and 100 million dimensions. The main conclusion is that current systems are surprisingly inefficient in dealing with such high-dimensional data.

(full paper) A containerized analytics framework for data- and compute-intensive pipeline applications

The paper [8] describes the result obtained by Yuriy Kaniovskyi, Martin Köhler and Siegfried Benkner, in the Research Group Scientific Computing at the University of Vienna and at the School of Computer Science in the University of Manchester. It presents a framework for executing high-level specifications of data-processing pipelines on heterogeneous architectures using a variety of programming paradigms. The presented framework allows several ways of executing each step in the pipeline, using different parallellization libraries (such as *OpenMP*, *TBB*, *MPI* and *Cilk*), different acceleration frameworks (such as *OpenACC*, *CUDA* and *OpenCL*) or different data-processing platforms (e.g., *MapReduce*, *Spark*, *Storm* and *Flink*). Moreover, the intermediate results might be passed in different ways, e.g., by storing it in *HDFS*, in a local file system, in memory, or by streaming it.

The approach starts with a high-level language for describing the global data-processing pipeline. It also allows the separate specification of the different implementation variants for each step in the pipeline, together with their computational requirements. These are self-contained execution units that can encapsulate legacy, native or accelerator-based code. This enables the framework to fuse data-intensive programming paradigms (via native *YARN* applications) with computation-intensive programming paradigms (via containerization). The framework then optimizes the executing plan by taking into account the specified *performance models* and the available resources as specified in a specially developed *generic resource description framework*.

(full paper) MapReduce Implementation of Strassen's Algorithm for Matrix Multiplication

The paper [4] presents the work by Minhao Deng and Prakash Ramanan, in the Electrical Engineering and Computer Science department at Whichita State University. It studies MapReduce implementations of Strassen's algorithm for multiplying two $n \times n$ matrices, introduced by Volker Strassen in [14]. The Strassen algorithm has time complexity $\Theta(n^{\log_2(7)})$, which is an improvement over the standard algorithm since the latter has time complexity $\Theta(n^3)$ and $\log_2(7) \approx 2.81$.

When considering implementations on MapReduce of the straightforward algorithm, it has been shown that this can be done in typically one or two passes. The paper investigates in how many passes the Strassen algorithm can be implemented. This algorithm consists of three parts: *Part I* where sums and differences of quadrants of the input matrices are computed, *Part II* where several products are computed of the matrices from Part I and *Part III* where each quadrant of the final matrix is computed by taking sums and differences of the matrices from Part II.

Direct implementation of these parts in MapReduce is shown to be possible in $\log_2(n)$, 1 and $\log_2(n)$ passes, respectively. Moreover, it is shown that *Part I* can be implemented in 2 passes, with total work $\Theta(n \log_2(7))$, and reducer size and reducer workload $o(n)$. It is conjectured that *Part III* can also be implemented in a constant number of passes and with small reducer size and workload.

(short paper) Scaling Out Continuous Multi-Way Theta Joins

The paper [6] presents work by Manuel Hoffmann and Sebastian Michel, performed at the Databases and Information Systems Group at the department of computer science at the University of Kaiserslautern. The paper presents generic tuple-routing schemes for computing distributed multiway theta-joins over streaming data. These schemes are implemented in an architecture where query plans consist of logical operators to *Apache Storm* topologies. The paper reports the first results for the *TPC-H* benchmark where the topologies are executed on *Amazon EC2* instances.

(short paper) Bi-Dimensional Binning for Big Genomic Datasets

The paper [3] describes results obtained by Pietro Pinoli, Simone Cattani, Stefano Ceri and Abdulrahman Kaitoua, in the Department of Electronics, Information, and Bioengineering at the Politecnico di Milano. The main result is a bi-dimensional binning strategy for the *SciDB* array database, motivated by implementation of the *GenoMetric Query Language* (GMQL) — a high-level query language for genomics — that the authors previously developed. Due to the array database particularities, it is not possible to dynamically split a region and distribute its replicas to an arbitrary number of adjacent cells. Yet in order to apply a binning strategy, all the regions need to be replicated an identical number of times. The authors propose to organize the bins in a two dimensional grid. In such a grid the values are located above the diagonal and it is possible to define spaces that would contain values for one bin without replicating the values. Such bins (spaces) can share values because they share some grid cells.

