Strata：虚拟化非易失性存储器上的高性能可扩展存储

**学号：2017282110259 姓名：周静玲**

**一、简介**

Strata是一个围绕高性能PCIe闪存存储设计的商业存储系统，但是目前却发现这个新型的闪存硬件面临一个和10年前相同的问题：没有充分利用服务器硬件。为了解决这一问题，借鉴硬件虚拟化的想法，提出了一种新颖的存储系统的设计：将功能划分为高性能网络连接闪存的地址虚拟化层和用于实现可扩展协议实现的托管环境。

基于闪存的存储设备的特点是速度快，价格昂贵和对设备要求严苛：即使是对于任意的IO，单个的设备能够饱和10Gb / s的网络连接，但这个过程中会消耗大量的CPU资源。同样的设备可能会花费与其安装的服务器相同的甚至会超过的成本。快速，非易失性介质的成本和性能特征改变了存储系统设计的演算，并为构建高效和高性能的数据中心存储提出了新的挑战。

在围绕PCIe闪存设计系统时，从高性能驱动器对大规模存储系统的影响当中有了两个发现：首先，这些设备要足够快，即使小处理开销也会阻止它们被完全利用，但仍然需要很多并发工作负载来完全饱和它们。因此，必须改变媒体从聚合到虚拟化的方式；其次，聚合仍然是实现冗余和规模等属性所必需的。但是，它必须避免传统存储阵列的单片控制器方法所带来的性能瓶颈：传统存储阵列是围绕着过时的假设而设计的，即媒体是系统中速度最慢的组件。为了能够在现有的数据中心环境中实用，必须与现有的客户端存储接口保持兼容，并支持快照和重复数据删除等标准企业功能。

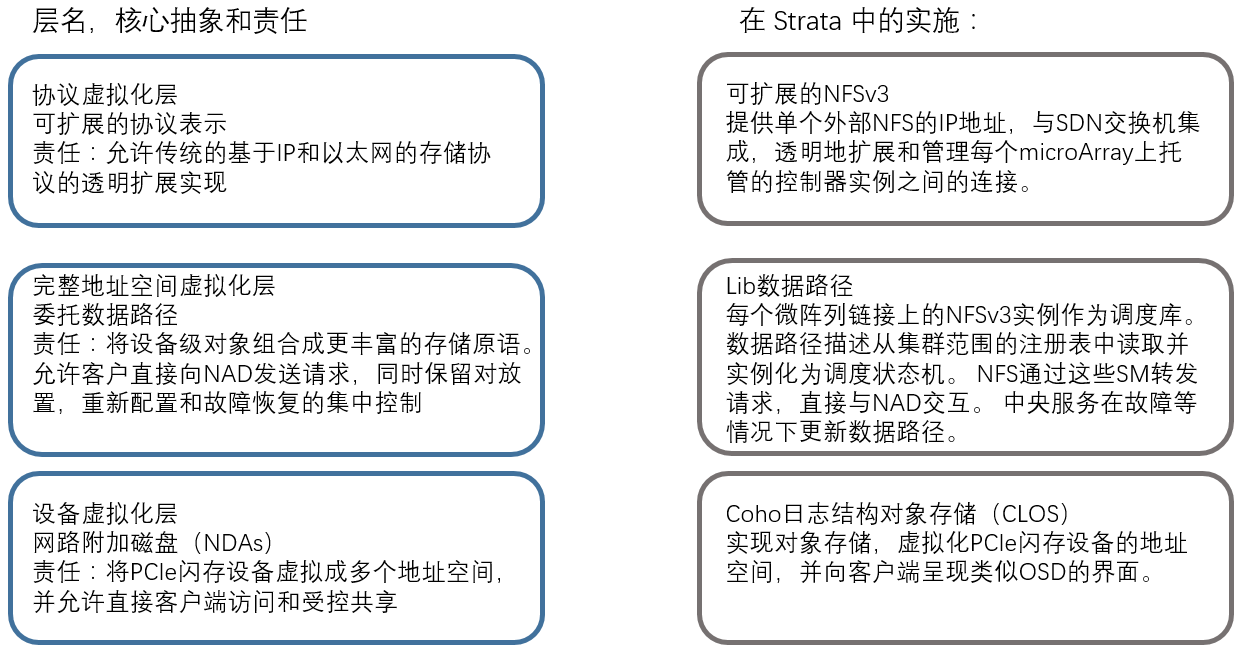
后文将探讨这两个发现结果对于存储虚拟机映像的可扩展的高性能NFSv3实现的影响。其中，构成设计基础的两个广泛的技术贡献：① 通过全局数据地址虚拟化，从客户端数据到物理资源的委托映射和请求调度接口允许客户端直接寻址数据，同时仍然提供在线数据移动所需的协调（例如，响应故障或负载平衡）；② SDN辅助存储协议虚拟化，允许客户端寻址跨多个真实服务器透明地扩展的单个虚拟协议网关（例如，NFS服务器）。 使用这种技术构建了一个可扩展的NFS服务器，但它也适用于其他协议（如iSCSI，SMB和FCoE）。

