INVITED PAPER

采用3D XPoint技术的平台存储性能

本文回顾了3-D XPoint技术在改变内存-存储层次结构方面所带来的计算潜力。

作者：Frank T. Hady, annie Foong, Bryan Veal, and dan Williams

ABSTRACT | 3D XPoint内存兼具高性能和非挥发性，它的到来有望从根本上改变硬件、系统软件和应用层面的内存存储层次。这种内存将首先被部署为块状可寻址的存储设备，即英特尔Optane SSD，即使以这种熟悉的形式，它也将推动基本的系统变革。访问时间与系统其他部分的速度一致，甚至更快，这将模糊存储和内存之间的界限。这些固态硬盘（SSD）的低延迟使得人们甚至可以重新思考基本的存储方法，使其更像内存。例如，衡量存储性能的方式从给定队列深度的输入输出操作（IO）转变为给定负载的响应时间，就像通常衡量内存一样。匹配这些固态硬盘的低延迟的系统变化已经很先进了，在许多情况下，它们使应用程序能够利用固态硬盘的性能。在其他情况下，需要额外的工作，特别是在最初考虑到慢速存储的政策设置上。在这些已经具备能力的系统之上的是真正的应用。系统级测试表明，诸如键值存储和实时分析等应用可以立即受益。这些应用的好处包括明显加快运行时间（高达3倍）和访问比DRAM支持的更大的数据集。新的可行的扩大应用内存足迹的机制包括本地应用支持或本地操作系统分页，这是使用SSD的一个重大变化。这种融合的下一步是通过处理器加载/存储操作访问3D XPoint内存。重要的操作系统支持已经到位。一贯的低延迟存储和快速持久性内存对计算的影响是巨大的，作为存储的应用程序和系统将首先受益于这种新技术。

2017年1月16日收到稿件；2017年6月16日修订；2017年7月17日接受。
出版日期2017年8月7日；当前版本日期2017年8月18日。
(通讯作者：Frank T. Hady。)
作者为英特尔公司，Hillsboro, OR 97124 USA（电子邮件：Frank.hady@ intel.com；annie.foong@intel.com；bryan.e.veal@intel.com；dan.j.williams@intel.com）。

数字对象标识符。10.1109/JPROC.2017.2731776

关键字｜大数据应用；计算机架构；计算机性能；数据库系统；分布式数据库；闪存；英特尔Optane固态硬盘；内存架构；非易失性内存；持久性内存；固态硬盘；系统软件；3D XPoint内存

I. 简介

几十年来，存储一直是计算机系统中的性能落后者。处理器的性能年复一年地不断提高，而硬盘上的数据仍然是以毫秒为单位。许多系统研究和工程创新一直致力于解决这一延迟鸿沟。最近，基于nand的固态硬盘(SSD)已经比硬盘的延迟提高了很多。超过10倍。即使是nand固态硬盘，其基本系统的平衡没有变化--储存量仍然是延迟滞后。新的内存技术，通常被称为存储级内存，有望颠覆这种几十年的不平等。作为固态硬盘或持久性存储器部署，这种新技术使持久性数据存储与系统的其他部分一样快，为系统的基本变化提供了新的机会。在这一类技术中，处于不同成熟阶段的有PCM[1]、MRAM[2]、ReRAM[3]、NVDIMM[4]和3D XPoint内存。在本文中，我们重点关注英特尔Optane固态硬盘的系统影响，它采用3D XPoint内存，最近开始出货。

作者是负责3D XPoint技术系统架构的团队的一员。当我们在2008年开始工作时，系统仍在向nand SSD过渡，普遍的看法是，即使实现了新的低延迟SSD，根深蒂固的系统软硬件做法也会掩盖应用的好处。许多研究人员专注于推动革命性的新系统和设

备设计[5]-[7]。然而，在发展中的更低延迟的固态硬盘的知识的武装下，我们的重点是进化的系统变化，只在需要时修改系统，同时仍然

使得系统应用可以获得较低延迟的存储性能优势。通过一系列的系统级分析，指出了存储路径中的瓶颈，我们提供了经验证据来指导优化，以确保系统硬件和软件不会成为低延迟SSD的瓶颈。在这里，我们将展示该系统已经准备好让应用程序有效地获得低延迟存储的好处，无论是作为存储还是作为扩展内存。在第二节中，我们探讨了系统配置，以满足存储类内存的三种主要使用模式：持久性存储，非持久性DRAM扩展，以及持久性内存。我们展示了低延迟固态硬盘和持久性内存设备是如何实现所有三种模式的。在第三部分，我们总结了历史上的系统硬件和软件的分析和优化，使今天的系统能够利用高性能的基于nand的PCI Express SSD。这为英特尔Optane固态硬盘的性能评估和进一步的系统启用提供了基线。为了能够充分评估低延迟固态硬盘的性能，在第四节中，我们解释了为什么我们必须从常见的每秒输入输出操作数（IOPS）-队列深度的存储性能测量方法转向基于响应时间的内存性能测量方法。性能测量方法。

在第五部分，通过基准和工作负载的描述，我们提供了使用英特尔Optane SSD的工作负载性能和数据集大小优势的例子。

在第六节中，我们介绍了已经在进行的系统启用，以使3D XPoint内存作为内存模块部署。

II. 作为存储年龄和记忆的3dxpoint内存

本文的主要焦点是可作为英特尔Optane SSD的3D XPoint内存。我们在系统、使用模式和持久性内存的背景下讨论这种SSD。对于这种新的内存，系统中存在三种主要的使用模式。

1) *储存*。需要持久性。应用程序使用操作系统和文件系统的服务，或使用自我实现的代码来管理策略和确保交易的正确性。它通常以固态硬盘的形式提供，通过软件管理的交易，使用系统几十年来对基于磁盘的存储所期望的块的抽象，使之可用。应用程序通过系统调用访问存储，如`read()`和`write()`，对设备进行异步的输入输出操作（IO）。

2) *DRAM扩展*。不期望有持久性。这个用例通常是由工程限制或对DRAM总容量的成本限制引起的。应用程序将对象作为内存与CPU进行寻址加载和存储指令，而操作系统通过分页在内存和存储之间移动数据。

3) *持久性记忆*。持久性是需要的。应用程序将对象定位为能够维持持久性要求的内存。持久性内存可以直接通过处理器的加载和存储指令来使用。虽然没有明确说明，但为了广泛使用，持久性内存设备必须有与处理器速度相称的延迟。

A. 储存

经验告诉我们，对于任何新技术，都会有客户以牺牲一些性能为代价，通过最大限度的向后兼容（例如，继续使用`read()`和`write()`系统调用）来寻求快速使用新技术。为此，3D XPoint内存将首先被用作Optane固态硬盘中的快速存储[图1(a)]。

Optane SSD已经优化了吞吐量，以提供低IO延迟。我们将在本文后面展示该固态硬盘的NVMe接口的影响。在固态硬盘本身，控制器将单个4KB的IO分散到多个3D XPoint内存通道中，以利用多个内存芯片的吞吐量，为单个IO提供低延迟。这与基于nand的固态硬盘形成鲜明对比，后者通常访问单个芯片以满足4KB（或更大）的IO。Optane固态硬盘包括一个用于从主机到介质的正常读写的纯硬件路径，避免了nand固态硬盘中经常出现的来自固件的额外延迟。通过使用3D XPoint内存，Optane固态硬盘还能够就地写入数据，避免了nand固态硬盘所需的擦写驱动的垃圾收集。正如我们将展示的那样，这种功能组合使Optane固态硬盘能够提供非常低的平均延迟，并且比nand固态硬盘的异常延迟小得多。

与固态硬盘一样，也可以将持久性内存作为通过使用支持持久性内存块设备模拟的操作系统，使现有的应用程序能够不加修改地执行。块模拟的一个例子是Linux内核4.1中的持久性内存块设备支持[8]。新的文件系统[6]，[9]，[10]已经出现，以充分利用持久性内存的字节寻址能力，允许应用程序重新使用熟悉的文件语义。Dulloor[11]创建了一个文件系统，实现了对持久性存储器的标准文件操作，为持久性存储器文件系统带来了显著的性能优势。

