**FlashKV: Accelerating KV Performance with Open-Channel SSDs**

非典型NSC

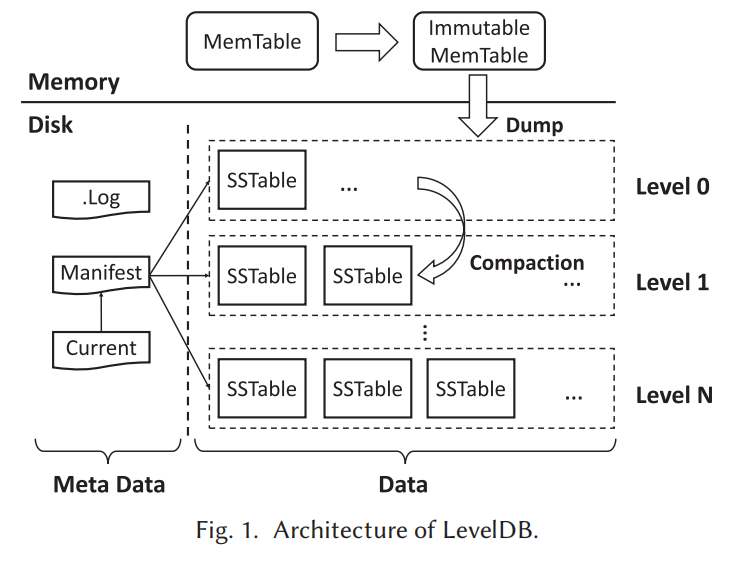
（仅“完全利用SSD并行性”这一点与“非匹配带宽”这一NSC动机相对应）

经典的基于LSM树的KV应用LevelDB：

query写入

MemTable满时

为后续操作生产新的MemTable



LSM树（log-structured merge tree）通过缓存更新并批量卸载（dumping）以将随即更新转换为顺序写；

compaction:某层容量满时，需将数据条目合并（merging）至下一层的操作；

dumping和merging（即SSTable）均采用日志结构更新，元数据采用就地更新

当前有效的manifest

compaction后

的有效SSTable

query写入时记录以防崩溃

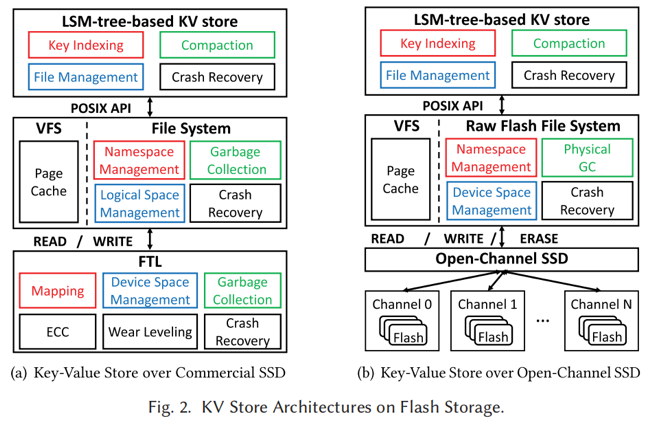
各层容量不同

递增

（批量）

商用SSD上基于LSM树的KV存储存在以下问题：

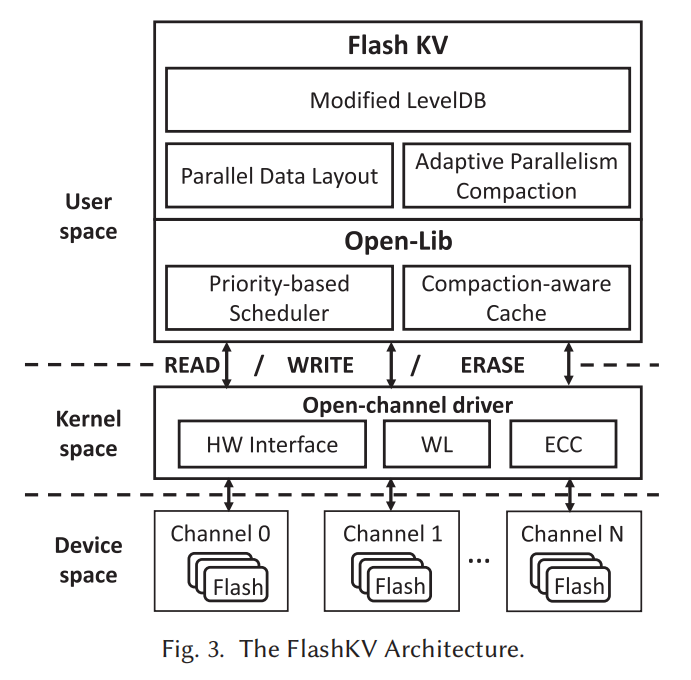
1. 基于LSM树的KV存储/FS/FTL间的功能重复（图2(a)相同颜色的模块功能重复），引起了严重的写放大及性能降低；



