Oracle 数据库

Enterprise applications

NetApp
May 19, 2024

目录

Oracle 数据库

基于ONTAP的Oracle数据库

ONTAP专为Oracle数据库而设计。几十年来、ONTAP针对关系数据库I/O的独特需求进行了优化、并专门为满足Oracle数据库的需求而创建了多种ONTAP功能、甚至是应Oracle Inc.本身的要求也是如此。

> 本文档将取代以前发布的技术报告_TR-3633：《基于ONTAP的Oracle数据库》；TR-4591：《Oracle数据保护：备份、恢复、复制》；TR-4592：《基于MetroCluster的Oracle》；以及TR-4534：《将Oracle数据库迁移到NetApp存储系统》_

除了ONTAP为数据库环境带来价值的多种可能方式之外、还有各种各样的用户要求、包括数据库大小、性能要求和数据保护需求。NetApp存储的已知部署包括从在VMware ESX下运行的大约6、000个数据库的虚拟化环境到当前大小为996 TB且在不断增长的单实例数据仓库等所有内容。因此、在NetApp存储上配置Oracle数据库的明确最佳实践很少。

在NetApp存储上运行Oracle数据库的要求可通过两种方式来满足。首先、当存在明确的最佳实践时、我们将专门予以说明。概括地说、我们将根据Oracle存储解决方案架构师的特定业务需求来说明他们必须考虑的许多设计注意事项。

ONTAP 配置

RAID和Oracle数据库

RAID是指使用冗余来保护数据不受驱动器丢失的影响。

在配置用于Oracle数据库和其他企业应用程序的NetApp存储时、有时会出现有关RAID级别的问题。有关存储阵列配置的许多传统Oracle最佳实践都包含有关使用RAID镜像和(或)避免使用某些类型的RAID的警告。尽管这些来源可以提供有效的支持、但它们并不适用于RAID 4以及ONTAP中使用的NetApp RAID DP和RAID-TEC技术。

RAID 4、RAID 5、RAID 6、RAID DP和RAID-TEC均使用奇偶校验来确保驱动器故障不会导致数据丢失。与镜像相比、这些RAID选项提供的存储利用率要高得多、但大多数RAID实施都有一个影响写入操作的缺点。在其他RAID实施中完成写入操作可能需要多次驱动器读取才能重新生成奇偶校验数据、此过程通常称为RAID惩罚。

但是、ONTAP不会受到这种RAID惩罚。这是因为NetApp WAFL (任意位置写入文件布局)与RAID层集成在一起。写入操作会在RAM中进行聚合、并准备为完整的RAID条带、包括奇偶校验生成。ONTAP无需执行读取即可完成写入、这意味着ONTAP和WAFL可以避免RAID惩罚。重做日志记录等延迟关键型操作的性能不受阻碍、随机数据文件写入不会因需要重新生成奇偶校验而产生任何RAID影响。

在统计可靠性方面、即使RAID DP也能提供比RAID镜像更好的保护。主要问题是RAID重建期间对驱动器的需求。使用镜像RAID集时、在重建到RAID集中的配对驱动器时、由于驱动器故障而导致数据丢失的风险远高于RAID DP集中三驱动器故障的风险。

Oracle数据库和存储容量管理

要使用可预测、可管理的高性能企业存储管理数据库或其他企业应用程序、需要在驱动器上留出一些可用空间来管理数据和元数据。所需的可用空间量取决于使用的驱动器类型和

业务流程。

可用空间是指未用于实际数据的任何空间、其中包括聚合本身上的未分配空间以及成分卷中的未使用空间。此外、还必须考虑精简配置。例如、卷可能包含1 TB的LUN、其中实际数据仅利用了50%的空间。在精简配置环境中、这将正确地显示为占用500 GB的空间。但是、在完全配置的环境中、1 TB的全部容量似乎正在使用中。500 GB的未分配空间将被隐藏。实际数据未使用此空间、因此应将此空间计入总可用空间。

对于用于企业级应用程序的存储系统、NetApp建议如下：

SSD聚合、包括**AFF**系统

> * NetApp建议*至少留出10%的可用空间。这包括所有未使用的空间、包括聚合或卷中的可用空间、以及由于使用完全配置而分配但实际数据未使用的任何可用空间。逻辑空间并不重要、问题是可用于数据存储的实际可用物理空间有多少。

建议10%的可用空间非常保守。SSD聚合可以支持利用率更高的工作负载、而不会对性能产生任何影响。但是、随着聚合利用率的提高、如果不仔细监控利用率、用尽空间的风险也会增加。此外、在以99%的容量运行系统时、可能不会影响性能、但可能会导致管理工作、试图在订购额外硬件时阻止系统完全填满、并且可能需要一些时间来采购和安装额外的驱动器。

HDD聚合、包括**Flash Pool**聚合

> * NetApp建议*使用旋转驱动器时至少留出15%的可用空间。这包括所有未使用的空间、包括聚合或卷中的可用空间、以及由于使用完全配置而分配但实际数据未使用的任何可用空间。当言论自由接近10%时、性能将受到影响。

Oracle数据库和Storage Virtual Machine

Oracle数据库存储管理集中在Storage Virtual Machine (SVM)上

SVM (在ONTAP命令行界面中称为Vserver)是存储的基本功能单元、将SVM与VMware ESX服务器上的子系统进行比较非常有用。

首次安装时、ESX没有预配置的功能、例如托管子操作系统或支持最终用户应用程序。在定义虚拟机(VM)之前、此容器为空容器。ONTAP与此类似。首次安装ONTAP时、在创建SVM之前、它无法提供数据。SVM特性用于定义数据服务。

与存储架构的其他方面一样、SVM和逻辑接口(LIF)设计的最佳选项在很大程度上取决于扩展要求和业务需求。

svms

目前尚无为ONTAP配置SVM的官方最佳实践。正确的方法取决于管理和安全要求。

大多数客户都会运行一个主SVM来满足大多数日常需求、但随后会创建少量SVM来满足特殊需求。例如、您可能希望创建：

- 由专业团队管理的关键业务数据库的SVM
- 开发组的SVM、该开发组拥有完全的管理控制权、因此可以独立管理自己的存储
- 用于敏感业务数据(例如人力资源或财务报告数据)的SVM、必须限制管理团队

在多租户环境中、可以为每个租户的数据提供一个专用SVM。每个集群、HA对和节点的SVM和LIP数量限制取决于所使用的协议、节点型号和ONTAP版本。 请参见 "NetApp Hardware Universe" 的限制。

使用ONTAP QoS进行Oracle数据库性能管理

安全高效地管理多个Oracle数据库需要有效的QoS策略。原因在于现代存储系统的性能不断提高。

具体而言、随着全闪存存储的采用率不断提高、工作负载得以整合。由于旧旋转驱动器技术的IOPS功能有限、依赖旋转介质的存储阵列往往仅支持数量有限的I/O密集型工作负载。早在存储控制器达到其限制之前、一两个高度活跃的数据库就会使底层驱动器饱和。这种情况已经改变。即使是功能最强大的存储控制器、数量相对较少的SSD驱动器的性能也可能会饱和。这意味着、可以充分利用控制器的全部功能、而不必担心随着旋转介质延迟峰值而导致性能突然崩溃。

作为一个参考示例、一个简单的双节点HA AFF A800系统能够在延迟超过1秒之前提供高达100万次随机IOPS。只有极少数单个工作负载才能达到此级别。要充分利用此AFF A800系统阵列、需要托管多个工作负载、而安全地执行此操作、同时确保可预测性、则需要QoS控制。

ONTAP中有两种类型的服务质量(QoS)：IOPS和带宽。QoS控制可应用于SVM、卷、LUN和文件。

IOPS QoS

IOPS QoS控制显然基于给定资源的总IOPS、但IOPS QoS的许多方面可能并不直观。最初、一些客户对达到IOPS阈值时延迟明显增加感到很不明白。限制IOPS自然会导致延迟增加。从逻辑上讲、它的功能类似于令牌系统。例如、如果包含数据文件的给定卷具有10000 IOPS限制、则到达的每个I/O都必须先接收令牌才能继续处理。只要在给定的一秒内使用的令牌不超过10000个、就不会出现延迟。如果IO操作必须等待接收其令牌、则此等待将显示为额外的延迟。工作负载越难超过QoS限制、每个IO在队列中等待处理的时间就越长、这在用户看来是延迟越高。

> 对数据库事务/重做日志数据应用QoS控制时要小心。虽然重做日志记录的性能需求通常要比数据文件低很多、但重做日志活动会变得突发。IO会以短暂的脉冲发生、并且似乎适合平均重做IO级别的QoS限制对于实际要求可能过低。结果可能会造成严重的性能限制、因为QoS会与每个重做日志突发事件结合使用。通常、重做和归档日志记录不应受QoS的限制。

带宽QoS

并非所有I/O大小都相同。例如、数据库可能会执行大量小块读取、从而导致达到IOPS阈值、 但是、数据库可能还会执行完整的表扫描操作、其中包含非常少量的大型块读取、占用的带宽非常大、但IOPS相对较少。

同样、VMware环境可能会在启动期间产生大量随机IOPS、但在外部备份期间执行的IO会更少、但会更大。

有时、有效管理性能需要IOPS或带宽QoS限制、甚至两者都需要。

最低/有保障的QoS

许多客户都希望解决方案能够提供有保障的QoS、而这种服务质量比看起来更难实现、而且可能会造成大量浪费。例如、如果要将10个数据库放置在10000 IOPS保证下、则需要对系统进行规模估算、以应对所有10个数据库同时以10000 IOPS运行的情形、总共需要100K IOPS。

最低QoS控制的最佳用途是保护关键工作负载。例如、假设一个ONTAP控制器的最大可能IOPS为50万次、并混合了生产和开发工作负载。您应将最大QoS策略应用于开发工作负载、以防止任何给定数据库独占控制器。然后、您可以对生产工作负载应用最低QoS策略、以确保它们在需要时始终具有所需的可用IOPS。

自适应 QoS

自适应服务质量(QoS)是指ONTAP功能、其中服务质量(QoS)限制基于存储对象的容量。它很少用于数据库、因为数据库大小与其性能要求之间通常没有任何关联。大型数据库可能几乎处于无活动状态、而小型数据库则可能是IOPS密集型最高的数据库。

自适应QoS对于虚拟化数据存储库非常有用、因为此类数据集的IOPS要求往往与数据库的总大小相关。包含1 TB VMDK文件的较新数据存储库所需的性能可能是2 TB数据存储库的一半左右。自适应QoS允许您在数据存储库中填充数据时自动增加QoS限制。

Oracle数据库和ONTAP效率功能

ONTAP空间效率功能针对Oracle数据库进行了优化。在几乎所有情况下、最佳方法都是保留默认值并启用所有效率功能。

数据压缩、数据缩减和重复数据删除等空间效率功能旨在增加给定物理存储量所需的逻辑数据量。这样可以降低成本和管理开销。

从较高层面来看、数据压缩是一个数学过程、通过该过程、可以检测数据模式并对其进行编码、从而减少空间需求。相反、重复数据删除会检测实际重复的数据块并删除无关的副本。数据缩减允许多个逻辑数据块共享介质上的同一物理块。

> 有关存储效率与预留百分比之间交互的说明、请参见以下有关精简配置的章节。

压缩

在全闪存存储系统推出之前、基于阵列的数据压缩的价值有限、因为大多数I/O密集型工作负载都需要大量磁盘轴才能提供可接受的性能。由于驱动器数量众多、存储系统所含容量总是远远超出所需容量。随着固态存储的兴起、这种情况发生了变化。不再需要纯粹为了获得良好的性能而大量过度配置驱动器。存储系统中的驱动器空间可以与实际容量需求相匹配。

与旋转驱动器相比、固态驱动器(SSD)的IOPS功能提高几乎始终可以节省成本、但数据压缩可以通过增加固态介质的有效容量来进一步节省成本。

数据压缩方法有多种。许多数据库都具有自己的数据压缩功能、但在客户环境中很少出现这种情况。原因通常是对压缩数据*进行更改*会对性能造成影响、此外、对于某些应用程序、数据库级数据压缩的许可成本较高。最后、还会对数据库操作产生整体性能影响。为执行数据压缩和解压缩的CPU支付较高的每CPU许可证成本毫无意义、而不是实际的数据库工作。更好的选择是将压缩工作负载分流到存储系统。

自适应数据压缩

自适应数据压缩已针对企业级工作负载进行了全面测试、未观察到对性能的影响、即使在延迟以微秒为单位的全闪存环境中也是如此。一些客户甚至报告说、使用数据压缩后性能会提高、因为数据会在缓存中保持压缩状态、从而有效地增加了控制器中的可用缓存量。

ONTAP以4 KB为单位管理物理块。自适应数据压缩使用默认的压缩块大小8 KB、这意味着数据以8 KB单位进行压缩。这与关系数据库最常使用的8 KB块大小匹配。随着将更多数据作为一个单元进行压缩、数据压缩算法的效率也会提高。32 KB压缩块大小比8 KB压缩块单元更节省空间。这确实意味着、使用默认8 KB块大小的自适应数据压缩确实会使效率率略低、但使用更小的数据压缩块大小也会有显著优势。数据库工作负载包含大量覆盖活动。要覆盖经过压缩的32 KB数据块中的8 KB、需要回读整个32 KB逻辑数据、对其进行解压缩、更新所需的8 KB区域、重新压缩、然后将整个32 KB写入驱动器。这对存储系统来说是一项非常昂贵的操作、因此、某些基于较大压缩块大小的竞争存储阵列也会对数据库工作负载的性能造成严重影响。

自适应数据压缩使用的块大小最多可以增加到32 KB。这可能会提高存储效率、对于事务日志和备份文件等不活动的文件、如果阵列上存储了大量此类数据、则应考虑使用此方法。在某些情况下、使用16 KB或32 KB块大小的活动数据库也可以通过增加要匹配的自适应数据压缩的块大小来受益。请咨询NetApp或合作伙伴代表、了解这是否适合您的工作负载。

在流式备份目标上、不应同时使用大于8 KB的数据压缩块大小和重复数据删除。原因是、对备份的数据所做的微小更改会影响32 KB数据压缩窗口。如果窗口发生变化、则生成的压缩数据会在整个文件中有所不同。重复数据删除在数据压缩后进行、这意味着重复数据删除引擎对每个压缩备份的看法不同。如果需要对流式备份进行重复数据删除、则只应使用8 KB块自适应数据压缩。最好使用自适应数据压缩、因为它的块大小较小、不会影响重复数据删除的效率。出于类似的原因、主机端压缩也会影响重复数据删除效率。

数据压缩对齐

数据库环境中的自适应数据压缩需要在一定程度上考虑数据压缩块对齐问题。对于随机覆盖非常特定的块的数据来说、这样做只是一个问题。这种方法在概念上类似于整体文件系统对齐、即文件系统的起点必须与4 k设备边界对齐、文件系统的块大小必须是4 k的倍数。

例如、只有当8 KB写入文件与文件系统本身内的8 KB边界对齐时、才会对其进行压缩。这一点意味着它必须位于文件的前8 KB、文件的后8 KB、依此类推。要确保正确对齐、最简单的方法是使用正确的LUN类型、创建的任何分区都应与设备起始位置偏移8K的倍数、并使用数据库块大小的倍数作为文件系统块大小。

备份或事务日志等数据是跨多个块按顺序写入的操作、所有这些块都会进行压缩。因此、无需考虑对齐。唯一关注的I/O模式是随机覆盖文件。

数据缩减

数据缩减是一项可提高数据压缩效率的技术。如前文所述、自适应数据压缩本身最多可节省2：1的空间、因为它仅限于在4 KB WAFL块中存储8 KB I/O。块大小越大、压缩方法的效率越高。但是、它们不适用于受到小块覆盖的数据。解压缩32 KB数据单元、更新8 KB部分、重新压缩以及回写驱动器会产生开销。

数据缩减的工作原理是、允许将多个逻辑块存储在物理块中。例如、具有高度可压缩数据(例如文本或部分全满块)的数据库可以从8 KB压缩到1 KB。如果不进行数据缩减、这1 KB的数据仍会占用整个4 KB块。实时数据缩减允许将1 KB的压缩数据与其他压缩数据一起存储在仅1 KB的物理空间中。它不是一种压缩技术；它只是一种在驱动器上分配空间的更高效的方式、因此不会产生任何可检测的性能影响。

节省的资金数额各不相同。已压缩或加密的数据通常无法进一步压缩、因此、数据集无法从数据缩减中受益。相比之下、新初始化的数据文件包含的块元数据和零数据略多、数据压缩率高达80：1。

对温度敏感的存储效率

温度敏感型存储效率(TSSE)在ONTAP 9.8及更高版本中提供、它依靠块访问热图来识别不常访问的块并以更高的效率对其进行压缩。

重复数据删除

重复数据删除是指从数据集中删除重复的块大小。例如、如果10个不同文件中存在相同的4 KB块、则重复数据删除会将所有10个文件中的4 KB块重定向到相同的4 KB物理块。结果是、这些数据的效率将提高10：1。

VMware子系统启动LUN等数据的重复数据删除效果通常非常好、因为它们包含同一操作系统文件的多个副本。我们观察到的效率为100：1甚至更高。

某些数据不包含重复数据。例如、Oracle块包含数据库全局唯一的标头和几乎唯一的尾部。因此、对Oracle数据库进行重复数据删除很少能节省超过1%的空间。对MS SQL数据库执行重复数据删除略有改进、但块级别的唯一元数据仍是一个限制。

在少数情况下、使用16 KB和大型块的数据库可节省多达15%的空间。每个块的初始4 KB包含全局唯一标头、而最后4 KB块包含接近唯一的尾部。内部块是重复数据删除的候选数据、但实际上、这几乎完全是由于对置零数据进行重复数据删除。

许多争用资源的阵列都声称可以根据数据库被复制多次的假设对数据库进行重复数据删除。在这方面、也可以使用NetApp重复数据删除、但ONTAP提供了一个更好的选择：NetApp FlexClone技术。最终结果是相同的；系统会为一个数据库创建多个副本、这些副本共享大多数底层物理块。与花时间复制数据库文件并对其进行重复数据删除相比、使用FlexClone的效率要高得多。实际上、它是无重复数据删除、而不是重复数据删除、因为从一开始就不会创建重复数据。

效率和精简配置

效率功能是精简配置的一种形式。例如、占用100 GB卷的100 GB LUN可能会压缩到50 GB。由于卷仍为100 GB、因此尚未实现实际节省。必须先减小卷大小、以便节省的空间可用于系统上的其他位置。如果稍后更改100 GB LUN会导致数据的可压缩性降低、则LUN大小会增大、卷可能会填满。

强烈建议使用精简配置、因为它可以简化管理、同时显著提高可用容量并节省相关成本。原因很简单—数据库环境通常包含大量空空间、大量卷和LUN以及可压缩数据。厚配置会为卷和LUN预留存储空间、以防它们最终达到100%全满并包含100%不可压缩数据。这种情况不大可能发生。通过精简配置、可以回收这些空间并将其用于其他位置、并可以基于存储系统本身进行容量管理、而不是基于许多较小的卷和LUN。

有些客户更喜欢对特定工作负载使用厚配置、或者通常根据既定的运营和采购实践使用厚配置。

*注意：*如果卷配置厚配置、则必须小心操作、以便完全禁用该卷的所有效率功能、包括使用解压缩和删除重复数据删除 `sis undo` 命令：此卷不应显示在中 `volume efficiency show` 输出。如果配置了效率功能、则仍会为卷部分配置效率功能。因此、覆盖保证的工作方式有所不同、这会增加配置忽略发生原因卷以意外用尽空间的可能性、从而导致数据库I/O错误。

效率最佳实践

NetApp建议执行以下操作：

AFF默认值

在纯闪存AFF系统上运行的ONTAP上创建的卷经过精简配置、并启用了所有实时效率功能。尽管数据库通常不会从重复数据删除中受益、并且可能包含不可压缩的数据、但默认设置适用于几乎所有工作负载。ONTAP旨在高效处理所有类型的数据和I/O模式、无论它们是否可节省空间。只有在完全了解原因且有优势的情况下、才应更改默认值。

一般建议

- 如果卷和(或) LUN未进行精简配置、则必须禁用所有效率设置、因为使用这些功能不会节省空间、而将厚配置与已启用空间效率相结合会发生原因发生意外行为、包括空间不足错误。
- 如果数据不会被覆盖(例如使用备份或数据库事务日志)、则可以通过在较低的冷却期启用TSSE来提高效率。
- 某些文件可能包含大量不可压缩数据、例如、在应用程序级别已启用数据压缩时、文件已加密。如果出现上述任一情况、请考虑禁用数据压缩、以便在包含可压缩数据的其他卷上执行更高效的操作。
- 不要在数据库备份中同时使用32 KB数据压缩和重复数据删除。请参见一节 [自适应数据压缩] 了解详细信息。

使用Oracle数据库进行精简配置

对Oracle数据库进行精简配置需要仔细规划、因为这样会导致在存储系统上配置的空间超过实际可用的空间。这是非常值得的努力、因为如果操作正确、可以显著节省成本并提高易管理性。

精简配置有多种形式、是ONTAP为企业级应用程序环境提供的许多功能不可或缺的组成部分。精简配置也与效率技术密切相关、原因相同：效率功能允许存储的逻辑数据比存储系统上的技术数据多。

几乎任何快照使用都涉及精简配置。例如、NetApp存储上的典型10 TB数据库包含大约30天的快照。这种安排会使活动文件系统中显示大约10 TB的数据、并将300 TB专用于快照。总存储量为312 TB、通常占用大约12 TB到15 TB的空间。活动数据库会占用10 TB的空间、其余300 TB的数据仅需要2 TB到5 TB的空间、因为系统仅会存储对原始数据所做的更改。

克隆也是精简配置的一个示例。一家主要NetApp客户为一个80 TB数据库创建了40个克隆、以供开发使用。如果使用这些克隆的所有40位开发人员都覆盖了每个数据文件中的每个块、则需要3.2 PB以上的存储。实际上、周转率较低、并且总空间需求接近40 TB、因为驱动器上仅存储更改。

空间管理

对应用程序环境进行精简配置时必须格外小心、因为数据变更率可能会意外增加。例如、如果为数据库表重新编制索引或对VMware子系统应用大规模修补、则快照占用的空间会快速增长。放错位置的备份可能会在很短的时间内写入大量数据。最后、如果文件系统意外用尽可用空间、则很难恢复某些应用程序。

幸运的是、这些风险可以通过仔细配置来解决 `volume-autogrow` 和 `snapshot-autodelete` 策略。正如其名称所暗示的那样、这些选项使用户能够创建策略、以自动清除快照占用的空间或增加卷以容纳更多数据。有多种选项可供选择、不同客户的需求也会有所不同。

请参见 "逻辑存储管理文档" 有关这些功能的完整讨论。

预留百分比

预留百分比是指卷中LUN在空间效率方面的行为。选项 `fractional-reserve` 设置为100%时、卷中的所有数据在使用任何数据模式时均可实现100%的周转率、而不会耗尽卷上的空间。

例如、假设数据库位于1 TB卷中的一个250 GB LUN上。创建快照会立即在卷中预留额外的250 GB空间、以保证卷不会因任何原因用尽空间。使用预留百分比通常会造成浪费、因为数据库卷中的每个字节极不可能需要覆盖。没有理由为从未发生的事件预留空间。但是、如果客户无法监控存储系统中的空间消耗、并且必须确保空间永远不会用尽、则需要100%预留百分比才能使用快照。

数据压缩和重复数据删除

数据压缩和重复数据删除都是精简配置的两种形式。例如、50 TB的数据占用空间可能会压缩为30 TB、从而节省20 TB的空间。要使数据压缩产生任何优势、必须将这20 TB中的一部分用于其他数据、或者购买的存储系统必须小于50 TB。这样、存储的数据就会超过存储系统上的技术可用数据。从数据角度来看、数据容量为50 TB、尽管它在驱动器上仅占用30 TB。

数据集的可压缩性总是有可能发生变化、从而导致实际空间消耗增加。这种消耗量的增加意味着、在监控和使用方面、必须像其他形式的精简配置一样管理数据压缩 `volume-autogrow` 和 `snapshot-autodelete`。

有关数据压缩和重复数据删除的详细信息、请参见链接efficiency.html

数据压缩和预留百分比

数据压缩是一种精简配置形式。预留百分比会影响数据压缩的使用、但需要注意的一点是、空间是在创建快照之前预留的。通常、只有当存在快照时、预留百分比才重要。如果没有快照、则预留百分比并不重要。而数据压缩则不是这种情况。如果在已进行数据压缩的卷上创建了LUN、则ONTAP会保留空间以容纳快照。此行为在配置期间可能会令人困惑、但这是预期行为。

例如、假设一个10 GB的卷具有一个5 GB的LUN、该LUN已压缩为2.5 GB、并且没有快照。请考虑以下两种情形：

- 预留百分比= 100会导致利用率达到7.5 GB
- 预留百分比= 0会导致利用率达到2.5 GB

第一种情形包括：当前数据占用2.5 GB空间、而源在预计快照使用时的周转率为100%时占用5 GB空间。第二种情形不会预留任何额外空间。

虽然这种情况可能看起来令人困惑、但在实践中不太可能遇到。数据压缩意味着精简配置、而在LUN环境中进行精简配置需要预留百分比。压缩的数据始终可以被不可压缩的内容覆盖、这意味着必须对卷进行精简配置、才能进行压缩、从而节省空间。

- NetApp建议*采用以下预留配置：
- 设置 `fractional-reserve` 如果已实施基本容量监控、则为0 `volume-autogrow` 和 `snapshot-autodelete`。
- 设置 `fractional-reserve` 如果没有监控能力或在任何情况下都无法排空空间、则为100。

可用空间和**LVM**空间分配

随着数据被删除、文件系统环境中活动LUN的精简配置效率可能会逐渐降低。除非删除的数据被零覆盖(另请参见 "ASMRU" 或者通过TRIM/UNMAP空间回收释放空间、"已擦除"的数据会在文件系统中占用越来越多的未分配空格。此外、活动LUN的精简配置在许多数据库环境中的用途有限、因为数据文件在创建时会初始化为其完整大小。

仔细规划LVM配置可以提高效率、并最大限度地减少存储配置和LUN大小调整的需求。使用Veritas VLVM或Oracle ASM等LVM时、底层LUN会划分为仅在需要时才使用的块区。例如、如果数据集的大小从2 TB开始、但随着时间的推移可能会增长到10 TB、则可以将此数据集放置在LVM磁盘组中组织的10 TB精简配置LUN上。在创建时、它只会占用2 TB的空间、并且只会在为满足数据增长而分配块区时占用额外空间。只要空间受到监控、此过程就会很安全。

Oracle数据库和ONTAP控制器故障转移/切换

要确保Oracle数据库操作不会被这些操作中断、需要了解存储接管和切换功能。此外、如果使用不当、接管和切换操作使用的参数可能会影响数据完整性。

- 在正常情况下、传入给定控制器的写入会同步镜像到其配对控制器。在NetApp MetroCluster环境中、写入操作也会镜像到远程控制器。写入操作在所有位置的非易失性介质中存储之前、不会向主机应用程序确认。
- 存储写入数据的介质称为非易失性内存或NVMEM。它有时也称为非易失性随机存取存储器(NVRAM)、尽管它充当日志、但也可视为写入缓存。在正常操作下、不会读取NVMEM中的数据；该数据仅用于在发生软件或硬件故障时保护数据。将数据写入驱动器后、数据将从系统中的RAM传输、而不是从NVMEM传输。

- 在接管操作期间、高可用性(HA)对中的一个节点会从其配对节点接管操作。切换本质上是相同的、但在IT适用场景 MetroCluster配置中、远程节点接管本地节点的功能。

在日常维护操作期间、存储接管或切换操作应该是透明的、而不是在网络路径发生更改时可能会短暂暂停操作。但是、网络连接可能很复杂、容易出错、因此NetApp强烈建议在将存储系统投入生产之前对接管和切换操作进行全面测试。这样做是确保所有网络路径配置正确的唯一方法。在SAN环境中、请仔细检查命令的输出 `sanlun lun show -p` 确保所有预期的主路径和辅助路径均可用。

发出强制接管或切换命令时必须小心。使用这些选项强制更改存储配置意味着、拥有驱动器的控制器的状态将被忽略、而备用节点将强制接管驱动器。不正确地强制执行接管可能会导致数据丢失或损坏。这是因为强制接管或切换可能会丢弃NVMEM的内容。接管或切换完成后、如果丢失这些数据、则从数据库的角度来看、存储在驱动器上的数据可能会还原到稍旧的状态。

很少需要使用普通HA对强制接管。在几乎所有故障情形下、节点都会关闭并通知配对节点、以便进行自动故障转移。在某些边缘情况下、例如、发生滚动故障时、节点之间的互连断开、然后一个控制器断开、此时需要强制接管。在这种情况下、节点之间的镜像会在控制器发生故障之前丢失、这意味着无故障控制器将不再具有正在进行的写入的副本。然后、需要强制执行接管、这意味着数据可能会丢失。

相同的逻辑适用场景会执行MetroCluster切换。在正常情况下、切换几乎是透明的。但是、灾难可能会导致运行正常的站点与灾难站点之间的连接断开。从运行正常的站点的角度来看、问题可能只是站点之间的连接中断、而原始站点可能仍在处理数据。如果节点无法验证主控制器的状态、则只能执行强制切换。

- NetApp建议*采取以下预防措施：
- 请格外小心、以免意外强制执行接管或切换。通常、不需要强制执行、强制执行更改可能会导致发生原因数据丢失。
- 如果需要强制接管或切换、请确保关闭应用程序、卸载所有文件系统并更改逻辑卷管理器(LVM)卷组。必须卸载ASM磁盘组。
- 如果发生强制MetroCluster切换、请将故障节点与所有运行正常的存储资源隔离。有关详细信息、请参见相关ONTAP版本的《MetroCluster管理和灾难恢复指南》。

MetroCluster和多个聚合

MetroCluster是一种同步复制技术、如果连接中断、则会切换到异步模式。这是客户最常见的请求、因为有保障的同步复制意味着站点连接中断会导致数据库I/O完全停止、从而使数据库无法使用。

借助MetroCluster、聚合可以在连接恢复后快速重新同步。与其他存储技术不同、MetroCluster不应要求在站点发生故障后进行完整的重新镜像。只需发送增量变更。

在跨聚合的数据集中、在滚动灾难场景中需要执行额外的数据恢复步骤的风险很小。具体而言、如果：(a)站点之间的连接中断、(b)连接恢复、(c)聚合达到一种状态、其中一些聚合已同步、而另一些聚合则未同步、 然后(d)主站点丢失、结果是聚合未彼此同步的运行正常的站点。如果发生这种情况、数据集的部分内容会彼此同步、如果不进行恢复、则无法启动应用程序、数据库或数据存储库。如果数据集跨越多个聚合、NetApp强烈建议使用多种可用工具之一利用基于快照的备份来验证在这种异常情况下的快速可恢复性。

数据库配置

Oracle数据库块大小

ONTAP在内部使用可变块大小、这意味着可以为Oracle数据库配置所需的任何块大小。但

是、文件系统块大小可能会影响性能、在某些情况下、较大的重做块大小可以提高性能。

数据文件块大小

某些操作系统可选择文件系统块大小。对于支持Oracle数据文件的文件系统、使用数据压缩时、块大小应为8 KB。如果不需要数据压缩、则可以使用8 KB或4 KB的块大小。

如果将数据文件放置在具有512字节块的文件系统上、则可能会出现文件错位。LUN和文件系统可能已根据NetApp建议正确对齐、但文件I/O可能错位。这种错位会导致发生原因出现严重的性能问题。

支持重做日志的文件系统所使用的块大小必须是重做块大小的倍数。这通常要求重做日志文件系统和重做日志本身都使用512字节的块大小。

重做块大小

在重做率非常高的情况下、4 KB块大小的性能可能会更好、因为重做率较高、可以在更少、更高效的操作中执行I/O。如果重做速率大于50 Mbps、请考虑测试4 KB的块大小。

在块大小为4 KB的文件系统上使用块大小为512字节的重做日志和许多非常小的事务时、客户发现了一些数据库问题。对一个4 KB文件系统块应用多个512字节更改所涉及的开销导致了性能问题、这些问题通过将文件系统更改为使用512字节的块大小得以解决。

> * NetApp建议*不要更改重做块大小、除非相关客户支持或专业服务组织建议您更改块大小、或者更改基于官方产品文档。

Oracle数据库参数：db_file_multiblock_read_count

。 `db_file_multiblock_read_count` 参数用于控制Oracle在顺序I/O期间单次操作读取的Oracle数据库块的最大数量

但是、此参数不会影响Oracle在任何和所有读取操作期间读取的块数、也不会影响随机I/O只有顺序I/O的块大小会受到影响。

Oracle建议用户不要设置此参数。这样可以使数据库软件自动设置最佳值。这通常意味着、将此参数设置为一个可产生1 MB I/O大小的值。例如、读取1 MB的8 KB块需要读取128个块、因此、此参数的默认值为128。

NetApp在客户站点发现的大多数数据库性能问题都与此参数的设置不正确有关。在Oracle版本8和9中更改此值有充分的理由。因此、参数可能会在不为人所用的情况下出现在中 `init.ora` 文件、因为数据库已原位升级到Oracle 10及更高版本。与默认值128相比、原有设置8或16会严重损害顺序I/O性能。

> * NetApp建议*设置 `db_file_multiblock_read_count` 参数不应出现在中 `init.ora` 文件NetApp从未遇到过更改此参数可提高性能的情况、但在许多情况下、它会明显损坏顺序I/O吞吐量。

Oracle数据库参数：filesystemio_options

Oracle初始化参数 `filesystemio_options` 控制异步和直接I/O的使用

与通常的看法相反、异步I/O和直接I/O并不互相排斥。NetApp发现、在客户环境中、此参数经常配置不当、这种配置不当直接导致许多性能问题。

异步I/O意味着可以并行处理Oracle I/O操作。在各种操作系统上提供异步I/O之前、用户配置了大量dbwriter进程并更改了服务器进程配置。使用异步I/O时、操作系统本身会代表数据库软件以高度高效的并行方式执行I/O。此过程不会使数据面临风险、Oracle重做日志记录等关键操作仍会同步执行。

直接I/O会绕过操作系统缓冲区缓存。UNIX系统上的I/O通常流经操作系统缓冲区缓存。这对于不维护内部缓存的应用程序非常有用、但Oracle在SGA中具有自己的缓冲区缓存。在几乎所有情况下、最好启用直接I/O并将服务器RAM分配给SGA、而不是依赖操作系统缓冲区缓存。Oracle SGA可以更高效地使用内存。此外、当I/O流经操作系统缓冲区时、它会受到额外处理的影响、从而增加了缓存。当低延迟是一项关键要求时、对于写入I/O负载繁重的情况、延迟增加尤为明显。

的选项 `filesystemio_options` 是：

- **async.** Oracle将I/O请求提交给操作系统进行处理。此过程允许Oracle执行其他工作、而不是等待I/O完成、从而提高I/O并行处理能力。
- **directio.** Oracle直接对物理文件执行I/O、而不是通过主机操作系统缓存路由I/O。
- 无。 Oracle使用同步和缓冲I/O在这种配置中、在共享和专用服务器进程之间进行选择以及dbwriter的数量更重要。
- **setall.** Oracle同时使用异步和直接I/O在几乎所有情况下、使用 `setall` 最佳。

> 。 `filesystemio_options` 参数在DNFS和ASM环境中无效。使用DNFS或ASM会自动导致同时使用异步和直接I/O

某些客户过去遇到过异步I/O问题、尤其是在以前的Red Hat Enterprise Linux 4 (RHEL4)版本中。互联网上的一些过时建议仍然建议避免异步IO、因为信息过时。异步I/O在所有当前操作系统上均保持稳定。如果操作系统没有已知错误、则没有理由禁用它。

如果数据库一直在使用缓冲I/O、则切换到直接I/O可能还需要更改SGA大小。禁用缓冲的I/O可消除主机操作系统缓存为数据库提供的性能优势。将RAM重新添加到SGA可修复此问题。最终结果应该是I/O性能有所提高。

