kudu结合impala的使用(用impala创建kudu的表)

darkalex | 5 | 2019-02-21 14:44

一.从Impala创建一个新的kudu表

从Impala在kudu中创建一个新表类似于将现有的kudu表映射到Impala表,需要自己字段模式和分区信息.

```
CREATE TABLE db_kudu.my_first_kudu_table
(
id bigint,
name string,
PRIMARY KEY(id)
)
PARTITION BY HASH PARTITIONS 16
STORED AS KUDU;
```

在 CREATE TABLE 语句中，必须首先列出构成主键的列。此外，主键列隐式标记为 NOT NULL 。 创建新的 Kudu 表时，需要指定一个分配方案。为了简单起见，上面的表创建示例通过散列 id 列分成 16 个分区。

可以使用 CREATE TABLE … AS SELECT 语句查询 Impala 中的任何其他表或表来创建表。以下示例将现有表 old_table 中的所有行导入到 Kudu 表 new_table 中。 new_table 中的列的名称和类型将根据 SELECT 语句的结果集中的列确定，需另外指定主键和分区。

```
CREATE TABLE db_kudu.new_table
PRIMARY KEY(ts,name)
PARTITION BY HASH(name) PARTITIONS 8
STORED AS KUDU
AS SELECT ts,name,value From du_kudu.old_table;
```

二.指定 Tablet Partitioning (Tablet分区)

表分为每个由一个或多个 tablet servers 提供的 tablets 。理想情况下，tablets 应该相对平等地拆分表的数据。 Kudu 目前没有自动（或手动）拆分预先存在的 tablets 的机制。在实现此功能之前，必须在创建表时指定分区。在设计表格架构时，考虑使用主键，可以将表拆分成以类似速度增长的分区。使用 Impala 创建表时，可以使用 PARTITION BY 子句指定分区：

```
CREATE TABLE cust_behavior (
  _id BIGINT PRIMARY KEY,
  salary STRING,
  edu_level INT,
  usergender STRING,
  `group` STRING,
  city STRING,
  postcode STRING,
  last_purchase_price FLOAT,
  last_purchase_date BIGINT,
  category STRING,
  sku STRING,
```

```
  rating INT,
  fulfilled_date BIGINT
)
PARTITION BY RANGE (_id)
(
    PARTITION VALUES < 1439560049342,
    PARTITION 1439560049342 <= VALUES < 1439566253755,
    PARTITION 1439566253755 <= VALUES < 1439572458168,
    PARTITION 1439572458168 <= VALUES < 1439578662581,
    PARTITION 1439578662581 <= VALUES < 1439584866994,
    PARTITION 1439584866994 <= VALUES < 1439591071407,
    PARTITION 1439591071407 <= VALUES
)
STORED AS KUDU;
```

如果有多个主键列，则可以使用元组语法指定分区边界： （'va'，1）， （'ab'，2）。该表达式必须是有效的 JSON 。

三.分区

PARTITION BY HASH

可以通过散列分发到特定数量的 "buckets" ，而不是通过显式范围进行分发，或与范围分布组合。指定要分区的主键列，以及要使用的存储桶数。通过散列指定的键列来分配行。假设散列的值本身不会表现出显着的偏差，这将有助于将数据均匀地分布在数据桶之间。 可以指定使用复合主键的定义。但是，在多个散列定义中不能提及一列。考虑两列a和b：HASH(a)，HASH(b)，HASH(a，b) × HASH(a)，HASH(a，b)等 没有指定列的 HASH 的 PARTITION BY HASH 是通过散列所有主键列来创建所需数量的桶的快捷方式。如果主键值均匀分布在其域中，并且数据偏移不明显，例如 timestamps (时间戳) 或 IDs (序列号)，则哈希分区是合理的方法。 以下示例通过散列 id 和 sku 列创建 16 个 tablets 。这传播了所有 16 个 tablets 的写作。在这个例子中，对一系列 sku 值的查询可能需要读取所有 16 个 tablets ， 因此这可能不是此表的最佳模式。

```
CREATE TABLE cust_behavior (
  id BIGINT,
  sku STRING,
  salary STRING,
  edu_level INT,
  usergender STRING,
  `group` STRING,
  city STRING,
  postcode STRING,
  last_purchase_price FLOAT,
  last_purchase_date BIGINT,
  category STRING,
  rating INT,
  fulfilled_date BIGINT,
  PRIMARY KEY (id, sku)
)
PARTITION BY HASH PARTITIONS 16
STORED AS KUDU;
```

高级分区

可以组合 HASH 和 RANGE 分区来创建更复杂的分区模式。您可以指定零个或多个 HASH 定义，后跟零个或一个 RANGE 定义。每个定义可以包含一个或多个列。 PARTITION BY HASH and RANGE 考虑上面的 简单哈希 示例，如果您经常查询一系列 sku 值，可以通过将哈希分区与范围分区相结合来优化示例。 以下示例仍然创建了16个 tablets ， 首先将 id 列分为 4 个存储区，然后根据 sku 字符串的值应用范围划分将每个存储区分为四个数据块。至少四片（最多可达16张）。当您查询相