Strata的核心是使用设备级对象存储和动态的全局地址空间虚拟化来实现存储系统中控制和数据路径之间的清晰高效的分离。闪存设备使用对象存储风格的接口分成虚拟地址空间，然后允许客户端以安全，低开销的方式直接与这些地址空间进行通信。为了构建更丰富的存储抽象，全局地址空间虚拟化层允许客户端将多个每个设备的地址空间与实现条带化和复制等属性的映射进行聚合。 这些委托地址空间映射的协调方式保留了与存储设备的直接客户端通信，同时仍然允许对数据放置，迁移，扩展和故障响应进行动态和集中的控制。

通过像NFS这样的传统协议提供这种存储会带来另一个可扩展性问题：这些协议的客户端通常期望一个服务器的IP地址，这个IP地址必须在多个服务器上动态平衡以避免成为性能瓶颈。 为了扩展请求处理和充分利用客户端和存储资源之间的全部交换带宽，开发了一个可扩展的协议表示层，作为架构下层的客户端，并与软件定义的网络 切换到横跨任意多个物理服务器的存储控制器的协议组件的实施。 通过将协议网关构建为地址虚拟化层的客户端，保留了将扩展访问委托给设备存储的功能，而无需在使用存储的终端主机上更改接口。

**二、架构**

新兴存储硬件的性能特点要求完全重新考虑存储架构，以便构建可扩展，低延迟的共享持久性存储器。 部署应用程序的现实是接口必须保持完全相同才能使存储系统具有相关性。Strata的架构旨在迈向这些目标中的第一个目标，同时保持务实的侧重点。



**图1-Strata网络存储架构**

图1描绘了Strata架构的三个层次。系统中每一层的目标和抽象都在左边一栏，实施这些目标的具体体现在右边。在基础上，通过对象存储接口访问设备，负责虚拟化设备的地址空间，并允许客户端与单个虚拟设备进行交互。这种方法反映了观点：现在这些存储设备的系统设计与十年前的CPU虚拟化类似：设备提供比大多数单个工作负载所需要的更高的性能，因此需要用于受控共享的轻量级接口，租赁。实现了一个每设备对象存储，允许一个设备被虚拟化为2128个稀疏对象的地址空间，每个稀疏对象的大小可以达到264个字节。实现与OSD规范的意图类似，本身是由网络连接的安全磁盘驱动的。尽管目前还没有广泛部署，但是通过pNFS使用OSD作为后端，NVMe命名空间抽象以及希捷Kinetic驱动器等新兴硬件，设备级对象存储正在得到重新关注。对象存储接口作为一个整体并不是一个重要的技术贡献，把这个层称为网络附加磁盘（NAD）。

