Apache Spark* SQL expression evaluation

Chenzhao Guo

5/13/2017

*Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

Agenda

- Expression, UDF, UDAF & UDTF
- API definition
- Optimization in planning stage
- Code generation: part of Project Tungsten*
- Hivemall*: ML tools based on Hive UDF/UDAF/UDTF
- Future plan: vectorization

*Other names and brands may be claimed as the property of others

Agenda

- **Expression, UDF, UDAF & UDTF**
- API definition
- Optimization in planning stage
- Code generation: part of Project Tungsten*
- Hivemall*: ML tools based on Hive UDF/UDAF/UDTF
- Future plan: vectorization

*Other names and brands may be claimed as the property of others

Expression, UDF, UDAF & UDTF

- Expression(built-in function) represents a part of a structured query
  - Add(+), Coalesce, EqualTo(=), GreaterThan(>), Lower, Factorial, CurrentTimestamp(now)
  - Average, Count, Max (aggregate expression)

```
SELECT average(age) FROM test WHERE company=‘intel’
```

  - Explode (generator)

```
SELECT explode(array(10, 20)) ->
 10
 20
```

- UDF(user defined function): a feature of Spark SQL* to define new column-based functions that extend the vocabulary of Spark SQL
  - UDAF(user defined aggregate function)
  - UDTF(user defined table function)

*Other names and brands may be claimed as the property of others

Agenda

- Expression, UDF, UDAF & UDTF
- **API definition**
- Optimization in planning stage
- Code generation: part of Project Tungsten*
- Hivemall*: ML tools based on Hive UDF/UDAF/UDTF
- Future plan: vectorization

*Other names and brands may be claimed as the property of others

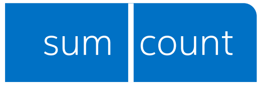
API definition -- Expression

```
abstract class Expression {
  def foldable: Boolean = false
  def nullable: Boolean
  def eval: Any
  def genCode: ExprCode
  …
}
```

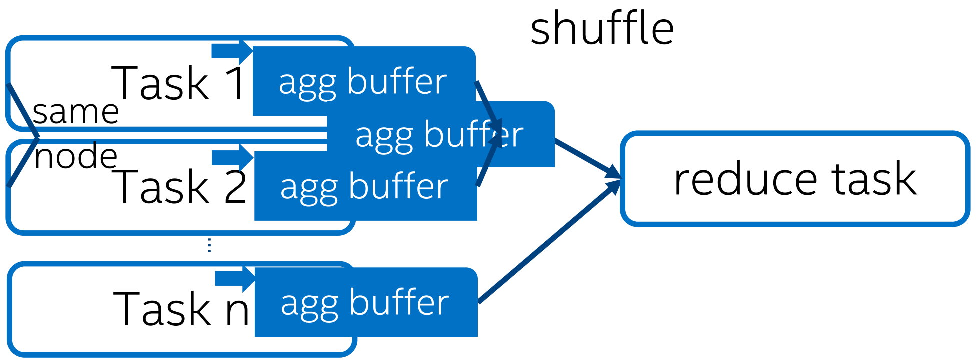
Add(1, 1)	Add(col1, col2)
true	false
false	col1.nullable || col2.nullable

```
case class Add(left: Expression, right: Expression) extends BinaryArithmetic {
  private lazy val numeric = TypeUtils.getNumeric(dataType)
  protected override def nullSafeEval(input1: Any, input2: Any): Any = {
    if (dataType.isInstanceOf[CalendarIntervalType]) {
      input1.asInstanceOf[CalendarInterval].add(input2.asInstanceOf[CalendarInterval])
    } else {
      numeric.plus(input1, input2)
    }
  }
  …
}
```

