Unified Compaction Strategy in Cassandra

Branimir Lambov,
Community over Code 2023

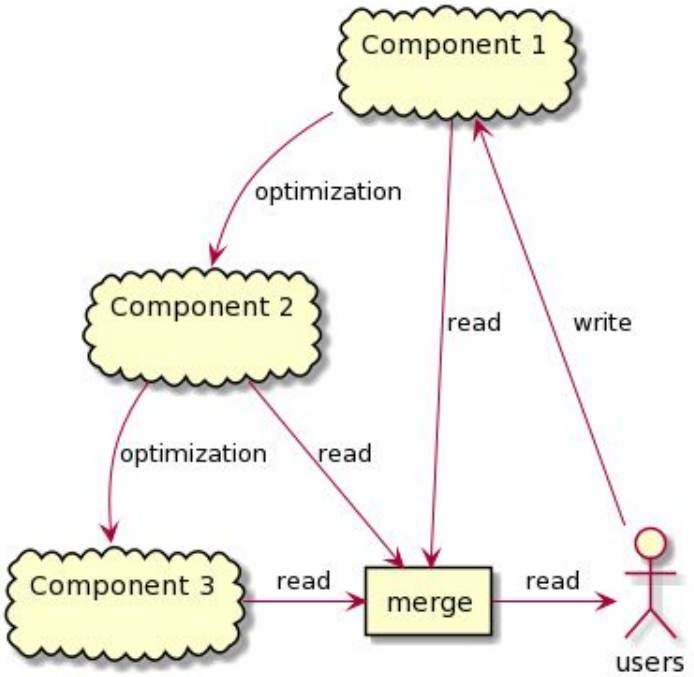
Cassandra

Distributed database with user-configurable partitioning.

Local data organized as a log-structured merge tree.

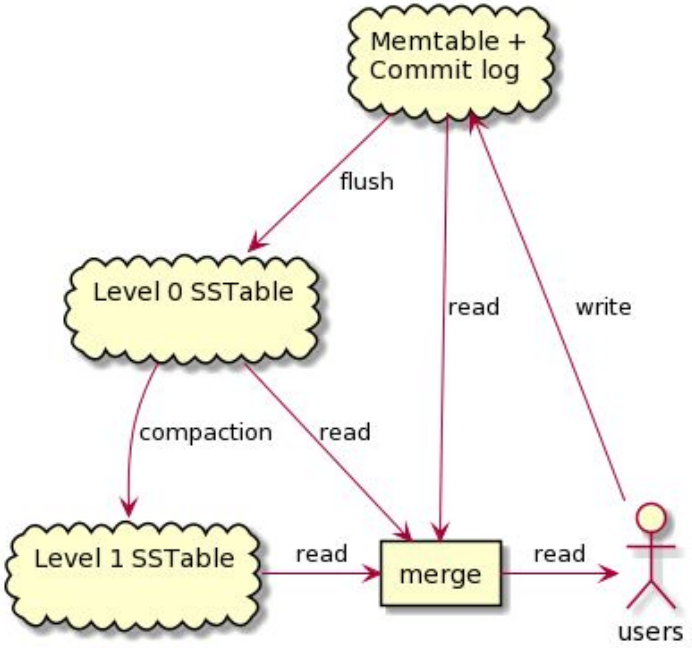
LSM-trees — 10,000 feet view

- Data is split among multiple components.
- Reads have to consult a subset or all to extract data.
- The components have some structure that define how hard reads are.
- The structure is maintained by moving data between components when certain limits are reached.



LSM-tree

- Writes are first sent to an in-memory buffer, the memtable (+ commit log for durability).
- When memtable memory is full, memtable is flushed to an on-disk sstable.
- Multiple sstables are compacted together depending on the rules specified by a compaction strategy.
- Reads consult memtables and sstables.



Compaction

Reorganizes data to make it easier to read.

