Tier Based Compaction Setting parameters

All of the compaction parameters should be located in the file hbase-compactions.xml

The set of all parameters can be different for different column families too. For example, It is possible to use the default algorithm on one column family, the tier based algorithm with one set of parameters for another, and a different set of parameters on a third one. The general template for each parameter is of the form

"hbase.hstore.compaction.**$schema**.**$attribute**"

The schema can be specified in the form "tbl.$TableName.cf.$FamilyName" or as the string "default". If the settings for some column family is not specified individually, the default settings are used in that case.

There are two kinds of parameters for the algorithm, the tier specific parameters, and the tier independent parameters. Tier independent parameters are set for the overall algorithm. On the other hand, tier specific parameters can be different for each tier and are used while executing the subroutine for a particular tier only.

For a tier independent parameter, "$schema" is just the parameter name. For a tier specific parameter, "$schema" is of the form "tier.$num.$parameterName"

A tier specific parameter can be set without specifying the tier. Then that value will be used as a default value for any tier for which that parameter has not been set explicitly.

For example, NumCompactionTiers, MinCompactSize are tier independent parameters. It can be set in the following way"

"hbase.hstore.compaction.default.MinCompactSize"
"hbase.hstore.compaction.tbl.table1.cf.family1.MinCompactSize"

If not set for family2, the default value is used for that case.

CompactionRatio is a tier specific parameter. We can set parameters by using all of the following keys:

"hbase.hstore.compaction.default.CompactionRatio"
"hbase.hstore.compaction.tbl.table1.cf.family1.CompactionRatio"
"hbase.hstore.compaction.tbl.table1.cf.family1.tier.2.CompactionRatio"

In this case, the tier 2 for the store corresponding to tbl.table1.cf.family1 will use the third parameter, for all other tiers in that store, the second parameter will be used. If there are other column-families for which no parameter has been set, the first value will be used for them.

Tier independent parameters

Parameter	Description	Default	Comments/Caution
CompactionPolicy	The policy to be used, default or tier-based	DefaultCompactionPolicy	Must be a valid class name
MaxCompactSize	Upper bound on the file size in minor compaction, in bytes	INF	Set high enough value
MinCompactSize	Lower bound below which every file is compacted, in bytes	0	Set small enough
ShouldExcludeBulk	Whether bulk load files should be excluded from minor compactions	default*	Make sure bulk load files are the oldest ones, for now
ShouldDeleteExpired	Whether TTL-expired files should be first deleted by compaction	default	Set as true
ThrottlePoint	The threshold size for assigning compactionSelections into the small/large queue, in bytes	default	Too large or too small will starve one of the queues
MajorCompactionPeriod	The interval at which major compactions are selected periodically, in milliseconds	default	keep long enough
MajorCompactionJitter	The fractional deviation from MajorCompactionPeriod which is randomly effected to avoid having major compactions in many stores at the same time.	default	default is good
NumCompactionTiers	The number of tiers to assign storeFiles in	1	Careful to set this correct
IsRecentFirstOrder	Whether the tiers will	true	Set as false if

	be tried for eligible compactions from newest to oldest		mixing tiers. Beware of starvation of small compactions.

Tier specific parameters

Parameter	Description	Default	Comments/Caution
MaxAgeInDisk	The maximum age of a file which can belong to this tier, in milliseconds. tier[i] contains all files with age between (tier[i-1].maxAgeInDisk, tier[i].maxAgeInDisk]	INF	Set this carefully (No default value). should be increasing from lowest to highest tier.
MaxSize	The maximum size of a file which can belong to this tier, in bytes. tier[i] contains all files with size between (tier[i-1].maxSize, tier[i].maxSize]	INF	Usually use only one of MaxAgeInDisk and MaxSizecriteria for tier assignment. However it is possible to use both.
CompactionRatio	The parameter used for ratio test in this tier	default	The most important parameter for each tier. Set as zero to block all compactions for this tier. Set higher for more aggressive compactions.
MinFilesToCompact	Minimum number of files to be in a selection in this tier	default	An important parameter. Should be at least 2.
MaxFilesToCompact	Maximum number of files to be in a selection in this tier	default	Setting this small will result in more IOPs in long term, but finish compactions quicker.
EndInclusionTier	This is a special feature which allows compaction selection	this.tierIndex	Be careful when setting this parameter. Following condition must be

	across more than one tier. Supposetier[i].endInTier = j, tier[i] contains files f[u],f[u+1,...,]f[v], and tier[v]contains files upto f[w]. Then all pairs [start, end) = [i,w) will be tried for i = u,u+1,...,v-1 using the parameter-set for this tier.		true:i>=tier[i].endInclusionTier>= tier[i-1].endInclusionTier>=0 for all i>0 (The second inequality is important for avoiding not-in-order compactions).

* default indicates the parameter from the default compaction algorithm. Even if ShouldExcludeBulk is not set for the tier based compaction algorithm, if tier based algorithm is turned on, it will use the parameter from the default algorithm.