Issues in Big Data Testing and Benchmarking

Alexander Alexandrov
Technische Universität Berlin
Einsteinufer 17
10587 Berlin, Germany
+49 30 314 23555
alexander.alexandrov@tu-berlin.de

Christoph Brücke
Technische Universität Berlin
Einsteinufer 17
10587 Berlin, Germany
+49 30 314 23555
christoph.bruecke@campus.tu-berlin.de

Volker Markl
Technische Universität Berlin
Einsteinufer 17
10587 Berlin, Germany
+49 30 314 23555
volker.markl@tu-berlin.de

ABSTRACT
The academic community and industry are currently researching and building next generation data management systems. These systems are designed to analyze data sets of high volume with high data ingest rates and short response times executing complex data analysis algorithms on data that does not adhere to relational data models. As these big data systems differ from standard relational database systems with respect to data and workloads, the traditional benchmarks used by the database community are insufficient. In this paper, we describe initial solutions and challenges with respect to big data generation, methods for creating realistic, privacy-aware, and arbitrarily scalable data sets, workloads, and benchmarks from real world data. We will in particular discuss why we feel that workloads currently discussed in the testing and benchmarking community do not capture the real complexity of big data and highlight several research challenges with respect to massively-parallel data generation and data characterization.

Categories and Subject Descriptors
D.2.5 [**Testing and Debugging**]: *testing tools, data generators*

General Terms
Measurement, Performance, Experimentation

Keywords
Big Data, Data Generation, Data Profiling, Workloads, Benchmarking

1. INTRODUCTION
The database systems building community is currently at a peak of new activity, creating novel systems for managing and analyzing what is commonly called "big data." Big data is usually characterized by the requirement to conduct advanced analytics on large volumes of data of variable format, which is ingested into the system with high-velocity with the need for fast response times. Novel big data analytics systems differ from traditional data analysis systems for varying reasons, they: (a) can process terabytes or even petabytes of data due to their scale-out abilities, employing massively parallel processing, (b) support complex data types in addition to relational sets of tuples (i.e., data of complex structure, such as text documents, hierarchies, graphs, or even images, audio, or video files), (c) allow for defining and processing complex analytics tasks that go beyond the traditional operations of the relational algebra (e.g., user-defined functions, data mining or machine learning algorithms, graph algorithms), (d) provide fault-tolerance in order to ensure termination even for long-running computations, and (e) compute answers with low-latency, producing results in a pipelined fashion.

Some examples of systems that showcase several of these features are Google MapReduce [DG04], its open source implementation Hadoop [Had13], its ecosystem of languages (e.g., Hive [TSJ+09], JAQL [BEG+11], Pig [ORS+08]) and libraries such as Mahout [Mah13], and other big data systems such as Asterix [ABG+12], GraphLab [LBG+12], Spark [Spa13] and our own Stratosphere system [ABE+10, Str13]. At the same time, there is a trend to make more traditional relational data analysis systems more scalable. Examples of these efforts are SAP Hana [FML+12], Impala [Imp13], Oracle Exadata [GSA+11], or the columnar storage extensions to Microsoft's and IBM's database products, to name a few.

While all these systems have advanced the capabilities of data analysis with respect to the five dimensions above, database testing and benchmarking have not moved forward to provide data generators, data sets, and workloads. In particular, we see the need to generate large, realistic data sets at scale, as well as the need for well-defined workloads that capture the nature of novel, modern analysis tasks.

2. BIG DATA GENERATION
Data generation tools and practices can be principally assigned to one of two classes: (a) reusing existing, well-known data generation tools, or (b) implementing custom, use-case tailored data generators. We first review the benefits of each one of these classes and then discuss some implications for the evaluation of big data analytics systems.

