7019-Test: Perf Comparison [LCS - provide_overlapping_tombstones + experiments]

Setup

3 nodes cluster.
200+GB of data per node.
Single table.
Schema:
First phase, run steady state with

```
--compaction "{'class': 'LeveledCompactionStrategy'}"
--compression "{'sstable_compression': 'LZ4Compressor'}"
```

Second phase, after running for 2 hours, change the compaction to

```
--compaction "{'class': 'LeveledCompactionStrategy',
               'provide_overlapping_tombstones': 'row'}"
```

GC is amortized over each background compaction, since the table sets `provide_overlapping_tombstones` to row.
QPS: 3K/s.
Read : Write : Delete = 5 : 4 : 1

Timings

Trunk:
Steady State Start Time: Mon Dec 14 17:41:37 PST 2020
Altering `provide_overlapping_tombstones` Time: Mon Dec 14 19:41:40 PST 2020

GarbageSkipper optimization that avoids the step if cassandra.shodow_sources_max_allowed_sstable_candidates >= N:
N == 20:
Steady State Start Time: Wed Jan 6 21:35:15 PST 2021
Altering `provide_overlapping_tombstones` Time: Now: Wed Jan 6 22:36:31 PST 2021

N == 10:
Steady State Start Time: Fri Jan 8 03:17:42 PST 2021
Altering `provide_overlapping_tombstones` Time: Now: Fri Jan 8 04:17:44 PST 2021

Result

Trunk:

Metric	Steady State	provide_overlapping_tombstones == row
Read Throughput	1.5k/s	1.5k/s
Read Latency avg.	4.65k micros	4.73k micros → 6.99k micros (smoothed average)
Read Latency p95	25.33k micros	20.39k micros → 21.66k micros
Read Latency p99	55.09k micros	58.85k micros → 62.20k micros
Write Throughput	1.5k/s	1.5k/s

Write Latency avg.	452.85 micros	458.59 micros
Write Latency p95	795.38 micros	800.38 micros
Write Latency p99	1.01k micros	1.04k micros → 1.18k micros

Optimization (N == 20):

Metric	Steady State	provide_overlapping_tombstones == row
Read Throughput	1.5k/s	1.5k/s
Read Latency avg.	6.36k micros	6.11k micros → 7.28k micros (smoothed average)
Read Latency p95	32.33k micros	23.22k micros → 27.5k micros
Read Latency p99	65.53k micros	62.85k micros → 77.81k micros
Write Throughput	1.5k/s	1.5k/s
Write Latency avg.	482.41 micros	501.31 micros
Write Latency p95	831.02 micros	877.25 micros
Write Latency p99	1.04k micros	1.26k micros

Optimization (N == 10):

Metric	Steady State	provide_overlapping_tombstones == row
Read Throughput	1.5k/s	1.5k/s
Read Latency avg.	4.26k micros	4.91k micros → 5.34k micros (smoothed average)
Read Latency p95	22.26k micros	21.75k micros → 25.32k micros
Read Latency p99	54.18k micros	55.78k micros → 61.69k micros
Write Throughput	1.5k/s	1.5k/s
Write Latency avg.	495.09 micros	506.38 micros
Write Latency p95	953.75 micros	964.25 micros
Write Latency p99	1.02k micros	1.02k micros