尽管Oracle SGA使用RAM几乎总是优于操作系统缓冲区缓存、但可能无法确定最佳值。例如、在具有许多间歇性活动Oracle实例的数据库服务器上、最好使用SGA大小非常小的缓冲I/O。这种安排允许所有正在运行的数据库实例灵活地使用操作系统上剩余的可用RAM。这是一种非常少见的情况、但在某些客户站点上也发现过。

> * NetApp建议*设置 `filesystemio_options` to `setall`但请注意，在某些情况下，丢失主机缓冲区缓存可能需要增加Oracle SGA。

Oracle Real Application Clusters (RAC)超时

Oracle RAC是一款集群软件产品、它具有多种类型的内部检测信号进程、用于监控集群的运行状况。

> 中的信息 "MissCount" 第节介绍了有关使用网络存储的Oracle RAC环境的重要信息、在许多情况下、需要更改默认Oracle RAC设置、以确保RAC集群在网络路径更改和存储故障转移/切换操作后不会受到影响。

磁盘超时

与存储相关的主RAC参数是 `disktimeout`。此参数用于控制表决文件I/O必须完成的阈值。如果 `disktimeout` 超过此参数后、RAC节点将从集群中逐出。此参数的默认值为200。此值对于标准存储接管和回

用过程应足够。

NetApp强烈建议在将RAC配置投入生产之前对其进行全面测试、因为许多因素会影响接管或恢复。除了完成存储故障转移所需的时间之外、传播链路聚合控制协议(Link Aggregate Control Protocol、LACP)更改也需要额外的时间。此外、SAN多路径软件必须检测到I/O超时、然后在备用路径上重试。如果数据库非常活跃、则在处理表决磁盘I/O之前、必须对大量I/O进行排队和重试。

如果无法执行实际的存储接管或恢复、则可以在数据库服务器上执行缆线拉拔测试来模拟这种影响。

- NetApp建议*：
- 退出 `disktimeout` 参数、默认值为200。
- 始终全面测试RAC配置。

MissCount

。 `misscount` 参数通常仅影响RAC节点之间的网络检测信号。默认值为30秒。如果网格二进制文件位于存储阵列上或操作系统启动驱动器不在本地、则此参数可能会变得很重要。这包括启动驱动器位于FC SAN上的主机、NFS启动的操作系统以及启动驱动器位于VMDK文件等虚拟化数据存储库上的主机。

如果存储接管或恢复中断了对启动驱动器的访问、则网格二进制位置或整个操作系统可能会暂时挂起。ONTAP完成存储操作以及操作系统更改路径和恢复I/O所需的时间可能会超过 `misscount` 阈值。因此、在恢复与启动LUN或网格二进制文件的连接后、节点会立即被逐出。在大多数情况下、发生逐出和后续重新启动时不会记录任何日志消息来指示重新启动的原因。并非所有配置都会受到影响、因此、请在RAC环境中测试任何SAN启动、NFS启动或基于数据存储库的主机、以便在与启动驱动器的通信中断时RAC保持稳定。

对于非本地启动驱动器或托管的非本地文件系统 `grid` 二进制文件、 `misscount` 需要更改才能匹配 `disktimeout`。如果更改了此参数、请执行进一步测试、以确定对RAC行为的任何影响、例如节点故障转移时间。

- NetApp建议*：
- 离开 `misscount` 参数、默认值为30、除非满足以下条件之一：
  - `grid` 二进制文件位于网络连接驱动器上、包括NFS、iSCSI、FC和基于数据存储库的驱动器。
  - 操作系统通过SAN启动。
- 在这种情况下、请评估影响操作系统或访问的网络中断的影响 `GRID_HOME` 文件系统。在某些情况下、此类中断发生原因会使Oracle RAC守护进程发生拖延、从而可能导致出现 `misscount`基于的超时和逐出。超时默认为27秒、即的值 `misscount` 减号 `reboottime`。在这种情况下，增加 `misscount` 至200以匹配 `disktimeout`。

主机配置

使用**IBM AIX**的**Oracle**数据库

使用ONTAP的IBM AIX上的Oracle数据库的配置主题。

并发I/O

要在IBM AIX上实现最佳性能、需要使用并发I/O如果不使用并发I/O、则性能可能会受到限制、因为AIX会执行序列化的原子I/O、从而产生大量开销。

最初、NetApp建议使用 `cio` 挂载选项、用于强制在文件系统上使用并发I/O、但此过程存在缺点、不再需要。自AIX 5.2和Oracle 10gR1推出以来、AIX上的Oracle可以打开单个文件以实现并发IO、而不是强制在整个文件系统上执行并发I/O。

启用并发I/O的最佳方法是设置 `init.ora` 参数 `filesystemio_options` to `setall`。这样、Oracle就可以打开特定文件、以便用于并发I/O

使用 `cio` 作为挂载选项、会强制使用并发I/O、这可能会产生负面影响。例如、强制执行并发I/O会在文件系统上禁用预读、这可能会损害Oracle数据库软件之外发生的I/O的性能、例如复制文件和执行磁带备份。此外，Oracle GoldenGate和SAP BR*Tools等产品与不兼容 `cio` 适用于某些Oracle版本的挂载选项。

- NetApp建议*：
- 请勿使用 `cio` 文件系统级别的挂载选项。而是通过使用来启用并发I/O `filesystemio_options=setall`。
- 仅使用 `cio` 如果无法设置、则应设置挂载选项 `filesystemio_options=setall`。

AIX NFS挂载选项

下表列出了Oracle单实例数据库的AIX NFS挂载选项。

文件类型	挂载选项
ADr主页	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144`
控制文件 数据文件 重做日志	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144`
`ORACLE_HOME`	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,intr`

下表列出了RAC的AIX NFS挂载选项。

文件类型	挂载选项
ADr主页	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144`
控制文件 数据文件 重做日志	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,nointr,noac`
`CRS/Voting`	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,nointr,noac`

文件类型	挂载选项
专用 ORACLE_HOME	rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144
共享 ORACLE_HOME	rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,nointr

单实例挂载选项与RAC挂载选项之间的主要区别在于添加了 noac 挂载选项。这种添加的效果是禁用主机操作系统缓存、从而使RAC集群中的所有实例都能获得一致的数据状态视图。

但使用 cio 挂载选项和 init.ora 参数 filesystemio_options=setall 与禁用主机缓存具有相同的效果、但仍需要使用 noac。 noac 对于共享为必填项 ORACLE_HOME 部署以提高Oracle密码文件和等文件的一致性 spfile 参数文件。RAC集群中的每个实例都有一个专用 ORACLE_HOME，则不需要此参数。

AIX jfs/JFS2挂载选项

下表列出了AIX jfs/jfs2挂载选项。

文件类型	挂载选项
ADr主页	默认值
控制文件 数据文件 重做日志	默认值
ORACLE_HOME	默认值

使用AIX之前 hdisk 在任何环境(包括数据库)中、设备都应检查参数 queue_depth。此参数不是HBA队列深度、而是与各个的SCSI队列深度相关 hdisk device. Depending on how the LUNs are configured, the value for `queue_depth 可能太低、无法获得良好性能。测试表明、最佳值为64。

使用HP-UX的Oracle数据库

使用ONTAP在HP-UX上配置Oracle数据库主题。

HP-UX NFS挂载选项

下表列出了单个实例的HP-UX NFS挂载选项。

文件类型	挂载选项
ADr主页	rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,suid
控制文件 数据文件 重做日志	rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,forcedirectio, nointr,suid

文件类型	挂载选项
ORACLE_HOME	rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,suid

下表列出了RAC的HP-UX NFS挂载选项。

文件类型	挂载选项
ADr主页	rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,noac,suid
控制文件 数据文件 重做日志	rw, bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,nointr,noac,forcedirectio,suid
CRS/表决	rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,nointr,noac,forcedirectio,suid
专用 ORACLE_HOME	rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,suid
共享 ORACLE_HOME	rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,nointr,noac,suid

单实例挂载选项与RAC挂载选项之间的主要区别在于添加了 noac 和 forcedirectio 挂载选项。这种添加的效果是禁用主机操作系统缓存、从而使RAC集群中的所有实例都能获得一致的数据状态视图。但使用 init.ora 参数 filesystemio_options=setall 与禁用主机缓存具有相同的效果、但仍需要使用 noac 和 forcedirectio。

原因 noac 对于共享为必填项 ORACLE_HOME 部署是为了提高Oracle密码文件和spfile等文件的一致性。RAC集群中的每个实例都有一个专用 ORACLE_HOME，则不需要此参数。

HP-UX VxFS挂载选项

对于托管Oracle二进制文件的文件系统、请使用以下挂载选项：

```
delaylog,nodatainlog
```

对于包含数据文件、重做日志、归档日志和控制文件的文件系统、如果HP-UX版本不支持并发I/O、请使用以下挂载选项：

```
nodatainlog,mincache=direct,convosync=direct
```

如果支持并发I/O (VxFS 5.0.1及更高版本或ServiceGuard Storage Management Suite)、请对包含数据文件、重做日志、归档日志和控制文件的文件系统使用以下挂载选项：

```
delaylog,cio
```

> 参数 `db_file_multiblock_read_count` 在VxFS环境中尤其重要。Oracle建议在Oracle 10g R1及更高版本中保持此参数未设置、除非另有明确指示。Oracle 8 KB块大小的默认值为128。如果此参数的值强制设置为16或更低、请删除 `convosync=direct` 挂载选项、因为它可能会损坏顺序I/O性能。此步骤会损害性能的其他方面、仅当的值为时才应执行此步骤 `db_file_multiblock_read_count` 必须更改默认值。

使用Linux的Oracle数据库

特定于Linux操作系统的配置主题。

Linux NFSv3 TCP插槽表

TCP插槽表相当于主机总线适配器(Host Bus Adapter、HBA)队列深度的NFSv3。这些表可控制任何时候都可以处理的NFS操作的数量。默认值通常为16、该值太低、无法实现最佳性能。在较新的Linux内核上会出现相反的问题、这会自动将TCP插槽表限制增加到使NFS服务器充满请求的级别。

为了获得最佳性能并防止出现性能问题、请调整控制TCP插槽表的内核参数。

运行 `sysctl -a | grep tcp.*.slot_table` 命令、并观察以下参数：

```
# sysctl -a | grep tcp.*.slot_table
sunrpc.tcp_max_slot_table_entries = 128
sunrpc.tcp_slot_table_entries = 128
```

所有Linux系统都应包括 `sunrpc.tcp_slot_table_entries`，但只有部分包括 `sunrpc.tcp_max_slot_table_entries`。它们都应设置为128。

小心
如果未设置这些参数、可能会对性能产生显著影响。在某些情况下、性能会受到限制、因为Linux操作系统发出的I/O不足在其他情况下、随着Linux操作系统尝试问题描述的I/O数超过可处理的I/O数、I/O时间会增加。

Linux NFS挂载选项

下表列出了单个实例的Linux NFS挂载选项。

文件类型	挂载选项
ADr主页	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144`

文件类型	挂载选项
控制文件 数据文件 重做日志	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,nointr`
ORACLE_HOME	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,nointr`

下表列出了RAC的Linux NFS挂载选项。

文件类型	挂载选项
ADr主页	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,actimeo=0`
控制文件 数据文件 重做日志	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,nointr,actimeo=0`
CRS/表决	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,nointr,noac,actimeo=0`
专用 `ORACLE_HOME`	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144`
共享 `ORACLE_HOME`	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,nointr,actimeo=0`

单实例挂载选项与RAC挂载选项之间的主要区别在于添加了 `actimeo=0` 挂载选项。这种添加的效果是禁用主机操作系统缓存、从而使RAC集群中的所有实例都能获得一致的数据状态视图。但使用 `init.ora` 参数 `filesystemio_options=setall` 与禁用主机缓存具有相同的效果、但仍需要使用 `actimeo=0`。

原因 `actimeo=0` 对于共享为必填项 `ORACLE_HOME` 部署是为了提高Oracle密码文件和spfile等文件的一致性。RAC集群中的每个实例都有一个专用 `ORACLE_HOME`，则不需要此参数。

通常、非数据库文件应使用与单实例数据文件相同的选项进行挂载、但特定应用程序可能具有不同的要求。避免使用挂载选项 `noac` 和 `actimeo=0` 如果可能、因为这些选项会禁用文件系统级预读和缓冲。这可能会发生原因为提取、转换和加载等过程带来严重的性能问题。

access和getattr

一些客户注意到、极高级别的其他IOPS (如访问和getATTR)可能会主导其工作负载。在极端情况下、读取和写入等操作可能只占总数的10%。这是包含使用的任何数据库的正常行为 `actimeo=0` 和 / 或 `noac` 在Linux上、因为这些选项会对Linux操作系统执行发生原因操作、以便不断地从存储系统中重新加载文件元数据。访问和getattr等操作是低影响操作、可通过数据库环境中的ONTAP缓存进行处理。不应将其视为真正的IOPS、例如读取和写入、因为它们会对存储系统产生真正的需求。但是、其他这些IOPS确实会产生一些负载、尤其是在RAC环境中。要解决这种情况、请启用DNFS、它会绕过操作系统缓冲区缓存并避免执行这些不必要的元数据操作。

Linux Direct NFS

一个额外的挂载选项、称为 `nosharecache`如果(a)启用了DNFS、并且(b)在单个服务器上多次挂载源卷(c)并使用嵌套NFS挂载、则需要使用此选项。此配置主要出现在支持SAP应用程序的环境中。例如、NetApp系统上的单个卷的目录可能位于 `/vol/oracle/base 然后、再按 /vol/oracle/home。条件 /vol/oracle/base 挂载于 /oracle 和 /vol/oracle/home 挂载于 `/oracle/home`的结果是来自同一源的嵌套NFS挂载。

操作系统可以检测到这一事实 /oracle 和 /oracle/home 位于同一个卷上、即同一个源文件系统。然后、操作系统会使用相同的设备句柄来访问数据。这样做可以改进操作系统缓存和某些其他操作的使用、但会干扰DNFS。如果DNFS必须访问某个文件、例如 spfile，开 /oracle/home，则可能会错误地尝试使用错误的数据路径。结果是I/O操作失败。在这些配置中、添加 nosharecache 将选项挂载到与该主机上的另一个NFS文件系统共享源FlexVol卷的任何NFS文件系统。这样做会强制Linux操作系统为该文件系统分配独立的设备句柄。

Linux Direct NFS和Oracle RAC

使用DNFS对于Linux操作系统上的Oracle RAC具有特殊的性能优势、因为Linux没有强制执行直接I/O的方法、而RAC需要执行直接I/O才能在节点间保持一致。作为临时解决策、Linux需要使用 actimeo=0 挂载选项、此选项会导致操作系统缓存中的文件数据立即过期。而此选项又会强制Linux NFS客户端不断重新读取属性数据、从而会损害延迟并增加存储控制器上的负载。

启用DNFS会绕过主机NFS客户端并避免这种损坏。多家客户报告说、在启用DNFS时、RAC集群的性能显著提高、ONTAP负载显著降低(尤其是与其他IOPS相关的负载)。

Linux Direct NFS和oranfstab文件

如果在Linux上使用DNFS和多路径选项、则必须使用多个子网。在其他操作系统上、可以使用建立多个DNFS通道 LOCAL 和 DONTROUTE 用于在一个子网上配置多个DNFS通道的选项。但是、这在Linux上不能正常工作、可能会导致意外的性能问题。在Linux中、用于DNFS流量的每个NIC都必须位于不同的子网上。

I/O计划程序

Linux内核允许对块设备的I/O计划方式进行低级控制。各种Linux发行版的默认值差别很大。测试表明、截止日期通常会获得最佳结果、但NOOP有时会稍好一些。性能差异极小、但如果需要从数据库配置中提取尽可能高的性能、请同时测试这两个选项。CFQ是许多配置中的默认设置、它已证明数据库工作负载存在严重的性能问题。

有关配置I/O计划程序的说明、请参见相关的Linux供应商文档。

多路径

某些客户在网络中断期间遇到崩溃、因为多路径守护进程未在其系统上运行。在最新版本的Linux上、操作系统和多路径守护进程的安装过程可能会使这些操作系统容易受到此问题的影响。软件包安装正确、但未配置为在重新启动后自动启动。

例如、RHEL5.5上的多路径守护进程的默认设置可能如下所示：

```
[root@host1 iscsi]# chkconfig --list | grep multipath
multipathd      0:off   1:off   2:off   3:off   4:off   5:off   6:off
```

可使用以下命令更正此问题：

```
[root@host1 iscsi]# chkconfig multipathd on
[root@host1 iscsi]# chkconfig --list | grep multipath
multipathd      0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

ASM镜像

ASM 镜像可能需要更改 Linux 多路径设置，以使 ASM 能够识别问题并切换到备用故障组。ONTAP 上的大多数 ASM 配置都使用外部冗余，这意味着数据保护由外部阵列提供，并且 ASM 不会镜像数据。某些站点使用正常冗余的 ASM 来提供双向镜像，通常在不同站点之间进行镜像。

中显示的Linux设置 "NetApp主机实用程序文档" 包括导致I/O无限期排队的多路径参数这意味着、没有活动路径的LUN设备上的I/O会根据需要等待I/O完成。这通常是可取的、因为Linux主机会根据需要等待很长时间、以便SAN路径更改完成、FC交换机重新启动或存储系统完成故障转移。

这种无限制排队行为会导致ASM镜像出现问题、因为ASM必须收到I/O故障、才能在备用LUN上重试I/O。

在Linux中设置以下参数 `multipath.conf` 用于ASM镜像的ASM LUN文件：

```
polling_interval 5
no_path_retry 24
```

这些设置会为ASM设备创建120秒超时。超时计算为 `polling_interval` * `no_path_retry` 以秒为单位。在某些情况下、可能需要调整确切的值、但120秒的超时时间对于大多数使用来说应该足以满足要求。具体来说、120秒应允许控制器接管或恢复发生、而不会产生会导致故障组脱机的I/O错误。

a较低 `no_path_retry` 值可以缩短ASM切换到备用故障组所需的时间、但这也会增加在控制器接管等维护活动期间发生不必要故障转移的风险。可以通过仔细监控ASM镜像状态来缓解此风险。如果发生不必要的故障转移、并且重新同步执行速度相对较快、则可以快速重新同步镜像。对于追加信息、请参见有关使用的Oracle软件版本的ASM快速镜像重新同步的Oracle文档。

Linux xfs、**ext3**和**ext4**挂载选项

* NetApp建议*使用默认挂载选项。

使用**ASMLi/AFD**的**Oracle**数据库(**ASM**筛选器驱动程序)

特定于使用AFC和ASMlib的Linux操作系统的配置主题

ASMlib块大小

ASMlib是一个可选的ASM管理库和关联实用程序。其主要价值是能够使用可读标签将LUN或基于NFS的文件标记为ASM资源。

最新版本的ASMlib会检测到一个名为逻辑块/物理块指数(Logical Blocks Per Physical Block Exponent、LBPPBE)的LUN参数。直到最近、ONTAP SCSI目标才报告此值。现在、它将返回一个值、指示首选的块大小为4 KB。这不是块大小的定义、但对于使用LBPPBE的任何应用程序来说、这是一个提示、即可以更高效地处理特定大小的I/O。但是、ASMlib会将LBPPBE解释为块大小、并在创建ASM设备时持久标记ASM标头。

此过程可能会在许多方面出现发生原因升级和迁移问题、所有这些问题都是由于无法在同一个ASM磁盘组中混用块大小不同的ASMlib设备。

例如、较早的阵列通常报告LBPPBE值为0、或者根本不报告此值。ASMlib会将此数据块大小解释为512字节。较新的阵列会被解释为块大小为4 KB。不能在同一个ASM磁盘组中混用512字节和4 KB设备。这样做会阻止用户使用两个阵列中的LUN或将ASM用作迁移工具来增加ASM磁盘组的大小。在其他情况下、RMAN可能不允许在块大小为512字节的ASM磁盘组与块大小为4 KB的ASM磁盘组之间复制文件。

首选解决方案是修补ASMlib。Oracle错误ID为13999609、此修补程序位于oracleasm-support-2.1.8-1及更高版本中。此修补程序允许用户设置参数 `ORACLEASM_USE_LOGICAL_BLOCK_SIZE` to `true` 在中 `/etc/sysconfig/oracleasm` 配置文件。这样做会阻止ASMlib使用LBPPBE参数、这意味着新阵列上的LUN现在可识别为512字节块设备。

> 选项不会更改先前由ASMlib标记的LUN上的块大小。例如、如果必须将包含512字节块的ASM磁盘组迁移到报告4 KB块的新存储系统、则可以选择此选项 `ORACLEASM_USE_LOGICAL_BLOCK_SIZE` 必须在新LUN标记为ASMlib之前进行设置。 如果设备已标记为oracleasm，则必须先重新格式化，然后再重新添加新的块大小。首先、使用取消配置设备 `oracleasm deletedisk`，然后使用清除设备的第一个1GB `dd if=/dev/zero of=/dev/mapper/device bs=1048576 count=1024`。最后、如果设备之前已分区、请使用 `kpartx` 命令删除陈旧分区或仅重新启动操作系统。

如果无法修补ASMlib、则可以从配置中删除ASMlib。此更改会造成系统中断、需要取消ASM磁盘的附加服务、并确保 `asm_diskstring` 参数设置正确。但是、这种更改不需要迁移数据。

ASM筛选驱动器**((AFD)**块大小

AAutomatic是一个可选的ASM管理库、它将取代ASMlib。从存储角度来看、它与ASMlib非常相似、但它还包括一些其他功能、例如、可以阻止非Oracle I/O、从而降低用户或应用程序错误可能损坏数据的几率。

设备块大小

与ASMlib一样、ADAD还会读取LUN参数Logical Blocks Per Physical Block Exponent (LBPPBE)、默认情况下会使用物理块大小、而不是逻辑块大小。

如果在现有配置中添加了AAutomatic、而ASM设备已格式化为512字节块设备、则可能会出现问题。AAutomatic驱动程序会将LUN识别为4K设备、如果ASM标签与物理设备不匹配、则会阻止访问。同样、迁移也会受到影响、因为不能在同一个ASM磁盘组中混用512字节和4 KB设备。这样做会阻止用户使用两个阵列中的LUN或将ASM用作迁移工具来增加ASM磁盘组的大小。在其他情况下、RMAN可能不允许在块大小为512字节的ASM磁盘组与块大小为4 KB的ASM磁盘组之间复制文件。

解决方案非常简单—AFAS包含一个参数、用于控制它使用的是逻辑块大小还是物理块大小。此全局参数会影响系统上的所有设备。要强制AfD使用逻辑块大小、请设置 `options oracleafd oracleafd_use_logical_block_size=1` 在中 `/etc/modprobe.d/oracleafd.conf` 文件

多路径传输大小

最近的Linux内核更改会强制实施发送到多路径设备的I/O大小限制、而AFD不会遵守这些限制。然后、I/O将被拒绝、从而导致LUN路径脱机。因此、无法安装Oracle Grid、配置ASM或创建数据库。

解决方案将在多路径.conf文件中为ONTAP LUN手动指定最大传输长度：

```
devices {
            device {
                vendor "NETAPP"
                product "LUN.*"
                max_sectors_kb 4096
        }
    }
```

> 即使当前不存在任何问题、如果使用AWAD来确保未来的Linux升级不会出现意外的发生原因问题、则应设置此参数。

使用Microsoft Windows的Oracle数据库

使用ONTAP在Microsoft Windows上配置Oracle数据库主题。

NFS

Oracle支持将Microsoft Windows与Direct NFS客户端结合使用。此功能提供了一条实现NFS管理优势的途径、其中包括跨环境查看文件、动态调整卷大小以及利用成本较低的IP协议。有关使用DNFS在Microsoft Windows上安装和配置数据库的信息、请参见Oracle官方文档。没有特别的最佳做法。

SAN

为了获得最佳压缩效率、请确保NTFS文件系统使用8K或更大的分配单元。使用4K分配单元(通常为默认分配单元)会对压缩效率产生负面影响。

Oracle数据库与Solaris

特定于Solaris OS的配置主题。

Solaris NFS挂载选项

下表列出了单个实例的Solaris NFS挂载选项。

文件类型	挂载选项
ADr主页	`rw,bg,hard,[vers=3,vers=4.1], roto=tcp, timeo=600,rsize=262144,wsize=262144`
控制文件 数据文件 重做日志	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp, timeo=600,rsize=262144,wsize=262144, nointr,llock,suid`
`ORACLE_HOME`	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp, timeo=600,rsize=262144,wsize=262144, suid`

的使用 `llock` 经验证、通过消除与获取和释放存储系统锁定相关的延迟、可以显著提高客户环境的性能。在配置了大量服务器以挂载相同文件系统且Oracle配置为挂载这些数据库的环境中、请谨慎使用此选项。尽管这种配

置非常少见、但也有少数客户使用。如果某个实例再次意外启动、则可能会发生数据损坏、因为Oracle无法检测到外部服务器上的锁定文件。NFS锁定不会在其他情况下提供保护；与NFS版本3一样、它们仅为建议使用。

因为 `llock` 和 `forcedirectio` 参数是互斥的、这一点很重要 `filesystemio_options=setall` 中存在 `init.ora` 文件 `directio` 已使用。如果没有此参数、则会使用主机操作系统缓冲区缓存、并且可能会对性能产生不利影响。

下表列出了Solaris NFS RAC挂载选项。

文件类型	挂载选项
ADr主页	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,noac`
控制文件 数据文件 重做日志	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,nointr,noac,forcedirectio`
CRS/表决	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,nointr,noac,forcedirectio`
专用 `ORACLE_HOME`	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,suid`
共享 `ORACLE_HOME`	`rw,bg,hard,[vers=3,vers=4.1],proto=tcp,timeo=600,rsize=262144,wsize=262144,nointr,noac,suid`

单实例挂载选项与RAC挂载选项之间的主要区别在于添加了 `noac` 和 `forcedirectio` 挂载选项。这种添加的效果是禁用主机操作系统缓存、从而使RAC集群中的所有实例都能获得一致的数据状态视图。但使用 `init.ora` 参数 `filesystemio_options=setall` 与禁用主机缓存具有相同的效果、但仍需要使用 `noac` 和 `forcedirectio`。

原因 `actimeo=0` 对于共享为必填项 `ORACLE_HOME` 部署是为了提高Oracle密码文件和spfile等文件的一致性。RAC集群中的每个实例都有一个专用 `ORACLE_HOME`，则不需要此参数。

Solaris UFS挂载选项

NetApp强烈建议使用日志记录挂载选项、以便在Solaris主机崩溃或FC连接中断时保持数据完整性。日志记录挂载选项还会保留Snapshot备份的可用性。

Solaris ZFS

必须仔细安装和配置Solaris ZFS，才能提供最佳性能。

mvector

Solaris 11对其处理大型I/O操作的方式进行了更改、这可能会导致SAN存储阵列出现严重的性能问题。NetApp错误报告“Solaris 11 ZFS性能回归”详细介绍了该问题。"解决方案将更改名为的操作系统参数 `zfs_mvector_max_size`。

以root用户身份运行以下命令：

```
[root@host1 ~]# echo "zfs_mvector_max_size/W 0t131072" |mdb -kw
```

如果此更改出现任何意外问题、可以通过以root用户身份运行以下命令轻松反转此更改：

```
[root@host1 ~]# echo "zfs_mvector_max_size/W 0t1048576" |mdb -kw
```

内核

要获得可靠的ZFS性能、需要对Solaris内核进行修补、以防止出现LUN对齐问题。此修复程序是在Solaris 10的修补程序147440-19以及Solaris 11的SRU 10.5中引入的。请仅将Solaris 10及更高版本与ZFS结合使用。

LUN配置

要配置LUN、请完成以下步骤：

1. 创建类型为的LUN `solaris`。
2. 安装指定的相应Host Utility Kit (HUK) "NetApp 互操作性表工具（IMT）"。
3. 完全按照所述执行HUK中的说明。基本步骤概述如下、但请参见 "最新文档" 正确的操作步骤。
   a. 运行 `host_config` 实用程序以更新 `sd.conf/sdd.conf` 文件这样可以使SCSI驱动器正确发现ONTAP LUN。
   b. 按照提供的说明进行操作 `host_config` 用于启用多路径输入/输出(MPIO)的实用程序。
   c. 重新启动。要在整个系统中识别任何更改、必须执行此步骤。
4. 对LUN进行分区并验证它们是否已正确对齐。有关如何直接测试和确认对齐的说明，请参阅“附录B：WAFL对齐验证”。

zpools

只能在中的步骤之后创建zpool "LUN配置" 执行。如果操作步骤未正确执行、则可能会因I/O对齐而导致性能严重下降。要在ONTAP上获得最佳性能、需要将I/O与驱动器上的4K边界对齐。在zpool上创建的文件系统使用有效块大小、该大小通过名为的参数进行控制 `ashift`，可通过运行命令来查看 `zdb -C`。

的值 `ashift` 默认为9、表示2^9或512字节。为了获得最佳性能、 `ashift` 值必须为12 (2^12=4k)。此值在创建zpool时设置、并且无法更改、这意味着具有的zpool中的数据 `ashift` 应通过将数据复制到新创建的zpool来迁移12以外的文件。

创建zpool后、请验证的值 `ashift` 然后继续。如果此值不是12、则表示未正确发现LUN。销毁zpool、确认相关Host Utilities文档中显示的所有步骤均已正确执行、然后重新创建zpool。

zpool和Solaris LDom

Solaris LDOM还要求确保I/O对齐正确。虽然LUN可能会作为4K设备正确地被发现、但LDOM上的虚拟vdsk设备不会继承I/O域中的配置。基于该LUN的vdsk默认为512字节的块。

需要一个额外的配置文件。首先、必须为各个LLOM修补Oracle错误27824910、以启用其他配置选项。此修补

程序已移植到所有当前使用的Solaris版本中。对LDOM进行修补后、即可按如下所示配置正确对齐的新LUN：

1. 确定要在新zpool中使用的一个或多个LUN。在此示例中、它是C2D1设备。

```
[root@LDOM1 ~]# echo | format
Searching for disks...done
AVAILABLE DISK SELECTIONS:
  0. c2d0 <Unknown-Unknown-0001-100.00GB>
     /virtual-devices@100/channel-devices@200/disk@0
  1. c2d1 <SUN-ZFS Storage 7330-1.0 cyl 1623 alt 2 hd 254 sec 254>
     /virtual-devices@100/channel-devices@200/disk@1
```

2. 检索要用于ZFS池的设备的VDC实例：

```
[root@LDOM1 ~]#  cat /etc/path_to_inst
#
# Caution! This file contains critical kernel state
#
"/fcoe" 0 "fcoe"
"/iscsi" 0 "iscsi"
"/pseudo" 0 "pseudo"
"/scsi_vhci" 0 "scsi_vhci"
"/options" 0 "options"
"/virtual-devices@100" 0 "vnex"
"/virtual-devices@100/channel-devices@200" 0 "cnex"
"/virtual-devices@100/channel-devices@200/disk@0" 0 "vdc"
"/virtual-devices@100/channel-devices@200/pciv-communication@0" 0 "vpci"
"/virtual-devices@100/channel-devices@200/network@0" 0 "vnet"
"/virtual-devices@100/channel-devices@200/network@1" 1 "vnet"
"/virtual-devices@100/channel-devices@200/network@2" 2 "vnet"
"/virtual-devices@100/channel-devices@200/network@3" 3 "vnet"
"/virtual-devices@100/channel-devices@200/disk@1" 1 "vdc" << We want
this one
```

3. 编辑 `/platform/sun4v/kernel/drv/vdc.conf`：

```
block-size-list="1:4096";
```

这意味着为设备实例1分配的块大小为4096。

作为另一个示例、假设需要为vdsk实例1到6配置4K块大小和 `/etc/path_to_inst` 内容如下：

```
"/virtual-devices@100/channel-devices@200/disk@1" 1 "vdc"
"/virtual-devices@100/channel-devices@200/disk@2" 2 "vdc"
"/virtual-devices@100/channel-devices@200/disk@3" 3 "vdc"
"/virtual-devices@100/channel-devices@200/disk@4" 4 "vdc"
"/virtual-devices@100/channel-devices@200/disk@5" 5 "vdc"
"/virtual-devices@100/channel-devices@200/disk@6" 6 "vdc"
```

4. 最终版本 `vdc.conf` 文件应包含以下内容：

```
block-size-list="1:8192","2:8192","3:8192","4:8192","5:8192","6:8192";
```

小心
配置vdc.conf并创建vdsk后、必须重新启动LLOM。这一步是不可避免的。块大小更改仅在重新启动后生效。继续进行zpool配置、并确保将ashift正确设置为12、如上所述。

ZFS意图日志(ZIL)

通常，没有理由在其他设备上查找ZFS意图日志(ZIL)。日志可以与主池共享空间。单独的ZIL主要用于使用在现代存储阵列中缺少写入缓存功能的物理驱动器。

对数偏差

设置 `logbias` 用于托管Oracle数据的ZFS文件系统上的参数。

