DAOS:一种扩展的高性能存储堆栈用于存储类内存

甄梁[1(&)]，约翰·隆巴迪[2] 穆罕默德·查拉维[3] 以及，迈克尔·亨内克[4]

[1]中国北京三环36号GTC英特尔中国有限公司
liang.zhen@intel.com

[2]英特尔公司SAS, 2 rue de Paris, 92196 Meudon Cedex, France
johann.lombardi@intel.com

[3]英特尔公司，1300 S MoPac Expy，奥斯汀，TX 78746，美国 mohamad.chaarawi@intel.com

[4]联想全球科技德国有限公司，Am Zehnthof 77,45307，德国埃森

摘要 分布式异步对象存储(Distributed Asynchronous Object Storage, DAOS)是一种开源的扩展存储系统，它从头设计以支持用户空间中的存储类内存(Storage Class Memory，SCM)和NVMe存储。它的高级存储API支持结构化、半结构化和非结构化数据模型，克服了传统的基于POSIX的并行文件系统的限制。对于HPC工作负载，DAOS提供了直接的MPI-IO和HDF5支持，以及对遗留应用程序的POSIX访问。本文介绍了DAOS存储引擎的体系结构及其高级应用程序接口。我们还描述了DAOS在IO500基准上的初始性能结果。

关键词:DAOS?SCM?持久性内存系统?并行文件系统?吗?NVMe?分布式存储游泳吗?筏

1介绍

数据密集型应用在商业、政府和学术界的出现，将现有的I/O模型扩展到超出限度。现代I/O工作负载的特点是，元数据与不对齐和碎片化的数据相结合的比例越来越大。传统的存储堆栈通过添加大量延迟和引入对齐约束，为这些工作负载提供了较差的性能。可负担的大容量持久性内存与高速fabric相结合的出现，为重新定义存储范式和高效支持现代I/O工作负载提供了独特的机会。

这场革命需要对整个存储栈进行彻底的重新思考。为了释放这些新技术的全部潜力，新的堆栈必须从头拥抱字节粒度的无共享接口。它还必须能够做到

D. K. Panda(编著):SCFA 2020, LNCS 12082, pp. 40-54, 2020。
https://doi.org/10.1007/978-3-030-48842-0_3

支持故障将成为常态的大规模分布式存储，同时保留 fabric 上的低延迟和高带宽访问。

DAOS 是一个完整的 I/O 架构，它将分布在 fabric 上的 SCM 和 NVMe 存储聚合到全局可访问的对象地址空间，在不影响性能的情况下提供一致性、可用性和弹性保证。

本文的第 2 节描述了 SCM 和 NVMe 存储对传统 I/O 栈构成的挑战。第 3 节介绍了 DAOS 的体系结构，并解释了它如何与新的存储技术集成。第 4 节概述了 DAOS 的数据模型和 I/O 接口，第 5 节介绍了 DAOS 的第一个 IO500 性能结果。

2 使用传统并行文件系统的限制

传统的并行文件系统建立在块设备之上。它们通过 OS 内核块 I/O 接口提交 I/O，该接口针对磁盘驱动器进行了优化。这包括使用 I/O 调度器来优化磁盘寻道、聚合和合并写入以修改工作负载的特征，然后将大量流数据发送到磁盘驱动器以实现高带宽。然而，随着 3D-XPoint 等新存储技术的出现，与传统存储相比，3D-XPoint 可以提供几个数量级的低延迟，为旋转磁盘构建的软件层成为这些新存储技术的纯粹开销。

此外，大多数并行文件系统可以使用支持 RDMA 的网络作为快速传输层，以减少层之间的数据复制。例如，将数据从客户端的页缓存传输到服务器的缓冲缓存，然后将其持久化到块设备。但是，由于传统存储栈中对块 I/O 和网络事件都缺乏统一的轮询或进度机制，I/O 请求处理严重依赖中断和多线程并发 RPC 处理。因此，I/O 处理期间的上下文切换将显著限制低延迟网络的优势。

对于传统并行文件系统的所有厚堆栈层，包括缓存和分布式锁，用户仍然可以使用 3D NAND、3D- xpoint 存储和高速结构来获得一些更好的性能，但也会因为软件堆栈带来的开销而失去这些技术的大部分好处。

3 DAOS，一个为 SCM 和 NVMe 存储构建的存储栈

分布式异步对象存储(Distributed Asynchronous Object Storage, DAOS)是一种为大规模分布式非易失性内存(massively Distributed Non - Volatile Memory, NVM)从头设计的开源软件定义对象存储。它提供了一个 key-value 存储接口，并提供了事务性非阻塞 I/O、版本化数据模型和全局快照等特性。

本节介绍 DAOS 的体系结构，讨论 DAOS 的几个核心组件，并解释为什么 DAOS 可以是具有高性能和弹性的存储系统。

3.1 DAOS 系统架构

DAOS 是一种利用下一代 NVM 技术的存储系统，如存储类内存(SCM)和 NVM express(NVMe)。它绕过了所有的 Linux 内核 I/O，它在用户空间中端到端运行，在 I/O 期间不执行任何系统调用。