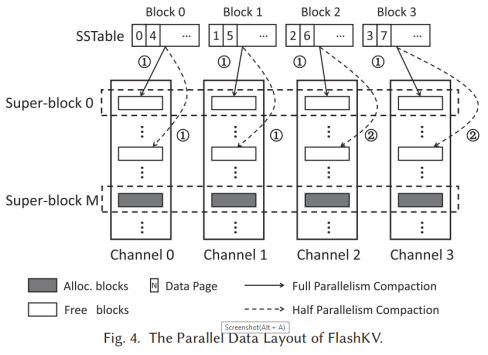
1. SSD的内部并行性未被充分利用（因为FTL将底层SSD抽象为块设备，则上层KV/FS不可见其内部特性）；
2. KV存储的特性未被利用（如：对于不同的数据文件/前后台线程，访问模式不同）。

基于2），考虑到OCSSD的开放性，将KV存储直接应用于OCSSD上，发现KV/FS间仍有功能重复且仍未利用KV特性。（此I/O栈为通用目的所设计，并非最适合KV→**调整IO栈结构**）

【FlashKV】



1. 利用OCSSD的开放性，绕过FS/FTL以减少功能重复（图3中用户空间的Open-Lib代替完成原内核空间的FS的功能，现内核空间Open-channel driver仅实现硬件相关的功能）；
2. Parallel Data Layout



因为绕过FS/FTL，FlashKV直接管理存储空间。FlashKV中有两种数据：元数据/SSTable。

对于元数据：存储于非易失区域，占空间小（日志文件一旦更新即丢弃之前的版本）；

对于SSTable：Parallel Data Layout。采用superblock方式组织存储空间，SSTable大小与superblock大小一致，一个superblock存储一个SSTable（这样能节省很多开销：仅需维持SSTable文件名与superblockID的映射，省略了原FS的索引节点信息即FTL的页/块映射信息）。且由于superblock的构成（各通道的相同偏移的块），其内部数据也存在并行性。**（通道的并行性→进而有superblock内部块的并行性→即superblock的并行性）**

1. Adaptive Parallelism Compaction

考虑到SSD读写性能的不平衡，对于compaction自适应的采取两种方式：

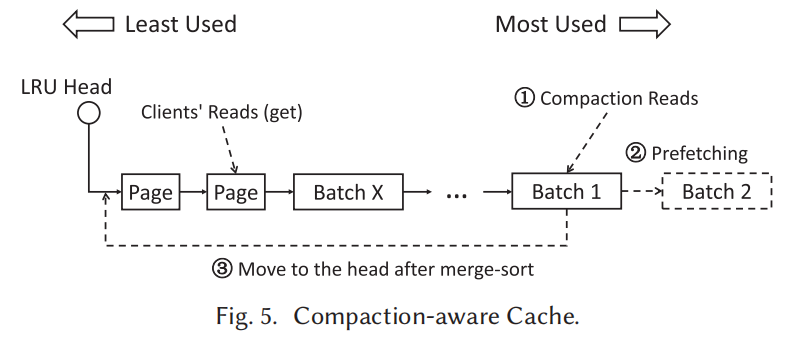
对于写密集负载：对于合并后的SSTable的写入采取**全**并行以尽快完成compaction（图4实线箭头）；

对于读密集负载：对于合并后的SSTable的写入采取**半**并行以避免阻碍读操作（图4虚线箭头(分①②两步完成)）。

为加快卸载避免堵塞（若某层SSTable达阈值则会堵塞），compaction还采用了**双线程**，即在compaction开始前，取无重复key的SSTable，分双线程进行compaction。

1. Compaction-Aware Cache

将读操作分为两类：①客户发出的get操作；②compaction线程发出的compaction读。前者尺寸小，key可能越层，不适合预取；后者尺寸大且批量卸载读取，读取的数据在同一SSTable中，key不会越层，适合预取。



LRU头会丢弃无用数据以维持LRU可用空间

FlashKV在读请求上加上tag（内含元数据），tag可帮助区分读操作类型。

1. Priority-Based Scheduler

将IO操作分为**前台请求**（客户请求get/put）及**后台请求**（compaction请求）。

为保证用户体验，前台请求优先级高于后台请求，相同优先级时，读请求优先级高于写请求（写延迟高）。擦除操作优先级取决于空闲superblock数量，足够时优先级低，反之优先级高。

其中，1）针对问题1），2）针对问题2），3）、4）和5）针对问题3）。