链滴

ClickHouse 的分布式引擎

作者：flowaters

原文链接：https://ld246.com/article/1513595607561

来源网站：链滴

分布式引擎

分布式引擎可以将集群操作透明成单机操作。

- 集群不支持水平扩展，必须修改配置文件。
- 分布式表中的子表必须是最终表，不能仍然是分布式表。
- 配置可以动态生效，服务不需要重启。

开启

通过修改配置文件，示例如下

```
<remote_servers>
   <logs>
      <shard>
         <!-- Optional. Shard weight when writing data. By default, 1. -->
         <weight>1</weight>
         <!-- Optional. Whether to write data to just one of the replicas. By default, false - write
data to all of the replicas. -->
         <internal_replication>false</internal_replication>
         <replica>
            <host>example01-01-1</host>
            <port>9000</port>
         </replica>
         <replica>
            <host>example01-01-2</host>
            <port>9000</port>
         </replica>
      </shard>
      <shard>
         <weight>2</weight>
         <internal_replication>false</internal_replication>
         <replica>
            <host>example01-02-1</host>
            <port>9000</port>
         </replica>
         <replica>
            <host>example01-02-2</host>
            <port>9000</port>
         </replica>
      </shard>
   </logs>
</remote_servers>
```

账号

可以设置用户名密码

```
         <replica>
            <host>example01-02-2</host>
            <port>9000</port>
```

```
            <user>default</user>
            <password></password>
        </replica>
```

副本

随机读做任意一个副本，失败则读下一个副本，可以通过 load_balancing 来配置读取策略

分片

可以为不同的分片配置不同的副本数，当然也可以都配置相同的副本数。

分片表达式：即一个hash函数，从一条日志得到一个hash值。可以有：

- 随机hash: rand()
- 按用户hash: UserID，可以简化用户级别的IN和JOIN操作
- 按用户来均匀hash: intHash64(UserID)，防止UserID不均匀

使用 Sharding schema的场景

- 数据量超级大
- 使用指定key的IN or JOIN时，如果指定key正好是sharding key，则可以使用Local IN or JOIN，必用GLOBAL IN or GLOBAL JOIN。

超大数据量场景

1. 两层sharding方式

- 将cluster分成不同的layers(可以类比为index), 每个layers包含不同的shards.
- 每一个client的数据对应于一个layer
- shards按需加入layer
- 分布式表建立在layer上
- 每一个分布式表只有一个shard(?)

In order for the small queries to not affect the entire cluster, it makes sense to locate data for single client on a single shard. Alternatively, as we’ ve done in Yandex.Metrica, you can set u bi-level sharding: divide the entire cluster into “layers”, where a layer may consist of multip e shards. Data for a single client is located on a single layer, but shards can be added to a laye as necessary, and data is randomly distributed within them. Distributed tables are created for each layer, and a single shared distributed table is created for global queries.

写数据的方式

有两种写数据的方式：

1. 将数据写入最终表：这种方式最灵活。
2. 将数据写入分布式表：需要指定sharding key集合。

数据写入shard时，可以设置每个shard的权重。

内部复制 internal_replication

1. true选项

描述：数据只写入一个replica
场景：最终表为 replicated tables，表自身带复制。

2. false选项

描述：数据写入所有的replica
场景：分布式引擎来控制复制

比较：false时，时间长了，数据会有少量差异，无法保证数据的最终一致性。

异步写入

数据是异步写入集群，落在磁盘的文件在目录 /var/lib/clickhouse/data/database/table/ 下.

在硬件问题引起重启时，如果数据有损坏，会转移到 broken 目录下，不再使用.

感想

表复制

数据写入一个节点，通过ZK通知到其它节点，来从写入节点同步写入的数据。所以可以配置无限个备
。

参考

1. distributed