一、热点问题

1.1 热点问题原因分析

1.1.1 什么是热点问题

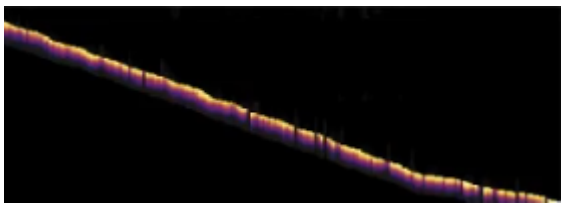
- 热点就是数据的一小部分承载了远超于其余部分的负载，负载可以是不同维度，例如：QPS、数据流量、请求带来的CPU消耗等
- 对于分布式数据库来说，最理想的情况是所有的请求均匀分布在所有的数据上
- 热点也有可能是tidb底层机制导致局部数据过热。出现热点不一定会有热点问题，TiDB有一定能力识别出热点region并通过调度解决，当出现TiDB无法识别热点问题和通过调度无法解决时才会存在热点问题，单个热点会导致整个集群性能瓶颈。
- 如果不能消除热点，只是增加tikv节点是不能提升性能。

1.1.2 顺序写

- 大多数热点问题都是由于顺序写导致的

```
1. 和业务模式有关不断地在表中追加数据，或者所写入的表包含索引且索引的值不断递增或递减
2. auto_increment问题，当表没有主键或没有int类型的主键时，TiDB会隐式创建一个的主键列Row_id（自带increment属性）
```

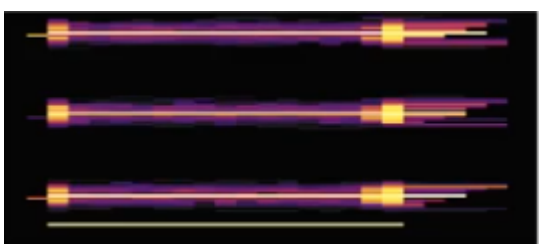
- 顺序写入可以观察到少量TiKV实例IO负载高、CPU压力大等特征



- 往新创建的表插入数据也会产生写热点现象，此现象因为新创建的表只有一个Region，开始的一段时间出现负载集中产生热点现象

1.1.3 热点小表

- 指的是特别少的一段数据被频繁访问，由于数据量较小往往被分配到单一Region内，无法通过调度解决，
- 通常产生读热点大量的范围扫描和Coprocessor携带了大量的下推表达式计算，需要关注Coprocessor metrics
- 写入热点主要为小范围大量更新操作



1.1.4 Region分布偏斜

- TiDB在做负载均衡调度时考虑的是集群里所有的region，具体到某个特定的表他锁包含的region不一定均匀分布
- region集中分布的TiKV实例更容易出现Region偏斜
- 中等规模的表容易由于region偏斜产生热点问题
- 读写都会出现此类问题，例如，高并发的范围扫描或一定范围内的随机插入

1.2 如何防止热点问题

1.2.1 分区表

- 虽然分区表并不是解决热点引入的，但设置Hash partition可以很好的避免顺序写产生的热点问题
- 分区表解决热点问题的原理

```
1. 在底层数据分布上不同的partition就是不同的表，因此单点顺序写通过Hash自然变成了多点写入
2. 刚创建表的集中写入问题也能得到改善，因为表创建完之后就包含了多个region，能更快的通过调度完成均衡
```

1.2.2 SHARD_ROW_ID_BITS

- 根据顺序写热点问题引入的，设置之后ROW ID生成将一定程度上随机而不是严格的递增
- 使用简单既能创建表时设置，也能通过alter语句进行调整

```
CREATE TABLE t (c int) SHARD_ROW_ID_BITS = 4;
ALTER TABLE t SHARD_ROW_ID_BITS = 4;
```

1.2.3 Pre-split Region

- 预分裂也是解决写热点问题，与其他方式相比效果最好
- 根据数据分布情况提前把表的Region分裂成合适的数量，并能发起调度是这些空Region在集群中均匀分布
- 缺点是需要做一些而外的操作，还需要对数据的数值范围分布有一定的了解

```
SPLIT TABLE table_name [INDEX index_name] BETWEEN (lower_value) AND (upper_value) REGIONS region_num;
SPLIT TABLE table_name [INDEX index_name] BY (value_list) [,(value_list)] ...
```

1.3 如何处理热点问题

1.3.1 Follower Read

- Follower Read 针对小表读热点问题
- 开启后查询不仅能发往leader也能发往follower，从而缓解了Region leader所在的TiKV实例的压力

```
SET @@tidb_replica_read = 'leader-and-follower';
```

1.3.2 Scatter Range

- Scatter Range 主要处理Region分布偏斜的问题
- 针对某个表开启后，TiDB会在后台产生调度，使这个表在所有TiKV实例均匀分布
- 调度完成后需要关停

```
开启：
curl -X POST http://{tidb_host}:10080/tables/{db}/{table}/scatter
关闭：
curl -X POST http://{tidb_host}:10080/tables/{db}/{table}/stop-scatter
```

1.3.3 Shuffle Leader/Region

- Shuffle Leader/Region 解决Region分布偏斜的非常规手段，只有其他方法都无法解决时才选择这种方式
- 通过PD提供的这两个调度器随机的移动Region的leader或副本的位置
- 因为随机的机制导致工作效率较低，但运行足够长的时间，Region分布偏斜问题基本上都能解决

```
创建调度器：
pd-ctl scheduler add shuffle-region-scheduler
pd-ctl scheduler add shuffle-leader-scheduler
移除调度器：
pd-ctl scheduler remove shuffle-region-scheduler
pd-ctl scheduler remove shuffle-leader-scheduler
```