# Big Data Generation

Tilmann Rabl
Middleware System Research Group
University of Toronto
rabl@fim.uni-passau.de

Big data challenges are end-to-end problems. When handling big data it usually has to be preprocessed, moved, loaded, processed, and stored many times. Current benchmarks related to big data, however, only focus on isolated aspects.

In this abstract, we briefly discuss the necessity of ETL like tasks in big data benchmarking and make a case for the parallel data generation framework (PDGF) for big data generation. PDGF is an open-source generic data generator that was implemented at the University of Passau and is currently adopted in TPC benchmarks.

## 1 Introduction

Many big data challenges begin with an ETL like process. Raw data is gathered, for example, from a web site or click streams (e.g. Netflix, Facebook, Google) or sensors (energy monitoring, application monitoring, telescope). The first challenge is to store the data at the high production rate. In the next step, the data is filtered and normalized and finally it is loaded in a system that will then do the processing. This preprocessing is often time-consuming and hinders an on-line processing of the data. Nevertheless, current big data benchmarks, e.g. GraySort [1], YCSB [2], HiBench [4], mostly concentrate on a single performance aspect rather than giving a holistic view and neglect the challenges in the initial ETL processes and data movement. A comprehensive big data benchmark should have an end-to-end semantic.

For a benchmark to be successful it has to be easy to use. Benchmarks that come with a complete tool chain are used more frequently than benchmarks that consist only of a specification. A recent example is the YCSB which is widely used. For a big data benchmark the most important tool is the data generator. In order to support the various steps of big data processing it would be beneficial to have a data generator that can generate the data in different phases consistently. This makes a verification of intermediate results as well as isolate single steps of the benchmark procedure possible and thus further increases the benchmarks applicability. This requires the data properties that are processed (such as dependencies and distributions) to be strictly computable. A data generation tool that follows this approach is the Parallel Data Generation Framework.

## 2 Parallel Data Generation Framework

The Parallel Data Generation Framework (PDGF) is a flexible, generic data generator that can be used to generate large amounts of relational data very fast. It was developed at the University of Passau and is currently used in the development of a standard ETL benchmark (described in [9]). PDGF exploits parallel random number generation for an independent generation of related values. The underlying approach is straight forward; the random number generator is a hash function which can generate any random number in a sequence in O(1) without having to compute other values. With such random number generators every random number can be computed independently. Based on the random number arbitrary values can generated using mapping functions, dictionary lookups and such. Quickly finding the right random number is possible by using a hierarchical seeding strategy (table $\rightarrow$ column $\rightarrow$ row). Furthermore, by using bijective permutations it is possible to generate consistent updates of the data where values may be inserted, changed and deleted. For details of this generation approach see [3, 5, 6, 7, 8].

## 3 A Big Data Generator

Based on PDGF, we can build a versatile data generator for big data benchmarking. Although PDGF was initially intended for relational data it features a post-processing module that enables a mapping to other data formats such as XML, RDF, etc. Since all data is deterministically generated and the generation is always repeatably it is possible to compute intermediate and final results of transformations. The underlying relational model also makes it possible to generate consistent queries on the data. This makes PDGF an ideal candidate tool for big data benchmarking.

What is missing for an easy to use benchmark is a driver that starts the execution, measures the performance and calculates the metrics. This is non trivial since there is no standard access language so far. However, relational input as generated by PDGF can be easily transformed in any other representation which will ease the implementation of such tool chains.

## 4 Biography

Tilmann Rabl is a postdoctoral researcher at the Middleware System Research Group (msrg.org) led by Prof. Dr. Hans-Arno Jacobsen at the University of Toronto. He finished his doctoral thesis on allocation and scaling in relational database systems in 2011 at the University of Passau. An significant part of his thesis was the development of the Parallel Data Generation Framework (PDGF). For this work he received a technical contribution award from the Transaction Performance Processing Council in 2011. His current work focuses on big data challenges in application performance management, in traffic monitoring and in power monitoring. He is involved in several benchmarking development efforts in the area of big data, ETL and data warehouses.

## References

[1] Graysort. http://sortbenchmark.org/.

[2] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with ycsb. In *SoCC*, pages 143–154, 2010.

[3] M. Frank, M. Poess, and T. Rabl. Efficient Update Data Generation for DBMS Benchmark. In *ICPE '12*, 2012.

[4] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang. The hibench benchmark suite: Characterization of the mapreduce-based data analysis. In *ICDEW*, 2010.

[5] M. Poess, T. Rabl, M. Frank, and M. Danisch. A PDGF Implementation for TPC-H. In *TPCTC '11*, 2011.

[6] T. Rabl, M. Frank, H. M. Sergieh, and H. Kosch. A Data Generator for Cloud-Scale Benchmarking. In *TPCTC '10*, pages 41–56, 2010.

[7] T. Rabl, A. Lang, T. Hackl, B. Sick, and H. Kosch. Generating Shifting Workloads to Benchmark Adaptability in Relational Database Systems. In *TPCTC '09*, pages 116–131, 2009.

[8] T. Rabl and M. Poess. Parallel data generation for performance analysis of large, complex RDBMS. In *DBTest '11*, page 5, 2011.

[9] L. Wyatt, B. Caufield, and D. Pol. Principles for an ETL Benchmark. In *TPC TC '09*, pages 183–198, 2009.