XDB - A novel Database Architecture for Data Analytics as a Service

Carsten Binnig, Abdallah Salama, Alexander C. Müller (University of Mannheim)
Erfan Zamanian, Harald Kornmayer (DHBW Mannheim)
Sven Lising (GSRN Mannheim)

Abstract

Parallel database systems are major platforms for supporting analytical queries over large data sets. However, in order to offer SQL-like services for data analytics in the cloud, providers such as Amazon and Google do often build their own systems (e.g., BigTable). One reason is that existing database systems do not fulfill important requirements such as elasticity and fine-grained fault-tolerance. In this poster, we present *XDB* [2, 3], a parallel database system which implements two novel concepts: (1) a partitioning scheme that supports elasticity with regard to data and queries, and (2) a fine-grained fault-tolerance scheme for short- and long-running queries.

1 Elastic Partitioning Scheme

The elastic partitioning scheme is designed to satisfy the following requirements: (1) data used for joins over foreign-key (fk) relationships should be co-located on the same node, and (2) tables should be partitioned into small partitions whose size is bound by a user-given threshold such that the table data can be easily re-distributed without re-partitioning.

In order to meet these requirements, we first present a novel partitioning method called reverse-reference (RREF) partitioning. RREF-partitioning is similar to reference (REF) partitioning [5], which equi-partitions a table with respect to another table based on a referential constraint. RREF-partitioning works in the opposite direction of a referential constraint, which means that it might introduce duplicates in different partitions.

SoCC'13, 1–3 Oct. 2013, Santa Clara, California, USA.
ACM 978-1-4503-2428-1.
http://dx.doi.org/10.1145/2523616.2525947

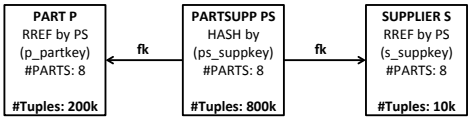

Figure 1: Partitioned TPC-H Schema (excerpt)

The procedure for partitioning all tables of a given schema is as follows: we first partition the largest table such that the user-given threshold is not exceeded. This can be achieved by a variant of HASH-partitioning, which additionally splits partitions if the threshold is exceeded. Starting from the largest table, all other tables are recursively partitioned using the REF- or the RREF-partitioning method. Figure 1 shows the result of partitioning three tables of the TPC-H schema [1] using a threshold of 100*k* tuples: the largest table `PS` is first split into 8 parts, while the other tables are RREF-partitioned.

2 Fine-grained Fault-Tolerance

Typically, databases handle node failures by restarting the complete query on a replica of the data. This scheme is good for short running queries and clusters, where node failures are rare. However, when running on clusters of commodity machines or on IaaS offerings (such as Amazon's Spot Instances as an extreme case) node failures are much more likely. In this case, a fine-grained fault-tolerance scheme which supports recovery from mid-query faults is essential to save computation costs and to deliver a decent performance.

The goal of the fine-grained fault-tolerance scheme in *XDB* is to recover from mid-query faults (i.e., to use materialized intermediate results to re-start a query). Compared to other existing systems like Hadoop [4, 6] and SCOPE [7], which implement a fine-grained fault-tolerance scheme by materializing each intermediate result, *XDB* materializes only some selected intermediate results by using a cost model, which takes the overheads for materialization into account. The main goal of our cost model is to find materializations, such that the success-rate to finish a query is maximized while the overall run-time of that query is minimized. That way *XDB* is able to decide that short-running queries often do not benefit from materializing intermediate results.

References

[1] TPC-H. http://www.tpc.org/tpch/.

[2] XDB. http://code.google.com/p/xdb/.

[3] C. Binnig, A. Salama, E. Zamanian, A. C. Müller, S. Listing, and H. Kornmayer. XDB - A novel Database Architecture for Data Analytics as a Service. Technical report, University of Mannheim, 2013. http://pi1.informatik.uni-mannheim.de/filepool/publications/XDB/xdb_techreport.pdf.

[4] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

[5] G. Eadon, E. I. Chong, S. Shankar, A. Raghavan, J. Srinivasan, and S. Das. Supporting table partitioning by reference in Oracle. In *SIGMOD Conference*, pages 1111–1122, 2008.

[6] T. White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 1st edition, 2009.

[7] J. Zhou, N. Bruno, M.-C. Wu, P.-Å. Larson, R. Chaiken, and D. Shakib. SCOPE: parallel databases meet MapReduce. *VLDB J.*, 21(5), 2012.