7019-Test: Perf Comparison [12/14 LCS - provide_overlapping_tombstones]

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':
```

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

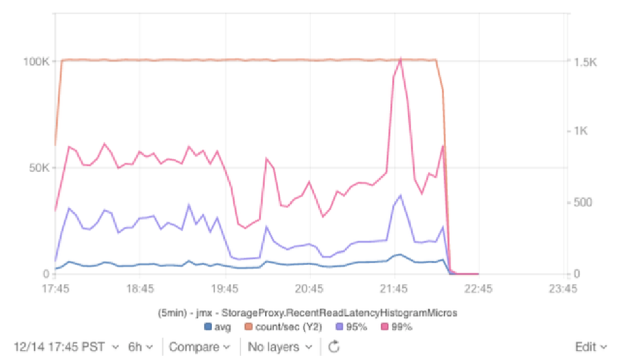
Timings

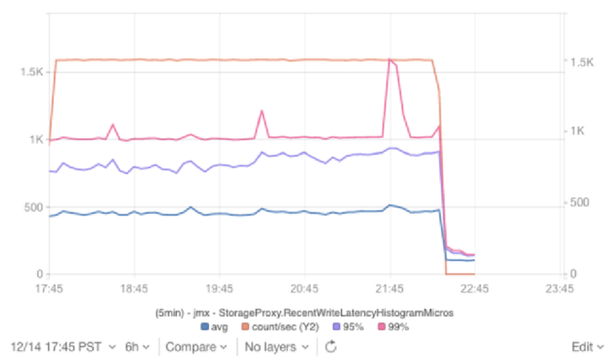
Steady State Start Time: Mon Dec 14 17:41:37 PST 2020
Altering `provide_overlapping_tombstones` Time: Mon Dec 14 19:41:40 PST 2020

Result

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
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

Read & Write Throughput and Latencies
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.





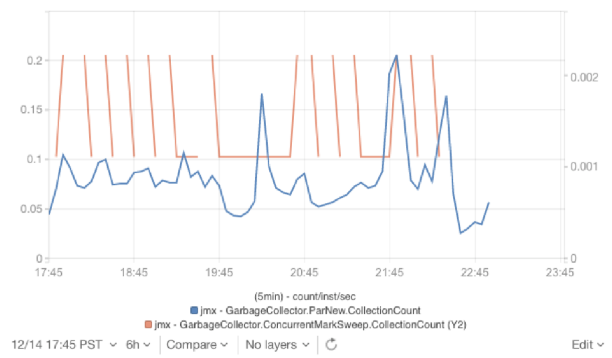
Timeouts

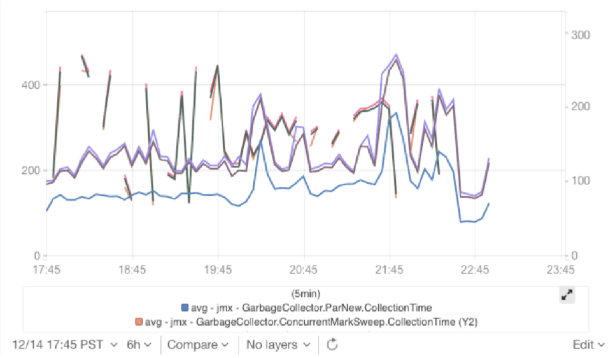
No timeouts are observed from the run.

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

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

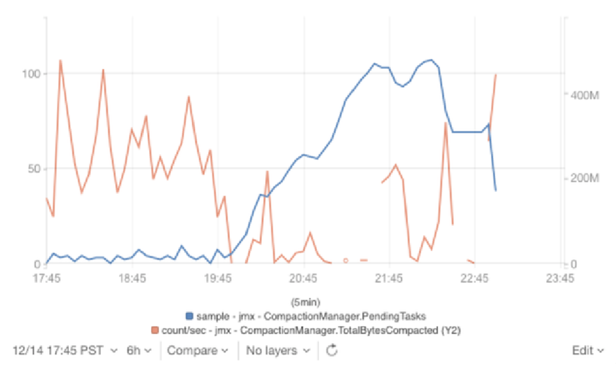


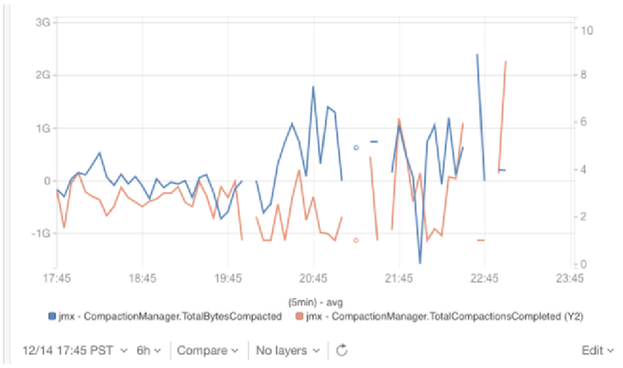


Compaction Rate & Throughput

The number of 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).

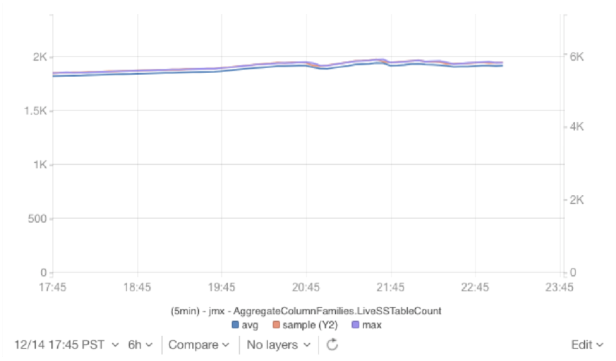


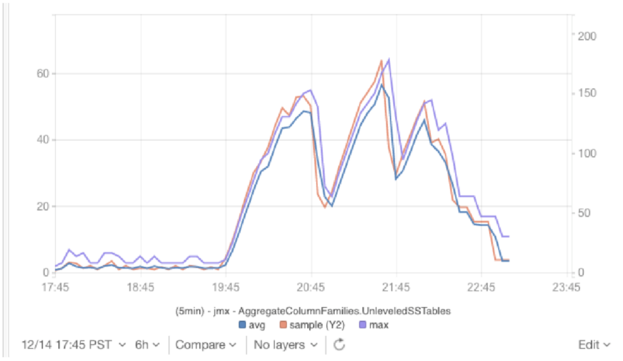


Live SSTable & Unleveled SSTable Count

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.


2K
1.5K
1K
6K
4K
2K
17:45
18:45
19:45
20:45
21:45
22:45
23:45
(5min) - jmx - AggregateColumnFamilies.LiveSSTableCount
avg
sample (Y2)
max
12/14 17:45 PST
6h
Compare
No layers
Edit


17:45
18:45
19:45
20:45
21:45
22:45
23:45
(5min) - jmx - AggregateColumnFamilies.UnleveledSSTables
avg
sample (Y2)
max
12/14 17:45 PST
6h
Compare
No layers
Edit