Generally, balances between

- the number of sstables read per query
  (read amplification)
- the number of times a piece of data is written and rewritten
  (write amplification)

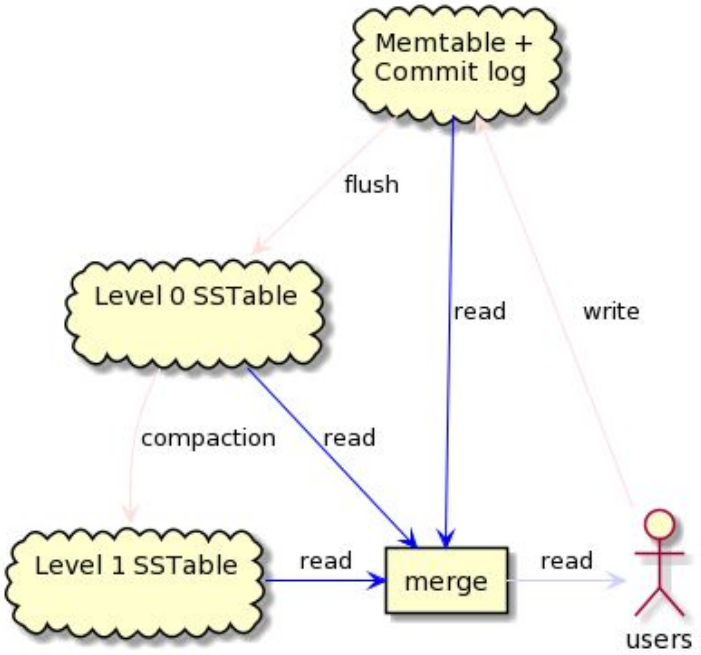
Different strategies balance between the two.

Read amplification

Number of sstables read per query.

Reduced by:

- Bloom filter
  (key-value workloads).
- Splitting sstables by time order
  (time-series workloads).
- Splitting sstables by partition order
  (most queries).

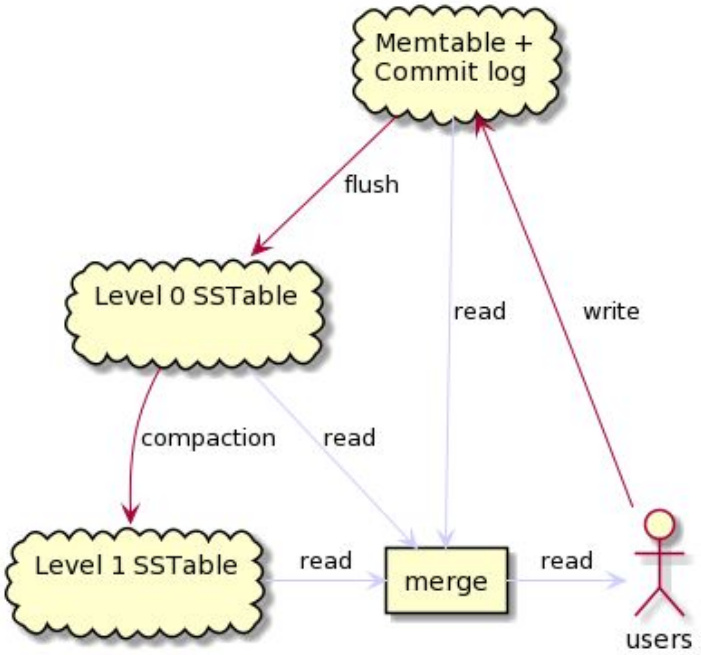


Write amplification

Number of times a piece of data is written and rewritten.

Includes commit log and flush, but dominated by compaction.

Only reflected in write latency and throughput under sustained load.



Cassandra's main compaction strategies

Size-tiered (STCS)

- Groups SSTables close in size in levels.
- Overlapping SSTables on level.
- Compacts when number of SSTables on level is at threshold or above.
- Compacts all SSTables on level.
- No SSTable splitting.

