# GPUDirect 存储器：存储器和 GPU 存储器之间的直接路径

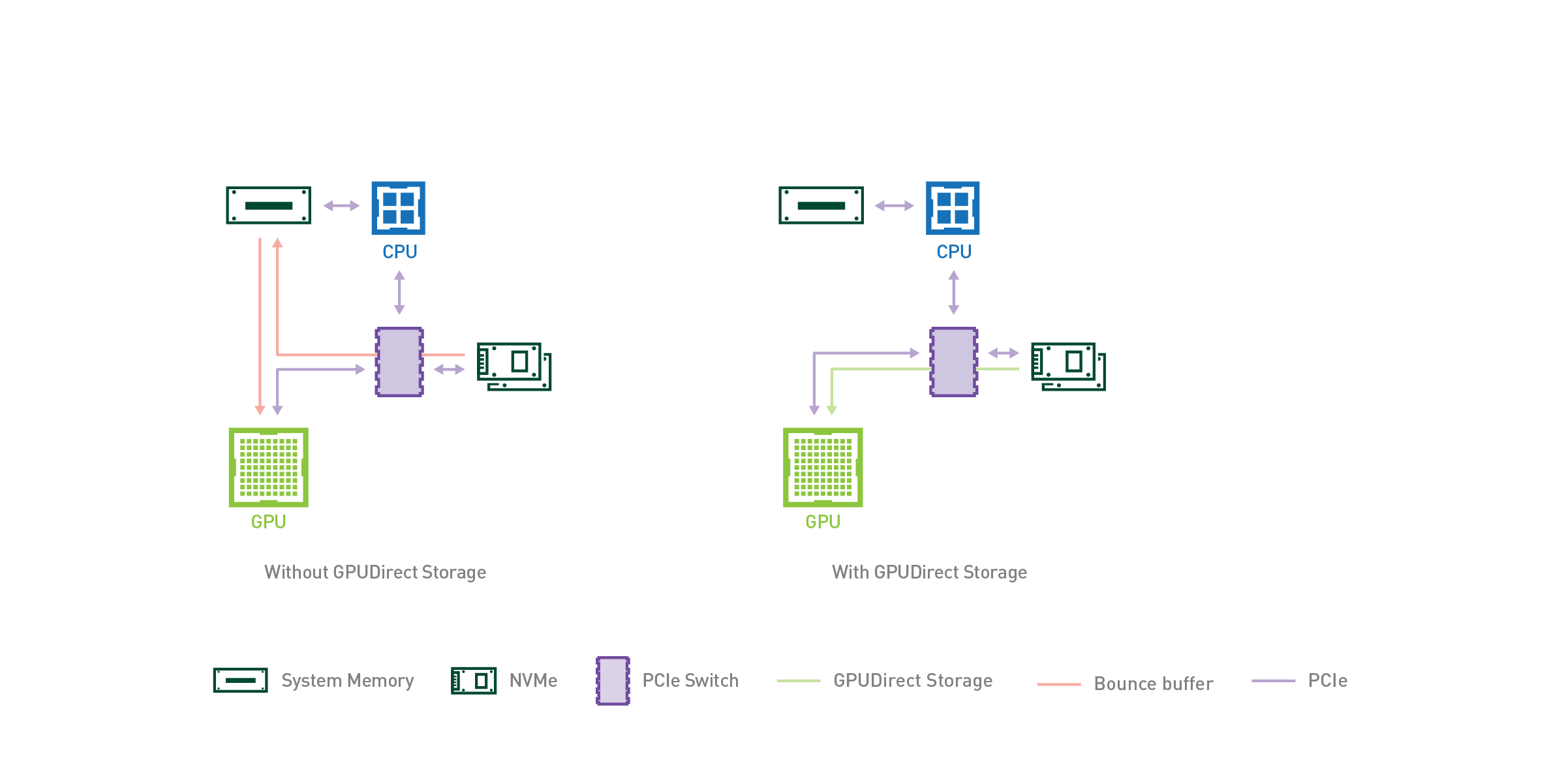
https://developer.nvidia.com/zh-cn/blog/gpudirect-storage/

## 保持 GPUs 忙碌

随着 AI 和 HPC 数据集的大小不断增加，加载给定应用程序的数据所花费的时间开始对整个应用程序的性能造成压力。在考虑端到端应用程序性能时，快速 GPUs 越来越缺乏慢 I / O 。

I / O ，将数据从存储器加载到 GPUs 进行处理的过程，历史上一直由 CPU 控制。随着计算速度从较慢的 CPU 转移到更快的 GPUs，I/O成为整个应用程序性能的瓶颈。