```
zfs set logbias=throughput <filesystem>
```

使用此参数可降低整体写入级别。在默认设置下、写入的数据会先提交到ZIL、然后再提交到主存储池。此方法适用于使用普通驱动器配置的配置、该配置包括基于SSD的ZIL设备和用于主存储池的旋转介质。这是因为它允许在可用延迟最低的介质上的单个I/O事务中进行提交。

如果使用的是具有自身缓存功能的现代存储阵列、则通常不需要使用此方法。在极少数情况下、可能需要将具有单个事务的写入提交到日志中、例如由高度集中且对延迟敏感的随机写入组成的工作负载。写入放大会产生一定的后果、因为记录的数据最终会写入主存储池、从而导致写入活动增加一倍。

直接I/O

许多应用程序(包括Oracle产品)都可以通过启用直接I/O来绕过主机缓冲区缓存此策略无法按预期用于ZFS文件系统。尽管会绕过主机缓冲区缓存，但ZFS本身仍会继续缓存数据。在使用FIO或SIO等工具执行性能测试时、此操作可能会导致误导性的结果、因为很难预测I/O是到达存储系统还是在操作系统中本地缓存。此操作还会使使用此类综合测试来比较ZFS与其他文件系统的性能变得非常困难。实际上、在实际用户工作负载下、文件系统性能几乎没有差别。

多个zpool

必须在zpool级别对基于ZFS的数据执行基于Snapshot的备份、还原、克隆和归档、并且通常需要多个zpool。zpool类似于LVM磁盘组、应使用相同的规则进行配置。例如、数据库的布局可能最好是将数据文件驻留在上

`zpool1` 以及上的归档日志、控制文件和重做日志 `zpool2`。此方法允许使用标准热备份、其中数据库将置于热备份模式、然后是的快照 `zpool1`。然后、数据库将从热备份模式中删除、并强制执行日志归档和的快照 `zpool2` 已创建。还原操作需要卸载zfs文件系统并使zpool完全脱机、然后执行SnapRestore还原操作。然后、可以将zpool重新联机并恢复数据库。

filesystemio_options

Oracle参数 `filesystemio_options` 与ZFS的工作方式不同。条件 `setall` 或 `directio` 使用时、写入操作是同步的、并会绕过操作系统缓冲区缓存、但读取操作会由ZFS进行缓冲。此操作会导致性能分析出现困难、因为I/O有时会被ZFS缓存截获并提供服务、从而使存储延迟和总I/O比看起来要小。

网络配置：

Oracle数据库的逻辑接口设计

Oracle数据库需要访问存储。逻辑接口(Logical Interface、Logical Interface、Logical Interface)是将Storage Virtual Machine (SVM)连接到网络并进而连接到数据库的网络管道。要确保每个数据库工作负载都有足够的带宽、并且故障转移不会导致存储服务丢失、需要正确的LIF设计。

本节概述了LIF的主要设计原则。有关更全面的文档、请参见 "ONTAP网络管理文档"。与数据库架构的其他方面一样、Storage Virtual Machine (SVM、在CLI中称为Vserver)和逻辑接口(LIF)设计的最佳选项在很大程度上取决于扩展要求和业务需求。

在制定LIF策略时、请考虑以下主要主题：

- *性能。*网络带宽是否足够?
- *故障恢复能力。*设计中是否存在单点故障?
- *易管理性。*网络能否无干扰地扩展?

这些主题适用于从主机到交换机再到存储系统的端到端解决方案。

LIF类型

LIF类型有多种。 "有关LIF类型的ONTAP文档" 请提供有关此主题的更完整信息、但从功能角度来看、可以将这些生命周期表分为以下几组：

- *集群和节点管理Lifs.*用于管理存储集群的Lifs。
- * SVM管理LIF.*允许通过REST API或ONTAPI (也称为ZAPI)访问SVM的接口、用于执行创建快照或调整卷大小等功能。SnapManager for Oracle (SMO)等产品必须能够访问SVM管理LIF。
- 数据**Lifs.** FC、iSCSI、NVMe/FC、NVMe/TCP、NFS接口 或SMB/CCIFS数据。

> 用于NFS流量的数据LIF也可通过从更改防火墙策略来进行管理 `data` to `mgmt` 或其他允许HTTP、HTTPS或SSH的策略。此更改可以避免配置每个主机以访问NFS数据LIF和单独的管理LIF、从而简化网络配置。无法为iSCSI和管理流量配置接口、尽管两者都使用IP协议。在iSCSI环境中、需要使用单独的管理LIF。

SAN LIF设计

SAN环境中的LIF设计相对简单、原因之一是：多路径。所有现代SAN实施都允许客户端通过多个独立的网络路径访问数据、并选择最佳访问路径。因此、与LIF设计相关的性能更易于解决、因为SAN客户端会自动在最佳可用路径之间对I/O进行负载平衡。

如果某个路径不可用、则客户端会自动选择其他路径。由此带来的设计精简性使SAN的生命周期通常更易于管理。这并不意味着SAN环境始终可以更轻松地进行管理、因为SAN存储的许多其他方面都比NFS复杂得多。这只是意味着SAN LIF的设计更简单。

性能

在SAN环境中、LIF性能最重要的考虑因素是带宽。例如、每个节点具有两个16 Gb FC端口的双节点ONTAP AFF集群允许每个节点之间最多32 Gb的带宽。

故障恢复能力

SAN AFF不会在SAN存储系统上进行故障转移。如果SAN LIF因控制器故障转移而失败、则客户端的多路径软件会检测到路径丢失、并将I/O重定向到其他LIF。对于ASA存储系统、将在短暂延迟后对lifs进行故障转移、但这不会中断IO、因为其他控制器上已存在活动路径。执行故障转移过程是为了恢复所有定义端口上的主机访问。

易管理性

在NFS环境中、LIF迁移是一项更为常见的任务、因为LIF迁移通常与在集群中重新定位卷相关联。在HA对中重新定位卷后、无需在SAN环境中迁移LIF。这是因为、在卷移动完成后、ONTAP会向SAN发送有关路径更改的通知、并且SAN客户端会自动重新优化。使用SAN迁移LIF主要与重大物理硬件更改相关。例如、如果需要无中断升级控制器、则会将SAN LIF迁移到新硬件。如果发现FC端口出现故障、则可以将LIF迁移到未使用的端口。

设计建议

NetApp提出以下建议：

- 请勿创建超出所需数量的路径。路径数量过多会使整体管理变得更加复杂、并且某些主机上的路径故障转移可能会出现发生原因问题。此外、对于SAN启动等配置、某些主机存在意外的路径限制。
- 很少有配置需要一个LUN具有四个以上的路径。如果有两个以上的节点向LUN公布路径、则其价值会受到限制、因为如果拥有LUN的节点及其HA配对节点发生故障、则托管LUN的聚合将无法访问。在这种情况下、在主HA对以外的节点上创建路径毫无用处。
- 虽然可以通过选择要包含在FC分区中的端口来管理可见LUN路径的数量、但通常在FC分区中包含所有潜在目标点并在ONTAP级别控制LUN可见性会更容易。
- 在ONTAP 8.3及更高版本中、默认使用选择性LUN映射(SLM)功能。通过SLM、任何新的LUN都会自动从拥有底层聚合的节点以及该节点的HA配对节点公布。这种安排无需创建端口集或配置分区来限制端口可访问性。为了获得最佳性能和故障恢复能力、每个LUN可在所需的最少节点上使用。
  *如果必须将LUN迁移到两个控制器之外、则可以使用添加其他节点 `lun mapping add-reporting-nodes` 命令、以便在新节点上公布LUN。这样会为LUN创建更多SAN路径以进行LUN迁移。但是、主机必须执行发现操作才能使用新路径。
- 不要过分关注间接流量。在I/O密集型环境中、最好避免间接流量、因为在这种环境中、每微秒的延迟都至关重要、但对于典型工作负载、可见的性能影响可以忽略不计。

NFS LIF设计

与SAN协议不同、NFS定义多个数据路径的能力有限。NFSv4的并行NFS (pNFS)扩展解决了这一限制、但由于

以太网速度已达到100 GB甚至超过100 GB、因此添加额外路径很少有价值。

性能和故障恢复能力

虽然衡量SAN LIF性能主要是计算所有主路径的总带宽、但要确定NFS LIF性能、需要更深入地了解确切的网络配置。例如、可以将两个10 Gb端口配置为原始物理端口、也可以将其配置为链路聚合控制协议(Link Aggregation Control Protocol、LACP)接口组。如果将其配置为接口组、则可以使用多个负载平衡策略、这些策略的工作方式会有所不同、具体取决于流量是交换流量还是路由流量。最后、Oracle Direct NFS (DNFS)提供了目前在任何操作系统NFS客户端中都不存在的负载平衡配置。

与SAN协议不同、NFS文件系统要求在协议层具有故障恢复能力。例如、LUN始终配置为启用多路径、这意味着存储系统可以使用多个冗余通道、每个通道都使用FC协议。另一方面、NFS文件系统取决于单个TCP/IP通道的可用性、该通道只能在物理层进行保护。这种安排就是为什么存在端口故障转移和LACP端口聚合等选项的原因。

在NFS环境中、性能和故障恢复能力均在网络协议层提供。因此，这两个主题是相互交织的，必须一起讨论。

将**LIP**绑定到端口组

要将LIF绑定到端口组、请将LIF IP地址与一组物理端口相关联。将物理端口聚合在一起的主要方法是LACP。LACP的容错功能相当简单；LACP组中的每个端口都会受到监控、并在发生故障时从端口组中删除。但是、对于LACP在性能方面的工作原理、存在许多误解：

- LACP不要求交换机上的配置与端点匹配。例如、可以为ONTAP配置基于IP的负载平衡、而交换机可以使用基于MAC的负载平衡。
- 使用LACP连接的每个端点都可以独立选择数据包传输端口、但不能选择用于接收的端口。这意味着、从ONTAP到特定目标的流量会绑定到特定端口、而返回流量可能会到达其他接口。但是、这不会造成发生原因问题。
- LACP不会始终均匀分布流量。在具有许多NFS客户端的大型环境中、结果通常甚至会使用LACP聚合中的所有端口。但是、环境中的任何一个NFS文件系统都仅限于一个端口的带宽、而不是整个聚合的带宽。
- 尽管ONTAP上提供了robin-robin LACP策略、但这些策略不会处理从交换机到主机的连接。例如、如果配置中的一个主机上有一个四端口LACP中继、而ONTAP上有一个四端口LACP中继、则仍然只能使用一个端口读取文件系统。虽然ONTAP可以通过所有四个端口传输数据、但目前尚无可通过所有四个端口从交换机发送到主机的交换机技术。仅使用一个。

在包含许多数据库主机的大型环境中、最常见的方法是使用IP负载平衡构建一个包含适当数量10 Gb (或更快)接口的LACP聚合。通过这种方法、只要存在足够多的客户端、ONTAP就可以均匀地使用所有端口。如果配置中的客户端较少、则负载平衡会中断、因为LACP中继不会动态重新分配负载。

建立连接后、特定方向的流量仅会放置在一个端口上。例如、对通过四端口LACP中继连接的NFS文件系统执行完整表扫描的数据库仅通过一个网络接口卡(Network Interface Card、NIC)读取数据。如果在此类环境中只有三个数据库服务器、则这三个服务器都可能从同一端口读取数据、而其他三个端口则处于空闲状态。

将**Lifs**绑定到物理端口

将LIF绑定到物理端口可以更精细地控制网络配置、因为ONTAP系统上的给定IP地址一次只与一个网络端口相关联。然后、可通过配置故障转移组和故障转移策略来实现故障恢复能力。

故障转移策略和故障转移组

故障转移策略和故障转移组控制了在网络中断期间的故障转移。配置选项已随ONTAP的不同版本而发生更改。请参见 "有关故障转移组和策略的ONTAP网络管理文档" 有关要部署的ONTAP版本的具体详细信息、请参见。

ONTAP 8.3及更高版本支持基于广播域管理LIF故障转移。因此、管理员可以定义可访问给定子网的所有端口、并允许ONTAP选择适当的故障转移LIF。某些客户可以使用这种方法、但由于缺乏可预测性、在高速存储网络环境中这种方法存在一些限制。例如、一个环境可以包括用于例行文件系统访问的1 Gb端口和用于数据文件I/O的10 Gb端口如果两种类型的端口都位于同一广播域中、则LIF故障转移可能会导致数据文件I/O从10 Gb端口移动到1 Gb端口。

概括地说、请考虑以下做法：

1. 将故障转移组配置为用户定义的组。
2. 使用存储故障转移(SFR)配对控制器上的端口填充故障转移组、以便在存储故障转移期间、这些LUN跟随聚合。这样可以避免产生间接流量。
3. 使用性能特征与原始LIF匹配的故障转移端口。例如、单个10 Gb物理端口上的LIF应包含一个具有单个10 Gb端口的故障转移组。一个四端口LACP LIF应故障转移到另一个四端口LACP LIF。这些端口将是广播域中定义的端口的子集。
4. 将故障转移策略设置为仅SFo-Partner。这样可以确保LIF在故障转移期间跟随聚合。

自动还原

设置 `auto-revert` 参数。大多数客户倾向于将此参数设置为 `true` 以使LIF还原到其主端口。但是、在某些情况下、客户会将此值设置为`false `so、以便在将LIF返回到其主端口之前可以调查意外故障转移。

LIF与卷的比率

一个常见的误解是、卷和NFS Sifs之间必须有1：1的关系。虽然要在集群中的任何位置移动卷而不创建额外的互连流量、都需要使用此配置、但这绝对不是一项要求。必须考虑集群间流量、但仅存在集群间流量并不会造成问题。为ONTAP创建的许多已发布基准主要包括间接I/O

例如、如果某个数据库项目包含的性能关键型数据库数量相对较少、并且总共只需要40个卷、则可能需要采用卷到LIF的1：1策略、这种安排需要40个IP地址。然后、可以将任何卷与关联的LIF一起移动到集群中的任何位置、流量将始终是直接的、即使是微秒级的延迟、也可以最大限度地减少每个源。

作为一个反例、客户与LI之间的1：1关系可能更易于管理大型托管环境。随着时间的推移、卷可能需要迁移到其他节点、这会对一些间接流量进行发生原因。但是、除非互连交换机上的网络端口饱和、否则不会检测到性能影响。如果存在问题、可以在其他节点上建立新的LIF、并可在下一个维护窗口更新主机、以便从配置中删除间接流量。

Oracle数据库的**TCP/IP**和以太网配置

许多基于ONTAP的Oracle客户使用以太网、即NFS、iSCSI、NVMe/TCP的网络协议、尤其是云。

主机操作系统设置

大多数应用程序供应商文档都包含特定的TCP和以太网设置、旨在确保应用程序以最佳状态运行。这些相同的设置通常足以提供基于IP的最佳存储性能。

以太网流量控制

此技术允许客户端请求发送方暂时停止数据传输。这通常是因为接收方无法足够快速地处理传入数据。一次、请求发送方停止传输比让接收方丢弃数据包造成的中断要少、因为缓冲区已满。如今、操作系统中使用的TCP堆栈

已不再是这种情况。事实上、流量控制造成的问题比它解决的问题多。

近年来、以太网流量控制导致的性能问题不断增加。这是因为以太网流量控制在物理层运行。如果网络配置允许任何主机操作系统向存储系统发送以太网流量控制请求、则会导致所有已连接客户端的I/O暂停。由于单个存储控制器为越来越多的客户端提供服务、因此其中一个或多个客户端发送流量控制请求的可能性会增加。在广泛的操作系统虚拟化过程中、客户站点经常会出现此问题。

NetApp系统上的NIC不应接收流量控制请求。根据网络交换机制造商的不同、实现此结果的方法也会有所不同。在大多数情况下、可以将以太网交换机上的流量控制设置为 `receive desired` 或 `receive on`，表示流量控制请求不会转发到存储控制器。在其他情况下、存储控制器上的网络连接可能不允许禁用流量控制。在这些情况下、必须将客户端配置为从不发送流量控制请求、方法是更改主机服务器本身的NIC配置或主机服务器所连接的交换机端口。

* NetApp建议*确保NetApp存储控制器不接收以太网流量控制数据包。这通常可以通过设置控制器所连接的交换机端口来实现、但某些交换机硬件存在一些限制、可能需要在客户端进行更改。

MTU大小

事实证明、使用巨型帧可以减少CPU和网络开销、从而在一定程度上提高1 Gb网络的性能、但其优势通常并不明显。

* NetApp建议*尽可能实施巨型帧、以实现任何潜在的性能优势并使解决方案适应未来需求。

在10 Gb网络中使用巨型帧几乎是强制性要求。这是因为大多数10 Gb实施在达到10 Gb标记之前都会达到每秒数据包数限制、而不会出现巨型帧。使用巨型帧可以提高TCP/IP处理的效率、因为它允许操作系统、服务器、NIC和存储系统处理的数据包数量较少、但数量较大。不同NIC的性能提升各不相同、但性能提升幅度很大。

对于巨型帧实施、人们普遍认为所有连接的设备都必须支持巨型帧、并且MTU大小必须端到端匹配、但这种看法并不正确相反、在建立连接时、这两个网络端点会协商双方可接受的最高帧大小。在典型环境中、网络交换机的MTU大小设置为9216、NetApp控制器设置为9000、客户端设置为9000和1514的混合。可以支持9000 MTU的客户端可以使用巨型帧、而只支持1514的客户端可以协商较低的值。

在完全交换的环境中、这种安排的问题很少见。但是、在路由环境中请注意、不会强制任何中间路由器对巨型帧进行分段。

- NetApp建议*配置以下内容：
- 巨型帧是需要的、但对于1 Gb以太网(GbE)则不需要巨型帧。
- 要在10GbE和更快的速度下实现最高性能、需要巨型帧。

TCP参数

通常会有三种设置配置不当：TCP时间戳、选择性确认(SACK)和TCP窗口缩放。Internet上的许多过时文档建议禁用其中一个或多个参数以提高性能。这一建议在多年前具有一定的价值、那时CPU功能要低得多、尽可能减少TCP处理开销会有好处。

但是、在现代操作系统中、禁用任何这些TCP功能通常不会带来明显的优势、同时还可能会损害性能。在虚拟化网络环境中、性能可能会受到损害、因为要高效处理数据包丢失和网络质量变化、需要使用这些功能。

*TCP NetApp建议*在主机上启用TCP时间戳、SACK和TCP窗口缩放，在任何当前操作系统中，所有这三个参数都应默认为打开。

适用于**Oracle**数据库的**FC**配置

为Oracle数据库配置FC SAN主要是遵循日常SAN最佳实践。

这包括典型的规划措施、例如、确保主机和存储系统之间的SAN具有足够的带宽、检查所有必需设备之间是否存在所有SAN路径、使用FC交换机供应商所需的FC端口设置、避免ISL争用、 并使用适当的SAN网络结构监控。

分区

一个FC分区不应包含多个启动程序。这种安排最初可能看起来有效、但启动程序之间的串扰最终会影响性能和稳定性。

虽然在极少数情况下、不同供应商的FC目标端口的行为会导致问题、但多目标区域通常被视为安全区域。例如、避免将NetApp和非NetApp存储阵列的目标端口都包含在同一分区中。此外、将NetApp存储系统和磁带设备置于同一分区更容易出现发生原因问题。

Oracle数据库和直连**ONTAP**连接

存储管理员有时倾向于通过从配置中删除网络交换机来简化其基础架构。在某些情况下、可以支持此功能。

iSCSI和**NVMe/TCP**

使用iSCSI或NVMe/TCP的主机可以直接连接到存储系统并正常运行。原因是路径问题。直接连接到两个不同的存储控制器会产生两条独立的数据流路径。丢失路径、端口或控制器不会阻止使用另一个路径。

NFS

可以使用直连NFS存储、但有一个重大限制—如果没有大量的脚本编写工作、故障转移将无法正常工作、这是客户的责任。

直连NFS存储的无中断故障转移之所以复杂、是因为本地操作系统上会发生路由。例如、假设主机的IP地址为192.168.1.1/24、并且直接连接到IP地址为192.168.1.50/24的ONTAP控制器。在故障转移期间、该192.168.1.50地址可以故障转移到另一个控制器、并且该地址可供主机使用、但主机如何检测到它的存在？原来的192.168.1.1地址仍然位于不再连接到操作系统的主机NIC上。发往192.168.1.50的流量将继续发送到无法运行的网络端口。

第二个操作系统NIC可配置为19 2.168.1.2、并且能够与故障转移的192.168.1.50地址通信、但本地路由表默认使用一个*且仅一个*地址与192.168.1.0/24子网通信。sysadmin可以创建一个脚本框架、用于检测失败的网络连接并更改本地路由表或启动和关闭接口。确切的操作步骤取决于所使用的操作系统。

在实践中、NetApp客户确实使用直连NFS、但通常仅适用于故障转移期间IO暂停的工作负载。使用硬挂载时、暂停期间不应出现任何IO错误。在服务还原之前、IO应挂起、可以通过故障恢复或手动干预在主机上的NIC之间移动IP地址。

FC直连

不能使用FC协议将主机直接连接到ONTAP存储系统。原因是使用了NPIV。用于向FC网络标识ONTAP FC端口的WWN使用一种称为NPIV的虚拟化类型。连接到ONTAP系统的任何设备都必须能够识别NPIV WWN。目前没有HBA供应商提供可安装在能够支持NPIV目标的主机中的HBA。

存储配置

FC SAN

Oracle数据库I/O的LUN对齐

LUN对齐是指针对底层文件系统布局优化I/O。

在ONTAP系统上、存储以4 KB为单位进行组织。一个数据库或文件系统的8 KB块应正好映射到两个4 KB块。如果LUN配置错误使对齐在任一方向上移动1 KB、则每个8 KB块将位于三个不同的4 KB存储块上、而不是两个。这种安排会增加发生原因延迟、并在存储系统中执行发生原因额外的I/O。

对齐也会影响LVM架构。如果在整个驱动器设备上定义了逻辑卷组中的物理卷(不创建分区)、则LUN上的第一个4 KB块与存储系统上的第一个4 KB块对齐。这是正确的对齐方式。分区会出现问题、因为它们会移动操作系统使用LUN的起始位置。只要偏移量以4 KB的整数单位移动、LUN就会对齐。

在Linux环境中、在整个驱动器设备上构建逻辑卷组。如果需要分区、请运行以检查对齐情况 `fdisk -u` 并验证每个分区的起始位置是否为八的倍数。这意味着分区从八个512字节扇区的倍数开始、即4 KB。

另请参见一节中有关数据压缩块对齐的讨论 "效率"。与8 KB压缩块边界对齐的任何布局也与4 KB边界对齐。

未对齐警告

数据库重做/事务日志记录通常会生成未对齐的I/O、此I/O可能会导致发生原因发出有关ONTAP上LUN错位的警告、从而使人产生误解。

日志记录会使用不同大小的写入顺序写入日志文件。不与4 KB边界对齐的日志写入操作通常不会出现发生原因性能问题、因为下一个日志写入操作会完成块。因此、ONTAP几乎能够将所有写入作为完整的4 KB块进行处理、即使某些4 KB块中的数据是在两个单独的操作中写入的。

使用等实用程序验证对齐情况 `sio` 或 `dd` 可以按定义的块大小生成I/O。可以使用查看存储系统上的I/O对齐统计信息 `stats` 命令：请参见 "WAFL对齐验证" 有关详细信息 …

Solaris环境中的对齐更为复杂。请参见 "ONTAP SAN 主机配置" 有关详细信息 …

小心
在Solaris x86环境中，请格外注意正确对齐，因为大多数配置都有多个分区层。Solaris x86分区片通常位于标准主引导记录分区表之上。

Oracle数据库LUN大小调整和LUN计数

要获得Oracle数据库的最佳性能和易管理性、选择最佳LUN大小和要使用的LUN数量至关重要。

LUN是ONTAP上的一个虚拟化对象、位于托管聚合中的所有驱动器上。因此、LUN的性能不受其大小的影响、因为无论选择何种大小、LUN都会利用聚合的全部性能潜能。

为了方便起见、客户可能希望使用特定大小的LUN。例如、如果数据库是基于LVM或Oracle ASM磁盘组构建的、其中每个磁盘组包含两个1 TB的LUN、则该磁盘组必须以1 TB为增量进行增长。最好使用八个500 GB的LUN来构建磁盘组、以便可以以较小的增量来增加磁盘组。

建议不要建立通用标准LUN大小、因为这样做会使易管理性复杂化。例如、如果数据库或数据存储库的大小介于1 TB到2 TB之间、则100 GB的标准LUN大小可能效果良好、但20 TB的数据库或数据存储库需要200个LUN。这意味着、服务器重新启动时间会更长、需要在各种用户界面中管理更多对象、SnapCenter等产品必须对许多对象执行发现。使用更少、更大的LUN可避免此类问题。

- LUN计数比LUN大小更重要。
- LUN大小主要由LUN计数要求控制。
- 避免创建超出所需数量的LUN。

LUN计数

与LUN大小不同、LUN计数会影响性能。应用程序性能通常取决于通过SCSI层执行并行I/O的能力。因此、两个LUN的性能优于一个LUN。使用Veritas VLVM、Linux LVM2或Oracle ASM等LVM是提高并行性的最简单方法。

虽然对随机I/O非常繁重的100% SSD环境进行的测试表明、LUN数量最多可增加到64个、但一般来说、NetApp客户从LUN数量增加到16个以上所获得的优势微乎其微。

- NetApp建议*：

通常、四到十六个LUN足以满足任何给定数据库工作负载的I/O需求。由于主机SCSI实施的限制、如果LUN数量少于四个、则可能会造成性能限制。

Oracle数据库LUN放置

数据库LUN在ONTAP卷中的最佳放置位置主要取决于各种ONTAP功能的使用方式。

Volumes

首次接触ONTAP的客户通常会感到困惑的一点是、FlexVol的使用、通常简称为"卷"。

卷不是LUN。这些术语与许多其他供应商产品(包括云提供商)同义。ONTAP卷只是管理容器。它们不会自行提供数据、也不会占用空间。它们是文件或LUN的容器、旨在提高和简化易管理性、尤其是大规模管理。

卷和LUN

相关LUN通常位于同一个卷中。例如、需要10个LUN的数据库通常会将所有10个LUN放置在同一个卷上。

- 采用1：1的LUN与卷比率(即每个卷一个LUN)是一种*不*正式的最佳实践。
- 而是应将卷视为工作负载或数据集的容器。每个卷可能有一个LUN、也可能有多个LUN。正确的问题解答取决于易管理性要求。
- 将LUN分散在不必要数量的卷上可能会导致额外开销和操作计划问题、例如快照操作、UI中显示的对象数量过多、并导致在达到LUN限制之前达到平台卷限制。

卷、**LUN**和快照

Snapshot策略和计划放置在卷上、而不是LUN上。如果包含10个LUN的数据集位于同一个卷中、则这些LUN只需要一个Snapshot策略。

此外、在一个卷中将给定数据集的所有相关LUN同位可实现原子快照操作。例如、如果基础LUN都位于一个卷上、则驻留在10个LUN上的数据库或包含10个不同操作系统的基于VMware的应用程序环境可以作为一个一致的对象进行保护。如果将它们放置在不同的卷上、则快照可能会(也可能不会)完全同步、即使是同时计划的也是如此。

在某些情况下、由于恢复要求、可能需要将一组相关LUN拆分为两个不同的卷。例如、一个数据库可能有四个用于数据文件的LUN和两个用于日志的LUN。在这种情况下、最好使用包含4个LUN的数据文件卷和包含2个LUN的日志卷。原因是独立可恢复性。例如、可以有选择地将数据文件卷还原到先前的状态、这意味着所有四个LUN都将还原到快照的状态、而日志卷及其关键数据不会受到影响。

卷、**LUN**和**SnapMirror**

SnapMirror策略和操作与快照操作一样、在卷上执行、而不是在LUN上执行。

通过在一个卷中将相关LUN同位、您可以创建一个SnapMirror关系、并通过一次更新来更新所有包含的数据。与快照一样、更新也是一项原子操作。保证SnapMirror目标具有源LUN的单个时间点副本。如果LUN分布在多个卷上、则这些副本之间可能一致、也可能不一致。

卷、**LUN**和**QoS**

虽然可以有选择地将QoS应用于各个LUN、但在卷级别设置QoS通常更容易。例如、给定ESX服务器中子系统使用的所有LUN都可以放置在一个卷上、然后应用ONTAP自适应QoS策略。因此、会产生一个可自行扩展的每TB IOPS限制、用于对所有LUN执行适用场景操作。

同样、如果数据库需要10万次IOPS并占用10个LUN、则在单个卷上设置一个10万次IOPS限制比在每个LUN上设置10个单独的10万次IOPS限制更容易。

多卷布局

在某些情况下、在多个卷之间分布LUN可能会很有用。主要原因是控制器条带化。例如、一个HA存储系统可能托管一个数据库、其中需要每个控制器的全部处理和缓存潜力。在这种情况下、典型的设计是、将一半的LUN放置在控制器1上的一个卷中、而将另一半LUN放置在控制器2上的一个卷中。

同样、控制器条带化也可用于负载平衡。如果HA系统托管100个数据库、每个数据库包含10个LUN、则可以设计该系统、其中每个数据库在两个控制器中的每个控制器上都接收一个5 LUN卷。这样、在配置更多数据库时、可以保证每个控制器的负载对称。

但是、这些示例均不涉及卷与LUN的比例为1：1。我们的目标仍然是通过在卷中主机代管相关LUN来优化易管理性。

例如、LUN与卷的比例为1：1就意味着容器化、在容器化中、每个LUN可能真正代表一个工作负载、需要逐个进行管理。在这种情况下、1：1的比例可能是最佳的。

Oracle数据库**LUN**大小调整和基于**LVM**的大小调整

当基于SAN的文件系统达到其容量限制时、可通过两种方法增加可用空间：

- 增加LUN的大小

- 将LUN添加到现有卷组并增加包含的逻辑卷

虽然可以选择调整LUN大小来增加容量、但通常最好使用LVM、包括Oracle ASM。存在LVM的一个主要原因是避免调整LUN大小。通过LVM、多个LUN会绑定到一个虚拟存储池中。从该池中划分出来的逻辑卷由LVM管理、并且可以轻松调整大小。另一个优势是、通过在所有可用LUN之间分布给定逻辑卷、可以避免特定驱动器上出现热点。通常、可以通过使用卷管理器将逻辑卷的底层块区重新定位到新LUN来执行透明迁移。

使用**Oracle**数据库进行**LVM**条带化

LVM条带化是指在多个LUN之间分布数据。结果是、许多数据库的性能显著提高。

在闪存驱动器时代之前、条带化用于帮助克服旋转驱动器的性能限制。例如、如果操作系统需要执行1 MB的读取操作、则从单个驱动器读取1 MB的数据将需要大量的驱动器磁头查找和读取、因为1 MB的传输速度较慢。如果在8个LUN上对1 MB的数据进行条带化、则操作系统可以问题描述并行执行8个128 K读取操作、从而减少完成1 MB传输所需的时间。

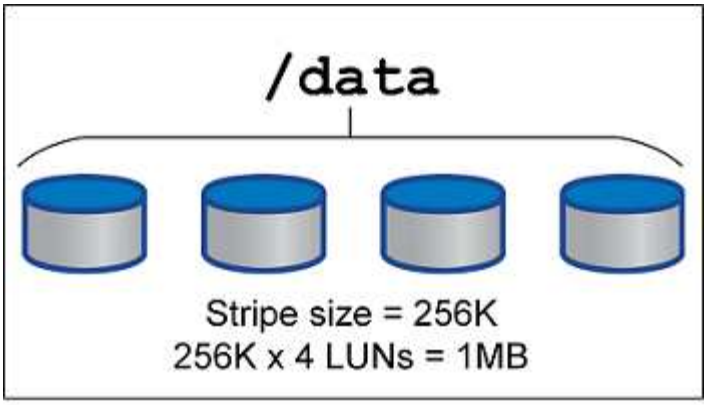
使用旋转驱动器进行条带化更为困难、因为必须事先知道I/O模式。如果条带化未正确调整为真正的I/O模式、则条带化配置可能会损害性能。使用Oracle数据库、尤其是使用全闪存配置时、条带化更易于配置、并且经验证可显著提高性能。

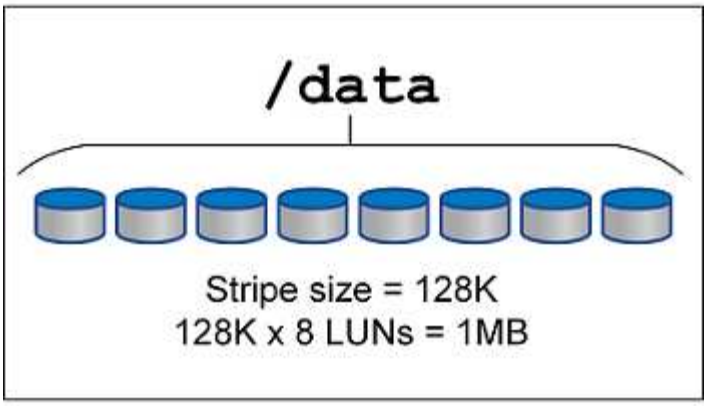
默认情况下、逻辑卷管理器(例如Oracle ASM)会进行条带化、但本机操作系统LVM则不会进行条带化。其中一些会将多个LUN绑定在一起、形成一个串联设备、从而导致数据文件只存在于一个LUN设备上。这会导致热点。其他LVM实施默认使用分布式块区。这与条带化类似、但更粗。卷组中的LUN会被划分为多个大块、称为块区、通常以MB为单位进行测量、然后逻辑卷会分布在这些块区中。结果是、文件的随机I/O应在各个LUN之间分布良好、但顺序I/O操作的效率不如所能达到的高。

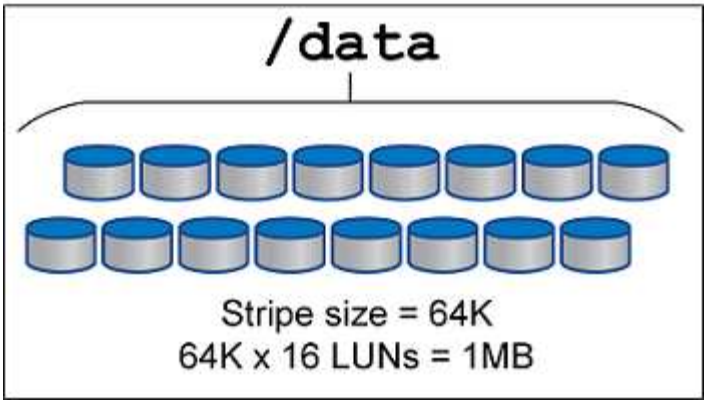
性能密集型应用程序I/O几乎始终为(a)基本块大小单位或(b) 1兆字节。

条带化配置的主要目标是确保单文件I/O可作为一个单元执行、多块I/O (大小应为1 MB)可在条带化卷中的所有LUN之间均匀并行。这意味着条带大小不能小于数据库块大小、条带大小乘以LUN数量应为1 MB。

下图显示了三个可能的条带大小和宽度调整选项。选择LUN数量是为了满足上述性能要求、但在所有情况下、单个条带内的总数据均为1 MB。







NFS

适用于Oracle数据库的NFS配置

NetApp提供企业级NFS存储已超过30年、由于其精简性、随着向基于云的基础架构的推进、其使用量也在不断增长。

NFS协议包含多个版本、但要求各不相同。有关使用ONTAP的完整NFS配置问题描述、请参见 "TR-4067：《基于ONTAP的NFS最佳实践》"。以下各节介绍了一些更关键的要求和常见的用户错误。

NFS版本

NetApp必须支持操作系统NFS客户端。

- 遵循NFSv3标准的操作系统支持NFSv3。
- Oracle DNFS客户端支持NFSv3。

- 遵循NFSv4标准的所有操作系统均支持NFSv4。
- NFSv4.1和NFSv4.2需要特定的操作系统支持。请参见 "NetApp IMT" 支持的操作系统。
- 为NFSv4.1提供Oracle DNFS支持需要Oracle 12.2.0.2或更高版本。

> 。 "NetApp支持表" 对于NFSv3和NFSv4、不包括特定的操作系统。通常支持所有符合RFC的操作系统。在联机IMT中搜索NFSv3或NFSv4支持时、请勿选择特定操作系统、因为不会显示任何匹配项。常规策略隐式支持所有操作系统。

Linux NFSv3 TCP插槽表

TCP插槽表相当于主机总线适配器(Host Bus Adapter、HBA)队列深度的NFSv3。这些表可控制任何时候都可以处理的NFS操作的数量。默认值通常为16、该值太低、无法实现最佳性能。在较新的Linux内核上会出现相反的问题、这会自动将TCP插槽表限制增加到使NFS服务器充满请求的级别。

为了获得最佳性能并防止出现性能问题、请调整控制TCP插槽表的内核参数。

运行 `sysctl -a | grep tcp.*.slot_table` 命令、并观察以下参数：

```
# sysctl -a | grep tcp.*.slot_table
sunrpc.tcp_max_slot_table_entries = 128
sunrpc.tcp_slot_table_entries = 128
```

所有Linux系统都应包括 `sunrpc.tcp_slot_table_entries`，但只有部分包括 `sunrpc.tcp_max_slot_table_entries`。它们都应设置为128。

小心
如果未设置这些参数、可能会对性能产生显著影响。在某些情况下、性能会受到限制、因为Linux操作系统发出的I/O不足在其他情况下、随着Linux操作系统尝试问题描述的I/O数超过可处理的I/O数、I/O时间会增加。

ADr和NFS

一些客户报告了因中的数据量过多而导致的性能问题 `ADR` 位置。通常、只有在积累了大量性能数据之后、才会出现此问题。I/O过多的原因未知、但此问题似乎是由Oracle进程反复扫描目标目录以查找更改引起的。

卸下 `noac` 和 / 或 `actimeo=0` 挂载选项允许进行主机操作系统缓存并降低存储I/O级别。

> * NetApp建议*不要放置 `ADR` 使用的文件系统上的数据 `noac` 或 `actimeo=0` 因为可能会出现性能问题。分开 `ADR` 如果需要、可将数据迁移到其他挂载点。

NFS-rootonly和mount-rootonly

ONTAP包含一个名为的NFS选项 `nfs-rootonly` 用于控制服务器是否接受来自高端口的NFS流量连接。作为一项安全措施、只有root用户才允许使用1024以下的源端口打开TCP/IP连接、因为此类端口通常保留供操作系统使用、而不是供用户进程使用。此限制有助于确保NFS流量来自实际操作系统NFS客户端、而不是模拟NFS客户端的恶意进程。Oracle DNFS客户端是用户空间驱动程序、但该进程以root用户身份运行、因此通常不需要更改的值 `nfs-rootonly`。这些连接是从低端口进行的。

。 `mount-rootonly` 仅选件适用场景NFSv3。它控制是否从大于1024的端口接受RPC挂载调用。使用DNFS时、客户端将再次以root身份运行、因此它可以打开1024以下的端口。此参数无效。

通过NFS 4.0及更高版本打开与DNFS连接的进程不会以root身份运行、因此需要1024以上的端口。。 `nfs-rootonly` 必须将参数设置为disabled、DNFS才能完成连接。

条件 `nfs-rootonly` 处于启用状态、则会在打开DNFS连接的挂载阶段挂起。sqlplus输出类似于：

