Level Compaction in HBase

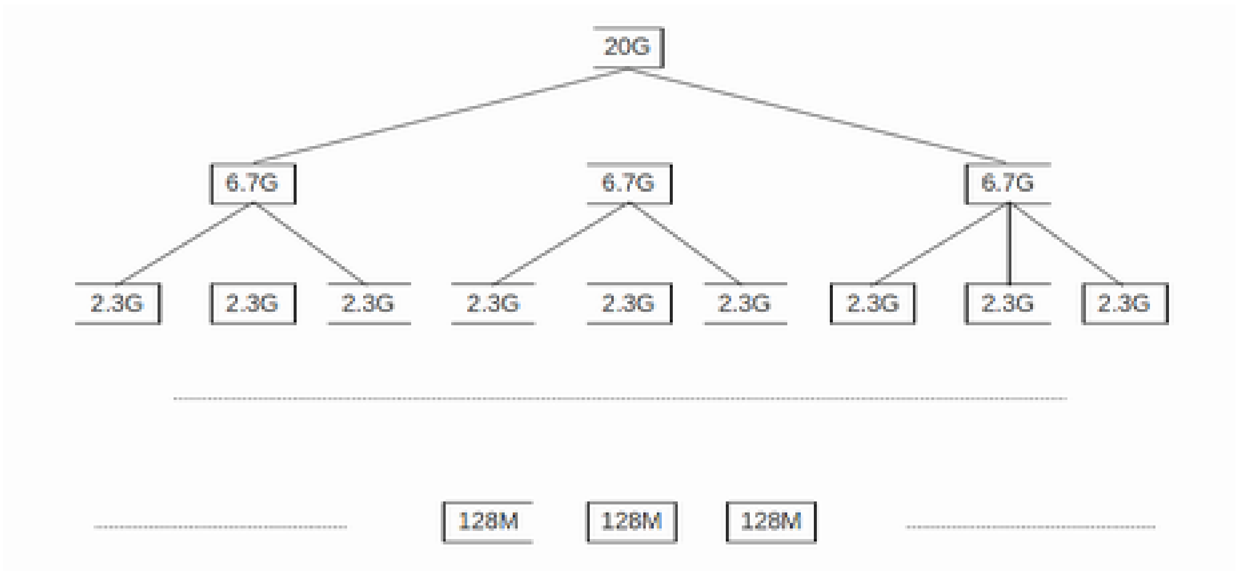
In this doc, we estimate the data io and compare level compaction and the normal compaction (a special one, not exactly the same as we have now). We found that there will be far less data io for normal compaction compared to level compaction.

With level compaction, each compaction will not have bursty huge data io. But we need many more number of compactions. That means, on average, level compaction may not be able to improve performance.

Data IO Estimation

HBase memstore is flushed into new small HFiles periodically while new data are inserted into a table, or existing data are updated/deleted. To reduce read latency, the number of HFiles should be tightly controlled. Therefore, periodically, these small HFiles are merged into bigger HFiles, which may be merged into even bigger HFiles until the region is split.

Currently, by default, the memstore flush size is 128 MB. So each newly created HFile is of size about 128MB assuming compression is not enabled. With constant size split policy, a region splits when one of its column families reaches 20 GB. For simplicity, let’s assume 3 files of the same size merges to a big file all the time:



With this assumption, before a region splits, there will be about:

- 80 compactions;
- 92.4 GB total data io;

- 1.15 GB average data io per compaction;
- smallest compaction reads 384 MB and writes 384 MB, total 768 MB data io;
- biggest compaction reads 20 GB and writes 20 GB, total 40 GB data io.

In most case, this kind of compaction strategy works very well. However, when the big compaction kicks in, it needs to do lots of data io which may take a while. To avoid such scenarios, we can adapt the level compaction strategy used by LevelDB.

With level compaction, each HFile belongs to a level. The newly created HFiles from memstore flush are always level 0. Level 0 files compact into level 1 files. Level 1 files compact into level 2 files. There are at most 10 level 1 files (configurable), 100 level 2 files and 1000 level 3 files. Each file is about the same size. To have at most 20 GB data, each file can be about 18.4 MB if there are 4 levels (including level 0). If level L is full, files in level L will be compacted into level L+1.

In each level (except level 0), the key range doesn't overlap. One level L file can overlap with at most 10 L+1 files. One level 0 file can overlap with all level 1 files, and other level 0 files.

Assume each level 0 file always overlap with 10 level 1 files, each level L (L>0) file always overlaps with 10 L+1 files. Also assume each compaction creates a new file. Before the region is split, there will be about:

- 1110 compactions (max number of files in level 1, 2 and 3);
- 406 MB average data io per compaction (read 11 files and write another 11 files, each file is about 18.4 MB);
- 440.1 GB total data io.

Assume each level L file overlaps with 5 L+1 files on average, the total data io would be about 240 GB, which is about 2.6 times that of previous compaction strategy. To have the same amount of data io as the normal compaction, the average overlap should be about 1.4 files, which could be a very aggressive assumption.

One thing we should mention is that the level 1 files could be promoted to level 2 if there is no level 2 files. However, if there are level 2 files, level 1 files may be created/compacted several times.

If we set the size of each file to 128 MB instead of 18.4 MB, the region should reach 20 GB if there are only 150 level 2 files. Assume each level 1 file overlaps with 5 level 2 files on average, before the region is split, there will be about:

- 160 compactions;
- 1.5 GB average data io per compaction;
- 240.0 GB total data io.

Summary

Based on some assumption and simplified estimation, which may not be very accurate, we may be able to claim, on average, level compaction needs more data io. Although it can improve the read heavy performance load, it can increase the data io load on HDFS, and the load on network bandwidth.