INTELIGENTNO PROCESIRANJE U SQL SERVERU 2019

MILOŠ RADIVOJEVIĆ

DATA PLATFORM MVP

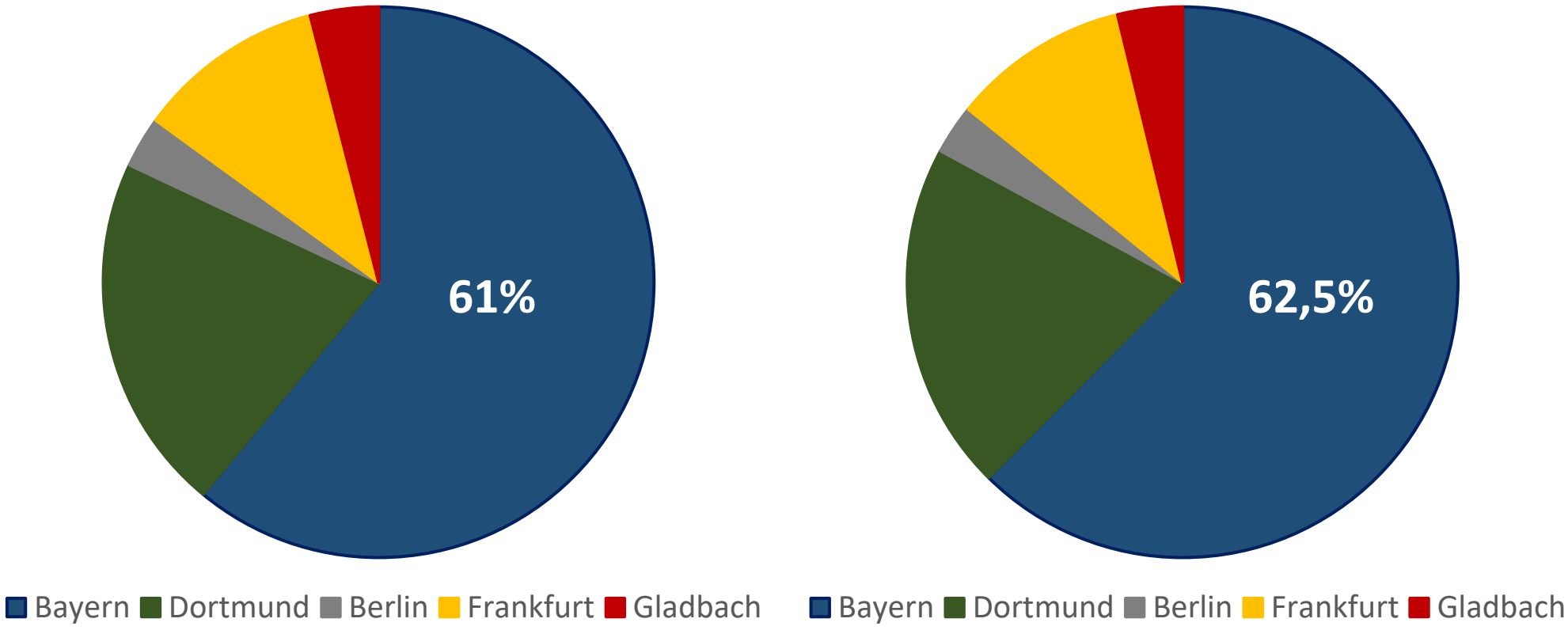
26. DECEMBAR 2020

APPROXIMATE QUERY PROCESSING

APPROXIMATE QUERY PROCESSING

- We don't always need precise answers
  - Aggregated queries
  - Quick overview to understand data
  - Check for rough values
  - Dashboards and visualization tools
- Large analytical systems require speed and interactivity

APPROXIMATE QUERY PROCESSING



APPROXIMATE QUERY PROCESSING

- A lot of research in this area, not too much in production
- Implemented or supported: Apache Spark, PostgreSQL, Oracle, Teradata, Vertica Analytics, Amazon Redshift DWH, Google BigQuery, BlinkDB (Facebook), Conviva, Verdict
- Microsoft Research

Approximate Query Processing: No Silver Bullet

Surajit Chaudhuri, Bolin Ding, Srikanth Kandula

May 2017

ACM - Association for Computing Machinery

https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/sigmod17_aqp_sb.pdf

Approximate Query Processing: No Silver Bullet

Surajit Chaudhuri, Bolin Ding, Srikanth Kandula
Microsoft Reseach
{surajitc,bolind,srikanth}@microsoft.com

ABSTRACT

In this paper, we reflect on the state of the art of Approximate Query Processing. Although much technical progress has been made in this area of research, we are yet to see its impact on products and services. We discuss two promising avenues to pursue towards integrating Approximate Query Processing into data platforms.

1. INTRODUCTION

While Big Data opens the possibility of gaining unprecedented insights, it comes at the price of increased need for computational resources (or risk of higher latency) for answering queries over voluminous data. The ability to provide approximate answers to queries at a fraction of the cost of executing the query in the traditional way, has the disruptive potential of allowing us to explore large datasets efficiently. Specifically, such techniques could prove effective in helping data scientists identify the subset of data that

approximation is interesting for applications, approximation at the query processing layer has proven ineffective for applications, because either the semantics of error models and the accuracy guarantees of AQP or the extent of savings in work accrued by AQP have been unsatisfactory.