```
SQL>startup
ORACLE instance started.
Total System Global Area 4294963272 bytes
Fixed Size                  8904776 bytes
Variable Size             822083584 bytes
Database Buffers         3456106496 bytes
Redo Buffers                7868416 bytes
```

可以按如下方式更改此参数：

```
Cluster01::> nfs server modify -nfs-rootonly disabled
```

> 在极少数情况下、您可能需要将NFS-rootonly和mount-rootonly更改为disabled。如果服务器管理的TCP连接数量非常多、则可能没有低于1024的可用端口、并且操作系统会强制使用更高的端口。要完成连接、需要更改这两个ONTAP参数。

NFS导出策略：super用户 和set_id

如果Oracle二进制文件位于NFS共享上、则导出策略必须包括超级用户和set_id权限。

用于用户主目录等通用文件服务的共享NFS导出通常会强制转换root用户。这意味着挂载了文件系统的主机上的root用户发出的请求会重新映射为权限较低的其他用户。这有助于防止特定服务器上的root用户访问共享服务器上的数据、从而保护数据安全。在共享环境中、set_id位也可能存在安全风险。set_id位允许以与调用命令的用户不同的用户身份运行进程。例如、由root用户拥有且具有set_id位的shell脚本以root用户身份运行。如果其他用户可以更改该shell脚本、则任何非root用户都可以通过更新脚本以root用户身份问题描述命令。

Oracle二进制文件包含root用户拥有的文件、并使用set_id位。如果在NFS共享上安装了Oracle二进制文件、则导出策略必须包含适当的超级用户和set_id权限。在以下示例中、此规则同时包含这两者 `allow-suid` 和许可 `superuser` 使用系统身份验证的NFS客户端的(root)访问权限。

```
Cluster01::> export-policy rule show -vserver vserver1 -policyname orabin
-fields allow-suid,superuser
vserver   policyname ruleindex superuser allow-suid
--------- ---------- --------- --------- ----------
vserver1  orabin     1         sys       true
```

NFSv4/4.1配置

对于大多数应用程序、NFS3和NFSv4之间的差别非常小。应用程序I/O通常非常简单、不会从NFSv4中提供的某些高级功能中显著受益。从数据库存储角度来看、较高版本的NFS不应视为"升级"、而应视为包含其他功能的NFS版本。例如、如果需要Kerberos隐私模式(krb5p)的端到端安全性、则需要NFSv4。

> *如果需要NFSv4功能、NetApp建议*使用NFSv4.1。在NFSv4.1中、NFSv4协议有一些功能增强功能、可提高某些边缘情况下的故障恢复能力。

与简单地将挂载选项从vs=3更改为vs=4.1相比、切换到NFSv4更为复杂。有关使用ONTAP配置NFSv4的更完整说明、包括有关配置操作系统的指导、请参见 "TR-4067：《基于ONTAP的NFS最佳实践》"。本技术报告的以下各节介绍了使用NFSv4的一些基本要求。

NFSv4域

有关NFSv4/4.1配置的完整说明不在本文档的讨论范围之内、但一个常见问题是域映射不匹配。从sysadmin的角度来看、NFS文件系统似乎运行正常、但应用程序会报告有关某些文件的权限和/或set_id的错误。在某些情况下、管理员错误地得出结论、认为应用程序二进制文件的权限已损坏、并在实际问题是域名时运行了chown或chmod命令。

在ONTAP SVM上设置NFSv4域名：

```
Cluster01::> nfs server show -fields v4-id-domain
vserver   v4-id-domain
--------- ------------
vserver1  my.lab
```

主机上的NFSv4域名在中进行设置 `/etc/idmap.cfg`

```
[root@host1 etc]# head /etc/idmapd.conf
[General]
#Verbosity = 0
# The following should be set to the local NFSv4 domain name
# The default is the host's DNS domain name.
Domain = my.lab
```

域名必须匹配。否则、中将显示类似以下内容的映射错误 `/var/log/messages`：

```
Apr 12 11:43:08 host1 nfsidmap[16298]: nss_getpwnam: name 'root@my.lab'
does not map into domain 'default.com'
```

应用程序二进制文件(如Oracle数据库二进制文件)包括root用户拥有的具有set_id位的文件、这意味着NFSv4域名不匹配会导致Oracle启动失败、并显示有关名为的文件的所有权或权限的警告 `oradism`，位于中 `$ORACLE_HOME/bin` 目录。它应如下所示：

```
[root@host1 etc]# ls -l /orabin/product/19.3.0.0/dbhome_1/bin/oradism
-rwsr-x--- 1 root oinstall 147848 Apr 17  2019
/orabin/product/19.3.0.0/dbhome_1/bin/oradism
```

如果此文件的所有权为mody、则可能存在NFSv4域映射问题。

```
[root@host1 bin]# ls -l oradism
-rwsr-x--- 1 nobody oinstall 147848 Apr 17  2019 oradism
```

要修复此问题、请选中 `/etc/idmap.cfg` 根据ONTAP上的v4-id-domain设置创建文件、并确保它们一致。如果不是、请进行所需的更改、然后运行 `nfsidmap -c`，然后等待片刻，让更改传播。然后、文件所有权应正确识别为root。如果用户尝试运行 `chown root` 更正NFS域配置之前、可能需要在此文件上运行 `chown root` 再次重申。

Oracle (Oracle)

Oracle数据库可以通过两种方式使用NFS。

首先、它可以使用通过操作系统中的本机NFS客户端挂载的文件系统。这有时称为内核NFS或kNFS。Oracle数据库挂载和使用NFS文件系统的方式与任何其他应用程序使用NFS文件系统的方式完全相同。

第二种方法是Oracle Direct NFS (DNFS)。这是在Oracle数据库软件中实施的NFS标准。它不会更改数据库管理程序配置或管理Oracle数据库的方式。只要存储系统本身具有正确的设置、DNFS的使用就应该对DBA团队和最终用户透明。

启用了DNFS功能的数据库仍会挂载常见的NFS文件系统。数据库打开后、Oracle数据库将打开一组TCP/IP会话并直接执行NFS操作。

直接NFS

Oracle的直接NFS的主要价值是绕过主机NFS客户端、直接在NFS服务器上执行NFS文件操作。要启用此功能、只需更改Oracle磁盘管理器(ODM)库即可。Oracle文档提供了此过程的说明。

使用DNFS可以显著提高I/O性能、并减少主机和存储系统上的负载、因为I/O是以尽可能最高效的方式执行的。

此外，Oracle DNFS还包括一个*选项*，用于实现网络接口多路径和容错。例如、可以将两个10 Gb接口绑定在一起、以提供20 Gb的带宽。一个接口发生故障会导致在另一个接口上重试I/O。整体操作与FC多路径非常相似。多路径早在几年前就已很常见、当时1 Gb以太网是最常用的标准。10 Gb NIC足以满足大多数Oracle工作负载的需求、但如果需要更多NIC、则可以绑定10 Gb NIC。

使用DNFS时、请务必安装Oracle文档1495104.1中所述的所有修补程序。如果无法安装修补程序、则必须对环境进行评估、以确保该文档中所述的错误不会出现发生原因问题。在某些情况下、无法安装所需的修补程序会导致无法使用DNFS。

请勿将DNFS与任何类型的轮叫名称解析结合使用、包括DNS、DDNS、NIS或任何其他方法。其中包括ONTAP中提供的DNS负载平衡功能。当使用DNFS的Oracle数据库将主机名解析为IP地址时、它在后续查找中不得更改。这可能会导致Oracle数据库崩溃并可能导致数据损坏。

直接NFS和主机文件系统访问

对于依赖主机上挂载的可见文件系统的应用程序或用户活动、使用DNFS有时可能会出现发生原因问题、因为DNFS客户端会从主机操作系统带外访问文件系统。DNFS客户端可以在不了解操作系统的情况下创建、删除和修改文件。

如果使用单实例数据库的挂载选项、则可以缓存文件和目录属性、这也意味着可以缓存目录的内容。因此、DNFS可以创建文件、在操作系统重新读取目录内容和文件对用户可见之前、存在一个短暂的延迟。这通常不是问题，但在极少数情况下，SAP BR*Tools等实用程序可能会出现问题。如果发生这种情况、请更改挂载选项以使用针对Oracle RAC的建议来解决此问题。此更改会导致禁用所有主机缓存。

只有在以下情况下才更改挂载选项：(a)使用DNFS；(b)问题是由于文件可见性滞后而导致的。如果未使用DNFS、则在单实例数据库上使用Oracle RAC挂载选项会导致性能下降。

> 请参见有关的注释 `nosharecache` 在中 "Linux NFS挂载选项" 适用于可能会产生异常结果的Linux专用DNFS问题描述。

Oracle数据库和NFS租用和锁定

NFSv3处于无状态。这实际上意味着、NFS服务器(ONTAP)不会跟踪挂载了哪些文件系统、由谁挂载或哪些锁定真正到位。

ONTAP确实具有一些记录挂载尝试的功能、因此您可以了解哪些客户端可能正在访问数据、并且可能存在建议锁定、但该信息并不能保证100%完整。此操作无法完成、因为跟踪NFS客户端状态不是NFSv3标准的一部分。

NFSv4状态

相反、NFSv4是有状态的。NFSv4服务器可跟踪哪些客户端正在使用哪些文件系统、哪些文件存在、哪些文件和/或文件区域被锁定等 这意味着NFSv4服务器之间需要定期进行通信、以使状态数据保持最新。

NFS服务器所管理的最重要状态是NFSv4锁定和NFSv4租约、它们彼此交织在一起。您需要了解每种方法本身的工作原理、以及它们之间的关系。

NFSv4锁定

对于NFSv3、建议使用锁定。NFS客户端仍可修改或删除"锁定"文件。NFSv3锁定本身不会过期、必须将其删除。这会造成问题。例如、如果您有一个集群应用程序创建了NFSv3锁定、而其中一个节点发生故障、您该怎么办？您可以对运行正常的节点上的应用程序进行编码、以解除锁定、但您如何知道这是安全的？可能是"故障"节点正常运行、但未与集群的其余部分通信？

对于NFSv4、锁定的持续时间有限。只要持有锁定的客户端继续向NFSv4服务器签入、就不允许任何其他客户端获取这些锁定。如果客户端无法签入NFSv4、则锁定最终会被服务器撤消、其他客户端将能够请求并获取锁定。

NFSv4租约

NFSv4锁定与NFSv4租约关联。当NFSv4客户端与NFSv4服务器建立连接时、它将获得租约。如果客户端获得锁定(锁定类型有多种)、则锁定与租约关联。

此租约已定义超时。默认情况下、ONTAP会将超时值设置为30秒：

```
Cluster01::*> nfs server show -vserver vserver1 -fields v4-lease-seconds

vserver   v4-lease-seconds
--------- ----------------
vserver1  30
```

这意味着、NFSv4客户端需要每30秒与NFSv4服务器签入一次、才能续订其租约。

任何活动都会自动续订租约、因此、如果客户端正在执行工作、则无需执行添加操作。如果某个应用程序变得安静并且没有执行实际工作、则需要执行某种保活操作(称为序列)。基本上只是说"我还在这里、请刷新我的租约"。

```
 *Question:* What happens if you lose network connectivity for 31 seconds?
NFSv3处于无状态。它不需要来自客户端的通信。NFSv4是有状态的、租赁期过后、租约将过期、锁定将被撤消、锁定的文件将提供给其他客户端使用。
```

借助NFSv3、您可以四处移动网络缆线、重新启动网络交换机、更改配置、并确保不会发生任何不良事件。应用程序通常只需耐心等待网络连接重新工作即可。

使用NFSv4时、您有30秒的时间(除非您已在ONTAP中增加了该参数的值)来完成工作。如果超过此限制、您的租约将超时。通常、这会导致应用程序崩溃。

例如、如果您有一个Oracle数据库、并且网络连接丢失(有时称为"网络分区")、超过租约超时时间、则数据库将崩溃。

下面是一个示例、说明在发生这种情况时Oracle警报日志中会发生什么情况：

```
2022-10-11T15:52:55.206231-04:00
Errors in file /orabin/diag/rdbms/ntap/NTAP/trace/NTAP_ckpt_25444.trc:
ORA-00202: control file: '/redo0/NTAP/ctrl/control01.ctl'
ORA-27072: File I/O error
Linux-x86_64 Error: 5: Input/output error
Additional information: 4
Additional information: 1
Additional information: 4294967295
2022-10-11T15:52:59.842508-04:00
Errors in file /orabin/diag/rdbms/ntap/NTAP/trace/NTAP_ckpt_25444.trc:
ORA-00206: error in writing (block 3, # blocks 1) of control file
ORA-00202: control file: '/redo1/NTAP/ctrl/control02.ctl'
ORA-27061: waiting for async I/Os failed
```

如果您查看系统日志、应会看到以下几个错误：

```
Oct 11 15:52:55 host1 kernel: NFS: nfs4_reclaim_open_state: Lock reclaim
failed!
Oct 11 15:52:55 host1 kernel: NFS: nfs4_reclaim_open_state: Lock reclaim
failed!
Oct 11 15:52:55 host1 kernel: NFS: nfs4_reclaim_open_state: Lock reclaim
failed!
```

日志消息通常是问题的第一个迹象、而不是应用程序冻结。通常、网络中断期间不会显示任何内容、因为尝试访问NFS文件系统的进程和操作系统本身会被阻止。

网络重新正常运行后、将显示这些错误。在上面的示例中、重新建立连接后、操作系统尝试重新获取锁定、但太晚了。租约已过期、锁定已被删除。这会导致错误传播到Oracle层、并在警报日志中显示此消息。根据数据库的版本和配置、这些模式可能会有所不同。

总之、NFSv3可以承受网络中断、但NFSv4更敏感、并会规定一个明确的租赁期限。

如果30秒超时不可接受、该怎么办？如果您管理的网络动态变化、交换机重新启动或缆线重新定位会导致网络偶尔中断、该怎么办？您可以选择延长租赁期限、但是否要延长、需要说明NFSv4宽限期。

NFSv4宽限期

如果重新启动NFSv3服务器、它几乎可以立即提供IO。它并没有保持任何关于客户的状态。这样、ONTAP接管操作通常看起来接近瞬时。一旦控制器准备好开始提供数据、它就会向网络发送一个ARP、以指示拓扑发生变化。客户端通常会近乎即时地检测到这一点、数据将恢复流动。

但是、NFSv4会短暂暂停。这只是NFSv4工作原理的一部分。

NFSv4服务器需要跟踪租约、锁定以及谁在使用哪些数据。如果NFS服务器发生故障并重新启动、断电片刻或在维护活动期间重新启动、则会导致租用/锁定以及其他客户端信息丢失。在恢复操作之前、服务器需要确定哪个客户端正在使用哪些数据。这就是宽限期的所在。

如果您突然关闭并重新启动NFSv4服务器。恢复后、尝试恢复IO的客户端将收到一个响应、该响应本质上说："我丢失了租用/锁定信息。是否要重新注册您的锁？" 这是宽限期的开始。在ONTAP上、默认为45秒：

```
Cluster01::> nfs server show -vserver vserver1 -fields v4-grace-seconds

vserver   v4-grace-seconds
--------- ----------------
vserver1  45
```

因此、在重新启动后、控制器将暂停IO、而所有客户端都将回收其租约和锁定。宽限期结束后、服务器将恢复IO操作。

租赁超时与宽限期

宽限期和租赁期是连接的。如上所述、默认租约超时为30秒、这意味着NFSv4客户端必须至少每30秒向服务器签入一次、否则它们将失去租约、进而失去锁定。有一个宽限期、允许NFS服务器重建租用/锁定数据、默认为45秒。ONTAP要求宽限期比租赁期长15秒。这样可以确保设计为至少每30秒续订一次租约的NFS客户端环境

能够在重新启动后与服务器签入。45秒的宽限期可确保所有希望至少每30秒续订一次租约的客户都有机会续订租约。

如果不接受30秒的超时时间、您可以选择延长租赁期限。如果要将租约超时时间增加到60秒、以承受60秒网络中断、则必须将宽限期至少增加到75秒。ONTAP要求该期限比租赁期高15秒。这意味着、在控制器故障转移期间、IO暂停时间将更长。

这通常不会是问题。通常、用户每年只更新ONTAP控制器一次或两次、并且很少会因硬件故障而发生计划外故障转移。此外、如果您的网络可能会发生60秒的网络中断、并且您需要将租赁超时时间设置为60秒、则可能不会反对偶尔发生的存储系统故障转移、从而导致75秒的暂停。您已确认您的网络经常暂停60秒以上。

Oracle数据库的**NFS**缓存

如果存在以下任一挂载选项、则会禁用主机缓存：

```
cio, actimeo=0, noac, forcedirectio
```

这些设置可能会对软件安装、修补和备份/还原操作的速度产生严重的负面影响。在某些情况下、尤其是对于集群应用程序、由于需要在集群中的所有节点之间实现缓存一致性、因此必然需要使用这些选项。在其他情况下、客户会错误地使用这些参数、从而导致不必要的性能损害。

许多客户会在安装或修补应用程序二进制文件期间临时删除这些挂载选项。如果用户验证在安装或修补过程中没有其他进程正在使用目标目录、则可以安全地执行此删除。

Oracle数据库的**NFS**传输大小

默认情况下、ONTAP会将NFS I/O大小限制为64K。

大多数应用程序和数据库的随机I/O使用的块大小要小得多、远远低于64K的最大值。大型块I/O通常会并行处理、因此最大64K也不会限制获得最大带宽。

在某些工作负载中、最大64K会产生限制。特别是、如果数据库执行的I/O数量较少但规模较大、则单线程操作(例如备份或恢复操作或数据库完整表扫描)运行速度会更快、效率也会更高。ONTAP的最佳I/O处理大小为256K。

给定ONTAP SVM的最大传输大小可按如下方式进行更改：

```
Cluster01::> set advanced
Warning: These advanced commands are potentially dangerous; use them only
when directed to do so by NetApp personnel.
Do you want to continue? {y|n}: y
Cluster01::*> nfs server modify -vserver vserver1 -tcp-max-xfer-size
262144
Cluster01::*>
```

小心
请勿将ONTAP上允许的最大传输大小减小到低于当前挂载的NFS文件系统的rsize/wsize值。在某些操作系统中、这可能会导致挂起甚至数据损坏。例如、如果NFS客户端当前设置为rsize/wsize 65536,则ONTAP最大传输大小可以在65536- 1048576之间进行调整,但不会产生任何影响,因为客户端本身是有限的。将最大传输大小减小至65536,可能会损坏可用性或数据。

Oracle数据库和NVFAIL

NVFAIL是ONTAP中的一项功能、可确保在灾难性故障转移情形下的完整性。

数据库在存储故障转移事件期间容易损坏、因为它们会维护大量内部缓存。如果在发生灾难性事件时需要强制执行ONTAP故障转移或强制执行MetroCluster切换、而不管整体配置的运行状况如何、则先前确认的结果可能会被有效丢弃。存储阵列的内容会及时向后跳转、数据库缓存的状态不再反映磁盘上数据的状态。此不一致性会导致数据损坏。

缓存可以在应用程序层或服务器层进行。例如、如果Oracle Real Application Cluster (RAC)配置中的服务器在主站点和远程站点上都处于活动状态、则该配置会在Oracle SGA中缓存数据。如果强制切换操作导致数据丢失、则会使数据库面临损坏的风险、因为存储在SGA中的块可能与磁盘上的块不匹配。

在操作系统文件系统层使用缓存不太明显。装载的NFS文件系统中的块可能会缓存在操作系统中。或者、可以将基于主站点上LUN的集群文件系统挂载到远程站点的服务器上、然后再次缓存数据。在这些情况下、NVRAM故障、强制接管或强制切换可能会导致文件系统损坏。

ONTAP通过NVFAIL及其关联设置、保护数据库和操作系统免受这种情况的影响。

ASM Recandation Utility和ONTAP零块检测

启用实时压缩后、ONTAP可以高效删除写入文件或LUN的置零块。Oracle ASM Recasation Utility (ARU)等实用程序的工作方式是向未使用的ASM块区写入零。

这样、数据库管理器便可在删除数据后回收存储阵列上的空间。ONTAP会截获零并取消分配LUN中的空间。回收过程速度极快、因为存储系统中不会写入任何数据。

从数据库角度来看、ASM磁盘组包含零、读取这些LUN区域会产生零流、但ONTAP不会将零存储在驱动器上。而是进行简单的元数据更改、以便在内部将LUN的置零区域标记为任何数据为空。

出于类似的原因、涉及置零数据的性能测试无效、因为零块实际上不会在存储阵列中作为写入进行处理。

> 使用ARU时、请确保已安装Oracle建议的所有修补程序。

Oracle数据库虚拟化

对于选择使用虚拟化来管理任务关键型数据库的NetApp客户来说、使用VMware、Oracle OLVM或KVM实现数据库虚拟化的做法越来越普遍。

可支持性

对于Oracle虚拟化支持策略、尤其是VMware产品支持策略、存在许多误解。听说Oracle完全不支持虚拟化、这

种情况并不少见。这一概念是不正确的、会导致错失从虚拟化中获益的机会。Oracle文档ID 249212.1讨论了实际要求、客户很少考虑这些要求。

如果虚拟化服务器上出现问题、而Oracle支持先前并不知道该问题、则可能会要求客户在物理硬件上重现该问题。运行尖端产品版本的Oracle客户可能不想使用虚拟化、因为可能会出现可支持性问题、但对于使用通用Oracle产品版本的虚拟化客户来说、这种情况并不是现实情况。

存储表示

考虑将数据库虚拟化的客户应根据业务需求制定存储决策。虽然这对于所有IT决策来说都是一个普遍正确的说法、但对于数据库项目来说尤其重要、因为要求的大小和范围差别很大。

存储表示有三个基本选项：

- 虚拟机管理程序数据存储库上的虚拟化LUN
- 由虚拟机上的iSCSI启动程序(而不是虚拟机管理程序)管理的iSCSI LUN
- 虚拟机挂载的NFS文件系统(而不是基于NFS的数据存储库)
- 直接设备映射。客户不喜欢VMware VMM、但物理设备通常仍与KVM和OLVM虚拟化直接映射。

性能

向虚拟化子系统提供存储的方法通常不会影响性能。主机操作系统、虚拟化网络驱动程序和虚拟机管理程序数据存储库实施均经过高度优化、只要遵循基本最佳实践、通常可以占用虚拟机管理程序与存储系统之间的所有可用FC或IP网络带宽。在某些情况下、使用一种存储表示方法可能比使用另一种存储表示方法更容易获得最佳性能、但最终结果应该是可比的。

易管理性

决定如何向虚拟化子系统提供存储的关键因素是可管理性。方法没有对错之处。最佳方法取决于IT运营需求、技能和偏好。

需要考虑的因素包括：

- *Transparency。*当VM管理其文件系统时，数据库管理员或系统管理员可以更轻松地确定其数据的文件系统源。访问文件系统和LUN的方式与使用物理服务器相同。
- *一致性。*如果虚拟机拥有其文件系统、则使用或不使用虚拟机管理程序层会影响易管理性。配置、监控、数据保护等过程同样适用于整个资产、包括虚拟化和非虚拟化环境。

  另一方面、在完全虚拟化的数据中心中、根据上述相同的原理(一致性、使用相同的配置、保护、监控和数据保护过程的能力)、在整个占用空间中使用基于数据存储库的存储可能更好。

- *稳定性和故障排除。*当虚拟机拥有其文件系统时、由于虚拟机上存在整个存储堆栈、因此提供良好、稳定的性能和解决问题会更加简单。虚拟机管理程序的唯一角色是传输FC或IP帧。如果配置中包含数据存储库、则会引入另一组超时、参数、日志文件和潜在错误、从而使配置复杂。
- *可移动性。*当VM拥有其文件系统时、移动Oracle环境的过程将变得更加简单。文件系统可以轻松地在虚拟化和非虚拟化子系统之间移动。
- *受制于供应商。*将数据放入数据存储库后、使用不同的虚拟机管理程序或将数据完全从虚拟化环境中取出将变得非常困难。
- *启用Snapshot。*由于带宽相对有限、虚拟化环境中的传统备份过程可能会成为一个问题。例如、四端

口10GbE中继可能足以满足许多虚拟化数据库的日常性能需求、但此类中继不足以使用RMAN或其他需要流式传输完整大小数据副本的备份产品执行备份。因此、日益整合的虚拟化环境需要通过存储快照执行备份。这样、无需纯粹为了满足备份窗口中的带宽和CPU要求而过度构建虚拟机管理程序配置。

使用子系统拥有的文件系统有时可以更轻松地利用基于快照的备份和还原、因为需要保护的存储对象可以更轻松地确定目标。但是、越来越多的虚拟化数据保护产品能够与数据存储库和快照完美集成。在决定如何将存储提供给虚拟化主机之前、应充分考虑备份策略。

部分驱动程序

为了获得最佳性能、使用完全虚拟化的网络驱动程序至关重要。使用数据存储库时、需要使用一个虚拟化的SCSI驱动程序。与虚拟化驱动程序相比、超虚拟化设备驱动程序可以使子系统更深入地集成到虚拟机管理程序中、而在模拟驱动程序中、虚拟机管理程序会花费更多的CPU时间来模拟物理硬件的行为。

过量使用**RAM**

过量使用RAM意味着在不同主机上配置的虚拟化RAM要多于物理硬件上的虚拟化RAM。否则可能会出现发生原因意外的性能问题。对数据库进行虚拟化时、虚拟机管理程序不得将Oracle SGA的底层块交换到存储中。这样做会导致性能结果高度不稳定。

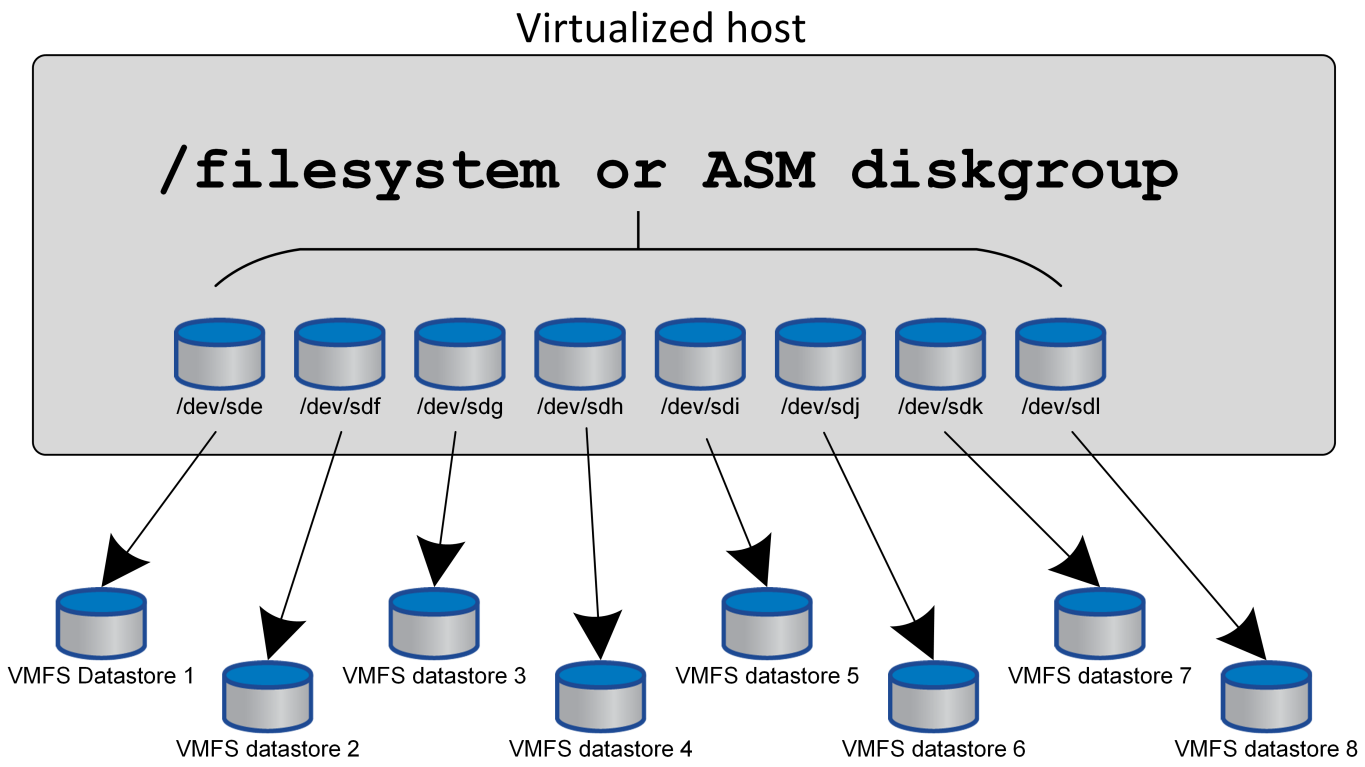
数据存储库条带化

将数据库与数据存储库结合使用时、需要考虑一个与性能相关的关键因素—条带化。

VMFS等数据存储库技术可以跨越多个LUN、但它们不是条带化设备。这些LUN会串联在一起。最终结果可能是LUN热点。例如、典型的Oracle数据库可能具有一个8-LUN ASM磁盘组。所有8个虚拟化LUN均可配置在一个8 LUN VMFS数据存储库上、但无法保证数据将驻留在哪些LUN上。得到的配置可能是所有8个虚拟化LUN都占用VMFS数据存储库中的一个LUN。这将成为性能瓶颈。

通常需要条带化。对于某些虚拟机管理程序(包括KVM)、可以按所述使用LVM条带化来构建数据存储库 "此处"。使用VMware时、架构看起来略有不同。每个虚拟化LUN都需要放置在不同的VMFS数据存储库上。

例如：



这种方法的主要驱动因素不是ONTAP、这是因为一个虚拟机或虚拟机管理程序LUN可并行处理的操作数存在固有限制。一个ONTAP LUN支持的IOPS通常远远超过主机可以请求的IOPS。单个LUN性能限制几乎是主机操作系统的结果。因此、大多数数据库需要4到8个LUN才能满足其性能需求。

VMware架构需要仔细规划其架构、以确保此方法不会遇到数据存储库和/或LUN路径最大值。此外、对于每个数据库、不需要一组唯一的VMFS数据存储库。主要需求是确保每个主机都有一组从虚拟化LUN到存储系统本身上后端LUN的干净的4到8 IO路径。在极少数情况下、即使数据存储库数量更多、也可能有利于满足真正的极致性能需求、但所有数据库中通常有95%的数据库需要使用4到8个LUN。在典型的OS/ONTAP /网络配置下、包含8个LUN的单个ONTAP卷最多可支持250、000次随机Oracle块IOPS。

分层

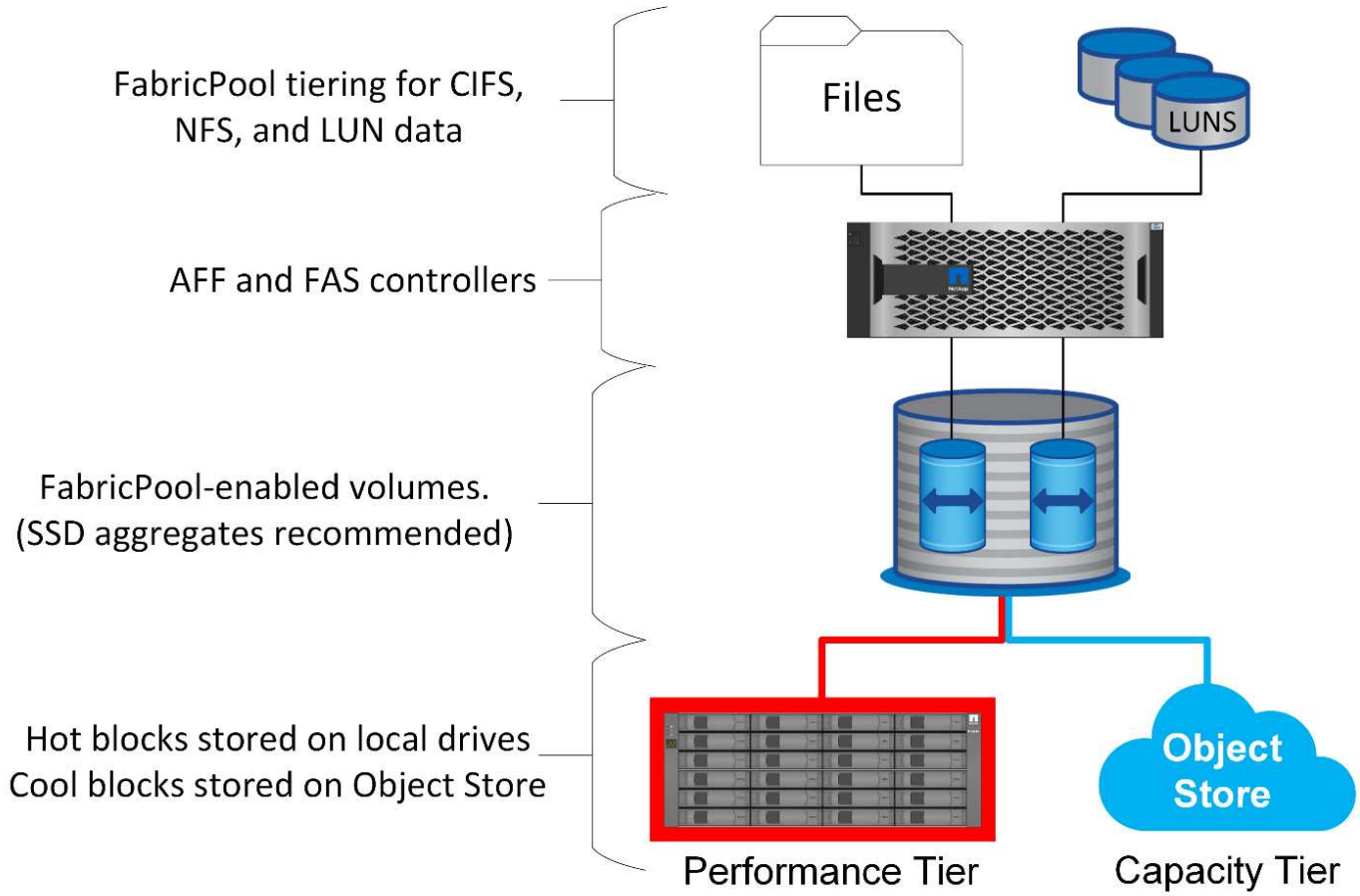
Oracle数据库FabricPool层概述

要了解FabricPool层如何影响Oracle和其他数据库、需要了解低级别FabricPool架构。

架构

FabricPool是一种分层技术、可将块分为热块或冷块、并将其放置在最合适的存储层中。性能层通常位于SSD存储上、并托管热数据块。容量层位于对象存储上、用于托管超酷数据块。对象存储支持包括NetApp StorageGRID、ONTAP S3、Microsoft Azure Blot存储、阿里云对象存储服务、IBM云对象存储、Google云存储和Amazon AWS S3。

可以使用多个分层策略来控制块的分类方式、这些策略可以按卷设置、也可以根据需要进行更改。在性能层和容量层之间仅移动数据块。定义LUN和文件系统结构的元数据始终保留在性能层上。因此、可在ONTAP上集中管理。文件和LUN的显示与任何其他ONTAP配置中存储的数据没有任何不同。NetApp AFF或FAS控制器会应用定义的策略将数据移动到相应的层。



对象存储提供程序

对象存储协议使用简单的HTTP或HTTPS请求来存储大量数据对象。对对象存储的访问必须可靠、因为从ONTAP进行数据访问取决于请求的及时处理。选项包括Amazon S3 Standard和Infrequent Access选项、以及Microsoft Azure Hot and Cool Blot Storage、IBM Cloud和Google Cloud。不支持Amazon Glacier和Amazon Archive等归档选项、因为检索数据所需的时间可能会超出主机操作系统和应用程序的容错范围。

NetApp StorageGRID也受支持、是最佳的企业级解决方案。它是一个高性能、可扩展且高度安全的对象存储系统、可以为FabricPool数据以及越来越可能成为企业应用程序环境一部分的其他对象存储应用程序提供地理冗余。

StorageGRID还可以通过避免许多公有云提供商为从其服务中读取数据而收取的传出费用来降低成本。

数据和元数据

请注意、此处的术语"数据"适用场景是指实际数据块、而不是元数据。仅对数据块进行分层、而元数据保留在性能层中。此外、只有读取实际数据块时、数据块的状态才会受到热或冷的影响。仅读取文件的名称、时间戳或所有权元数据并不会影响底层数据块的位置。

备份

虽然FabricPool可以显著减少存储占用空间、但它本身并不是一个备份解决方案。NetApp WAFL元数据始终保留在性能层上。如果灾难性灾难破坏了性能层、则无法使用容量层上的数据创建新环境、因为该环境不包含WAFL元数据。

但是、FabricPool可以成为备份策略的一部分。例如、可以为FabricPool配置NetApp SnapMirror复制技术。镜

像的每一半都可以与对象存储目标建立自己的连接。结果是生成两个独立的数据副本。主副本由性能层上的块以及容量层中的关联块组成、副本是第二组性能和容量块。

分层策略

Oracle数据库**FabricPool**分层策略

ONTAP提供了四个策略、用于控制性能层上的Oracle数据如何成为重新定位到容量层的候选对象。

仅快照

。 `snapshot-only` `tiering-policy` 仅适用于未与活动文件系统共享的块。实际上、它需要对数据库备份进行层化。创建快照后、块将成为分层的候选块、然后将其覆盖、从而导致块仅存在于快照中。之前的延迟 `snapshot-only` 块被视为冷却由控制 `tiering-minimum-cooling-days` 卷的设置。从ONTAP 9.8开始、此范围为2到183天。

许多数据集的更改率较低、因此此策略节省的空间极少。例如、在ONTAP上观察到的典型数据库每周的更改率小于5%。数据库归档日志可能会占用大量空间、但它们通常仍存在于活动文件系统中、因此不适合在此策略下进行分层。

自动

。 `auto` 层划分策略可将层划分扩展到快照特定的块以及活动文件系统中的块。块被视为冷却之前的延迟由控制 `tiering-minimum-cooling-days` 卷的设置。从ONTAP 9.8开始、此范围为2到183天。

此方法可启用在中不可用的层选项 `snapshot-only` 策略。例如、数据保护策略可能需要保留90天的某些日志文件。如果将冷却期设置为3天、则会将超过3天的任何日志文件从性能层中分层出来。此操作可释放性能层上的大量空间、同时仍允许您查看和管理整个90天的数据。

无

。 `none` 分层策略可防止从存储层对任何其他块进行分层、但容量层中的任何数据仍会保留在容量层中、直到被读取为止。如果随后读取该块、则会将其移回并放置在性能层上。

使用的主要原因 `none` 分层策略可防止对块进行分层、但随着时间的推移更改策略可能会很有用。例如、假设某个特定数据集已广泛分层到容量层、但却出现了对全部性能功能的意外需求。可以更改此策略、以防止任何其他分层、并确认随着IO增加而读回的任何块仍保留在性能层中。

全部

。 `all` 层策略将取代 `backup` 自ONTAP 9.6起的策略。。 `backup` 策略仅应用于数据保护卷、即SnapMirror或NetApp SnapVault目标。。 `all` 策略的功能相同、但不限于数据保护卷。

使用此策略、块将立即视为冷数据块、并有资格立即分层到容量层。

此策略尤其适用于长期备份。它还可用作分层存储管理(HSM)的一种形式。过去、HSM通常用于将文件的数据块分层到磁带、同时使文件本身在文件系统上保持可见。使用的FabricPool卷 `all` 通过策略、您可以将文件存储在一个可见且易于管理的位置、但几乎不会占用本地存储层上的空间。

Oracle数据库和FabricPool检索策略

分层策略用于控制将哪些Oracle数据库块从性能层分层到容量层。检索策略控制读取已分层的块时发生的情况。

Default

所有FabricPool卷的初始设置为 `default`，这意味着该行为由`云检索策略控制。`具体行为取决于所使用的层策略。

- `auto`–仅检索随机读取的数据
- `snapshot-only`–检索所有按顺序或随机读取的数据
- `none`–检索所有按顺序或随机读取的数据
- `all`不从容量层检索数据

读取时

正在设置 ... `cloud-retrieval-policy` 到读取时会覆盖默认行为、因此读取任何分层数据都会将该数据返回到性能层。

例如、某个卷在下可能已长时间使用不多 `auto` 分层策略和大多数块现在已分层。

如果业务需求发生意外变化、需要重复扫描某些数据以准备特定报告、则可能需要更改 `cloud-retrieval-policy` to `on-read` 以确保读取的所有数据(包括按顺序读取的数据和随机读取的数据)都返回到性能层。这样可以提高卷的顺序I/O性能。

提升

提升策略的行为取决于层策略。如果此层策略为 `auto`，然后设置 `cloud-retrieval-policy` `to` `promote` 在下次分层扫描时从容量层恢复所有块。

如果此层策略为 `snapshot-only`，则返回的唯一块是与活动文件系统关联的块。通常、这不会产生任何影响、因为只有数据块在下进行了分层 `snapshot-only` 策略将是专门与快照关联的块。活动文件系统中不会存在分层块。

但是、如果卷上的数据是通过卷SnapRestore或文件克隆操作从快照还原的、则活动文件系统现在可能需要一些因仅与快照关联而分层出的块。可能需要临时更改 `cloud-retrieval-policy` 策略为 `promote` 以快速检索所有本地所需的块。

从不

请勿从容量层检索块。

层策略

Oracle数据库完整文件FabricPool层

虽然FabricPool分层在块级运行、但在某些情况下、可用于提供文件级分层。

许多应用程序数据集都按日期进行组织、随着数据老化、访问这些数据的可能性通常越来越小。例如、银行可能

有一个PDF文件存储库、其中包含五年的客户对账单、但只有最近几个月处于活动状态。FabricPool可用于将旧数据文件重新定位到容量层。冷却期为14天、可确保最近14天的PDF文件仍保留在性能层上。此外、至少每14天读取一次的文件将保持热状态、因此仍保留在性能层上。

策略

要实施基于文件的分层方法、您必须拥有已写入且随后未修改的文件。。 `tiering-minimum-cooling-days` 策略应设置得足够高、以便可能需要的文件仍保留在性能层上。例如、如果某个数据集需要最近60天的数据且性能最佳、则需要设置 `tiering-minimum-cooling-days` 期限为60。根据文件访问模式、也可以实现类似的结果。例如、如果需要最近90天的数据、而应用程序正在访问这90天的数据、则数据将保留在性能层上。通过设置 `tiering-minimum-cooling-days` 从2开始、当数据变得不太活跃后、您会收到分层提示。

。 `auto` 要对这些块进行层化、需要使用策略、因为只有 `auto` 策略会影响活动文件系统中的块。

> 任何类型的数据访问都会重置热图数据。病毒扫描、索引编制甚至是读取源文件的备份活动会阻止分层、因为需要分层 `tiering-minimum-cooling-days` 从未达到阈值。

Oracle部分文件**FabricPool**分层

由于FabricPool在块级别工作、因此可能会更改的文件可以部分分层到对象存储、同时也可以部分保留在性能层上。

这在数据库中很常见。已知包含非活动块的数据库也是FabricPool层的候选数据库。例如、供应链管理数据库可能包含历史信息、这些信息在需要时必须可用、但在正常操作期间不会访问。可以使用FabricPool有选择地重新定位非活动块。

例如、使用的FabricPool卷上运行的数据文件 `tiering-minimum-cooling-days` 90天期限将在性能层上保留前90天访问的任何块。但是、任何在90天内未访问的内容都会重新定位到容量层。在其他情况下、正常应用程序活动会将正确的块保留在正确的层上。例如、如果数据库通常用于定期处理前60天的数据、则要低得多 `tiering-minimum-cooling-days` 可以设置期限、因为应用程序的自然活动可确保不会过早重新定位块。

。 `auto` 对数据库使用策略时应谨慎。许多数据库都定期开展活动、例如季度末流程或重新编制索引操作。如果这些操作的期限大于 `tiering-minimum-cooling-days` 可能会发生性能问题。例如、如果季度末处理需要1 TB的数据、而这些数据在其他情况下未被触及、则这些数据现在可能位于容量层上。从容量层读取的速度通常非常快、可能不会出现发生原因性能问题、但具体结果取决于对象存储配置。

策略

。 `tiering-minimum-cooling-days` 策略应设置得足够高、以保留性能层上可能需要的文件。例如、如果数据库中可能需要最新60天的数据且性能最佳、则需要设置 `tiering-minimum-cooling-days` 期限为60天。根据文件的访问模式、也可以实现类似的结果。例如、如果需要最近90天的数据、而应用程序正在访问这90天的数据、则数据将保留在性能层上。设置 `tiering-minimum-cooling-days` 在数据变得不太活跃后、将立即对数据进行分层。

。 `auto` 要对这些块进行层化、需要使用策略、因为只有 `auto` 策略会影响活动文件系统中的块。

> 任何类型的数据访问都会重置热图数据。因此、数据库完整表扫描甚至读取源文件的备份活动都会阻止分层、因为需要分层 `tiering-minimum-cooling-days` 从未达到阈值。

Oracle数据库归档日志层

FabricPool最重要的用途或许是提高已知冷数据(如数据库事务日志)的效率。

大多数关系数据库都在事务日志归档模式下运行、以提供时间点恢复。通过记录事务日志中的更改来提交对数据库的更改、事务日志将保留而不被覆盖。因此、可能需要保留大量归档事务日志。许多其他应用程序工作流也存在类似的例子、这些工作流生成的数据必须保留、但极不可能被访问。

FabricPool通过提供具有集成层的单个解决方案解决了这些问题。文件会存储在通常的位置并始终可访问、但在主阵列上几乎不会占用任何空间。

策略

使用 `tiering-minimum-cooling-days` 如果策略设置为几天、则会在性能层上保留最近创建的文件(即近期最可能需要的文件)中的块。然后、旧文件中的数据块将移至容量层。

。 `auto` 在达到冷却阈值时强制执行提示分层、而不管日志是已删除还是仍位于主文件系统中。将所有可能需要的日志存储在活动文件系统中的一个位置也可以简化管理。没有理由通过搜索快照来查找需要还原的文件。

某些应用程序(如Microsoft SQL Server)会在备份操作期间会对事务日志文件进行节段、以便日志不再位于活动文件系统中。可以使用节省容量 `snapshot-only` 分层策略、但 `auto` 策略对日志数据没有用处、因为活动文件系统中的日志数据很少会冷却下来。

采用FabricPool快照层的Oracle

FabricPool的初始版本针对备份用例。唯一可以分层的块类型是不再与活动文件系统中的数据关联的块。因此、只能将快照数据块移至容量层。当您需要确保性能不会受到影响时、这仍然是最安全的一种层选项。

Policies—本地快照

可通过两种方法将非活动快照块分层到容量层。首先是 `snapshot-only` 策略仅针对快照块。虽然 `auto` 策略包括 `snapshot-only` 块、它还会对活动文件系统中的块进行分层。这可能并不可取。

。 `tiering-minimum-cooling-days` 值应设置为一个时间段、以便在性能层上提供还原期间可能需要的数据。例如、关键生产数据库的大多数还原方案都包括前几天某个时间的还原点。设置A `tiering-minimum-cooling-days` 值为3可确保对文件进行任何还原都能使文件立即实现最高性能。活动文件中的所有块仍位于快速存储上、而无需从容量层中恢复。

Policies—复制的快照

使用SnapMirror或SnapVault复制的快照仅用于恢复、通常应使用FabricPool `all` 策略。使用此策略、可以复制元数据、但所有数据块都会立即发送到容量层、从而实现最高性能。大多数恢复过程都涉及顺序I/O、这本身就很高效。应评估从对象存储目标恢复的时间、但在设计完善的架构中、此恢复过程不需要比从本地数据恢复明显慢。

如果复制的数据也要用于克隆、则 `auto` 策略更合适、使用 `tiering-minimum-cooling-days` 包含预计在克隆环境中定期使用的数据的价值。例如、数据库的活动工作集可能包括前三天读取或写入的数据、但也可能包括另外6个月的历史数据。如果是、则 `auto` SnapMirror目标上的策略可使工作集在性能层上可用。

传统应用程序备份包括Oracle Recovery Manager等产品、这些产品可在原始数据库位置之外创建基于文件的备份。