如图 1 所示，DAOS 构建在三个构建块之上。第一个是持久性内存和持久性内存开发工具包(persistent memory Development Toolkit，PMDK)[2]。DAOS 使用它存储所有内部元数据、应用程序/中间件关键索引和延迟敏感的小 I/O。在系统启动期间，DAOS 使用系统调用来初始化持久内存的访问。例如，它将启用 dax 文件系统的持久性内存文件映射到虚拟内存地址空间。当系统启动和运行时，DAOS 可以通过 load 和 store 等内存指令直接访问用户空间中的持久性内存，而不是通过一个厚厚的存储堆栈。

持久性内存速度快，但容量低，成本效益低，因此仅靠持久性内存创建大容量存储层实际上是不可能的。DAOS 利用第二个构建块 NVMe ssd 和存储性能开发工具包(SPDK)[7]软件来支持大 I/O 和更高延迟的小 I/O。SPDK 提供了一个 C 库，可以链接到一个存储服务器，该存储服务器可以提供与 NVMe ssd 之间的直接、零拷贝数据传输。DAOS 服务可以完全从用户空间以异步方式通过 SPDK 队列对提交多个 I/O 请求，然后在 SPDK I/O 完成后为存储在持久内存 ssd 中的数据创建索引。

Libfabric[8]和底层高性能 fabric(如全路径架构或 InfiniBand(或标准 TCP 网络))是 DAOS 的第三个构建块。Libfabric 是一个定义 OFI 用户空间 API 的库，并将 fabric 通信服务导出到应用程序或存储服务。DAOS 的传输层是使用 libfabric/OFI 插件构建在 Mercury[9]之上的。它为消息和数据传输提供了一个基于回调的异步 API，为进行网络活动提供了一个无线程的轮询 API。DAOS 服务线程可以主动轮询来自 Mercury/libfabric 的网络事件，作为异步网络操作的通知，而不是使用因上下文切换而对性能产生负面影响的中断。

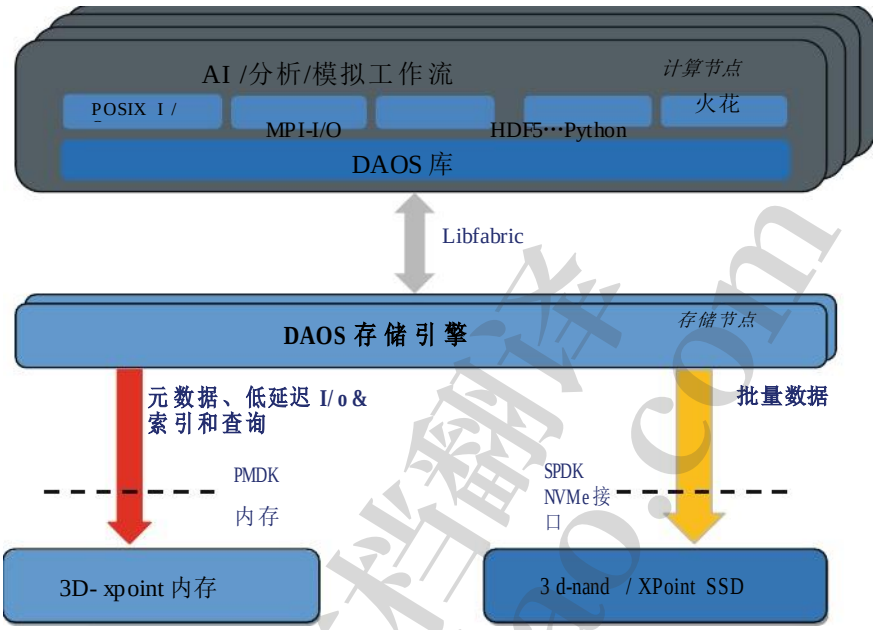

图 1所示。DAOS系统架构

总的来说，DAOS 构建在新的存储和网络技术之上，并完全在用户空间中运行，绕过了所有 Linux 内核代码。因为它是专门为 SCM 和 NVMe 设计的，所以它不支持基于磁盘的存储。传统的存储系统，如 Lustre[11]、Spectrum Scale[12]或 CephFS[10]，可以用于基于磁盘的存储，并且可以在 DAOS 和此类外部文件系统之间移动数据。

3.2 DAOS I/O 服务

从栈分层的角度看，DAOS 是一种采用客户-服务器模型的分布式存储系统。DAOS 客户端是一个与应用程序集成的库，它运行在与应用程序相同的地址空间中。DAOS 库公开的数据模型直接与所有传统数据格式和中间件库集成，这些将在第 4 节中介绍。