体系结构的中间层提供了一个全局地址空间，它支持IO处理器的高效组合，将虚拟对象上的客户端请求转换为一组NAD级物理对象上的操作。将特定虚拟对象的IO处理器图形称为其数据路径，并在全局虚拟地址映射中维护每个对象的数据路径描述。客户端使用调度库来实例化每个数据路径所描述的处理图，并对图的叶子处的物理对象执行直接IO。通过一致性协议访问虚拟地址映射，该协议允许中央服务在虚拟对象被客户端主动使用时更新虚拟对象的数据路径。更具体地说，数据路径允许将物理对象组合成更丰富的存储原语，从而提供条带化和复制等属性。这一层的目标是在可伸缩性和效率之间取得平衡：它支持直接客户端访问设备级对象，而不会牺牲集中管理数据放置，故障恢复以及更高级的存储功能，如重复数据删除和快照。

最后，顶层执行协议虚拟化，允许客户端通过标准协议（如NFS）访问存储，而不会损失客户端对NAD的直接请求的可扩展性。表示层与10Gb软件定义的以太网交换结构紧密集成，允许外部客户端连接到单个TCP端点的错觉，同时透明地动态平衡所有NAD上协议实例的流量到单个IP地址。每个协议实例都是下图层的瘦客户端，可以与其他协议实例进行通信，以执行协议所需的任何附加同步，例如，维护NFS名称空间一致性。

请求通过启用OpenFlow的交换机从客户端传送到Strata，交换机根据负载将其分派到运行在MicroArray（μArray）上的相应协议处理程序---一个配置了闪存设备的小型主机和足够的网络和CPU来饱和它们，包含代表单个NAD的软件堆栈。为了提高性能，每个层都以库的形式实现，允许单个进程处理从客户端到媒体的请求流。NFSv3实现充当底层调度层的客户端，其将虚拟对象的请求转换成对物理对象的一个或多个请求，通过对本地物理对象的功能调用发出，并通过RPC发送到远程对象。本文其余部分的重点是扩展NFS的具体实现，值得注意的是，该设计旨在允许应用程序直接与NFS实现使用的相同数据路径库进行链接，从而生成在多租户多显示存储系统中，具有最小的网络和设备级开销。

在这篇文章中，首先只讨论NFS作为协议虚拟化的具体实现，其次Strata最初设计为一个软件层，它位于托管虚拟机的相同物理服务器上；最后，完整实现在每个存储节点上都包含了一层旋转磁盘，以使冷数据可以更经济地存储在闪存层后面。在后面的内容中，将会更详细地讨论Strata地址空间虚拟化，动态重新配置和可伸缩协议支持的三个相关方面。在提供整个系统的性能评估之前，先介绍一下这三个组件在NFSv3实现中如何与VM镜像存储交互的一些细节。

**三、数据路线**

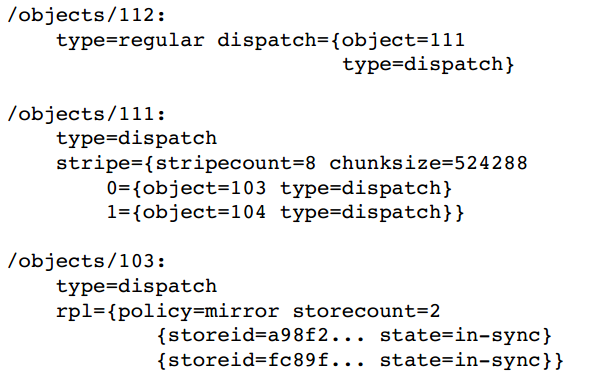
Strata为数据提供了一个通用的库接口，这是所描述的更高级别的客户端特定协议的基础。这个库提供了一个虚拟对象的概念，这个虚拟对象可以在整个集群范围内使用，并且可以包含多个捆绑在一起的物理对象，用于并行数据访问，容错或其他原因。 该库提供了由NAD提供的对象存储接口的超集，以及用于跨NAD管理对象布置的额外接口，在对象范围被复制或分割时维护数据不变量，并协调对数据的并发访问和并发操作描述其布局的虚拟地址映射。

