TRIM

一、 FE 与 FTL 交互：

FE 对于 trim 命令的处理：

1. 对于小范围的 trim 命令处理，FE 直接将 trim 命令转为写命令即可；

2. 对于大范围的 trim 命令处理：

1）对原始 trim 进行处理，

一个 trim 命令内 range 若有互相重叠的情况，需要对原始 trim 命令进行合并操作；

range 中若包含非 4K 对其的头或者尾，需要将头尾进行拆分后，将对齐部分的 range 下发给 FTL，非对齐部分作为写 IO 处理。（目前应该不涉及非对齐的 场景）

2）trim 命令下发前，若 CACHE 中存在写 node 且还未下发给 FTL 的，需要将与 trim 范围有冲突的 强制 flush 到 FTL（简单点可以直接 flush 掉所有 dirty 的 node）

3）cache 需要记录所有的未完成的 trim 信息，当 read 命令进来后，若 read 与 trim 有冲突，则需要将 read 命令 pending 住。

3. trim 命令交互：

trim 命令由 FE 生成 trimNode，其结构与 writeNode 一致：

```
typedef struct _FeReqTrim_t
{
   U08 u08Lun;//2
   U08 u08Fua;//3
   U08 u08Rev;//4
   IDXList_t SubReqList;//12
```

```
    U08 *pu08DataBuffer;//16
} FeReqTrim_t;
```

通过 SubNode，将具体的 trim 的 range 范围传递给 FTL。

```
typedef struct _FeSubReqNode_t
{
    U32 u32Lba;
    U32 u32SectCnt;//each sector is 4K
} FeSubReqNode_t;
```

trim 命令放入 FE2BE 的 write_queue 中，trim 命令完成后，放入 write_cmpl_queue 中，执行优先级与 write 命令一致。

二、 FTL 处理 TRIM 流程

FTL 处理 trim 命令的思路是将 trim 范围内的 mapping 进行更改，故 trim 命令需要触发 FTL 的 mapping merge 流程。具体处理过程如下：

1. 获取 trim 命令：

从 write_queue 中获取 trim 命令，即，在获取当前 trim 命令的时候，一定能保证 trim 命令之前的 write 命令都已经下发，若有之前的

write 命令 pending， 则按照当前的处理 write_queue 的逻辑，不会获取到后面的 tirm 命令的，保证了 trim 和 write 命令的先后关系。

需要保证在处理 trim 命令之前，已经处理完了所有 FE2FTL 的 read_queue 中的命令，故 FTL 在进行调度时，发现当前命令为 trim 时，需要

先处理掉所有的 read 命令。

若出现当前 trim 命令时出现 pending， 也不能处理后续的 write 命令。

2. 处理 trim 命令：

1). 生成一条 trim 命令的 record，用于记录当前 trim 命令时刻， host 写入的数据的情况，record 结构：

```
typedef struct _TrimCkpt_t
{
    SpbId_t Spb;
```

```
    U32 Wptr;
    U32 SpbSn;
    U32 LuSn;
} TrimCkpt_t;
```

对于每一个命令， FTL 需要记录当前的 HOST 写入的 ASPB 的 SpbId，SN 以及 write_ptr，记录当前 LU 的 SN。

该记录用于 trim 与读写数据冲突时，比较新老数据的依据。

当 merge mapping 或者 read 的时候，已知 LAA 命中在一个 trim 命令的范围内，则比较当前 mapping 中的 PAA 与 trim 的新老，

具体的比较方式为： 当前 PAA 的 SpbSn 与 TrimCkpt_t 中的 SpbSn 相比，谁更小则谁为老数据；若 SpbSn 相同，则比较 PAA 对应

的 PageOff 与 TrimCkpt_t 中的 Wptr, 谁更小谁为老数据。

2). 将 trim 命令入 trim_proc_queue：

FTL 将所有收到的 trim 命令入 trim_proc_queue, 入队完成后，即可返回，让 FTL 从 FE2FTL 的 queue 中继续处理其他的 CMD。

同时，FTL 启动处理 trim queue 的任务。

入队时，根据当前 trim 命令的大小，将小范围的命令放入入 high_queue ，大范围的命令放入 low_queue。

3). 处理 trim_proc_queue:

优先从 high_queue 中获取 trim 命令进行处理，若 high_queue 为空，则从 low_queue 中获取。

支持当正在处理一个 low_queue 中的 trim 命令时，有高优先级的 trim 命令入队。则可以打断正在执行的低优先级的 trim 命令，先执行

高优先级的命令；高优先级命令处理完成后，若没有新的高优先级命令，则返回之前打断的低优先级命令任务继续执行。

4). trim 超时处理：

从 trim 命令入队开始，开始记录 trim 命令处理时间， 当 trim 命令在设定的阈值时间内没有完成，则需要提前返回 trim 命令至 FE。同时

该 trim 信息需要持久化到 NAND 上进行保存，防止在返回 FE 后，发生了掉电，导致 trim 信息丢失。（保存到 NAND 上的命令存在一个上限，

即只支持设定阈值的 trim 命令支持超时返回。若已经达到给阈值， 则 trim 命令需要等待处理完成后，才返回 FE）

trim 的持久化信息如下：

```
typedef struct _TrimInfo_t
{
    TrimCkpt_t ckpt;
    U16 LaaCnt;
    U16 RangeCnt;
    u64 lba[MAX_RANGE_PER_TRIM];
    u32 count[MAX_RANGE_PER_TRIM];
} TrimInfo_t;
```

trim 的持久化信息存储在 sysLu 中，位于 SLUT 之后。

、5). trim merge mapping:

FTL 处理 trim 命令的主流程。 需要将 trim 命令的 range 中包含的 mapping 修改为 UNMAP_PAA，用以标志该映射项被 trim 掉了。FTL 中

存放最新的 mapping 的地方为 RLUT，则需要启动 merge 流程将 RLUT 和 trim 一起进行处理，否则若单独处理 trim 的 merge， 当 trim 命令

完成后，无法获知 RLUT 中的记录与 trim 命令的先后顺序，造成数据不一致。

基于现有的 RLUT 的 merge 流程（即 convert 流程），增加一种触发 merge 的条件，即 trim 命令触发 convert。

需要修改当前的 convert 流程实现，细节如下：

a. convert 运行后，check 下是否有需要处理的 trim range 信息。若有，则优先根据 trim range 中包含的 L2PP 进行加载， 然后再根据 RLUT

当中的 LAA 进行 L2PP 加载。

b. 对加载起来的 L2PP 进行 merge 操作时，先 merge RLUT 中的 mapping，再 merge trim range。 merge trim range 时，需要根据 trim 的 ckpt_info

来 16 进行判断 mapping 的新旧。

c. 在 convert 流程完成时，需要将当前处理的 trim 的信息记录到 UserLuDesc 中，记录的信息结构体如下：

```
typedef struct _TrimFence_t
{
    U32 Sn;
    u32 RangeIdx;
    LbaRange_t Range;
} TrimFence_t;
```

Sn 为当前 merge 的 trim 的 LuSn，用于上电重建过程中判断是否是已经完成的 trim 命令。

RangeIdx 为当前 trim 命令的哪个 range；

Range 为当前正常处理的 tirm range 范围。

需要修改当前的 RLUT 流程实现，细节如下：

a. 需要支持 RLUT 在没有 FULL 的情况下进行 merge 操作，依然采用先将 RLUT flush 后再进行 merge，只是在 merge

完成后，当前 RLUT 不进行回收，继续作为 BUSY 的 RLUT 进行使用。ASPB 上需要记录当前的 RLUT 更新的位置（updatePtr）。

b. RLUT 在 merge 的过程中支持插入新的 mapping。同时， convert 需要记录一个 boundary，在 merge RLUT 的时候，超过

boundary 的 mapping 不进行 merge 处理。

c. RLUT 支持进行多次 merge 操作，以当前 ASPB 的 updatePtr 作为起点，以当前的 RLUT 的 fillIndex 作为终点进行 merge

操作。

d. 若发现 updataPtr 与当前 RLUT 的 fillIndex 相等，则当 trim 触发 convert 时，无需进行 RLUT 的 flush 和 merge 操作。

6） trim info 重建

当 FTL 重新上电时，需要对 TRIM 的持久化信息进行重建，已完成之前掉电后为成功执行的 trim 处理，重建过程如下：

a. 读取存储在 sysLu 中的 tirm info；

b. check 每笔 trim info 是否需要继续处理：

因为该笔 trim 命令可能已经处理完成，但是持久化信息还在。故需要将 trim info 中记录的 luSn 与 UserLuDesc 中的记录的 trim

的 SN 进行比较，若小于后者，则不需要恢复；若相等，说明是真正执行的这一笔 trim 信息，则比较 range 范围来确定恢复的 range；

若大于后者，说明是还未执行的 trim 命令，需要恢复。

c. 将恢复出的 trim info 重新生成 trim 命令，进入 trim 的 queue 中进行后续的处理。

7）trim 与读写命令交互

a. trim 与 write：

不需要特殊处理，trim 命令和 write 命令位于同一个 FE2FTL queue 中，按照顺序处理命令即可；

b. trim 与 read：

当 trim 命令没有达到 FTL 的时候，需要 FE pending read 命令；

当 trim 命令达到 FTL 后，FTL 通过 trim queue，记录了 trim 的信息，read 命令到达 FTL 后，需要比较

mapping 和 trim 的新旧，从而返回正确的 read 数据。

，