Levelled (LCS)

- Explicitly tracks SSTable level.
- Non-overlapping SSTable run on each level (except L0).
- Compacts when size of run is above threshold.
- Selects an SSTable to compact with overlapping ones in next level.
- Splits on size.

Target state of the compaction hierarchy

Size-tiered (STCS)

- SSTables grouped in levels by powers of the threshold.
- Less than threshold-many SSTables on level.

Levelled (LCS)

- One non-overlapping SSTable run on each level.
- Size of level below power of fan factor.

Target state of the compaction hierarchy

Size-tiered (STCS)

- SSTables grouped in levels by powers of the threshold.
- Less than threshold-many SSTables on level.

Levelled (LCS)

- SSTable runs grouped in levels by powers of the fan factor.
- At most one SSTable run on each level.

Compaction strategies in academia

Tiered

- Multiple overlapping SSTables per level at rest.
- One compaction to promote to next level.
- Low write and high read amplification.

Levelled

- One SSTable per level at rest.
- Multiple compactions to promote to next level.
- Low read and high write amplification.

- Levels grow by a specified fan factor.
- Splitting (vertical/sharding or horizontal/runs) is an orthogonal concern.

(Deletions/overwrites are ignored in descriptions.)

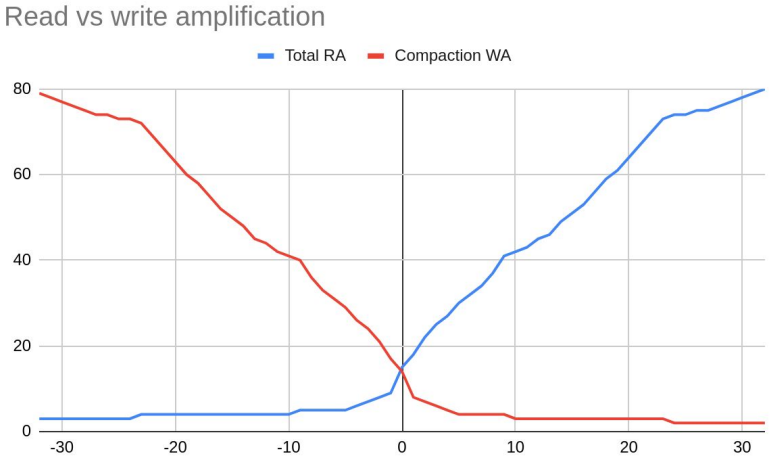
Basic unified compaction strategy

- Levels determined by SSTable size.
- Per-level size grows by a specified fan factor $f \geq 2$.
- Up to t-1 SSTables per level at rest, where $t = 2$ (levelled) or $t = f$ (tiered).
- Result moves up a level when it grows enough.

Level	Min sstable size	Max sstable size
0	0	$m \cdot f$
1	$m \cdot f$	$m \cdot f^2$
2	$m \cdot f^2$	$m \cdot f^3$
3	$m \cdot f^3$	$m \cdot f^4$
…	…	…
n	$m \cdot f^n$	$m \cdot f^{n+1}$

Basic unified compaction strategy

- Levels determined by SSTable size.
- Per-level size grows by a specified fan factor $f \geq 2$.
- Up to t-1 SSTables per level at rest, where $t = 2$ (levelled) or $t = f$ (tiered).
- Result moves up a level when it grows enough.
- Configurable read and write amplification via one integer scaling parameter w:
  - $f = 2 + w$, t = f when $w \geq 0$ (tiered)
  - $f = 2 - w$, t = 2 when $w \leq 0$ (levelled)



Per-level scaling parameters

Separate value of w for each level.

For example:

- 2, 2, 2, -8 / T4, T4, T4, L10
- 4, 2, 0, -2 / T6, T4, L2, L4

Better ability to accept bursts of data and still keep it well organized for reads.

Especially helpful when the table uses a lot of deletions.