Since the establishment of standardized benchmarks as a "gold standard" for performance evaluation of database systems in the early 90's, experimental results reported in research papers often reuse data sets and queries from well-known benchmarks, like TPC-H, TPC-C [TPC13], and XMLGen [XML13]. This practice is justified by two main factors. First, the synthetic data used by standardized or public benchmarks typically adheres to a short textual specification that is well-known in the database community. Reusing data sets from such benchmarks therefore makes the data properties and their impact on the evaluated tasks more comprehensible and increases the trust in the reported experiment results. Second, well-known benchmarks typically provide open-source tools for data and workload generation, which can be adapted and used by third parties relatively easy. This reduces the overall effort required to prepare and execute

DBTEST'13, June 24, 2013, New York City, NY, USA

"proof-of-concept" experiments and allows researchers to spend more time working on the actual prototypes rather than the tooling to evaluate them.

An alternative approach that sometimes is preferred for specialized experimental studies is to define and implement a custom data generator tailored towards the requirements of the concrete experiments at hand. If the experiments are recognized as relevant by the database community, the data and tasks described in the original research are often reused by other authors in follow-up work. For example, Pavlo et al. followed this approach in their comparison of approaches for large-scale data analytics [PPR+09] and implemented a synthetic generator that generates a collection of linked HTML documents and associated data (e.g., user traffic, PageRank). The data generator and the tasks have since then been used in several other papers dealing with large-scale data analytics systems [DQJ+10, JOS+10]. For graph data, the Kronecker multiplication approach suggested by Leskovec et al. [LCK+05] offers a simple algorithm for synthetic generation of unlabeled graphs with real world characteristics (e.g., shrinking diameter, skewed degree distribution). Due to the lack of publically available real-world graphs in the terabyte range, Kronecker graphs are often featured in the evaluation sections of several graph-mining papers over the past few years [KTF09, KTA+11].

Principally, the main issue with both classes is the inherent simplicity in the statistical structure of the generated data. In the first case, this simplicity is driven by the need for concise and understandable specification for standardized benchmarks. In the second case, the main hindering factor is the complexity introduced in the data generation programs by the need for correlated data and the amount of resources that researchers are willing to invest in their development.

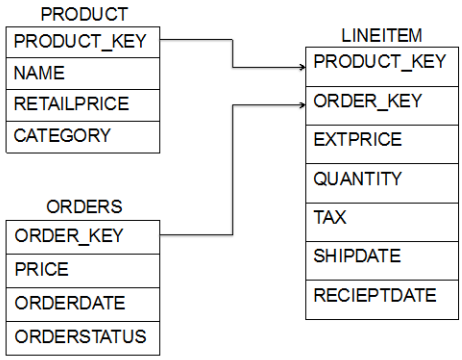

Figure 1: Simplified Retail Database Schema

In reference to the characteristics of new big data analysis systems presented in Section 1, the use of oversimplified synthetic data creates a subtle pitfall that may impact the relevance of research results for real-world applications. The reason for this is that per definition such systems must work in a distributed execution environment (cluster or cloud), and also must use some form of data-parallelism in order to ensure scale-out. These design decisions are highly sensitive to data skew, which often is present in many target application domains "a priori" and potentially changes over time. To illustrate the problem, consider the retail database schema depicted on Figure 1 and a use-case, where the benchmarks or experimental setup models an application that wants to compute the top-k most purchased items per product category. Since some product categories are naturally more in demand than others, introducing a skew over the product category distribution in the joined LINEITEM-PRODUCT view is critical to the relevance of the generated data. As most systems will process each product category group in parallel, skew will obviously influence system performance for this particular task. Moreover, for an online computation of the same counts in a streaming setting, the degree of skew will depend on the time of the current window (e.g., in the U.S. shopping peaks between Thanksgiving Day & Christmas and attains a maximum on "Black Friday"). In this case, assuming an evenly distributed load across time is an oversimplification that can influence the relevance of the experimental results for real-world applications.

