Paper Reviews - Machine Learning on Relational Database

Parth Doshi 1215200012 pdoshi4@asu.edu

Towards a Unified Architecture for in-RDBMS Analytics

Due to the ever increasing usage of data analysis by enterprise applications using statistical approaches, there is an ongoing battle for supremacy in offering sophisticated in-database analytics amongst the database vendors. However the key challenge in this evolving domain remains - the implementation of statistical techniques from scratch proves to be a complicated time-consuming development process. The paper advocates for a unique architecture for in-RDBMS analytics to tackle this problem. They propose Bismarck, a unique architecture implementing various statistical analytics in existing RDBMS systems.

The paper presents empirical studies of performance impacting factors - order of data storage and computation parallelism for a unimodal multicore RDBMS. It presents Incremental Gradient Descent, an algorithm famous for convex programming problems with data access patterns similar to SQL aggregation functions. The authors use this key observation to build Bismark, showcasing implementation of IGD for different tasks using user-defined aggregation available with RDBMS. They demonstrate the feasibility of their architecture through integration with commercial and open-source RDBMS while presenting the experiments to validate their higher performance claims.

Strengths

1. The unified architecture helps to analyze the performance optimizations for these analytics techniques in a generic way compared to an ad-hoc per-technique approach.
2. To integrate a new analytical task, the unified architecture would only require changes to a small section of the code base, a relatively simple process compared to the complicated development process involving implementing from scratch.
3. This approach is 2–4x faster than current in-database analytic tools for simple techniques while for newer techniques like matrix factorization, it is relatively faster.

Weakness

1. The paper doesn't mention the ease of programming interface for end-developers who have been using different statistical techniques for different RDBMS.
2. The paper provides a lack of evidence for the adoption of this kind of architecture. The traditional RDBMS users might be reluctant to transition.
3. The paper provides a lack of information on the flexibility of these statistical tasks in tackling evolving requirements in diverse datasets.

SystemML: Declarative Machine Learning on Spark

The ever-evolving, ever-increasing need for customized machine learning applications and available data demands for an exploitation of popular distributed frameworks like MapReduce or Spark. However, these data-parallel frameworks represent significant challenges in terms of productivity and translational algorithmic errors to the end-users. The paper aims at introducing Apache SystemML to address these challenges using declarative Machine Learning It aims at scaling beyond main memory, moving towards a declarative style of programming, a less painful implementation for distributed programs.

The paper describes the end to end approach of SystemML with Apache Spark, providing insights like decoupling of physical implementation from program design and holistic inter-operator program optimization. It also delves into various optimizers and runtime techniques with transparenting switching and automatic detection of LA optimizations improving performance and usability.

Strengths

1. Due to the ability to express custom algorithms in familiar language covering the relevant linear algebra primitives and statistics, SystemML leads to an increase in productivity for end users.
2. By applying cost-based compilation techniques thereby generating efficient execution plans for in-memory and distributed operations,SystemML offers a transparent running of ML algorithms on distributed, data-parallel frameworks.
3. Open-source. It allows the RDBMS community to leverage SystemML as a benchmark and testbed for further potential research.

Weakness

1. Physical Data Independence is often noted to be poor, thereby facilitating a need to decide on the questions of dense vs sparse, distributed vs local, which data type to use etc a priori.
2. A significant challenge is represented in the form of Tuning, which requires evident systems knowledge of caching and buffer pools.
3. Also tuning might be workload specific, while having exhaustive parameters could lead to tuning fatigue. Labor intensive especially for RDBMS.

Scalable Linear Algebra on a Relational Database System

With data analytics and machine learning becoming an integral part for most RDBMS, there seems to be a growing demand for scalable linear algebra. Linear Algebra is an integral part as most common statistical algorithms can be expressed with matrix transformations. The goal has been to optimize these primitive operations in parallel or distributed database systems. In most relational systems, there is a support for cost-based optimizations, vital for scalability of computations and prominent for its ability to make relational systems scale.

The paper presents a modified parallel/distributed RDBMS as a competitive platform for scalable linear algebra. The authors reason that new systems are not absolutely necessary with existing relational technology capable of being used as a foundation. The results suggest that the addition of scalable linear algebra must be on top of the structured (relational) data abstractions present in the system rather than being directly on top of the raw data-flow operators of the system.

Strengths

1. A performant SQL can be virtually turned into a performant execution engine for linear algebra through a few lines of changes.
2. The ETL nightmare of “extract-transform-reload” can be eliminated if the aim is to perform analysis of stored data in RDBMS.
3. A domain specific language or API can potentially use the high-level linear algebra transformations, thereby translating the computation to a database computation with no need to implement this from scratch all over again.

Weakness

1. There is a possibility that the optimizer would be unable to optimize the order of a chain of distributed multiplication of matrices expressed in SQL.
2. Key choices regarding matrix blocking and chunking must be made by the end-user to implement distributed matrix operations.
3. An end user may get confused by the extended SQL operations instead of the high level language interface offered in SystemML and Riot.