B. DRAM扩展

此外，即使是以SSD的形式，3D XPoint内存也可以有效地通过分页（也称为交换）来扩展DRAM[图1(b)]，而不需要改变应用程序。分页来扩展DRAM一直是操作系统虚拟内存管理的主流。我们发现，分页的效果令人惊讶。

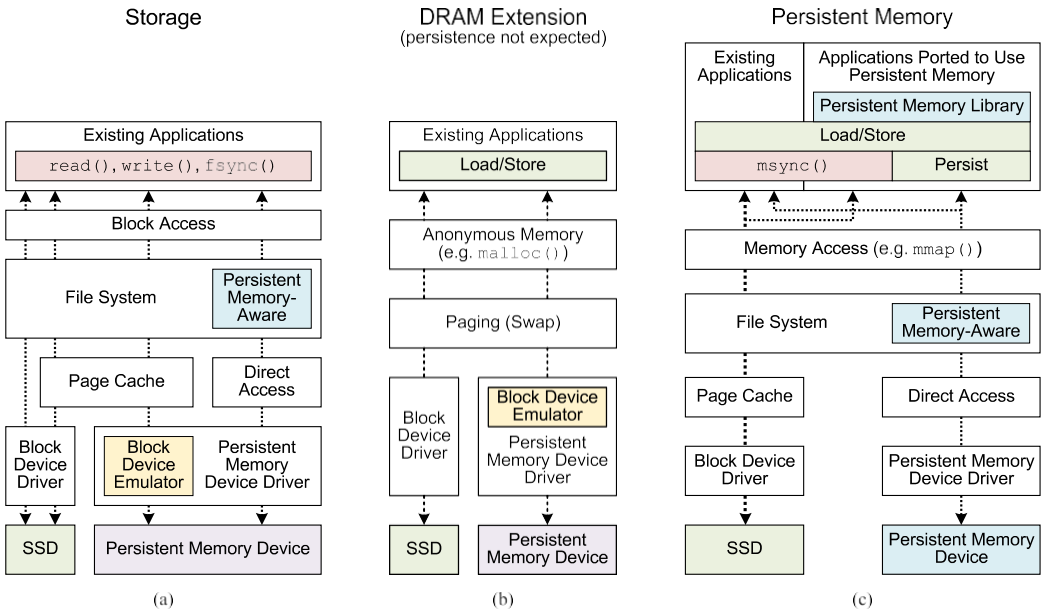


图1. *固态硬盘和持久性存储器设备的可能使用模式：（a）作为存储；（b）作为DRAM的扩展；或（c）作为持久性存储器。*

自Linux*内核3.18以来，Optane SSD的低延迟性很好（有一些策略参数的调整）。管理程序也可以提供分页，在不调整或修改操作系统的情况下提供一个优化途径。[12].

C. 持久性记忆

已经使用内存映射存储的应用程序（例如，通过mmap()），在需要时提交写入（例如，通过msync()），已经有效地实现了持久性内存感知[图1(c)（左边）]。应用程序或操作系统不需要为持久性内存的正确性而改变代码。性能的提高将取决于工作负载，但应用程序将正确工作。我们将在第III-B节回到这个话题，并进行量化。

作为迈向原生持久性内存支持的一步，操作系统已经为持久性内存设备添加了驱动[13]，允许更高层次的设备建立在它们之上[图1（c）（右）]。Linux内核已经集成了直接访问块层（DAX）作为持久性内存的优化。反过来，DAX支持已经被添加到Ext4和XFS文件系统中[图1(a)（右）]。使用DAX，一个未经修改的应用程序可以立即获得持久性内存与全块设备模拟的好处[14]。在支持DAX的文件上，传统的read()和write()系统调用绕过了操作系统多余的页面缓存，而是作为内存拷贝实现到持久性内存。

同样，对于利用内存映射的文件或设备的应用程序，DAX再次删除了页面缓存，将持久性内存直接映射到应用程序的虚拟地址空间，在那里可以直接用加载和存储CPU指令进行访问。然而，持久化需要使用额外的指令来刷新CPU缓存，以便将数据提交给持久化内存。

为了适应现有的应用程序，持久性的要求促使持久性内存与文件系统类似地被管理[13]。然而，持久性内存也可以通过提供高级功能的辅助库来实现，以方便应用程序的移植。例如，NVM库[15]是一个由7个库组成的集合，旨在实现包括事务性对象存储、内存池管理和通过RDMA协议访问远程持久性内存的使用情况。

III. 十年来的优化和相关工作

2008年，我们意识到固态硬盘有可能比nand固态硬盘的响应时间快一个数量级，因此开始研究系统其他部分所需的优化，以有效地访问这种固态硬盘。我们开始测量系统层面的延迟和开销。在整个过程中，我们使用了基于英特尔至强处理器的服务器平台上的Linux作为参考系统，以保持一个共同的基线，并根据需要灵活地实施改变。我们相信（并通过经验验证），所获得的架构洞察力普遍适用于其他系统软件。一个最小的存储路径由存储软件、驱动程序、存储协议和控制器组成（图2）。我们对整个存储路径进行了检测（对软件使用代码检测，对硬件使用PCI Express*（PCIe*）和SATA分析器）。我们检查了从应用程序到SATA SSD的IO的时间profile，在2008年可用的平台和操作系统上[16]。当时，延迟仍然是由存储设备主导的。一块固态硬盘需要大约140微秒，而系统的其他硬件和软件--从存储互连到应用程序的所有东西--需要40微秒。正如我们所显示的

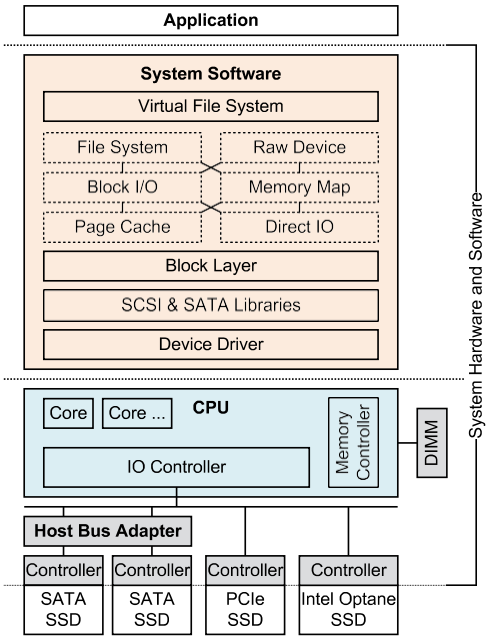


图2. *导致IO延迟的系统硬件和系统软件的一个例子。*

在本节后面，虽然系统延迟不是2008年的瓶颈，但对于Optane固态硬盘来说，它本来就是瓶颈。有了这些知识，我们开始了三个架构目标。1）减少系统延迟；2）扩大吞吐量；和
3）提高系统软件的计算效率（如果需要）。

A. 平台硬件优化

我们观察到，作为存储连接点的SATA或SAS给存储路径增加了大量的延迟开销。需要一个主机总线适配器（HBA）来从PCIe（无处不在的CPU-IO系统互连）转换到SAS或SATA协议。这样的HBA给存储路径增加了大约25μs的延迟。此外，HBA的一个主要作用是聚合多个低带宽的存储设备（每秒数百兆字节），以配合上游系统互连带宽（每秒数千兆字节）。这对较慢的存储子系统是有意义的，但对低响应时间和/或高带宽的存储则没有意义。低延迟固态硬盘的最佳接口必然是去除HBA而直接连接到CPU的接口。

将固态硬盘直接连接到PCIe系统总线上，也可以通过CPU支持的PCIe通道提供可扩展的带宽。Amdahl的平衡系统法则[17]认为，一个系统每条CPU指令需要一位IO。一个现代的服务器系统能够提供超过100 GHz的计算能力。假设CPI（每条指令的时钟数）为1，这个系统将需要12.5GB/s的顺序IO带宽，或300万个4KB的随机访问操作的IOPS。这意味着需要20-40个SATA或SAS固态硬盘。或20000-30000个硬盘（假设是随机访问）。一个现代的服务器处理器能够通过直接暴露出超过40GB/s的PCIe通道来支持这种IO需求。要实现Amdahl的平衡，更好的选择只是四个4-GB/s的PCIe固态硬盘--实现了部署的便利性、更低的功耗和更低的成本。