With the advent of big data comes the requirement to quickly generate huge data sets. This is particularly a challenge when generating data sets with key/foreign-key relationships or other complex correlations across tables. Using specialized random number generators with seed skipping allows for doing so in parallel without having to communicate data generated on one node of a shared-nothing cluster to another [RFS+10, FPR12, ASP+11, ATM12], resulting in toolkits such as PDGF [PDG13] or Myriad [Myr13]. Both toolkits provide a set of domain-specific primitives for data generation that facilitate the transparent use of seed-skip PRNGs and complementary techniques for scalable generation of complex data.

3. GENERATING REALISTIC DATA SETS

The advances in new methods for scalable generation of realistic data highlight an important practical question: "If the data generator program can be expressed in terms of a small set of special primitives, then to which extent and in which scenarios can the specification process itself be executed automatically?" A naïve general approach is based on the analysis of empirical observations in the modeled domain and the subsequent synthesis of a data generator specification from these observations. In business scenarios, however, the analysis is often done in the context of a reference dataset that represents a ground truth for the derived data generator. This section sketches our vision for an integrated framework for such usage scenarios. We propose an extensible architecture with clean separation between the data generation primitives and the methods and techniques used to extract relevant features from the ground truth data set.

A large problem for benchmarking and testing of big data systems is the lack of realistic data sets. Many synthetic data sets follow simplistic assumptions (e.g., few correlations, mostly uniform distributions, oversimplified schema) that are not representative for real-world data. A promising, generalizable, and more effective way is to automatically extract the domain information from a ground truth data set, which is often available in practice. Figure 2 illustrates our envisioned pipeline. The domain information is first extracted from the reference database in the form of domain constraints, which can be either hard (e.g., foreign keys, unique keys, and other functional dependencies) or soft (e.g., local statistical models). The obtained structural, semantic, and statistical information is then unified into an intermediate model representing the schema information with annotated constraints. A final synthesis pass transforms the intermediate representation into a data generator specification for a specific target environment like the Myriad. This specification is then used to create a concrete data generator instance that is able to mimic the original data set.

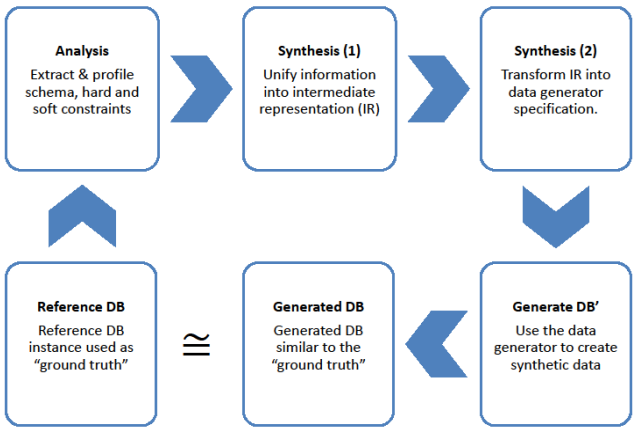

Figure 2: A Pipeline for the Analysis & Synthesis of Data Generators

We note that in the first step of this process, the circumstances in which the analysis is performed will influence its depth and consequently the quality of the collected domain information.

If the reference database cannot be accessed directly and the domain information is available only in a derived form, such as in a database catalog, the analysis must be performed *indirectly* and can only extract the available catalog information. This information commonly consists of attribute value statistics (e.g., frequency values, histograms, number of distinct values, and number of NULLs), schema information, and integrity constraints (e.g., referential integrity, primary keys, and unique constraints as well as other constraints representing domain invariants).

Alternatively, if the reference database is available *directly*, advanced profiling methods could be leveraged to obtain information beyond the catalog in order to capture a more accurate domain model. This approach will require us to determine additional characterizations of the dataset to be generated (e.g., advanced multivariate statistics [SHM+06] and soft constraints [IMH+04, BH03, SBH+06]) on the data with scalable methods (see [HIL+09] for an overview of statistical methods, and [Nau13] for an overview of data profiling). Using these techniques will allow for determining the essential characteristics of real-world data sets and correspondingly will enable one to scale up or down synthetic clones.

