FTL Write

一、Backgroud

2754 的空间管理将分为 user lu\middle lu\sys lu\boot lu\rpmb lu，其中 middle lu\sys lu 主要 FW 内部自己的管理空间，而 host 会直接访问的是 user lu\boot lu\rpmb lu，这些空间的写都将经过 data cache 放入其中的 cache node，同时 data cache 依靠自己的机制和策略，将 cache node 以 FEReqNode 的形式放入 queue。cache 当前的设计是将有两个 queue：分高优先级和低优先级，高优先级存放 read 和 HPL，低优先级存放 write 和其他。而 FTL 最终将从这两个 queue 上摘下 FTLReqNode，进行不同的读写等操作处理。

二、FTLReqNode Queue Process

- 由于目前计划设计是两个 queue 将读写分开，期望读比写更高优先级的被调度执行，FTL 将无法保证读写的互斥情况（读比写先执行，读到老数据？），所以需要 data cache 去做读写互斥。
- Queue 里面放入了 HPL 和其它操作的 node，这些 node 与 IO node 的处理优先级？
- 针对 write node 的管理：

目前由于一个 write queue 里面还放入了其它 node，导致 FTL 还需遍历 queue 把所有 write node 摘出来管理，设计是将 queue 中所有 write node 全部取出用链表管理。或者 cache 进行优化设计，让 IO 读写的 queue 更纯粹？

- write node 的组装：主要是针对 user lu，boot lu 和 rpmb lu 是 SLC spb，最小写单元就是一个 page，而 user lu 的 SLC 和 TLC 的 PU 考虑到 die 大小、坏块、raid 等情况，其大小并不是固定的，所以需要将 write node 组装成符合 PU 的大小。比如一个 write node 32K size，而 TLC 的 PU 是 192K，那一笔 FTL 请求就需要取 6 个 write node。

	CH0/CE0/PL0	CH0/CE0/PL1	CH0/CE0/PL2	CH0/CE0/PL3	CH1/CE0/PL0	CH1/CE0/PL1	CH1/CE0/PL2	CH1/CE0/PL3	CH0/CE1/PL0	CH0/CE1/PL1	CH0/CE1/PL2	CH0/CE1/PL3	CH1/CE1/PL0	CH1/CE1/PL1	CH1/CE1/PL2	CH1/CE1/PL3
Page 0	D0	D1	D2	D3	D12	D13	D14	D15	D24	D25	D26	D27	D36	D37	D38	Parity 0
Page 1	D4	D5	D6	D7	D16	D17	D18	D19	D28	D29	D30	D31	D39	D40	D41	Parity 1
Page 2	D8	D9	D10	D11	D20	D21	D22	D23	D32	D33	D34	D35	D42	D43	D44	Parity 2
RAID																

- io write node 取的规则，分两种情况：

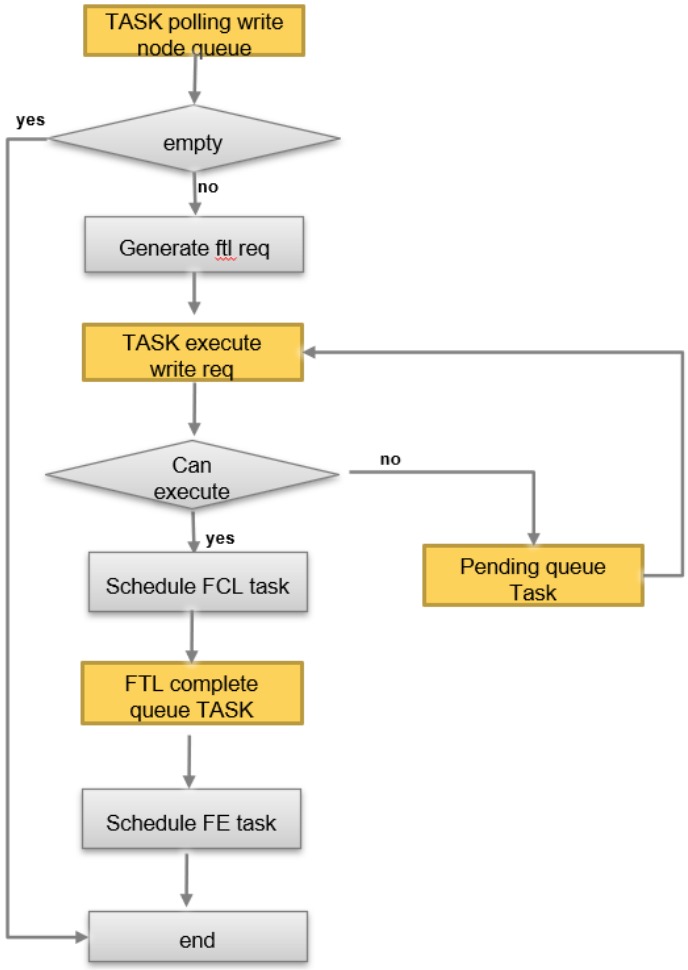
- normal write：只要 queue 里面有 write node，将一直按照 PU 大小去取，满足一个 PU 就生成 ftl req 去执行，直到取到剩余 write node 无法满足 PU 大小时停止，或者拿不到 ftl req 资源时停止。
- flush all：主要是 Idle flush、PMU flush、Shutdown Flush、Host flush。所有 write node 将全部取出写下去，先按 PU 大小取，满足一个 ftl req 就执行，而剩余不满足 PU 的 write node 也会取出，同时 FTL 会 PAD dummy data 成一个 PU 然后写下去。