邻范围的 sku 值时，您很有可能只需从四分之一的 tablets 中读取即可完成查询。默认情况下，使用 PARTITION BY HASH 时，整个主键是散列的。要只对主键进行散列，可以使用像 PARTITION BY HASH(id, sku) 这样的语法来指定它。

```
CREATE TABLE cust_behavior (
  id BIGINT,
  sku STRING,
  salary STRING,
  edu_level INT,
  usergender STRING,
  `group` STRING,
  city STRING,
  postcode STRING,
  last_purchase_price FLOAT,
  last_purchase_date BIGINT,
  category STRING,
  rating INT,
  fulfilled_date BIGINT,
  PRIMARY KEY (id, sku)
)
PARTITION BY HASH (id) PARTITIONS 4,
RANGE (sku)
(
  PARTITION VALUES < 'g',
  PARTITION 'g' <= VALUES < 'o',
  PARTITION 'o' <= VALUES < 'u',
  PARTITION 'u' <= VALUES
)
STORED AS KUDU;
```

Multiple PARTITION BY HASH Definitions 再次扩展上述示例，假设查询模式将是不可预测的，但希望确保写入分布在大量 tablets 上，可以通过在主键列上进行散列来实现整个主键的最大分配。

```
CREATE TABLE cust_behavior (
  id BIGINT,
  sku STRING,
  salary STRING,
  edu_level INT,
  usergender STRING,
  `group` STRING,
  city STRING,
  postcode STRING,
  last_purchase_price FLOAT,
  last_purchase_date BIGINT,
  category STRING,
  rating INT,
  fulfilled_date BIGINT,
  PRIMARY KEY (id, sku)
)
PARTITION BY HASH (id) PARTITIONS 4,
             HASH (sku) PARTITIONS 4
STORED AS KUDU;
```

该示例创建16个分区。也可以使用 HASH(id,sku) PARTITIONS 16。但是，对于 sku 值的扫描几乎总是会影响所有16个分区，而不是可能限制为 4 。 Non-Covering Range Partitions Kudu 1.0 及更高版本支持使用非覆盖范围分区，其解决方案如下： • 没有未覆盖的范围分区，在需要考虑不断增加的主键的时间序列数据或其他模式的情况下，服务于旧数据的 tablet 的大小相对固定，而接收新数据的 tablets 将不受限制地增长。 • 在希望根据其类别（如销售区域或产品类型）对数据进行分区的情况下，无需覆盖范围分区，则必须提前了解所有分区，或者如果需要添加或删除分区，请手动重新创建表。 例如引入或消除产品类型。 此示例每年创建一个 tablet （共5个 tablets ），用于存储日志数据。该表仅接受 2012 年至 2016年 的数据。这些范围之外的键将被拒绝。

```
CREATE TABLE sales_by_year (
  year INT, sale_id INT, amount INT,
  PRIMARY KEY (sale_id, year)
)
PARTITION BY RANGE (year) (
  PARTITION VALUE = 2012,
  PARTITION VALUE = 2013,
  PARTITION VALUE = 2014,
  PARTITION VALUE = 2015,
  PARTITION VALUE = 2016
)
STORED AS KUDU;
```

当记录开始进入 2017 年时，他们将被拒绝。在这一点上， 2017 年的范围应该如下：

```
ALTER TABLE sales_by_year ADD RANGE PARTITION VALUE = 2017;
```

在需要数据保留的滚动窗口的情况下，范围分区也可能会丢弃。例如，如果不再保留 2012 年的数据，则可能会批量删除数据：

请注意，就像删除表一样，这不可逆转地删除存储在丢弃的分区中的所有数据。

分区原则

对于大型表格，如事实表，目标是在集群中拥有核心数量的 tablets 。 对于小型表格（如维度表），目标是足够数量的 tablets ，每个 tablets 的大小至少为 1 GB 。 一般来说，请注意，在当前的实现中， tablets 的数量限制了读取的并行性。增加 tablets 数量超过核心数量可能会有减少的回报。

将数据插入 Kudu 表

Impala 允许您使用标准 SQL 语句将数据插入 Kudu 。 此示例插入单个行。

```
INSERT INTO my_first_table VALUES (99, "sarah");
```

此示例使用单个语句插入三行。

```
INSERT INTO my_first_table VALUES (1, "john"), (2, "jane"), (3, "jim");
```