早期版本的PCIe固态硬盘于2007年出现在市场上[18]。认识到需要一个高效的标准接口，英特尔与业界合作，为PCIe SSD创建了NVM Express*（NVMe*）规范[19]。

B. 系统软件优化

在2008年，一个典型的存储软件堆栈（例如，微软*Windows*操作系统或Linux操作系统）产生了大约12μs的处理量[16]。有了硬盘驱动器，软件栈的计算要求就不是问题了。有了百万IOPS的固态硬盘，支持一个固态硬盘的计算要求将达到十个处理器内核的饱和。一台商品服务器拥有大约20-40个核心，一个商业2U服务器机箱能够容纳多达24个2.5英寸的SSD。充分利用这些核心需要提高存储堆栈的处理效率，以确保存储的计算需求不会成为系统的瓶颈。我们的分析促使我们以两种方式减少这种计算要求。1）设计一个推送IO完成的控制器接口，而不是依赖主机发起的状态寄存器读取，如那些由常见的应用主机控制器接口（AHCI）所使用的；2）通过将IO完成的处理引导到请求的内核（通过多个队列和矢量中断），使性能随内核扩展。这两个建议都被NVMe采纳。

通过这些优化，我们的分析表明，文件系统、IO块和协议处理、上下文切换和驱动之间的处理将被平均分配。Linux NVMe驱动开发者避免了通常的SCSI/ATA协议层，而是直接将块和驱动层加入到一个高度优化的NVMe SSD的存储栈中。Bjørling的blk-mq（多队列块层）补丁[20]减少了块存储的IO堆栈中的指令数量和争夺，这使得Linux存储堆栈的处理要求从9μs降低到4μs。Yang等人[21]进一步推动了这一进程，他们通过轮询的方式从路径中移除上下文切换，显示出存储堆栈的经验最小可能性为1.5 μs。

鉴于许多应用程序通过文件系统访问存储，我们也检查了Ext3文件系统的开销，发现代码路径的开销相对较小，只有1.5μs（未显示）。然而，我们看到，策略和元数据布局是基于硬盘假设的（即为顺序访问而优化），对性能的影响更大。当时，已经有多项工作在进行，以设计NVM感知的文件系统[6], [22], [23]。

最近，Sehgal等人[24]通过对NVM感知文件系统（PMFS和F2FS）与五个传统文件系统（Ext2、Ext3、Ext4、XFS和NILFS2）的全面实证比较，验证了我们早期的质量见解。他们表明，通过选择正确的策略--记事模式、分配策略，以及避免使用原地执行的缓冲区缓存，现有的文件系统在真实世界的工作负载基准中接近PMFS的5%。与此相反，最近的文件系统如NOVA[25]超过了PMFS的性能。

最后，我们希望验证通过分页支持DRAM扩展使用模式的可行性[图1(b)]。从历史上看，分页并不是一个可行的选择，因为存储延迟很高，即使是使用nand SSD。Yang [26]表明，在Linux内核2.x中，以硬盘为中心的分页政策导致了高离群值（>60ms）。我们的分析（从Linux内核3.18开始）表明，通过策略调整，实现分页的代码只产生5μs的开销。始终预取和交换多块数据的默认策略对硬盘来说可能是最理想的，但对Optane SSD来说并非如此。

在Linux内核4.6.7上使用单核进行测量，nand固态硬盘的页面缺失平均需要92μs，但英特尔Optane固态硬盘只需要16μs。[1]为了改善多核分页的性能扩展，Linux开发人员正在努力消除竞争的来源[27]。我们将在接下来的章节中展示使用英特尔Optane固态硬盘来扩展DRAM在实际工作中的可行性。

除了注意到的代码低效外，我们发现系统软件和硬件机制几乎是最佳的，包括文件系统和分页实现。我们已经暴露了由于优化慢速存储的策略而导致的延迟异常，例如，驱动器中的中断凝聚，IO调度器中的IO合并，预取，使用日志结构而不是就地更新，以及使用中间缓冲区间的缓存。这样的策略对硬盘甚至是nand SSD都是有意义的。然而，对于英特尔Optane固态硬盘来说，系统的平衡点已经发生了变化；复杂的IO调度和缓冲不仅没有必要，而且会妨碍到它。通常情况下，这些策略的改变是简单易行的，提供了最佳的优化投资回报。

图3显示了本节描述的大部分跨行业工作的影响，绘制了三种不同平台接口--ATA HBA、SAS HBA和PCIe SSD--所有与主机的4-GB/s PCIe链接的IOPS与系统延时的关系。[2]请注意，只显示了系统软件和硬盘的延迟和吞吐量，不包括媒体延迟，以关注系统的影响。对于SATA和

[1]在单处理器3.4-GHz英特尔酷睿i7-4770平台上进行测量。

[2]在一个2.3GHz的双处理器英特尔至强E5-2695平台上测量，集成AHCI SATA HBA、Broadcom* SAS 9207-8I HBA和英特尔SSD DC P3700系列。

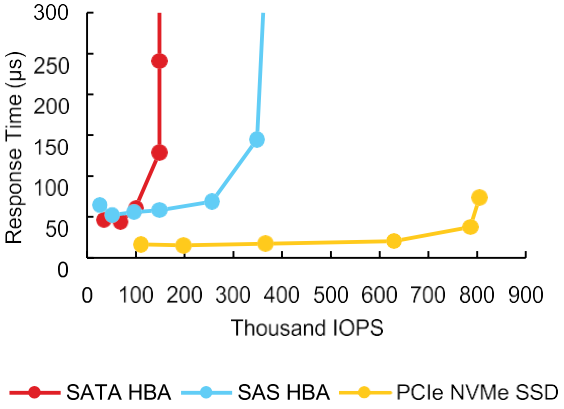


图3. *99.99百分位数的响应时间（不包括媒体）与三个SSD接口的4KB读取的IOPS的关系图。*

在SAS数据点上，我们连接了尽可能多的SATA nand SSD，使一个SATA或SAS HBA达到饱和，以提供一个公平的比较。系统软件和硬件优化的结合使NVMe接口达到接近PCIe的理论最大值（约800 000 4-kB IOPS），同时在最大IOPS之前保持明显的低延迟。

最后，我们将一个优化良好的系统与3D XPoint内存结合起来，形成了图4所示的英特尔Optane SSD延迟。为了进行比较，我们包括了2008年的SATA SSD系统和nand NVMe SSD的数据。在PCIe总线上，从NVMe门铃写入到完成中断，从英特尔Optane固态硬盘访问一个4KB的数据块的时间不到8μs，显示了之前讨论的Optane固态硬盘内置的延迟优化的结果。从应用到SSD的时间总共约为11μs，显示出系统增加了不到4μs。系统（在正确计算或优化的情况下）为应用程序提供了超低延迟的存储时间。

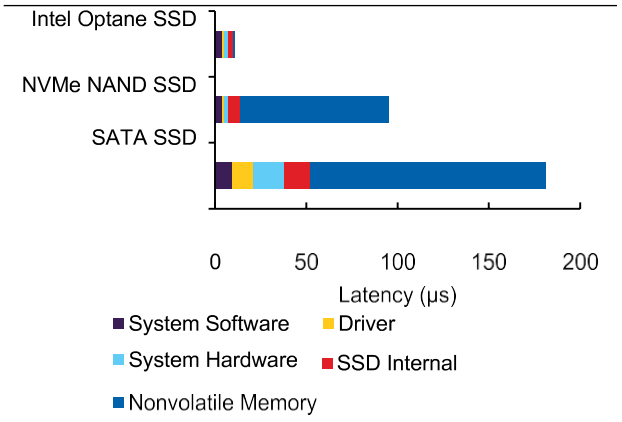


图4. *与nand SSD相比，英特尔Optane SSD从应用到设备的延迟。*

IV. 评估绩效

当我们努力调整系统，尝试使用低延迟存储时，我们发现我们测量存储的方式需要改变。存储设备性能的测量有很长的历史，有一套既定的语言和测量方法。虽然传统的方法学已经用于测量硬盘驱动器和固态硬盘，但在本节中，我们展示了这些基于队列深度的方法学如何不能准确地将传统的固态硬盘与那些延迟与用于访问它们的软件栈相似的固态硬盘进行比较。我们引入了一个
基于IO响应时间的改进方法，反映了
的平台内存子系统测量。

