HBase 数据压缩介绍与实战

为了提高 HBase 存储的利用率，很多 HBase 使用者会对 HBase 表中的数据进行压缩。目前 HBase 可以支持的压缩方式有 GZ（GZIP）、LZO、LZ4 以及 Snappy。它们之间的区别如下：

- GZ：用于冷数据压缩，与 Snappy 和 LZO 相比，GZIP 的压缩率更高，但是更消耗 CPU，解压/压缩速度更慢。
- Snappy 和 LZO：用于热数据压缩，占用 CPU 少，解压/压缩速度比 GZ 快，但是压缩率不如 GZ 高。
- Snappy 与 LZO 相比，Snappy 整体性能优于 LZO，Snappy 压缩率比 LZO 更低，但是解压/压缩速度更快。
- LZ4 与 LZO 相比，LZ4 的压缩率和 LZO 的压缩率相差不多，但是LZ4的解压/压缩速度更快。

各种压缩各有不同的特点，我们需要根据业务需求（解压和压缩速率、压缩率等）选择不同的压缩格式。多数情况下，选择Snppy或LZ0是比较好的选择，因为它们的压缩开销底，能节省空间。这里介绍一下 HBase 中使用 Snappy 的方法，其他的压缩设置方法和这个类似。

如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

创建表时指定压缩格式

在创建 HBase 表的时候我们可以指定数据的压缩格式，如下；

```
hbase(main):010:0> create 'iteblog',{NAME=>'f1'}, {NAME=>'f2',COMPRESSION=>'Snappy'}
```

```
Created table iteblog
Took 1.2539 seconds
=> Hbase::Table - iteblog
hbase(main):011:0> describe 'iteblog'
Table iteblog is ENABLED
iteblog
COLUMN FAMILIES DESCRIPTION
{NAME => 'f1', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR
 => 'false', KEEP_DELETED_CELLS => 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_E
NCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLO
OMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY =>'false', CACHE_BLOOM
S_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLO
CKCACHE => 'true', BLOCKSIZE => '65536'}
{NAME => 'f2', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR
 => 'false', KEEP_DELETED_CELLS => 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_E
NCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLO
OMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY =>'false', CACHE_BLOOM
S_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'SNAPPY', BL
OCKCACHE => 'true', BLOCKSIZE => '65536'}
2 row(s)
Took 0.0522 seconds
```

上面例子的表 iteblog 有两个列族，我们选择对 f2 列族进行 Snappy 压缩， f1 列族数据不压缩。

对已有的表设置压缩

当然，如果我们表已经创建了，同样也可以对其进行压缩，方式如下：

```
hbase(main):001:0>  alter 'iteblog', NAME => 'f', COMPRESSION => 'snappy'
Updating all regions with the new schema...
27/27 regions updated.
Done.
Took 9.5146 seconds

hbase(main):003:0> describe 'iteblog'
Table iteblog is ENABLED
iteblog
COLUMN FAMILIES DESCRIPTION
{NAME => 'f', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR
=> 'false', KEEP_DELETED_CELLS => 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_E
NCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLO
OMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOO
```

```
MS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'SNAPPY',
BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
1 row(s)
Took 0.0782 seconds
```

设置完之后，其实数据并没有被压缩，我们需要对当前表执行 major_compact 命令手动进行压缩：

```
hbase(main):002:0> major_compact 'iteblog'
Took 1.2255 seconds
```

这样，iteblog 表的数据就可以被压缩了。