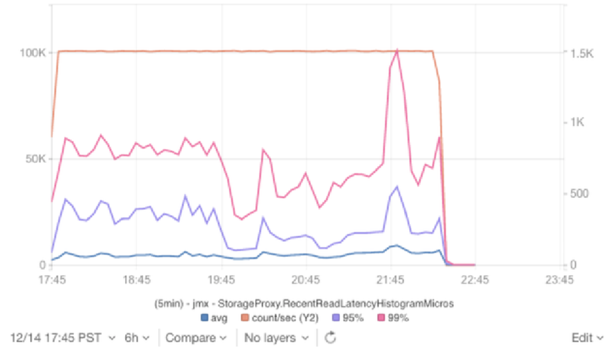
Read & Write Throughput and Latencies

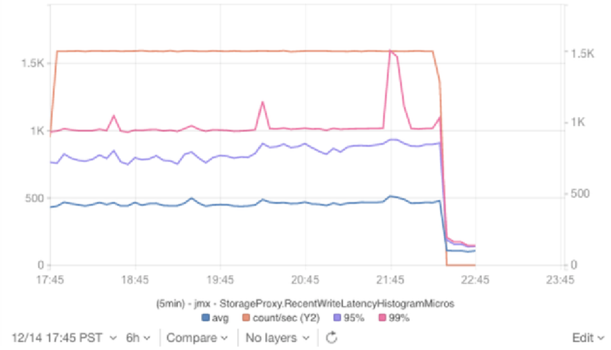
Trunk

After altering the `provide_overlapping_tombstones` to `row` for the table, we can observe the dip in read latencies. The smoothed average latency in the second phase is close to the latency from the first phase, but with a higher volatility. It also spiked towards the end of the test.

For the write latency, there is no significant change before and after altering the table. There is also a spike of write latency near the end of the test.

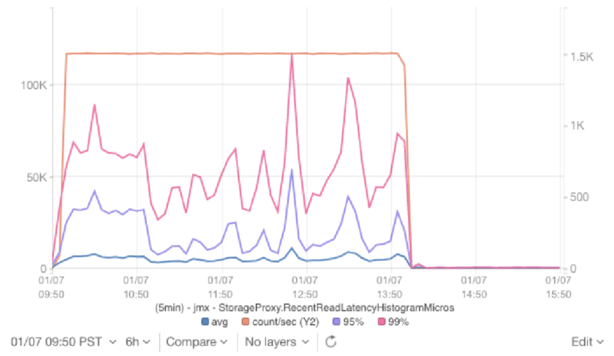
The spikes are related with the compaction load.

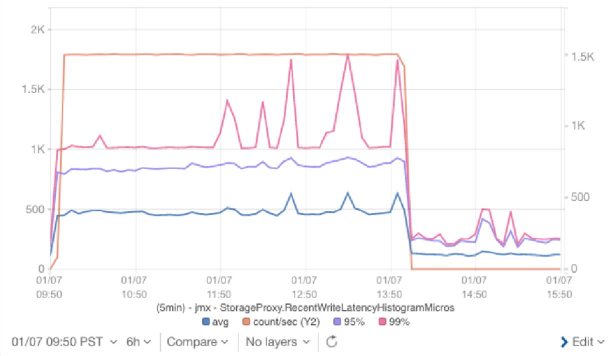




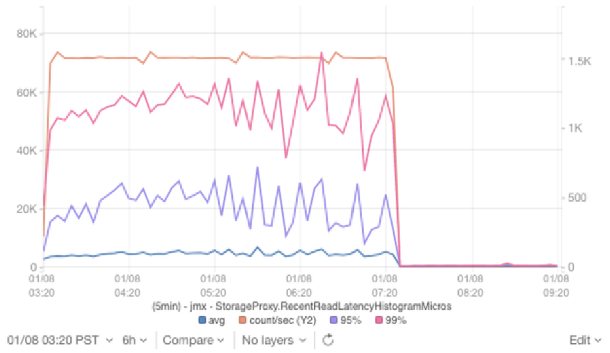
Optimization

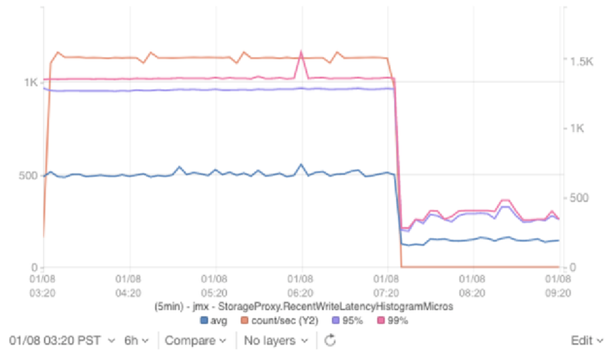
When N == 20, the read latency drops generally, but the tail latency also gets higher.
The write tail latency increases.