The integration of data profiling and data generation workflows is relevant in the era of big data for a number of reasons. First, many institutions publish their data sets in order to let others perform their experiments on them. However, database sizes are becoming larger and larger. Consequently, it is becoming increasingly difficult to transfer these huge data sets to the person wishing to use them due to network and bandwidth constraints. Therefore, it is desirable to have a compact specification of the data sets, i.e., a synopsis or profile from which one can automatically generate a data generator specification and thus the dataset. Second, data profiling will increase the relevance tests or benchmarks. Huppler [Hup09] describes five key aspects for a good benchmark, namely a good benchmark has to be relevant, repeatable, fair, verifiable, and economical. Section 2 mainly addressed the latter one, while data profiling will help to improve the relevance.

Currently, we are developing a prototype called Oligos [Oli13] that adheres to our aforementioned vision. The initial version of Oligos can generate data generator specifications for the Myriad Toolkit [Myr13] from the system catalog of a database system. Our long-term vision is provide a modular API that will allow learning advanced statistics and correlation information, in order to generate even more realistic data sets.

4. AN APPLICATION: REGRESSION TESTING OF BIG DATA SYSTEMS

An important part of the maintenance lifecycle of commercial big data systems as well as general data management systems is devoted to the diagnosis of performance regressions observed by customers in a production setting. When trying to reproduce the problematic behavior in a test environment, database system developers often face the problem of missing data – even though the database schema and the problematic queries can be provided by the customer as part of the regression report, the actual database instance typically cannot be obtained (e.g., due to privacy restrictions). Typically, what is available is the database catalog, which contains a statistical approximation of the reference database in the form of value distributions, cardinalities, and histograms on columns or column groups. As a fallback solution, developers currently trick the optimizer of a test database by feeding customer catalog data in order to obtain the query access paths of the actual production system. As the underlying data is missing and the database catalog is usually lacking crucial information (e.g., on multivariate distributions) synthetic data sets generated in the lab are not representative. Thus, information on how the query access paths perform requires further assistance and feedback from the client. The lack of a complete and representative regression database therefore slows down the maintenance process and causes additional costs. The methods for data generation based on data and workload characterization as envisioned in Oligos and Myriad would offer a remedy to this problem.

5. OPEN ISSUES AND CONCLUSIONS

We have given an overview of issues in big data benchmarking and testing, with a strong focus on data generation. We believe that efficiently generating a huge, realistic data set is an important prerequisite for the advancement, evaluation, and fair comparison of big data systems. Myriad [Myr13], PDGF [DPG13], and Oligos [Oli13] are a first step in this direction. However, in the context of big data generation and benchmarking, a large number of challenges remain open.

However, in the context of big data generation and benchmarking, a large number of challenges remain open. For realistic data generation from a given reference dataset the challenges exist both in the analysis and the synthesis phase.

During the analysis phase, a combination of data characterization and profiling methods can be identified and applied in order to increase the quality of the domain information that can be inferred directly from the reference database. Such methods will allow to efficiently determine multi-key dependencies, in particular referential integrity, as well as to profile data with complex structure (e.g., text, graphs, NF^2 and hierarchical data). In order to preserve privacy when conducting data profiling, data obfuscation methods may as well be required. [Nau13] lists further challenges in the area of data profiling.

Inferred schema information and constraints must be then unified into an intermediate representation (IR) in the synthesis phase. Two problems exist in this context. First, in order to facilitate the subsequent translation of the IR into a data generator specification, the IR should lend itself to the features and primitives common to the underlying data-generation engines. Second, the unification process should determine and handle

inconsistencies in the domain information collected in the analysis phase. Recently, Arasu [AKL11] and Torlak [Tor12] suggested two different constraint-based languages for data generator specification that can serve as a starting point for the development of a suitable IR and synthesis algorithm. For both languages, the authors give sufficiency conditions for the existence of a data set fulfilling the input constraints and provide approximate algorithms to find such an instance. The approach presented in [Tor12] uses a mix of hard (dimension or integrity) and soft (statistical) constraints and is restricted to dimension models, whereas [AKL11] works on general relational models and relies solely on soft (cardinality) constraints (hard constraints are represented implicitly as a special form of soft constraints). As the target language in our setting is likely to include primitives that directly enforce certain types of hard constraints (e.g. unique keys, foreign keys), we believe that a distinction between soft and hard constraints in the IR is a more promising approach.