```
 `tiering-minimum-cooling-days` policy of a few days preserves the most
recent backups, and therefore the backups most likely to be required for
an urgent recovery situation, on the performance tier. The data blocks of
the older files are then moved to the capacity tier.
。 `auto`
策略是最适合备份数据的策略。这样可以确保在达到冷却阈值时及时分层、而不管这些文件是已删除
还是仍位于主文件系统中。将所有可能需要的文件存储在活动文件系统中的一个位置也可以简化管理
。没有理由通过搜索快照来查找需要还原的文件。
```

。 `snapshot-only` 可以使策略有效、但该策略仅适用于不再位于活动文件系统中的适用场景块。因此、必须先删除NFS或SMB共享上的文件、然后才能对数据进行分层。

对于LUN配置、此策略的效率甚至会更低、因为从LUN中删除文件只会从文件系统元数据中删除文件引用。LUN上的实际块将一直保留在原位、直到被覆盖为止。这种情况可能会在删除文件和覆盖块并成为可进行层的候选块之间造成长时间延迟。移动有一些好处 `snapshot-only` 块到容量层、但总体而言、FabricPool备份数据管理最适合与结合使用 `auto` 策略。

> 这种方法有助于用户更高效地管理备份所需的空间、但FabricPool本身并不是一种备份技术。将备份文件分层到对象存储可简化管理、因为这些文件在原始存储系统上仍然可见、但对象存储目标中的数据块依赖于原始存储系统。如果源卷丢失、则对象存储数据将不再可用。

Oracle数据库和对象存储访问中断

使用FabricPool对数据集进行分层会导致主存储阵列与对象存储层之间存在依赖关系。有许多对象存储选项可提供不同级别的可用性。请务必了解主存储阵列与对象存储层之间可能断开连接的影响。

如果向ONTAP发出的I/O需要容量层中的数据、而ONTAP无法访问容量层来检索块、则此I/O最终会超时。此超时的影响取决于所使用的协议。在NFS环境中、ONTAP会根据协议使用EjUKEBOX或EDELAY响应进行响应。某些较早的操作系统可能会将此错误视为错误、但Oracle Direct NFS客户端的当前操作系统和当前修补程序级别会将此错误视为可检取的错误、并继续等待I/O完成。

适用场景SAN环境超时时间更短。如果对象存储环境中需要某个块、但该块在两分钟内仍不可访问、则会向主机返回读取错误。ONTAP卷和LUN会保持联机、但主机操作系统可能会将文件系统标记为处于错误状态。

对象存储连接问题 `snapshot-only` 策略不太值得关注、因为只有备份数据是分层的。通信问题会使数据恢复速度变慢、但不会影响正在使用的数据。。 `auto` 和 `all` 策略允许对活动LUN中的冷数据进行分层、这意味着对象存储数据检索期间出现错误可能会影响数据库可用性。采用这些策略的SAN部署只能与专为实现高可用性而设计的企业级对象存储和网络连接结合使用。NetApp StorageGRID是一个更好的选择。

Oracle数据保护

借助**ONTAP**实现**Oracle**数据保护

NetApp知道、数据库中的任务关键型数据最多。

企业无法在不访问其数据的情况下运营、有时数据决定了业务。这些数据必须受到保护；但是、数据保护不仅仅是确保备份可用、它还需要快速可靠地执行备份、同时还要安全地存储这些备份。

数据保护的另一面是数据恢复。如果无法访问数据、则企业会受到影响、并且可能无法运行、直到数据还原为止。此过程必须快速可靠。最后、必须保护大多数数据库免受灾难的影响、这意味着需要维护数据库的副本。副本必须足够最新。使副本成为一个完全正常运行的数据库还必须快速而简单。

> 本文档可替代先前发布的技术报告_TR-4591：《Oracle数据保护：备份、恢复和复制》

规划

正确的企业级数据保护架构取决于数据保留、可恢复性以及在各种事件期间对中断的承受能力方面的业务要求。

例如、考虑范围内的应用程序、数据库和重要数据集的数量。为单个数据集构建备份策略以确保符合典型SLA要求相当简单、因为无需管理太多对象。随着数据集数量的增加、监控变得更加复杂、管理员可能不得不花费越来越多的时间来解决备份故障。随着环境达到云和服务提供商规模、需要采用完全不同的方法。

数据集大小也会影响策略。例如、由于数据集非常小、因此对于使用100 GB数据库进行备份和恢复、有许多选项可供选择。只需使用传统工具从备份介质中复制数据、通常就能提供足够的恢复回路(Recovery)。100 TB数据库通常需要完全不同的策略、除非RTO允许发生多天中断、在这种情况下、可以使用基于副本的传统备份和恢复操作步骤。

最后、备份和恢复过程本身之外还有其他因素。例如、是否存在支持关键生产活动的数据库、从而使恢复成为仅由熟练的数据库管理人员执行的罕见事件？或者、数据库是否属于一个大型开发环境、在该环境中、恢复频繁发生、并由一个通才型IT团队进行管理？

Oracle数据库**RTO**、**RPO**和**SLA**规划

借助ONTAP、您可以根据业务需求轻松定制Oracle数据库数据保护策略。

这些要求包括恢复速度、允许的最大数据丢失量以及备份保留需求等因素。数据保护计划还必须考虑数据保留和还原方面的各种法规要求。最后、必须考虑不同的数据恢复场景、从因用户或应用程序错误而导致的典型和可预见的恢复到包括站点完全丢失在内的灾难恢复场景。

对数据保护和恢复策略进行微小更改可能会对存储、备份和恢复的整体架构产生显著影响。在开始设计工作之前、必须定义并记录标准、以避免使数据保护架构复杂化。不必要的功能或保护级别会导致不必要的成本和管理开销、而最初被忽视的要求可能会导致项目方向错误或需要在最后一刻更改设计。

恢复时间目标

恢复时间目标(Recovery Time目标、Recovery Time目标、Recovery Time目标、Recovery Time目标、Recovery Time目标、Recovery Time目标)定义了恢复服务所允许的最长时间。例如、人力资源数据库的RTO可能为24小时、因为虽然在工作日无法访问此数据会非常不便、但业务仍可继续运营。相比之下、支持银行总分类账的数据库将以分钟甚至几秒钟计量的最短时间。RTO不可能为零、因为必须有方法区分实际服务中断和例行事件(例如网络数据包丢失)。但是、RTO接近零是一项典型要求。

恢复点目标

恢复点目标(RPO)定义了可容忍的最大数据丢失。在许多情况下、RPO完全取决于快照或SnapMirror更新的频率。

在某些情况下、可以通过更频繁地有选择地保护某些数据来提高RPO的主动性。在数据库环境中、RPO通常是指在特定情况下可能丢失多少日志数据的问题。在典型的恢复情形中、如果数据库因产品错误或用户错误而损坏、则RPO应为零、这意味着不会丢失任何数据。恢复操作步骤包括还原数据库文件的早期副本、然后重影日志文件、以使数据库状态达到所需的时间点。此操作所需的日志文件应已位于原始位置。

在异常情况下、日志数据可能会丢失。例如、意外事件或恶意事件 `rm -rf *` 数据库文件的数量可能会导致所有数据被删除。唯一的选择是从备份(包括日志文件)进行还原、而某些数据将不可避免地丢失。在传统备份环境中、要提高RPO、唯一的选择是对日志数据执行重复备份。但是、由于数据会不断移动、而且很难将备份系统作为一项持续运行的服务来维护、因此这一点存在一些限制。高级存储系统的优势之一是能够保护数据免受文件意外或恶意损坏、从而提供更好的RPO、而无需移动数据。

灾难恢复

灾难恢复包括在发生物理灾难时恢复服务所需的IT架构、策略和过程。这可能包括洪水、火灾或有恶意或疏忽意图的人员。

灾难恢复不仅仅是一组恢复过程。它是一个完整的过程、可以识别各种风险、定义数据恢复和服务连续性要求、并提供具有相关过程的正确架构。

在确定数据保护要求时、必须区分典型的RPO和RTO要求以及灾难恢复所需的RPO和RTO要求。某些应用程序环境要求RPO为零、RTO接近零、以应对从相对正常的用户错误到破坏数据中心的火灾等数据丢失情形。然而，这种高水平的保护会产生成本和行政后果。

通常、非灾难数据恢复要求应严格、原因有两个。首先、破坏数据的应用程序错误和用户错误是可以预见的、几乎是不可避免的。其次、只要存储系统不被销毁、设计一个能够实现零RPO和低RTO的备份策略不难。没有理由不解决容易补救的重大风险、这就是为什么本地恢复的RPO和RTO目标应该积极主动的原因。

根据发生灾难的可能性以及相关数据丢失或业务中断的后果、灾难恢复RTO和RPO要求的差别更大。RPO和RTO要求应基于实际业务需求、而不是一般原则。它们必须考虑多种逻辑和物理灾难情形。

逻辑灾难

逻辑灾难包括由用户、应用程序或操作系统错误以及软件故障导致的数据损坏。逻辑灾难还可能包括外部人员利用病毒或蠕虫或利用应用程序漏洞进行的恶意攻击。在这些情况下、物理基础架构未损坏、但底层数据不再有效。

一种日益常见的逻辑灾难类型称为勒索软件、在这种情况下、攻击向量用于对数据进行加密。加密不会损坏数据、但在向第三方付款之前、加密将使数据不可用。越来越多的企业正成为勒索软件黑客攻击的专门目标。针对这种威胁、NetApp提供防篡改快照、在这些快照中、即使存储管理员也无法在配置的到期日期之前更改受保护的数据。

物理灾难

物理灾难包括基础架构的组件发生故障、导致其冗余能力超出范围、从而导致数据丢失或服务长时间丢失。例如、RAID保护可提供磁盘驱动器冗余、而使用HBA可提供FC端口和FC缆线冗余。此类组件的硬件故障是可以预见的、不会影响可用性。

在企业环境中、通常可以使用冗余组件保护整个站点的基础架构、直到唯一可预见的物理灾难情形是站点完全丢失。灾难恢复规划则取决于站点到站点复制。

同步和异步数据保护

理想情况下、所有数据都会在地理位置分散的站点之间同步复制。由于以下几个原因、此类复制并不总是可行甚至不可能实现：

- 同步复制不可避免地会增加写入延迟、因为必须先将所有更改复制到这两个位置、然后应用程序/数据库才能继续处理。所产生的性能影响有时是不可接受的、从而排除了使用同步镜像的可能性。
- 随着100% SSD存储的采用率不断提高、更有可能注意到额外的写入延迟、因为性能预期包括数十万次IOPS和亚微秒延迟。要充分发挥使用100% SSD的优势、可能需要重新审视灾难恢复策略。
- 数据集的字节数持续增长、在确保足够的带宽来支持同步复制方面面临着挑战。
- 数据集的复杂性也在不断增加、在管理大规模同步复制方面也面临着挑战。
- 基于云的策略通常涉及更长的复制距离和延迟、进一步排除了同步镜像的使用。

NetApp提供的解决方案既包括可满足最严苛数据恢复需求的同步复制、也包括可提高性能和灵活性的异步解决方案。此外、NetApp技术还可以与许多第三方复制解决方案(例如Oracle DataGuard)无缝集成

保留时间

数据保护策略的最后一个方面是数据保留时间、数据保留时间可能差别很大。

- 通常要求在主站点上执行14天的夜间备份、在二级站点上执行90天的备份。
- 许多客户创建独立的季度归档、存储在不同的介质上。
- 不断更新的数据库可能不需要历史数据、备份只需保留几天。
- 根据法规要求、可能需要在365天内恢复到任意事务的时间点。

使用**ONTAP**可获得**Oracle**数据库

ONTAP旨在最大程度地提高Oracle数据库的可用性。本文档不会介绍完整的ONTAP高可用性功能问题描述。但是、与数据保护一样、在设计数据库基础架构时、基本了解此功能非常重要。

HA 对

高可用性的基本单位是HA对。每个对都包含冗余链路、以支持将数据复制到NVRAM。NVRAM不是写入缓存。控制器中的RAM用作写入缓存。NVRAM的用途是临时记录数据、以防止发生意外系统故障。在这方面、它类似于数据库重做日志。

NVRAM和数据库重做日志均用于快速存储数据、从而可以尽快提交对数据的更改。直到稍后在ONTAP和大多数数据库平台上的一个称为检查点的过程中、才会更新驱动器(或数据文件)上的永久性数据。在正常操作期间、不会读取NVRAM数据和数据库重做日志。

如果控制器突然出现故障、NVRAM中可能会存储一些尚未写入驱动器的待处理更改。配对控制器会检测到故障、控制驱动器并应用NVRAM中存储的所需更改。

接管和交还

接管和交还是指在HA对中的节点之间转移存储资源职责的过程。接管和返回有两个方面：

- 管理允许访问驱动器的网络连接
- 驱动器本身的管理

支持CIFS和NFS流量的网络接口配置了主位置和故障转移位置。接管包括将网络接口移动到与原始位置位于同一子网的物理接口上的临时主端口。交还包括将网络接口移回其原始位置。可以根据需要调整确切的行为。

在接管和回放期间、不会重新定位支持iSCSI和FC等SAN块协议的网络接口。而是应使用包含完整HA对的路径来配置LUN、从而生成主路径和二级路径。

> 此外、还可以配置指向其他控制器的其他路径、以支持在较大集群中的节点之间重新定位数据、但这不是HA过程的一部分。

接管和返回的第二个方面是磁盘所有权的传输。具体过程取决于多个因素、包括接管/还原的原因以及发出的命令行选项。目标是尽可能高效地执行操作。虽然整个过程看起来可能需要几分钟时间、但驱动器所有权从一个节点转换到另一个节点的实际时刻通常可以以秒为单位进行衡量。

接管时间

在接管和备份操作期间、主机I/O会短暂暂停、但在配置正确的环境中、不应发生应用程序中断。I/O延迟的实际过渡过程通常以秒为单位、但主机可能需要更多时间来识别数据路径中的更改并重新提交I/O操作。

中断的性质取决于协议：

- 在过渡到新物理位置后、支持NFS和CIFS流量的网络接口会向网络发出地址解析协议(Address Resolution Protocol、ARP)请求。这会导致网络交换机更新其介质访问控制(MAC)地址表并恢复处理I/O在计划内接管和移交的情况下、中断通常以秒为单位进行衡量、在许多情况下、无法检测到。某些网络可能较慢、无法完全识别网络路径的变化、而某些操作系统可能会在很短的时间内排队等待大量I/O、必须重试。这会延长恢复I/O所需的时间
- 支持SAN协议的网络接口不会过渡到新位置。主机操作系统必须更改正在使用的一个或多个路径。主机观察到的I/O暂停取决于多个因素。从存储系统角度来看、无法提供I/O的时间段仅为几秒。但是、不同的主机操作系统可能需要额外的时间才能使I/O在重试之前超时。较新的操作系统能够更快地识别路径更改、但较旧的操作系统通常需要长达30秒才能识别更改。

下表显示了存储系统无法为应用程序环境提供数据的预期接管时间。在任何应用程序环境中都不应出现任何错误、接管应显示为IO处理中的短暂暂停。

	NFS	AFF	ASA
计划内接管	15秒	第个问题	2-3秒
计划外接管	30秒	第个问题	2-3秒

校验和和和和**Oracle**数据库完整性

ONTAP及其支持的协议包括多项功能、可保护Oracle数据库完整性、包括空闲数据和通过网络传输的数据。

ONTAP中的逻辑数据保护包括三个关键要求：

- 必须防止数据损坏。

- 必须保护数据免受驱动器故障的影响。
- 必须防止对数据所做的更改丢失。

以下各节将讨论这三种需求。

网络损坏：校验和

最基本的数据保护级别是校验和、校验和是随数据一起存储的一种特殊错误检测代码。使用校验和(在某些情况下、使用多个校验和)检测网络传输期间的数据损坏。

例如、FC帧包含一种称为循环冗余校验(CRC)的校验和形式、用于确保有效负载在传输过程中不会损坏。发射器会同时发送数据和数据的CRC。FC帧的接收器重新计算已接收数据的CRC、以确保其与已传输的CRC匹配。如果新计算的CRC与附加到帧的CRC不匹配、则数据将损坏、FC帧将被丢弃或拒绝。iSCSI I/O操作包括TCP/IP和以太网层的校验和、并且为了提供额外保护、还可以在SCSI层提供可选的CRC保护。TCP层或IP层会检测到线路上的任何位损坏、从而导致数据包重新传输。与FC一样、SCSI CRC中的错误会导致丢弃或拒绝操作。

驱动器损坏：校验和

校验和还用于验证存储在驱动器上的数据的完整性。写入驱动器的数据块使用校验和功能进行存储、该功能会产生与原始数据相关的不可预测的数字。从驱动器中读取数据时、将重新计算校验和并将其与存储的校验和进行比较。如果不匹配、则数据已损坏、必须由RAID层进行恢复。

数据损坏：写入丢失

最难检测的损坏类型之一是写入丢失或放错位置。确认写入后、必须将其写入到正确位置的介质中。通过使用随数据存储的简单校验和、可以相对容易地检测原位数据损坏。但是、如果只是写入丢失、则先前版本的数据可能仍存在、校验和将是正确的。如果将写入放置在错误的物理位置、则关联的校验和将再次对存储的数据有效、即使写入操作已销毁其他数据。

应对此挑战的解决方案如下：

- 写入操作必须包含元数据、用于指示预期写入位置。
- 写入操作必须包含某种版本标识符。

当ONTAP写入块时、它会包含有关块所属位置的数据。如果后续读取发现某个块、但在位置456发现元数据时、元数据指示该块属于位置123、则表示该写入已放错位置。

检测完全丢失的写入操作会更加困难。解释非常复杂、但从本质上说、ONTAP存储元数据的方式是、写入操作会导致更新到驱动器上的两个不同位置。如果写入丢失、则后续读取的数据和关联元数据将显示两个不同的版本标识。这表示驱动器未完成写入。

丢失和放错位置的写入损坏极为少见、但随着驱动器不断增长、数据集逐渐扩展到EB级、风险也会增加。支持数据库工作负载的任何存储系统都应包括失写检测。

驱动器故障：**RAID**、**RAID DP**和**RAID-TEC**

如果发现驱动器上的数据块已损坏、或者整个驱动器发生故障且完全不可用、则必须重新生成数据。这在ONTAP中通过使用奇偶校验驱动器来实现。数据在多个数据驱动器之间进行条带化、然后生成奇偶校验数据。该数据与原始数据分开存储。

ONTAP最初使用的是RAID 4、该RAID 4会为每组数据驱动器使用一个奇偶校验驱动器。这样、组中的任何一个驱动器都可能发生故障、而不会导致数据丢失。如果奇偶校验驱动器发生故障、则不会损坏任何数据、可以构建

新的奇偶校验驱动器。如果一个数据驱动器发生故障、则其余驱动器可与奇偶校验驱动器结合使用来重新生成缺失的数据。

如果驱动器较小、则两个驱动器同时发生故障的统计几率可以忽略不计。随着驱动器容量的增长、在驱动器发生故障后重建数据所需的时间也会相应增加。这增加了第二个驱动器故障导致数据丢失的时间范围。此外、重建过程会在无故障驱动器上创建大量额外的I/O。随着驱动器老化、导致第二个驱动器故障的额外负载风险也会增加。最后、即使持续使用RAID 4不会增加数据丢失的风险、数据丢失的后果也会更加严重。RAID组发生故障时丢失的数据越多、恢复数据所需的时间就越长、从而延长业务中断时间。

这些问题促使NetApp开发了NetApp RAID DP技术、这是RAID 6的变体。此解决方案包含两个奇偶校验驱动器、这意味着RAID组中的任何两个驱动器都可能发生故障、而不会造成数据丢失。驱动器的大小持续增长、这最终导致NetApp开发了NetApp RAID-TEC技术、该技术引入了第三个奇偶校验驱动器。

一些历史数据库最佳实践建议使用RAID-10、也称为条带化镜像。这提供的数据保护比RAID DP更少、因为存在多种双磁盘故障情形、而在RAID DP中则没有。

还有一些历史数据库最佳实践表明、出于性能考虑、RAID-10优于RAID-4/5/6选项。这些建议有时会提及RAID惩罚。虽然这些建议通常是正确的、但不适用于在ONTAP中实施RAID。性能问题与奇偶校验重新生成有关。在传统RAID实施中、处理数据库执行的例行随机写入需要多次磁盘读取才能重新生成奇偶校验数据并完成写入。惩罚定义为执行写入操作所需的额外读取IOPS。

ONTAP不会产生RAID惩罚、因为写入会暂存到内存中、在该内存中会生成奇偶校验、然后作为单个RAID条带写入磁盘。完成写入操作不需要执行任何读取操作。

总之、与RAID 10相比、RAID DP和RAID-TEC可提供更多的可用容量、更好地防止驱动器故障、并且不会影响性能。

硬件故障保护：**NVRAM**

任何为数据库工作负载提供服务的存储阵列都必须尽快为写入操作提供服务。此外、必须保护写入操作、使其不会因意外事件(例如断电)而丢失。这意味着任何写入操作都必须安全地存储在至少两个位置。

AFF和FAS系统依靠NVRAM来满足这些要求。写入过程的工作原理如下：

1. 入站写入数据存储在RAM中。
2. 必须对磁盘上的数据所做的更改会记录到本地节点和配对节点上的NVRAM中。NVRAM不是写入缓存、而是类似于数据库重做日志的日志。在正常情况下、不会读取它。它仅用于恢复、例如在I/O处理期间发生电源故障后。
3. 然后、写入操作会向主机确认。

从应用程序角度来看、此阶段的写入过程已完成、数据会受到保护、不会丢失、因为数据会存储在两个不同的位置。更改最终会写入磁盘、但从应用程序角度来看、此过程是带外过程、因为它发生在确认写入之后、因此不会影响延迟。此过程再次类似于数据库日志记录。对数据库所做的更改会尽快记录在重做日志中、然后确认已提交更改。数据文件的更新发生得更晚、不会直接影响处理速度。

如果某个控制器发生故障、配对控制器将接管所需磁盘的所有权、并在NVRAM中回显已记录的数据、以恢复发生故障时正在进行的任何I/O操作。

硬件故障保护：**NVFAIL**

如前文所述、写入操作只有在至少另一个控制器上记录到本地NVRAM和NVRAM后才会得到确认。此方法可确保硬件故障或断电不会导致传输中I/O丢失如果本地NVRAM发生故障或与HA配对节点的连接发生故障、则不会

再镜像此传输中的数据。

如果本地NVRAM报告错误、则此节点将关闭。此关闭会导致故障转移到HA配对控制器。由于发生故障的控制器尚未确认写入操作、因此不会丢失任何数据。

除非强制执行故障转移、否则ONTAP不允许在数据不同步时进行故障转移。以这种方式强制更改条件即表示数据可能会留在原始控制器中、并且数据丢失是可以接受的。

如果强制执行故障转移、数据库尤其容易受到损坏的影响、因为数据库会在磁盘上保留大量内部数据缓存。如果发生强制故障转移、先前确认的更改将被有效丢弃。存储阵列的内容会及时有效地向后跳转、数据库缓存的状态不再反映磁盘上数据的状态。

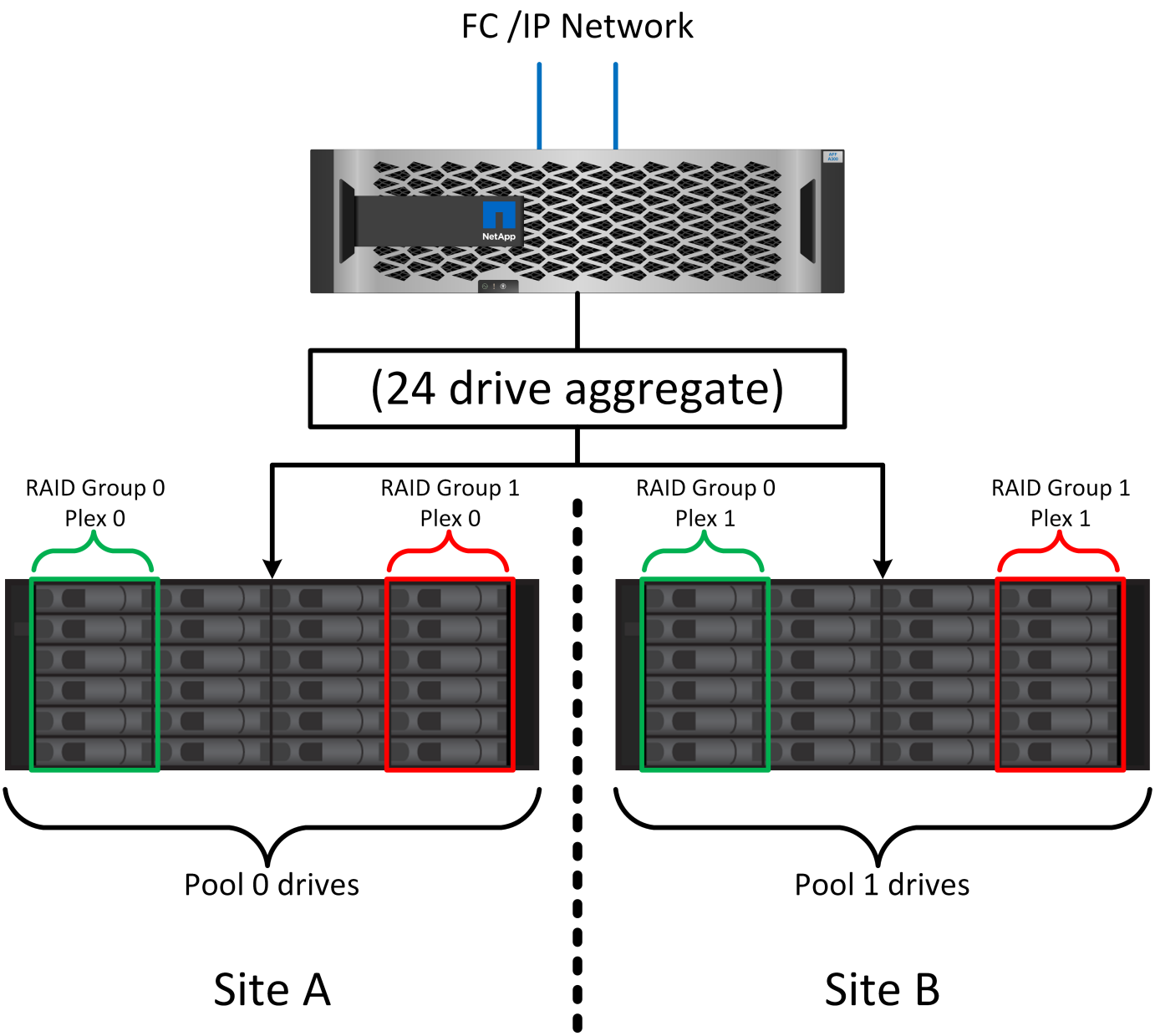
为了保护数据免受这种情况的影响、ONTAP允许对卷进行配置、以便针对NVRAM故障提供特殊保护。触发此保护机制后、卷将进入名为NVFAIL的状态。此状态会导致I/O错误、即发生原因A应用程序会关闭、以使其不使用陈旧数据。数据不应丢失、因为存储阵列上应存在任何已确认的写入。

通常的后续步骤是、管理员先完全关闭主机、然后再手动将LUN和卷重新联机。虽然这些步骤可能涉及一些工作、但这种方法是确保数据完整性的最安全方法。并非所有数据都需要这种保护、这就是可以逐个卷配置NVFAIL行为的原因。

站点和磁盘架故障保护：**SyncMirror**和**plexes**

SyncMirror是一种镜像技术、可增强但不会取代RAID DP或RAID-TEC。它会镜像两个独立RAID组的内容。逻辑配置如下：

- 驱动器会根据位置配置到两个池中。一个池由站点A上的所有驱动器组成、另一个池由站点B上的所有驱动器组成
- 然后、基于RAID组的镜像集创建一个通用存储池(称为聚合)。从每个站点提取的驱动器数量相等。例如、一个包含20个驱动器的SyncMirror聚合将由站点A的10个驱动器和站点B的10个驱动器组成
- 给定站点上的每组驱动器都会自动配置为一个或多个完全冗余的RAID-DP或RAID-TEC组、而与镜像的使用无关。这样可以提供持续的数据保护、即使在站点丢失后也是如此。



上图显示了一个示例SyncMirror配置。在控制器上创建了一个包含24个驱动器的聚合、其中12个驱动器来自站点A上分配的磁盘架、12个驱动器来自站点B上分配的磁盘架这些驱动器被分组为两个镜像RAID组。RAID组0在站点A上包含一个6驱动器丛、该丛镜像到站点B上的6驱动器丛同样、RAID组1在站点A上包含一个6驱动器丛、该丛镜像到站点B上的6驱动器丛

SyncMirror通常用于为MetroCluster系统提供远程镜像、每个站点有一个数据副本。有时、它会用于在单个系统中提供额外的冗余级别。尤其是、它可以提供磁盘架级冗余。驱动器架已包含双电源和控制器、总体比金属板稍多、但在某些情况下、可能需要额外保护。例如、一家NetApp客户为汽车测试期间使用的移动实时分析平台部署了SyncMirror。该系统分为两个物理机架、由独立UPS系统的独立电源供电。

=校验和

对于习惯于使用Oracle RMAN流式备份并迁移到基于快照的备份的数据库用户来说、校验和主题特别重要。RMAN的一项功能是、它会在备份操作期间执行完整性检查。尽管此功能具有一定的价值、但其主要优势是用于未在现代存储阵列上使用的数据库。将物理驱动器用于Oracle数据库时、几乎可以肯定、随着驱动器老化、最终会发生损坏、而在真正的存储阵列中、基于阵列的校验和可以解决这一问题。

对于真正的存储阵列、数据完整性可通过在多个级别使用校验和来保护。如果基于IP的网络中的数据损坏、则传输控制协议(TCP)层会拒绝数据包数据并请求重新传输。FC协议包括校验和、封装的SCSI数据也是如此。将ONTAP置于阵列上后、它将具有RAID和校验和保护功能。可能会发生损坏、但与大多数企业阵列一样、系统会检测到并更正此问题。通常、整个驱动器发生故障、提示重建RAID、数据库完整性不受影响。ONTAP检测校验和错误的频率较低、这意味着驱动器上的数据已损坏。然后、驱动器将出现故障、并开始RAID重建。同样、数据完整性也不受影响。

Oracle数据文件和重做日志架构还旨在提供尽可能高级别的数据完整性、即使在极端情况下也是如此。在最基本的层面上、Oracle块包括对几乎每个I/O进行校验和和和基本逻辑检查如果Oracle未崩溃或使表空间脱机、则数据完好无损。数据完整性检查的程度可以调整、Oracle也可以配置为确认写入。因此、几乎所有崩溃和故障情形都可以恢复、在极少数情况下发生不可恢复的情况时、系统会立即检测到损坏。

大多数使用Oracle数据库的NetApp客户在迁移到基于快照的备份之后不再使用RMAN和其他备份产品。在使用SnapCenter执行块级恢复时、仍然可以使用RMAN。但是、在日常工作中、RMAN、NetBackup和其他产品仅偶尔用于创建每月或每季度归档副本。

有些客户选择运行 `dbv` 定期对其现有数据库执行完整性检查。NetApp不建议采用这种做法、因为它会产生不必要的I/O负载。如上所述、如果数据库之前未遇到问题、则可能会出现 `dbv` 检测问题几乎为零、此实用程序会在网络和存储系统上创建非常高的顺序I/O负载。除非有理由认为存在损坏、例如暴露于已知的Oracle错误、否则没有理由运行 `dbv`。

备份和恢复基础知识

Oracle数据库和基于快照的备份

基于ONTAP的Oracle数据库数据保护的基础是NetApp Snapshot技术。

关键值如下：

- *精简性。*快照是指特定时间点数据容器内容的只读副本。
- *效率。*创建快照时不需要任何空间。只有在数据发生更改时才会占用空间。
- *易管理性。*基于快照的备份策略易于配置和管理、因为快照是存储操作系统的本机部分。如果存储系统已启动、则它可以随时创建备份。
- *可扩展性。*一个文件和LUN容器最多可保留1024个备份。对于复杂的数据集、可以通过一组一致的快照来保护多个数据容器。
- 无论卷包含1024个快照还是无快照、性能都不受影响。

虽然许多存储供应商都提供快照技术、但ONTAP中的快照技术是独一无二的、可为企业级应用程序和数据库环境带来显著优势：

- Snapshot副本是底层任意位置写入文件布局(Write-Anywhere File Layout、WAFL)的一部分。它们不是附加技术或外部技术。由于存储系统是备份系统、因此可简化管理。
- Snapshot副本不会影响性能、但某些边缘情形除外、例如、快照中存储的数据量如此之多、以致于底层存储系统会填满。
- 术语"一致性组"通常用于指作为一致的数据集合进行管理的一组存储对象。特定ONTAP卷的快照构成一致性组备份。

ONTAP快照的扩展能力也优于竞争技术。客户可以存储5个、50个或500个快照、而不会影响性能。卷中当前允许的最大快照数为1024。如果需要额外保留快照、可以选择将快照级联到其他卷。

因此、保护ONTAP上托管的数据集非常简单、并且具有高度可扩展性。备份不需要移动数据、因此可以根据业务需求定制备份策略、而不是网络传输速率、大量磁带驱动器或磁盘暂存区的限制。

快照是否为备份?

有关将快照用作数据保护策略的一个常见问题是、"实际"数据和快照数据位于同一个驱动器上。丢失这些驱动器将导致主数据和备份均丢失。

这是一个合理的问题。本地快照用于满足日常备份和恢复需求、在这方面、快照是备份。在NetApp环境中、几乎99%的恢复方案都依靠快照来满足最苛刻的恢复时间要求。

但是、本地快照绝不是唯一的备份策略、这就是NetApp提供SnapMirror和SnapVault复制等技术来快速高效地将快照复制到一组独立驱动器的原因。在采用快照和快照复制功能且架构合理的解决方案中、磁带的使用量可以降至最低、甚至可以每季度归档一次、也可以完全避免。

基于**Snapshot**的备份

使用ONTAP Snapshot副本保护数据有多种选择、快照是复制、灾难恢复和克隆等许多其他ONTAP功能的基础。本文档不会介绍有关Snapshot技术的完整问题描述、但以下各节将提供一般概述。

创建数据集快照的主要方法有两种：

- 崩溃状态一致的备份
- 应用程序一致的备份

崩溃状态一致的数据集备份是指在一个时间点捕获整个数据集结构。如果数据集存储在单个NetApp FlexVol卷中、则此过程非常简单；可以随时创建Snapshot。如果数据集跨越多个卷、则必须创建一致性组(CG)快照。创建CG快照的选项有多种、包括NetApp SnapCenter软件、本机ONTAP一致性组功能以及用户维护的脚本。

当备份点恢复足以满足要求时、主要使用崩溃状态一致的备份。当需要更精细的恢复时、通常需要应用程序一致的备份。

"应用程序一致"中的"一致"一词通常用词不当。例如、将Oracle数据库置于备份模式称为应用程序一致的备份、但数据不会以任何方式保持一致或处于静态。数据在整个备份过程中持续更改。相比之下、大多数MySQL和Microsoft SQL Server备份确实会在执行备份之前将数据置于静噪状态。VMware可能会使某些文件保持一致、也可能不会使其保持一致。

一致性组

术语"一致性组"是指存储阵列能够将多个存储资源作为一个映像进行管理。例如、一个数据库可能包含10个LUN。阵列必须能够以一致的方式备份、还原和复制这10个LUN。如果LUN的映像在备份时不一致、则无法还原。复制这10个LUN要求所有副本之间完全同步。

在讨论ONTAP时、不经常使用术语"一致性组"、因为一致性一直是ONTAP中卷和聚合架构的基本功能。许多其他存储阵列将LUN或文件系统作为单独的单元进行管理。然后、可以选择将其配置为"一致性组"以实现数据保护、但这是配置中的一个额外步骤。

ONTAP始终能够捕获一致的本地和复制数据映像。虽然ONTAP系统上的各种卷通常不会正式描述为一致性组、但它们就是一致性组。该卷的快照是一致性组映像、该快照的还原是一致性组还原、SnapMirror和SnapVault均提供一致性组复制。

一致性组快照

一致性组快照(CG快照)是基本ONTAP快照技术的扩展。标准快照操作会为单个卷中的所有数据创建一致的映像、但有时需要在多个卷甚至多个存储系统之间创建一组一致的快照。这样就会生成一组快照、这些快照的使用方式与仅包含一个卷的快照相同。它们可用于本地数据恢复、为灾难恢复目的进行复制或作为一个一致的单元进行克隆。

已知的最大CG-Snapsh图 用途是用于大小约为1 PB且跨越12个控制器的数据库环境。在此系统上创建的CG快照已用于备份、恢复和克隆。

大多数情况下、如果数据集跨越多个卷且必须保留写入顺序、则选定管理软件会自动使用CG快照。在这种情况下、无需了解CG快照的技术详细信息。但是、在某些情况下、复杂的数据保护要求需要对数据保护和复制过程进行详细控制。可以选择自动化工作流或使用自定义脚本来调用CG-Snapshot API。要了解cG-Snapshot的最佳选项和角色、需要对该技术进行更详细的说明。

创建一组CG快照的过程分为两步：

1. 在所有目标卷上建立写入隔离。
2. 在隔离状态下创建这些卷的快照。

写入隔离是按序列建立的。这意味着、在多个卷之间设置隔离过程时、写入I/O会冻结在序列中的第一个卷上、因为它会继续提交到稍后显示的卷。最初、这可能看起来违反了保留写入顺序的要求、但只有在主机上异步发出的适用场景I/O、而不依赖于任何其他写入。

例如、数据库可能会对大量异步数据文件更新进行问题描述、并允许操作系统重新排列I/O、然后根据自己的计划程序配置完成这些更新。无法保证此类I/O的顺序、因为应用程序和操作系统已释放保留写入顺序的要求。

作为一个计数器示例、大多数数据库日志记录活动都是同步的。在确认I/O并保留这些写入顺序之前、数据库不会继续进行日志写入。如果日志I/O到达隔离的卷、则不会进行确认、应用程序会阻止进一步写入。同样、文件系统元数据I/O通常是同步的。例如、文件删除操作不能丢失。如果带有xfs文件系统的操作系统删除了某个文件、而更新了xfs文件系统元数据以删除对该文件的引用的I/O则会登录到隔离的卷上、则文件系统活动将暂停。这样可以保证CG快照操作期间文件系统的完整性。

在目标卷之间设置写入隔离后、这些卷便可创建快照了。无需同时创建快照、因为从依赖写入的角度来看、卷的状态是冻结的。为了防止创建CG快照的应用程序出现缺陷、初始写入隔离包括一个可配置的超时时间、在此超时时间内、ONTAP会自动释放隔离并在定义的秒数后恢复写入处理。如果所有快照都是在超时期限到期之前创建的、则生成的一组快照是有效的一致性组。

从属写入顺序

从技术角度来看、一致性组的关键在于保留写入顺序、尤其是依赖写入顺序。例如、向10个LUN写入数据的数据库会同时向所有LUN写入数据。许多写入操作是异步发出的、这意味着它们的完成顺序并不重要、实际完成顺序会因操作系统和网络行为而异。

在数据库继续执行其他写入操作之前、磁盘上必须存在某些写入操作。这些关键写入操作称为依赖写入。后续写入I/O取决于磁盘上是否存在这些写入。对这10个LUN执行任何快照、恢复或复制操作都必须确保依赖写入顺序得到保证。文件系统更新是依赖写入顺序写入的另一个示例。必须保留文件系统更改的顺序、否则整个文件系统可能会损坏。

战略

基于快照的备份有两种主要方法：

- 崩溃状态一致的备份
- 受Snapshot保护的热备份

崩溃状态一致的数据库备份是指在一个时间点捕获整个数据库结构、包括数据文件、重做日志和控制文件。如果数据库存储在单个NetApp FlexVol卷中、则此过程非常简单；可以随时创建Snapshot。如果数据库跨越多个卷、则必须创建一致性组(CG)快照。创建CG快照的选项有多种、包括NetApp SnapCenter软件、本机ONTAP一致性组功能以及用户维护的脚本。

崩溃状态一致的Snapshot备份主要在备份点恢复已足够时使用。在某些情况下、可以应用归档日志、但在需要更精细的时间点恢复时、最好使用联机备份。

基于快照的联机备份的基本操作步骤如下所示：

1. 将数据库放置在中 `backup` 模式。
2. 为托管数据文件的所有卷创建快照。
3. 退出 `backup` 模式。
4. 运行命令 `alter system archive log current` 强制日志归档。
5. 为托管归档日志的所有卷创建快照。

此操作步骤将生成一组快照、其中包含处于备份模式的数据文件以及处于备份模式时生成的关键归档日志。这是恢复数据库的两项要求。为方便起见、还应保护控制文件等文件、但唯一的绝对要求是保护数据文件和归档日志。

虽然不同的客户可能有非常不同的策略、但几乎所有这些策略最终都基于下面所述的相同原则。

基于**Snapshot**的恢复

在为Oracle数据库设计卷布局时、首先要决定是否使用基于卷的NetApp SnapRestore (VBSR)技术。

基于卷的SnapRestore可以将卷几乎即时还原到较早的时间点。由于卷上的所有数据均已还原、因此VBSR可能并不适用于所有使用情形。例如、如果整个数据库(包括数据文件、重做日志和归档日志)存储在单个卷上、而此卷通过VBSR还原、则数据会丢失、因为较新的归档日志和重做数据会被丢弃。

还原不需要VBSR。许多数据库都可以通过使用基于文件的单文件文件系统(Single File SnapRestore、SFSR)进行还原、或者只需将文件从快照复制回活动文件系统即可。

当数据库非常大或必须尽快恢复时、最好使用VBSR、而使用VBSR需要隔离数据文件。在NFS环境中、给定数据库的数据文件必须存储在未受任何其他类型文件污染的专用卷中。在SAN环境中、数据文件必须存储在专用FlexVol卷上的专用LUN中。如果使用卷管理器(包括Oracle自动存储管理[ASM])、则磁盘组还必须专用于数据文件。

通过以这种方式隔离数据文件、可以将其还原到早期状态、而不会损坏其他文件系统。

Snapshot 预留

对于SAN环境中包含Oracle数据的每个卷、 `percent-snapshot-space` 应设置为零、因为在LUN环境中为快照预留空间没有用处。如果预留百分比设置为100、则包含LUN的卷的快照需要该卷中具有足够的可用空间(不包括快照预留)来吸收所有数据的100%周转率。如果预留百分比设置为较低的值、则所需的可用空间量相应较少、但始终不包括Snapshot预留。这意味着会浪费LUN环境中的快照预留空间。

在NFS环境中、有两种选择：

- 设置 `percent-snapshot-space` 基于预期的Snapshot空间消耗。
- 设置 `percent-snapshot-space` 将活动空间和快照空间占用情况统一置零并进行管理。

使用第一个选项时、 `percent-snapshot-space` 设置为非零值、通常约为20%。然后、此空间将对用户隐藏。但是、此值不会对利用率造成限制。如果预留百分比为20%的数据库的周转率为30%、则快照空间可能会超出预留百分比的界限并占用未预留空间。

将预留设置为20%这样的值的主要优势是、验证某些空间始终可用于快照。例如、预留为20%的1 TB卷仅允许数据库管理员(Database Administrator、DBA)存储800 GB数据。此配置可确保至少为快照占用200 GB的空间。

时间 `percent-snapshot-space` 设置为零时、卷中的所有空间均可供最终用户使用、从而提高可见性。数据库管理员必须了解、如果发现1 TB卷利用快照、则这1 TB空间将在活动数据和Snapshot周转率之间共享。

最终用户之间没有明确的首选方案一和备选方案二。

ONTAP和第三方快照

Oracle文档ID 604683.1介绍了第三方快照支持的要求以及可用于备份和还原操作的多个选项。

第三方供应商必须保证公司的快照符合以下要求：

- 快照必须与Oracle建议的还原和恢复操作集成。
- 快照必须在快照点保持数据库崩溃状态一致。
- 系统会为快照中的每个文件保留写入顺序。

ONTAP和NetApp Oracle管理产品符合这些要求。

借助SnapRestore快速恢复Oracle数据库

NetApp SnapRestore技术可在ONTAP中从快照快速恢复数据。

当关键数据集不可用时、关键业务运营将中断。磁带可能会中断、甚至从基于磁盘的备份中恢复的速度也可能很慢、无法通过网络传输。SnapRestore通过近乎即时地还原数据集来避免这些问题。即使是PB级数据库、也只需几分钟的时间即可完全还原。

SnapRestore有两种形式：基于文件/LUN和基于卷。

- 单个文件或LUN可以在几秒钟内还原、无论它是2 TB LUN还是4 KB文件。
- 文件或LUN容器可以在几秒钟内还原、无论数据大小是10 GB还是100 TB。

"文件或LUN容器"通常指FlexVol卷。例如、一个卷中可能有10个LUN构成一个LVM磁盘组、或者一个卷可能会存储包含1000个用户的NFS主目录。您可以将整个卷作为一个操作来还原、而不是对每个文件或LUN执行还原操作。此过程还适用于包含多个卷的横向扩展容器、例如FlexGroup或ONTAP一致性组。

SnapRestore之所以能够如此快速高效地工作、是因为快照的性质、从本质上说、快照是一个在特定时间点卷内容的并行只读视图。活动块是可以更改的实际块、而快照是创建快照时构成文件和LUN的块的状态的只读视图。

ONTAP仅允许对快照数据进行只读访问、但可以使用SnapRestore重新激活这些数据。快照将重新启用为数据的读写视图、从而将数据恢复到先前的状态。SnapRestore可以在卷或文件级别运行。该技术本质上是相同的、

但行为略有不同。

Volume SnapRestore

基于卷的SnapRestore会将整个数据卷返回到先前的状态。此操作不需要移动数据、这意味着还原过程基本上是瞬时的、尽管处理API或CLI操作可能需要几秒钟时间。还原1 GB的数据并不比还原1 PB的数据更复杂、也不会更耗时。这一功能是许多企业客户迁移到ONTAP存储系统的主要原因。即使是最大的数据集、它也能以秒为单位提供一个RTO。

基于卷的SnapRestore的一个缺点是、卷内的更改会随着时间的推移而累积。因此、每个快照和活动文件数据都取决于到那时为止所做的更改。将卷还原到早期状态意味着、系统将会先对数据进行所有后续更改、然后再进行相应的更改。但是、不太明显的是、这包括随后创建的快照。这并不总是可取的。

例如、数据保留SLA可能指定30天的夜间备份。如果将数据集还原到五天前使用卷SnapRestore创建的快照、则会丢弃前五天创建的所有快照、从而违反SLA。

有许多选项可用于解决此限制：

1. 可以从先前的快照复制数据、而不是对整个卷执行SnapRestore。此方法最适合较小的数据集。
2. 快照可以克隆、而不是还原。此方法的限制是、源快照是克隆的依赖项。因此、除非同时删除克隆或将其拆分成独立的卷、否则无法将其删除。
3. 使用基于文件的SnapRestore。

File SnapRestore

基于文件的SnapRestore是一种基于快照的更精细还原过程。系统会还原单个文件或LUN的状态、而不是还原整个卷的状态。无需删除任何快照、此操作也不会对先前的快照创建任何依赖关系。文件或LUN将立即在活动卷中可用。

在对文件或LUN执行SnapRestore还原期间、不需要移动数据。但是、需要进行一些内部元数据更新、以反映文件或LUN中的底层块现在同时位于快照和活动卷中这一事实。此过程不会对性能产生任何影响、但会阻止创建快照、直到创建完成为止。根据所还原文件的总大小、处理速率约为5 Gbps (18 TB/小时)。

Oracle数据库联机备份

在备份模式下保护和恢复Oracle数据库需要两组数据。请注意、这不是唯一的Oracle备份选项、但最常见。

- 备份模式下数据文件的快照
- 数据文件处于备份模式时创建的归档日志

如果需要完全恢复(包括所有已提交的事务)、则需要第三项：

- 一组当前的重做日志

可以通过多种方法恢复联机备份。许多客户使用ONTAP命令行界面还原快照、然后使用Oracle RMAN或sqlplus完成恢复。在大型生产环境中、这种情况尤为常见、在这些环境中、数据库还原的概率和频率极低、任何还原操作步骤都由技能娴熟的数据库管理人员来处理。为了实现完全自动化、NetApp SnapCenter等解决方案包括一个具有命令行和图形界面的Oracle插件。

一些大型客户采用了一种更简单的方法、即在主机上配置基本脚本、以便在特定时间将数据库置于备份模式、以

便为计划的快照做准备。例如、计划命令 `alter database begin backup` 23：58时、 `alter database end backup` 00：02、然后将快照直接计划在午夜在存储系统上。这样、便形成了一个简单、高度可扩展的备份策略、无需外部软件或许可证。

数据布局

最简单的布局是将数据文件隔离到一个或多个专用卷中。它们必须未受任何其他文件类型的污染。这是为了确保数据文件卷可以通过SnapRestore操作快速还原、而不会销毁重要的重做日志、控制文件或归档日志。

SAN对专用卷中的数据文件隔离具有类似要求。对于Microsoft Windows等操作系统、一个卷可能包含多个数据文件LUN、每个LUN都具有一个NTFS文件系统。对于其他操作系统、通常会有一个逻辑卷管理器。例如、对于Oracle ASM、最简单的选择是将ASM磁盘组的LUN限制为一个可作为一个单元进行备份和还原的卷。如果出于性能或容量管理原因需要更多卷、则在新卷上创建更多磁盘组可简化管理。

如果遵循这些准则、则可以直接在存储系统上计划快照、而无需执行一致性组快照。原因是Oracle备份不需要同时备份数据文件。联机备份操作步骤旨在使数据文件能够持续更新、因为它们会在数小时内缓慢流式传输到磁带。

如果使用分布在卷之间的ASM磁盘组、则会出现复杂情况。在这些情况下、必须执行cG-Snapshot、以确保ASM元数据在所有成分卷之间保持一致。

*注意：*验证ASM `spfile` 和 `passwd` 文件不在托管数据文件的磁盘组中。这会影响有选择地还原数据文件和仅还原数据文件的能力。

本地恢复过程**—NFS**

此操作步骤可以手动驱动、也可以通过SnapCenter等应用程序驱动。基本操作步骤如下所示：

1. 关闭数据库。
2. 将数据文件卷恢复到所需还原点之前的快照。
3. 将归档日志重放至所需位置。
4. 如果需要完全恢复、则重放当前重做日志。

此操作步骤假定所需的归档日志仍存在于活动文件系统中。否则、必须还原归档日志、或者可以将RMAN/sqlplus定向到快照目录中的数据。

此外、对于较小的数据库、最终用户可以直接从中恢复数据文件 `.snapshot` 目录、而无需自动化工具或存储管理员协助即可执行 `snaprestore` 命令：

本地恢复过程**—SAN**

此操作步骤可以手动驱动、也可以通过SnapCenter等应用程序驱动。基本操作步骤如下所示：

1. 关闭数据库。
2. 将托管数据文件的磁盘组静置。操作步骤因所选的逻辑卷管理器而异。使用ASM时、此过程需要卸载磁盘组。对于Linux、必须卸载文件系统、并且必须停用逻辑卷和卷组。目标是停止要还原的目标卷组上的所有更新。
3. 将数据文件磁盘组还原到所需还原点之前的快照。
4. 重新激活新还原的磁盘组。

5. 将归档日志重放至所需位置。
6. 如果需要完全恢复、请重放所有重做日志。

此操作步骤假定所需的归档日志仍存在于活动文件系统中。否则、必须通过使归档日志LUN脱机并执行还原来还原归档日志。这也是一个将归档日志划分为专用卷非常有用的示例。如果归档日志与重做日志共享一个卷组、则必须先将重做日志复制到其他位置、然后才能还原整个一组LUN。此步骤可防止丢失这些最终记录的事务。

Oracle数据库存储**Snapshot**优化备份

在Oracle 12c发布后、基于Snapshot的备份和恢复变得更加简单、因为无需将数据库置于热备份模式。因此、可以直接在存储系统上计划基于快照的备份、同时仍保留执行完整或时间点恢复的能力。

尽管数据库管理器(操作步骤)对数据库管理器(数据库管理器)比较熟悉、但长期以来、可以使用数据库处于热备份模式时未创建的快照。在恢复期间、需要对Oracle 10g和11g执行额外的手动步骤、才能使数据库保持一致。采用Oracle 12c、 `sqlplus` 和 `rman` 包含额外的逻辑、用于重放未处于热备份模式的数据文件备份上的归档日志。

如前文所述、恢复基于快照的热备份需要两组数据：

- 在备份模式下创建的数据文件的快照
- 数据文件处于热备份模式时生成的归档日志

在恢复期间、数据库会从数据文件读取元数据、以选择恢复所需的归档日志。

经过存储快照优化的恢复需要略有不同的数据集才能实现相同的结果：

- 数据文件的快照、以及用于标识快照创建时间的方法
- 从最近的数据文件检查点到快照的确切时间的归档日志

在恢复期间、数据库会从数据文件中读取元数据、以确定所需的最早归档日志。可以执行完全恢复或时间点恢复。执行时间点恢复时、了解数据文件快照的时间至关重要。指定恢复点必须在快照创建时间之后。NetApp建议为快照时间至少添加几分钟、以考虑时钟变化。

有关完整的详细信息、请参见Oracle 12c文档各个版本中有关"使用存储Snapshot优化进行恢复"主题的Oracle文档。另请参见Oracle文档ID 604683.1、了解有关Oracle第三方快照支持的信息。

数据布局

最简单的布局是将数据文件隔离到一个或多个专用卷中。它们必须未受任何其他文件类型的污染。这是为了确保数据文件卷可以通过SnapRestore操作快速还原、而不会销毁重要的重做日志、控制文件或归档日志。

SAN对专用卷中的数据文件隔离具有类似要求。对于Microsoft Windows等操作系统、一个卷可能包含多个数据文件LUN、每个LUN都具有一个NTFS文件系统。对于其他操作系统、通常也会有一个逻辑卷管理器。例如、对于Oracle ASM、最简单的选择是将磁盘组限制为一个卷、该卷可以作为一个单元进行备份和还原。如果出于性能或容量管理原因需要更多卷、则在新卷上创建更多磁盘组可简化管理。

如果遵循这些准则、则可以直接在ONTAP上计划快照、而无需执行一致性组快照。原因是针对快照优化的备份不需要同时备份数据文件。

如果ASM磁盘组分布在多个卷中、则会出现复杂情况。在这些情况下、必须执行cG-Snapshot、以确保ASM元

数据在所有成分卷之间保持一致。

[注]验证ASM spfile和passwd文件是否不在托管数据文件的磁盘组中。这会影响有选择地还原数据文件和仅还原数据文件的能力。

本地恢复过程—NFS

此操作步骤可以手动驱动、也可以通过SnapCenter等应用程序驱动。基本操作步骤如下所示：

1. 关闭数据库。
2. 将数据文件卷恢复到所需还原点之前的快照。
3. 将归档日志重放至所需位置。

此操作步骤假定所需的归档日志仍存在于活动文件系统中。否则、必须还原归档日志、或 `rman` 或 `sqlplus` 可以定向到中的数据 `.snapshot` 目录。

此外、对于较小的数据库、最终用户可以直接从中恢复数据文件 `.snapshot` 目录、而无需借助自动化工具或存储管理员来执行SnapRestore命令。

本地恢复过程—SAN

此操作步骤可以手动驱动、也可以通过SnapCenter等应用程序驱动。基本操作步骤如下所示：

1. 关闭数据库。
2. 将托管数据文件的磁盘组静置。操作步骤因所选的逻辑卷管理器而异。使用ASM时、此过程需要卸载磁盘组。对于Linux、必须卸载文件系统、并停用逻辑卷和卷组。目标是停止要还原的目标卷组上的所有更新。
3. 将数据文件磁盘组还原到所需还原点之前的快照。
4. 重新激活新还原的磁盘组。
5. 将归档日志重放至所需位置。

此操作步骤假定所需的归档日志仍存在于活动文件系统中。否则、必须通过使归档日志LUN脱机并执行还原来还原归档日志。这也是一个将归档日志划分为专用卷非常有用的示例。如果归档日志与重做日志共享一个卷组、则必须在还原整个LUN集之前将重做日志复制到其他位置、以避免丢失最终记录的事务。

完全恢复示例

假设数据文件已损坏或销毁、需要完全恢复。要执行此操作的操作步骤如下所示：

