Hive SARGs Proposal

Owen O'Malley

22 May 2013

1 Overview

The goal is to present the predicates that are pushed down to Hive's RecordReaders via hive.io.filter.expr.serialized in an easy to use format. In particular, the current serialized form of the filter expression contains the relevant filter expression, but also includes many details that would need to be understood and processed by each reader wishing to filter rows. For example, in the current expression provided to the RecordReader, the simple condition $name =$ "$Fred$" is represented by an expression tree of *ExprNodeGenericFuncDesc* with an operator node of *GenericUDFOPEqual* to represent the = operator and children of *ExprNodeColumnDesc* for the column and *ExprNodeConstantDesc* for the string literal. The intent of the new API is that it can be supported by HCatalog and used in MapReduce and Pig without an undue effort.

The original use case is to optimize the ORC record reader to skip over the groups of rows based on the index information which contains the minimum and maximum for each column. By using the SARGs, the reader can avoid reading, decompressing, and parsing large numbers of rows.

The primitive clauses consist of:

- $Column > Literal$
- $Column >= Literal$
- $Column < Literal$
- $Column >= Literal$
- $Column <> Literal$
- $Column = Literal$
- $Column\ in\ \{Literal1, ...\}$
- $Column\ between\ Literal1, Literal2$

The types of the primitive operators will include:

- long (represents boolean, tinyint, smallint, int, and bigint)
- double (represents float and double)
- string

The $\neg$ operator is included rather than using the opposite operator because *NULL* handling means that $\neg(a > b)$ is not the same as $a <= b$.

To simplify evaluation, especially in the presence of "maybe" answers, the code will present the formula in conjunctive normal form where the top-level operator is $\wedge$ and each subexpression only has a single $\vee$ operator with primitive clauses that may be negated.

The normalization is done by pushing down the negation to the leaves and removing even number of negations. Finally, any top level disjunctions are distributed over the conjunctions.

- $A \vee (B \wedge C)$ becomes $(A \vee B) \wedge (A \vee C)$
- $\neg(A \vee \neg B)$ becomes $\neg A \wedge B$

By putting the expression in a normal form, the expression evaluation can be optimized by evaluating the sub-expressions with the fewest primitive cluases first.

All functions and udfs will be considered as indeterminate and eliminate the disjunction sub-expression that they occur in.

2 Future work

In the future other types will be included, such as timestamp and decimal. The primitive clauses will be extended to include presence in a bloom filter.

Follow up work will use the SARGs to filter the individual rows so that we can deserialize the remaining columns only when the filter is satisfied.