Note:

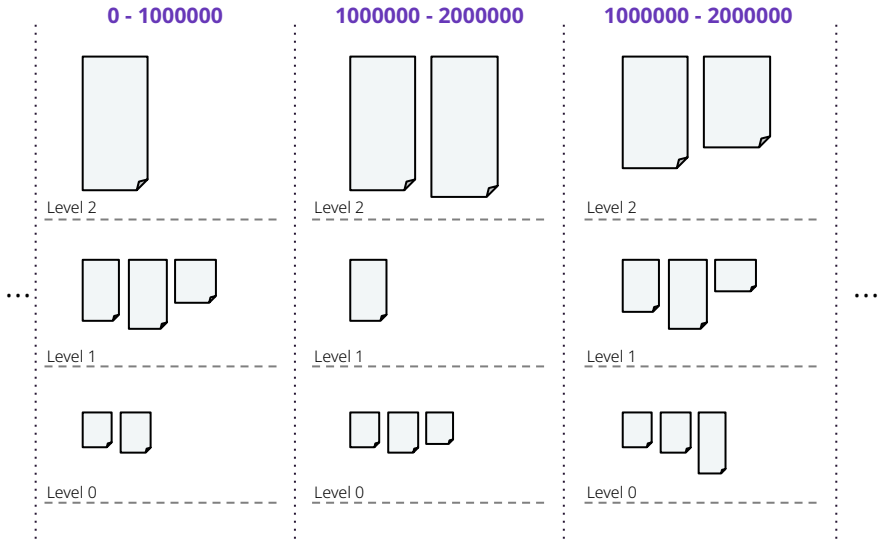
- w can be specified as Tf / Lf for readability.
- Last value repeats for all higher levels.

Sharding

Splitting SSTables at predefined positions.

- Can be achieved via data directories.
- Can be used to easily scale STCS or basic UCS by ~10x.
- Reduces space overhead.
- Adds parallelism.

- Many small SSTables on lower levels.
- Moving boundaries is hard.



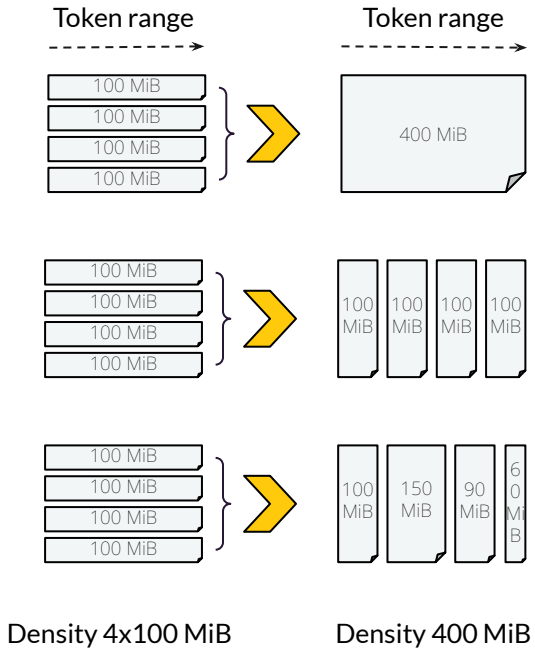
Density and overlap

Density: size of SSTable divided by token share.

- Grows when SSTables are bigger in size.
- Also grows when SSTables are split.

Overlap: only count overlapping SSTables towards threshold.

- Overlap drives read amplification.
- Compaction buckets can be formed from overlap sections (transitively extended).