为了避免IO瓶颈，数据路径接口的用户直接访问数据，其中可能是本地客户端或协议网关，例如NFS服务器。为此，他们使用虚拟地址映射将来自虚拟对象的请求映射到物理对象。这不仅仅是从虚拟对象（id，range）对到一组物理对象（id，range）对的指针。相反，每个虚拟范围都与该范围的特定处理器以及特定于处理器的上下文相关联。 Strata使用面向调度的编程模型，在这个模型中，当请求从一个始发客户端传递，经过一系列转换，并最终到达适当的存储设备时，对请求执行一系列操作。模型借鉴了以往的一些数据包处理系统的思想（X-Kernel Scout Click），将它们调整到一个存储上下文中，沿着流水线的模块通过一组分层地址空格，并且可以在通过时分叉和/或收集请求和响应。

调度库提供了一个请求处理器的集合，可以独立或与其他处理器组合。每个处理器将存储请求（例如，读取或写入请求）作为输入，并向其子代产生一个或多个请求。 NAD暴露孤立的稀疏物体; 处理器执行翻译，允许多个对象为了某种功能目的而被组合，并将它们呈现为单个对象，其可以被其他处理器使用。 基于请求的地址转换构建存储功能的想法已经在其他系统中使用，通常作为批量管理的基础; Strata将其从底层存储系统中解放出来，并将其视为一流的调度抽象。

调度模块的组成与Click相似，但存储域中的应用程序带有许多不同之处。首先，请求在到达存储设备时通常得到确认，因此它们与数据包转发逻辑的不同之处在于它们通过调度堆栈向下行进，然后返回;处理器包含处理请求和响应的逻辑。其次，请求在遍历处理器时被拆分或合并很常见。例如，复制处理器可能会复制请求并将其发送到多个节点，然后在将单个响应传递回其父节点之前收集所有响应。最后，当处理器描述快速的基于库的请求调度逻辑时，它们通常依赖于来自系统的附加设施。 Strata允许处理器实现访问API以获得共享的，集群范围的状态，这可以在控制路径上使用，例如存储副本配置。它还提供了用于后台功能的功能，如NAD故障检测和响应。处理器组织的目的是允许将调度决策推送到客户端实现，并且在对性能影响最小的情况下进行，同时仍然从用于维护系统和响应故障的通用系统基础结构中受益。

**3.1 虚拟地址的映射**



**图2-虚拟对象到物理对象范围的映射**

图2显示了典型对象的虚拟地址映射中存储的相关信息。每个对象都有一个标识符，一个类型，一些特定于类型的上下文，还可能包含其他元数据，如缓存大小或修改时间信息。

虚拟地址映射的入口点是一个常规对象。这不包含自己的位置信息，而是委托给顶级调度对象。在图2中，对象112是一个常规对象，委托给一个调度处理器，其上下文由对象111标识（这里的ID是相反的顺序，因为调度图从下到上创建，但是从上到下遍历）。因此，当客户端打开文件112时，它使用对象111中的数据作为上下文来实例化调度器。此上下文通知调度程序，它将通过条带处理器委派IO，对象使用2个条带，条带宽度为512K。调度器又实例化8个处理器（每个条带一个），每个处理器配置有存储在与每个条带相关联的对象中的信息（例如，条带0使用对象103）。最后，当条带调度器在条带0上执行IO时，它将使用对象103的对象描述符中的上下文来实例化复制的处理器，该处理器镜像写入其复制集中列出的NAD，并且向最近的同步复制品（距离当前只是本地或远程）。

除了这里描述的条带化和镜像处理器之外，映射还可以支持其他更高级的处理器，例如擦除编码，或者对任意对象的字节范围映射（其中还支持重复数据删除）。

**3.2 调度**

IO请求由一系列调度程序处理，每个调度程序都有一些共同的功能。 如果调度程序跨越不同子处理器所覆盖的范围，或者将请求克隆到多个子请求（例如，用于复制），调度员可能必须将请求分段，并且必须收集子请求的结果并处理部分故障。

包含在标准库中的复制和分条模块代表处理器在通过调度堆栈时转换请求的方式。复制处理器允许将请求分割并同时发布到一组副本对象。每个对象中的请求地址保持不变，直到所有副本确认请求完成为止，才会返回响应。处理器优先从本地副本读取数据，但在发生故障（错误响应或超时）时将请求转发到远程复制副本。它强制全局排序写入请求并将它们并行传输到所有副本。它还定期向每个副本的日志提交一个轻量级检查点，以保持同步点的持续记录;这些检查点用于崩溃恢复。