正如GPUDirect-RDMA（远程直接存储器地址）在网络接口卡（ NIC ）和 GPU 内存之间直接移动数据时提高了带宽和延迟，一种称为 GPU 直接存储的新技术使本地或远程存储（如 NVMe 或 NVMe over Fabric ， NVMe oF ）和 GPU 内存之间实现了直接数据路径。 GPU 直接 RDMA 和 GPU 直接存储器都避免了通过 CPU 内存中的反弹缓冲区的额外拷贝，并使 NIC 或存储器附近的直接内存访问（DMA）引擎能够在直接路径上将数据移入或移出 GPU 内存—所有这些都不会给CPU或GPU带来负担。如图 1 所示。对于 GPU 直接存储，存储位置无关紧要；它可以在机柜内、机架内或通过网络连接。在 CPU DGX-2 中，从NVIDIA系统内存（SysMem）到 GPUs 的带宽被限制为 50gb / s ，而来自 SysMem 、许多本地驱动器和许多 NICs 的带宽可以组合起来，从而在 DGX-2 中达到近 200gb / s 的带宽上限。

在本博客中，我们将扩展到一个[上一篇文章](https://news.developer.nvidia.com/making-gpu-i-o-scream-on-platforms-of-today-and-tomorrow/)演示 GPU 直接存储：一个概念证明，可以通过 NVMe 从位于给定服务器的本地存储或机柜外部的存储器直接访问内存（ DMA ）。我们证明了从存储器到 GPU 的直接内存访问缓解了 CPU I / O 瓶颈，并提高了 I / O 带宽和容量。此外，我们根据 [RAPIDS](https://rapids.ai/) 项目的 GPU – 加速 [CSV 阅读器](https://rapidsai.github.io/projects/cudf/en/0.8.0/api.html#module-cudf.io.csv) 提供了在圣何塞 GTC19 上展示的初始性能指标。最后，我们将提供一些关键应用程序的建议，这些应用程序可以利用更快和更高的带宽、更低的延迟和更大的存储与 GPUs 之间的容量。

在以后的文章中，当这个特性接近产品化时，我们将描述如何对它进行编程。一组新的 cuFile API 将被添加到 CUDA 中，以支持这个特性，并将本机集成到 RAPIDS ‘ cuDF 库中。

## 直接内存访问是如何工作的？

PCI Express （ PCIe ）接口将高速外围设备（如网卡、 RAID / NVMe 存储设备和 GPUs 连接到 CPU s 。用于 Volta GPUs 的系统接口 PCIe Gen3 可提供 16 GB / s 的聚合最大带宽。一旦将协议头的低效率和其他开销考虑在内，最大可达数据速率超过 14gb / s 。

直接内存访问（ DMA ）使用复制引擎在 PCIe 上异步移动大数据块，而不是加载和存储。它卸载了计算元素，让它们可以自由地进行其他工作。在 GPUs 和存储相关设备（如 NVMe 驱动程序和存储控制器）中有 DMA 引擎，但通常在 CPU 中没有。在某些情况下， DMA 引擎无法针对给定的目标进行编程；例如， GPU DMA 引擎不能以存储为目标。没有 GPU 直接存储，存储 DMA 引擎无法通过文件系统以 GPU 内存为目标。

然而， DMA 引擎需要由 CPU 上的驱动程序编程。当 CPU 对 GPU 的 DMA 进行编程时，从 CPU 到 GPU 的命令可能会干扰到 GPU 的其他命令。如果可以使用 NVMe 驱动器或存储附近其他地方的 DMA 引擎来移动数据，而不是使用 GPU 的 DMA 引擎，那么 CPU 和 GPU 之间的路径就没有干扰。与 GPU 的 DMA 引擎相比，我们在本地 NVMe 驱动器上使用 DMA 引擎将 I / O 带宽提高到 13.3 GB / s ，相对于下表 1 所示的 12 . 0 GB / s 的 CPU 到 GPU 内存传输速率，性能提高了大约 10% 。

## 缓解 I / O 瓶颈及相关应用

随着研究人员将数据分析、人工智能和其他 GPU 加速应用程序应用于越来越大的数据集，其中一些数据集将无法完全放入 CPU 内存甚至本地存储，因此，缓解存储和 GPU 内存之间的数据路径上的 I / O 瓶颈将变得越来越重要。数据分析应用程序对大量数据进行操作，这些数据往往从存储中流入。在许多情况下，计算与通信的比率（也许用每字节的 flops 表示）非常低，这使得它们受到 IO 限制。例如，为了使深度学习能够成功地训练神经网络，每天要访问许多组文件，每个文件的大小为10MB，并多次读取。在这种情况下，优化数据传输到 GPU 可能会对训练人工智能模型的总时间产生重大而有益的影响。除了数据摄取优化之外，深度学习培训还经常涉及检查点的过程，即在模型训练过程的各个阶段，将训练好的网络权重保存到磁盘上。根据定义，检查点位于关键 I / O 路径上，减少相关开销可以缩短检查点周期和加快模型恢复。

