hbase很有价值的读写性能提升

NoSQL现在风生水起，hbase的使用也越来越广，但目前几乎所有的NoSQL产品在运维上都没法和DB相提并论，在这篇blog中来总结下我们在运维hbase时的一些问题以及解决的方法，也希望得到更多hbase同行们的建议，:)
在运维hbase时，目前我们最为关注的主要是三大方面的状况：
1. Cluster load；
2. 读写；
3. 磁盘空间。

1. Cluster load
集群的load状况直接反映了集群的健康程度，load状况的获取非常容易，直接部署ganglia即可得到，由于hbase以优秀的可伸缩性著称，因此多数情况下load超出接受范围时加机器是一个不错的解决方法，当然，这还和系统的设计和使用hbase的方式有关。
如有出现个别机器load比较高的现象，通常是由于集群使用的不均衡造成，需要进行一定的处理，这个放到读写部分再说吧。

2. 读写
读写方面的信息主要包括了读写的次数以及速度，这个通过ganglia看其实不怎么方便，最好还是自己改造下实现，读写次数反映了集群的使用程度，一般来说需要根据压力测试中形成的报告来设置一个读写次数的阈值，以保护系统的稳定。

读写速度方面主要是显示当前的读写速度状况（当然，也需要有历史的报表），如读写速度是在可接受范围，其实就可以不用过多的关心了，如读写速度不OK的话，则需要进行一定的处理。

如读速度不OK，则需要关注以下几点：
* 集群均衡吗？
集群是否均衡主要需要通过三个方面来判断：各region server的region数是否均衡、table的region是否均衡分布还有就是读请求是否均衡分布。

通常各region server的region数是均衡的，这个是目前hbase master的balance策略来保证的，顶多就是有短暂时间的不均衡现象。
table 的region分布则不一定是均衡的，对于有多个table的情况，完全有可能出现某张表的一堆的region在同一台上的现象，对于这种情况，一种方法 是修改master的balance策略，让其在balance时考虑table的region的balance，我们目前采用的是另外一种方法，提供了 一个界面来手工balance table的region，如果是由于table的region不均衡导致了读速度的不OK，可以采用这种办法解决下。

读 请求是否均衡分布一方面取决于table的region的分布状况，另一方面则取决于应用的使用方法，如table的region分布均衡，读请求仍然不 均衡分布的话，说明应用的请求有热点的状况，如这种状况造成了读速度的不OK，可以手工将region进行拆分，并分配到不同的region server上，这是hbase很简单的一种应对热点的解决方法。

* cache的命中率如何？
cache的命中率目前通过ganglia以及region server的log都不是很好看，因此我们也进行了改造，更直白的显示cache的命中率的变化状况。

如 cache的命中率不高，首先需要看下目前系统用于做LRUBlockcache的大小是不是比较小，这主要靠调整region server启动的-Xmx以及hfile.block.cache.size参数来控制，可通过修改hbase-env.sh，增加export HBASE_REGIONSERVER_OPTS=”-Xmx16g”来单独的调整region server的heap size，如LRUBlockCache已足够大，cache命中率仍然不高的话，则多数情况是由于随机读无热点造成的，这种情况如果要提升cache命 中率的话，就只

能靠加机器了。
在cache的使用率上要关注应用对数据的读取方式，经常整行数据读取的适合设计在同一cf里，不经常整行读取的适合设计在多cf里。

* bloomfilter打开了吗？
默 认情况下创建的table是不打开bloomfilter的（可通过describe table来确认，如看到BLOOMFILTER => ‘NONE’ 则表示未打开），对于随机读而言这个影响还是比较明显的，由于bloomfilter无法在之后动态打开，因此创建表时最好就处理好，方法类 似如此：create ‘t1’, { NAME => ‘f1’, BLOOMFILTER => ‘ROWCOL’ }。

* Compact
在某些特殊的应用场景下，为了保证写速度的平稳，可能会考虑不进行Compact，不进行compact会导致读取数据时需要扫描大量的文件，因此compact还是有必要做的。

* 应用的使用方式
应用在读取数据时是随机读、有热点的随机读还是连续读，这个对读速度也有很大的影响，这个需要结合业务进行分析，总的来说，hbase在随机读上效率还是很一般的，这和它的存储结构有一定关系。
另外，应用在读取时如每次都是读取一行的所有数据的话，schema设计的时候在同一个cf下就比较合适。

如写速度不OK，则需要关注以下几点：
* 集群均衡吗？
除 了和读一样的集群均衡性问题外，还有一个问题是rowKey的设计问题，因为hbase是按rowKey连续存储的，因此如应用写入数据时rowKey是 连续的，那么就会造成写的压力会集中在单台region server上，因此应用在设计rowKey时，要尽可能的保证写入是分散的，当然，这可能会对有连续读需求的场景产生一定的冲击。

* 日志中是否出现过以下信息？
** Flush thread woke up with memory above low water.
日 志中出现这个信息说明有部分写出现过阻塞等待的现象，造成这个现象的原因是各个region的memstore使用的大小加起来超过了总的阈值，于是阻塞 并开始找一个region进行flush，这个过程会需要消耗掉一些时间，通常来说造成这个的原因是单台region server上region数太多了，因此其实单台region server上最好不要放置过多的region，一种调节方法是调大split的fileSize，这样可以适当的减少region数，但需要关注调整后 读性能的变化。

** delaying flush up to
当日志中出现这个信息时，可能会造成出现下面的现象，从而产生影响，这个通常是store file太多造成的，通常可以调大点store file个数的阈值。

** Blocking updates for
当 日志中出现这个信息时，表示写动作已被阻塞，造成这个现象的原因是memstore中使用的大小已超过其阈值的2倍，通常是由于上面的delaying flush up to造成的，或者是region数太多造成的，或者是太多hlog造成的，这种情况下会造成很大的影响，如内存够用的话，可以调大阈值，如其他原因则需要 对症下药。

* split造成的？
split会造成读写短暂的失败，如写的数据比较大的话，可能会有频繁的split出现，对于这种情况主要需要依靠调大split的filesize（hbase.hregion.max.filesize）来控制。

3. 磁盘空间
磁 盘空间可直接通过hdfs的管理界面查看，磁盘空间如占用比较多的话，可以关注下表的压缩是否开启（describe表后，COMPRESSION => ‘NONE’ 表示未开启），默认是不开启的，在创建表时可create ‘t1’,{NAME => “cf1”,COMPRESSION => “LZO”}，如为已经创建的表，则需要先disable、alter、enable后再执行下major_compact，在我们的几个案例中大概能压 缩到原大小的20%–30%，还是很可观的。

如压缩已开启，占用仍然比较多的话，基本就只能增加机器或升级硬盘了，由于hbase需要对每列重复存储rowkey，因此会有一定的空间浪费。