```
[oracle@host1 ~]$ sqlplus / as sysdba
Connected to an idle instance.
SQL> startup mount;
ORACLE instance started.
Total System Global Area 1610612736 bytes
Fixed Size                  2924928 bytes
Variable Size            1040191104 bytes
Database Buffers          553648128 bytes
Redo Buffers               13848576 bytes
Database mounted.
SQL> recover automatic;
Media recovery complete.
SQL> alter database open;
Database altered.
SQL>
```

时间点恢复示例

整个恢复操作步骤只需一个命令： `recover automatic`。

如果需要时间点恢复、则快照的时间戳必须已知、并且可按如下方式进行标识：

```
Cluster01::> snapshot show -vserver vserver1 -volume NTAP_oradata -fields
create-time
vserver   volume        snapshot   create-time
--------  ------------  ---------  -------------------------
vserver1  NTAP_oradata  my-backup  Thu Mar 09 10:10:06 2017
```

快照创建时间显示为3月9日和10：10：06。为了安全起见、快照时间增加了一分钟：

```
[oracle@host1 ~]$ sqlplus / as sysdba
Connected to an idle instance.
SQL> startup mount;
ORACLE instance started.
Total System Global Area 1610612736 bytes
Fixed Size                  2924928 bytes
Variable Size            1040191104 bytes
Database Buffers          553648128 bytes
Redo Buffers               13848576 bytes
Database mounted.
SQL> recover database until time '09-MAR-2017 10:44:15' snapshot time '09-
MAR-2017 10:11:00';
```

此时将启动恢复。考虑到可能的时钟差异、它指定了10：11：00的快照时间(比记录的时间晚一分钟)和10：44的目标恢复时间。接下来、sqlplus请求所需的归档日志、以达到所需的恢复时间10：44。