分条处理器通过稀疏对象集合分配数据。它被参数化为一个条带大小（以字节为单位）和一个对象列表来充当有序的条带集。如果请求跨越一个条带边界，处理器将该请求分成一组每个条带请求，并异步发布这些请求，在返回之前收集响应。静态的，基于地址的条带是一个相对简单的负载平衡和数据分布机制相比，如一致的哈希放置方案。经验告诉我们的是，这种方法是有效的，因为数据放置在对象地址空间内往往是合理的一致的;因为使用相当大的条带大小（默认为512KB）足以在正常操作下保持足够低的请求碎片开销。

**3.3 相关性**

Strata客户端也参与一个简单的协调协议，以便即使在使用该对象时也可以更新虚拟对象的虚拟地址映射。在线重新配置提供了从故障中恢复，响应容量变化，甚至移动对象以响应观察或预测的负载。

虚拟地址映射存储在通过Apache Zookeeper实现的分布式同步配置数据库中，该数据库也可用于软件堆栈中其他位置服务所需的任何低带宽同步。一致性协议建立在配置数据库之上。它目前针对每个对象的单个写入器进行了优化，其工作原理如下：当客户端希望写入虚拟对象时，它首先在配置数据库中声明锁定。如果对象已经被锁定，客户端请求持有者释放它，以便客户端可以声明它。如果持有人没有在合理的时间内自愿放行，持有人被认为是没有反应的，并使用下面所说的机制从系统中进行隔离。这足以让对象移动，首先在所需的位置创建新的不同步的物理对象，然后请求释放对象的锁定器。对象的用户将重新获得下一次写入的锁定，并在此过程中发现新的不同步副本并启动重新同步。当新副本同步时，可能会重复相同的过程以删除位于不合需要位置的副本。

**4.网络附加磁盘**

在设备级别，将底层存储多路复用到对象中，由128位标识符命名，并由稀疏的264字节数据地址空间组成。 这些地址空间目前由一个垃圾收集日志结构化对象存储支持，但是对象存储的实现对上面的层是不透明的，如果较新的存储技术使得不同的访问模式更有效，则可以替换它们。 还通过允许每个对象将低优先级或不常用的数据刷新到磁盘来提供增加的容量，但是这又被隐藏在对象接口后面。 磁盘分层，垃圾收集和文件系统布局的细节超出了本文的范围。

物理对象接口在很大程度上是传统的基于对象的存储设备，具有用于稀疏对象的CRUD接口，以及一些扩展来协助群集协议。 它比现有的块设备接口（比如SCSI命令集）要简单得多，但也比甚至更窄的接口（比如键值存储的接口）更直接和更通用。 提供低级硬件抽象层允许定制实现，以适应单个闪存实现的最佳实践，并且随着新技术的推出，还可以在媒体接口级别进行更为戏剧性的设计更改。

**4.1 网络集成**

通过与管理型以太网交换机紧密耦合来解决以往分布式系统中存在的行为不当的节点，方法借鉴了Sane和Ethane等的系统思想，使用托管网络来强制独立端点之间的隔离。系统与基于OpenFlow的交换机以及VMM上的软件交换相集成，以确保Strata对象只能由其授权客户端访问。从最初的在企业环境中常见的硬件支持的隔离形式的以太网VLAN转向Open Flow，是因为他为流量隔离提供了更灵活的隧道抽象。此外， 还公开了一个隔离的专用虚拟网络，用于集群内部的带外控制和管理操作，允许NAD本身在集群监视器的控制下访问远程对象以进行对等重新同步和重新组织。

**五、在线重配置**

为了保持其性能和可靠性，Strata必须响应两大类事件。第一类包括直接在数据路径上发生的故障。调度库通过代表客户端重新配置受影响的虚拟对象，立即自动从这些故障中恢复。 第二类包括设备故障和负载不平衡等事件。这些由专门的集群监视器处理，执行大规模的重新配置任务以保持整个系统的健康。在所有情况下，重新配置都是在线执行的，对客户端可用性的影响最小。