DAOS I/O 服务器是一个多租户守护进程，它直接运行在数据存储节点或容器中。它可以直接访问持久性内存和 NVMe ssd，如前一节所介绍的那样。它将元数据和小的 I/O 存储在持久性内存中，而将大的 I/O 存储在 NVMe ssd 中。DAOS 服务器不依赖于生成 pthreads 来并发处理 I/O。相反，它为每个传入的 I/O 请求创建一个 Argobots[6]用户级线程(ULT)。Argobots ULT 是与执行流(xstream)相关联的轻量级执行单元，该流映射到 DAOS 服务的 pthread。这意味着传统的 POSIX I/O 函数调用、pthread 锁或来自任何 ULT 的同步消息等待调用都可以

阻塞执行流上所有结果的进度。然而，由于 DAOS 使用的所有构建块都提供了非阻塞的用户空间接口，因此 DAOS I/O ULT 在系统调用中永远不会被阻塞。相反，如果 I/O 或网络请求仍在飞行中，它可以主动产生执行。I/O ULT 最终将由系统 ULT 重新调度，该系统 ULT 负责从网络和 SPDK 轮询完成事件。ULT 创建和上下文切换是非常轻量级的。基准测试表明，一个 xstream 每秒可以创建数百万个结果，每秒可以进行超过 1000 万次 ULT 上下文切换。因此，它非常适合 DAOS 服务器端 I/O 处理，它应该支持微秒级的 I/O 延迟(图 2)。

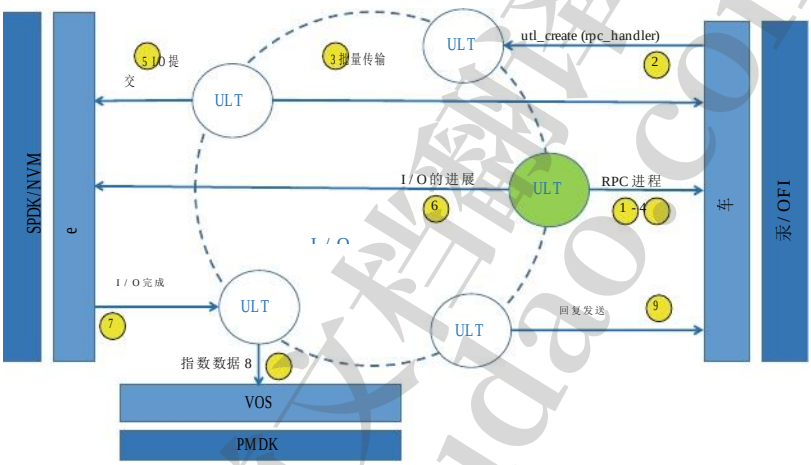

图 2所示。DAOS服务器端 I/O 处理

3.3 数据保护和数据恢复

DAOS 存储公开为允许用户通过键-值或键-数组 API 访问的对象。为了避免扩展问题和维护每个对象元数据(如描述对象数据局部性的对象布局)的开销，DAOS 对象仅由一个 128 位的 ID 标识，该 ID 有几个编码位来描述数据分布和对象的保护策略(复制或纠删码、条带计数等)。DAOS 可以使用这些位作为提示，对象 ID 的剩余位作为伪随机种子，根据 DAOS 存储池的配置生成对象的布局。这就是所谓的算法对象布局。与 Ceph 的数据放置技术类似，只是 DAOS 没有使用 CRUSH[10]作为算法。

本文仅从高层角度描述数据保护与恢复协议。详细的放置算法和恢复协议信息可以在在线 DAOS 设计文档[5]中找到。

数据保护

为了获得超低延迟的 I/O，DAOS 存储服务器将应用程序数据和元数据存储在连接内存总线的 SCM 中，以及通过 PCIe 连接的 ssd 上。DAOS 服务器使用加载/存储指令访问内存映射的持久性内存，SPDK API 从用户空间访问 NVMe ssd。如果持久性内存中存在不可纠正的错误或 SSD 媒体损坏，在没有额外保护的 DAOS 上运行的应用程序将导致数据/元数据丢失。为了保证弹性和防止数据丢失，DAOS 同时提供复制和纠删编码来实现数据保护和恢复。

当启用数据保护时，可以复制 DAOS 对象，或者将其分块成数据和校验片段，然后跨多个存储节点存储。当存储设备或存储节点故障时，DAOS 仍可降级访问，数据冗余可从副本或校验数据[15]恢复。

