本文是弗里堡大学Alberto Lerner和哥本哈根大学Philippe Bonnet在SIGMOD 2021发表的一篇综述。固态驱动器 (SSD) 只是硬盘驱动器 (HDD) 快速替代品的时代已一去不复返。 得益于 NVMe 生态系统，如今SSD 可以通过特定接口和现代 I/O 框架进行访问。 随着时间的推移，SSD 用途越来越广泛，现在可以支持从冷的高密度存储到热的低延迟存储的各种用例。找出哪种设备变体可以更好地支持给定的工作负载需要深厚的领域知识。本文第一个目标在于探讨不同SSD设计可以较好地支持哪些工作负载。

近来一种新型 SSD 在后摩尔时代计算机系统中起着至关重要的作用。 这些设备除了存储数据之外还可以运行应用程序逻辑，人们可以对SSD内部进行设计使其满足特定工作负载地需求。 因此，它们可以随存储数据量优雅地扩展处理能力。 本教程的第二个目标是为这种新型 SSD 建立设计空间，并为硬件、系统以及数据库研究人员提供探索新型SSD设计空间的工具。

Introduction

Initial SSDs：SSD与HDD相比具有很大的差异。HDD允许原地更新SSDs不允许。HDDs顺序的执行请求，SSDs可以并行执行请求。HDD错误率相对较低，SSDs依赖ECC纠错码。简单来说SSDs比HDDs复杂的多。在起初发展的时候为了更快的被接受SSDs采用了和HDDs类似的接口，这使得SSDs隐藏了其内部机制并对外表现为一个块设备。SSDs中的逻辑地址LBA并不能反映物理地址PLB。因此，应用程序失去了控制数据放置的能力，即决定要连续存储哪些数据块。此外不像HDDs的请求可以被OS调度，SSDs内部对于OS来说不透明，因此OS很难对请求进行重排序以降低延迟。总的来说SSDs很快，但是性能很难建模。

Diversification of SSDs：**对数据库系统性能模型必不可少，SSDs在这些系统中的重要性早已被认识到。然而，数据库系统历来控制数据放置和IO调度，以满足它们所依赖的不同工作负载的需求（如下图所示）。在这些工作负载下SSD性能如何不是数据库管理员可以回答的，因为这取决于特定的SSD配置，毕竟SSD不是一个标准统一的设备。

	access type	operation	queue depth	block size	observations
log writing	sequential	write	1	small	circular, latency sensitive
checkpointing	sequential	write	high	large	may never be read, low priority
buffer flushes	random	write	highly variable	average	bursty
index traversal	mixed	read	mixed	average	prefetching opportunities

Table 1: Characterization of some of the workloads generated by a database system.

SSD性能表现取决于其使用single-cell还是multi-cell NAND-Flash、DRAM cache还是DRAM-less cache、轻度还是重度overprovisioning、低延迟导向还是高吞吐导向、single plane还是multi-plane等等。对于数据库管理人员来说更好的问题在于：**对于给定的一个工作负载，如何选定最适合的SSDs。**

Modern Interfaces and Abstractions：有些工作建议为SSDs配备外部接口，把控制权还给应用程序，其中一个显著的例子就是Open Channel SSDs接口。简单来说，OCSSD设备暴露其内部结构，对应用程序而言是一个white box。此外，OCSSD设备也将IOs的调度控制权交给了应用程序，这使得应用程序得以重新控制数据摆放以及请求调度，同时应用也需适应给定设备的NAND-Flash属性。*Stream*接口降低了不同工作负载之间的干扰，支持*stream*接口的SSDs设备对于应用来说是black box。

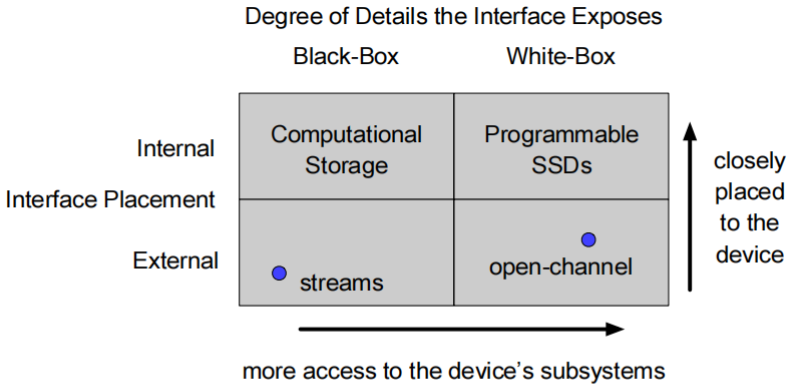

Figure 1: Interfaces fall into quadrants depending on how closely they are placed relative to the device and the offered visibility into its mechanisms.

Specialization and Co-Design:

目前SSD设备不再单纯作为存储设备了，应用逻辑可以被卸载到SSDs中，且目前可以有两种实现方式。**第一种为*computational storage***。该设备通常为应用逻辑配有FPGA逻辑单元，FPGA不需要穿越总线例如PCIe子系统到达设备。CSD依赖内部的黑盒SSD接口来处理进出设备的数据和请求。CSD的一个经典应用场景为早起谓词评价(early predicate evaluation)。调用者可能希望扫描数据集的块，但对那些没有通过过滤条件的块中的条目不感兴趣。然后，调用者将谓词计算逻辑推入设备，而设备则将其应用于块，从而生成一个过滤后的流。换句话说，过滤后的数据永远不会离开设备，从而节省了宝贵的数据传输。**第二种为programmable devices**，即SSD的内部机制可以被编程。这里的假设是，一个可编程的设备是模块化的，并且内部接口描述了可替换的组件。