Another big open area is the provisioning of workloads. Traditional benchmarks focus on simple workloads that essentially follow the relational algebra or an NF² algebra/ XQuery. For evaluating and testing big data analytics systems, we will require more complex workloads that involve machine learning algorithms, information extraction, and graph analysis/mining. The lack of a standardized data analysis language currently is a big obstacle for arriving at realistic, comparable, and universally useful workload specifications. Ideally, until a standardized declarative language is available use-case repositories may be a first step in this direction.

6. ACKNOWLEDGMENTS

We thank Berni Schiefer from IBM and Tillmann Rabl from the University of Toronto for interesting discussions. Our investigations were funded by a CAS grant from IBM, the ICT Labs of the European Institute of Technology as well as the DFG (German National Science Foundation) via the Stratosphere Collaborative Research Unit.

7. REFERENCES

[ABE+10] A. Alexandrov, D. Battré, S. Ewen, M. Heimel, F. Hueske, O. Kao, V. Markl, E. Nijkamp, D. Warneke: Massively Parallel Data Analysis with PACTs on Nephele. PVLDB Vol. 3, No. 2, pp. 1625–1628 (2010)

[ABG+12] S. Alsubaiee, A. Behm, R. Grover, R. Vernica, V. Borkar, M. J. Carey, C. Li: ASTERIX: Scalable Warehouse-Style Web Data Integration. In Proceedings of the Ninth International Workshop on Information Integration on the Web, Article 2, ACM, (2012)

[AKL11] A. Arasu, R. Kaushik, J. Li: Data Generation using Declarative Constraints. Proceeding of the SIGMOD Conference, pp. 685-696 (2011)

[ASP+11] A. Alexandrov, B. Schiefer, J. Poelman, S. Ewen, T. Bodner, V. Markl: Myriad - Parallel Data Generation on Shared-Nothing Architectures, In Proc. ASBD, pp. 30-33 (2011)

[ATM12] A. Alexandrov, K. Tzoumas, V. Markl: Myriad: Scalable and Expressive Data Generation, In Proc. VLDB(5) pp. 1890-1893 (2012)

[BH03] P. Brown, P. Haas: BHUNT: Automatic Discovery of Fuzzy Algebraic Constraints in Relational Data. VLDB 2003: 668-679

[BEG+11] K. S. Beyer, V. Ercegovac, R. Gemulla, A. Balmin, M. Eltabakh, C.-C. Kanne, E. J. Shekita: Jaql: A scripting language for large scale semistructured data analysis. In Proc. of VLDB Conference. (2011)

[DG04] J. Dean, S. Ghemawat: MapReduce: simplified data processing on large clusters, In OSDI, pp. 137-150 (2004)

[FML+12] F. Färber, N. May, W. Lehner, P. Große, I. Müller, H. Rauhe, J. Dees: The SAP HANA Database -- An Architecture Overview. IEEE Data Eng. Bull. 35(1): 28-33 (2012)

[FPR12] M. Frank, M. Poess, T. Rabl: Efficient update data generation for DBMS benchmarks. ICPE 2012: 169-180

[GSA+11] R. Greenwald, R. Stackowiak, M. Alam, M. Bhuller.. Achieving extreme performance with Oracle Exadata. McGraw-Hill Osborne Media (2011)

[Had13] http://hadoop.apache.org/, last accessed 05-10-2013

[HIL+09] P. J. Haas, I. Ilyas, G. Lohman, V. Markl: Discovering and Exploiting Statistical Properties for Query Optimization in Relational Databases: A Survey. Statistical Analysis and Data Mining 1(4): 223-250 (2009)