复制和数据恢复

复制确保了数据的高可用性，因为当任何副本存活时，对象都是可访问的。DAOS 的复制使用主从协议进行写:主副本负责将请求转发到从副本，并处理分布式事务状态。

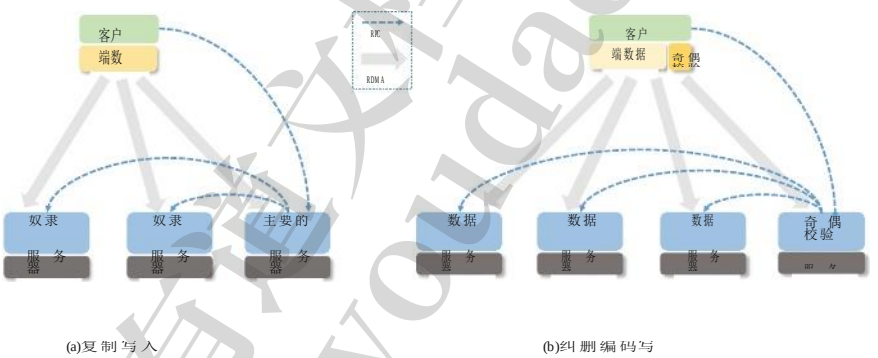

图 3所示。复制和纠删编码的消息和数据流

DAOS 的主从模型与传统复制模型略有不同，如图 3a 所示。主副本只将 RPC 转发到从副本服务器。然后，所有副本将发起一个 RDMA 请求，并直接从客户机缓冲区获得数据。DAOS 选择这种模型是因为在大多数 HPC 环境中，客户端和服务器之间的 fabric 带宽远高于服务器之间的带宽(服务器之间的带宽将用于数据恢复和再平衡)。如果将 DAOS 部署在服务器间带宽较高的非高性能计算用例中，则可以将 DAOS 的数据传输路径更改为传统模型。

DAOS 使用两阶段提交协议的变体来保证复制的更新的原子性:如果一个副本不能应用更改，那么所有副本也应该放弃更改。如果没有失败，这个协议是相当直接的。

然而，如果在两阶段事务期间处理复制写的服务器发生故障，DAOS 将不会遵循传统的两阶段提交协议(等待故障节点恢复)。相反，它将失败节点排除在事务之外，然后通过算法选择不同的节点作为替换，并向前推进事务状态。如果失效节点在某一时刻又回来了，它会忽略其本地的交易状态，依靠数据恢复协议来赶上交易状态。

当 DAOS 的运行状况监视系统检测到存储目标的故障事件时，它将该事件报告给高度复制的基于 RAFT[14]的池服务，该服务可以全局激活池中所有存储服务器上的重建服务。DAOS 服务器的重建服务能够及时扫描存储在本地持久性内存中的对象 id，独立计算每个对象的布局，并通过检查失效目标是否在其布局内找出所有受影响的对象。重建服务还将这些受影响的对象 id 发送到算法选择的回退存储服务器。然后，这些备用服务器通过从幸存的副本中提取数据来重建受影响对象的数据。

在这个过程中，没有中心位置来执行数据/元数据扫描或数据重建:重建服务的 I/O 工作负载将完全分散和并行化。

纠删编码(Erasure Coding)和数据恢复

DAOS 还可以支持用于数据保护的纠删编码(erasure coding，EC)，它比复制具有更高的空间和带宽效率，但需要更多的计算。

由于 DAOS 客户机是一个轻量级库，它与计算节点上的应用程序链接，而计算节点上的计算资源比 DAOS 服务器多得多，因此数据编码由客户机在写时处理。客户端计算奇偶校验，为数据和奇偶校验片段创建 RDMA 描述符，然后向奇偶组的领导服务器发送 RPC 请求来协调写操作。EC 的 RPC 和数据流与复制相同:EC 写的所有参与者都应该直接从客户端缓存中拉数据，而不是从 leader 服务器缓存中拉数据(图 3b)。DAOS EC 还使用相同的两阶段提交协议作为复制，以保证对不同服务器的写操作的原子性。

如果写操作没有与 EC 条带大小对齐，大多数存储系统必须经历一个读/编码/写过程，以保证数据和校验的一致性。这个过程是昂贵且低效的，因为它将产生比实际 I/O 大小多得多的流量。它还需要分布式加锁来保证读写之间的一致性。通过其多版本数据模型，DAOS 可以通过仅将部分写数据复制到校验服务器来避免这个昂贵的过程。在一定的时间之后，如果应用程序继续写入并最终组成一个完整的条带，奇偶服务器可以简单地基于所有这些复制的数据计算奇偶校验。否则，奇偶校验服务器可以协调奇偶组中的其他服务器，从部分覆盖的数据及其旧版本中生成一个合并视图，然后为其计算奇偶校验，并将合并后的数据与新的奇偶校验一起存储。

当发生故障时，ec 保护数据的降级模式读取比复制更重要:通过复制，DAOS 客户端可以简单地切换到从不同的副本读取。但使用 EC 时，客户端必须获取完整的数据条带和

必须在飞行中重建丢失的数据碎片。ec 保护数据的降级模式写处理与复制的处理相同:两阶段提交事务可以继续，而不会被故障服务器阻塞，而是在为事务选择回退服务器后立即继续。

EC 的重建协议也与复制相似，但它产生的数据移动要比复制多得多。这是所有基于奇偶校验的数据保护技术的特点。

端到端数据完整性

DAOS 存储系统中存在三种典型故障:

•服务崩溃。DAOS 通过运行类似八卦的协议 SWIM[13]来捕获这一点。

•NVMe SSD 故障。DAOS 可以通过 SPDK 轮询设备状态来检测此类故障。

•存储介质故障导致的数据损坏。DAOS 可以通过存储和验证校验和来检测这一点。

为了支持端到端校验和和检测静默数据损坏，在将数据写入服务器之前，DAOS 客户端会计算所写入数据的校验和。当接收到写入时，DAOS 服务器既可以验证校验和，也可以直接存储校验和和数据，而无需验证。服务器端验证可以由用户根据性能要求启用或禁用。

当应用程序读回数据时，如果读与原始写入对齐，那么服务器可以只返回数据和校验和。如果读取与原始写入不一致，DAOS 服务器将验证所有涉及的数据区段的校验和，然后计算正在读取的数据部分的校验和，并将数据和校验和返回给客户端。然后客户端在将数据返回给应用程序之前再次验证校验和。如果 DAOS 客户端在读取时检测到校验和错误，它可以为这个特定对象启用降级模式，然后切换到另一个副本进行读取，或者在受 EC 保护的客户端上重构数据飞行。客户端还将校验和错误报告回服务器。DAOS 服务器将收集本地验证和清除检测到的所有校验和错误，以及客户端报告的错误。当错误数量达到阈值时，服务器请求池服务，将坏设备从存储系统中排除，并为其触发数据恢复。

DAOS 的校验和存储在持久性内存中，并受到持久性内存模块的 ECC 的保护。如果持久性内存出现不可纠正的错误，存储服务将被 SIGBUS 杀死。在这种情况下，池服务将禁用整个存储节点，并在幸存的节点上启动数据恢复。

4 DAOS 数据模型和 I/O 接口

本节描述了 DAOS 的数据模型(为该数据模型构建的本机 API)，并解释了如何在该数据模型上实现 POSIX 名称空间。

4.1 DAOS 数据模型

DAOS 数据模型有两种不同的对象类型:允许应用程序表示多维数组的数组对象;键/值存储对象，原生支持常规 KV I/O 接口和多级 KV 接口。KV 和数组对象都有版本化数据，这允许应用程序做出破坏性的更改，并回滚到数据集的旧版本。DAOS 对象始终属于称为 DAOS 容器的域。每个容器都是一个私有对象地址空间，可以由独立于存储在同一个 DAOS 池[1]中的其他容器的事务修改(图 4)。

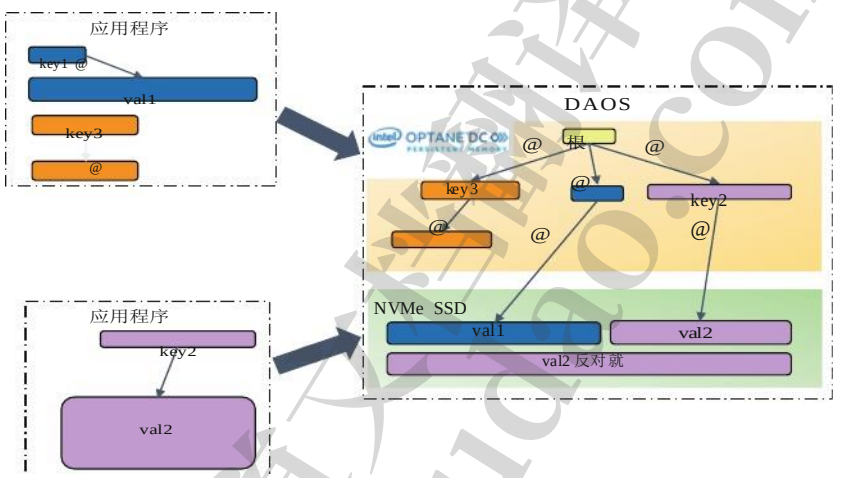

图 4所示。DAOS数据模型

DAOS 容器将通过几个 I/O 中间件库向应用程序公开，从而提供了一条平滑的迁移路径，对应用程序的更改很少(有时甚至没有)。通常，目前所有的 I/O 中间件都运行在 POSIX 之上，并涉及到将中间件数据模型序列化为目录和文件的 POSIX 方案(字节数组)。DAOS 提供了更丰富的 API，为中间件库和应用程序提供了更好、更高效的构建块。通过将 POSIX 视为在 DAOS 上实现的中间件 I/O 库，支持所有构建在 POSIX 之上的库。但与此同时，中间件 I/O 库可以被移植到 DAOS 上本地工作，从而绕过 POSIX 序列化步骤，该步骤有几个缺点，本文将不讨论这些缺点。在 DAOS 库之上实现的 I/O 中间件库包括 POSIX、MPI- I/O 和 HDF5。将来会有更多的 I/O 中间件和框架被移植到直接使用本地 DAOS 存储 API。

4.2 DAOS POSIX 支持

POSIX 不是 DAOS 存储模型的基础。它是作为一个库构建在 DAOS 后端 API 之上，就像任何其他 I/O 中间件一样。POSIX 名称空间可以封装在 DAOS 容器中，应用程序可以将其挂载到文件系统树中。

