链滴

(数据开发篇)-001-ORC 文件谨慎 ALTER

作者：sixleaves

原文链接：https://ld246.com/article/1579175863614

来源网站：链滴

一、背景

● 由于公司很多表因为历史原因， 一开始都没有使用 ORC 格式，而是直接才用了 TextFile 格式，着业务的迭代， 表越变越大，导致存储资源十分紧张。为了节省存储空间，经过调研，我们决定采用 ORC 格式，大概可以节省 75% 的存储空间， 对于 PB 级别的数据， 是十分可观的。

● 在做完 ORC 格式后，我们又对这些表进行拉链操作。拉链的过程，遇到一个十分怪异的错误。不使用 Hive 还是 Spark 都会抛出异常，导致 SQL 失败。

Hive 报错：

```
Error: java.lang.RuntimeException: org.apache.hadoop.hive.ql.metadata.HiveException: Hive Runtime Error while processing row [Error getting row data with exception
java.lang.ClassCastException: java.util.ArrayList cannot be cast to org.apache.hadoop.io.Text
```

Spark 报的警告信息(对敏感字段作了打码)：

```
20/01/09 15:55:34 [main] WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<[redacted]:string,[redacted]:string,dt:string>) is different from the
schema when this table was created by Spark SQL(struct<[redacted]:string,[redacted]:array<string>,dt:string>). We have to fall back to the table schema from Hive metasto
re which is not case preserving.
```

1.1 问题表现：

● 该问题出错的表为分区表，当你 `select * from xxxx where dt='yyyy-mm-dd'` **查询最近一天的区** 能够成功，但是**查询第一天**或者**不限制分区**查询，却会抛出上述的错误。

二、问题定位

2.1 通过更加具体的报错提示，确定大体问题原因。

● 通过结合 Hive 的报错提示和 Spark 的警告信息，我们可以大概知道问题的原因是因为， **元数据录的字段类型和真实的字段类型不一致，导致强制类型转换抛出异常。** 但是具体的原因，依然我们进步排查。

● 在 Hive 开发的 Jira 上可以找到以下几个 issue.

ORC Schema Evolution Issues

https://issues.apache.org/jira/browse/HIVE-10591

https://issues.apache.org/jira/browse/HIVE-11981

通过上述三个 issue 的描述，可以知道两个 ORC 相关的重要信息

- 从 **Hive2.1 之前** 是 **不支持修改字段类型**.
- 从 **Hive2.1 开始** 才 支持**字段类型**的更改。
- 从 **Hive2.2 开始**,才 支持 **字段顺序**的更改。

而我公司使用的 Hive 是 1.1.0,自然不能够支持 Hive 的字段类型修改。

所以问题大体原因： 应该是由于**该表的字段类型做过变更， 由于该表又是 ORC 格式，导致无法兼旧的分区，但是对于新的分区则按更改后的字段类型，生成对应格式的 ORC 文件。**

2.2 通过 Git 查看了 SQL 代码的更改时间，进行印证。

- 既然找到了，大体问题，为了进一步印证， 我们过 Git 对比下更改前和更改后的版本。
- 最后发现更改后 sql,将原先的 Array 类型改成了 String 类型。

2.3 寻找开始有问题的分区。

- 通过 2.1 和 2.2 我们可以断定问题的原因。 **但是如何确定从哪一天的分区开始不兼容的？我们这表有 400 多个分区，如何快速的进行查找?**
- 由于表是分区表，那么这两部分不兼容的分区肯定是集中在一起，并且应为分区的是按时间递增，以**我们可以用二分查找来定位这个问题。**
- 最后定位到了 2019-02-20 号这个分区是最后一个旧格式分区，21 号开始就是新格式分区。所以以推断 **2019-02-20 之前的分区都是旧格式分区。**

那么我们要如何解决不兼容的分区？总不该把数据进行删除了，博主想了两种办法。 下面就以模拟的区表来做验证。

三、 问题模拟

```
create database tmp;
create table tmp.test(
    id string,
        apps Array<string>
) partitioned by(dt string)
stored as orc;

insert overwrite table tmp.test partition(dt='2020-01-13')
select '1', array("a-app","b-app");

alter table tmp.test change apps apps array<array<string>>;

insert overwrite table tmp.test partition(dt='2020-01-14')
select '1', array(array("c-app","d-app")))
select * from tmp.test;
```

插入模拟数据

```
spark-sql> create table tmp.test(
         >     id string,
         >         apps Array<string>
         > ) partitioned by(dt string)
         > stored as orc;
20/01/16 19:18:35 WARN HiveMetaStore: Location: hdfs://localhost:8020/user/hive/warehouse/tmp.db/test specified for non-external table:test
Time taken: 0.955 seconds
spark-sql>
         > insert overwrite table tmp.test partition(dt='2020-01-13')
         > select '1', array("a-app","b-app");
20/01/16 19:18:36 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
20/01/16 19:18:38 WARN log: Updating partition stats fast for: test
20/01/16 19:18:38 WARN log: Updated size to 410
Time taken: 1.985 seconds
spark-sql>
```

