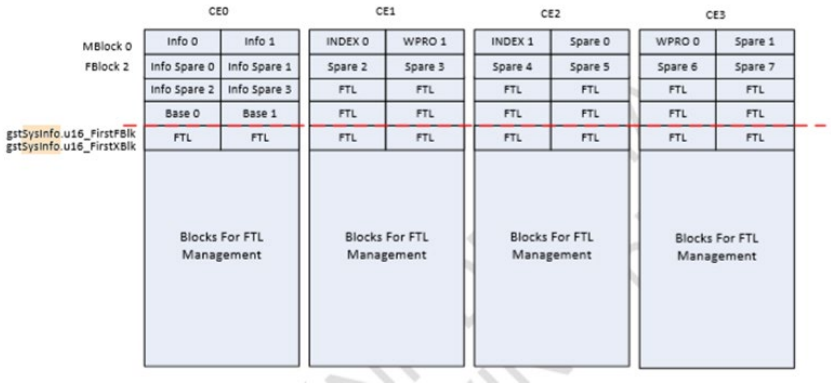
空间管理

1. 物理空间管理：

使用 pool 管理不同物理空间，NAND 提供的空间按照 FTL 使用上的区别，分为：

1） ROOT_BLK_POOL:管理一些 ROOT meta 以及 specail 的 user data(boot LUN data， rpmb LUN data、health data 以及 FTL boot info 等数据)，该 POOL 以 BLOCK 作为粒度来分配空间。



（图中 index/wpro 取消，由 FTL 统一管理）

该 POOL 空间来自于前 N 个 NAND BLOCK,待 system info block 分配完成后，剩余的 BLOCK 就被 FTL 管理到 ROOT_BLK_POOL 中。具体空间划分为：

a) FTL_MGR_BLOCK:

存放 FTL 管理 boot block 的信息，控制 boot_blk_pool 中所有 block 的分配，是 FTL 的根节点。

b) FTL_META_BLOCK:

用于存放 FTL 的 ROOT meta 信息：

c) DEV_INFO_BLOCK:

用于存放 device 信息相关的 block，包含 health info 等设备相关的数据信息

d) EVLOG_BLOCK:

存放 evLog 的 block

e) WK_LUN_BLOCK:

存放 BOOT LUN 和 RPMB LUN data 的 block

f)　　OTHER_BLOCK:

冗余 BLOCK,用于上述 BLOCK 产生坏块时进行替换。

注：

除 evLog block 外，其他的各分区有双备份或者多备份的策略，并且各自具备一定数量的 spare block。

BLOCK 的分配方式：

ROOT_BLK_POOL 中维护一张 block 的分配表 root_blk_bitmap，将 rom 的 info block 标记为 1， 然后剩余的为 0 即可分配 bit。其中 FTL_MGR_BLOCK 从前完后申请，其余的 BLOCK 从后往前申请，申请出 BLOCK 后将相应的 bit 置位，root_blk_bitmap 需要持久化保存。