When N == 10, the read latency is more choppy. It provides both lower and higher latencies comparing with the one from steady state phase over the test runtime. If compare with the run (N == 20), the latency graph is more stable.
The write latency is about the same.





Timeouts

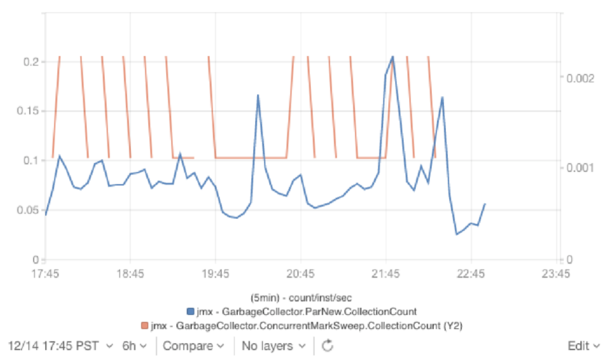
No timeouts are observed from the run.

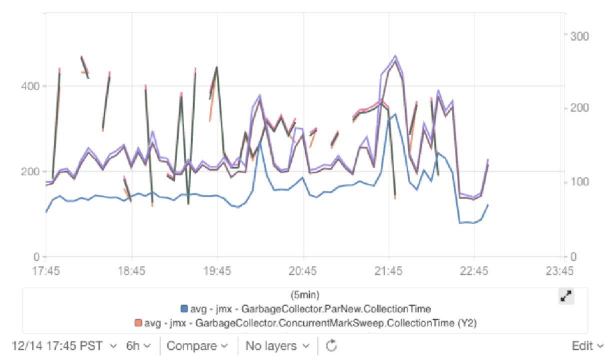
No data to display
count/sec
jmx - ClientRequestMetrics.ReadTimeouts
jmx - ClientRequestMetrics.ReadUnavailables (Y2)
jmx - ClientRequestMetrics.WriteTimeouts
jmx - ClientRequestMetrics.WriteUnavailables (Y2)
12/14 17:45 PST
6h
Compare
No layers
Edit

JVM GC Count & Duration

Trunk:

`ParNew` is more frequent and takes longer time in the second phase.



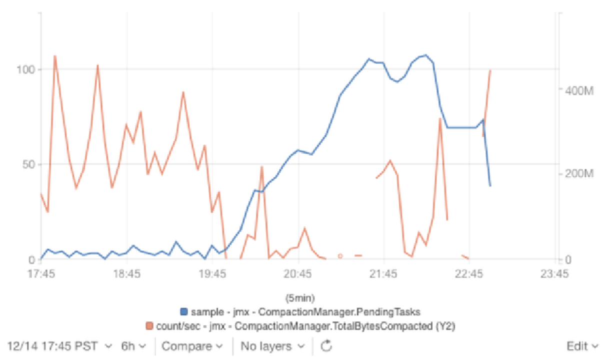


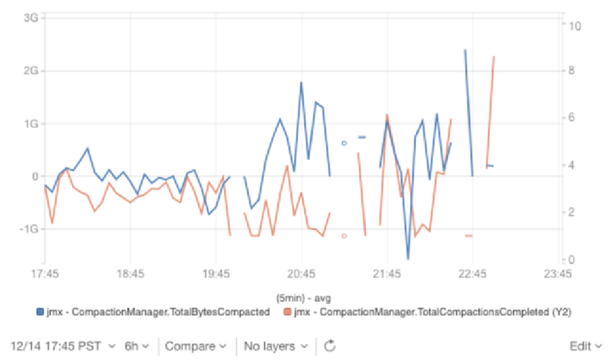
Compaction Rate & Throughput

Trunk:

The number of the pending compaction tasks builds up fast after enabling `provide_overlapping_tombstones == row`. The cause should be that each compaction task takes a longer time to complete due to the garbage skipping step introduced in the second phase.