三、Write Task schedule

为适配当前 FW 架构设计，整个写 flow 也将以 task 形式进行拆分。

首先，ftl req 也将拥有两个队列来管理，分别是 pending queue 和 complete queue。Pending queue 将管理由于各种原因无法执行的 ftl req，complete queue 将用来处理完成 FCL 操作后回来的 ftl req。注意此处 FTL 没有 submit queue，因为这个其实就是 cache 创建的 ftl write node queue，从 write node queue 取出生成 req 后会直接执行，无法执行的话会进入 pending queue，不再需要单独的 submit queue 来管理。

所以总共 3 个 queue：write node queue/pending queue/complete queue，每个 queue 将拥有一个 task 来处理，再加上本身业务的执行需要一个 task，一共 4 个 task。写流程这 4 个 task 的简单调度过程：



四、Write flow detail step

- Get PU info

User lu 的 PU 可能不是固定的，写之前需要获取该信息，其它 LU 的最小写单元就是一个 page，不需要去获取。

- Generate ftl req and submit to execute。

将多个 write node 的 buffer 及 laa list，放入 buffer sgl 及 lrange，然后生成写 req 之后调度 ftl req execute task。

- Judge ftl req 。

主要是判断当前 LU 被 block 条件，如：LU 正在 flush、lack page resource、p2l resource、flow control 等等，若 LU 被 block，当前 req 将无法执行，会挂入 pending

queue，被 pending 之前 req 已经拿到的资源需要释放。放入 pending queue 的 req，会被 pending TASK 重新调度执行。

- Check flow control

目前计划做一个简单的 fc，主要用来 balance between io write and gc，大致策略是通过 GC threshold and perf 去控制 io。

- Reserve resource
- Reserve page：若 page 不满足 PU 写，req 挂入 pending queue，同时 trigger allocation task 去分配 SPB。
- Reserve p2l entry：若 p2l entry 不够，req 挂 pending queue，同时 trigger p2l merge TASK，以释放 p2l resource.
- Reserve raid entry

因为资源不够而需要 pending 的 req，已经拿到的资源需要释放出来。

- Assign page

真正的分配物理 page，生成 PAA

- Setup meta \Setup ncl cmd \ setup RAID

填 meta info，填 ncl cmd 需要的信息，若支持 RAID，需要完成 RAID 相关的 setup

10、Submit to fcl to execute

11、FCL 执行将完成操作的 req push 到 FTL complete queue

Fcl cmd 中需要记住自己的 caller 是哪个 ftl req，done 后将 ftl req push 到 queue。

12、FTL task 处理 complete queue，执行 write done 操作，write done 分两种：

Early done:

- 会先直接 call back 前端 io done，执行 free write node、cache node 等操作。
- FCL 真正 complete 回来后，执行与 normal done 同样的操作，但不再 call back FE。

Normal done:

A、check FCL 执行结果，若有 err，将执行 user write err handling

B、update mapping，可能会 trigger flush p2l 或者 trigger merge

C、release lock，若该 lock block 其它 req，将 trigger 该 req 执行

D、release req

关于是否做 early done 的 FTL 与 FCL 的交互：需要 early done 的 req，将在 ncl cmd 置一个 try early done 的 flag，FCL 完成 early done 也需要置一个 flag 告诉 FTL。未置 flag 都当做 normal done 处理。

五、FUA write

当 req 取到的包含设置 FUA 的 write node 时，该 req 将设置 cache off write flag，并在 fcl cmd 上设置 no cache program 的 flag。

六、write err handling

write err 发生后，不同的情况将有不同的处理方式。

- 该 write 是 normal done：set grown defect\force close spb\GC SPB at later。如果需要 retry，将分配新的 SPB 把数据重新写入。
- 该 write 是 early done：set grown defect\force close spb\GC SPB at later。在 enable

RAID feature 的情况下，如果需要 retry，将通过先通过 RAID 恢复数据，然后写入新的 SPB。若未开 RAID，数据将丢失，无法 retry。