Poster Paper

Extending Database Accelerators for Data Transformations and Predictive Analytics

Knut Stolze
IBM Germany Research & Development GmbH
Schönaicher Straße 220
71032 Böblingen, Germany
stolze@de.ibm.com

Felix Beier
IBM Germany Research & Development GmbH
Schönaicher Straße 220
71032 Böblingen, Germany
febe@de.ibm.com

Daniel Martin
IBM Germany Research & Development GmbH
Schönaicher Straße 220
71032 Böblingen, Germany
danmartin@de.ibm.com

ABSTRACT

The IBM DB2 Analytics Accelerator (IDAA) integrates the strong OLTP capabilities of DB2 for z/OS with very fast processing of OLAP workloads using Netezza technology. The accelerator is attached to DB2 as analytical processing resource – completely transparent for user applications. But all data modifications must be carried out by DB2 and are replicated to the accelerator internally. However, this behavior is not optimized for ELT processing and predictive analytics or data mining workloads where multi-staged data transformations are involved. We present our work for extending IDAA with accelerator-only tables, which enable direct data transformations without any necessary interventions by DB2. Further, we present a framework for executing arbitrary in-database analytics operations on the accelerator while ensuring data governance aspects like privilege management on DB2 and allowing to ingest data from any other source directly to the accelerator to enrich analytics e. g., with social media data. The evolutionary framework design maintains compatibility with existing infrastructure and applications, a must-have for the majority of customers, while allowing complex analytics beyond read-only reporting.

Keywords

analytics, data mining, db2, mainframe, idaa

1. INTRODUCTION

The IBM DB2 Analytics Accelerator (IDAA) [1] is an extension for IBM's® DB2® for z/OS® database system. It's primary objective is the extremely fast execution of complex, analytical queries on a snapshot of the data copied from DB2. However, when it comes to more complex, multi-staged data analysis tasks like data mining, the accelerator can often provide limited improvements only. Predictive analytics tools like SPSS [4] resort to multiple SQL statements, each implementing a step or stage in a chain of data preparation, transformation, and evaluation tasks. For each stage, base data needs to be transferred to IDAA before mining algorithms can be run and result data has to be materialized within DB2 before it can be used as input for the next stage or iteration. A key requirement for enhancing these workloads is to minimize data movement while still exploiting the accelerator for this task. We solve this issue with accelerator-only tables (AOTs) which are discussed in Sec. 2. The second use case is the application of the analytic algorithms in the pipeline. A generic framework is required which allows to pass code for arbitrary algorithms to IDAA while still implementing data governance aspects correctly. A seamless approach, completely transparent to user applications is discussed in Sec. 3.

2. ACCELERATOR-ONLY TABLES AND DATA INGESTION

Maintaining a copy of the DB2 data in the accelerator for query processing is the main use case of IDAA. However, this design involves a lot of data movements in case of multi-staged algorithms that require result materialization inside DB2 before the next step can be executed. The first building block in our current efforts are accelerator-only tables (AOTs), i. e., tables whose data solely resides inside IDAA (cf. Fig. 1), and DB2 only keeps a *proxy* or *table reference* which is usually named *nickname* in federation contexts [5]. This proxy is used for storing meta data in the DB2 catalog and acts as indicator for delegating any query on the corresponding AOT to IDAA. For creating AOTs the `CREATE TABLE` statement was extended by an additional `IN ACCELERATOR` clause. AOTs are populated with `INSERT` statements comprising a list of values or a sub-select which might invoke arbitrary transformation procedures (cf. Sec. 3), retrieving the data from other regular accelerated tables or AOTs. Likewise, `UPDATE`s and `DELETE`s are handled. The second way for populating AOTs is the new IDAA Loader [2] (cf. Fig. 1). The data to be loaded can originate from a variety of sources, even from applications not running on System z which opens up a wide range of new use cases. Data can be ingested in both, regular DB2 tables and AOTs. In the past, IDAA was not concerned about transactions because only the cursor stability isolation level was supported. Queries were executed under snapshot isolation in Netezza. With AOTs, IDAA has to be aware of the DB2 transaction context so that correct results are guaranteed, i. e., uncommitted data modifications of the own transaction are handled. At the same time, concurrent execution of multiple queries in a single transaction are also supported.