今天常见的存储性能指标是IOPS。
存储基准工具，如fio[28]和Iometer[29]，通常被用来测量IOPS。这些工具运行一个特定数量的线程，每个线程向设备发出IO。该工具试图产生的并发未完成IO的总数被称为测试的队列深度。在常见的性能基准测试方法中，例如Tom's Hardware使用的"四角测试"[30]，队列深度被配置为操纵SSD上的负载，而IOPS被测量。因此，队列深度被报告为*X*轴上的自变量，而IOPS成为*Y*轴上的因变量（图5）。

高容量的nand固态硬盘有许多nand芯片，并依靠IO在芯片上的平行分布来克服每个芯片的延迟并提供高的IOPS
[31]。在前面描述的性能测试中，高队列深度被用来以足够高的速率提供IO，以利用并行性的好处。如图5所示，基于nand的英特尔固态硬盘DC
P3700系列的IOPS随着队列深度的增加而增加。[3]

对于英特尔Optane SSD DC P4800X系列，图5显示了Optane
SSD的低延迟介质和前面描述的设计所带来的IOPS的快速增长，在队列深度仅为8的情况下，最大IOPS达到了高潮。这是一个明显的改进；例如，nand
SSD在队列深度为128时达到约300 000
IOPS，而英特尔Optane
SSD在队列深度为4时达到类似的IOPS。
然而，使用队列深度来代表负载，并没有充分暴露英特尔Optane SSD等低延迟存储的性能改进。

图2显示了从用户级应用程序测量每个IO的处理时间的三个因素：系统软件、系统硬件和SSD本身。这些层的平均未完成IO数量与该层的延迟成正比。从历史上看，几乎所有的时间都是在存储设备本身花费的，所以队列

[3]性能是在系统上测得的，系统配备了两个英特尔至强E5-2699 v4处理器，主频2.2GHz，采用英特尔涡轮增压技术，最高频率3.6GHz；英特尔服务器板S2600WT系列；256 Gbytes DRAM；Ubuntu*服务器操作系统16.04.2 LTS x86_64；Linux内核4.4.0-21- generic（Ubuntu构建）；以及fio 2.2.10。SMP中断的亲和力被改变，禁用了固态硬盘的irqbalance。CPU治理器被设置为性能，udevd被禁用。

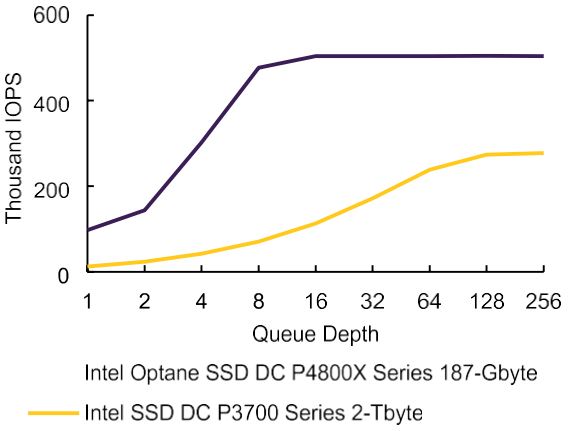


图5.在*给定的队列深度下，nand固态硬盘和英特尔Optane固态硬盘之间的IOPS比较，给定的随机分布的工作负载是4KB的IO，其中70%为读取，30%为写入。*

深度是设备上负载的一个很好的代表。对于具有存储级内存的固态硬盘，固态硬盘的延迟与系统延迟是相近的。这使得队列深度不能代表固态硬盘的负载，因此也不能作为比较的基础。

图6中提供了一个例子。考虑一个工作负载，它要求在一个系统上每20μs发出一个IO，这个系统对每个IO增加了10μs的延迟。对于具有100μs延迟的SSD来说，这个工作负载导致SSD的平均队列深度为5[图6（a）]。对于第二个延迟较低的10μs的SSD，SSD的平均队列深度为0.5[图6(b)]。同样的系统负载导致了SSD的队列深度大大降低。有一半的时间，第二个SSD没有工作。

为了在两个SSD之间进行更公平的比较，工作负载可以增加到每4μs一个IO，以实现第二个SSD的整体平均队列深度为5。然而，现在SSD本身看到的队列深度只有2.5，因为系统负载也增加了[图6(c)]。这种试图对方法进行修补并产生

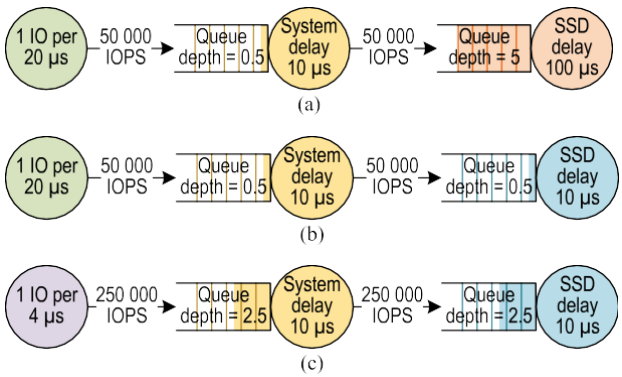


图6.*具有以下特点的系统的平均排队深度比较*
(a) 较高延迟的固态硬盘；(b) 较低延迟的固态硬盘；以及(c) 较高请求率下较低延迟的固态硬盘。

相同的队列深度并不能导致公平的性能比较。

也许更重要的是，队列深度的IOPS忽略了数据被检索的等待时间。延迟本身是提供最相关的比较的性能指标。图7显示了基于nand的固态硬盘和英特尔Optane固态硬盘之间的延迟，或在最小负载下的延迟比较。平均而言，我们看到nand固态硬盘产生了约10倍于英特尔Optane SSD的延迟。此外。如图7所示，这在队列中无法显示。在深度与IOPS的对比图中，两个固态硬盘之间的延迟离群值相差很大，最高达到99.999%。我们认为较低的延迟离群值是一种较高的服务质量（QoS）。在第四节，我们将展示从更高的QoS中受益的工作负载。

从延迟扩展开来，图8显示了负载下的延迟，或响应时间，以及响应时间的QoS，用于产生图5的相同测试。在这里，负载直接由IOPS表示，而不考虑用于产生负载的队列深度，响应时间作为性能指标呈现。在对数尺度的*Y*轴上，nand SSD的响应时间随着负载的增加而增加。对于英特尔Optane固态硬盘，其曲线让人联想到系统内存的测量，响应时间一直很低，只有当负载使固态硬盘的带宽达到饱和时才会明显上升。

此外，图8显示英特尔Optane SSD的响应时间具有更高的QoS。即使在重载情况下，第99.999个百分点的响应时间仍然低于100μs。另一方面，对于nand固态硬盘，第99个百分点的响应时间超过1毫秒，而第99.999个百分点的响应时间接近10毫秒。

我们认为，我们绘制响应时间与吞吐量的方法，无论队列深度如何，包括高分位数异常值的QoS，是衡量和比较SSD性能的正确方法。事实上，系统内存性能在历史上已经被报告为