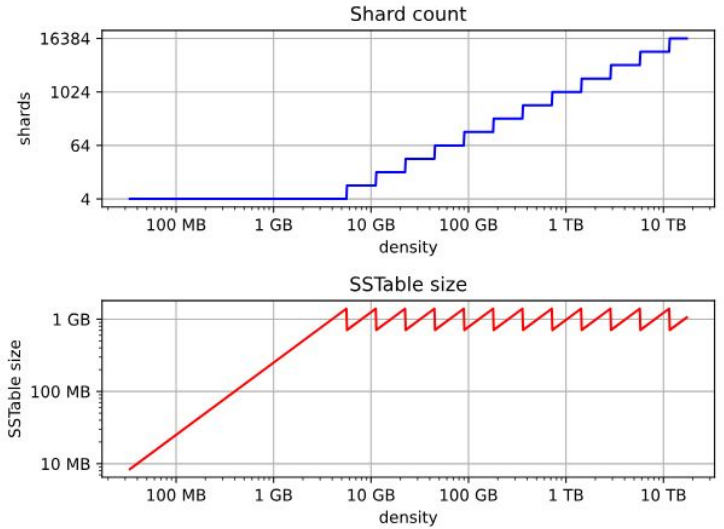
Unified compaction strategy

- Levels determined by SSTable *density*.
- Per-level density grows by a specified fan factor $f \geq 2$.
- Up to t-1 *overlapping* SSTables per level at rest, where $t = 2$ (levelled) or $t = f$ (tiered).
- Result moves up a level when it grows enough.
- Configurable read and write amplification via one integer scaling parameter w:
  - $f = 2 + w$, t = f when $w \geq 0$ (tiered)
  - $f = 2 - w$, t = 2 when $w \leq 0$ (levelled)

UCS sharding scheme

When compaction starts:

- Calculate expected result density.
- Define boundaries to split into SSTables of close to target size t.
- Only split in the middle;
  i.e. total number of shards is a power of 2 multiple of base shard count b.
- Any boundaries also apply to all higher densities.

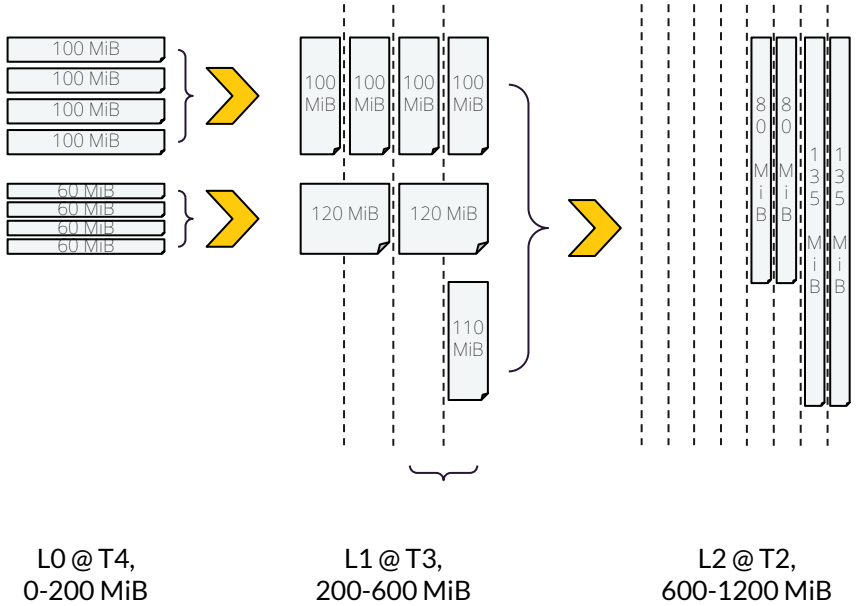


Size vs shard count for t = 1 GiB and b = 4.