2）SLC_SUB_SPB_POOL

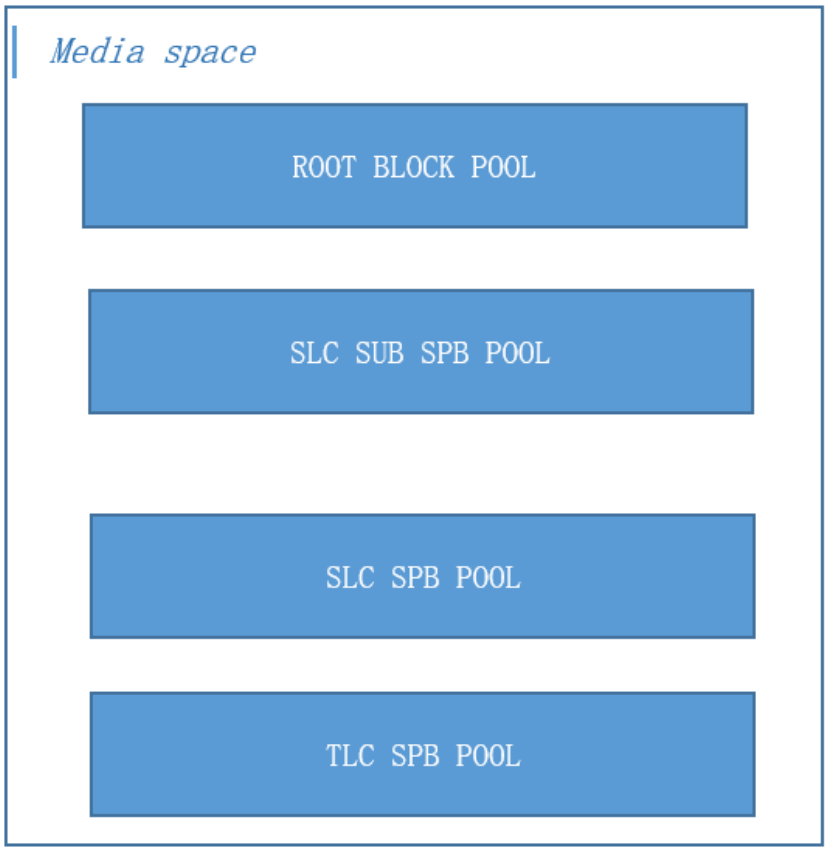
该 POOL 提供以 SPB 作为分配粒度的 SLC 空间，支持 SPB 的 BLOCK 数配置（可以为最大并发度的 N 分之一）。

3）SLC_SPB_POOL

该 POOL 提供以 SPB 作为分配粒度的 SLC 空间，SPB 长度为最大并发度，不可以配置。

4） TLC_SPB_POOL

该 POOL 提供以 SPB 作为分配粒度的 TLC 空间，SPB 长度为最大并发度，不可以配置。支持 dynamic SLC, 将满足条件的 TLC 临时作为 SLC 使用，支持 dynamic SLC 占比的配置。



2. 空间地址描述：

1） WeruId & SpbId:

WeruId: 用于标识物理 superBlock， 从 block0 开始，将整个 media space 顺序编码。

SpbId: FTL 使用的逻辑 superBlock 编号，spb 与 weru 存在一对一（普通 pool）和多对一(sub_slc_pool)的关系。

2) Lpda:

Lpda 用于描述 SPB 中的逻辑地址，在一个 SPB 中顺序编码。Lpda 的构成由

SpbId + offset 组成：

Lpda = spbId<< block_shift | offset;

同时，FTL 内部管理的所有 mapping 都是 lda 到 Lpda 的映射。

3) Wpda:

Wpda 即 wearLine pda，用于描述在物理 SPB 上的地址，是 FTL 与 BE 交互的地址信息。

Wpda = WeruId<< block_shift | offset;

即 Wpda 这里看不到 subSpb 的存在。

4）地址转换：

FTL 与 BE 交互时，需要完成 Lpda 到 Wpda 的地址转换。

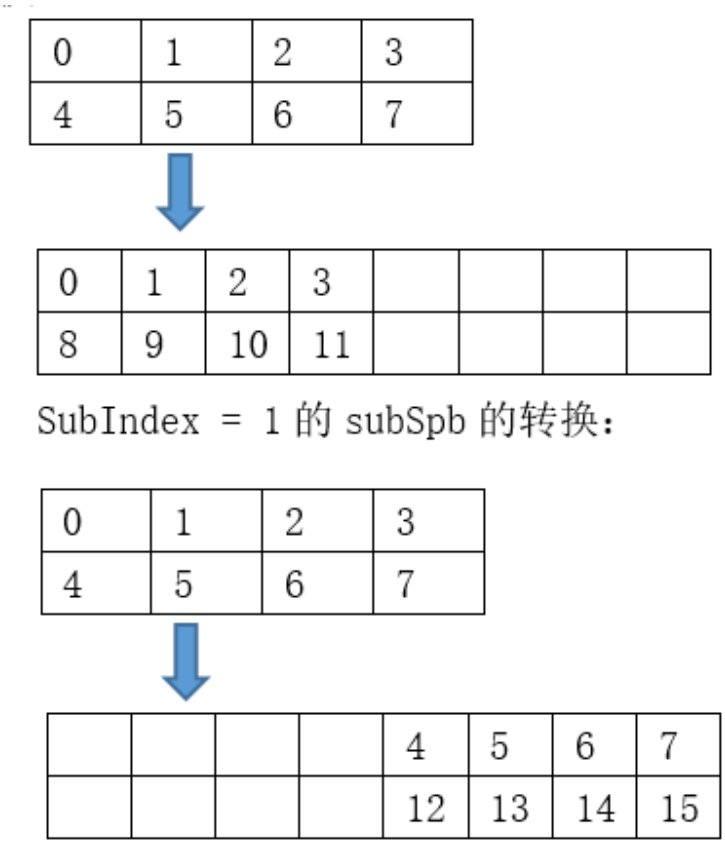
a. 对于非 SUB_SLC_POOL 的 slc spb:

这里的 spb 与 weru 的大小一致，只需要转换 BLOCKID 即可。

b. 对于 SUB_SLC_POOL 中的 slc spb:

需要根据其 subIndex 来看是位于物理 spb 上的哪一段，再结合 subSpb 上的 offset 来进行转换。

举例说明：假如物理 spb 分为 2 个 subSpb，那么 subIndex = 0 的 subSpb 上的转换：



c. 对于 TLC_SPB_POOL 中的 spb:

需要根据不同的 nand 的 program order 来决定转换逻辑：（已一个 4plane, 2 die 的 tlc 为例）

Die0:

0	1	2	3
4	5	6	7
8	9	10	11

0	1	2	3				
8	9	10	11				
16	17	18	19				

Die1:

12	13	14	15
16	17	18	19
20	21	22	23

				4	5	6	7
				12	13	14	15
				20	21	22	23

3. 逻辑 LUN 空间管理

在物理空间之上，基于不同的逻辑需求，划分出几个逻辑 LUN：

1）LUN 类型：

a. USER_LUN

提供用户空间的 LUN，用于存储用户数据,最小粒度为 4k。

b. MIDDLE_LUN

存放 USER_LUN meta 信息的 LUN,主要包括 USER_LUN 的 12p, p21，最小粒度为 4k。

c. SYS_LUN

存放 MIDDLE_LUN 的 meta 的 LUN，主要是 MIDDLE_LUN 的 l2p，最小粒度为 4k。

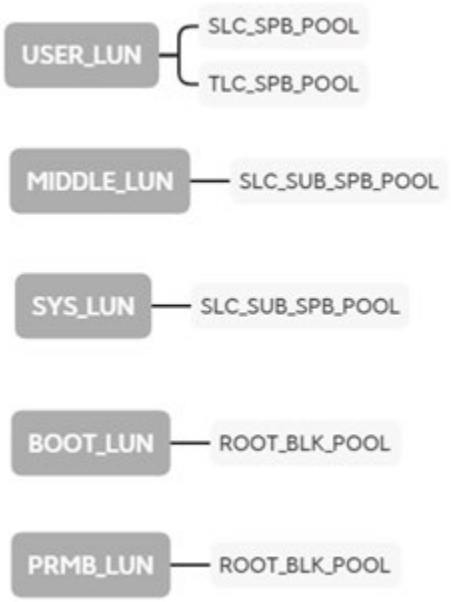
d. WK_LUN

存放 BOOT_LUN 和 PRMB LUN 的 data。

（middle_lun 不是必须的，当空间比较小的时候，可以不需要 middle_lun）

2） POOL 空间分配：

不同的 LUN 可以根据自身需求从不同的物理 pool 中获取空间，当前的分配策略为：



a) force_slc 模式：

该模式下，user data 首先需要写入 SLC 中，若 SLC 空间不足，需要 gc SLC 到 TLC 后继续使用 SLC。

在该模式下，又可以配置是否支持 dynamic SLC。配置了 dynamic SLC 后，TLC 中的部分或者全部的 SPB 在满足一定条件下可以转为 SLC 使用，在初始写入时，转换比为 100%；后续转换比为 dynamic SLC 的配置数据。

b) normal_slc 模式：

该模式下，优先使用 SLC_CACHE 即 SLC_SPB_POOL 中的 SPB, 当 SLC_CACHE 中的 SPB 耗尽后，使用 TLC direct write。同样，可以配置 dynamic SLC。同 a 中的方式一致。(初次写都要为 SLC)