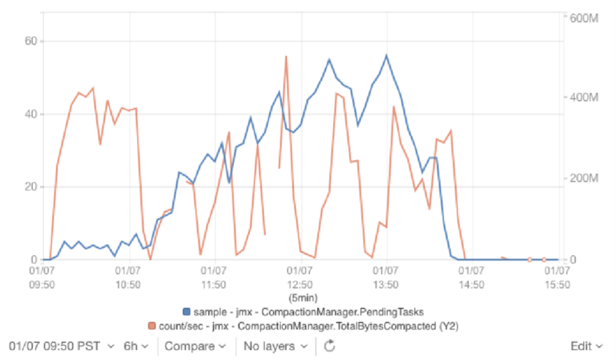
Meanwhile, the compaction throughput is less in the second phase. Because each compaction task spends more time in computation (i.e. skipping tombstone'd data).

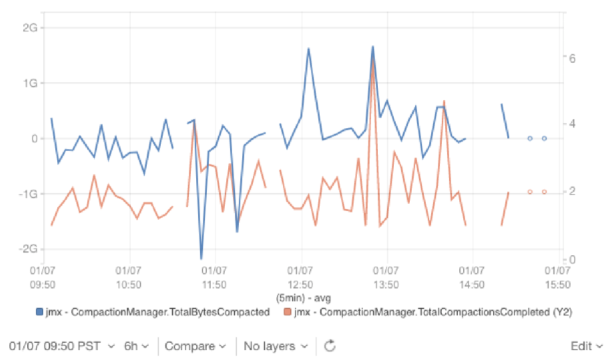




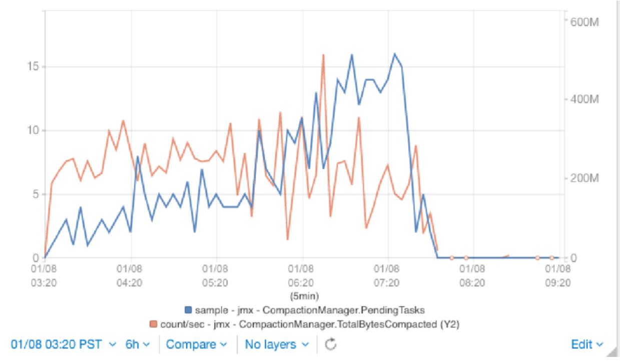
Optimization:

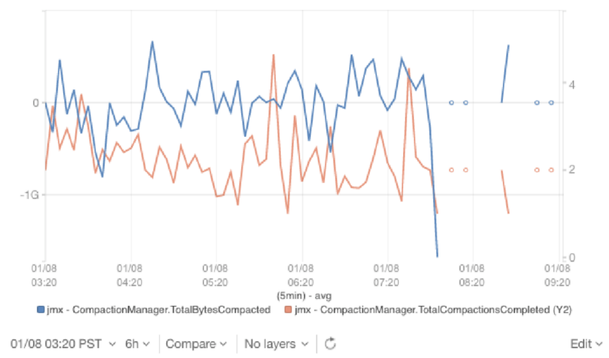
When N == 20, the number of the pending tasks is lower and the compaction throughput is higher comparing with the Trunk run. However, the compaction throughput is still lower than when the feature is disabled.





When N == 10, the number of the pending tasks is further lower. The compaction throughput is higher when comparing with the run (N == 20). However, it is still lower when comparing with the steady state phase.



