**SIZE TIERED COMPACTION STRATEGY :**

The traditional suggested compaction technique is the SizeTieredCompactionStrategy (STCS), which is advised for workloads that involve a lot of writing. If no other compaction strategy is specified, this one is used by default.

When Cassandra has amassed a predetermined number (by default, 4) of SSTables with comparable sizes, STCS starts the compaction process. SSTables are combined by STCS into a single, bigger SSTable. STCS aggregates these bigger SSTables to create even larger SSTables. There are several SSTables of different sizes present at any given time.

Although STCS effectively reduces the size of a write-intensive task, it slows down reads since the merge-by-size procedure does not organize data according to rows. Because of this, there's a greater chance that many SSTables will have different copies of the same row. Furthermore, because SSTable size acts as a compaction trigger for STCS, deleted data is not removed in a predictable manner. SSTables might not expand swiftly enough to routinely combine and remove outdated data, though.

The majority of STCS compactions are small compactions that combine several SSTables into one. On the other hand, two SSTables—one for repaired data and one for unrepaired data—will exist for each data directory during a significant compaction carried out with STCS. The amount of disk space required for the new and old SSTables concurrently during STCS compaction can exceed a normal amount of disk space on a node as the largest SSTables get bigger.

This phenomena is referred to as space amplification, expanding SSTable problem, and outgrowing compaction capability of a cluster. For STCS, major compactions are not advised.

To decide which SSTables to merge, STCS computes the average size of the SSTables. We refer to this procedure as bucketing. Based on that average size, the bucket into which an SSTable will be grouped is determined using the following options. SSTables are grouped in the bucketing process according to how much their sizes deviate from the average—either by 50% or 150%. An alternative approach to express this calculation is that SSTables with sizes between [average-size × bucket\_low] and [average-size × bucket\_high] are grouped by the bucketing procedure.

**Leveled Compaction Strategy (LCS)**

|  |  |
| --- | --- |
|  |  |

While UCS is currently the best option for all workloads, the Leveled Compaction Strategy (LCS) is advised for workloads that include a lot of reading. It offers reasonable writing operations and mitigates some of the STCS read operation problems. Each level in this strategy's hierarchy comprises a collection of SSTables. SSTables are written in the first level (L0), where they are not certain to be non-overlapping, when data in memtables is flushed. These first SSTables are combined with larger SSTables in level L1 using LCS compaction. By default, each level is ten times larger than the one before it. An SSTable is assured not to overlap with other SSTables in the same level after it is written to L1 or higher. All overlapping SSTables are combined into a single new SSTable in the following level to achieve compaction. Since most L0 SSTables cover the whole range of partitions, we virtually always need to include all L1 SSTables for L0 → L1 compactions. By creating partitions to meet a predetermined SSTable size, LCS compacts SSTables as it moves from one level to the next. Compaction will also occur when a level reaches its size limit because each level has a predetermined size. Compaction can start on the following level when new SSTables are created there, and so on, until all levels have been compacted according to the defaults.

If an excessive number of SSTable reads are made at the L0 level, a failsafe is in place. If there are more than 32 SSTables in L0, an STCS compaction will occur. SSTables from L0 are swiftly combined into L1, where they will be compacted into non-overlapping SSTables, thanks to this compaction.

With just about 10% of the disk required for execution, LCS is less disk-hungry than STCS, but it requires more IO and CPU. The amount of compaction is appropriate for continuous small compactions in a workload with a lot of reading. However, because it will take a lot of CPU and disk IO, it is not a viable option for workloads that require a lot of writing. It is not advised to use major compactions for LCS.

**STRAVED SS TABLES :**

A suboptimal leveling could leave LCS with starving sstables. Because SSTables in lower levels are not merged and compacted, high level SSTables may become stranded instead of compacted. For instance, this might prevent tombstones from being dropped by lower tiers. These hungry SSTables will be included in other compactions if they are not resolved in a predetermined number of compaction rounds. Usually, when a user reduces the sstable\_size parameter, this happens.

**BOOT STRAPPING :**

SSTables are streamed from other nodes during startup. The new node will have a large number of SSTables in L0 since many of them will be streaming from a remote note in addition to being flushed from the new writes to memtables. Until the bootstrapping is finished, only STCS in L0 is run in order to prevent a clash between the flushing and streaming SSTables.