API definition -- Aggregate Expression



aggregation buffer for average



aggregate without groupby

- Core API :
  - **Initialize** a new buffer
  - **Update** the buffer with a row
  - **Merge** 2 buffers together
  - **Evaluate** final result with the buffer

```
case class Average(child: Expression) extends DeclarativeAggregate {
  override lazy val initialValues = Seq(
          Literal(0),
          Literal(0)
  )
  override lazy val updateExpressions = Seq(
          Add(sum, child),
          If(IsNull(child), count, count + 1L)
  )
  override lazy val mergeExpressions = Seq(
          sum.left + sum.right,
          count.left + count.right
  override lazy val evaluateExpression = sum / count
  ...
}
```

API definition – Easier way to write & register UDF & UDAF

- Register UDF:

```
sqlContext.udf.register("strLen", (s: String) => s.length())
```

- Register UDAF:

```
class MyAggregate extends UserDefinedAggregateFunction {
    …
}
sqlContext.udf.register(“my_aggregate", new MyAggregate )
```

- Pros: easier to implement compared to extending Expressions
- Cons: black box for Spark hence no optimization

Agenda

- Expression, UDF, UDAF & UDTF
- API definition
- **Optimization in planning stage**
- Code generation: part of Project Tungsten*
- Hivemall*: ML tools based on Hive UDF/UDAF/UDTF
- Future plan: vectorization

*Other names and brands may be claimed as the property of others

Optimization in planning stage

- Rules in analysis stage: PromoteStrings, **ImplicitTypeCasts**, DateTimeOperations, …
- Rules in optimization stage: **ConstantFolding**, PushDownPredicate, LikeSimplification, …

```
case class Concat(left: Expression, right: Expression) {
override def inputTypes: Seq[AbstractDataType] = Seq.fill(children.size)(StringType)
…
}
```

- concat example: *SELECT concat(“Spark”, “SQL”) => “SparkSQL”*
- *SELECT concat(framework, version) from version_statistics*

id	framework	version
1	“Spark”	2.1
2	“Flink”	1.2
3	“Hadoop”	2.7

```
def implicitCast(e: Expression, expectedType: AbstractDataType): Option[Expression] = {
    val ret: Expression = (e.datatype, expectedType) match {
        case (any: AtomicType, StringType) if any != StringType => Cast(e, StringType)
    …
    }
}
```

Optimization in planning stage

- Constant Folding in optimization stage

```
abstract class Expression {
  def foldable: Boolean = false
  …
}
```

```
case class Add(left: Expression, right: Expression) {
override def foldable: Boolean = left.foldable && right.foldable
…
}
```

- Add(1,1) -> Add(Literal(1), Literal(1))
- Literal(1).foldable = true
- Add(1,1).foldable = true

➡ Add(1,1) -> 2

```
object ConstantFolding extends Rule[LogicalPlan] {
  def apply(plan: LogicalPlan): LogicalPlan = plan transform {
     case q: LogicalPlan => q transformExpressionsDown {
       case l: Literal => l
       case e if e.foldable => Literal.create(e.eval(EmptyRow), e.dataType)
     }
  }
  …
}
```

Agenda

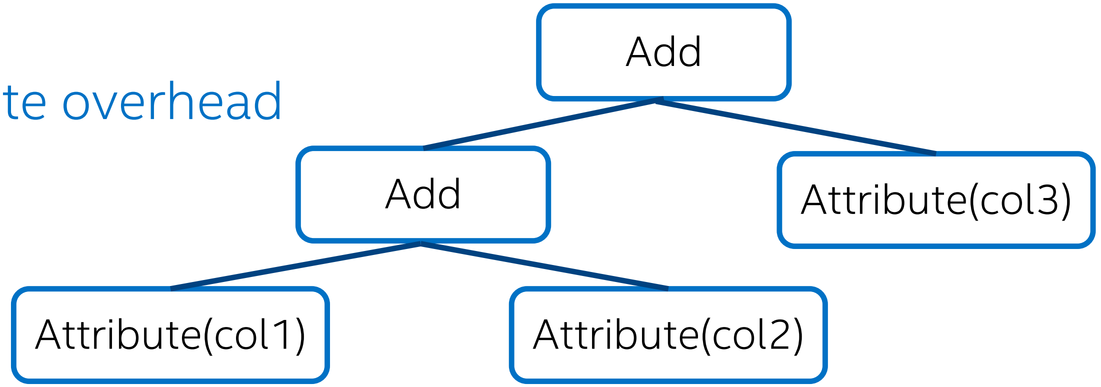
- Expression, UDF, UDAF & UDTF
- API definition
- Optimization in planning stage
- **Code generation: part of Project Tungsten***
- Hivemall*: ML tools based on Hive UDF/UDAF/UDTF
- Future plan: vectorization