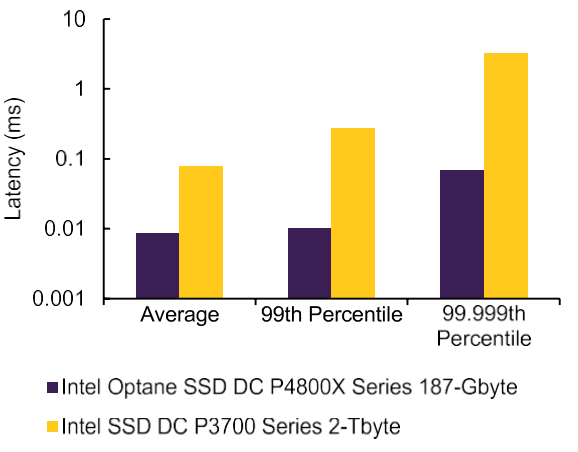


图7.*在随机分布的4kB IO的工作负载下，70%的读取和30%的写入，nand SSD和Intel Optane SSD的平均延迟和延迟QoS的比较。*

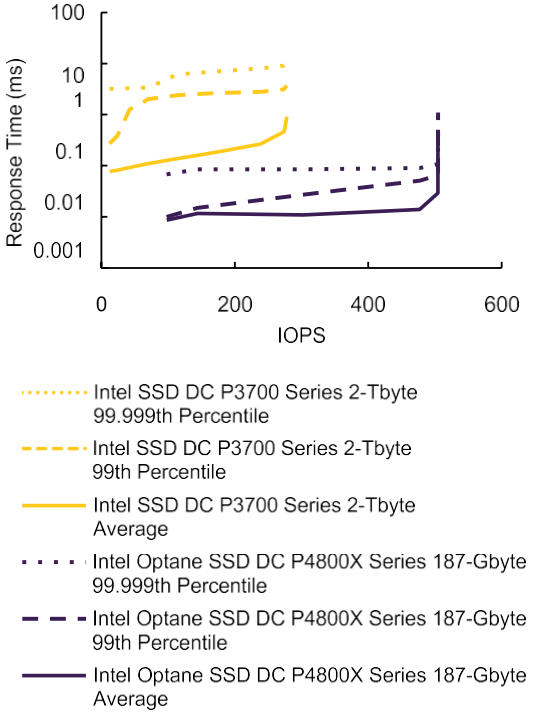


图8.*在给定的IOPS下，nand固态硬盘和英特尔Optane固态硬盘的平均响应时间、第99百分位数和第99.999百分位数的比较，给定的工作负载是随机分布的4KB IO，其中70%为读取，30%为写入。*

在给定负载下的响应时间[32]。我们敦促其他人在存储方面采用这种方法。测量响应时间与IOPS的关系将使系统专家和应用设计者能够直接衡量一个给定的固态硬盘的平均和异常响应时间将如何影响他们打算放在系统上的工作负载的性能。

V. 用例和参考应用

在了解了系统组件后，我们转而分析其对实际使用的影响。对于最初的英特尔Optane固态硬盘部署来说，两个最实用的使用模式是。更快的存储和DRAM扩展。

A. 为数据库提供更快的存储。RocksDB

在过去的十年中，已经出现了为高性能多核系统优化的数据库，其特点是基于Nand的存储。RocksDB[33]是这种数据库的一个开源例子。这个可嵌入的数据库使用日志结构化合并（LSM）树来提供一个高性能的可嵌入的键值存储。嵌入的LSM树驱动了应用程序的大部分存储级别的行为。在固态硬盘上，读往往是随机寻址的，而写一般是顺序寻址的，而且读比写多。满足一个用户对网页的请求需要许多键值的检索。在这样的存储

系统应该快速返回许多值，这意味着它必须为混合读/写工作负载提供高的IOPS描述，所以许多用户的请求可以用一个SSD快速满足。存储器还必须快速提供这些IO，并有很好的延迟一致性，所以个别用户不需要长时间等待响应。换句话说，固态硬盘应该为混合随机读取和连续写入工作负载提供高IOPS和出色的QoS。

在一个有两个英特尔至强处理器（2.5GHz，每个12个核心，启用英特尔超线程技术）、256GB DDR4、CentOS* Linux发行版7.2、使用XFS且不改变操作系统的系统中，对英特尔Optane SSD原型（英特尔Optane SSD DC P4800X的前身）与nand SSD（英特尔SSD DC P3600系列）进行了测试。启用了TRIM。英特尔SSD DC P3600系列被填充到50%的容量，而英特尔Optane SSD原型被填充到75%。RocksDB是根据rocksdb.org上公布的测试设置的：一个10亿个键的数据库，8个"碎片"，每个有2500万个键值对，20B个键，800B个值，50%的压缩，大约100GB的磁盘。读取。所有线程随机读取所有键。读/写。线程随机读取键值，而一个写者线程的更新速度达到约80 000键/秒。图9和图10中包含了体验式运行的结果，都显示了性能与RocksDB线程数量的关系。

用户吞吐量（图9）显示了英特尔Optane SSD原型的巨大优势。在这个实验中，线程的数量与核心的数量大致对应。英特尔Optane固态硬盘原型的低延迟导致只读工作负载的IOPS大幅提高，因为没有足够的未决访问来充分利用nand固态硬盘的芯片级参数。每分钟计算时间的耦合访问，而SSD的延迟导致了3倍的优势。在英特尔Optane SSD原型的吞吐量上。

与吞吐量同样重要的是用户响应时间。对于这个指标，必须考虑SSD的响应时间分布的尾部，所以图10显示了第99百分之百的响应时间。该图表中的延迟代表了对终端用户响应时间的一个组成部分。对于基于nand的固态硬盘来说，工作负载是一个特别困难的问题，因为在nand芯片上，写入需要很长的时间来完成，一些随机读取可能会发现自己在等待写入完成。3D XPoint内存完成写入的速度要快得多，所以英特尔Optane SSD原型机不会遭受那么多的碰撞惩罚。请注意，对于所有的线程数，英特尔Optane固态硬盘原型在1/10的时间内完成了请求。基于nand的固态硬盘所需的时间，导致了10倍的打赌。响应性。

RocksDB本身就是一个重要的应用，但它也是一类非常重要的应用的代表，即键值存储，它对后台连续写入时的随机读取延迟很敏感。如图所示，英特尔Optane固态硬盘原型在这个非常重要的工作负载中提供了相对于Nand固态硬盘的明显的吞吐量和响应性优势。

B. 用于实时分析的更快存储。Aerospike ACT

Aerospike*是一个企业级的NoSQL数据库，在编码时考虑了NVM和SSD[34], [35]。Aerospike的目标是关键任务、实时分析工作负载，如广告竞价和欺诈预防。这类用例在当前读写时有严格的响应时间要求。Aerospike提供了Aerospike认证工具（ACT）[36]，使用户有更好的方法来评价SSD的性能。ACT是专门为确保SSD设备能够维持高性能实时数据库所需的大量交易而设计的，同时又能通过实时应用的高QoS期望。虽然ACT是一个微观基准，而不是实际的Aerospike数据库，但它代表了Aerospike（企业）认为代表其数据库性能的特定工作负载特征。

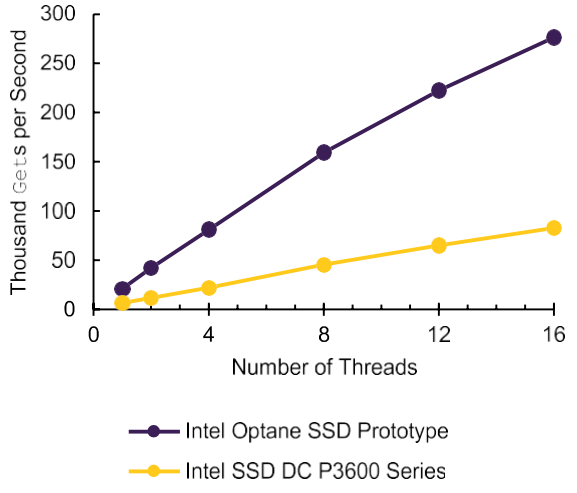


图9.*RocksDB获取操作的吞吐量。*

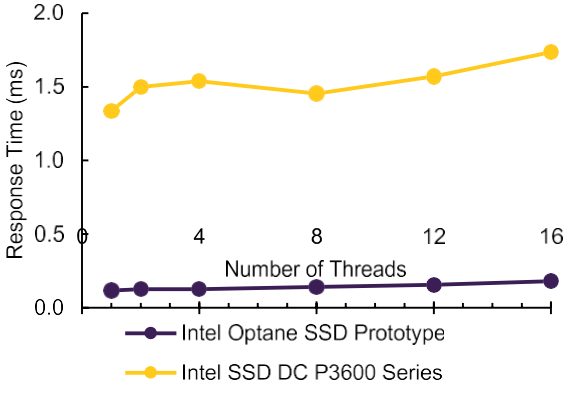


图10.*RocksDB获取操作的第99个百分点的响应时间。*

ACT执行大块（128-KB）读写和小块（1.5-KB）读写的组合，模拟标准的实时数据库读写负载。ACT测量小的读取请求（模拟数据库查询）的响应时间，这些请求在大的读写请求（模拟Aerospike的日志结构文件系统的后台垃圾收集过程）的情况下发生。基线ACT工作负载包括以下内容。