**5.1 对象重配置**

许多错误恢复机制直接构建到调度库中。 这些机制允许客户端通过重新配置数据路径上的单个虚拟对象来快速从故障中恢复。

**5.1.1 IO 错误**

复制IO处理器以明显的方式响应读取错误：立即将失败的请求重新提交给不同的副本。 另外，客户维护每个设备的错误数量; 如果设备的总计错误数超过可配置的阈值，则后台任务将使设备脱机并协调系统范围的重新配置。

IO处理器通过在发生故障时同步重新配置虚拟对象来响应写入错误。这涉及三个步骤。首先，受影响的副本在配置数据库中被标记为不同步。这充当了一个全局的，持久的指示，即副本可能不被用来提供读取，因为它包含潜在的陈旧数据。其次，尽力尝试通知NAD该错误，以便它可以启动后台任务来重新同步受影响的副本。这使系统几乎立即从瞬时故障中恢复。最后，IO处理器在单独的设备上分配一个特殊的补丁对象，并将其添加到副本集。一旦副本已经被标记为不同步，在重新同步之前不会再发出写入;通过在这些恶化的条件下提供临时缓冲区来吸收写入，修补程序可防止设备故障妨碍进度。通过分配修补程序对象，IO处理器可以继续满足新写入的复制要求，而不同步副本在后台进行修复。只要同步副本或不同步副本及其所有修补程序都可用，副本集仍然可用。

**5.1.2 再同步**

除了为客户提供通过虚拟地址映射直接访问设备之外，Strata还提供了许多后台服务来维护单个虚拟对象和整个系统的运行状况。其中最重要的是resync服务，它提供了一个后台任务，可以重新同步跨多个设备复制的对象。

重新同步建立在特殊的NAD resync API的基础之上，暴露了对象存储的底层日志结构。 NAD与其商店中的每个物理对象保持一个日志序列号（LSN）;当一个记录被附加到一个对象的日志时，它的LSN是单调递增的。IO处理器使用这些LSN对对跨存储复制的物理对象所做的更改进行全局排序，并验证所有副本是否已收到全部更新。

如果写入失败导致副本不同步，则客户端可以请求系统重新同步副本。它通过调用承载不同步副本的NAD上的重新同步RPC来完成此操作。然后，服务器启动一个后台任务，它将来自同步副本的丢失日志记录进行流式处理，并将其应用到本地不同步副本，并使用LSN标识本地副本丢失的记录。

在重新同步期间，后台任务具有对不同步副本的独占写访问权限，因为所有客户端都已重新配置为使用修补程序。因此，重新同步任务可以在客户端继续写入时追逐同步对象日志的尾部。当大量数据被复制时，重新同步任务进入最终停止和复制阶段，在该阶段中，它获得对副本集中的所有副本的独占写入访问，完成重新同步，应用在该中间期间接收到的任何客户端写入，将副本标记为在配置数据库中处于同步状态，并删除该修补程序。

确保重新同步能够及时进行以限制数据丢失的漏洞，这一点非常重要。非常繁重的客户端写入负载可能会干扰重新同步任务，并在最糟糕的情况下导致无限的传输时间。由于这个原因，当一个对象被重新同步时，客户端写入被限制，重新同步请求被优先。

**5.1.3 紧急恢复**

在不洁净的情况下，必须特别小心。在干净关闭时，通过从配置数据库中删除它们的锁来释放所有对象。当发现具有陈旧锁的副本集时（即识别无响应IO处理器的锁），将检测到崩溃。发生这种情况时，假定配置数据库中标记为同步的副本确实处于同步状态是不安全的，因为崩溃可能是在配置数据库更新过程中发生的;相反，必须直接查询该集合中的所有副本以确定其状态。

在一般情况下，IO处理器检索集合中每个副本的LSN，并确定哪些副本（如果有的话）不同步。如果所有副本具有相同的LSN，则不需要重新同步。如果发现不同的LSN，则将具有最高LSN的副本指定为授权副本，并且将所有其他副本标记为不同步，并启动重新同步任务。