除了数据分析和深度学习之外，研究网络交互的图形分析还有很高的 I / O 需求。当遍历一个图来寻找有影响的节点或从这里到那里的最短路径时，计算只占总求解时间的一小部分。从当前节点开始，确定下一步要去哪里，可能涉及来自一个 PB 大小的数据湖的 1 到数百个文件的 I / O 查询。虽然本地缓存有助于跟踪可直接操作的数据，但图形遍历对延迟和带宽都很敏感。随着 NVIDIA 通过[cuGraph 库](https://github.com/rapidsai/cugraph) RAPIDS 扩展了 GPU 图形分析加速功能，消除文件 I / O 开销对于继续提供光速解决方案至关重要。

## 将存储和带宽选项扩展到 GPUs

数据分析和人工智能之间的一个共同主题是，用于获取见解的数据集通常是海量的。NVIDIA DGX-2 由 16 个 Tesla V100 组成，包含 30TB NVMe SSD 内存（ 8x 3 . 84TB ）和 1.5TB 系统内存的库存配置。启用驱动器的 DMA 操作允许快速访问内存，同时增加带宽、降低延迟和潜在的无限存储容量。

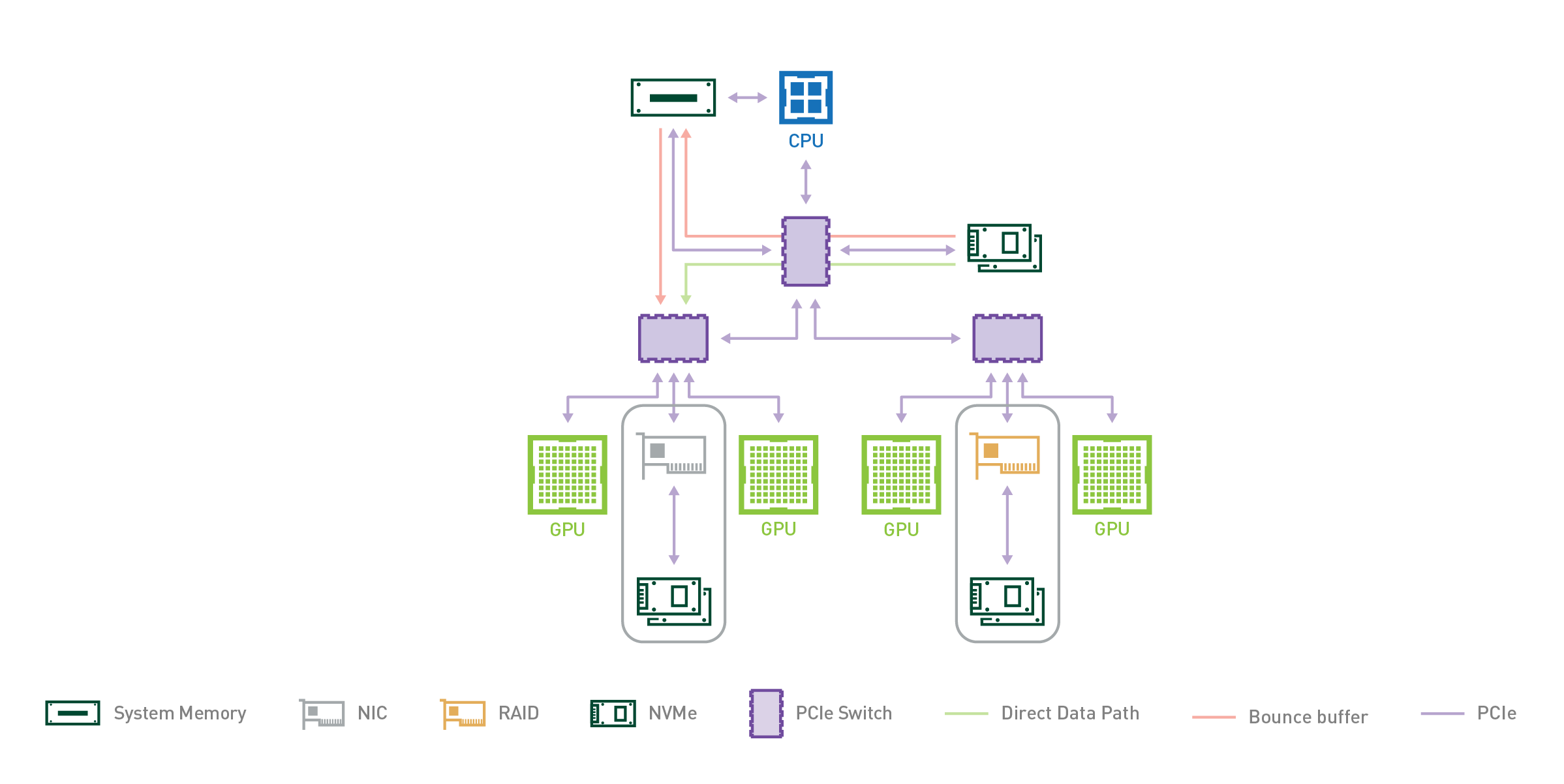


图2 ：从存储模块外部获得更多带宽和更多存储空间。远离 PCI 交换机的 NIC 支持 NVMe-oF 的远程存储扩展，而RAID卡支持附近的存储。所示的 RAID 卡仅为原型，并不表示当前或未来的DGX-2产品。

DGX-2 机柜包含两个 CPU ，每个 CPU 都有两个 PCIe 子树实例，如图 2 所示。从存储器或 SysMem 到 GPUs 的多条 PCIe 路径由两个级别的 PCIe 交换机支持，这使得 DGX-2 成为 GPU 直接存储原型化的良好测试工具。