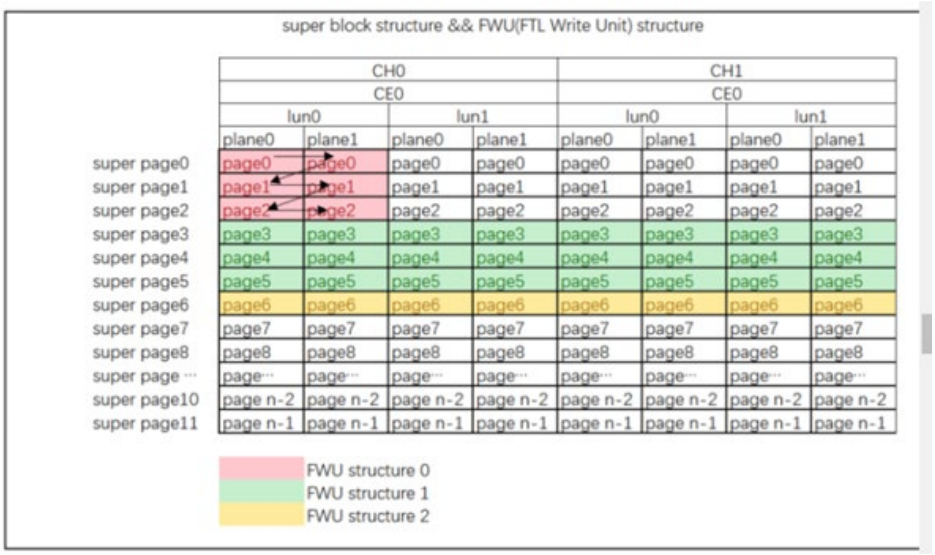
MIDDLE_LUN 以及 SYS_LUN 使用 SLC_SUB_SPB_POOL 中的 SPB,根据具体的 NAND 配置 SUB SPB 的大小。

BOOT_LUN&RPMB_LUN 使用 ROOT_BLK_POOL 中的 BLOCK 空间，该 POOL 中的空间采用预留的机制，不需要动态申请和释放。

3） SPB 内的空间分配

a. USER_LUN

使用 FWU（FTL write unit）来封装不同的 program 要求。一个 FWU 会记录对应 program 空间的起始位置，结束位置与大小，也就是一堆 PDA 的集合，对外呈现一段连续的逻辑空间。其它业务在向一个 FWU 写数据时，完全可以把它当成一段连续的数据 buffer，而不需要考虑最终数据的 Layout。真正决定怎么去放一个 FWU 数据由专门的逻辑根据不同的 NAND 以及当前 SPB 的坏块情况等因素来确定其具体大小。



b. MIDDLE_LUN & SYS_LUN

这两个 LUN 上按照 page 对齐的规则写入（实际下刷 meta 时也会尽力按照 muti-plane 方式，但受下刷数据量影响，不做强制 mutil-plane）。

c. BOOT_LUN & RPMB_LUN

这两个 LUN 基于 BLOCK 管理，按照 page 写入。

4. SPB 规划:

对 spb 按照 EC 值进行排序，偏大的优先考虑设置为 SLC，小的为 TLC。对都使用 SLC 的 LUN， 按照 EC 由小到大，分别分配给 SLC_SUB_SPB_POOL 和 SLC_SPB_POOL；ROOT_BLOCK_POOL 固定分配前几个 SPB。

1） 预留出 ROOT_BLOCK_POOL 中的 BLOCK：根据需求，预留出 3-4 个 SPB 的 BLOCK 数，分配到 ROOT_BLK_POOL 中。

2） 根据设定的 user_data 数据量大小，计算出 mapping 的大小（l2p 和 p2l），得到 middle_lun 需要的空间和 sys_lun 需要的空间，然后根据各自的 OP 需求，得到所需的 SPB 个数分配到 SLC_SUB_SPB_POOL 中。

3）根据设置的配置参数[A1] 和 SLC 模式，得到 user_lun 中 slc_cache 的大小，得到 SLC_SPB_POOL 的 SPB 个数；最后剩余的 SPB 分配到 TLC_SPB_POOL 中。此时需要 check TLC_SPB_POOL 中的有效空间（除去坏块）是否满足 user_data 的容 量以及 OP， 若不满足则会导致 pool 的初始化失败，尝试调整各 LUN 的 op 后重新进行初始化，若仍然不满足，则开卡失败。

