* LSM-tree由于其内存表的结构特征，相比InnoDB引擎可能积攒更多的待checkpoint的WAL日志（这个问题在各个表读写不均衡时情况更为明显），为了提升崩溃恢复时的系统启动速度，X-Engine支持在Crash Recovery时并行的回放WAL日志。

X-Engine对比Rocksdb，更偏向读优化，因此它具有更浅的层次结构,而其整个compaction系统的设计均面向这个浅层次的LSM-tree架构做优化。

* X-Engine磁盘数据部分包含持久化的L1和L2，以及零时存在的L0层。其设计理念是：L2层为全量冷数据，L1层为被频繁读写的热点数据。而L0层是在compaction来不及时，暂存内存表转储的数据。在系统负载较低时，L0会直接与L1层合并落盘。
* 由于数据按2MB的Extent粒度存储，L1层的数据与L2层的compaction可以基于读写频次触发的调度策略。这样可以让热点数据尽量保留在L1层，提升读盘时的IO效率。相关技术可见[X-Engine如何精准的冷热分离](https://mp.weixin.qq.com/s?__biz=MzU0NDcwNjQxNA==&mid=2247483977&idx=1&sn=3da92a85a20e4c18c7dd9365aa3469bf&chksm=fb795848cc0ed15e23fe9ad95a126aaab590463bb78fe17fa912437d804688c631b8063baf11&token=992153913&lang=zh_CN" \l "rd)
* LSM-tree由于其架构特点，在Memtable刷盘或者Compaction时，会触发版本变更导致的Cache主动失效，此过程会导致性能的抖动。X-Engine支持在Memtable刷盘以及Compaction之后，根据内存数据的访问统计，进行主动的Cache回填，尽最大可能降低Cache抖动。
* X-Engine的数据分散在最大为1GB大小的多个数据文件中，所有的数据文件按2MB大小切割为Extent，Extent是空间分配的基本单位。Extent会再切分为不同的Page，Page为不定长，所有的Page均默认压缩存储。通常情况下相对InnoDB,X-Engine可以获得3~5倍的存储空间优势。
* 完善的Information\_schema表，包含内部各组件的内存消耗分布，每张表的磁盘空间消耗，后台任务(compaction/flush/空间回收)运行状态统计等。
* QueryTrace功能，可以将一条SQL在运行过程中的每个阶段的耗时分布详细记录下来，包括加锁解锁时间/加锁解锁次数/IO读写次数/IO耗时。此工具可以辅助快速定位慢查询的瓶颈点。

X-Engine团队前后6年的努力，才取得上述些许进展和成果。  
在此过程中我们深刻认识到，开发一个功能完备性能优异的存储引擎的难度以及所需要的时间。未来我们会在LSM-tree存储引擎方向上继续探索和研究。