如果在恢复过程中无法查询副本，则会在配置数据库中将其标记为已分发，并且将从可用副本中选择具有最高LSN的副本作为授权副本。在这种情况下，写入可能已经被提交给分发的副本，而不是任何其他副本。如果分支副本将来再次变得可用，则必须丢弃这些额外的写入。这是通过将副本滚动回上一个检查点，并从日志中的该点开始重新同步来实现的。面对这种回滚的一致性可以通过确保在向客户端写入之前在配置数据库中成功标记为不同步来保证。因此写入失败可以保证在配置数据库中标记副本不同步（并创建相应的补丁）或者传播回客户端。

**5.2 系统重配置**

Strata还提供高度可用的监测服务，监视系统的健康情况，并根据需要协调整个系统的恢复过程。 监视器收集来自客户端，SMART诊断工具和NAD RPC的信息以评估系统的状态。监视器基于上述每个对象重新配置机制构建，以响应各个客户端无法解决的事件，例如整个系统的负载不平衡，存储容量接近以及设备故障。

**5.2.1 再平衡**

Strata提供了一个重新平衡工具，可以执行系统级的重新配置来修复损坏的副本，防止NAD填满容量，并改善跨NAD的负载分配。这个设施反过来被用来从设备故障中恢复并扩展到新的硬件上。

重新平衡分两个阶段进行。在第一阶段，监视器检索当前的系统配置，包括每个虚拟对象的所有NAD和虚拟地址图的状态。然后根据可定制的放置策略构建副本的新布局。这个过程是可编写的，可以很容易地定制，以满足个别部署的特定性能和持久性要求。默认策略使用贪婪算法，该算法考虑了许多旨在确保复制的物理对象不共享故障域的标准，尽可能避免容量不平衡，并使迁移开销保持在合理低的水平。新的布局制定为重新平衡计划，描述需要应用于各个副本集以实现所需配置的更改。

在第二阶段，监视器通过在单独的NAD上启动重新同步任务来协调重新平衡计划的执行，以实现必要的数据迁移。 当副本需要移动时，迁移分三步进行：

1.将新副本添加到目标NAD

2.执行重新同步任务来传输数据

3.旧副本从源NAD中删除

这需要两个重新配置事件副本集，第一个扩展它包括新的副本，第二个在重新同步完成后修剪原件。 监视器在所有NAD和客户机之间协调这个过程，以获得所有修改后的虚拟对象。

**5.2.2 设备故障**

Strata确定NAD在从响应的NAD收到硬件故障通知，例如闪存设备失败或错误计数过多，或者观察到NAD停止响应超过可配置的超时请求时发生了故障。无论哪种情况，监视器都会通过使NAD脱机并启动全系统重新配置来修复冗余。

监视器在离线状态下执行的第一件事是将其与数据路径VLAN断开连接。这是直接与环境中的以太网交换机集成的一个很大的好处：在采取纠正措施之前，NAD为所有请求流量同步地从网络断开连接，避免由于出现过载组件失败，然后在不一致的状态超时后返回。而不是试图用完全端主机机制，如看门狗触发重新启动，或者双方达成的协议，向所有客户提供NAD的失败，地层禁用VLAN，并要求未能NAD的（单独）控制VLAN重新连接事件它在未来恢复生机。

从这一点来看，恢复逻辑是直截了当的。NAD在配置数据库中被标记为失败，并且启动重新平衡作业以修复在失败的NAD上包含副本的任何副本集。

**5.2.3 弹性扩展**

Strata响应引入新硬件的方式与响应故障的方式大致相同。 当监视器注意到已经安装了新的硬件时，它使用再平衡设施来生成合并新设备的布局。 由于通常在条带下方配置复制，因此可以按照各个条带的粒度迁移虚拟对象，从而允许单个条带化文件利用多个设备的聚合性能。 无论是整个文件还是单个条带，都可以使用现有的resync机制将对象移动到另一个NAD，即使文件处于联机状态。 以受控方式填充新的NAD，以限制后台IO对活动客户端工作负载的影响。