批量插入 批量插入时，至少有三种常用选择。每个可能有优点和缺点，具体取决于您的数据和情况。 Multiple single INSERT statements 这种方法具有易于理解和实现的优点。这种方法可能是低效的，因为 Impala 与 Kudu 的插入性能相比具有很高的查询启动成本。这将导致相对较高的延迟和较差的吞吐量。 Single INSERT statement with multiple VALUES 如果包含超过 1024 个 VALUES 语句，则在将请求发送到 Kudu 之前，Impala 将其分组为 1024 （或 batch_size 的值）。通过在 Impala 方面摊销查询启动处罚，此方法可能会执行比多个顺序 INSERT 语句略好。要设置当前 Impala Shell 会话的批量大小，使用以下语法：set batch_size = 10000;增加 Impala 批量大小会导致 Impala 使用更多的内存。需验证对群集的影响并进行相应调整。 Batch Insert 从 Impala 和 Kudu 的角度来看，通常表现最好的方法通常是使用 Impala 中的 SELECT FROM 语句导入数据。 1、如果您的数据尚未在 Impala 中，则一种策略是 从文本文件（如 TSV 或 CSV 文件）导入。 2、创建 Kudu 表，注意指定为主键的列不能为空值。 3、通过查询包含原始数据的表将值插入到 Kudu 表中，如以下示例所示：

```
INSERT INTO my_kudu_table
SELECT * FROM legacy_data_import_table;
```

在许多情况下，使用 C ++ 或 Java API 直接插入到 Kudu 表中的数据， Impala 可直接查询，而不需要任何 INVALIDATE METADATA 语句或其他 Impala 存储类型所需的其他语句。 INSERT and Primary Key Uniqueness Violations 在大多数关系数据库中，尝试插入已插入的行，则插入将失败，因为主键重复。然而，Impala 不会失败查询。相反，它会生成一个警告，但是继续执行 insert 语句的其余部分。 更新行

UPDATE my_first_table SET name="bob" where id = 3;

1

批量更新 可以使用批量插入中相同的方法 批量更新 。

UPDATE my_first_table SET name="bob" where age > 10;

1

删除行

DELETE FROM my_first_table WHERE id < 3;

1

批量删除 您可以使用 “插入批量” 中概述的相同方法 批量删除 。

DELETE FROM my_first_table WHERE id < 3;

删除表

如果表是使用 Impala 中的内部表创建的，则使用 CREATE TABLE ，标准 DROP TABLE 语法会删除底层的 Kudu 表及其所有数据。如果表被创建为一个外部表，使用 CREATE EXTERNAL TABLE，Impala 和 Kudu 之间的映射被删除，但 Kudu表保持原样，并包含其所有数据。

DROP TABLE my_first_table;

INSERT，UPDATE 和 DELETE 操作期间的故障

INSERT ，UPDATE 和 DELETE 语句不能被视为整体事务。如果这些操作中的一个无法部分通过，其他操作可能已被其他进程修改或删除（在 UPDATE 或 DELETE 的情况下）。 更改表属性 可以通过更改表的属性来更改 Impala 与给定 Kudu 表相关的元数据。这些属性包括表名， Kudu 主地址列表，以及表是否由 Impala （内部）或外部管理。 Rename an Impala Mapping Table

ALTER TABLE my_table RENAME TO my_new_table;

1

使用 ALTER TABLE … RENAME 语句重命名表仅重命名 Impala 映射表，无论该表是内部还是外部表。这样可以避免可能访问基础的 Kudu 表的其他应用程序的中断。 Rename the underlying Kudu table for an internal table 如果表是内部表，则可以通过更改 kudu.table_name 属性重命名底层的 Kudu 表：

ALTER TABLE my_internal_table SET TBLPROPERTIES('kudu.table_name' = 'new_name')

1 2

Remapping an external table to a different Kudu table 如果另一个应用程序在 Impala 下重命名了 Kudu 表，则可以重新映射外部表以指向不同的 Kudu 表名称。

ALTER TABLE my_external_table_ SET TBLPROPERTIES('kudu.table_name' = 'some_other_kudu_table')

1 2

Change the Kudu Master Address

ALTER TABLE my_table SET TBLPROPERTIES('kudu.master_addresses' = 'kudu-new-master.example.com:7051');

```
1  2
```

Change an Internally-Managed Table to External

ALTER TABLE my_table SET TBLPROPERTIES('EXTERNAL' = 'TRUE');

猜你喜欢

评论留言

IT话题 **ActiveMQ学习之增加NIO通讯协议**

IT话题 **“>”（大于号）CSS选择器是什么意思?**

IT话题 **mysql中int（11）的列大小是多少?**

IT话题 **vue 全局前置守卫引起死循环的原因与解决方法**

IT话题 **聊聊artemis消息的推拉模式**

IT话题 **Linux DMA框架简述**

IT话题 **ubuntu 18,.04 搭建gerrit服务器 - gerrit 使用说明**

IT话题 **page结构和物理内存对应关系**

IT话题 **如何按索引从std :: vector <>擦除元素?**

IT话题 **职场神器，免费的PDF编辑你难道不心动吗?**

评论留言

请自觉遵守互联网相关的政策法规，严禁发布色情、暴力、反动的言论！评论不超过100字...

发表评论

客服电话
400-150-9800

电子邮箱
service@misuland.com

联系地址
上海市静安区，江场三路，181号11楼