Spark SQL 查询中 Coalesce 和 Repartition 暗示（Hint）

如果你使用 Spark RDD 或者 DataFrame 编写程序，我们可以通过 coalesce 或 repartition 来修改程序的并行度：

```
val data = sc.newAPIHadoopFile(xxx).coalesce(2).map(xxxx)
或
val data = sc.newAPIHadoopFile(xxx).repartition(2).map(xxxx)

val df = spark.read.json("/user/iteblog/json").repartition(4).map(xxxx)
val df = spark.read.json("/user/iteblog/json").coalesce(4).map(xxxx)
```

通过 coalesce 或 repartition 函数我们一方面可以减少 Task 数据从未达到减少作业输出文件的数量；同时我们也可以加大并行度从而提高程序的运行效率。

我们现在越来越多的人使用 Spark SQL 来编写程序，可是在 Spark 2.4 之前，我们是不能直接在 SQL 里面使用 coalesce 或 repartition的。值得高兴的是，国内的开发者为 Spark SQL 开发了一个功能，使得我们在 Spark SQL 里面也能用这两个函数，详见 SPARK-24940 。这个功能在 Spark 2.4 已经发布了，这样我们可以通过 COALESCE 或 REPARTITION 关键字暗示来设置程序的并行度。使用如下：

```
package com.iteblog

import java.util.UUID

import org.apache.spark.sql.SparkSession

object Iteblog {

  case class Person(name: String, age: Int)

  def main(args: Array[String]) {

    val spark = SparkSession
      .builder()
      .appName("iteblog example")
      .master("local[2]")
      .enableHiveSupport()
      .getOrCreate()

    // For implicit conversions like converting RDDs to DataFrames
    import spark.implicits._

    val person = 1.to(10000).map { i =>
      Person(UUID.randomUUID().toString.substring(1, 6), i % 100)
    }

    val df = spark.sparkContext.parallelize(person,2).toDF()
    df.createOrReplaceTempView("person")
    spark.sql("create table iteblog0 as select age,count(*) from person where age between 10 a
nd 20 group by age").explain()
  }
}
```

上面程序的物理计划如下：

```
== Physical Plan ==
Execute CreateHiveTableAsSelectCommand CreateHiveTableAsSelectCommand [Database:defa
ult}, TableName: iteblog0, InsertIntoHiveTable]
+- *(2) HashAggregate(keys=[age#4], functions=[count(1)])
   +- Exchange hashpartitioning(age#4, 200)
      +- *(1) HashAggregate(keys=[age#4], functions=[partial_count(1)])
         +- *(1) Project [age#4]
            +- *(1) Filter ((age#4 >= 10) && (age#4 <= 20))
```

```
            +- *(1) SerializeFromObject [staticinvoke(class org.apache.spark.unsafe.types.UTF8String, StringType, fromString, assertnotnull(input[0, qwe.App$Person, true]).name, true, false) AS name#3, assertnotnull(input[0, qwe.App$Person, true]).age AS age#4]
               +- Scan[obj#2]
```

如果我们加上 REPARTITION 关键字暗示，如下：

```
spark.sql("create table iteblog1 as select /*+ REPARTITION(4) */ age,count(*) from person where age between 10 and 20 group by age").explain()
```

则物理计划变成下面的

```
== Physical Plan ==
Execute CreateHiveTableAsSelectCommand CreateHiveTableAsSelectCommand [Database:default}, TableName: iteblog1, InsertIntoHiveTable]
+- Exchange RoundRobinPartitioning(4)
  +- *(2) HashAggregate(keys=[age#4], functions=[count(1)])
    +- Exchange hashpartitioning(age#4, 200)
      +- *(1) HashAggregate(keys=[age#4], functions=[partial_count(1)])
        +- *(1) Project [age#4]
          +- *(1) Filter ((age#4 >= 10) && (age#4 <= 20))
            +- *(1) SerializeFromObject [staticinvoke(class org.apache.spark.unsafe.types.UTF8String, StringType, fromString, assertnotnull(input[0, qwe.App$Person, true]).name, true, false) AS name#3, assertnotnull(input[0, qwe.App$Person, true]).age AS age#4]
               +- Scan[obj#2]
```

可以看到第四行多了 +- Exchange RoundRobinPartitioning(4)，其他的不变。通过指定 coalesce 或 repartition 暗示，我们就可以在 Spark SQL 里面指定并行度。

注意，如果你使用 Spark 2.4 以下版本，在 Spark SQL 里面加入 /*+ REPARTITION(4) */ 暗示，语句也不会运行错误，只不过并不会修改如何并行度相关属性而已。