We firmly believe that the value proposition of AQP, outlined in the opening of this article, is considerable in the world of big data. We should not give up the pursuit of such systems. However, critical rethinking of our approach to AQP research is warranted with the exclusive goal of making such systems practical [30]. The first step in such rethinking is to be clear about what combinations of the four dimensions of AQP will make it possible for applications to find AQP systems attractive. In this article, we suggest two research directions to pursue based on our reflection.

One promising approach may be to cede control over accuracy to the user. This approach is based on accepting the reality that AQP systems will not be able to offer a priori (*i.e.*, before the query

APPROXIMATE QUERY PROCESSING IN SQL SERVER 2019

- Provides aggregations over very large amounts of data where responsiveness is more important than absolute accuracy
- Currently only one function - APPROX_COUNT_DISTINCT
  - It uses significantly less memory resources
  - Error rate compared to the COUNT DISTINCT counterpart
  - within 2% for 97% Workloads
- Efficient for datasets that consist of millions of rows or more and
- Aggregation of columns with many different values

APPROX_COUNT_DISTINCT

- Sie verwendet den HyperLogLog-Algorithmus
- Philippe Flajolet et al.
- http://algo.inria.fr/flajolet/Publications/FlFuGaMe07.pdf

2007 Conference on Analysis of Algorithms, AofA 07 DMTCS proc. **AH**, 2007, 127–146

HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm

Philippe Flajolet[1] and Éric Fusy[1] and Olivier Gandouet[2] and Frédéric Meunier[1]

[1] *Algorithms Project, INRIA–Rocquencourt, F78153 Le Chesnay (France)*
[2] *LIRMM, 161 rue Ada, 34392 Montpellier (France)*

This extended abstract describes and analyses a near-optimal probabilistic algorithm, HYPERLOGLOG, dedicated to estimating the number of *distinct* elements (the *cardinality*) of very large data ensembles. Using an auxiliary memory of m units (typically, "short bytes"), HYPERLOGLOG performs a single pass over the data and produces an estimate of the cardinality such that the relative accuracy (the *standard error*) is typically about $1.04/\sqrt{m}$. This improves on the best previously known cardinality estimator, LOGLOG, whose accuracy can be matched by consuming only 64% of the original memory. For instance, the new algorithm makes it possible to estimate cardinalities well beyond 10^9 with a typical accuracy of 2% while using a memory of only 1.5 kilobytes. The algorithm parallelizes optimally and

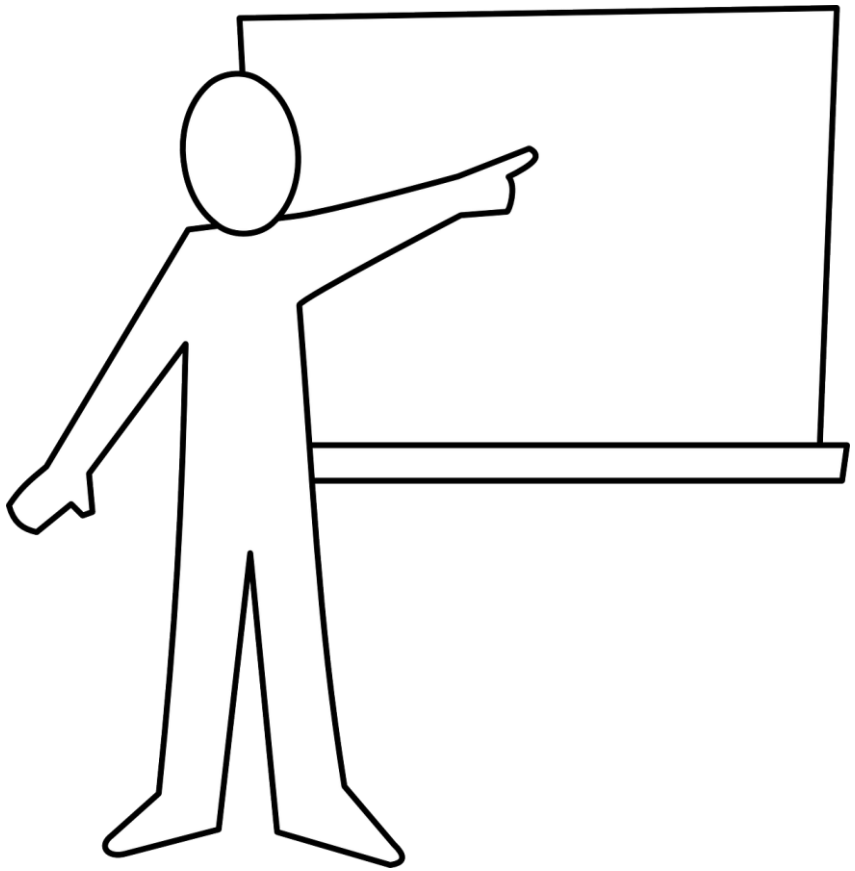
HYPERLOGLOG ALGORITHM

- It hashes each element to make the data distribution more uniform
- After hashing all the elements, it looks for the binary representation of each hashed element
- HLL looks number of leading zero bits in the hash value of each element and finds maximum number of leading zero bits

$$Number\ of\ distinct\ elements = 2^{k+1}$$

- where k is maximum number of leading zeros in data set

APPROXIMATE QUERY PROCESSING



DEMO

APPROXIMATE QUERY PROCESSING

- **Sometimes** a better execution plan (Stream Aggregate instead of Hash Match Aggregate)
- **Significantly** less memory
- **Sometimes** faster
  - Speed isn't the main reason for using it, it might even be slower
- Useful for
  - > millions of rows
  - Aggregation of a column or columns with many different values