*Other names and brands may be claimed as the property of others

Code generation: part of Project Tungsten*

- Code generation:
  - Collapse multiple expressions into one to eliminate overhead

```
abstract class Expression {
  def genCode: ExprCode
  …
}
```



Evaluate col1+col2+col3 from leaf to root if without codegen

- Project Tungsten*:
  - Hardware offers increasingly large network & disk IO bandwith
  - Main focus: better efficiency of memory & CPU
  - *Also includes memory management & cache-aware computation besides codegen*

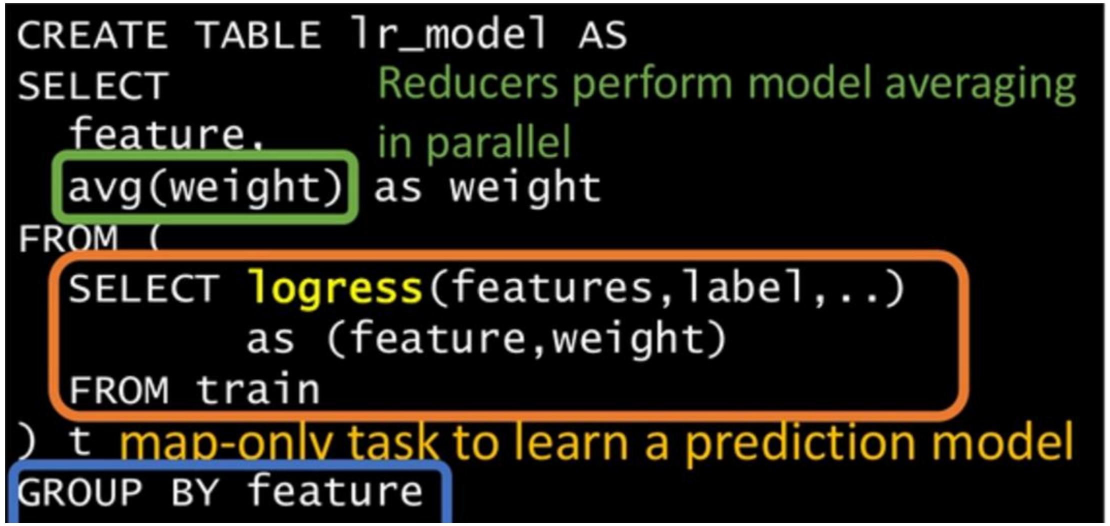
Agenda

- Expression, UDF, UDAF & UDTF
- API definition
- Optimization in planning stage
- Code generation: part of Project Tungsten*
- **Hivemall*: ML tools based on Hive UDF/UDAF/UDTF**
- Future plan: vectorization

*Other names and brands may be claimed as the property of others

Hivemall*: ML tools based on Hive UDF/UDAF/UDTF

- Machine learning on SQL
- Supports Hive, Spark, Pig
- Apache Incubator project



```
CREATE TABLE lr_model AS
SELECT
  feature,
  avg(weight) as weight
FROM (
  SELECT logress(features,label,..)
         as (feature,weight)
  FROM train
) t
GROUP BY feature
```

Reducers perform model averaging in parallel
map-only task to learn a prediction model

Image from https://github.com/apache/incubator-hivemall

- Other state-of-the-art machine learning algorithms: Soft Confidence Weighted, Adaptive Regularization of Weight Vectors, Factorization Machines, and AdaDelta

