# Chunkserver渐进式合并方案讨论

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **编号** | **文档版本** | **修订章节** | **修订原因** | **修订日期** | **修订人** |
| **1** | 1.0 |  | 创建 | 2011.3.15 | 华庭 |
| **2** |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# 渐进式合并中跨版本合并问题

下面以3个版本的sstable合并来说明这一问题，假设一个sstable的range为[1,100]，版本为v1，存在两个转储的memtable，版本分别为v2和v3。由于每个sstable共有3个备份，分别存储在不同的cs上，每个cs合并速度不一致，所以就有可能有些cs已经将该sstable合并到v2，在有些cs中，该sstable处于v1，也有可能在有些cs中，该sstable为v3。该sstable在多个cs中的可能合并路径如下表所示：

|  |  |  |
| --- | --- | --- |
| V1 -> V2 | V2 -> V3 | V1 -> V3 |
| [1,100]不分裂 | [1,100]不分裂 | [1,100]不分裂 |
| [1,100]不分裂 | [1,100]分裂为[1,50]，(50,100] | 不分裂仍为[1,50]，(50,100] |
| [1,100]分裂为[1,50]，(50,100] | 不分裂仍为[1,50]，(50,100] | 不分裂仍为[1,50]，(50,100] |
| [1,100]分裂为[1,50]，(50,100] | [1,50]分裂为[1,25],(25,50]  (50,100]不分裂 | [1,100]分裂为[1,25],(25,50],  (50,100] |
| [1,50]不分裂  (50,100)分裂为(50,75],(75,100] | [1,100]分裂为[1 ,50], (50,75],(75,100] |
| [1,50]分裂为[1,25],(25,50]  (50,100)分裂为(50,75],(75,100] | [1,100]分裂为[1,25],(25,50],  (50,75],(75,100] |

当前cs的每日合并实现中，由V1合并到V3必须先由V1 合并到 V2，然后由V2 合并到 V3，不能实现由V1直接合并到V3，这种比较简单的算法保证了多个cs上的同一个sstable从V1合并到V3后，V3版本的sstable range保持一致。但在渐进式合并中，存在V1直接合并到V3的情况，也存在V1合并到V2，再由V2合并到V3的情况，无论根据何种路径到达V3，V3版本的sstable的range都必须是唯一确定的，多个cs中的相应的sstable必须保持一致性。由于需要支持column group，必须在sstable与memtable合并之前先确定sstable分裂点的rowkey，这也导致现有实现的sstable分裂算法不能满足要求。

为了保证从同一个起点版本V1合并到同一个目标版本V3过程中，合并的结果与合并的路径无关，我们考虑了如下的五种方案，每种方案都有优缺点，每种方案的切入点都不一样，主要有如下几个切入点：

1. 限制合并的路径，使其只能按照V1 -> V2，V2 -> V3的顺序将V1合并到V3。
2. 全局仲裁V1 -> V2，V2 -> V3，V1->V3过程中的range分裂和合并。
3. 根据V3版本最终数据相同的假设，每个cs缓存V3版本sstable的全量数据，按一定的规则分裂和合并sstable。
4. 根据同一range 在V3版本的数据相同，并假设抽样数据能够反映合并后sstable的可能大小变化规律，根据抽样的range合并到V3版本后的数据大小与V1版本中相同range大小的关系，决定将整个range分裂成几个子range或是合并多个range为一个range。
5. 保持tablet的range不变，每个cs对tablet中的sstable的分裂合并完全自主，不管别的cs中同一个tablet中sstable的分裂合并，由于同一tablet在不同cs上的sstable二进制可能是不一样的，迁移以tablet为单位。

# 合并方案

## sstable仅与下一个版本的memtable合并

本方案与当前的每日合并方案类似，强制限制每次只能将sstable合并到下一个版本，sstable不能跨版本合并，sstable的分裂方式需要修改，对column group的支持需要保证在合并sstable之前计算出分裂点的rowkey，而不能当前已写入磁盘的sstable文件大小来决定是否分裂。这种方案最大限度利用现有代码，需改动的地方少，实现最为简单。这种方案中每个tablet中只有一个sstable，一旦sstable分裂，tablet也同时分裂。如果需要将多个sstable合并，只需将多个tablet合并为一个tablet，一个tablet中包含多个sstable文件，在下一次合并时，可以将tablet中的所有sstable合并为一个sstable文件。当前cs的实现已经对一个tablet包含多个sstable情况做了部分支持。