UCS progression example

Parameters:

- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1



UCS progression example

Parameters:

- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1

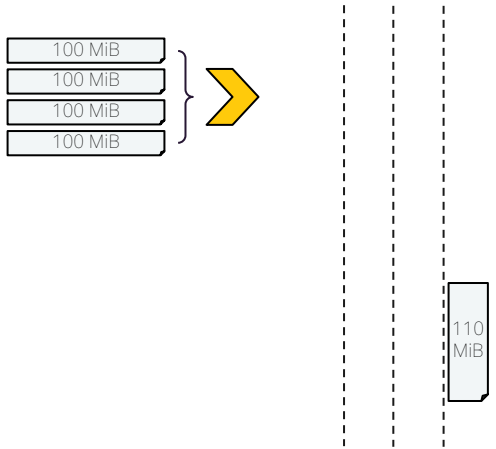
L0 @ T4,
0-200 MiB

L1 @ T3,
200-600 MiB

UCS progression example

Compaction triggered, calculate shard count:

400MiB with token range 1 → 4 shards

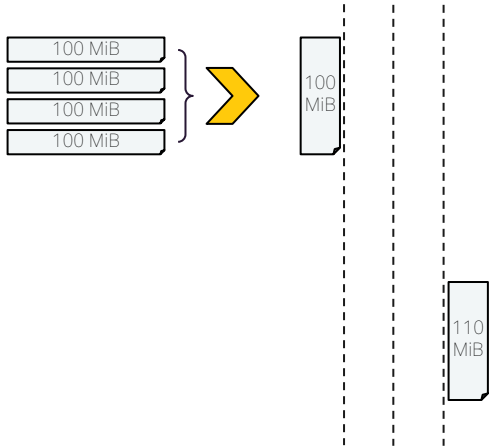


Parameters:

- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1

UCS progression example

Write from beginning, splitting on each boundary.

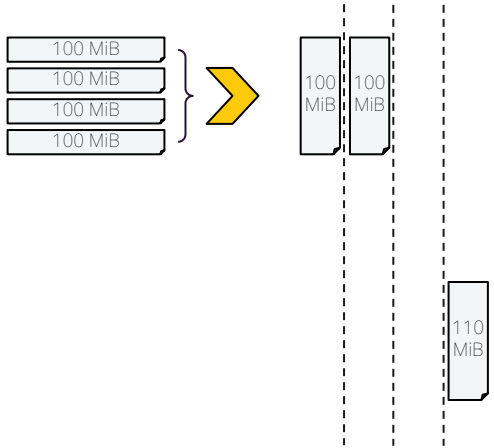


Parameters:

- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1

UCS progression example

Write from beginning, splitting on each boundary.

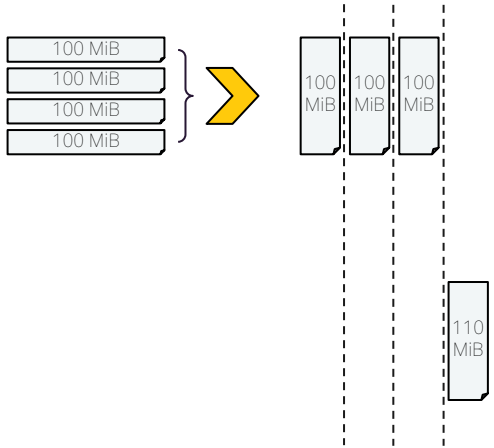


Parameters:

- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1

UCS progression example

Write from beginning, splitting on each boundary.



Parameters:

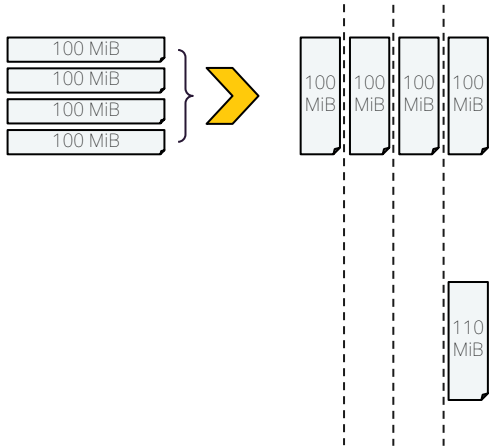
- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1

L0 @ T4,
0-200 MiB

L1 @ T3,
200-600 MiB

UCS progression example

Each resulting SSTable has density 400 MiB.