*Other names and brands may be claimed as the property of others

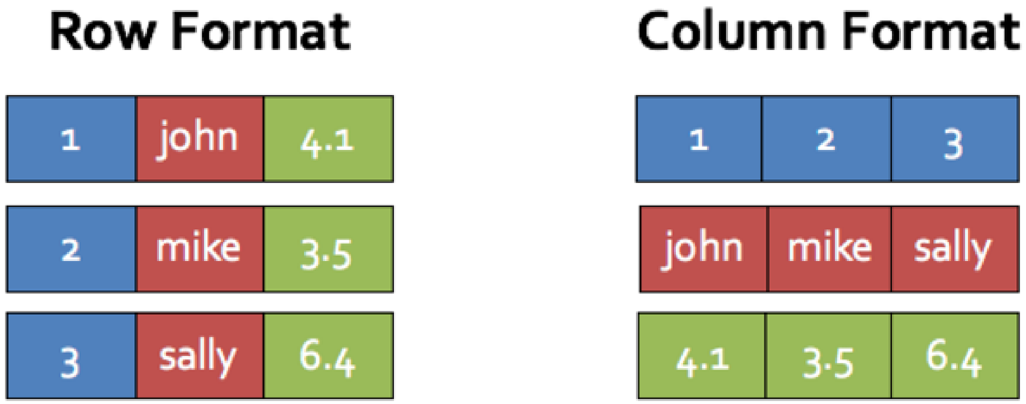
Agenda

- Expression, UDF, UDAF & UDTF
- API definition
- Optimization in planning stage
- Code generation: part of Project Tungsten*
- Hivemall*: ML tools based on Hive UDF/UDAF/UDTF
- **Future plan: vectorization**

*Other names and brands may be claimed as the property of others

- # Future plan: vectorization

- Now: process data one row at a time
- One possible future: batch multiple rows together in a columnar format
- Already implemented: vectorized Parquet reader that does decompression and decoding in column batches

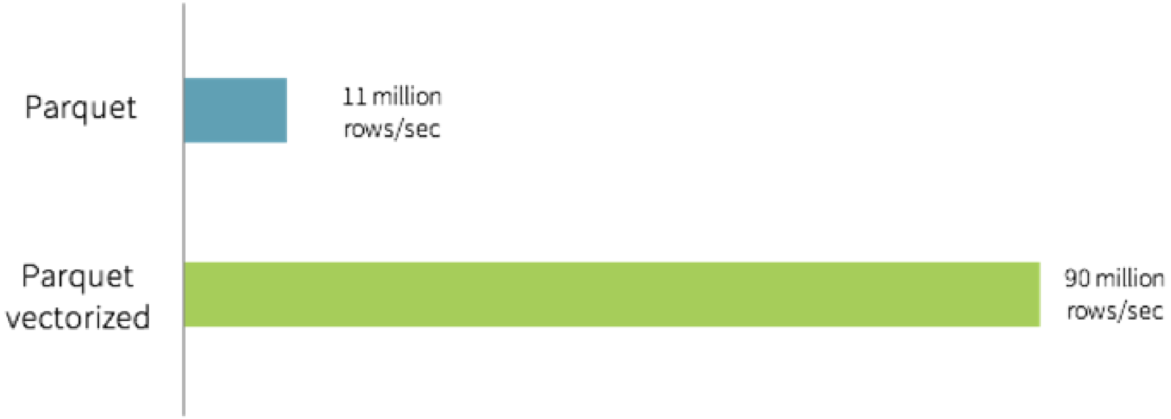


Row Format

1	john	4.1
2	mike	3.5
3	sally	6.4

Column Format

1	2	3
john	mike	sally
4.1	3.5	6.4



Benchmark from Databricks *

For more complete information about performance and benchmark results, visit www.intel.com/benchmarks *Other names and brands may be claimed as the property of others

- Q & A

Legal Disclaimer

- No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.
- The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.
- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].
- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.
- Intel, the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.
- *Other names and brands may be claimed as the property of others.