rocksdb相对于leveldb的改进

1. 性能上：

- 多线程compaction
- 多线程memtable写入
- 减少DB互斥锁持有
- 优化了基于level的compaction方式以及全局compaction方式
- 前缀布隆过滤器
- memtable布隆过滤器
- 单个布隆过滤器覆盖整个sstable
- 写锁优化
- 提升了Iter::Prev()操作的性能
- 在进行跳表搜索时调用更少的比较器
- 使用huge page申请内存

2. 用法特点上：

- 列族 column families
- 事务以及根据下标的批量写
- 备份以及检查点
- Merge操作符
- compaction过滤器
- Rocksdb Java
- 持久化cache
- 在自动进行compaction的同时也可以执行手动compaction
- bulk loading
- 前向迭代器和尾部迭代器
- single delete
- 范围删除文件
- pin iterator key/value

3. 数据结构和格式上：

- 在memory-only的场景下使用plain table
- memtable有基于向量和基于哈希两种格式
- 基于时钟的cache
- 可插入的information log

- 用blob注释事务日志写入（用于复制）

4. 协调性上：

- 比例限制 （rate limiting）
- 可调整的减速和停止阈值
- 有参数可使所有文件保持open
- 有参数可保留index block 和 布隆过滤器block在block cache中
- 多种WAL恢复模式
- Fadvise hints for readahead and to avoid caching in OS page cache（可协调预读、防止在OS页面缓存中缓存）
- 可以选择将L0文件的index和布隆过滤器块放在内存中
- 更多的压缩类型：zlib、lz4、zstd
- 压缩字典
- 校验和类型：xxhash
- 不同的层大小增大方式以及每层的压缩类型

5. 管理能力上：

- statistics
- Thread-local profiling
- 命令行工具中的更多命令
- 用户定义的table properties
- 事件监听器
- 更多DB性质
- 动态参数调整
- 从string或map中获取参数
- 持久化参数到option files中。

> 来源: RocksDB Features that are not in LevelDB

compaction操作对rocksdb性能的影响

- 单线程情况下：cpu执行compaction就无法处理读写请求。
- 多线程情况下：L0到L1的compaction操作无法与其它层的compaction操作并行

rocksdb的compaction策略

（compaction主要是重新排序数据、清理过期数据和重复数据等）

- universal compaction(tired compaction的一种)
  - 每一个run包含了一段时间内加入的数据
  - 不同的run的时间段不会重叠
  - 只对相邻时间段的run进行compaction
  - 当run累积到一定的数量N时触发compaction
- leveled compaction