由于本方案的简单性，我们将在4月份的cs版本中采用这种方案实现渐进式合并，对评价历史库来说，性能已经足够，根据评价历史库的数据量，可能ups每天需要转储2到3次，而评价历史库的查询量比较小，cs每天完成3次左右的“每日合并”不会有问题，更简单的办法是将历史库的多个转储memtable通过一次“每日合并”完成，不需要用到渐进式合并方案。但是，这种方案也有如下的限制：

1. 只能合并到下一个版本，不能跨版本合并，将消耗更多的cs和ups资源。
2. 对写多读少的系统效率太低。
3. 本质上是“每日合并”的水平扩展，由于数据量增大导致转储为多个memtable，将一次“每日合并”扩展为多次“每日合并”。当每天转储的memtable个数超过5个以后，这种方案太不经济，可操作性太低，没有可扩展性。

## rootserver仲裁tablet的合并与分裂

每个tablet中包含一个sstable，rootserver存储了所有tablet的range信息，对同一个tablet的多个版本的当前状态最清楚，如果由rootserver发起talbet range的合并和分裂最为直接。根据实现的不同可以分为两种方案：

### 并行化方案

并行化方案要求rootserver告诉cs每个tablet的分裂点和合并点，无论tablet的当前版本和合并后的最终版本，由rootserver的统一调度都能保证tablet和sstable的range在多个cs上都是一致的。

这种方案难点在如果找到一种好的启发式算法，让rootserver根据该算法分裂合并sstable，保证sstable大小大体均匀。Rootserver不但要管理tablet的信息，还需要管理每个tablet中的sstable信息，每个tablet和sstable都有多个版本，每当cs将一个sstable按照rootserver指定的方式合并完成以后，将相关的sstable信息（比如sstable大小，版本）上报给rootserver，rootserver根据当前tablet和sstable信息决定分裂合并点。大致流程如下：

1. cs向rootserver询问当前tablet版本的合并到目标版本过程中的分裂点和合并点的rowkey。
2. rootserver根据当前tablet和sstable的状态信息和以前该tablet的数据增长情况，计算出分裂点和合并点的rowkey，并返回给cs。
3. cs更加rootserver返回的分裂合并信息对tablet和sstable做相应的操作。

### 串行化方案

串行化方案允许cs按照自己的方式分裂合并sstable，但第一个上报的cs的结果为准，剩下的cs都必须按照第一个cs的合并结果进行分裂合并。为了避免cs做一些不必要的合并工作，rootserver需要调度cs，让cs对某个tablet先合并到某个版本，剩下的两个备份按照已经合并好的结果进行合并，这个方案串行化了tablet的合并。以第一节的跨版本合并为例，假设cs1上的sstable已近由V1合并到V2，并分裂成了两部分[1,50]，(50,100]，而cs2和cs3都为V1，当前转储的最新memtable版本为V3，如果rootserver决定让cs1先合并到V3，cs2合并完成后将合并结果报告给rootserver，然后rootserver将sstable的分裂合并点通知cs2和cs3，cs2和cs3按照cs1的结果进行分裂合并。如果rootserver决定让cs2先合并，然后cs1和cs3按照cs2的合并结果进行合并。

### 缺点

1. 无论是并行方案或是串行方案，都需要对rootserver做大量修改，当前人手不够。
2. Rootserver做的事情太多了，将成为系统的单点，一旦rootserver拥塞或是宕机，就不能合并了。
3. 算法复杂，可操作性低。

## 全量缓存整个tablet合并后的数据

每个tablet中有一个或多个sstable文件，无论tablet以何种合并路径到达目标版本，目标版本的最终数据应该是相同的，如果每个tablet的数据不大，全量缓存整个tablet合并后的数据，设定一个简单的规则，就可以保证多个cs对同一个tablet的分裂和合并是一致的。Ob系统初始化时，默认每个tablet只有一个sstable，当一次合并后，sstable可能分裂为多个sstable，但这些分裂出的sstable仍然属于同一个tablet，下一次合并时，将整个tablet合并后的数据缓存到内存中，然后重新按分裂规则生成sstable文件，这同时解决了sstable的分裂和合并问题。

