Apache Spark 2.4 中解决复杂数据类型的内置函数和高阶函数介绍

Apache Spark 2.4 是在11月08日正式发布的，其带来了很多新的特性具体可以参见这里，本文主要介绍这次为复杂数据类型新引入的内置函数和高阶函数。本次 Spark 发布共引入了29个新的内置函数来处理复杂类型（例如，数组类型），包括高阶函数。

如果想及时了解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

在 Spark 2.4 之前，为了直接操作复杂类型，有两种典型的解决方案：

- 将嵌套结构展开为多行，并应用某些函数，然后再次创建结构；
- 编写用户自定义函数（UDF）。

新的内置函数可以直接操作复杂类型，高阶函数可以使用匿名 lambda 函数直接操作复杂值，类似于UDF，但具有更好的性能。

在本博客中，通过一些示例，我们将展示一些新的内置函数以及如何使用它们来处理复杂的数据类型。

典型处理方式

让我们首先通过以下示例来回顾一下 Spark 2.4 之前的典型解决方案。

Option 1 – Explode and Collect

我们使用 explode 函数将数组拆分成多行，并计算 val + 1，最后再使用 collect_list 重构数组，如下所示：

```
SELECT id,
       collect_list(val + 1) AS vals
FROM   (SELECT id,
               explode(vals) AS val
        FROM iteblog) x
GROUP BY id
```

这种方式容易出错并且效率低下，主要体现为三个方面。
首先，我们必须努力确保通过使用唯一键（unique key）来进行分组以便将新生成的数组完全组成为原始数组。其次，我们需要进行 group by 操作，这意味着需要进行一次 shuffle 操作; 但是 shuffle 操作并不保证重组后的数组和原始数组中数据的顺序一致；最后，使用这种方式非常低效。

Option 2 – User Defined Function

接下来，我们选择使用 Scala UDF，它可以接收 Seq[Int] 并对其中每个元素进行加 1 操作：

```
def addOne(values: Seq[Int]): Seq[Int] = {
  values.map(value => value + 1)
}
val plusOneInt = spark.udf.register("plusOneInt", addOne(_: Seq[Int]): Seq[Int])
```

或者，我们也可以使用 Python UDF，如下：

```
from pyspark.sql.types import IntegerType
from pyspark.sql.types import ArrayType

def add_one_to_els(elements):
return [el + 1 for el in elements]

spark.udf.register("plusOneIntPython", add_one_to_els, ArrayType(IntegerType()))
```

然后我们可以在 SQL 里面如下使用：

SELECT id, plusOneInt(vals) as vals FROM iteblog

这种方式更加简单快速，并且可以避免很多陷阱。但这种方式可能仍然效率低下，因为 Scala 或 Python 中的数据序列化可能很昂贵。

新的内置函数

下面我们来看看直接操作复杂类型的新内置函数。 《Apache Spark 2.4 新增内置函数和高阶函数使用介绍》 列举了每个函数的示例。
每个函数的名称和参数标注了它们处理数据类型，T 或 U 表示数组；K，V 表示 map 类型。

高阶函数（Higher-Order Functions）

为了进一步处理数组和 map 类型，我们使用了 SQL 中支持的匿名 lambda 函数或高阶函数语法，使用 lambda 函数作为参数。lambda 函数的语法如下：

```
argument -> function body
(argument1, argument2, ...) -> function body
```

符号 ->
左边表示参数列表，符号右边定义函数体，在函数体中可以使用参数和其他变量来计算新的值。

使用匿名 Lambda 函数进行转换

首先我们来看一下使用带有匿名 lambda 函数的 transform 函数的例子。假设有一个表，包含三列数据：integer 类型的 key，integer 数组的 values，Integer 类型的二维数组 nested_values。如下：

key	values	nested_values
1	[1, 2, 3]	[[1, 2, 3], [], [4, 5]]

当我们执行下面 SQL 的时候：

SELECT TRANSFORM(values, element -> element + 1) FROM iteblog;

transform 函数通过执行lambda
函数遍历数组种的每个元素并进行加一操作，然后创建一个新数组。
除了参数之外，我们还可以在 lambda 函数中使用其他变量，例如：key，这是表的另外一列：

```
SELECT TRANSFORM(values, element -> element + key) FROM iteblog;
```

如果你想要处理更复杂的嵌套类型，比如表中的 nested_values 列，你可以使用嵌套的 lambda 函数：

```
SELECT TRANSFORM(
   nested_values,
   arr -> TRANSFORM(arr,
    element -> element + key + SIZE(arr)))
FROM iteblog;
```

在内层的 lambda 函数中你可以使用 key 和 arr 这些在 lambda
函数上下文之外的变量以及表的其他字段值。

总结

Spark 2.4 引入了 24 个新的内置函数，如 array_union，array_max ，array_min等，以及 5 个高阶函数，如 transform, filter
等，这些函数都可以用于处理复杂类型。完整的列表可以参见《Apache Spark 2.4 新增内置函数和高阶函数使用介绍》。

本文英文原文：Introducing New Built-in and Higher-Order Functions for Complex Data Types in Apache Spark 2.4

本博客文章除特别声明，全部都是原创！
原创文章版权归过往记忆大数据（过往记忆）所有，未经许可不得转载。
本文链接:【】（）