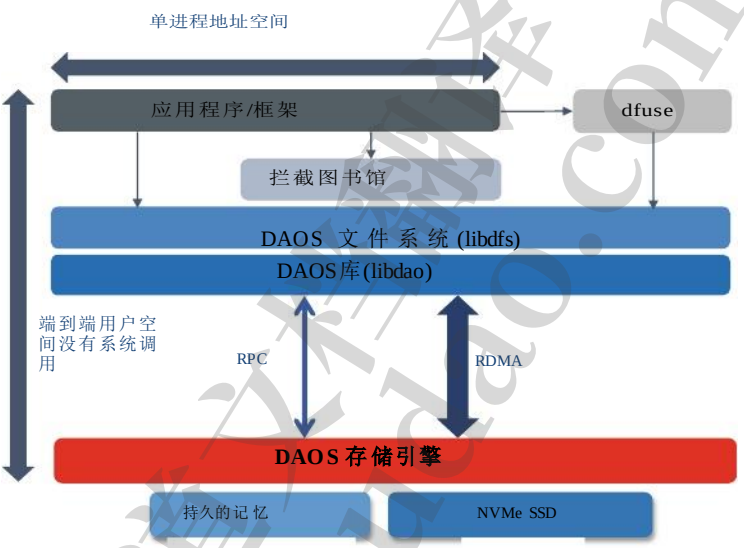

图 5所示。DAOS POSIX 支持

图 5 显示了 POSIX DAOS 的软件堆栈。POSIX API 将通过 fuse 驱动程序使用 DAOS 存储引擎 API(通过 libdaos)和 DAOS 文件系统 API(通过 libdfs)。这将继承 FUSE 的开销，包括系统调用等。对于大多数文件系统操作来说，这种开销是可以接受的，但如果所有 I/O 操作都必须通过系统调用，则像读写这样的 I/O 操作实际上可能会招致显著的性能损失。为了对那些性能敏感的操作启用操作系统旁路，已经在栈中添加了一个拦截库。这个库将与 dfuse 一起工作，并允许拦截 POSIX 读(2)和写(2)调用，以便通过 libdaos 直接从应用程序上下文发出这些 I/O 操作(不需要任何应用程序更改)。

在图 5 中，在 dfuse/截取库和 libdaos 之间有一层，叫做 libdfs。libdfs 层直接在 DAOS API 之上提供了一个类似于 POSIX 的 API。它提供了本地 libdaos 库之上的文件和目录抽象。在 libdfs 中，POSIX 命名空间被封装在容器中。文件和目录都映射到容器内的对象。命名空间容器可以挂载到 Linux 文件系统树中。封装的 POSIX 文件系统的数据和元数据都将完全分布在所有可用的存储上

DAOS 池的。dfuse 守护进程与 libdfs 链接，来自 FUSE 的所有调用都将经过 libdfs，然后是 libdao，它们可以访问由 DAOS 服务器公开的远程对象存储。

此外，如上所述，libdfs 可以通过多个接口向终端用户公开，包括 SPARK、MPI-IO 和 HDF5 等框架。当有一个 shim 层用于 libdfs 作为 I/O 中间件插件时，用户可以直接将应用程序与 libdfs 链接起来。这种方法是透明的，不需要对应用程序进行更改。

5 的性能

DAOS 软件堆栈仍在大量开发中。但它在新的存储类存储技术上所能达到的性能已经在 ISC19 和 SC19 会议上得到了展示，在 DAOS 0.6 版本上的 IO500 基准套件的第一个结果最近已经提交给了[16]。IO500 是由虚拟 I/O 研究所(VI4IO)[17]组织的跟踪超级计算机存储性能和存储技术的社区活动。IO500 测试套件包括数据和元数据工作负载以及并行命名空间扫描工具，并计算单个排名评分用于比较。工作负载包括:

IOR-Easy:格式良好的大型顺序 I/O 模式的带宽

•IOR-Hard:大步 I/O 工作负载的带宽与小的非对齐 I/O 传输(47001 字节)

•MDTest-Easy:对 0 字节文件的元数据操作，对每个 MPI 任务使用单独的目录

•MDTest-Hard:对共享目录下 3901 字节的小文件进行元数据操作

Find:通过目录遍历查找相关文件

我们已经调整了用于 IOR 和 MDTEST 的 I/O 驱动程序，使其直接在第 4 节中描述的 DFS API 上工作。驱动程序被推送并接受到上游 ior-hpc 存储库以供参考。开发新的 IO 驱动程序相对容易，因为如前所述，DFS API 与 POSIX API 非常相似。下面总结了为 IOR 和 mdtest 实现 DFS 后端的步骤。同样的方案也可以应用于使用 POSIX API 的其他应用程序:

•添加一个初始化回调来连接到 DAOS 池，并打开将封装命名空间的 DAOS 容器。然后在该容器上创建 DFS 挂载。

•为所有需要的操作添加回调函数，并用相应的 DFS API 替换 POSIX API。IOR 和 mdtest 中使用的所有 POSIX 操作都有相应的 DFS API，这使得映射变得很容易。例如:

-将 mkdir()改为 dfs_mkdir();

-将 open64()改为 dfs_open();

-将 write()修改为 dfs_write();

——
等 等 。

-添加一个 finalize 回调来卸载 DFS 挂载并关闭池和

容器处理。

发布了两个 IO500 结果列表:“完整列表”或“排名列表”包含在任意数量的客户节点上实现的性能结果。“10 个节点挑战”列表包含了 10 个客户端节点的结果，这为比较与客户端节点数量[3]相匹配的 IO500 负载提供了标准化的基础。对于这两个列表，对于存储系统的大小都没有限制。可选的数据字段可以提供关于数据和元数据存储设备的数量和类型的信息;当在提交中出现时，这些信息可以用来判断存储系统的相对效率。

为了提交给 SC19[16]的 IO500，IO500 基准测试已经在英特尔的 DAOS 原型集群“Wolf”上运行。“Wolf”集群的 8 个双插槽存储节点使用英特尔 Xeon 铂金 8260 处理器。每个存储节点配置 12 个英特尔 Optane 数据中心持久性内存模块(dcpmm)，容量 512 GiB(每个节点总共 3 TiB, app-direct/interleaved 模式配置)。“Wolf”集群的双 socket 客户端节点使用英特尔 Xeon E5-2699 v4 处理器。DAOS 存储节点和客户端节点都为每个节点配备了两个英特尔 omnipath 100 适配器。

图 6 显示了 IO500“10 节点挑战”2019 年 11 月版前 4 位存储系统的 IO500 IOR 带宽。DAOS 既获得了第一名的总体排名，也获得了最高的”bw”带宽评分(四个 IOR 工作负载的几何平均)。由于其多版本数据模型，DAOS 不需要对小的或未对齐的写操作进行读-修改-写操作(在传统的 POSIX 文件系统中，这将产生额外的 I/O 流量和锁定争用)。DAOS 存储引擎的这个属性为“硬”和“简单”IOR 工作负载带来了非常相似的 DAOS 带宽，并在许多不同的工作负载之间提供了可预测的性能。

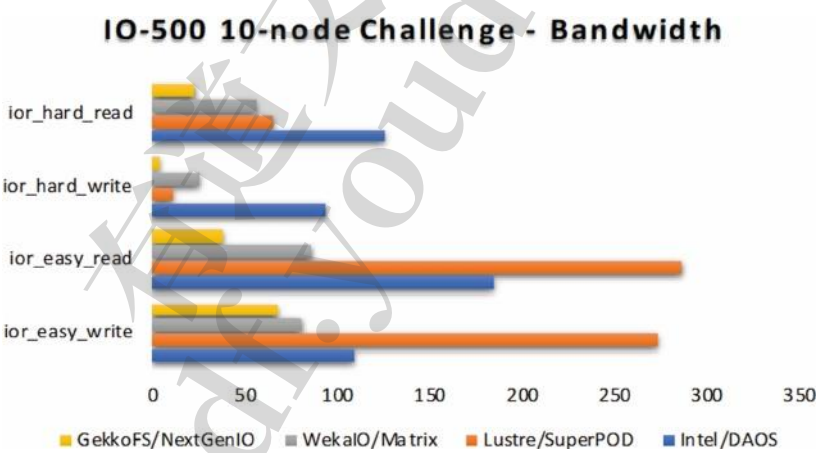

图 6所示。IO500 10节点挑战-IOR 带宽，单位为 GB/s

图 7 展示了 IO500“10 节点挑战”2019 年 11 月版本中，排名前 4 的存储系统的 mdtest 元数据性能。DAOS 主导了整体的“md”元数据评分(所有 mdtest 工作负载的几何平均值)，与最近的竞争者相比几乎有 3 倍的差异。这主要是由于轻量级的端到端用户空间存储栈，结合超低延迟网络和 DCPMM 存储媒体。与 IOR 带宽结果一样，DAOS 元数据性能在所有测试中都非常均匀，而许多其他文件系统在不同的元数据工作负载之间表现出很大的差异。