Parameters:

- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1

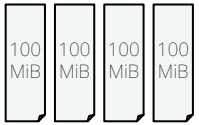
L0 @ T4,
0-200 MiB

L1 @ T3,
200-600 MiB

UCS progression example

Delete sources.

Shard boundaries no longer relevant.



Parameters:

- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1

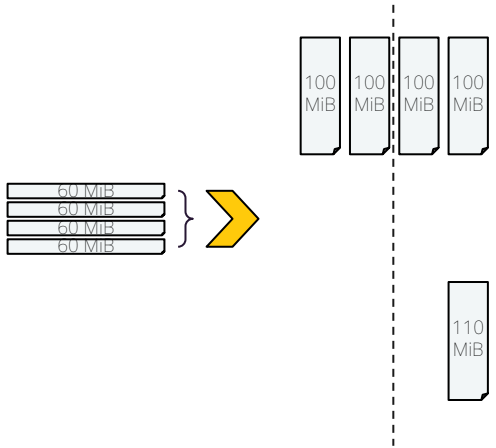
L0 @ T4,
0-200 MiB

L1 @ T3,
200-600 MiB

UCS progression example

New set of sources, calculate shards:

240 MiB with token range 1 → 2 shards

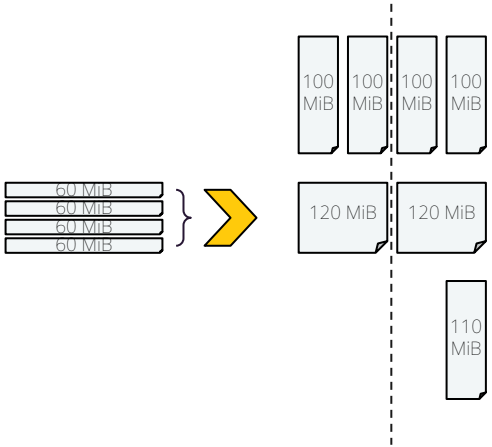


Parameters:

- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1

UCS progression example

Switch writer once.



Parameters:

- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1

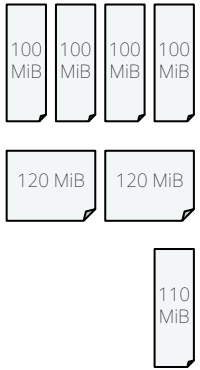
L0 @ T4,
0-200 MiB

L1 @ T3,
200-600 MiB

UCS progression example

Delete sources.

Shard boundaries no longer relevant.



Parameters:

- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1

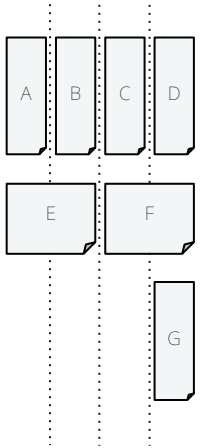
L0 @ T4,
0-200 MiB

L1 @ T3,
200-600 MiB

UCS progression example

Identify overlap sections: AE, BE, CF, DFG

DFG triggers threshold.



Parameters:

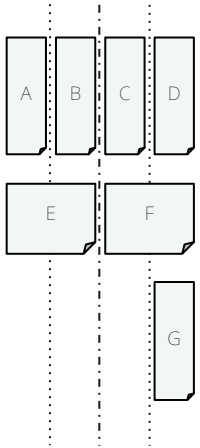
- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1

L0 @ T4,
0-200 MiB

L1 @ T3,
200-600 MiB

UCS progression example

Extend DFG bucket for overlaps of F to CDFG.