1) 1000 IOPS（1.536 MB/s）的随机分布的1.5kb的读数（测量）。
2) 1.536 MB/s的128KB读取。
3) 1.536 MB/s的128KB写入量。

ACT报告随机分布的读数的响应时间的QoS，每小时一次，持续48小时。为了获得合格的分数，在每个1小时的间隔内，以下内容必须成立。

1) 第95百分位数的反应时间≤1毫秒。
2) 第99百分位数的反应时间≤8毫秒。
3) 99.9百分位数的响应时间≤64毫秒。

ACT的工作负载增加了测量和背景IO请求的吞吐量，直到故障点为止，其方法是将工作负载乘数逐步应用到基线工作负载上。在比较固态硬盘的性能时，对给定的固态硬盘产生合格等级的最大倍数是衡量的标准，越大越好。

我们的测试比较了英特尔Optane SSD原型与基于nand的PCIe 1.6-TB英特尔SSD DC P3700系列和基于nand的SATA 400-GB英特尔SSD DC S3700系列的性能。[4]ACT被配置为使用8个队列，每个队列有8个线程，测试运行了2个小时。（官方测试需要48个小时；我们为了方便起见，使用了较短的运行时间，因为我们之前已经验证了运行的长度。

不影响所测试的SSD的结果）。用标准的

倍数，比基于nand的SSD提高了3.75倍（图11）。在这个倍数下，ACT已经有效地饱和了设备的吞吐量，同时保持了高服务质量。

我们发现，即使有ACT的不妥协要求------。它规定了最高级别的QoS期望，即第95百分位数的响应时间≤1毫秒。我们从我们的微观测试中知道，我们可以做得更好。我们收紧了在ACT的要求之外，进一步提高了QoS标准，即第95个百分点的响应时间≤128μs。有了这个新的标准，英特尔Optane固态硬盘比基于nand的固态硬盘的8倍工作负载乘数实现了125倍的工作负载乘数，产生了15倍的改进（图11）。

C. 为Memcached扩展DRAM

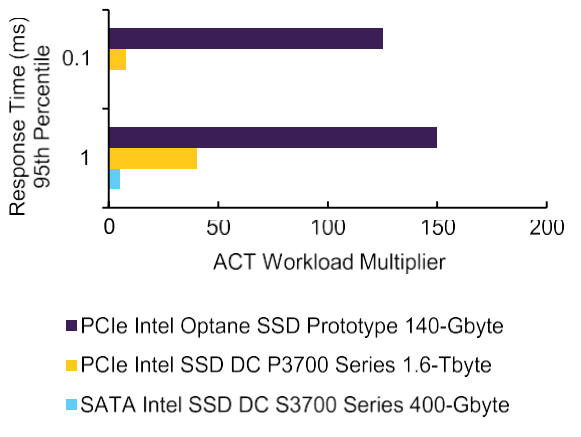
许多现代的应用程序在编写时都期望整个数据集都能放在内存中。Amdahl内存定律[17]认为，支持1 MIPS所需的DRAM的兆字节数，即α，是1。Gray和Shenoy[37]观察到，对于新的以数据为中心的工作负载，需要将α修改为4-10。一个现代系统将需要超过1TB的DRAM。我们从早期的分页微观测试中推断出，分页是一个具有低延迟SSD的引人注目的选择。

为了确定选定的参考应用程序是否能获得这样的好处，我们选择了memcached，一个被广泛部署的开源内存中键值分布式缓存[38]。Memcached可以在网络上访问一个缓存服务器。我们使用了我们认为是最好的情况下的工作负载--随机的4KB的获取命令（读取）。

经验性的测量结果显示在图12中。由于这些测量是在英特尔Optane固态硬盘出现之前进行的，所以我们用一个非生产性的基于DRAM的固态硬盘来代替。

ACT测试，英特尔Optane固态硬盘达到了150倍的工作负荷。

[4]在CentOS Linux 7.1.1503版、英特尔酷睿i7处理器4770、华硕* H87I-PLUS主板和4-GB DDR3内存上测量。



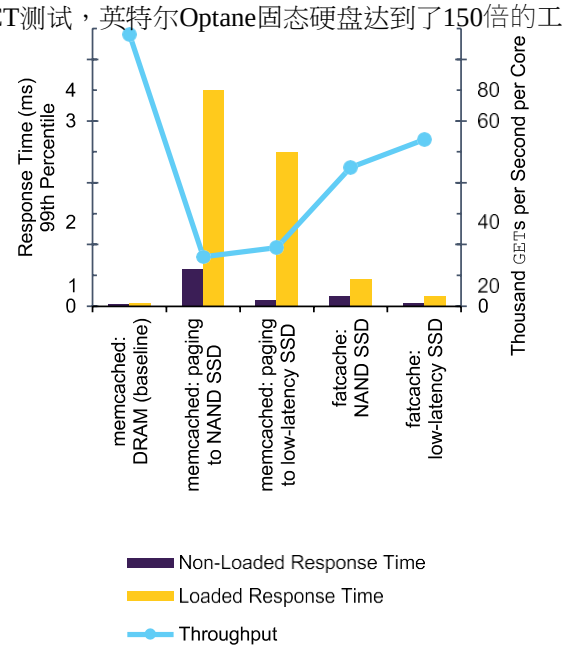


图11. *三个固态硬盘的最高ACT工作负载乘数，给定的第95百分位响应时间为1和0.1ms。*

图12.*memcached和fatcache在不同设备上的响应时间和吞吐量表现。*

NVMe SSD的延迟与英特尔Optane SSD相似，同时保持相同的系统软件和硬件开销。我们对两者的性能进行了比较，鉴于Optane SSD的田园诗般的响应曲线，我们相信结果仍然适用。

分页到快速存储的响应时间要长45倍比起使用DRAM的memcached，不到20%的吞吐量。把。[5]这种次优的性能让我们感到惊讶。从根源上讲，我们发现这个问题是由于通用的、不知道上下文的操作系统分页造成的。由于memcached的编写考虑到了DRAM，memcached在每次访问时都会更新缓存统计数据（时间数据结构，如最后修改时间），并进行小规模的写入，用于memcached的获取指令（读）或设置指令（写）。因此，页面被弄脏了，必须被写回去。即使是在memcached的读取中也是如此。

Memcached代表了一类需要修改代码才能真正利用分页的低延迟存储的应用。我们需要一个与数据分离的时间性数据结构的应用程序。我们发现了一个memcached的开源变体，叫做fatcache，是以SSD为基础编写的[39]。Fatcache将其元数据和数据组织成独立的存储块大小的单元桶，它管理自己的磁盘IO，而不是依赖操作系统提供的分页。使用低延迟的SSD，fatcache至少可以达到memcached单独使用DRAM的60%的吞吐量。

脂肪缓存仍然会产生存储软件的处理费用。我们使用单个核心来精确测量这种影响，同时期望使用多个实例的Fatcache，在多个核心上并行运行，将进一步扩大吞吐量。脂肪缓存在未加载的系统上会产生额外的10μs的延迟，而在99%的百分位上会产生额外的110μs的延迟。鉴于数据中心的网络往返时间从300μs到1.2ms不等，我们认为分布式缓存的工作负载可以容忍额外的延迟。对于这个用例，低延迟的固态硬盘是一个可行的、比额外的DRAM更便宜的选择，可以为应用提供更大的数据集大小。

