**STRATEGIES OF COMPACTION**

* **Unified Compaction Strategy (UCS)**
* **Size-Tiered Compaction Strategy (STCS)**
* **Leveled Compaction Strategy (LCS)**
* **Time Window Compaction Strategy (TWCS)**

**UNIFIED COMPACTION STRATEGY :**

For the majority of workloads, including mixed read-write, time-series, read-heavy, and write-heavy workloads, the UnifiedCompactionStrategy (UCS) is advised. UCS may be made to behave like any legacy compaction strategy, thus there's no need to employ them.Compaction approach UCS incorporates new features along with the greatest aspects of previous strategies. With a novel sharding method that compacts partitioned data in parallel, UCS has been engineered to optimize the speed of compactions—a critical factor for nodes with high densities. Furthermore, UCS can alter parameters while in flight to move from one strategy to another, unlike STCS, LCS, or TWCS, which will need to compile the data completely if the compaction strategy is altered.

In actuality, a variety of compaction techniques can be applied simultaneously, with varying settings for every tier of the hierarchy. Lastly, because UCS is stateless, it may perform compaction decisions without the use of metadata.

**The definition of the grouping is improved by two crucial ideas:**

Since both methods produce levels that expand exponentially with the size of SSTables (or non-overlapping SSTable runs), they can be broadly regarded as equal. Consequently, the presence of more than a certain amount of SSTables on a single level causes a compaction.

Density can be used in place of size to produce a leveled hierarchy while enabling SSTables to be divided at any point throughout the writing of a compaction's output. Density can be calculated by dividing an SSTable's size by the width of the token range it spans.

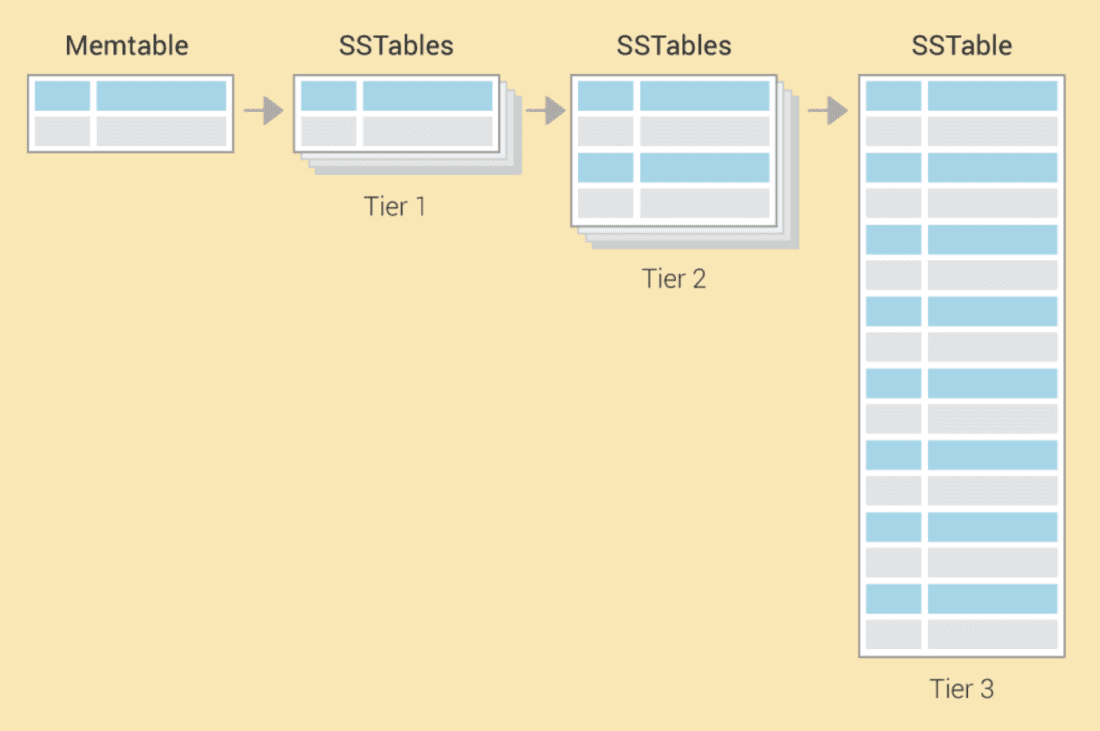
**AMPLIFICATION OF READ AND WRITE :**

When a piece of data needs to be rewritten multiple times in its lifespan (write amplification, or WA), UCS can balance the number of SSTables consulted to serve a read (read amplification, or RA). The behavior of the compaction can be adjusted by a single variable scaling parameter, which switches between a read- and write-heavy mode. Any time the scaling value is altered, the compaction approach will adapt as well. An operator might choose, for instance, to:

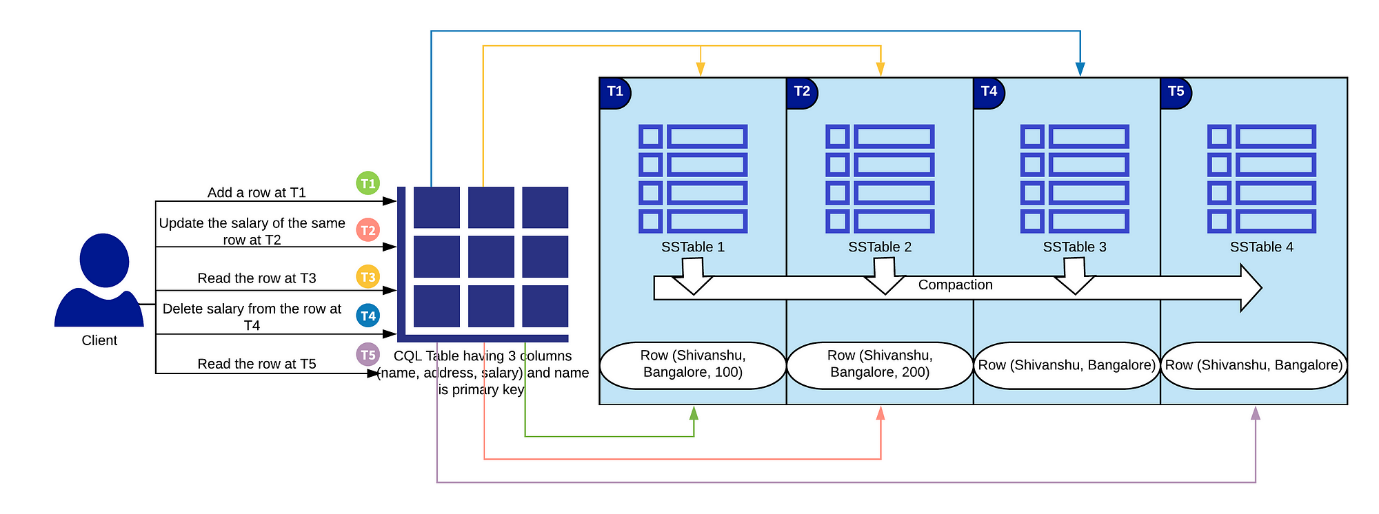
* Reduce the scaling value for a particular table that is read-heavy and could benefit from lower latencies, lowering the read amplification at the cost of more complex writes.
* When compaction is found to be unable to keep up with the volume of writes to a table, raise the scaling option to decrease the write amplification.

Any such modification only starts the compactions required to get the hierarchy in a state that is consistent with the new arrangement. Any extra effort that has already been completed is beneficial and included, such as when changing from a negative to a positive parameter.

In addition, UCS can achieve the same compaction as TWCS by configuring the scaling parameters to emulate a high tiered fanout factor.



To accomplish the required read and write amplification, UCS combines sharding with tiered and leveled compaction. Token range is used to sort SSTables, which are further divided into levels:



**SIZE BASED LEVELING :**

The method divides SSTables along certain shard boundaries, the number of which increases as an SSTable's density does. Concurrent compactions are made possible by the splitting's creation of SSTables that do not overlap. But for the time being, let's set density and splitting aside and examine how SSTables are arranged into levels in the event that they are never split.The average size of all the SSTables written when a memtable is flushed is used to determine the memtable flush size (sf). Memtables are flushed to level zero (L0). This variable, sf is meant to serve as the foundation for the hierarchy that all freshly-flushed SSTables eventually end up in. The level L for an SSTable of size s is computed using a fixed fanout factor f and sf as follows:

L={⌊logfssf⌋0if s≥sf

**DENSITY BASED LEVELING :**

All of the formulas and results hold true if we swap out the size s from the previous discussion with the density measure

d=s/v,

where v is the percentage of the token space that the SSTable covers. But today, the output may be divided at any moment in time utilizing density. The new SSTables that are created when several SSTables are split and compacted will be denser than the original SSTables. For instance, when compacted and split, four input SSTables covering 1/10 of the token area each will result in four new SSTables spanning 1/40 of the token space each when the scaling parameter is set to T4.

Because the higher density value exceeds the maximum density for the initial compacted level, these new SSTables—which will be the same size but denser—will be shifted to the next higher level. This procedure will be repeated for every shard (token range) if we can make sure that the split points are fixed (see below), carrying out separate compactions at the same time.

Assisted by   
Angala Sandeep Kumar   
https://www.linkedin.com/in/a-sandeep-kumar-061263237?utm\_source=share&utm\_campaign=share\_via&utm\_content=profile&utm\_medium=android\_app