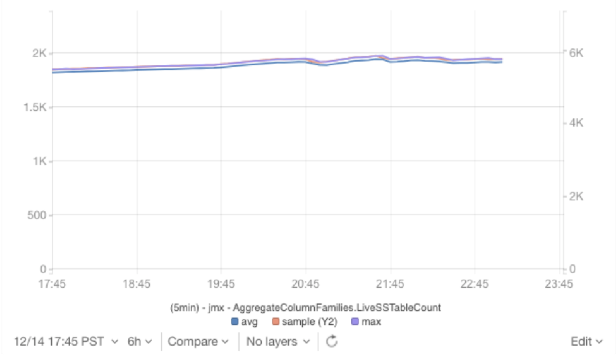


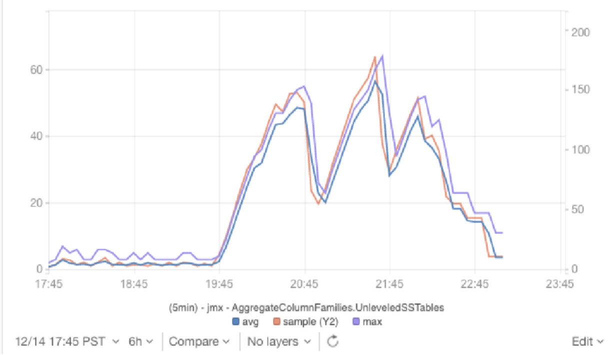
Live SSTable & Unleveled SSTable Count

Trunk:

The number of live SSTables rises slightly when enabling `provide_overlapping_tombstones`. It eventually reduced to a similar level in the first phase.

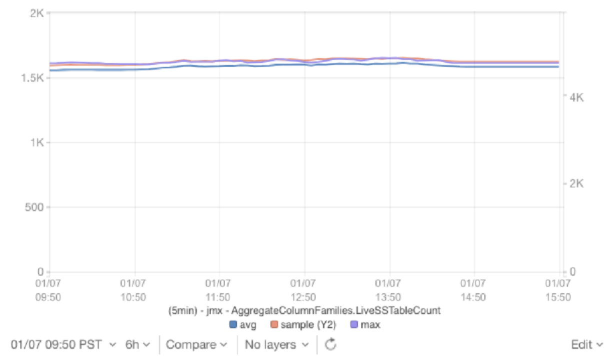
The number of unleveled SSTables rises a lot right after the schema change. The count oscillates between 55 and 178 in the second phase. Meanwhile, in the first phase, the count is stable around 6.

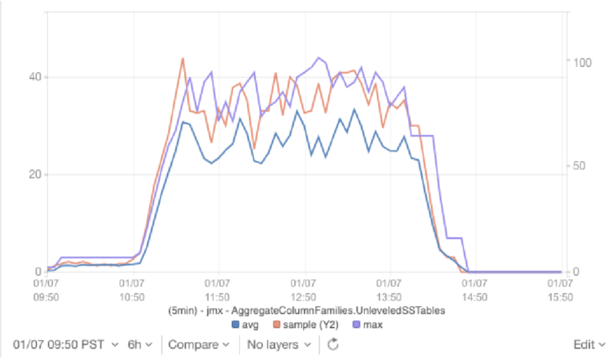




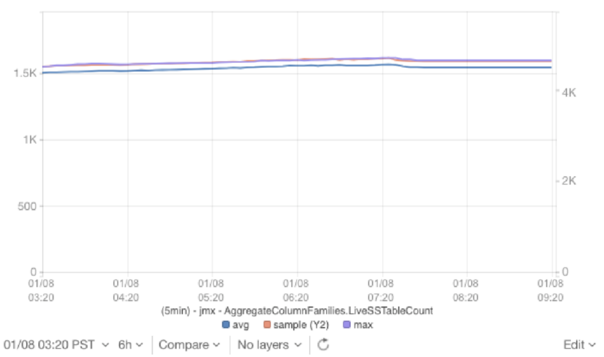
Optimization:

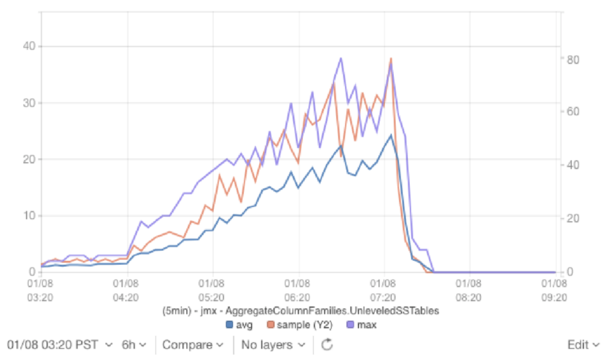
When N == 20, we have less unleveled SSTables comparing with the Trunk run. The number of the unleveled sstables are more stable over time.





When N == 10, the number of the unleveled sstables ramps up slower and the sum is slightly lower when comparing with the run (N == 20).

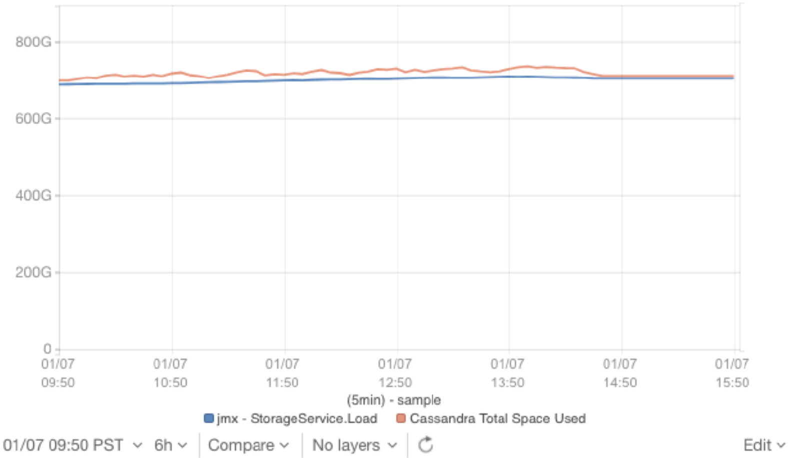




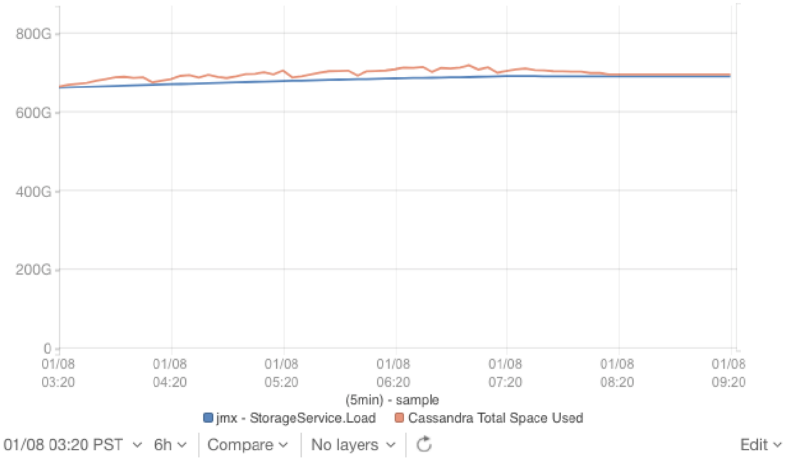
Disk Space Used

Optimization:

When N == 20, we do not see a significant drop in disk usage after enabling the GarbageSkipper step.



When N == 10, the disk usage is still about the same after enabling the GarbageSkipper step.


800G
600G
400G
200G
01/08 03:20
01/08 04:20
01/08 05:20
01/08 06:20
01/08 07:20
01/08 08:20
01/08 09:20
(5min) - sample
jmx - StorageService.Load
Cassandra Total Space Used
01/08 03:20 PST
6h
Compare
No layers
Edit