HBASE-16417: Parameter Tuning for Basic In-Memory Compaction Policy

One of the main sources of memory management overhead is the skiplist data structure used to index the active memory segment (A).
Not only is it bigger in size compared to a flat index, it is also fragmented whereas a static index is stored in a consecutive block of memory. Therefore, flat storage incurs smaller overhead in terms of allocation, GC, and cache misses.
We first tune the size of this data structure.

We evaluate the Basic policy with different bounds on the active segment fraction: A=0.25, 0.1, 0.05, and 0.02.
None (no compaction) has no static memory segments, hence its throughput is designated with a single point A=1.
We measure throughput in a write-only Zipf workload, on SSD, for the four values of A, and with a pipeline size S=2.
The throughput of five runs for each segment size are depicted in Figure 1.
The store scales as the active segment memory fraction decreases.

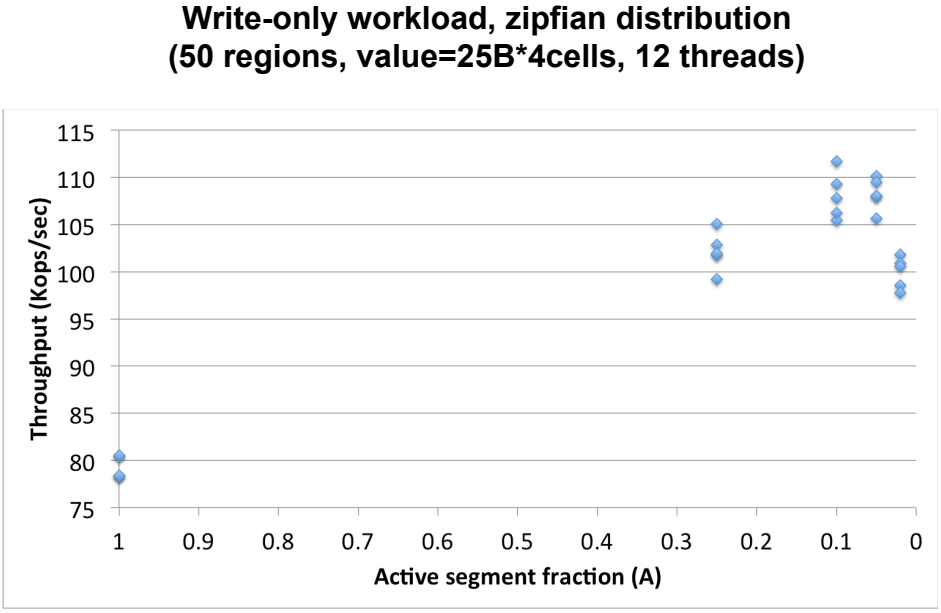

Figure 1. Tuning of the active segment memory fraction, A with S=2, Zipf distribution on SSD. Five experiments are conducted for each A. The write throughput is higher for small values of A

The next parameter is the pipeline size.
When A is less than 10%, it is clear that merging the active segment into the much bigger static data structure over and over again can be inefficient, as it creates a new index and releases the old one.
The alternative is to batch multiple segments in the pipeline before merging into a single segment.
On the flip side, the read API's must scan all the segments, which degrades their performance.
Figure 2 depicts the throughput results as a function of the number of segments in the pipeline.
The peak throughput is achieved with S=5 on SSD (Figure 2a) and S=4 on HDD (Figure 2b).

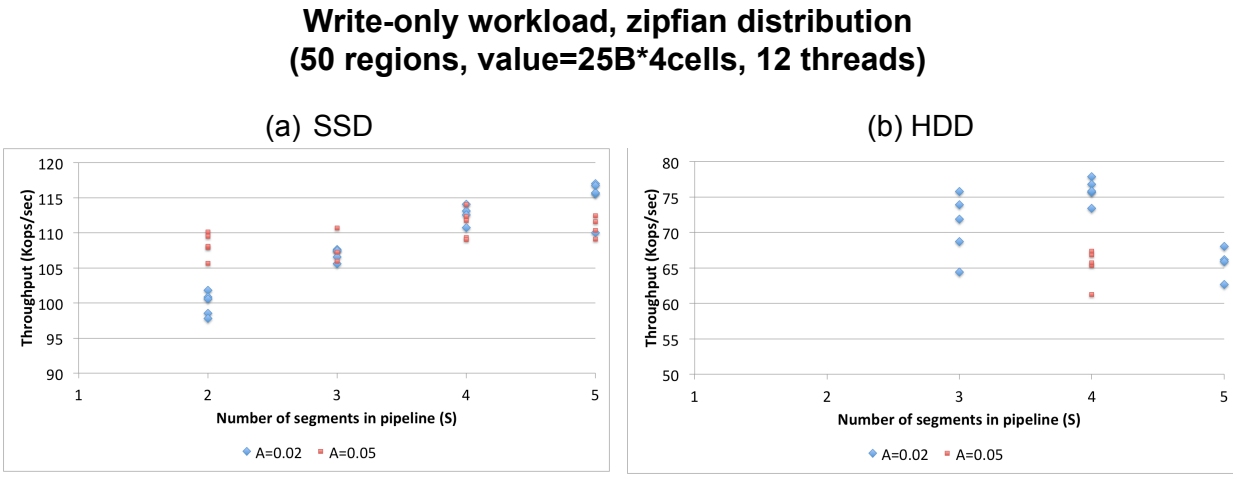

Figure 2. Tuning of the pipeline size bound, S. Five experiments are conducted for each S.