**Designing a True Direct-Access File System with DevFS**

主机处理器➕固态硬盘内嵌入式处理器

应用领域：各类应用（本文主要是将FS移至设备内，会用到FS的应用都会受益）

设计动机：过长的IO栈（这会带来开销）

传统存储堆栈需要将应用程序捕获到操作系统（OS）中，并与多个软件层进行交互。对于现代超快存储来说，软件交互大大放大了访问延迟。

对此问题（内核级FS存在的问题），之前的研究提出了混合用户级FS，但他无法保证文件系统的完整性、崩溃一致性和安全性。

为减少OS级别的开销，希望应用程序能提供直接的存储访问。文章提出了DevFS，其是完全嵌入于存储设备中的可直接访问文件系统，能提供直接、并发的访问，而不损害文件系统的完整性、崩溃一致性和安全性。DevFS还支持传统的文件系统层次结构，如文件和目录及其相关功能。

DevFS设计原理如下：

1. 分解文件系统数据结构，以接受硬件级并行性；

DevFS将每个基本数据单元（即文件）映射到独立的硬件资源。每个文件都有自己的I/O队列（per-file queue）和内存日志（per-file log），它允许跨不同文件并发I/O。DevFs采用POSIX接口，接口是基于指令的：用户级库拦截来自应用的I/O调用并将其转化为DevFS指令，在接收时，DevFS控制器（即设备CPU）通过该请求的文件描述符将其移动至per-file队列以进行后续处理和存储写入。

1. 在不影响用户级直接访问的前提下保证FS完整性；

FS原数据由可信任的DevFS进行更新；多进程共享数据时，DevFS跨应用程序共享per-file结构并序列化更新各结构。

1. 运用存储硬件电容简单的实现崩溃一致性；

传统OS级FS依赖日志或copy-on-write机制实现崩溃一致性，这会带来double write开销或很高的GC开销。DevFS运用存储硬件电容安全地就地更新数据和原数据。

1. 减少FS在内存中所占有的空间；

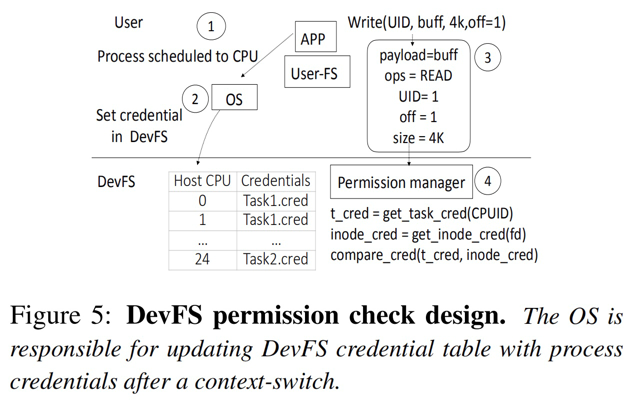
因为DevFS在设备中，占内存，应尽量减少其对内存的占用：

（1）**按需内存分配：**观察到原数据均在文件打开或创建时分配，在文件被删除时被删除，但均在I/O执行时才会被使用。基于此，DevFS按需进行内存分配，在读写请求开始时分配，在文件关闭时被删除；同时，根据内存可用空间大小动态分配per-file队列和日志大小；

（2）**原数据结构的反向缓存：**同样是基于上述观察事实，对于非活跃文件（即关闭的文件），将其原数据结构移至主机内存，并在该文件被删除时将这些原数据结构删除；

（3）**分解FS结构：**观察到数据结构规模较大时可能出现文件关闭仍需访问原数据结构的情况。基于此，将文件关闭时仍需被访问的数据和其他数据区分开来，这些仍需被访问的原数据结构仍存储在设备内存中，其他数据在文件关闭时移至主机内存中。

1. OS级状态共享。



因为DevFS独立于OS，其缺少对OS状态（即进程凭证）的可视性。为安全的进行文件访问，DevFS在设备内存中维护有一个凭证表，其只能被OS访问和更新。请求被恒定的主机CPU ID标记。处理请求前，DevFS根据其CPU ID进行凭证表表查找并对比两者的凭证。

Summary：

DevFS, a direct-access file system inside the storage hardware using the compute capability and device-level RAM to avoid software interaction overhead, was proposed in 2018. DevFS 1) provides each file with a separate I/O queue and journal to embrace hardware-level parallelism; 2) updates file system metadata as a trusted central entity, shares per-file structures across applications and serializes updates to these structures when multiple process share data to maintain file system integrity without compromising direct user-level access; 3) simplifies crash consistency with storage hardware capacitance; 4) aggressively moves inactive (closed) file system data structures off the device to the host memory to reduce the device memory footprint of the file system; and 5) provides safe operations without trapping into the kernel by extending the OS to share application credentials. With above techniques, DevFS provides applications with a high-performance direct-access file system that does not compromise integrity, concurrency, crash consistency, or security. Evaluation of an emulated DevFS prototype shows more than 2x higher I/O throughput with direct access and up to a 5x reduction in device RAM utilization.