cassandra 3.x官方文档(4)---分区器 - 原谅我一生放荡不羁爱自由 - 博客频道

分类：
NoSql（28）

目录(?)[+]

写在前面

cassandra3.x官方文档的非官方翻译。翻译内容水平全依赖本人英文水平和对cassandra的理解。所以强烈建议阅读英文版cassandra 3.x 官方文档。此文档一半是翻译，一半是个人对cassandra的认知。尽量将我的理解通过引用的方式标注，以示区别。另外文档翻译是项长期并有挑战的工作，如果你愿意加入cassandra git book,可以发信给我。当然你也可以加入我们的QQ群,104822562。一起学习探讨cassandra.

一个分区器决定了数据将会在集群中的节点中如何分布(包括副本)。从根本上说，一个分区器就是一个方法，根据hash从partition key产生一个token，代表一行数据。每一行数据会通过这个hash值分布在集群中。

Murmur3Partitioner 和RandomPartitioner 都是使用token将数据均匀分配到每个节点。通过ring或者其他的分组方式如keyspace,将来自所有table的数据均匀的分配。这是事实即使表使用不同的partition keys，比如usernames,或者timestamps.不仅如此，到集群的读和写请求也能均匀的分布。负载均衡被简化了因为每一部分的hash值范围都平均收到相同数量的行。更多详细的信息，请看一致性hash

这两个分区器的主要不同点在于如何去产生token hash值。RandomPartitioner 使用加密hash所以相比较Murmur3Partitioner需要花费更多的时间去产生hash值。Cassandra实际上并不需要一个加密的hash,因此使用Murmur3Partitioner能够有3-5倍的性能提升。

Cassandra提供一下partitioners,可以在cassandra.yaml文件中配置。

- Murmur3Partitioner(默认)：基于MurmurHash hash值将数据均匀的分布在集群
- RandomPartitioner：基于MD5 hash值将数据均匀的分布在集群中
- ByteOrderedPartitioner：通过键的字节来保持数据词汇的有序分布

Murmur3是Cassandra1.2+ 默认的分区策略。这也是大多数情况新的集群的正确的选择。然而，分区器并不是可适应的。数据通过某个分区器分区后，不是很容易就能转换为另一个分区器的。

Note
如果使用了虚拟节点，你不需要去计算tokens.如果不使用虚拟节点，必须要计算tokens，然后分配给cassandra.yaml文件中[initial_token](http://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/configCassandra_yaml.html#configCassandra_yaml__initial_token)参数。可以参考Generating tokens,然后使用你用的分区器

对应的方法来产生token。

Murmur3Partitioner

Murmur3Partitioner 是默认的分区器，提供了更快的hashing.相比较其他的分区器，极大的提高了性能。Murmur3Partitioner 可以在虚拟节点情况下使用，如果你不使用虚拟节点，你必须要计算tokens。像Generating tokens中描述的一样。

在新集群中使用Murmur3Paritioner;你不能在一个现有的集群中更换分区器，去使用一个不同的分区方式。Murmur3Partitioner 使用MurmurHash方法，这个hashing方法为partition key创建一个64位的hash值。可能的范围值是-2^63 到(2^63)-1.

使用Murmur3Partitioner，可以在一个CQL 查询中使用token function 对结果分页

RandomPartitioner

RandomPartitioner 是Cassandra1.2之前版本的默认分区器，为了后续兼容性被包含进来了。RandomPartitioner可以和虚拟节点一起使用，然而，如果你不使用虚拟节点，你必须要计算tokens。像Generating tokens中描述的一样。RandomPartitioner 使用行key的MD5 hash值将数据均匀的分布在集群的节点上，hash 值的范围值是(2^127)-1

使用Murmur3Partitioner，可以在一个CQL 查询中使用token function 对结果分页

ByteOrderedPartitioner

Cassandra提供ByteOrderedPartitioner为的是有序分区。为了后续兼容性被包含进来了。通过键的字节来对行词汇进行排序。可以看partition key数据的实际值来计算token，采用16进制表示key的首字母。例如，如果你想让行按字母顺序排列，你可以指定一个tokenA使用16进制的41表示。

使用有序分区器允许通过主键有序扫描。这意味着你可以扫描行就好像在索引中移动游标。例如，如果你的程序使用user names作为行键值，你可以扫描用户(姓名在Jake和Joe之间）。这对于Random分区器，这种方式的查询是做不到的，因为键值按照MD5的顺序存储，而不是顺序的。

尽管对于有序分区器来说扫描行这种能力听起来是一个分棒的特性，但通过table indexes也能实现同样的功能。

因为以下原因，不建议使用有序分区器:

负载平衡难

需要更多的管理开销去实现集群的负载平衡。一个顺序的分区器需要管理员根据行键值的可能的分布情况去手动计算 partition ranges。在实践中，一旦数据已经加载后，需要经常性的改变节点的token去适应实际数据的分布。

顺序写导致热点

如果你的程序在某一段时间内的写入或者更新包含很多按顺序排列的行的时候，它们不会均匀的分布在集群上，会分布同一节点上。当系统处理和时间相关的数据的时候这是一个常见的问题。

多表时负载不平衡

如果你的应用程序用到多个表，这些表有不同的行键值和不同的数据分布。在同一个集群中，对于一张表一个有序的分区对于另一个表可能会导致热点和不均匀分布。

顶

0