Review Report-2

By Harsha Salim (USC ID: [redacted])

Paper: E. Begoli, et al., Apache Calcite: A Foundational Framework for Optimized Query Processing Over Heterogeneous Data Sources, in SIGMOD

Summary

This paper's goal is to introduce and explain the working of Apache Calcite, a software framework that provides query processing, optimization and query language support to many popular data processing systems. It starts off by talking about its history, which includes the need for such a system. Earlier, conventional relational database systems were heavily used, and there were many use cases that were hindered by its general-purpose design, which lead to the advocacy for "specialized engines" like column stores and stream processing engines. However, the emergence of specialized systems resulted in two major problems. The first was that there were additional elements to be implemented, like query optimizers, support for query languages and language integrated queries. Modifying just the underlying storage introduced this additional work, which was shared among all the specialized stores', which in turn would result in each one of them implementing these common components in their own way, which is a waste of effort. This highlights the need for a unifying framework, which was provided by Calcite. The second problem was the need to integrate several specialized systems together. Organizations may have varied needs, and hence would require several specialized systems. Hence there was a requirement to support optimized queries across heterogeneous data stores. This was also solved using Calcite. It provided a unified query optimization framework, which was extensible and flexible across different types of data processing systems. This adaptability is a significant technical contribution which lead to its wide adoption in the industry and in the open-source community. The framework provides support for various query languages and execution models, facilitating the use of SQL and its extensions across different data sources. Calcite's architecture employs a dynamic programming approach for query optimization, which is a notable enhancement over other existing systems. This method helps avoid local minima in query execution plans, contributing to more efficient query processing. Its optimizer uses a tree of relational operators as its internal representation. The optimization engine primarily consists of three components: rules, metadata providers, and planner engines. It has uses adapters to define table schemas and views in external storage systems. It has multiple pluggable components that provides default behavior which can be replaced by custom behavior. Apart from the common data manipulation operators, Calcite also has support for other operators, such as window operator, which is very useful for analytical queries. It also has a feature called Traits, to describe and control the physical properties of data, which is very useful during optimization. The paper also mentions the extensions that Calcite provides with respect to semi-structured data, streaming and geospatial queries, and language-integrated query language. This is following by an enumeration of how and to what extent Calcite was

adopted by existing storage systems. It then outlines future work aimed at enhancing Calcite's capabilities, including the development of new adapters for non-relational data sources and improvements to its modular planner design. These future directions emphasize ongoing innovation and adaptation to emerging data management practices. This paper positions Apache Calcite as a pivotal framework in the landscape of data processing and query optimization, addressing the complexities associated with heterogeneous data environments.

Paper Strengths

The paper makes a strong point regarding the need for a unifying storage optimization framework. This need arises as organizations increasingly utilize specialized systems like Apache Hive, Storm, etc., that often lack a unified optimization framework. This additional complexity of implementing optimizers and support for SQL and other query languages was duplicated work for developers. By proposing a comprehensive solution through Apache Calcite, the second problem of integration of several specialized systems together was also solved. The integration of diverse systems is of crucial importance in analytics, making it highly applicable in real-world scenarios where multiple data sources are utilized.

The framework employs dynamic programming techniques, which enhances the efficiency of query processing by avoiding local minima in execution plans, which is a significant improvement over other existing systems. It also extends the basic query operators to support more complex operations, enabling it to efficiently recognize optimization opportunities like the window operator, which is very useful for analytical queries. The paper describes many more features of Calcite that help in query optimization.

Another strength about the paper is the in-depth explanation about its architecture and its various elements. Each component was described, and its impact on the product was also discussed. For example, since Calcite is not a pure database system, rather a framework that sits on top of existing systems, it does not have a storage layer with schemas and data models defined. So, the paper describes adapters, a component of Calcite through which schema information can be defined. This helps the readers to understand how it can function and can be integrated with other systems, thereby providing a more holistic view of where Calcite would be present in the data management system from a high-level perspective.

The paper also mentions the pluggable components of Calcite, which make it a versatile framework for query processing and optimization, accommodating a wide range of data models and processing engines. Most of the components, including the optimizer, the planner engines, language extensions and view substitutions are pluggable. This modularity not only enhances performance but also allows for easy customization to fit specific use cases.

Paper Weaknesses

Calcite includes a performance testing module but does not evaluate query execution directly. This limitation means that assessing the performance of systems built with Calcite is challenging, particularly when comparing it to similar frameworks. Fair comparisons are complicated due to significant engineering and architectural changes in the different systems. However, this also means that we cannot compare Calcite with other frameworks using any performance metrics, nor can we craft a fair benchmark. The unifying nature of Calcite, along with its versatility would make it difficult to create heterogeneous benchmarks that can effectively evaluate its performance.

While the paper mentioned that it was unable to compare its performance with respect to similar frameworks, it does not mention the limitations of Calcite compared to the other frameworks. This lack of clarity can mislead readers about its capabilities and applicability.

Due to the complex nature of the framework, it seems reasonable that there are many components to cover in the paper. However, for readers who may not have a strong background in query optimization frameworks, this complexity can make it difficult to grasp the fundamental ideas being presented. For instance, the section regarding using lattices for materialized views is very brief and does not provide sufficient explanation regarding how it is implemented, or how it helps in query processing and optimization.

Opinion

Despite the weaknesses mentioned above, particularly the lack of benchmarking, the paper makes a strong case for Calcite's adoption in complex systems. The modularity, extensibility, and optimization techniques it introduces are valuable contributions. While a direct comparison with similar frameworks would strengthen the argument, the paper still provides meaningful insights. Hence, I accept this paper.