如果tablet太大，需要对tablet进行分裂，tablet的分裂需要rootserver的参与，分裂步骤为：当前tablet的三个备份都版本为最新版，当tablet最后一个备份上报tablet信息到rootserver时。rootserver发起tablet分裂，三个备份确认tablet分裂，rootserver修改root table。

如果tablet太小，需要与相邻的tablet合并，tablet合并同样需要rootserver的参与，合并步骤为：rootserver查找root table，发现低于阈值的tablet，该tablet的所有备份都为最新版，并且该tablet的相邻tablet的所有备份也为最新版，如果需要发起迁移，保证这两个需要合并的tablet在相同的cs上，然后发起合并tablet，所有cs都确认tablet合并后，修改root table。

本方案的缺点也是很明显的，需要缓存整个tablet合并后的数据，需要相当多的内存，对特别大的tablet缓存所有数据几乎不可能，需要采用额外的处理方法特别处理。如果sstable的大小在64M左右，每个tablet中的sstable数据量少于4个，这种方案是有一定的优越性的，如果使用10个线程进行合并，使用小于2.5G的内存解决了跨版本的分裂合并问题，而且实现也比较简单，还是值得的。毕竟现在的“每日合并”的10个线程也有可能用到额外的1.3G内存用于缓存sstable合并后的数据。

## 启发式tablet合并与分裂

在2.3节方案基础上，为了节约合并中使用的内存，抽样tablet中固定范围的行数据，计算其合并后大小，将目标版本的tablet分裂或合并为几个range，让后按照该range对tablet进行合并。

仍然以第一节中的跨版本合并为例，本方案根据同一range 在V3版本的数据相同，并假设抽样数据能够反应合并后sstable的可能大小变化规律，根据抽样的range合并到V3版本后的数据大小与V1版本中相同range大小的关系，决定将整个range分裂成几个子range或是合并多个range为一个更大的range。假设tablet中只有一个sstable，range为[1,100]，抽样的范围为[1,5]，假设在V1版本中，range [1,5]所有行数据为压缩前的大小为10M，在V1版本合并到V2版本过程中，抽样range合并后未压缩的数据大小为20M，如果抽样相对准确的话，将sstable分裂成两个sstable就比较合理，于是将tablet分为两个range [1,50]，(50,100]，然后按照这两个range进行合并。当从V1版本合并到V3时，同样计算抽样range合并后未压缩的数据大小，假设为5M，那么合并后任然为1个sstable，range 为[1,100]。对于从V2版本合并到V3版本，同样计算抽样range合并后未压缩的数据大小，根据目标版本的数据的一致性，抽样range合并后未压缩的数据大小也应该为5M，在V2版本，tablet中有2个sstable，所以需要将这两个sstable合并为一个sstable，合并后的范围为[1,100]。从这个过程中可以看出，无论tablet以何种路径由V1版本合并到V3版本，最终tablet的范围为[1,100]，只有一个sstable，这正是我们需要的。

本方案中同样涉及到tablet的分裂和合并，实现方式与2.3节方法相同，请参考2.3节。这种方案比2.3节中描述的方案更节省内存，只需要缓存抽样range合并后的数据，2.3节的方案可以看成是本方案100%抽样的特殊情况，100%抽样也就意味着缓存了tablet合并后的所有数据。本方案实现也相对简单，不用对rootserver做太多修改，是暂时想到的方案中值得推荐的一种。但这种方案也有缺点：

1. 抽样不一定准确，根据抽样数据划分的sstable range不一定均匀。
2. 仍然需要额外的内存缓存抽样数据。
3. 迁移必须以tablet为单位，tablet中sstable的个数不确定。

## Tablet range保持不变，cs自主分裂合并

本方案cs完全自主对tablet中的一个或是多个sstable做分裂合并，不保证与其它cs上同一个tablet中的sstable数量或是二进制上是一致的，但要保证多个备份的tablet能读取到一致的数据，需要额外的数据一致性保证机制，不能使用当前校验sstable文件crc是否一致的方法。

这种方案还没有仔细想过，但缺点很明显，需要额外的数据校验算法，tablet的分裂和合并非常难解决。如果解决不了上诉的两个问题，该方案也没有可操作性。