[Hup93] K. Huppler: The Art of Building a Good Benchmark. TPCTC 2009: 18-30 (2009)

[IMH+04] I. Ilyas, V. Markl, P. Haas, P. Brown, A. Aboulnaga: CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies. SIGMOD Conference 2004: 647-658

[Imp13] https://github.com/cloudera/impala, last accessed 05-10-2013

[LBG+12] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, J. M. Hellerstein: DistributedGraphLab: A framework for machine learning and data mining in the cloud. Proceedings of the VLDB Endowment, 5(8), pp. 716-727 (2012)

[Mah13] Mahout: http://mahout.apache.org/, last accessed 04-21-2013

[Myr13] https://github.com/TU-Berlin-DIMA/myriad-toolkit/wiki, last accessed 05-10-2013

[Nau13] http://www.hpi.uni-potsdam.de/naumann/publications/publications_by_type/year/2013/2276/Nau13.html, SIGMOD Record (2013)

[Oli13] https://github.com/TU-Berlin-DIMA/myriad-toolkit/wiki/Using-Oligos-Guide, last accessed 05-10-2013

[ORS+08] C. Olston, B. Reed, U. Srivastava, R. Kumar, A. Tomkins: Pig Latin: A Not-So-Foreign Language for Data Processing. Proceedings of the SIGMOD Conference (SIGMOD), pp. 1099-1110, (2008)

[PDG13] http://www.paralleldatageneration.org/drupal6/, last accessed 05-10-2013

[RFS+10] T. Rabl, M. Frank, H. Sergieh, H. Kosch: A Data Generator for Cloud-Scale Benchmarking. TPCTC 2010: 41-56

[SBH+06] Y. Sismanis, P. Brown, P. Haas, B. Reinwald: GORDIAN: Efficient and Scalable Discovery of Composite Keys. VLDB 2006: 691-702

[SHM+06] U. Srivastava, P. Haas, V. Markl, M. Kutsch, T.Tran: ISOMER: Consistent Histogram Construction Using Query Feedback. ICDE (2006)

[Spa13] http://spark-project.org/, last accessed 05-10-2013

[Str13] http://www.stratosphere.eu/, last accessed 05-10-2013

[Tor12] E. Torlak: Scalable test data generation from multidimensional models. Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (2012)

[TPC13] http://www.tpc.org, last accessed 05-10-2013

[TSJ+09] A. Thusoo, J. S.Sarma, N. Jain, Z. Shao, P.Chakka, S. Anthony, H. Liu, P. Wyckoff, R. Murthy: Hive - A Warehousing Solution Over a Map-Reduce Framework. PVLDB 2(2), pp. 1626-1629 (2009)

[XML13] http://www.xml-benchmark.org/, last accessed 05-10-2013

[PPR+09] A. Pavlo, E. Paulson, A. Rasin, D. Abadi, D. DeWitt, S. Madden, M. Stonebraker: A comparison of approaches to large-scale data analysis. SIGMOD Conference 2009: 165-178

[DQJ+10] J. Dittrich, J. Quiané-Ruiz, A. Jindal, Y. Kargin, V. Setty, J. Schad: Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing). PVLDB 3(1): 518-529 (2010)

[JOS+10] D. Jiang, B. C. Ooi, L. Shi, S. Wu: The Performance of MapReduce: An In-depth Study. PVLDB 3(1):472-483 (2010)

[LCK+05] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos: Realistic, Mathematically Tractable Graph Generation and Evolution, Using Kronecker Multiplication. PKDD 2005: 133-145

[KTF09] U. Kang, C.E. Tsourakakis, C. Faloutsos: PEGASUS: A Peta-Scale Graph Mining System. ICDM 2009: 229-238

[KTA+11] U. Kang, C.E. Tsourakakis, A.P. Appel, C. Faloutsos, J. Leskovec: HADI: Mining Radii of Large Graphs. TKDD 5(2): 8 (2011)