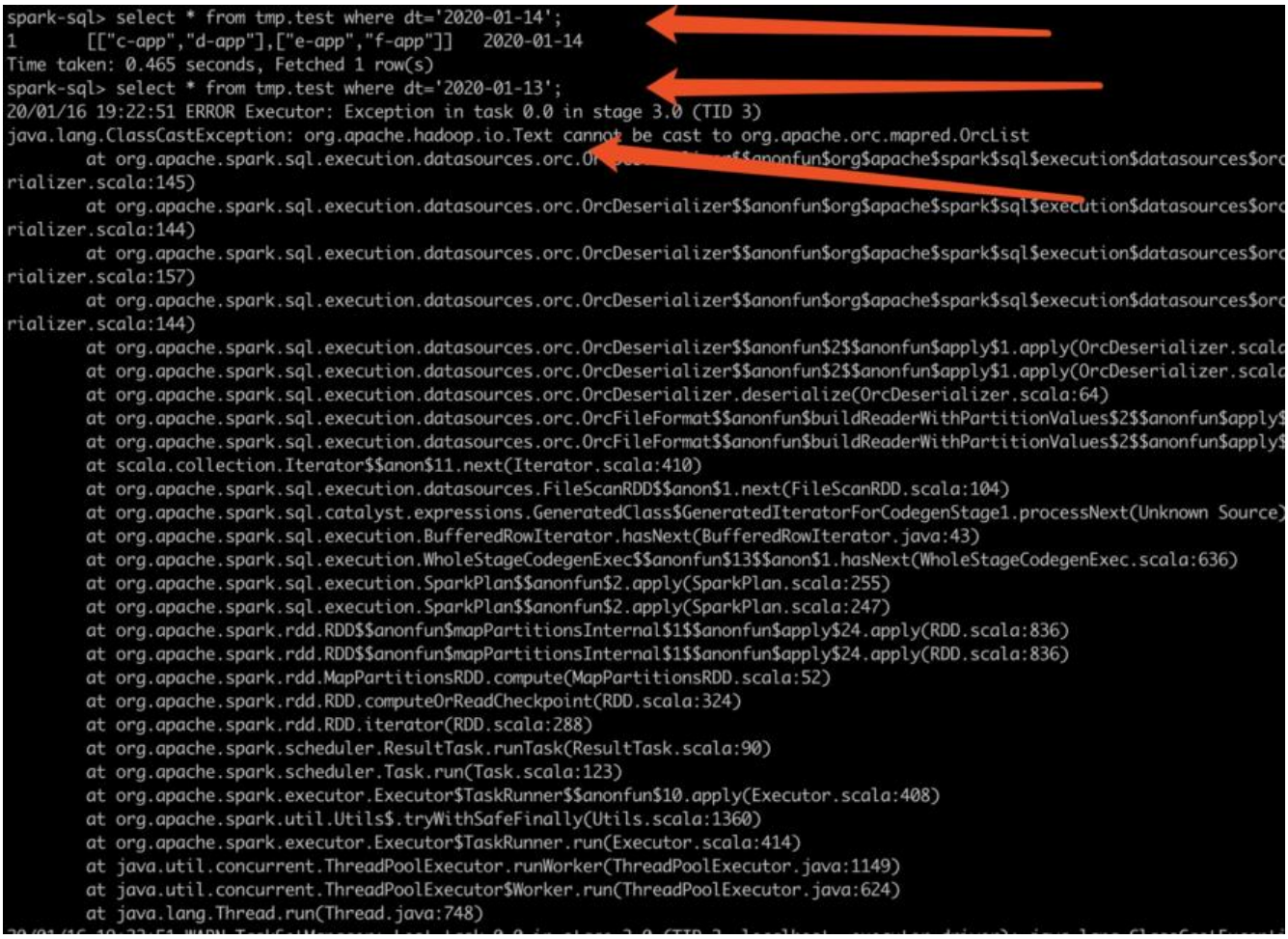
使用Hive Alter

```
hive> alter table tmp.test change apps apps array<array<string>>;
OK
Time taken: 0.142 seconds
hive>
```