表 1 的左列列出了向 GPU 传输数据的各种来源，第二列列出了从该源测得的带宽，第三列标识了此类路径的数量，最后一列是中间两列的乘积，显示了该类源可用的总带宽。对于 4 个 PCIe 树（ 12-12 . 5 GB / s ）中的每一个，从 CPU 的系统内存（ SysMem ）有一条路径，另一条路径来自每个 PCIe 树上挂起的¼个驱动器的另一条路径，速度为 13.3 GB / s 。 DGX-2s 每对 GPUs 都有一个 PCIe 插槽。该插槽可以由一个NIC占用，该NIC的测量速度为 10 . 5 GB / s ，或者，在本博客中使用的原型中，可以使用 RAID 卡，其测量速度为 14 GB / s 。 NVMe of （over fabric）是一种通用协议，它使用 NIC 访问远程存储，例如通过Infiniband网络。如果在8个PCIe插槽中使用RAID卡（图 2 中每个PCIe子树2个），则在所有源上添加的PCIe带宽的右侧列总和为215 GB/s；如果在这些插槽中使用NIC，则总和会更低。

GPU直接存储的一个主要优点是，无论是存储在存储模块内部还是外部、系统内存或 NVMe 驱动器上的快速数据访问都是跨各种源的累加。使用内部 NVMe 和系统内存并不排除使用 NVMe 或 RAID 存储。最后，这些带宽是双向的，支持复杂的编排，其中数据可以从分布式存储中引入，缓存在本地磁盘中，并且可以通过在 CPU 系统内存中共享的数据结构与 CPU 协作，总带宽超过 GPU峰值IO的90%。对这三个源中的每一个的读写操作可能同时发生。图 3 对各种来源进行颜色编码，并将相加组合显示为堆叠条形图。在下面的列标签中，源实例的数量在括号中，例如16个NVMe驱动器或8个执行NVMe操作的 NIC 。每个选项可用的粗略容量显示在列标签的最后。