VI. 未来的机会

即使到目前为止，许多人做了积极的系统优化，平台存储堆栈的硬件/软件仍然为访问3D XPoint内存的时间带来了微秒级的延迟。此外，存储访问的最小颗粒度为512 B（实际中通常为4 kB），当访问只需要小数据量时，进一步增加了延迟。通过对系统硬件和软件系统的额外改变，实现持久的内存使用模式，3D XPoint内存还有更多性能优势。

虽然NVDIMM设备一直是嵌入式存储用例的主流，例如，作为存储控制器的写缓存或较大存储时代引擎的快速日志设备，它们在一般用途的存储应用中没有得到广泛部署。小容量和电池管理的物流使目前的NVDIMMs被限制在小众的使用情况下。基于3D XPoint技术的英特尔内存模块有望消除一般用途案例的障碍。然而，在充分实现持久性内存的容量和延迟优势之前，需要对操作系统的传统存储栈进行重大改变。这与已经引入的低延迟存储的变化是直接平行的。这次的目标是使持久性存储器能够只用一到两条指令就能将数据传送到媒体上。Linux已经实现了DAX，开发管道的进一步改进将使应用程序在通往持久性存储器设备的IO路径上避免操作系统的开销[40]。

消除或摊销与文件系统反复协调以提交DAX功能文件更新的开销，是一个正在进行的开发课题。一个将文件系统从方程中完全移除的机制是Device-DAX。Device-DAX把一整卷的持久性内存，一个通常可以容纳整个文件系统的容量范围，变成一个可以被DAX映射的设备专用文件。这给了应用程序对内存的完全所有权，并且不需要通知操作系统内核或文件系统的新更新。然而，这种水平的原始访问和责任只适合于一小部分专门的应用程序。将Device-DAX的旁路功能引入基于DAX文件系统的文件的工作仍在继续。我们相信其他主要的操作系统会提供类似的功能。

领先的应用程序可以迅速利用这些功能，而其他数据中心的应用程序可能需要一段时间才能准备好使用全部功能。子扇区大小的字节对齐的读和写，可以在几条指令中提交，改变了存储的传统假设。研究人员已经显示出显著的优势。Oukid等人[41]重写了一个内存数据库，使其具有持久性内存意识。对于他们的应用，他们表明持久性存储器的应用可测量的性能接近于DRAM。他们进一步揭示了在断电情况下明显缩短停机时间的好处。其他研究人员[42]、[43]通过在应用程序中加入持久性内存的意识，在从键值存储、排序和连接以及事务性日志等使用案例中暴露出明显的性能优势。操作系统将继续成熟他们的持久性内存访问机制，使新一代的数据中心应用成为可能。

[5]在一个双插槽2.9-GHz的英特尔®至强®E5-2690上测量的。平台。

VII. 结论

基于新技术3D XPoint内存的超高速英特尔Optane固态硬盘已经进入市场。这些固态硬盘提供的数据延迟可与系统本身相媲美，颠覆了我们几十年来认为存储是最慢的系统组件的看法。系统已经发生了深刻的变化，包括将固态硬盘转移到PCIe，取消HBA，以及调整操作系统的代码路径和策略，为这一进步铺平道路。随着这些变化的到位，这些新的固态硬盘所提供的性能可供系统使用。

低延迟的存储访问模糊了存储和内存之间的界限。基于固态硬盘队列深度的测量不再有意义，因为大部分IO的寿命是在系统中度过的，而不是在存储设备中。固态硬盘应该像内存一样用带宽（IOPS）与延迟的测量方法来衡量。长期以来一直避免的分页存储，现在是一个可行的策略。当一个4KB的页面只有16μs时，许多应用程序可以通过现有的操作系统分页，像使用额外级别的内存一样使用存储。因此，这些新的高性能固态硬盘不只是像内存一样被测量，而且还可能像内存一样被使用。

基于应用的研究表明，英特尔Optane固态硬盘的性能是可以通过应用获得的。RocksDB。一个重要的键值存储，显示了3倍的改进当在英特尔Optane固态硬盘上运行时。实时分析ACT基准测试也显示，在英特尔Optane固态硬盘上运行时，性能优势超过3倍。在这两种情况下更低的延迟，以及负载下更理想的性能（更好的QoS）是这些系统级性能优势的原因。此外，现在为解决方案的创新敞开了大门，以利用实现高吞吐量和提供出色的QoS的能力，这是基于Nand的SSD所不可能做到的。

当用作分页存储时，英特尔Optane固态硬盘再次带来了类似的应用性能，但却带来了另一个关键优势--数据集大小的增加。这种效果在名为fatcache的memcached变体和内存分析应用中都有体现。在这些例子中，固态硬盘是快速的，系统是快速的，而应用程序在运行时间减少或数据集大小增加中受益。

虽然很重要，但快速的SSD并不是这项技术的终点。操作系统已经在进行必要的改变，通过DAX模式的创建将新的内存技术暴露为持久性内存。应用程序将受益于传统的文件系统与持久性内存文件系统的使用，并通过直接使用持久性内存获得最大利益。新的快速内存的出现为系统架构创造了一个令人兴奋的时刻，在近期内有机会利用快速的固态硬盘来获得直接的系统级应用性能优势，并通过持久性内存获得更大的好处。

鸣谢

这项工作是许多人的成果，特别是英特尔的SSD团队，作者也是该团队的一员。作者要特别感谢A. Fazio的领导和J. Smits。W.Fang, S. Vyas, R. Medel, S. Mehta, K. Putnam, J. Tang, and A. Torok提供了本文中引用的许多实验和测量。最后，他们要感谢那些慷慨地与他们分享工作负载的客户。Intel、Intel Core、Intel Xeon、Intel Optane和3D XPoint是Intel公司或其子公司在美国和/或其他国家的商标。其他名称和品牌可能被认为是其他人的财产。

参考文献

[1] (2016年5月17日)。*IBM的科学家们实现了存储记忆的突破*。[在线]。Available: https://www-03.ibm.com/press/us/en/pressrelease/49746.wss

[2] C.Nguyen, D. Burkard, K. Dobbins, and C.Bohac（2016年8月5日）。*MRAM对汽车非易失性存储器存储的改进*。[在线]。Available: https://www.everspin.com/file/1101/download

[3] A.Shilov.(Aug. 12, 2016).*西部数据将使用3D ReRAM作为特殊用途SSD的存储类内存*。[在线]。Available: http://www.anandtech.com/show/10562/western-digital-to-use-3d-reram-as-storage-class-memory-for-specialpurpos-ssds

[4] Crucial。*Crucial NVDIMMs。强大而持久的服务器内存性能*。[在线]。Available: http://www.crucial.com/usa/en/memory-server-nvdimm

[5] V.Prabhakaran, T. L. Rodeheffer, and L.Zhou, "Transactional flash," in *Proc. 8th USENIX Conf.操作。Syst.设计。实施。(OSDI)*，美国加州圣地亚哥，2008。

[6] J.Condit等人，"通过字节寻址的持久性内存实现更好的I/O"，在*Proc. 22nd ACM Symp.Oper.Syst.原则（SOSP）*。Big Sky, MT, USA, 2009, pp.133-146.

[7] H.Volos, A. J. Tack, and M. M. Swift, "Mnemosyne: Lightweight persistent memory," in *Proc. 15th ACM Int.Conf.Archit.支持程序。Lang.操作。Syst.(ASPLOS)*, Newport Beach, CA, USA, 2011, pp.91-104.

[8] C.Hellwig, R. Zwisler, and B. Harrosh (Apr. 1, 2015).*[PATCH 2/2] pmem。添加一个持久性内存的驱动*。[在线]。可用：https://lwn.net/Articles/640114/

[9] X.Wu and A. L. N. Reddy, "SCMFS: A file system for storage class memory," in *Proc. Int. Conf.Conf.High Perform.Comput., Netw., Storage Anal.(SC)*, Seattle, WA, USA, 2011, pp.