**6.存储协议**

Strata通过为托管协议服务器提供执行运行时来支持传统协议。协议在调度接口之上被构建为精简的表示层; 多个协议实例可以并行操作。实现还可以利用基于SDN的协议扩展来跨分布式运行时环境透明地分布多个客户端。

**6.1 可扩展的NFS**

Strata的设计使应用程序开发人员可以主要专注于实现协议规范，而不用担心如何在磁盘上组织数据，但大家期望很多存储协议可以作为提供调度库的简单包装来实现。 例如，NFS实现非常干净地映射到高级调度API，只提供协议特定的扩展，如RPC编组和NFS样式的访问控制。它利用配置数据库来存储NFS名称空间和后端对象之间的映射，并且仅依赖分条和复制处理器来实现数据路径。 而且，Strata允许在多个后端节点上实例化NFS服务器，自动在后端计算资源上分配额外的处理开销。

**6.2 SDN的协议缩放**

扩展传统存储协议可能具有挑战性，特别是当协议最初不是为分布式后端设计时。对于传统阵列来说，协议可扩展性的限制可能不会造成严重的问题，传统的阵列已经处于相对狭窄的网络接口之下，但是它们可能成为Strata分布式架构的性能瓶颈。

限制传统IP存储协议的访问带宽的核心属性是在单个IP地址后面呈现存储服务器。幸运的是，新兴的“软件定义”网络（SDN）交换机提供的接口允许应用程序通过以太网交换机对数据包转发进行比以前更精确的控制。

使用OpenFlow协议，软件控制器可以通过将特定于流的规则推送到交换机的转发路径来与交换机进行交互。OpenFlow规则实际上是通配的数据包过滤器和相关的操作，它们告诉交换机在识别匹配的数据包时该做什么。SDN交换机解释这些流量规则，并将其推送到交换机的TCAM或L2 / L3转发表中。

通过在单个流量的粒度上操纵交换机的流量，Strata协议实现能够向多个客户端呈现单个逻辑IP地址。无论何时打开新的NFS会话，都会在交换机上安装规则以触发故障事件，并且生成的异常路径将确定将该会话转发到哪个协议实例。服务会监视网络活动并根据需要迁移客户端连接，以保持均匀的工作负载分配。

协议扩展API封装并扩展了传统的套接字API，允许协议实现绑定到所有实例的共享IP地址上并侦听。然后，客户端负载均衡器监视所有这些连接上的流量需求，并启动流迁移，以响应任何单独的物理连接上的过载。

最简单的形式是客户端迁移完全在传输层处理。当协议负载平衡器注意到特定的NAD过载时，它会更新路由表，将最繁忙的客户端工作负载重定向到不同的NAD。一旦客户端的流量被转移，它将从新的NAD收到一个TCP RST，并建立一个新的连接，从而将流量透明地迁移到新的NAD。

Strata还为需要应用层协调以确保迁移安全的情况提供了挂钩。例如，NFS实现向负载平衡器注册了一个预迁移例程，它允许源NFS服务器在连接重定向到目标服务器之前，清除任何挂起的非幂等请求（例如创建或删除）。

**七、结论**

存储系统设计面临着由于其组件介质性能密度急剧增长而导致的海洋变化。现在，即使是少量的网络连接的闪存设备构成的分布式存储系统也能够匹配传统系统提供的负载，而传统的系统将需要多个旋转磁盘机架。

Strata是一种企业级存储体系结构，可以响应PCIe存储设备的性能特点。通过使用均衡的闪存，计算和网络资源构建模块，然后将设计与基于SDN的以太网交换机集成配对，Strata提供了一个增量式部署，可动态扩展的存储系统。

Strata最初的设计专门针对VMware ESX的企业部署，这是当今企业环境中新存储部署的主要推动力之一。该系统为这个特定的NFS环境实现了高性能和可扩展性，同时允许应用程序通过新协议直接与虚拟化的网络闪存硬件进行交互。这是通过将存储实现清晰地划分为底层，低开销的虚拟化层和用于实现存储协议的可伸缩框架来实现的。