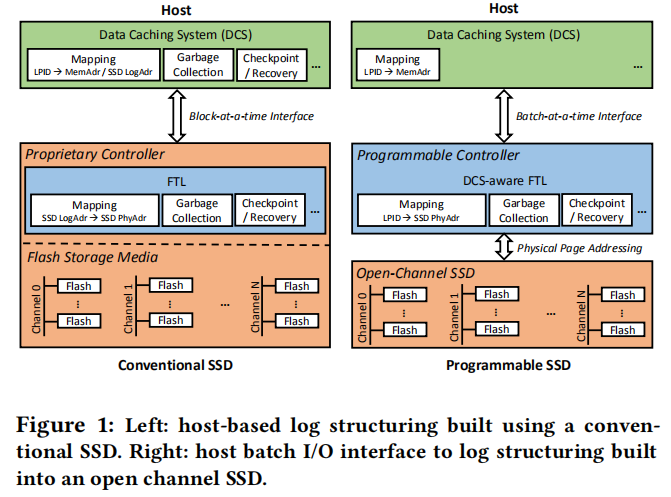
**Improving CPU I/O Performance via SSD Controller FTL Support for Batched Writes**

主机处理器➕固态硬盘内嵌入式处理器

应用领域：一般性（计算功能可由用户编程）数据密集型应用



可编程性：① 实现批量写接口；

② SSD内嵌入式处理器

在设备端完成批量写相关

处理（数据恢复、垃圾回收）

使用二级存储来存储数据虽成本较低（与主存存储数据相比），但由于数据访问存取的开销，其性能不佳。由此（为批量写）考虑使用日志结构存储（传统SSD未向主机端暴露布局，两者之间依赖日志地址LogAdr联系（如图，主机端：LPID→LogAdr，SSD：LogAdr→PhyAdr）），采用批量写减少I/O数量以减少数据访问存取的开销，但这引起了数据管理的开销。

开销举例如下：

映射时，主机将LPID（LPAGE ID/逻辑页ID）映射至一个基于主机的地址（图1中MenAdr），SSD将这个基于主机的地址看做逻辑地址再映射给物理地址（这一映射关系表明，SSD并不知晓主机buffer中的逻辑块信息）；

根据映射时的操作，则垃圾回收时，主机端还需回溯前次写buffer以找到有效数据的LPID以实现reloacte；

recovery时，除数据外，还需维护上述映射关系。

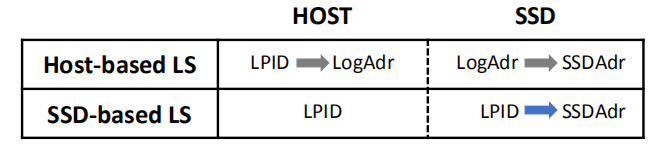
（上述操作均存在重复：FTL执行SSD端数据恢复/GC后，主机端还需重复执行主机端相应的数据恢复/GC）

由此，在OCSSD基础上使用一种适用于批量处理的接口（batch-at-a-time），使其能减少各类开销，提高性能。

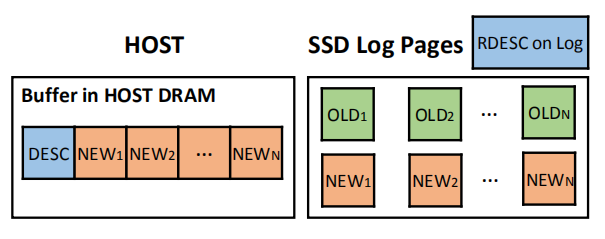
（因OCSSD将SSD信息暴露给主机端，则以下可行）

主机端将存储设备视为一系列固定大小的逻辑页（LPAGE），其由LPID标识。LPAGEs被写入buffer中（buffer大小为256-512个块大小），I/O批量处理整个buffer。buffer中第一个块中存储有描述buffer中各LPAGE的元数据（DESC），DESC包含buffer中包含的各逻辑块及偏移量（<LPID,offset>）。

由此可使用DESC与LPID在SSD与主机间建立联系，将映射关系简化：



recovery采用日志，在将数据写入物理地址之前，先由DESC创建RDESC，并将其加入SSD的恢复日志中。DESC与LPID相关联，LPID又在映射中与物理地址相关联，由此关联关系，RDESC可用于恢复。（会产生一定延迟，因为写数据前需先写恢复日志）



需要注意的是，RDESC仅用于恢复MAP，因其中不包括数据，不能rewrite。可通过读取物理地址中数据来检查是否已经成功写入了。采用多线程写数据以减少延迟，采用单线程更新RDESC中MAP。

若有写失败，则需将其重写至其他地址。创建一个修订日志记录RDESCA，其形式与RDESC一致，但仅包括写失败的LPID。RDECSA也在写数据前写入SSD的恢复日志中。若RDESCA其中的写仍失败则不断创建RDESCA直至完全写成功。完全写成功后才能更新MAP，更新MAP成功后返回确认ACK。

**（recovery时三者的顺序：log(RDESC/RDESCA)→写数据→更新MAP）**

上述过程仍存在一个问题，对于RDESC中的写是否已经成功，只能通过依次读取物理地址的数据来检查，如此的成本很高。引入done记录，若RDESC中已完全更新，则打上done以进行标识。

日志空间一般都是物理上连续的，但对于那些逻辑上连续的日志，为使RDESC的写入快速且持久，采用FORWARD（后继块）/back（前块）链接保证日志空间的有序性。

日志可能写入失败，各RDESC采用一个FORWARD链接的有序向量，向量大小（即日志可写入的块的数量）由一参数设定，日志写失败会触发重写，直至有序向量大小所表示的可写入块的数量耗尽。

日志除当前写入的新物理位置之外，还记录有之前各次写入的物理位置，这有利于垃圾回收（即垃圾回收时也可使用同一日志进行）。

垃圾回收机制采用一个全局比特向量GBITV记录各EBLOCK（即擦除单位：块）中有效RBLOCK（即读取单位：页面），通过各RBLOCK的TAG标志可关联其对应LPID，进而在数据迁移后更新MAP。这一过程可直接由SSD实现，能够避免主机回溯控制器DRAM中的整个EBLOCK。

垃圾回收机制同时也采用冷热数据区分以提高效率。

仍存在问题：若做GC的EBLOCK包含日志，则还未能实现在迁移的同时实现link链表的简单操作。

Summary：

Microsoft Research proposed a programmable SSD in 2019 to achieve great performance via SSD controller FTL support for batched writes in context of open-channel SSD and built it based on the Dragon Fire Card (DFC). Programmable SSD changes the current block-at-a-time interface to the batch-at-a-time interface and exploits a new FTL that supports multi-page I/O functionality without host-based log structuring in a controller for an Open Channel SSD, accelerating data accessing and improving I/O cost/performance by batching writes. Evaluations show that programmable SSD is between 1.91x - 2.49x faster than conventional SSD with both GC and checkpointing enabled.

利用OCSSD开放性，改造FTL和接口，设备端利用日志结构构造映射，通过批量写降低数据迁移开销 (RDESC/RDESCA)

（此与NSC“数据迁移开销大”的动机对应）