The Case for Small Data Management

Jens Dittrich

Saarland University
infosys.cs.uni-saarland.de

ABSTRACT

Exabytes of data; several hundred thousand TPC-C transactions per second on a single computing core; scale-up to hundreds of cores and a dozen Terabytes of main memory; scale-out to thousands of nodes with close to Petabyte-sized main memories; and massively parallel query processing are a reality in data management. But, hold on a second: for how many users exactly? How many users do you know that really have to handle these kinds of massive datasets and extreme query workloads? On the other hand: how many users do you know that are fighting to handle relatively small datasets, say in the range of a few thousand to a few million rows per table? How come some of the most popular open source DBMS have hopelessly outdated optimizers producing inefficient query plans? How come people don't care and love it anyway? Could it be that most of the world's data management problems are actually quite small? How can we increase the impact of database research in areas when datasets are small? What are the typical problems? What does this mean for database research? We discuss research challenges, directions, and a concrete technical solution coined PDbF: Portable Database Files.

1. INTRODUCTION

Database researchers consider small data management solved and proceed building database systems for even bigger data. However, we are under attack by other communities which are building their own systems: With the NoSQL [3] movement, including highly scalable key-(value/document) stores, we have witnessed a wave of distributed systems focussing on handling big data. Interestingly, NoSQL attacks databases in two dimensions: the first is scalability ("We can scale to thousands of nodes with ease!"). One of our students phrased this like: "I would never use a database system! NoSQL systems are so much faster." A sentence he did not dare to repeat anymore after a 90 min database class setting things straight for him. The second argument against database systems is interfacing ("SQL is awkward!"). So, even for "small data" where databases perform very very well, people would be unwilling to pick the database system. In a recent conversation, a professor in business informatics (specialized in data management) explained to me how they used a NoSQL-document store to query a few thousand RDF tuples. Another example is MapReduce, which is often used for tiny, Megabyte-sized datasets [1].

This paper is asking the question: what happens if the data is small? Say up to a few million rows only? What happens if you do **not** need to execute several ten thousand queries every second? How can we increase the impact of database technology in that space? Is this a problem of teaching developers? A problem of MOOCifying or flipping [2] database knowledge? Is all of this just another "64K is enough" for everyone argument [4]? Or should we simply continue building cars with 100,000 horse powers just to make the two mile trip to the supermarket?

2. PORTABLE DATABASE FILES

We should work on an open file format allowing non-DB people to easily share and (re-)process data. No, I neither mean XML nor anything alike (some of us tried hard, but we also learned that it did not get wide adoption and was replaced by JSON). I mean something that is as easy to use and portable as pdf, yet has the capabilities of a database. We need some sort of pdf with query processing capabilities. The widely used SQlite file format is already a nice step towards portability (being superior than proprietary MS-Access file formats in that respect). However, what we believe and foster in this paper goes beyond that: we believe we need **a full fusion of pdf and databases**. We coin this **portable database files** (pdbf). A pdbf is downward compatible to pdf. A pdbf can be viewed by a special browser (which is just an extension or wrapper of an existing pdf viewer). That pdbf-browser may overlay certain regions of the underlying pdf with data input and output facilities which are processed by a query engine (just as pdf forms, yet no external DB connection necessary). The pdbf-browser visualizes the contents of the database using: (1) a database file, and (2) a pdbf configuration file, specifying geometry and behavior of the dynamic overlays. **Both files are kept as attachments to the pdf.** With this any part of the pdfb may become dynamic and database-driven.

3. USE-CASES

Open this pdf with Adobe Reader (Adobe Professional or XI is **not** required) and inspect the attachment 'usecases.sqlite' SQLite or SQLite Database Browser. See table 'UseCases'.

4. REFERENCES

[1] Y. Chen, S. Alspaugh, and R. H. Katz. Interactive Analytical Processing in Big Data Systems: A Cross-Industry Study of MapReduce Workloads. *PVLDB*, 5(12), 2012.
[2] `http://datenbankenlernen.de`.
[3] J. Dittrich. Say No! No! and No! In *CIDR*, 2013.
[4] M. Fowler. Introduction to NoSQL. In *GOTO*, 2013.