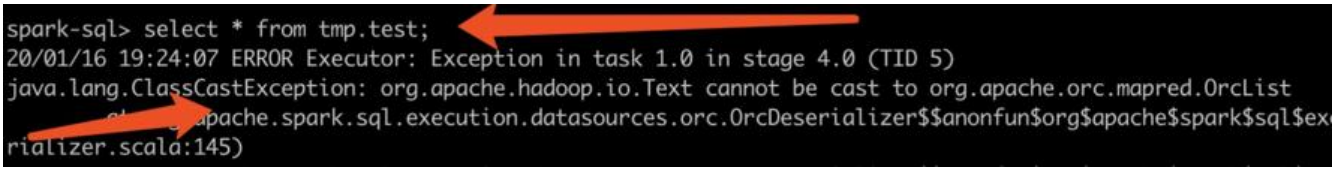
Alter后插入新数据

```
spark-sql>
         > insert overwrite table tmp.test partition(dt='2020-01-13')
         > select '1', array("a-app","b-app");
20/01/16 19:18:36 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
20/01/16 19:18:38 WARN log: Updating partition stats fast for: test
20/01/16 19:18:38 WARN log: Updated size to 410
Time taken: 1.985 seconds
spark-sql> insert overwrite table tmp.test partition(dt='2020-01-14')
         > select '1', array(array("c-app","d-app"),array("e-app","f-app"));
20/01/16 19:21:06 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<id:string,apps:array<array<string>>,dt:
 this table was created by Spark SQL(struct<id:string,apps:array<string>,dt:string>). We have to fall back to the table schema from
ng.
20/01/16 19:21:07 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<id:string,apps:array<array<string>>,dt:
 this table was created by Spark SQL(struct<id:string,apps:array<string>,dt:string>). We have to fall back to the table schema from
ng.
20/01/16 19:21:07 WARN log: Updating partition stats fast for: test
20/01/16 19:21:07 WARN log: Updated size to 463
20/01/16 19:21:07 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<id:string,apps:array<array<string>>,dt:
 this table was created by Spark SQL(struct<id:string,apps:array<string>,dt:string>). We have to fall back to the table schema from
ng.
20/01/16 19:21:07 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<id:string,apps:array<array<string>>,dt:
 this table was created by Spark SQL(struct<id:string,apps:array<string>,dt:string>). We have to fall back to the table schema from
ng.
20/01/16 19:21:07 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<id:string,apps:array<array<string>>,dt:
 this table was created by Spark SQL(struct<id:string,apps:array<string>,dt:string>). We have to fall back to the table schema from
ng.
20/01/16 19:21:07 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<id:string,apps:array<array<string>>,dt:
 this table was created by Spark SQL(struct<id:string,apps:array<string>,dt:string>). We have to fall back to the table schema from
ng.
Time taken: 0.735 seconds
```

指定分区查询



全表查询



可以很清晰的看到更改完格式后,旧的分区数据无法正常读取,但是读取新的分区格式却没有问题.

解决方案

这里提供两种解决方法的思路

- 方案一、通过外表挂载回写数据.
- 方案二、将 orc 文件拷贝到另外一个临时目录，通过内表挂载, 回写数据.

2.1 先建恢复表

```
create table tmp.test_4_recover(
    id string,
        apps Array<string>
) partitioned by(dt string)
stored as orc;
```

2.2 拷贝数据

执行hadoop distcp或者hdfs dfs -cp拷贝数据

```
hdfs dfs -cp /user/hive/warehouse/tmp.db/test/dt=2020-01-13 /user/hive/warehouse/tmp.db
test_4_recover/
```

2.3 挂载分区数据

```
MSCK REPAIR TABLE tmp.test_4_recover;
```

```
hive> MSCK REPAIR TABLE tmp.test_4_recover;
OK
Partitions not in metastore:    test_4_recover:dt=2020-01-13
Repair: Added partition to metastore test_4_recover:dt=2020-01-13
Time taken: 0.278 seconds, Fetched: 2 row(s)
```

2.4 回写数据恢复并查看

```
insert overwrite table tmp.test partition(dt='2020-01-13')
select
  id,
  array(apps)
from
   tmp.test_4_recover where dt='2020-01-13'
```

```
spark-sql> insert overwrite table tmp.test partition(dt='2020-01-13')
         > select
         >    id,
         >    array(apps)
         > from
         >     tmp.test_4_recover where dt='2020-01-13';
20/01/16 19:41:45 WARN log: Updating partition stats fast for: test
20/01/16 19:41:45 WARN log: Updated size to 448
Time taken: 1.152 seconds
spark-sql> select * from tmp.test;
1       [["c-app","d-app"],["e-app","f-app"]]   2020-01-14
1       [["a-app","b-app"]]     2020-01-13
Time taken: 0.12 seconds, Fetched 2 row(s)
```

四、后续

● 如果采用了 ORC 格式，一定要谨慎的 ALTER 修改字段类型。因为这会导致 Hive 的元数据保存的最新的字段类型信息，但是旧的 ORC 文件 底层存储结构并没有发生改变。一旦再去查这些旧的数据会抛出异常。目前直到测试用的最新版本的Hive依然没有修复这个问题。

● 修复方案，推荐采用外表挂载的方式修复。有的公司会对外表进行限制，或者限制不能有两张表指同一个 location, 那么这时候可以先将 旧的 orc 格式文件拷贝到另外一个 临时目录，再建一张临时，将数据挂载到临时表，再用新的逻辑处理这些旧数据，重新插回新表中。

● 如果表是orc文件抛出类试cast to 类型的错误.很可能是因为更改了字段类型导致.

微信阅读体验更佳

数仓之路

微信扫描二维码，关注我的公众号