

LSM Configurations

Write Buffer size = 4 MB (`write_buffer_size`)

SST file size = 1 MB (`target_file_size_base`)

Number of files is level 0 = 4 (`level0_file_num_compaction_trigger`)

Size ratio = 2 (`max_bytes_for_level_multiplier`)

Size of level 1 = Write Buffer size * Number of file is level 0 * Size ratio = 32 MB
(`max_bytes_for_level_base`)

`level_compaction_dynamic_level_bytes` = false

`num_levels` -> this should be computed based on your tree size (9 should work in this setup)

`disable_wal` = true

`cache_index_and_filter_blocks` = true

Block size = 4 KB (`block_size`)

Bloom filter bits per key = 10 (`bloom_bits`)

Compression should be disabled (`compression_type=none`)

Direct I/O should be enabled if you are measuring throughput/latency. (`use_direct_reads`,
`use_direct_io_for_flush_and_compaction`)

Workload

Key Size = 64 bytes

Value Size = 440 bytes

Bulk load the database with 10M `filluniquerandom` operations. Total tree size becomes
~5GB

Then run `waitforcompaction`. Back up this DB and reuse the same copy for every run
(`use_existing_db=true`, `disable_auto_compactions=true`).

Lookup = random point lookups (`readrandom`). Note: reads is per thread, so total lookups =
threads * reads.

Skewed lookups = `read_random_exp_range` (0 = uniform; 5, 10, 20 = increasing skew)

Cache Configurations

LRU: `cache_type=lru_cache`, `cache_high_pri_pool_ratio=0.5`

HCC: `cache_type=auto_hyper_clock_cache/ fixed_hyper_clock_cache`

Always set `cache_type` explicitly (db_bench default is HCC).

Always set `cache_numshardbits` explicitly (`shards = 2^cache_numshardbits`; defaults differ: LRU uses 64 shards, HCC uses 1 at 32 MB).

Block Cache size = 64 MB (`cache_size`)

`lru_cache, auto_hyper_clock_cache, fixed_hyper_clock_cache`

`cache_high_pri_pool_ratio` (vary between 0, 0.5 & 1)

Other

Fixed seed (`seed=42`) so both caches see the same keys

Threads never more than the number of CPU cores

Repeat each run 3 times

`statistics=1, histogram=1`

Metrics

(i) Throughput (ops/sec), (ii) Read latency (p50, p99, p99.9), (iii) Block cache miss ratio, split by data, index, filter (`rocksdb.block.cache.{data,index,filter}.{hit,miss}`)

Experiments

E1 Cache size: 32 MB, 128 MB, 512 MB, 2 GB. Threads = 1, uniform and skewed.

(At 32 MB even the index (~60 MB) does not fit; at 128 MB all index + filter fits;)

E2 Thread scaling: 1, 2, 4, 8, ... up to #cores. Cache = 2 GB (fully cached, warmed up). Uniform and skewed.

E3 Shards x Threads: LRU with 1, 4, 16, 64 shards vs HCC with 1 and 4 shards. Cache = 1 GB, max threads.

E4 Skew: `read_random_exp_range = 0, 5, 10, 20`. Cache = 64 MB, threads = 1 and max.

Notes

Warm-up: run lookups once before measuring. Statistics counters are cumulative and `resetstats` does not clear them, so subtract the warm-up counts.