```
ORA-00279: change 551760 generated at 03/09/2017 05:06:07 needed for
thread 1
ORA-00289: suggestion : /oralogs_nfs/arch/1_31_930813377.dbf
ORA-00280: change 551760 for thread 1 is in sequence #31
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
ORA-00279: change 552566 generated at 03/09/2017 05:08:09 needed for
thread 1
ORA-00289: suggestion : /oralogs_nfs/arch/1_32_930813377.dbf
ORA-00280: change 552566 for thread 1 is in sequence #32
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
ORA-00279: change 553045 generated at 03/09/2017 05:10:12 needed for
thread 1
ORA-00289: suggestion : /oralogs_nfs/arch/1_33_930813377.dbf
ORA-00280: change 553045 for thread 1 is in sequence #33
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
ORA-00279: change 753229 generated at 03/09/2017 05:15:58 needed for
thread 1
ORA-00289: suggestion : /oralogs_nfs/arch/1_34_930813377.dbf
ORA-00280: change 753229 for thread 1 is in sequence #34
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
Log applied.
Media recovery complete.
SQL> alter database open resetlogs;
Database altered.
SQL>
```

使用使用快照完成数据库恢复 `recover automatic` 命令不需要特定的许可、但需要使用进行时间点恢复 `snapshot time` 需要Oracle高级压缩许可证。

Oracle数据库管理和自动化工具

ONTAP在Oracle数据库环境中的主要价值来自核心ONTAP技术、例如即时Snapshot副本、简单的SnapMirror复制以及高效创建FlexClone卷。

在某些情况下、直接在ONTAP上配置这些核心功能即可满足要求、但更复杂的需求则需要一个业务流程层。

SnapCenter

SnapCenter是NetApp的旗舰级数据保护产品。从很低的层面来看、它在执行数据库备份的方式上与SnapManager产品类似、但它是从头开始构建的、用于在NetApp存储系统上提供单一管理平台来进行数据保护管理。

SnapCenter包括一些基本功能、例如基于快照的备份和还原、SnapMirror和SnapVault复制、以及大型企业大规模运行所需的其他功能。这些高级功能包括扩展的基于角色的访问控制(Role-Based Access Control、RBAC)功

能、可与第三方业务流程产品集成的REST API、对数据库主机上的SnapCenter插件进行无中断集中管理、以及专为云规模环境设计的用户界面。

REST

ONTAP还包含丰富的ESTful API集。这样、第三方供应商就可以创建与ONTAP深度集成的数据保护和其他管理应用程序。此外、希望创建自己的自动化工作流和实用程序的客户也可以轻松使用这种ESTful API。

Oracle灾难恢复

使用ONTAP进行Oracle数据库灾难恢复

灾难恢复是指在发生灾难性事件(例如火灾、导致存储系统甚至整个站点遭到破坏)后恢复数据服务。

> 本文档可替代先前发布的技术报告_TR-4591：《Oracle数据保护》_和_TR-4592：《基于MetroCluster的Oracle》

灾难恢复可以通过使用SnapMirror轻松复制数据来实现、当然、许多客户每小时更新一次镜像副本。

对于大多数客户而言、灾难恢复不仅需要拥有远程数据副本、还需要能够快速利用这些数据。NetApp提供了两种技术来满足这一需求—MetroCluster和SnapMirror主动同步

MetroCluster是指硬件配置中的ONTAP、其中包括低级同步镜像存储和许多附加功能。MetroCluster等集成解决方案简化了当今复杂的横向扩展数据库、应用程序和虚拟化基础架构。它将多个外部数据保护产品和策略替换为一个简单的中央存储阵列。此外、它还可以在一个集群模式存储系统中提供集成的备份、恢复、灾难恢复和高可用性(HA)功能。

SnapMirror活动同步基于SnapMirror同步。通过MetroCluster、每个ONTAP控制器都负责将其驱动器数据复制到远程位置。使用SnapMirror主动同步时、您实际上拥有两个不同的ONTAP系统、它们会维护LUN数据的独立副本、但会相互协作、为该LUN提供一个实例。从主机角度来看、它是一个LUN实体。

虽然SnapMirror主动同步和MetroCluster在内部的工作方式截然不同、但对于主机来说、结果却非常相似。主要区别在于粒度。如果您只需要选择要同步复制的工作负载、则SnapMirror活动同步是更好的选择。如果您需要复制整个环境甚至数据中心、MetroCluster是一个更好的选择。此外、SnapMirror主动同步目前仅支持SAN、而MetroCluster支持多协议、包括SAN、NFS和SMB。

MetroCluster

MetroCluster物理架构和Oracle数据库

要了解Oracle数据库在MetroCluster环境中的运行方式、需要对MetroCluster系统的物理设计进行一些说明。

> 本文档可替代先前发布的技术报告_TR-4592：《基于MetroCluster的Oracle》

MetroCluster可用于3种不同的配置

- 具有IP连接的HA对

- 具有FC连接的HA对
- 具有FC连接的单个控制器

[注意]术语"连接"是指用于跨站点复制的集群连接。它不是指主机协议。无论用于集群间通信的连接类型如何、MetroCluster配置均支持所有主机端协议。

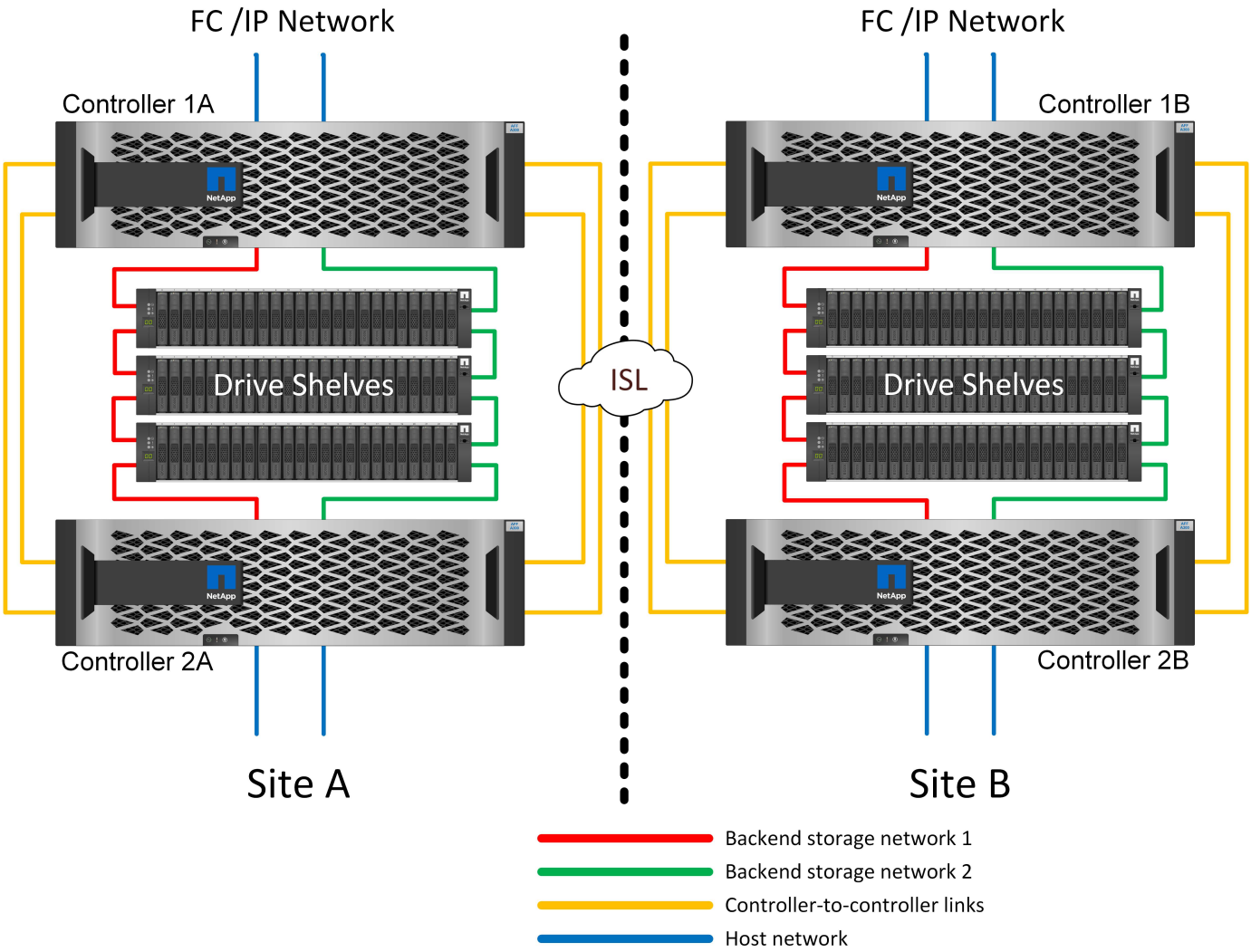
MetroCluster IP

HA对MetroCluster IP配置会在每个站点上使用两个或四个节点。与双节点选项相比、此配置选项会增加复杂性和成本、但它具有一个重要优势：站点内冗余。简单的控制器故障不需要通过WAN访问数据。数据访问仍通过备用本地控制器保持在本地。

大多数客户选择IP连接是因为基础架构要求更简单。过去、使用暗光纤和FC交换机配置高速跨站点连接通常比较容易、但如今、高速、低延迟IP电路更容易获得。

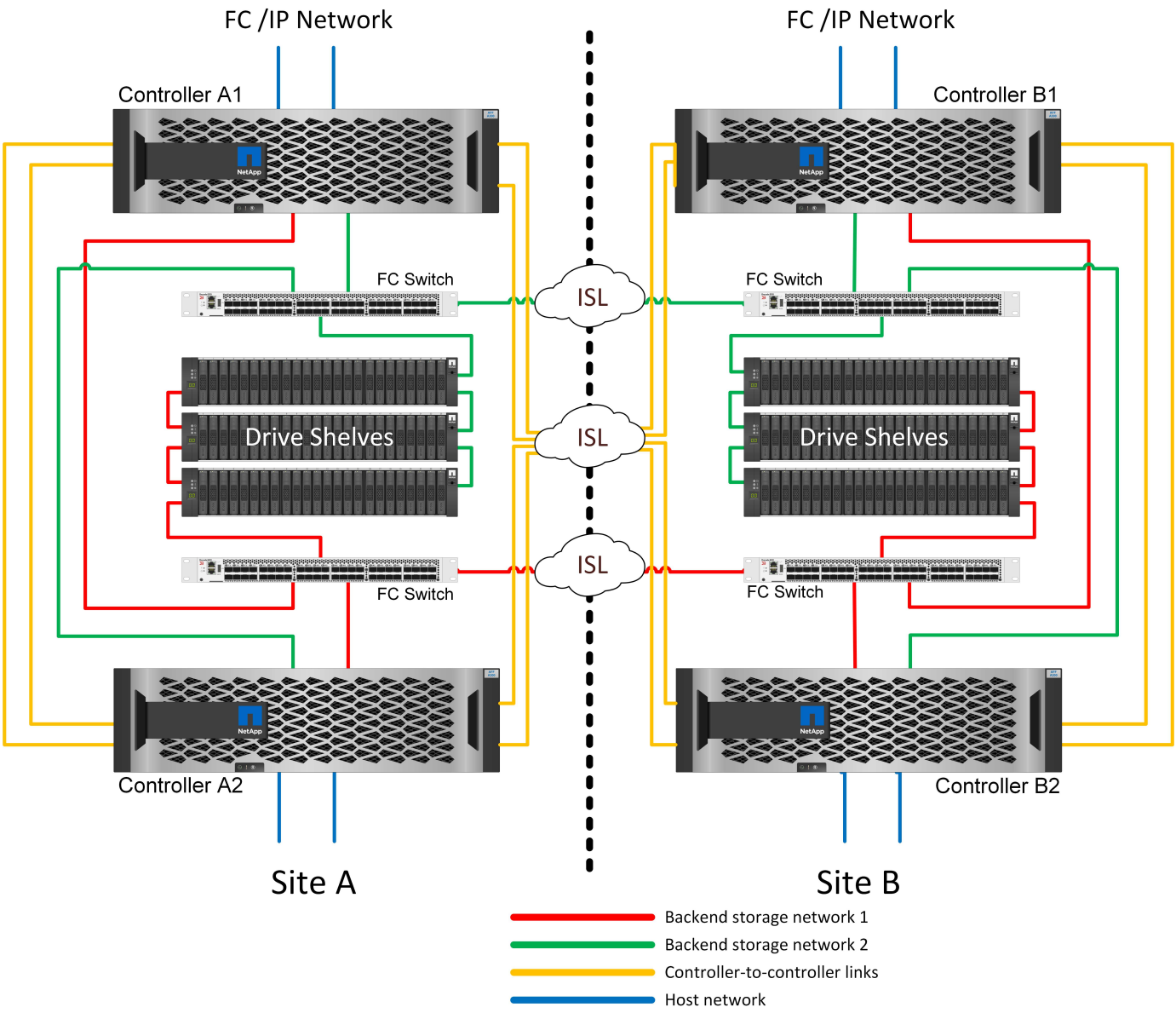
此外、该架构也更加简单、因为只有跨站点连接用于控制器。在FC SAN连接的MetroCluster中、控制器会直接写入另一站点上的驱动器、因此需要更多的SAN连接、交换机和网桥。相反、IP配置中的控制器会通过控制器写入相对的驱动器。

对于追加信息、请参阅ONTAP官方文档和 "MetroCluster IP 解决方案架构和设计"。



HA对FC SAN连接的MetroCluster

HA对MetroCluster FC配置会在每个站点上使用两个或四个节点。与双节点选项相比、此配置选项会增加复杂性和成本、但它具有一个重要优势：站点内冗余。简单的控制器故障不需要通过WAN访问数据。数据访问仍通过备用本地控制器保持在本地。



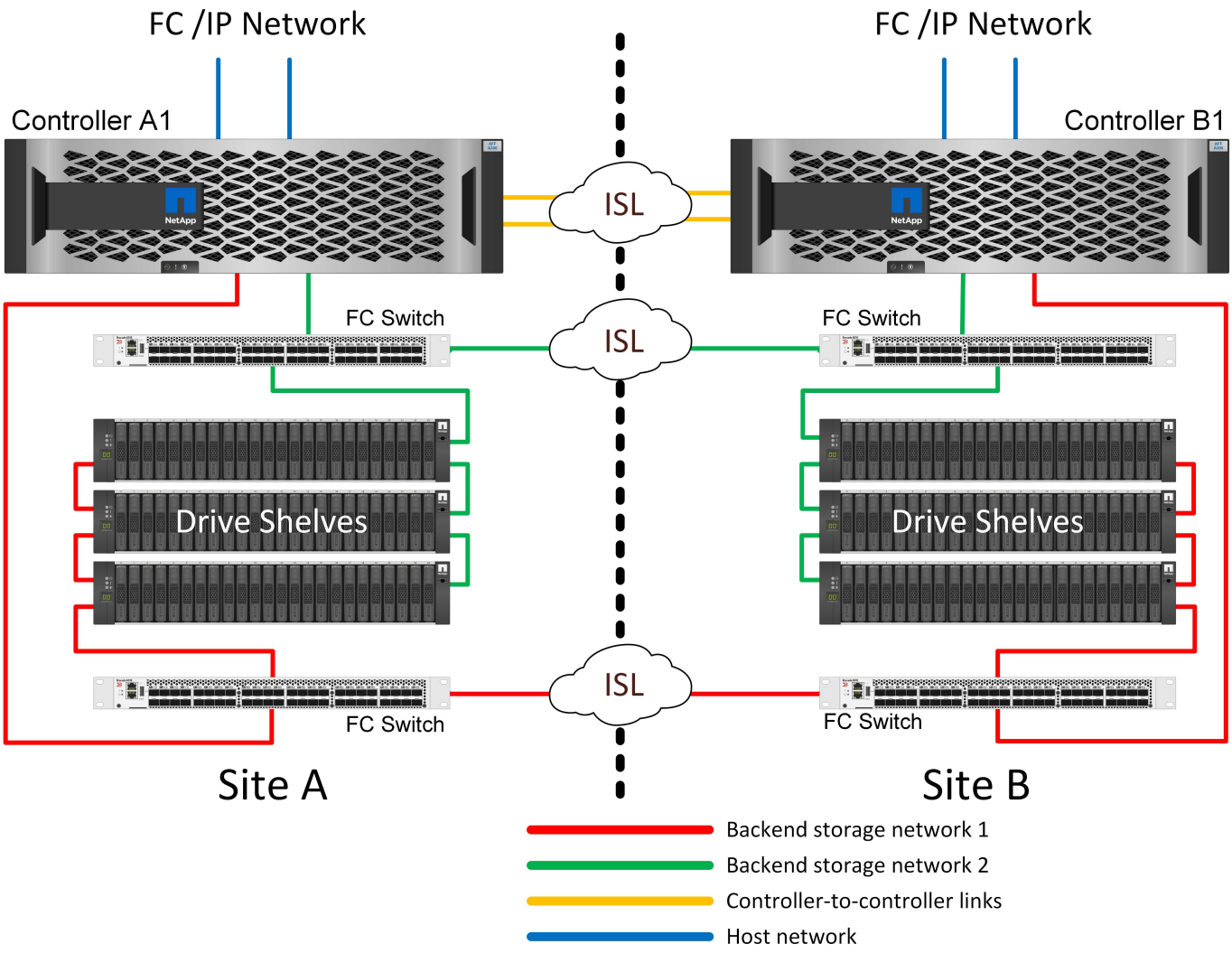
某些多站点基础架构不是为主动-主动操作而设计的、而是更多地用作主站点和灾难恢复站点。在这种情况下、通常最好使用HA对MetroCluster选项、原因如下：

- 尽管双节点MetroCluster集群是一个HA系统、但控制器意外故障或计划内维护要求数据服务必须在相反站点联机。如果站点之间的网络连接无法支持所需的带宽、则性能会受到影响。唯一的选择是同时将各种主机操作系统和相关服务故障转移到备用站点。HA对MetroCluster集群可消除此问题、因为丢失控制器会导致在同一站点内进行简单的故障转移。
- 某些网络拓扑不是为跨站点访问而设计的、而是使用不同的子网或隔离的FC SAN。在这些情况下、双节点MetroCluster集群将不再充当HA系统、因为备用控制器无法向对面站点上的服务器提供数据。要提供完全冗余、需要使用高可用性对MetroCluster选项。
- 如果将双站点基础架构视为一个高可用性基础架构、则适合使用双节点MetroCluster配置。但是、如果系统

在站点发生故障后必须长时间运行、则首选HA对、因为它会继续在单个站点中提供HA。

双节点FC SAN连接MetroCluster

双节点MetroCluster配置仅为每个站点使用一个节点。这种设计比HA对选项更简单、因为需要配置和维护的组件更少。此外、它还降低了布线和FC交换方面的基础架构需求。最后、它还可以降低成本。



这种设计的明显影响是、单个站点上的控制器故障意味着数据可以从另一个站点访问。这种限制不一定是问题。许多企业都拥有多站点数据中心运营、并采用延伸型高速低延迟网络、这些网络本质上充当一个基础架构。在这些情况下、首选配置是双节点版本的MetroCluster。目前、多家服务提供商以PB级的规模使用双节点系统。

MetroCluster故障恢复能力功能

MetroCluster 解决方案 中没有单点故障：

- 每个控制器都有两条通往本地站点上的驱动器架的独立路径。
- 每个控制器都有两条通往远程站点上驱动器架的独立路径。
- 每个控制器都有两条独立的路径连接到另一站点上的控制器。
- 在HA对配置中、每个控制器都有两个指向其本地配对节点的路径。

总之、可以删除配置中的任何一个组件、而不会影响MetroCluster提供数据的能力。这两个选项在故障恢复能力方面的唯一区别是、发生站点故障后、HA对版本仍然是整体HA存储系统。

MetroCluster逻辑架构和Oracle数据库

要了解Oracle数据库如何在MetroCluster环境alsop中运行、需要对MetroCluster系统的逻辑功能进行一些说明。

站点故障保护：NVRAM和MetroCluster

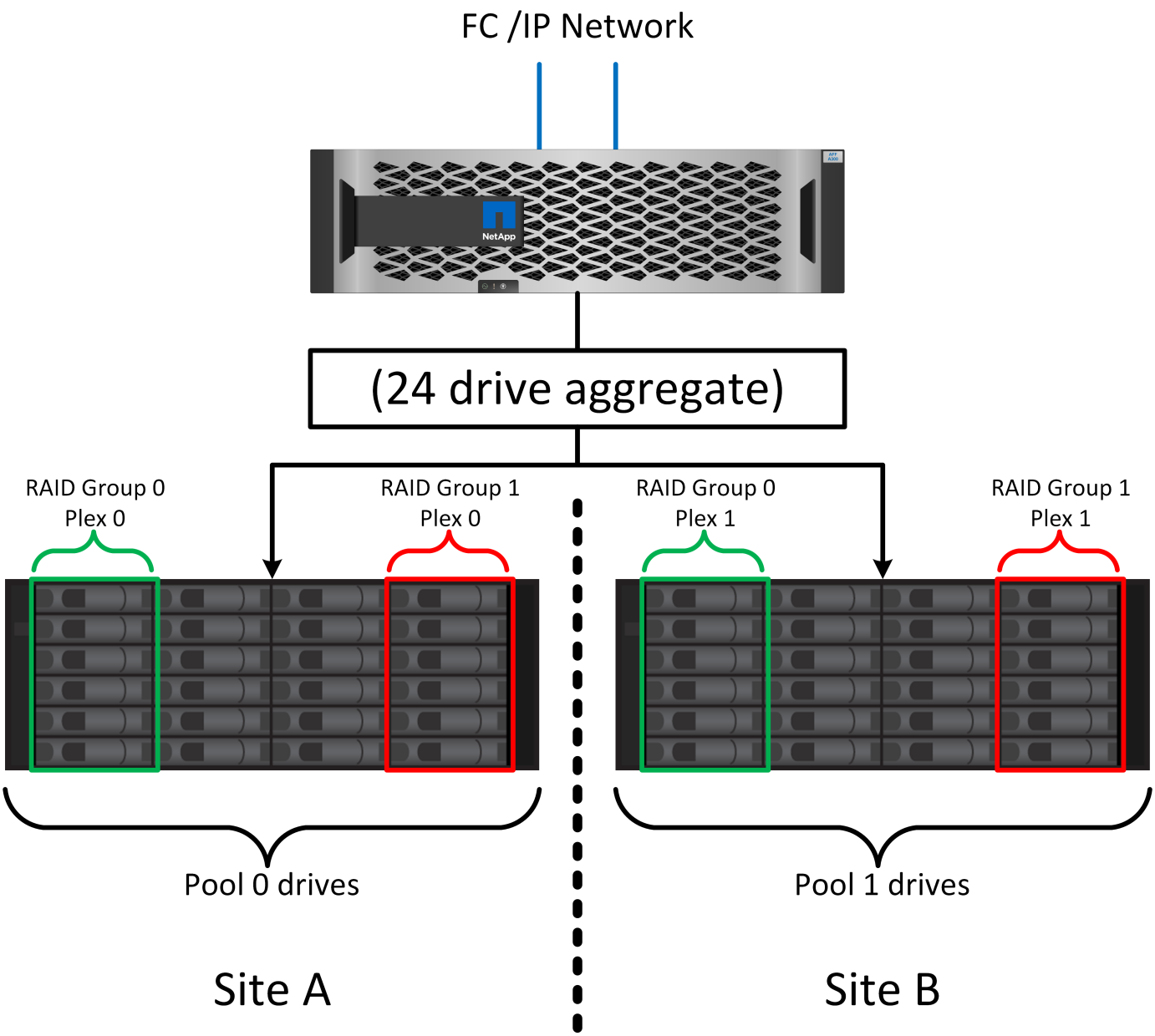
MetroCluster通过以下方式扩展NVRAM数据保护：

- 在双节点配置中、NVRAM数据通过交换机间链路(ISL)复制到远程配对节点。
- 在HA对配置中、NVRAM数据会同时复制到本地配对节点和远程配对节点。
- 写入只有在复制到所有配对项后才会得到确认。此架构通过将NVRAM数据复制到远程配对节点来保护传输中的I/O免受站点故障的影响。驱动器级数据复制不涉及此过程。拥有聚合的控制器负责通过向聚合中的两个plexes写入数据来进行数据复制、但在站点丢失时、仍必须防止传输中I/O丢失。只有当配对控制器必须接管发生故障的控制器时、才会使用复制的NVRAM数据。

站点和磁盘架故障保护：SyncMirror和plexes

SyncMirror是一种镜像技术、可增强但不会取代RAID DP或RAID-TEC。它会镜像两个独立RAID组的内容。逻辑配置如下：

1. 驱动器会根据位置配置到两个池中。一个池由站点A上的所有驱动器组成、另一个池由站点B上的所有驱动器组成
2. 然后、基于RAID组的镜像集创建一个通用存储池(称为聚合)。从每个站点提取的驱动器数量相等。例如、一个包含20个驱动器的SyncMirror聚合将由站点A的10个驱动器和站点B的10个驱动器组成
3. 给定站点上的每组驱动器都会自动配置为一个或多个完全冗余的RAID DP或RAID-TEC组、而不依赖于镜像的使用。在镜像下使用RAID可提供数据保护、即使在站点丢失后也是如此。



上图显示了一个示例SyncMirror配置。在控制器上创建了一个包含24个驱动器的聚合、其中12个驱动器来自站点A上分配的磁盘架、12个驱动器来自站点B上分配的磁盘架这些驱动器被分组为两个镜像RAID组。RAID组0在站点A上包含一个6驱动器丛、该丛镜像到站点B上的一个6驱动器丛同样、RAID组1在站点A上包含一个6驱动器丛、该丛镜像到站点B上的6驱动器丛

SyncMirror通常用于为MetroCluster系统提供远程镜像、每个站点有一个数据副本。有时、它会用于在单个系统中提供额外的冗余级别。尤其是、它可以提供磁盘架级冗余。驱动器架已包含双电源和控制器、总体比金属板稍多、但在某些情况下、可能需要额外保护。例如、一家NetApp客户为汽车测试期间使用的移动实时分析平台部署了SyncMirror。该系统分为两个物理机架、配有独立的电源和独立的UPS系统。

冗余故障：**NVFAIL**

如前文所述、写入操作只有在至少另一个控制器上记录到本地NVRAM和NVRAM后才会得到确认。此方法可确保硬件故障或断电不会导致传输中I/O丢失如果本地NVRAM发生故障或与其他节点的连接发生故障、则无法再镜像数据。

如果本地NVRAM报告错误、则此节点将关闭。此关闭会导致在使用HA对时故障转移到配对控制器。使用MetroCluster时、行为取决于所选的整体配置、但可能会自动故障转移到远程便签。在任何情况下、数据都不会丢失、因为发生故障的控制器尚未确认写入操作。

站点间连接故障会阻止NVRAM复制到远程节点、这种情况更为复杂。写入操作不再复制到远程节点、因此、如果控制器发生灾难性错误、可能会导致数据丢失。更重要的是、在这些情况下尝试故障转移到其他节点会导致数据丢失。

控制因素是NVRAM是否同步。如果NVRAM已同步、则可以安全地进行节点间故障转移、而不会丢失数据。在MetroCluster配置中、如果NVRAM与底层聚合plexes处于同步状态、则可以安全地继续执行切换、而不会丢失数据。

除非强制执行故障转移或切换、否则ONTAP不允许在数据不同步时执行故障转移或切换。以这种方式强制更改条件即表示数据可能会留在原始控制器中、并且数据丢失是可以接受的。

如果强制执行故障转移或切换、则数据库和其他应用程序尤其容易受到损坏的影响、因为它们在磁盘上维护着更大的内部数据缓存。如果发生强制故障转移或切换、先前确认的更改将被有效丢弃。存储阵列的内容会及时有效地向后跳转、缓存的状态不再反映磁盘上数据的状态。

为了防止出现这种情况、ONTAP允许对卷进行配置、以便针对NVRAM故障提供特殊保护。触发此保护机制后、卷将进入名为NVFAIL的状态。此状态会导致发生原因应用程序崩溃的I/O错误。此崩溃会导致应用程序关闭、以便它们不会使用过时数据。数据不应丢失、因为日志中应存在任何已提交的事务数据。通常的后续步骤是、管理员先完全关闭主机、然后再手动将LUN和卷重新联机。虽然这些步骤可能涉及一些工作、但这种方法是确保数据完整性的最安全方法。并非所有数据都需要这种保护、这就是可以逐个卷配置NVFAIL行为的原因。

HA对和MetroCluster

MetroCluster有两种配置：双节点和HA对。就NVRAM而言、双节点配置与HA对的行为相同。如果发生突然故障、配对节点可以重放NVRAM数据、以确保驱动器一致、并确保未丢失任何已确认的写入。

HA对配置也会将NVRAM复制到本地配对节点。简单的控制器故障会导致配对节点上的NVRAM重放、就像不使用MetroCluster的独立HA对一样。如果站点突然完全丢失、远程站点还具有必要的NVRAM、以使驱动器保持一致并开始提供数据。

MetroCluster的一个重要方面是、在正常运行条件下、远程节点无法访问配对节点数据。每个站点本质上都是一个独立的系统、可以承担相反站点的特性。此过程称为切换、其中包括计划内切换、在此过程中、站点操作会无系统地迁移到相反站点。此外、还包括站点丢失以及在灾难恢复过程中需要手动或自动切换的计划外情况。

切换和切回

术语切换和切回是指在MetroCluster配置中的远程控制器之间过渡卷的过程。此过程仅会对远程节点执行适用场景。如果在四卷配置中使用MetroCluster、则本地节点故障转移与前面所述的接管和恢复过程相同。

计划内切换和切回

计划内切换或切回类似于节点之间的接管或交还。此过程包含多个步骤、看起来可能需要几分钟时间、但实际发生的是存储和网络资源的多阶段平稳过渡。控制传输的速度比执行完整命令所需的时间快得多。

接管/交还与切换/切回之间的主要区别在于对FC SAN连接的影响。使用本地接管/备份时、主机会丢失指向本地节点的所有FC路径、并依靠其本机MPIO切换到可用的备用路径。端口不会重新定位。通过切换和切回、控制器上的虚拟FC目标端口将过渡到另一站点。它们实际上暂时不再存在于SAN上、然后重新出现在备用控制器上。

SyncMirror超时

SyncMirror是一种ONTAP镜像技术、可针对磁盘架故障提供保护。如果磁盘架相隔一段距离、则可以实现远程数据保护。

SyncMirror不提供通用同步镜像。结果是可用性更好。某些存储系统使用持续的全镜像或无镜像、有时称为Domino模式。这种形式的镜像在应用程序中受到限制、因为如果与远程站点的连接断开、所有写入活动都必须停止。否则、写入将在一个站点上存在、而在另一个站点上不存在。通常、此类环境会配置为在站点间连接丢失的时间较短(例如30秒)时使LUN脱机。

这种行为适合一小部分环境。但是、大多数应用程序都需要一个解决方案、该系统可以在正常运行条件下提供有保障的同步复制、但可以暂停复制。站点间连接完全断开通常被视为近乎灾难的情况。通常、此类环境会保持联机并提供数据、直到修复连接或正式决定关闭环境以保护数据为止。仅由于远程复制失败而要求自动关闭应用程序的要求并不常见。

SyncMirror支持同步镜像要求、并具有超时的灵活性。如果与远程控制器和/或丛的连接断开、30秒计时器将开始倒计时。当计数器达到0时、写入I/O处理将继续使用本地数据。数据的远程副本可用、但会及时冻结、直到连接恢复为止。重新同步利用聚合级快照使系统尽快恢复到同步模式。

值得注意的是、在许多情况下、在应用程序层实施这种通用的全Domino模式或全无Domino模式复制效果更佳。例如、Oracle DataGuard包括最大保护模式、可保证在任何情况下进行长实例复制。如果复制链路出现故障的时间超过可配置的超时时间、数据库将关闭。

使用光纤连接**MetroCluster**自动执行无人看管切换

自动无人值守切换(Automatic无人值守切换、AUSO)是一项光纤连接的MetroCluster功能、可提供一种跨站点HA形式。如前文所述、MetroCluster有两种类型：每个站点上一个控制器或每个站点上一个HA对。HA选项的主要优势是、计划内或计划外控制器关闭仍可使所有I/O都位于本地。单节点选项的优势在于降低成本、复杂性和基础架构。

AUSO的主要价值是提高光纤连接MetroCluster系统的HA功能。每个站点都会监控相反站点的运行状况、如果没有节点可提供数据、则AUSO会导致快速切换。在每个站点只有一个节点的MetroCluster配置中、此方法尤其有用、因为它使配置在可用性方面更接近HA对。

AUSO无法在HA对级别提供全面监控。HA对可以提供极高的可用性、因为它包含两根冗余物理缆线、用于节点到节点的直接通信。此外、HA对中的两个节点均可访问冗余环路上的同一组磁盘、从而为一个节点提供另一条路由来监控另一个节点的运行状况。

MetroCluster集群存在于节点间通信和磁盘访问均依赖于站点间网络连接的站点之间。监控集群其余部分的检测信号的能力有限。在另一个站点因网络问题而实际关闭而不是不可用的情况下、AUSO必须区分这种情况。

因此、如果HA对中的控制器检测到因特定原因(例如系统崩溃)而发生的控制器故障、则该控制器可能会提示接管。如果完全断开连接(有时称为丢失检测信号)、它还会提示接管。

只有在原始站点上检测到特定故障时、MetroCluster系统才能安全地执行自动切换。此外、拥有存储系统的控制器必须能够保证磁盘和NVRAM数据保持同步。控制器无法仅因为与源站点断开连接而保证切换的安全性、而源站点仍可正常运行。有关自动执行切换的其他选项、请参见下一节中有关MetroCluster Tieb破碎 机(MCTB)解决方案的信息。

具有光纤连接**MetroCluster**的**MetroCluster Tieb**破碎 机

。 "NetApp MetroCluster Tieb破碎 机" 软件可以在第三个站点上运行、以监控MetroCluster环境的运行状况、发送通知、并在发生灾难时强制执行切换(可选)。有关Tieb破碎 机的完整问题描述、请参见 "NetApp 支持站点"

但MetroCluster Tieb破碎 机的主要用途是检测站点丢失。它还必须区分站点丢失和连接丢失。例如、切换不应因TiebREAKER无法访问主站点而发生、这就是TiebREAKER同时监控远程站点联系主站点的能力的原因。

使用AUSO自动切换也与MCTB兼容。AUSO反应非常迅速、因为它可以检测特定的故障事件、然后仅在NVRAM和SyncMirror plexes处于同步状态时调用切换。

相反、Tieb破碎 机位于远程位置、因此必须等待计时器经过、然后才能宣布站点停机。Tieb破碎 机最终会检测到由AUSO涵盖的那种控制器故障、但通常、在Tieb破碎 机开始工作之前、AUSO已启动切换、并且可能已完成切换。Tieb破碎 机生成的第二个切换命令将被拒绝。

*注意：*强制切换时、MCTB软件不会验证NVRAM是否同步和/或plexes是否同步。如果已配置自动切换、则应在维护活动期间禁用、从而导致NVRAM或SyncMirror plexes失去同步。

此外、MCTB可能无法解决导致以下一系列事件的滚动灾难：

1. 站点之间的连接中断30秒以上。
2. SyncMirror复制超时、并且会继续在主站点上执行操作、从而使远程副本过时。
3. 主站点丢失。结果是主站点上存在未复制的更改。因此、切换可能不受欢迎、原因有很多、其中包括：
    - 主站点上可能存在关键数据、这些数据最终可能是可恢复的。允许应用程序继续运行的切换将有效地丢弃这些关键数据。
    - 运行正常的站点上的某个应用程序在站点丢失时使用了主站点上的存储资源、此应用程序可能已缓存数据。切换会导致数据版本过时、与缓存不匹配。
    - 运行正常的站点上的某个操作系统在站点丢失时使用了主站点上的存储资源、此操作系统可能已缓存数据。切换会导致数据版本过时、与缓存不匹配。最安全的方法是、将Tieber4配置为在检测到站点故障时发送警报、然后由某人决定是否强制执行切换。可能需要先关闭应用程序和/或操作系统、才能清除缓存的任何数据。此外、还可以使用NVFAIL设置来添加进一步的保护、并帮助简化故障转移过程。

使用**MetroCluster IP**的**ONTAP**调解器

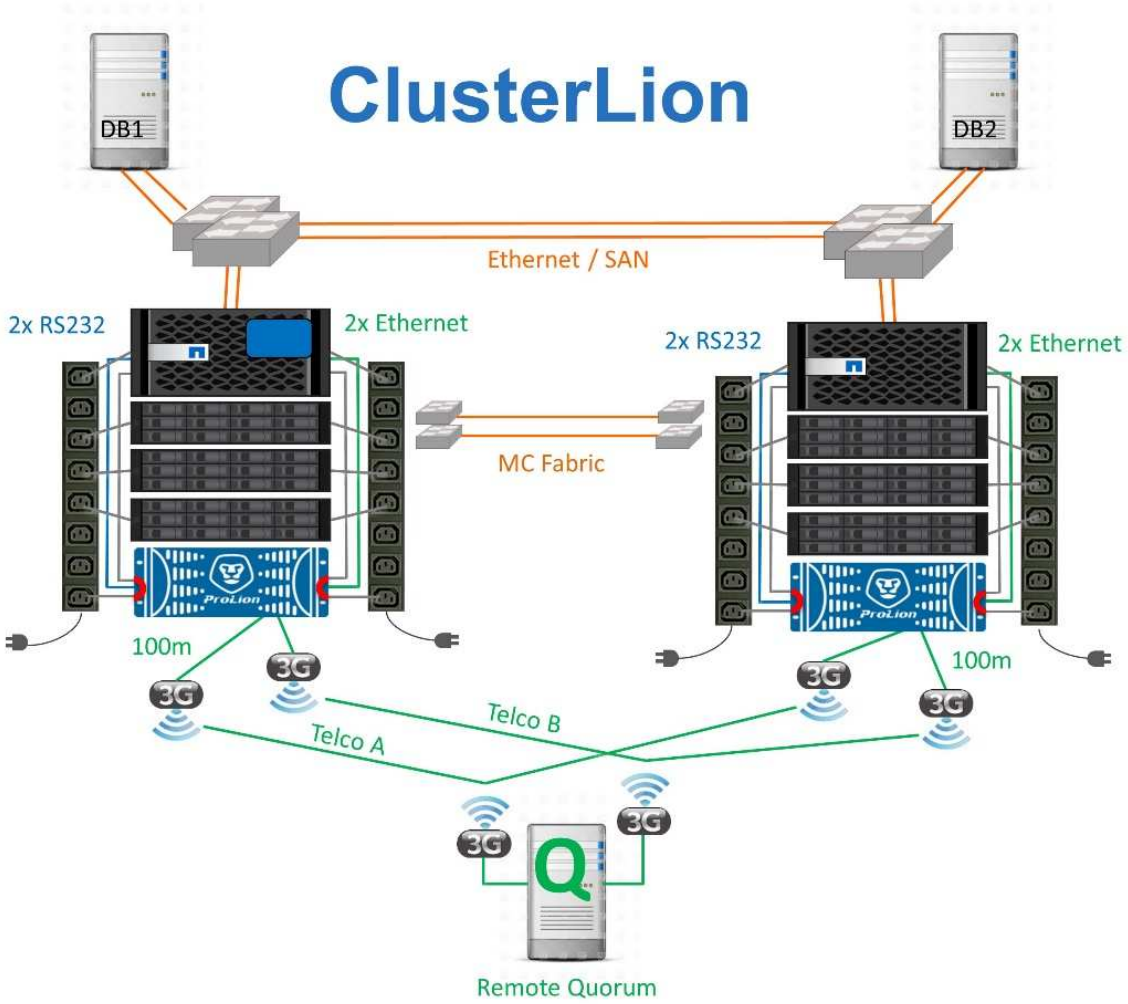
ONTAP调解器可与MetroCluster IP和某些其他ONTAP解决方案结合使用。它的功能与上述MetroCluster Tieb破碎 机软件非常相似、但也包括一项关键功能、即执行自动无人值守切换。

光纤连接的MetroCluster可以直接访问相反站点上的存储设备。这样、一个MetroCluster控制器就可以通过从驱动器中读取检测信号数据来监控其他控制器的运行状况。这样、一个控制器就可以识别另一个控制器的故障并执行切换。

相比之下、MetroCluster IP架构会通过控制器-控制器连接独占路由所有I/O；无法直接访问远程站点上的存储设备。这会限制控制器检测故障和执行切换的能力。因此、需要将ONTAP调解器作为Tieb破碎 机设备来检测站点丢失并自动执行切换。

使用**ClusterLion**的虚拟第三站点

ClusterLion是一种高级MetroCluster监控设备、可充当虚拟第三站点。通过这种方法、可以在双站点配置中安全地部署MetroCluster、并提供完全自动化的切换功能。此外、ClusterLion还可以执行额外的网络级监控并执行切换后操作。完整文档可从ProLion获得。



- ClusterLion设备可通过直接连接的以太网和串行缆线监控控制器的运行状况。
- 这两个设备通过冗余3G无线连接相互连接。
- ONTAP控制器的电源通过内部继电器供电。如果站点发生故障、包含内部UPS系统的ClusterLion会在调用切换之前断开电源连接。此过程可确保不会出现脑裂情况。
- ClusterLion会在30秒SyncMirror超时时间内执行切换、或者根本不执行切换。
- 除非NVRAM和SyncMirror plexes的状态保持同步、否则ClusterLion不会执行切换。
- 由于ClusterLion仅在MetroCluster完全同步时执行切换、因此不需要NVFAIL。此配置允许站点范围的环境(例如扩展Oracle RAC)保持联机、即使在计划外切换期间也是如此。
- 支持包括光纤连接MetroCluster和MetroCluster IP

采用**SyncMirror**的**Oracle**数据库

使用MetroCluster系统进行Oracle数据保护的基础是SyncMirror、这是一种性能最高的横向扩展同步镜像技术。

利用**SyncMirror**实现数据保护

最简单的一个层面是、同步复制意味着、在确认镜像存储之前、必须对镜像存储的两端进行任何更改。例如、如果数据库正在写入日志、或者VMware子系统正在修补、则写入操作绝不能丢失。作为协议级别、在将写入提交

到两个站点上的非易失性介质之前、存储系统不得确认写入。只有这样、才能安全地继续操作、而不会丢失数据。

使用同步复制技术是设计和管理同步复制解决方案的第一步。最重要的注意事项是了解在各种计划内和计划外故障情形下可能发生的情况。并非所有同步复制解决方案都能提供相同的功能。如果您需要的解决方案能够实现零恢复点目标(RPO)、即零数据丢失、则必须考虑所有故障情形。特别是、如果由于站点间连接断开而无法进行复制、则会产生什么预期结果?

SyncMirror数据可用性

MetroCluster复制基于NetApp SyncMirror技术、该技术旨在高效地切换至同步模式和切换至同步模式之外。此功能可满足需要同步复制、但也需要数据服务高可用性的客户的要求。例如、如果与远程站点的连接断开、则通常最好让存储系统继续在未复制的状态下运行。

许多同步复制解决方案只能在同步模式下运行。这种类型的全或全不复制有时称为Domino模式。此类存储系统将停止提供数据、而不是允许本地和远程数据副本处于不同步状态。如果强制中断复制、重新同步可能会非常耗时、并且可能会使客户在重新建立镜像期间完全丢失数据。

SyncMirror不仅可以在无法访问远程站点时无缝切换出同步模式、还可以在恢复连接后快速重新同步到RPO = 0状态。远程站点上的陈旧数据副本也可以在重新同步期间保留在可用状态、从而确保本地和远程数据副本始终存在。

如果需要Domino模式、则NetApp提供SnapMirror同步(SM-S)。此外、还存在应用程序级选项、例如Oracle DataGuard或SQL Server Always On可用性组。可以选择操作系统级磁盘镜像。有关追加信息和选项、请咨询您的NetApp或合作伙伴客户团队。

使用MetroCluster进行Oracle数据库故障转移

MetroCluster是ONTAP的一项功能、可通过站点间的RPO = 0同步镜像保护Oracle数据库、并可进行扩展以在一个MetroCluster系统上支持数百个数据库。它也易于使用。使用MetroCluster并不一定会增加或更改运行企业级应用程序和数据库的任何最佳网络竞赛。

通常的最佳实践仍然适用、如果您的需求仅需要RPO = 0数据保护、则MetroCluster可以满足该需求。但是、大多数客户使用MetroCluster不仅可以实现RROT=0的数据保护、还可以在灾难情形下提高RTO、并在站点维护活动中提供透明的故障转移。

Oracle数据库、MetroCluster和NVFAIL

NVFAIL是ONTAP中的一项通用数据完整性功能、旨在最大限度地提高数据库的数据完整性保护。

> 本节将详细介绍基本ONTAP NVFAIL、以涵盖MetroCluster特定的主题。

使用MetroCluster时、写入操作在至少另一个控制器上登录到本地NVRAM和NVRAM后才会得到确认。此方法可确保硬件故障或断电不会导致传输中I/O丢失如果本地NVRAM发生故障或与其他节点的连接发生故障、则无法再镜像数据。

如果本地NVRAM报告错误、则此节点将关闭。此关闭会导致在使用HA对时故障转移到配对控制器。使用MetroCluster时、行为取决于所选的整体配置、但可能会自动故障转移到远程便签。在任何情况下、数据都不会丢失、因为发生故障的控制器尚未确认写入操作。

站点间连接故障会阻止NVRAM复制到远程节点、这种情况更为复杂。写入操作不再复制到远程节点、因此、如果控制器发生灾难性错误、可能会导致数据丢失。更重要的是、在这些情况下尝试故障转移到其他节点会导致数据丢失。

控制因素是NVRAM是否同步。如果NVRAM已同步、则可以安全地进行节点间故障转移、而不会丢失数据。在MetroCluster配置中、如果NVRAM与底层聚合plexes处于同步状态、则可以安全地继续执行切换、而不会丢失数据。

除非强制执行故障转移或切换、否则ONTAP不允许在数据不同步时执行故障转移或切换。以这种方式强制更改条件即表示数据可能会留在原始控制器中、并且数据丢失是可以接受的。

如果强制执行故障转移或切换、则数据库尤其容易受到损坏的影响、因为数据库在磁盘上维护着更大的内部数据缓存。如果发生强制故障转移或切换、先前确认的更改将被有效丢弃。存储阵列的内容会及时有效地向后跳转、数据库缓存的状态不再反映磁盘上数据的状态。

为了保护应用程序免受这种情况的影响、ONTAP允许对卷进行配置、以便针对NVRAM故障提供特殊保护。触发此保护机制后、卷将进入名为NVFAIL的状态。此状态会导致I/O错误、发生原因应用程序会关闭以使其不使用陈旧数据。不应丢失数据、因为存储系统上仍存在任何已确认的写入、对于数据库、任何已提交的事务数据都应出现在日志中。

通常的后续步骤是、管理员先完全关闭主机、然后再手动将LUN和卷重新联机。虽然这些步骤可能涉及一些工作、但这种方法是确保数据完整性的最安全方法。并非所有数据都需要这种保护、这就是可以逐个卷配置NVFAIL行为的原因。

手动强制NVFAIL

要强制与分布在各个站点上的应用程序集群(包括VMware、Oracle RAC等)进行切换、最安全的方法是指定 `-force-nvfail-all` 在命令行中。此选项可作为紧急措施使用、以确保所有缓存数据均已转储。如果主机正在使用最初位于发生灾难的站点上的存储资源、则会收到I/O错误或陈旧的文件句柄 (`ESTALE`)错误。Oracle数据库崩溃、文件系统要么完全脱机、要么切换到只读模式。

切换完成后、 `in-nvfailed-state` 标记、并且LUN需要置于联机状态。完成此活动后、可以重新启动数据库。这些任务可以自动执行、以减少RTO。

dr-force－nvfail

作为一般安全措施、请设置 `dr-force-nvfail` 在正常操作期间可能从远程站点访问的所有卷上的标志、表示它们是故障转移之前使用的活动。此设置的结果是、所选远程卷在进入后将不可用 `in-nvfailed-state` 切换期间。切换完成后、 `in-nvfailed-state` 标记、并且LUN必须置于联机状态。完成这些活动后、可以重新启动应用程序。这些任务可以自动执行、以减少RTO。

结果类似于使用 `-force-nvfail-all` 用于手动切换的标志。但是、受影响的卷数量可以仅限于那些必须防止应用程序或具有陈旧缓存的操作系统访问的卷。

对于不使用的环境、有两个关键要求 `dr-force-nvfail` 在应用程序卷上：

- 在主站点丢失后、强制切换的发生时间不得超过30秒。
- 在执行维护任务期间、或者在SyncMirror plexes或NVRAM复制不同步的任何其他情况下、不得发生切换。第一个要求可通过Tieber4软件来满足、该软件配置为在站点发生故障后30秒内执行切换。此要求并不意味着必须在检测到站点故障后30秒内执行切换。这确实意味着、如果自某个站点确认正常运行后30秒内已过、则不再安全地强制执行切换。

如果已知MetroCluster配置不同步、则可以通过禁用所有自动切换功能来部分满足第二项要求。更好的选择是、

使用Tiebre机会 解决方案监控NVRAM复制和SyncMirror plexes的运行状况。如果集群未完全同步、则Tieb破碎机不应触发切换。

NetApp MCTB软件无法监控同步状态、因此、如果MetroCluster因任何原因而不同步、则应将其禁用。ClusterLion具有NVRAM监控和丛监控功能、可以将其配置为不触发切换、除非确认MetroCluster系统已完全同步。

基于**MetroCluster**的**Oracle**单实例

如前所述、MetroCluster系统的存在并不一定会增加或更改数据库的任何最佳操作实践。客户MetroCluster系统上当前运行的大多数数据库都是单个实例、并遵循Oracle on ONTAP文档中的建议。

使用预配置的操作系统进行故障转移

SyncMirror在灾难恢复站点提供数据的同步副本、但要使数据可用、需要使用操作系统和相关应用程序。基本自动化可以显著缩短整个环境的故障转移时间。通常会使用Veritas Cluster Server (VCS)等集群软件产品在各个站点之间创建集群、在许多情况下、可以使用简单的脚本来驱动故障转移过程。

如果主节点丢失、则会将集群软件(或脚本)配置为在备用站点使数据库联机。一种方法是、创建为构成数据库的NFS或SAN资源预先配置的备用服务器。如果主站点发生故障、则集群软件或脚本化备用站点将执行一系列类似以下内容的操作：

1. 强制执行MetroCluster切换
2. 发现FC LUN (仅限SAN)
3. 挂载文件系统和/或挂载ASM磁盘组
4. 正在启动数据库

此方法的主要要求是在远程站点上运行操作系统。它必须预配置Oracle二进制文件、这也意味着必须在主站点和备用站点上执行Oracle修补等任务。或者、也可以将Oracle二进制文件镜像到远程站点、并在声明发生灾难时进行挂载。

实际激活操作步骤非常简单。LUN发现等命令只需对每个FC端口执行几个命令即可。文件系统挂载只不过是一个 `mount` 命令、数据库和ASM均可通过CLI使用一个命令来启动和停止。如果在切换之前灾难恢复站点未使用卷和文件系统、则无需设置 `dr-force- nvfail` 卷上。

使用虚拟化操作系统进行故障转移

数据库环境的故障转移可以扩展到包括操作系统本身。理论上、这种故障转移可以使用启动LUN来完成、但大多数情况下、这种故障转移是通过虚拟化操作系统来完成的。操作步骤类似于以下步骤：

1. 强制执行MetroCluster切换
2. 挂载托管数据库服务器虚拟机的数据存储库
3. 启动虚拟机
4. 手动启动数据库或将虚拟机配置为自动启动数据库、例如、ESX集群可以跨越多个站点。发生灾难时、可以在切换后将灾难恢复站点上的虚拟机置于联机状态。只要在发生灾难时托管虚拟化数据库服务器的数据存储库未在使用中、就不需要进行设置 `dr-force- nvfail` 在关联卷上。

基于MetroCluster的扩展Oracle RAC

许多客户通过跨站点扩展Oracle RAC集群来优化其RTO、从而形成完全主动-主动配置。整体设计变得更加复杂、因为它必须包括Oracle RAC的仲裁管理。此外、还可以从两个站点访问数据、这意味着强制切换可能会导致使用过时的数据副本。

尽管两个站点上都存在数据副本、但只有当前拥有聚合的控制器才能提供数据。因此、对于扩展RAC集群、远程节点必须通过站点到站点连接执行I/O。结果会增加I/O延迟、但这种延迟通常不是问题。RAC互连网络还必须跨站点延伸、这意味着无论如何都需要一个高速、低延迟的网络。如果增加的延迟使发生原因出现问题、则可以主动-被动方式运行集群。然后、需要将I/O密集型操作定向到拥有聚合的控制器本地的RAC节点。然后、远程节点会执行较轻的I/O操作、或者纯粹用作热备用服务器。

如果需要主动-主动扩展RAC、则应考虑使用ASM镜像代替MetroCluster。ASM镜像允许首选使用特定的数据副本。因此、可以构建一个扩展RAC集群、在该集群中、所有读取操作都在本地进行。读取I/O不会跨越站点、从而尽可能地降低延迟。所有写入活动仍必须传输站点间连接、但使用任何同步镜像解决方案时、此类流量都是不可避免的。

> 如果在Oracle RAC中使用启动LUN (包括虚拟化启动磁盘)、则 `misscount` 可能需要更改参数。有关RAC超时参数的详细信息、请参阅 "采用ONTAP的Oracle RAC"。

双站点配置

双站点扩展RAC配置可以提供主动-主动数据库服务、这些服务可以在许多(并非所有)灾难情形下无系统地经受住。

RAC投票文件

在MetroCluster上部署扩展RAC时、首要考虑事项应该是仲裁管理。Oracle RAC有两种管理仲裁的机制：磁盘检测信号和网络检测信号。磁盘检测信号可使用表决文件监控存储访问。对于单站点RAC配置、只要底层存储系统提供HA功能、单个表决资源就足够了。

在早期版本的Oracle中、投票文件放置在物理存储设备上、但在当前版本的Oracle中、投票文件存储在ASM磁盘组中。

> NFS支持Oracle RAC。在网格安装过程中、会创建一组ASM进程、以将网格文件使用的NFS位置显示为ASM磁盘组。此过程对最终用户几乎是透明的、安装完成后无需持续进行ASM管理。

双站点配置的第一个要求是、确保每个站点始终可以访问一半以上的表决文件、并确保灾难恢复过程不会中断。在表决文件存储在ASM磁盘组中之前、此任务非常简单、但如今管理员需要了解ASM冗余的基本原则。

ASM磁盘组有三个冗余选项 `external`， `normal`，和 `high`。换言之、未镜像、镜像和三向镜像。名为的新选项 `Flex` 也可用、但很少使用。冗余设备的冗余级别和放置位置控制了在故障情形下发生的情况。例如：

- 将表决文件放置在上 `diskgroup` 使用 `external` 冗余资源可确保在站点间连接断开时逐出一个站点。
- 将表决文件放置在上 `diskgroup` 使用 `normal` 每个站点只有一个ASM磁盘的冗余可确保在站点间连接断开时在两个站点上逐出节点、因为两个站点都不会有多数仲裁。
- 将表决文件放置在上 `diskgroup` 使用 `high` 如果一个站点上有两个磁盘、而另一个站点上有一个磁盘、则可以在两个站点均正常运行且可相互访问时执行主动-主动操作。但是、如果单磁盘站点与网络隔离、则该站点将被逐出。

RAC网络检测信号

Oracle RAC网络检测信号可监控集群互连中的节点可访问情况。要保留在集群中、一个节点必须能够与一半以上的其他节点联系。在双站点架构中、此要求会为RAC节点数创建以下选项：

- 如果在每个站点上放置相同数量的节点、则会在网络连接断开时在一个站点上执行逐出。
- 将N个节点放置在一个站点上、而将N+1个节点放置在另一个站点上、可以确保站点间连接断开会导致站点中剩余的网络仲裁节点数量增加、而将节点数量减少。

在Oracle 12cR2之前的版本中、无法控制站点丢失期间哪一端会发生逐出。如果每个站点的节点数相等、则逐出操作由主节点控制、主节点通常是要启动的第一个RAC节点。

Oracle 12cR2引入了节点加权功能。通过此功能、管理员可以更好地控制Oracle如何解决脑裂问题。例如、以下命令可为RAC中的特定节点设置首选项：

