IntegrationTestLoadSmallValues

10M rows (1.8 GB)

Summary

Use of a precomputed dictionary improved compression time/throughput by 20-40% at each level. Given a target compaction time limit, this enables selection of higher compression levels that can still fit within the available budget. For this test a compaction budget of two minutes (120 seconds, +- 30 seconds) was arbitrarily chosen. Imagine a hypothetical use case where longer compaction times would eventually compromise global ingestion throughput. LZ4, Snappy, and LZMA (at max level 9) results are included for comparison. Use of the precomputed dictionary, in combination with the higher level allowable for the given time budget, improved results over even LZMA: 73.5% at LZMA level 9 vs 73.8% at ZStandard level 15 with precomputed dictionary, where ZStandard was 15.6x faster than LZMA at achieving the result.

No Dictionary

Codec/Level	**On Disk Size**	**Compression**	**Compaction Time (sec)**
-	1,984,528,440	-	-
(LZ4)	901,868,717	54.5%	32
(SNAPPY)	853,309,441	57.0%	33
ZStandard:			
1	587,130,216	70.4%	42
3	581,754,296	70.7%	43
5	579,616,563	70.8%	52
6	576,213,501	70.9%	58
7	575,932,305	71.0%	64
12	557,383,171	71.9%	147
~~15~~	~~555,402,218~~	~~72.0%~~	~~223~~
~~18~~	~~554,696,633~~	~~72.0%~~	~~432~~
~~22~~	~~554,693,110~~	~~72.0%~~	~~535~~
(LZMA, level 9)	525,018,303	73.5%	1704

With Dictionary (k=32,d=6)

Codec/Level			
ZStandard:			
1	592,115,443	70.2%	26
3	563,545,592	71.6%	29
5	544,946,521	72.5%	35
6	543,241,456	72.6%	36
7	522,370,471	73.7%	52
12	521,391,902	73.7%	88
15	**520,652,905**	**73.8%**	**109**
~~18~~	~~499,449,760~~	~~74.8%~~	~~417~~
~~22~~	~~499,442,482~~	~~74.8%~~	~~430~~

—

Load 10 million rows. Flush, then major compact, so it becomes one big HFile, segmented into 2K blocks.

```
    create "IntegrationTestLoadSmallValues", { NAME => 'i', VERSIONS => 1000, BLOCKSIZE => 2048 },
    SPLIT_POLICY => 'org.apache.hadoop.hbase.regionserver.DisabledRegionSplitPolicy'

    ./bin/hbase org.apache.hadoop.hbase.test.IntegrationTestLoadSmallValues
```

—

Extract blocks

```
    ./bin/hbase org.apache.hadoop.hbase.test.util.HFileBlockExtracter file:///Users/apurtell/tmp/train/t hdfs://
    localhost:8020/hbase/data/default/IntegrationTestLoadSmallValues/fca08ac71d0127f1c06ced66784afab2/i/
    561dd26bfffb47228b02bfcf2f63f114
```

    ^C after > 25000 block files written, prune training set to 00000 - 24999 (25,000 blocks)

Train

```
    zstd --maxdict=$((1024*1024)) -o ITLSV.dict --train-fastcover=k=32,d=6 t/*
    Training on 25000 samples of total size 51581456
    Testing on 25000 samples of total size 51581456
    Computing frequencies
    Building dictionary
    Breaking content into 32768 epochs of size 1574
    statistics ...
    Constructed dictionary of size 1048576
    Save dictionary of size 1048576 into file dict

    ./bin/hadoop fs -copyFromLocal ITLSV.dict hdfs://localhost:8020/dicts/ITLSV.dict
```

—

Alter compression level, then get resulting compacted store file size, repeat for all desired levels.

```
    alter "IntegrationTestLoadSmallValues", {NAME=>'i', COMPRESSION=>'ZSTD'} , CONFIGURATION =>
    { 'hbase.io.compress.zstd.level' => '1' }

    alter "IntegrationTestLoadSmallValues", {NAME=>'i', COMPRESSION=>'ZSTD'} , CONFIGURATION =>
    { 'hbase.io.compress.zstd.dictionary' => 'hdfs://localhost:8020/dicts/ITLSV.dict' ,
    'hbase.io.compress.zstd.level' => '1'  }
```