[10] R.Zwisler, V. Verma, S. Kumar, M. Wilcox, and R. Gittins *Persistent Memory File System*.[在线]。可用：https://github.com/linux-pmfs/pmfs

[11] S.R. Dulloor等人，"用于持久性内存的系统软件"，在*Proc.9Eur.Conf.Comput.Syst.(EuroSys)*, Amsterdam, The Netherlands, 2014, p. 15.

[12] A.Kudryavtsev.(2016年2月1日)。*SSD作为系统内存？是的，用ScaleMP的*

技术。[在线]。Available: https://storagebuilders.intel.com/blog/ssd-as-a- system-memory-yes-with-scalemps- technology-2

[13] A.Rudoff，"新兴非易失性存储器技术的编程模型"，*Login*，第38卷，no.3, p. 39-45, 2013.

[14] J.Corbet.(2015年4月15日)。*持久性记忆支持的进展*。[在线]。Available: https://lwn.net/Articles/640113/.

[15] A.Rudoff.*NVM图书馆*。[在线]。Available: http://pmem.io/nvml/

[16] A.P. Foong, B. Veal, and F. T. Hady, "Towards SSD-ready enterprise platforms," in *Proc. 1st Int.Workshop Accel.数据管理。系统。Using Modern Processor Storage Archit.(ADMS)*，新加坡，2010年，第15-21页。

[17] J.L. Hennessy和D. A. Patterson, *计算机结构。A Quantitative Approach*，5th ed.美国加州旧金山：Morgan Kaufmann，2011。

[18] Z.Kerekes.(2007).*SSD市场历史*。[在线]. Available: http://www.storagesearch.com/ssd-history-2007.html

[19] NVM Express.[在线]。Available: http://nvmexpress.org/

[20] M.Bjøling, J. Axboe, D. Nellans, and P. Bonnet, "Linux block IO: introducing multi queue SSD access on multi-core systems," in *Proc. 6th ACM Int.Syst.存储会议（SYSSTOR）*，以色列海法，2013，第22页。

[21] J.Yang, D. B. Minturn, and F. Hady, "When poll is better than interrupt," in *Proc. 10th USENIX Conf.File Storage Technol.(FAST)*，美国加州圣何塞，2012，第3页。

[22] C.梅森。(2017年3月8日)。*Btrfs*.[在线]。可用：https://btrfs.wiki.kernel.org/index.php/Main_Page

[23] 甲骨文。(2010).*Oracle Solaris ZFS管理指南*。[在线]。Available: http://docs.oracle. com/cd/E19253-01/819-5461/.

[24] P.Sehgal, S. Basu, K. Srinivasan, and K.Voruganti, "An empirical study of file systems on NVM," in *Proc. 31 IEEE Symp.Mass Storage Syst.Technol.(MSST)*, Santa Clara, CA, USA, May/Jun. 2015, pp.1-14.

[25] J.Xu and S. Swanson, "NOVA: A log-structured file system for hybrid volatile/non-volatile main memories," in *Proc. 14 USENIX Conf.File Storage Technol.*, Santa Clara, CA, USA, 2016, pp.323-338.

[26] J.Yang.(2016年12月23日)。*Pmbench*.[在线]。Available: https://bitbucket.org/jisooy/pmbench

[27] T.Chen.(2016年10月20日)。*mm/swap: 常规页面交换的优化*。[在线]。Available: https://lwn.net/Articles/704359/

[28] J.Axboe.(2017年3月13日)。*fio HOWTO*。[在线]。Available: https://github.com/axboe/fio/blob/master/HOWTO

[29] 开放源码开发实验室。*Iometer*.[在线]. Available: http://iometer.org/

[30] C.Ramseyer.(Mar. 14, 2015).*我们如何测试HDD和SSD*。[在线]。Available: http://www.tomshardware.com/reviews/how-we-test-storage.4058.html

[31] K.Grimsrud, "IOPS schmIOPS!SSD性能中真正重要的是什么"，在*Proc.Flash Memory Summit*, Santa Clara, CA, USA, 2013.

[32] B.雅各布，*《记忆系统》。你不能避免它，你不能忽视它，你不能伪造它*。美国加州圣拉斐尔：摩根和克莱普，2009年。

[33] D.Borthakur.(Nov. 21, 2013).*引擎盖下：构建和开源RocksDB*。[在线].Available: https://www.facebook.com/notes/facebook-engineering/under-the-hood building and-open-sourcing-rocksdb/10151822347683920/.

[34] Aerospike.(2016年4月12日)。*白皮书。为关键任务、实时应用构建企业级数据库架构*。[在线]。Available: http://www.aerospike.com/resource/white-paper-building-an-enterpris-grade-database-architecture-for-mission-critical-real-time-applications/.

[35] B.Bulkowski，"NVMe，存储类内存和操作数据库。Real-world results," in *Proc.Flash Memory Summit*, Santa Clara, CA, USA, 2016.

[36] Aerospike。*Aerospike认证工具（ACT）*。[在线]。Available: http://www.aerospike.com/act-for-ssds/

[37] J.Gray和P. Shenoy，"数据工程中的经验法则"，在*Proc.16 IEEE Int.Conf.数据工程（IDCE）*，美国加州圣地亚哥，2000年3月，第3-10页。

[38] B.Fitzpatrick.*Memcached*.[在线]. 可用：http://memcached.org/

[39] M.Rajshekhar and Y. Yue (2013). *fatcache*.[在线]。Available: https://github.com/ twitter/fatcache

[40] D.Williams和T. Kasanicky，"管理持久性内存"，在*Proc.Vault Linux Storage Filesyst.Conf. (Vault)*, Raleigh, NC, USA, 2016.

[41] I.Oukid, D. Booss, W. Lehner, P. Bumbulis, and T.Willhalm, "SOFORT: A hybrid SCM-DRAM storage engine for fast data recovery," in *Proc. 10th Int.讲习班数据管理。(DaMoN)*, Snowbird, MT, USA, 2014, p. 8.

[42] S.D. Viglas, "Write-limited sorts and joins for persistent memory," *Proc.VLDB Endowment*, vol. 7, no.5, pp.413-424, 2014.

[43] J. Huang, K. Schwan, and M. K. Qureshi, "NVRAM-aware logging in transaction systems, " *Proc.VLDB Endowment*, vol. 8, no.4, pp. 389-400, 2014.

关于作者

Frank T. Hady在美国弗吉尼亚大学夏洛茨维尔分校获得电气工程学士和硕士学位，在美国马里兰大学学院公园分校获得电气工程博士学位。

他是英特尔研究员和美国俄勒冈州希尔斯伯勒市英特尔公司非易失性存储器解决方案组（NSG）的存储技术组主任。他的团队研究未来的固态硬盘和系统级存储的进展。

他负责确定英特尔固态硬盘和其他存储产品的架构，并与合作伙伴共同创建行业标准。他曾是NSG的3D XPoint**存**储的首席架构师，领导英特尔Optane SSD**的架构定**义和快速存储的系统进展。他在网络、存储和I/O创新方面发表了30**多篇**论文，并持有30**多份文件。**美国专利。

Annie Foong在美国威斯康星大学麦迪逊分校获得计算机工程博士学位。

她于1999年加入位于美国俄勒冈州希尔斯伯勒的英特尔公司，目前是NVM解决方案组的首席工程师。她是英特尔Optane数据中心SSD的首席架构师。

Bryan Veal于2005年在美国乔治亚州雅典的乔治亚大学获得了计算机科学硕士学位。

他于2005年加入美国俄勒冈州希尔斯伯勒市的英特尔公司，目前是英特尔Optane数据中心SSD的平台架构师。

丹-威廉姆斯于2002年加入位于美国俄勒冈州Hills- boro的英特尔公司，目前是开源技术中心的一名软件工程师。他是一个Linux内核的开发者，实现了下一代的存储技术和持久性内存。他也是美国国家航空航天局的成员之一。

Linux基金会技术顾问委员会。