5. 坏块管理：

FTL 对于坏块的处理，支持初始替换和动态替换，同时在无法替换后，支持带坏块继续使用。

1） 坏块替换：

a. 替换范围： 除 ROOT_BLK_POOL 以外的其他 POOL 的 SPB。

b. 替换限制： 达到替换设置的最大替换次数后，后续坏块不再替换。

c. 替换时机： 在初始开卡时执行初始替换，后续新增坏块时执行动态替换。

d. 替换逻辑： 只执行同 plane 替换

初始替换：按照 SPB 坏块个数从小到大进行排序，从头尾开始进行替换，即选用尾部上的 SPB 上的 BLOCK 去替换头上的 SPB， 替换次数达到最大替换次数，或者替换到无法再执行时退出。

动态替换：在未达到最大替换次数时执行动态替换。不预留专门提供替换而存在的 SPB， 当产生新增坏块时，从 TLC_SPB_POOL 中获取一个 free 的包含坏块且能够提供替换块的 SPB 作为坏块提供方，执行替换后，坏块提供方的 SPB 在坏块总数未达到不能使用的阈值时，可以继续供 TLC_SPB_POOL 使用。

e. 设置最大可用坏块个数阈值 ：当某个 SPB 的坏块个数达到这一阈值时，整个 SPB 将不再使用。

f.替换信息存储：预留最大替换次数个 node 内存，将该信息固化到 BOOT_BLK_POOL 中 ftl root block 当中进行存储和恢复。（用 spb arry 方式）

g.坏块信息表： FTL 中只保存一份坏块信息表，即替换后的坏块信息表，不记录原始坏块信息表。

2） 使用坏块 SPB：

当替换不能再执行时，SPB 中就会包含有坏块。FTL 支持包含坏块的 SPB 继续使用。在写 IO path， GC ，P2L 表以及重建等与 SPB 空间布局相关的操作都需要考虑坏块的存在。

3）大量坏块的处理：

当产生大量无法替换的坏块后，需要 check FTL 是否需要进入 read_only 状态：SYS_LUN & MIDDLE_LUN & USER_LUN 的空间是否已经足够。

6. GC 策略：

每个 POOL 各自有自己的 GC 策略：

1） ROOT_BLK_POOL: 由于基于 BLOCK 管理，没有 GC 逻辑，而是使用 spare BLOCK 的方式进行 BLOCK 的乒乓使用。

2） SLC_SUB_SPB_POOL:

按照 free spb 设定 5 个水位:block/gc_start/gc_end/bgc_start/bgc_end。其中 block 为完全 pending 上层 IO; gc_end >= gc_start + 1; bgc_end >= bgc_start + 1; bgc_start > gc_end。

3） SLC_SPB_POOL：

同 2）中设定 5 个水位。同时根据不同是 slc cache 模式设定不同的值。

a. force_slc：

5 个阈值为：0/2/3/MAX/MAX (MAX 是指全部的 BLOCK)

b. force_slc + dynamic_slc:

该模式下 SLC_SPB_POOL 将 GC 的控制权交由 TLC_SPB_POOL 执行。

c. normal_slc: (direct tlc mode)

5 个阈值为：NA/NA/NA/MAX/MAX (NA 是指无效水位，即该阈值不发生作用)

d. normal_slc + dynamic_slc:

该模式下 SLC_SPB_POOL 将 GC 的控制权交由 TLC_SPB_POOL 执行。

4） TLC_SPB_POOL:

同 2）中设定 5 个水位。

a. Force_slc + dynamic:

该模式下，根据 slc_cache + dynamic slc 中 free 的量来启动 slc GC:

5 个阈值： 0/2/3/MAX/MAX

同时，还需要根据 free 的 tlc spb 个数设定 TLC 的 gc 水位：

5 个阈值： 3/5/7/8/10 (暂定)

b. Normal_slc+ dynamic:

slc gc：NA/NA/NA/MAX/MAX (包含 slc_cache 和 dynamic slc)

tlc gc: 3/5/7/8/10 (暂定)

slc gc 优先级高于 tlc。

c. 非 dynamic:

只有 tlc gc: 3/5/7/8/10　　(暂定)