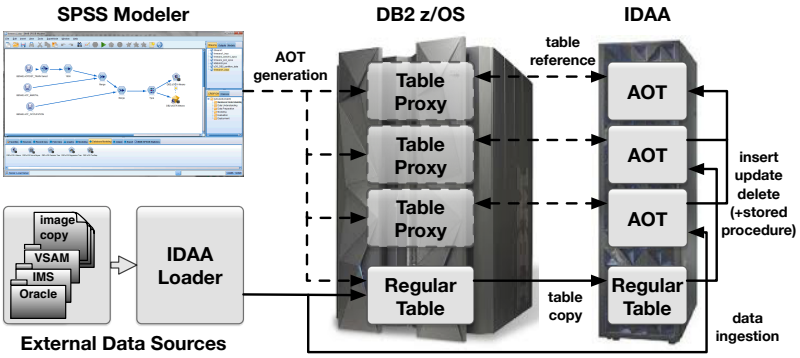

Figure 1: Overview IDAA with Accelerator-Only Tables

3. IN-ACCELERATOR ANALYTICS

IDAA has now a framework for invoking generic, customer-specific stored procedures (SPs). A SP can be called in DB2z, and IDAA forwards the procedure execution to the Netezza backend (cf. Fig. 2 and Fig. 3). We integrated the Netezza analytics package [3] which is used by SPSS [4]. The SPSS modeler provides means to define nodes, to create and populate AOTs, e. g., by joining accelerated tables, and then to invoke an analytics SP running in IDAA. Such a scenario is illustrated in Fig. 1 where the k-means clustering algorithm is applied on some filtered, enriched, and sampled input data. Intermediate results are materialized into AOTs and finally fed to the k-means SP. However, two challenges occur here. First, user privileges need to be verified for all data sources referenced by such a black box SP. Therefore, IDAA resolves all dependencies without executing the SP code. These table references are passed to DB2 which in turn validates necessary user privileges before the actual operation is carried out. The second issue arises from views existing on the DB2 side, which are not present on the accelerator. The framework exploits DB2's query acceleration capabilities to extract the view definition and implicitly translate it to the Netezza SQL dialect. A temporary view is created in the Netezza backend using that definition.

4. RELATED WORK

Using accelerator-only tables is similar to the concept of pass-through functionality in federated systems based on SQL/MED [5]. However, the overall use case for AOTs is quite different. AOTs are conceptually DB2 tables and can be manipulated using DB2's SQL dialect. It is just that AOTs store all their data in the analytics-optimized query engine and not in the transactional storage engine of DB2 for z/OS itself. Contrary to that, federated systems provide a means to access tables and data residing in another, stand-alone database system. The integration of the different query engines is on a much deeper level in IDAA.

MySQL provides an internal interface to plugin different storage engines with different characteristics [6]. IDAA in DB2 for z/OS implements a similar architecture by combining DB2's and Netezza's characteristics and the respective underlying storage mechanisms. However, the integration of both happens on the SQL dialect and SQL optimizer layer and not on the buffer pool and storage layers.

5. TRADEMARKS

6. REFERENCES

[1] P. Bruni et al. *Reliability and Performance with IBM DB2 Analytics Accelerator V4.1*. IBM Redbooks, 2014.

[2] IBM. *DB2 Analytics Accelerator Loader for z/OS*, 2014. http://www-03.ibm.com/software/products/en/db2-analytics-accelerator-loader-for-zos.

[3] IBM. *IBM Netezza Analytics – In-Database Analytics Developer's Guide, Release 3.0.1*, 2014.

[4] IBM. *SPSS Software*, 2014. http://www.ibm.com/software/analytics/spss/.

[5] ISO/ IEC 9075-9:2003. *Information Technology – Database Languages – SQL – Part 9: Management of External Data (SQL/ MED)*, 2nd edition, 2003.

[6] A. Lentz. MySQL Storage Engine Architecture. *MySQL Developer Articles, MySQL AB, May*, 2004.

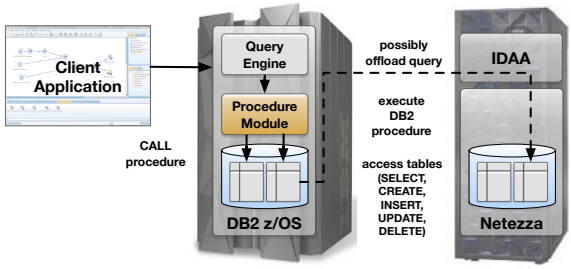

Figure 2: Stored Procedure Execution in DB2

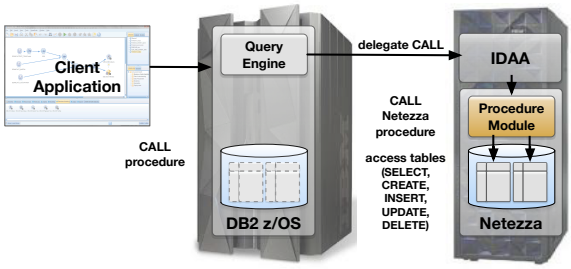

Figure 3: Stored Procedure Execution in Netezza