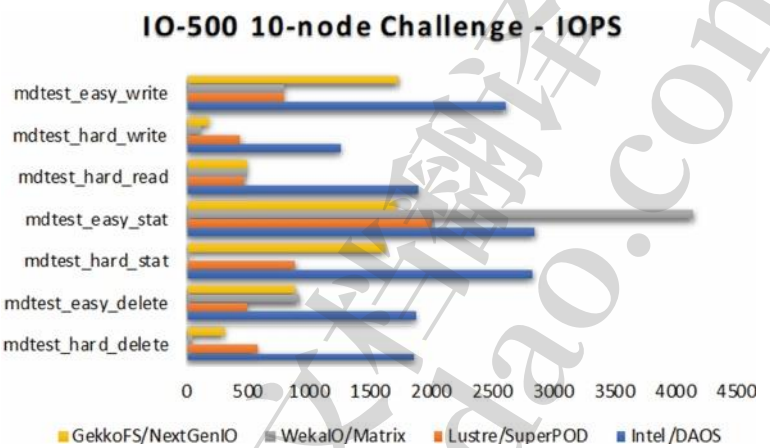

图 7所示。IO500 10节点挑战- mdtest在 kIOP/s 中的性能

DAOS 在 2019 年 11 月的”完整列表”上取得了第二名，仅使用了 26 个客户节点。通过更大的客户端节点集，尤其是那些随着客户端节点数量扩展的元数据测试，可以期待更好的性能。因此，直接与“完整列表”上的其他存储系统进行比较(其中一些系统使用数百个客户机节点进行了测试)没有“10 节点挑战”那么有意义。

IO500 结果的完整列表和 IO500 基准套件的详细描述可以在 Ref.[16]上找到。

6 结 论

随着存储类内存和 NVMe 存储的广泛使用，软件栈开销成为整个存储系统的一部分。传统的存储系统要充分利用这些存储硬件设备的优势是非常困难的。提出了 DAOS，作为针对这些新概念设计的软件栈

存储技术，描述了 DAOS 的技术特点，并解释了它如何实现高性能和高弹性。

在性能方面，IO500 的测试结果证明了 DAOS 能够充分利用新的存储设备及其用户空间接口。比 IO500 列表上的绝对排名更重要的是，DAOS 性能在 IO500 工作流程中非常均匀，而其他文件系统有时在个别 IO500 测试之间表现出数量级的性能差异。

本文仅简要介绍了 DAOS 的几个核心技术组件及其现有的 POSIX I/O 中间件。其他支持的 I/O 库如 MPI-I/O 和 HDF5 不在本文讨论范围内，将成为未来研究的主题。其他基于 DAOS/libdfs 的 I/O 中间件插件仍在开发中。DAOS 的路线图、设计文档和开发状态可以在 github[5]和英特尔 DAOS 网站[4]上找到。

参考文献

1.刘伟等:科学应用中极端规模系统的 DAOS(2017)。https://arxiv.org/pdf/1712.00423.pdf

2.Rudoff,。:持久性内存编程 api(2018)。https://storageconference。我们/ 2018 /演示/ Rudoff.pdf

3.N. Monnier, J. Lofstead, M. Lawson, M. Curry。:使用 IO500 和 mistral 分析平台存储。召开:第四届国际并行数据系统研讨会，2019(2019)。https://conferences.computer.org/sc19w/2019/pdfs/PDSW2019-6YFSp9XMTx6Zb1FALMAAsH / 5 pvxonjobjwd2nqgl1mub3/6lk0ohjlepg2budbxxppoq.pdf

4.DAOS。https://wiki.hpdd.intel.com/display/DC/DAOS+Community+Home

5.DAOS github。https://github.com/daos-stack/daos

6.Seo, S.等人:Argobots:一个轻量级的底层线程和任务框架。IEEE 反式。Distrib 平行。系统 29(3)(2018)。https://doi.org/10.1109/tpds.2017.2766062

7.SPDK。https://spdk.io/

8.Libfabric。https://ofiwg.github.io/libfabric/

9.汞。https://mercury-hpc.github.io/documentation/

10.威尔，s.a.，勃兰特，s.a.，米勒，e.l.，马尔扎恩，C。：CRUSH:受控的、可扩展的、去中心化的复制数据放置。见:2006 年 ACM/IEEE 超级计算会议论文集，SC 2006(2006)。https://doi.org/10.1109/sc.2006.19

11.Braam, P.J.: The Lustre storage architecture(2005)。https://arxiv.org/ftp/arxiv/papers/1903/1903.01955.pdf

12.傻瓜，F.，哈斯金，R.。：GPFS:用于大型计算集群的共享磁盘文件系统。见:第一次 USENIX 文件和存储技术会议论文集，蒙特雷，CA，2002 年 1 月 28-30 日，pp 231-244(2002)。http://www.usenix.org/publications/library/程序/ fast02 /

13.Das, A., Gupta, I., Motivala, A.。：SWIM:可扩展的弱一致性感染式进程组成员协议。载:2002 年国际可靠系统和网络会议论文集，DSN 2002, pp. 303-312 (2002)

14.D. Ongaro, J. Ousterhout。：In search of an understanding consensus algorithm(2014)。https://www.usenix.org/system/files/conference/atc14/atc14-paper-ongaro.pdf

15.巴顿,E。DAOS:一种用于极端存储规模存储的架构(2015)。https://www。sniа.org/sites/default/files/SDC15_presentations/dist_sys/EricBarton_DAOS_Architecture_Extreme_Scale.pdf

16.IO500 榜单，2019 年 11 月 https://www.vi4io.org/io500/list/19-11/start

17.Kunkel, J.等:I/O 虚拟研究所。https://www.vi4io.org/start