Parameters:

- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1

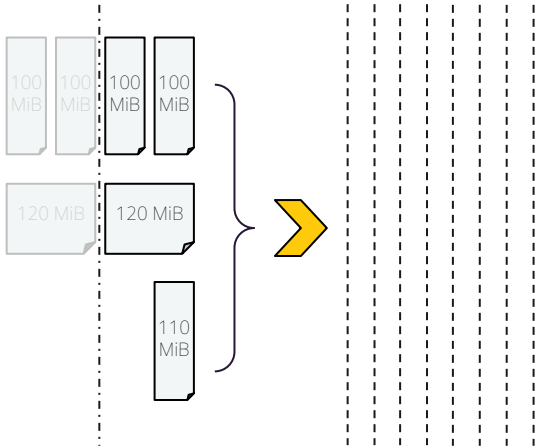
L0 @ T4,
0-200 MiB

L1 @ T3,
200-600 MiB

UCS progression example

Start compaction on bucket and calculate shards:

430 MiB in ½ space → 860 MiB density, 8 shards



Parameters:

- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1

L0 @ T4,
0-200 MiB

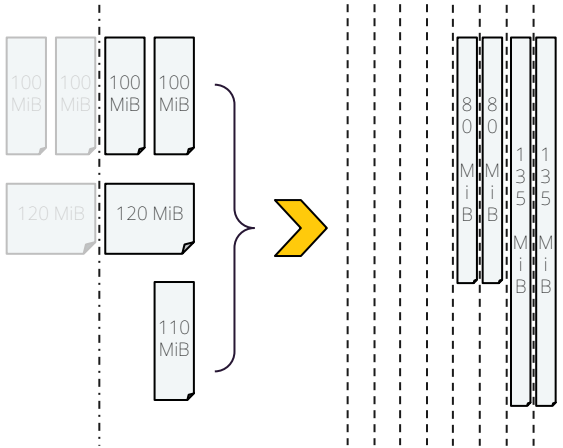
L1 @ T3,
200-600 MiB

L2 @ T2,
600-1200 MiB

UCS progression example

2 SSTables with density 640 MiB

2 SSTables with density 1080 MiB



Parameters:

- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1

L0 @ T4,
0-200 MiB

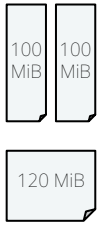
L1 @ T3,
200-600 MiB

L2 @ T2,
600-1200 MiB

UCS progression example

Delete source SSTables.

Shard and bucket boundaries no longer relevant.



Parameters:

- Base SSTable size m = 50 MiB
- Scaling parameters T4, T3, L2, L4
- Target SSTable size t = 100 MiB
- Base shard count b = 1

L0 @ T4,
0-200 MiB

L1 @ T3,
200-600 MiB

L2 @ T2,
600-1200 MiB

Compaction prioritization

When compaction is late and there are too many compactions waiting, prioritize ones that reduce read amplification most:

- Buckets with the highest overlap.
- On equal overlap, prefer lower level.
- On equal and same level, choose randomly.

Avoids accumulation of SSTables on any level and should lead to a stable state able to handle sustained load.

Time series workloads with TTL

UCS supports whole table expiration.

Level is a proxy for age.

UCS avoids unnecessarily mixing SSTables of different age in compactions.

Higher-fan-factor UCS (e.g. with scaling parameter T20) works pretty well.

Changing scaling parameters

As the strategy is stateless, it just switches to different target state.

New compactions may be triggered.

Work already done is still beneficial.

Splitting/sharding is not affected.

Upgrade from LCS and STCS

UCS has corresponding scaling parameters: L10 for LCS default, T4 for STCS default.

Density understands the progression of data in both.

Overlaps allows trigger decisions to work correctly initially as well as in mixed states.

Upgrading from LCS may trigger some compactions.

Further extensions / future work

- Target size growth: Control how much growth should be taken by the shard/SSTable count vs. the SSTable size.
- Time-based levels: Allow a time component in levelling decisions, to fully cover TWCS applications, and to also support modes like “compact everything together every week” for controlling tombstone numbers.
- Adaptive compaction: Measure read/write load and costs of reads and writes and change scaling parameters to optimize resource usage.

Thank you!