Reading Summary of Spark Sql: Relational Data Processing in Spark

T11 PangCloud: Tianyang Liu, Yicheng Wang, Su Pu

Link: https://docs.google.com/document/d/1cNBARx9cXq9rqyVNKNcy2SMybb_fBiOMpba6Hz8UyQc/edit

Spark Sql is a spark module for structured data processing. Different from the basic APIs of Spark Rdd, Spark Sql interface contains more information on data structure and execution schedule. Inside Spark, Sql utilizes the information to enhance optimization. There are several ways to run Spark Sql: basic Sql, DataFrames API, and Datasets API. When various computational engines are called to execute the same task, we could pick different APIs and languages to do. This unification implies that developers could easily switch among most familiar APIs to finish the same computation task.

This paper introduces Spark Sql which is a relational data processing framework utilizing Spark functional programming API. Spark Sql through DataFrame APIs and programming Spark code, provides a fusion process of relation and process. It also realizes a optimizer that could be extended, with the name of Catalyst. Through using the functional programming language Scala, users can easily realizes rule based optimization in Catalyst. This article also proposes the abstraction of Spark Sql API called DataFrame. DataFrame is a distributed set of identical mode rows, whose concept is similar to the table in a database. It also provides an interface similar to Spark Rdd for operations. DataFrame supports all common relational operators. We can construct DataFrames with programming language local object Rdd. Spark Sql can use refraction to automatically deduce the mode of these objects. Spark Sql could use array storage to cache the hot data in memory. Spark Sql implements array compression to save memory use. It could through caching data in memory to accelerate mutual search and iteration algorithms. Catalyst optimizer will use search plan as an internal tree in storage. After executing Sql search, it firstly decode it as grammar tree, then use application base data to analyze unsolved logic plans. And it will stop useless search ahead of time. Then for useful plans, Catalyst uses rule based optimization. And from optimized logic plan, it generates several physical plan, and choose a physical plan based on cost model. Last, choose physical transformation as Rdds and use Spark to execute. For better accomodation, Spark Sql can also integrate with other tools like the Spark machine learning library.

Four strong points of this paper are:

First, DataFrame API can both save time and storage. DataFrame is the main abstraction in Spark Sql's API, which is a distributed collection of rows with a homogeneous schema.The DataFrame storage format is very specail compare to existing data frame APIs in R and Python, it can automaticcaly store data in a columnnar format which is significantly more compact than Java/Python objects, therefore it is faster and take less storage than other format. Beside. DataFrame can be meanipulated in RDD (distributedbuted collections in Spark), and more than RDDs, it can keep track of native Java/Python objects. Another feature of the SparkFrame is a lazy scheme, each DataFrame object is a logical plan to compute a dataset and will not be executed until the user give a special "output operation" call, it provide a lot of freadom to let users optimize the operations and save a lot of time doing redundant execution.

Second, constructed the Catalyst Optimizer to better work for Spark Sql. By using the Catalyst optimizer, it is easier to add new optimizatin techiques and featres to the Spark Sql. And also enable external developers to extend the optimizer. Compared with other extensible optimizers, Cayalyst can use the pattern-matching programming language: Scala, to not only make rules easy to specify but also let the developers use the full programming language.

Third, the main advantage of handling "big data". The paper consider of the "big data" enviornment which is unstructed and semistructured. The Spark Sql includeds a schema algorithm for JSON and other semistructured data. The paper evaluate the algorithm with the Spark's Machine Learning Library which is also using DataFrames. Illustrate the system is good for dealing with machine learning pipeline with Json data source.

Finally, the paper addressed DataFrame API which have improve performance for the pipline computation and developers can proposed their own piplines easily. In brief, the main advantages of Spark Sql is to support a claiming DataFrame API that allows relational processing, as well as the functional programming language. This is because the pattern alignment function provided by Scala makes rule based optimization much more easier.

Weak points of this paper are: First, Catalyst search optimizer is very simple and basic. Currently Catalyst can just support some simple rules like filter down push. One more complicated search optimizer would be ideal

because it is of vital importance to the performance. Second, Spark Sql cache manager asks users to provide cache notice, but we hope that in the future Spark Sql could support an automatic realization table.

The Spark Sql in the paper still lacks of predicate pushdown for key-value, it can introduced support for limited forms of filtering like Hbase and Cassandra in the future.

References:
[1] Meng X, Bradley J, Yuvaz B, et al. Mllib: Machine learning in apache spark[J]. JMLR, 2016, 17(34): 1-7.
[2] Nothaft F A, Massie M, Danford T, et al. Rethinking data-intensive science using scalable analytics systems[C]//Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015: 631-646.