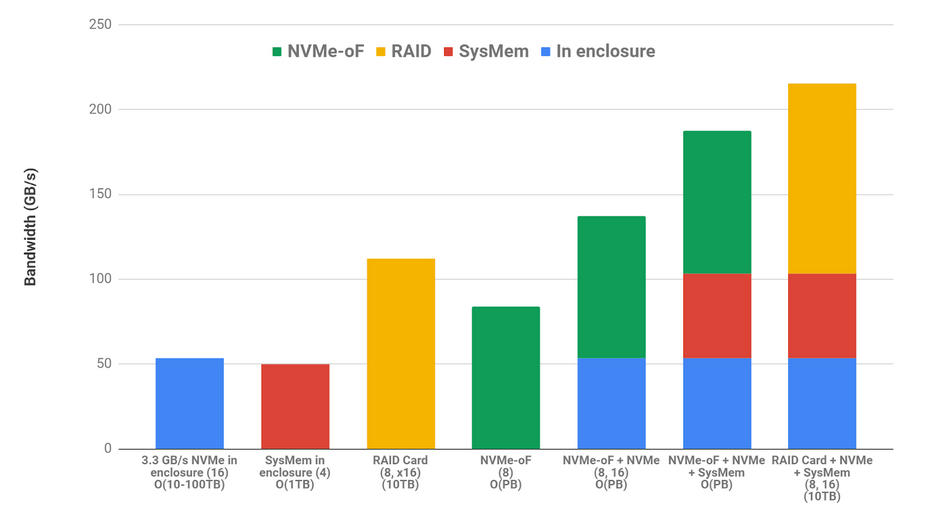


图 3 ：来自不同来源的带宽限制是累加的

## 数据操纵案例研究

从存储器到GPU的直接路径也适用于不完全适合GPU帧缓冲区的数据集。在一个实验中，NVIDIA 使用了1TB的输入数据集和DGX-2的512GB聚合 GPU 内存，用GPU直接存储来证明，即使在内存超额订阅的情况下，16GPUs 的数据I/O速度也比主机内存快。直接读取和写入数据的速度提高了2倍，但分块、使用更小的批处理和其他优化进一步提高了速度。总的来说， GPU 直接存储将数据操作速度提高了4.3倍。

## GPU 直接存储的值

GPU 直接存储器提供的关键功能是，它使 DMA 能够通过这个文件系统从存储器到 GPU 存储器。它以多种方式提供价值：

* 2-8 倍的带宽，直接在存储器和 GPU 之间传输数据。
* 显式的数据传输既不出错也不经过跳出缓冲区，也具有较低的延迟；我们演示了低3.8倍的端到端延迟的示例。
* 避免显式和直接传输的错误可以使延迟在 GPU 并发性增加时保持稳定和平坦。
* 在存储器附近使用 DMA 引擎对 CPU 负载的影响较小，并且不会干扰 GPU 负载。使用更大尺寸的 GPU 直接存储，带宽与部分 CPU 利用率之比要高得多。我们观察到（但没有在本博客中以图形方式显示）当其他 DMA 引擎将数据推入或拉入 GPU 内存时， GPU 利用率仍然接近于零。
* GPU 不仅成为带宽最高的计算引擎，而且成为 IO 带宽最高的计算单元，例如 215 GB / s ，而 CPU 的 50 GB / s 。
* 无论数据存储在何处，所有这些好处都是可以实现的——实现对 PB 级远程存储的快速访问，甚至比 CPU 内存中的页缓存都要快。
* 从 CPU 存储器、本地存储器和远程存储器进入 GPU 存储器的带宽可以相加地组合起来，使进入和流出 GPUs 的带宽几乎饱和。这变得越来越重要，来自大型分布式数据集的数据被缓存在本地存储器中，工作表可以缓存在 CPU 系统内存中，并与 CPU 协同使用。

除了使用 GPUs 而不是 CPU 加快计算的好处外，一旦整个数据处理管道转移到 GPU 执行，直接存储就起到了一个力倍增器的作用。这一点变得尤为重要，因为数据集大小不再适合系统内存，而且 GPUs 的数据 I / O 增长成为处理时间的瓶颈。当人工智能和数据科学继续重新定义可能的艺术时，启用直接路径可以减少甚至完全缓解这个瓶颈。

## 致谢

我们非常感谢 GPU Direct Storage 团队对这项工作的体系结构、设计、评估和调整所做的杰出贡献，这些工作为本内容提供了基础，包括 Kiran Modukuri、Akilesh Kailash、Saptarshi Sen，还有 Sandeep Joshi 。特别感谢尼古拉·萨哈尼赫和芮兰对 TPC-H 的帮助。

我们要感谢以下同行为 GPU Direct Storage 提供原型制作和初始数据收集的帮助：

* 我们使用 Micron 9200 MTFDHAL3T8TCT 3 . 84TB 和三星 MZ1LW960HMJP 960GB NVMe 驱动器进行本地存储。
* 存储模块内部或外部可能有一个存储控制器，用于执行 RAID 0 、 RAID 5 和本地化控制。我们使用了来自 MicroChip 的 SmartROC 3200PE RAID 控制器、 anHPE 智能阵列 P408i-a SR Gen10 12G SAS 模块化控制器和 HPE 智能阵列 P408i-p SR Gen10 12G SAS PCIe 插件控制器。
* 可以通过结构启用存储。我们使用 Mellanox MCX455A-ECAT ConnectX4 NIC和E8存储解决方案以及4个Mellanox ConnectX5。