文章说明SSD多样性和应用程序控制不需要相互排斥。两者都应该激发非常适合数据密集型系统和应用程序的新SSD设计。总的来说作者认为：重新探讨外部接口是co-designed SSD的先决条件，理解内部接口的作用也同样重要。

External Interfaces

每个Block都有自己的地址LBA，一个块设备是一个带有简单读写接口的内存抽象。经验上来说顺序读写要比随机读写快上几个数量级但是SSDs天然不支持这一特性，有很多工作旨在解决这一限制。

- **Open-Channel**使用NVMe扩展重新定义了SSD接口。其对外暴露了设备的内部细节以及基于read、write、erase命令支持向量化IOs。操作的粒度由SSD定义。主机端负责数据放置、IO调度。具体来说，主机端负责管理一个写指针，该指针追踪数据应该写在何处以保证块内的顺序写操作。OCSSDs在Linux中通过LightNVM支持。
- 自2020年7月以来，**ZNS**一直是NVMe标准的一部分。它定义了分区命名空间，即，区域中逻辑地址空间的分区，由逻辑块组成，约束逻辑块必须在区域内按顺序写入。ZNS驱动器尚未商业化。在QEMU和Linux堆栈中有对ZNS的支持。
- 块设备接口的扩展如*directives*，使主机能够与嵌入的FTL合作。

Internal Interfaces And Mechanisms

Caching

Flash控制器中的DRAM缓冲区可以作为高速缓存。许多ssd使用它来吸收写操作并提供对它的读取，并具有预期的性能改进。使用典型的LRU策略管理的缓存可能不能反映不同操作的优先级。例如，具有更大操作的工作负载（每个操作有更多的页面）的工作负载将比较小的操作工作负载占用更大的缓存部分，即使工作负载在读/写频率方面是相似。因此，一些工作试图以不同的方式管理缓存替换策略。

Scheduling

LUN是组织NAND-Flash单元的基本单元。一个SSD由许多lun组成，每个lun都可以随时提供一个请求。通常，读取和写命令被分解为单独的Flash（页面大小的）操作。特别是写操作，它可以选择被分配给任何可用的LUN。较大的业务操作将倾向于同时使用更多的lun，从而在其他可能正在进行的业务中造成“延迟”效应。换句话说，在飞行中的操作之间几乎没有孤立之处。一些工作通过在工作负载之间建立公平的调度来解决这个问题，而另一些工作则求助于暂停/重新调度昂贵的操作，这是干扰[44]的根本原因。

Data Placement

在大多数ssd中，来自不同工作负载的页面可以被放置在同一个块上。块级别的混合页面至少有两个负面的性能影响。第一，如果工作负载呈现不同的页面生命周期，则它们的页面将在不同的时间失效（删除）。这将使用但空块自然发生的机会降到最低。空块对于快速垃圾收集至关重要，因为它们允许擦除命令立即执行，跳过了将仍然有效的页面从受害者块复制到新块的需要。第二，放置在同一块中的不同工作负载的热页面不能并行读取。因为一个LUN的页面缓冲区一次只能提供一个读取操作。有些工作通过分离每个工作负载可以使用的块来解决这些问题。有些甚至更进一步；他们“雕刻”了设备的区域，并将这些区域分配给不同的工作负载，试图实现确定性的响应时间。

ECC Calibration

考虑一个处理图像的工作负载。如果一个页面存储了一个图像的像素，也许一些应用程序会容忍页面上的几位被翻转，以交换性能。这是一些ECC责任转移到应用程序上，允许它们决定是否使用错误的页面。

Computational Devices

SNIA机构区分了基于压缩或加密等预定义功能的**固定计算存储服务**以及通过代码上传的**可编程计算存储服务**。其中通过代码上传的可编程机制包括：操作系统映像、容器化应用程序、FPGA位流和eBPF字节码。NVMe计算存储标准预计将在2022年进行扩展。虽然它将定义将代码上载到计算存储的机制，但它将不解决如何开发为给定的数据密集型系统开发高效和安全的计算存储程序。这仍然是一个悬而未决的问题。

The Co-Designed Continuum

从SSD的适应性角度出发总结了五种协同设计的思想：

- **调整工作负载以适应SSDs设备**
- **精心挑选合适的商用设备**
- **进一步裁剪调整设备以适应工作负载**
- **使用Computational Storage的旁路接口**
- **针对特定的工作负载设计专用设备**

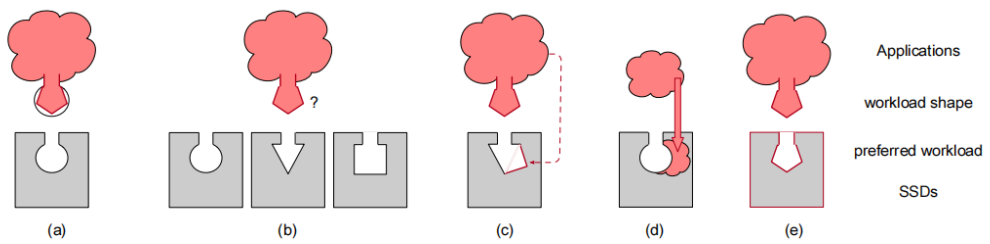

Figure 2: Continuum of adjustments an application, on top, and a device, on bottom, can make in order to interact efficiently. The interaction occurs through an application-generated workload, to which a device may be more or less efficiently tuned. The steps on the continuum are: (a) the application unilaterally adapts to the SSD characteristics to perform well; (b) the application weighs in among alternative SSDs before selecting one; (c) the application configures some aspects of a device to its needs through an external interface before using the device; (d) a computational device serves as an application platform, running part of its logic through an internal, black-box interface; (e) a programmable device replaces or adjusts some of its components through an internal, white-box interface to fit the exact needs of an application.