4. CONCLUSION

The presentations and keynotes at BeyondMR'17 provided an overview of current developments and emerging issues in the area of algorithms, computational models, architectures and interfaces for systems that provide large-scale parallelization. The workshop attracted 17 submissions from which the program committee led by Paris Koutris from the University of Wisconsin-Madison accepted 6 full papers and 3 short papers.

Keynotes and papers covered topics such as user interfaces, integration of programming paradigms, algorithmics and leveraging new hardware features. The contributions suggest that while MapReduce has been extended and replaced by newer models, there is an active area of research centred around data-management systems based on MapReduce and extensions, providing ever more insight for developing more effective graph processing systems and data-intensive workflow systems.

The sessions were well attended with an average of 35 participants, and the proceedings are published in the ACM Digital Library [5].

Acknowledgements: We would like to thank PC members, keynote speakers, authors, local workshop organizers and attendees for making BeyondMR 2017 a success. We also express our appreciation for the support from Google Inc. Finally we would like to thank the anonymous reviewers for their constructive remarks on how to improve this report.

5. REFERENCES

[1] Lars Arge, Michael T. Goodrich, Michael Nelson, and Nodari Sitchinava. Fundamental parallel algorithms for private-cache chip multiprocessors. In *Proc. of SPAA '08*, pages 197–206, New York, NY, USA, 2008. ACM.
[2] Christoph Boden, Andrea Spina, Tilmann Rabl, and Volker Markl. Benchmarking data flow systems for scalable machine learning. In Hidders [5], pages 5:1–5:10.
[3] Simone Cattani, Stefano Ceri, Abdulrahman Kaitoua, and Pietro Pinoli. Bi-dimensional binning for big genomic datasets. In Hidders [5], pages 9:1–9:4.
[4] Minhao Deng and Prakash Ramanan. Mapreduce implementation of Strassen's algorithm for matrix multiplication. In Hidders [5], pages 7:1–7:10.
[5] Jan Hidders, editor. *Proc. of BeyondMR'17*, New York, NY, USA, 2017. ACM.
[6] Manuel Hoffmann and Sebastian Michel. Scaling out continuous multi-way theta-joins. In Hidders [5], pages 8:1–8:4.
[7] Dylan Hutchison, Bill Howe, and Dan Suciu. LaraDB: A minimalist kernel for linear and relational algebra computation. In Hidders [5], pages 2:1–2:10.
[8] Yuriy Kaniovskyi, Martin Koehler, and Siegfried Benkner. A containerized analytics framework for data and compute-intensive pipeline applications. In Hidders [5], pages 6:1–6:10.
[9] Leonid Libkin, Juan Reutter, and Domagoj Vrgoč. Trial for RDF: Adapting graph query languages for RDF data. In *Proc. of PODS '13*, pages 201–212, New York, NY, USA, 2013. ACM.
[10] Yu Liu and Peter McBrien. SPOWL: Spark-based OWL 2 reasoning materialisation. In Hidders [5], pages 3:1–3:10.
[11] Shoumik Palkar, James J. Thomas, Anil Shanbhag, Malte Schwarzkopt, Saman P. Amarasinghe, and Matei Zaharia. Weld: A common runtime for high performance data analytics. In *Proc. of CIDR 2017*. www.cidrdb.org, January 2017.
[12] Martin Przyjaciel-Zablocki, Alexander Schätzle, and Georg Lausen. Querying semantic knowledge bases with SQL-on-Hadoop. In Hidders [5], pages 4:1–4:10.
[13] Jacek Sroka, Artur Leśniewski, Mirosław Kowaluk, Krzysztof Stencel, and Jerzy Tyszkiewicz. Towards minimal algorithms for big data analytics with spreadsheets. In Hidders [5], pages 1:1–1:4.
[14] Volker Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13(4):354–356, August 1969.
[15] Leslie G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, August 1990.
[16] Reynold Xin and Josh Rosen. Project Tungsten: Bringing Apache Spark closer to bare metal. `https://databricks.com/blog/2015/04/28/project-tungsten-bringing-spark-closer-to-bare-metal.html`, April 2015.