```
[root@host-a ~]# /grid/bin/crsctl set server css_critical yes
CRS-4416: Server attribute 'CSS_CRITICAL' successfully changed. Restart
Oracle High Availability Services for new value to take effect.
```

重新启动Oracle高可用性服务后、配置如下所示：

```
[root@host-a lib]# /grid/bin/crsctl status server -f | egrep
'^NAME|CSS_CRITICAL='
NAME=host-a
CSS_CRITICAL=yes
NAME=host-b
CSS_CRITICAL=no
```

Node `host-a` 现在指定为关键服务器。如果两个RAC节点彼此隔离、 `host-a` 不会影响、和 `host-b` 被逐出。

> 有关完整的详细信息、请参见Oracle白皮书《Oracle Clusterware 12c Release 2 Technical Overview》。"

对于12cR2之前的Oracle RAC版本、可通过按如下所示检查CRS日志来识别主节点：

```
[root@host-a ~]# /grid/bin/crsctl status server -f | egrep
'^NAME|CSS_CRITICAL='
NAME=host-a
CSS_CRITICAL=yes
NAME=host-b
CSS_CRITICAL=no
 [root@host-a ~]# grep -i 'master node' /grid/diag/crs/host-
a/crs/trace/crsd.trc
2017-05-04 04:46:12.261525 :   CRSSE:2130671360: {1:16377:2} Master Change
Event; New Master Node ID:1 This Node's ID:1
2017-05-04 05:01:24.979716 :   CRSSE:2031576832: {1:13237:2} Master Change
Event; New Master Node ID:2 This Node's ID:1
2017-05-04 05:11:22.995707 :   CRSSE:2031576832: {1:13237:221} Master
Change Event; New Master Node ID:1 This Node's ID:1
2017-05-04 05:28:25.797860 :   CRSSE:3336529664: {1:8557:2} Master Change
Event; New Master Node ID:2 This Node's ID:1
```

此日志指示主节点为 `2` 和节点 `host-a` ID为 `1`。这一事实意味着 `host-a` 不是主节点。可以使用命令确认主节点的标识 `olsnodes -n`。

```
[root@host-a ~]# /grid/bin/olsnodes -n
host-a  1
host-b  2
```

ID为的节点 `2` 为 `host-b`，即主节点。在每个站点上具有相同节点数的配置中、站点使用 `host-b` 是指在两组因任何原因丢失网络连接时仍可正常运行的站点。

标识主节点的日志条目可能会在系统中过期。在这种情况下、可以使用Oracle集群注册表(OCR)备份的时间戳。

```
[root@host-a ~]#  /grid/bin/ocrconfig -showbackup
host-b     2017/05/05 05:39:53     /grid/cdata/host-cluster/backup00.ocr
0
host-b     2017/05/05 01:39:53     /grid/cdata/host-cluster/backup01.ocr
0
host-b     2017/05/04 21:39:52     /grid/cdata/host-cluster/backup02.ocr
0
host-a     2017/05/04 02:05:36     /grid/cdata/host-cluster/day.ocr     0
host-a     2017/04/22 02:05:17     /grid/cdata/host-cluster/week.ocr     0
```

此示例显示主节点为 `host-b`。此外、它还表示主节点与发生了变化 `host-a` to `host-b` 5月4日2：05到21：39之间的某个时间。只有在检查了CRS日志后、才能安全地使用这种标识主节点的方法、因为主节点可能在上次OCR备份后发生更改。如果发生了此更改、则OCR日志中应该会显示此更改。

大多数客户都选择一个投票磁盘组来为整个环境提供服务、并在每个站点上选择相同数量的RAC节点。磁盘组

应放置在数据库所在的站点上。其结果是、连接断开会导致在远程站点上发生逐出。远程站点将不再具有仲裁、也无法访问数据库文件、但本地站点仍会照常运行。恢复连接后、远程实例可以重新联机。

发生灾难时、需要执行切换、以使运行正常的站点上的数据库文件和表决磁盘组联机。如果灾难允许AUSO触发切换、则不会触发NVFAIL、因为集群已知处于同步状态、并且存储资源正常联机。此操作速度非常快、应在之前完成 `disktimeout` 期限到期。

由于只有两个站点、因此无法使用任何类型的自动外部中断软件、这意味着强制切换必须手动操作。

三站点配置

使用三个站点构建扩展RAC集群更容易。托管MetroCluster系统一半的两个站点也支持数据库工作负载、而第三个站点则充当数据库和MetroCluster系统的断路器。Oracle TiebREAKER配置可能非常简单、只需将ASM磁盘组的一个成员放置在第三个站点上即可进行表决、也可能包括在第三个站点上运行的实例、以确保RAC集群中的节点数为奇数。

> 有关在扩展RAC配置中使用NFS的重要信息、请参阅Oracle文档中的"Quorum Failure group"(仲裁故障组)。总之、可能需要修改NFS挂载选项以包括软选项、以确保与托管仲裁资源的第三站点断开连接不会挂起主Oracle服务器或Oracle RAC进程。

SnapMirror活动同步

采用SnapMirror活动同步的Oracle数据库

SnapMirror主动同步可为各个Oracle数据库和应用程序环境实现选择性的RPO = 0同步镜像。

SnapMirror主动同步本质上是SAN的一项增强型SnapMirror功能、允许主机从托管LUN的系统以及托管其副本的系统访问LUN。

SnapMirror主动同步和SnapMirror同步共享一个复制引擎、但是、SnapMirror主动同步还包括一些附加功能、例如、企业级应用程序的透明应用程序故障转移和故障恢复。

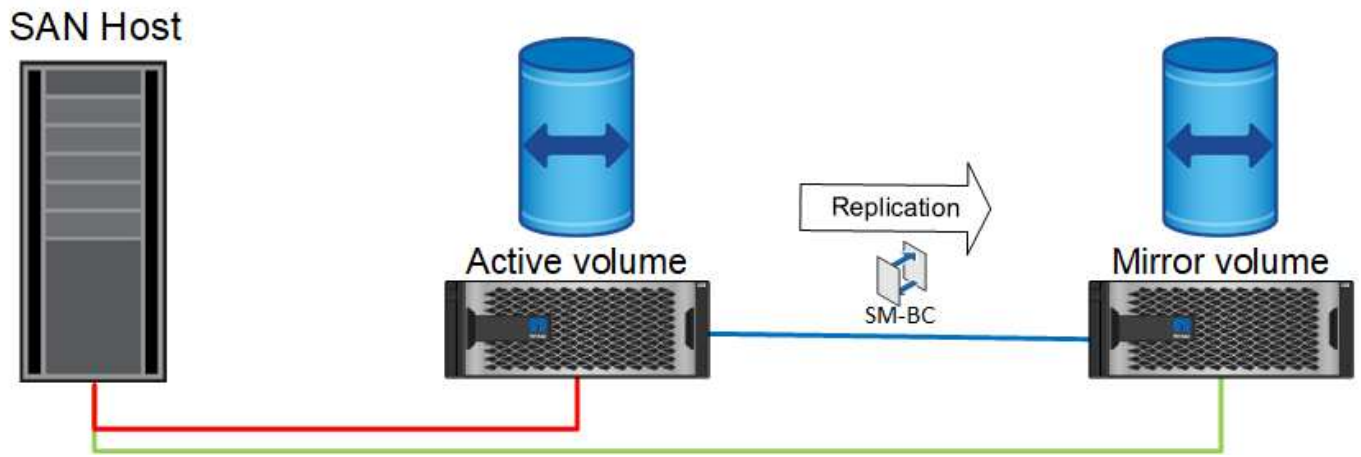
在实践中、它的工作原理与粒度版本的MetroCluster类似、它可以为各个工作负载启用选择性粒度RPO = 0同步复制。低级别的路径行为与MetroCluster截然不同、但从主机角度来看、最终结果是类似的。

路径访问

通过SnapMirror、活动同步使主存储阵列和远程存储阵列中的存储设备对主机操作系统可见。路径可通过非对称逻辑单元访问(AANAA)进行管理、ANAA是一种行业标准协议、用于确定存储系统与主机之间的优化路径。

访问I/O的最短设备路径被视为主动/优化路径、其余路径被视为主动/非优化路径。

SnapMirror活动同步关系位于不同集群上的一对SVM之间。这两个SVM都能够提供数据、但ALUA会优先使用当前拥有LUN所在驱动器所有权的SVM。远程SVM的IO将通过SnapMirror活动同步互连在中代理。



同步复制

在正常操作下、远程副本始终是RPO = 0的同步副本、但有一个例外。如果无法复制数据、则使用SnapMirror主动同步时、将不再需要复制数据并恢复提供IO。如果客户认为复制链路丢失是近乎灾难的情况、或者不希望在无法复制数据时业务运营暂停、则首选此选项。

存储硬件

与其他存储灾难恢复解决方案不同、SnapMirror主动同步可提供非对称平台灵活性。每个站点的硬件不必相同。通过此功能、您可以调整用于支持SnapMirror活动同步的硬件的大小。如果需要支持完整的生产工作负载、远程存储系统可以与主站点完全相同；但是、如果灾难导致I/O减少、则与远程站点上较小的系统相比、可能会更经济高效。

ONTAP调解器

ONTAP调解器是从NetApp支持下载的软件应用程序。调解器可自动执行主站点和远程站点存储集群的故障转移操作。它可以部署在内部或云端托管的小型虚拟机(VM)上。配置后、它将作为第三个站点来监控这两个站点的故障转移场景。

使用**SnapMirror**活动同步进行**Oracle**数据库故障转移

在SnapMirror活动同步上托管Oracle数据库的主要原因是、在计划内和计划外存储事件期间提供透明的故障转移。

SnapMirror主动同步支持两种类型的存储故障转移操作：计划内和计划外、其工作方式略有不同。管理员手动启动计划内故障转移、以便快速切换到远程站点、而计划外故障转移则由第三个站点上的调解器自动启动。计划内故障转移的主要目的是执行增量修补和升级、执行灾难恢复测试或采用一种正式策略、全年在站点之间切换操作、以证明完全的活动同步功能。

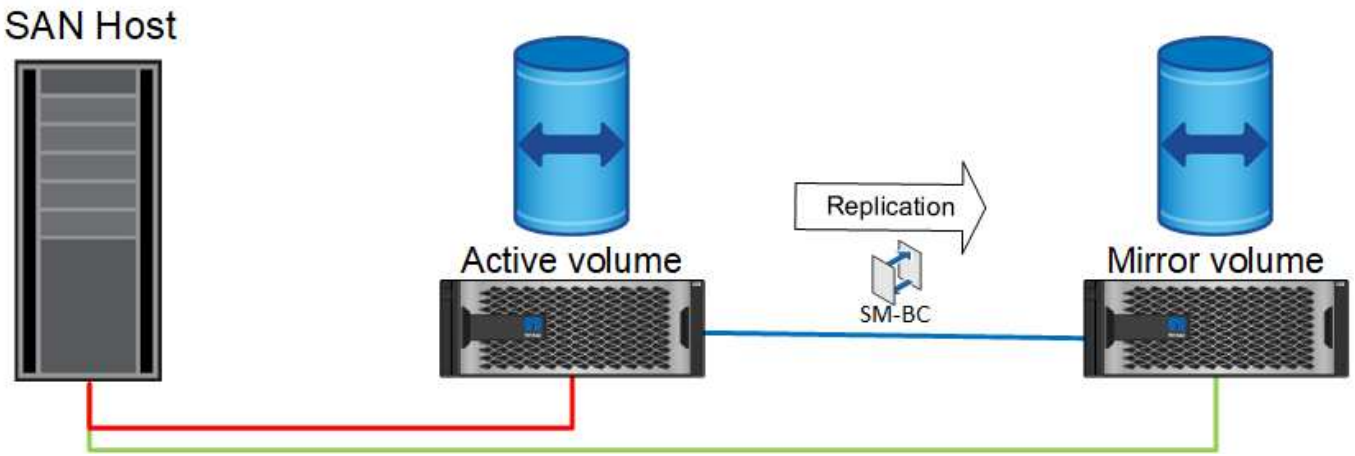
这些图显示了在正常操作、故障转移和故障恢复操作期间发生的情况。为便于说明、它们显示了一个复制的LUN。在实际的SnapMirror活动同步配置中、复制基于卷、其中每个卷包含一个或多个LUN、但为了简化情况、卷层已被删除。

正常运行

在正常操作下、可以从本地副本或远程副本访问LUN。红线表示ALOA公布的优化路径、结果应该是会优先沿着此路径发送IO。

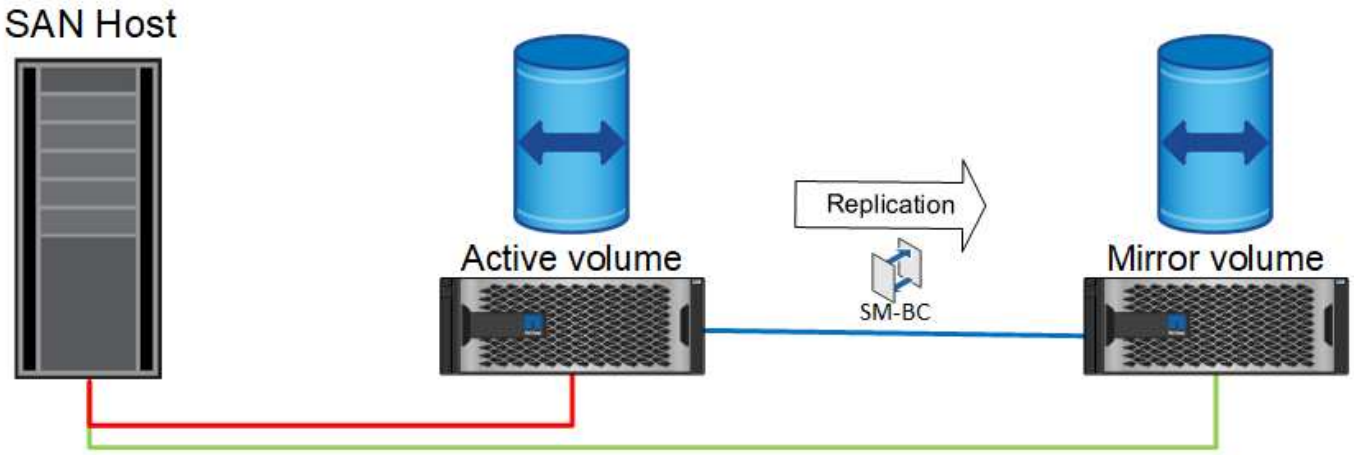
绿线是活动路径、但可能会产生较长的延迟、因为该路径上的IO需要通过SnapMirror活动同步路径传递。额外的

延迟取决于用于SnapMirror活动同步的站点之间的互连速度。



失败

如果活动镜像副本因计划内或计划外故障转移而变得不可用、则很明显、它将不再可用。但是、远程系统具有同步副本、并且指向远程站点的SAN路径已存在。远程系统能够为该LUN提供IO服务。

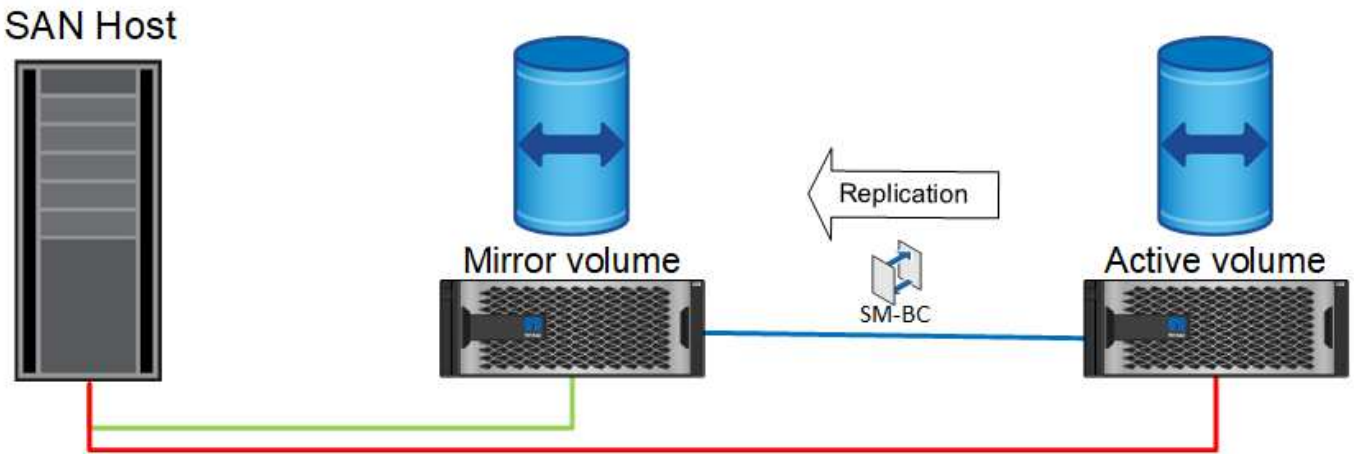


故障转移

故障转移会导致远程副本成为活动副本。这些路径将从"Active/Optimized"(活动/优化)更改为"Active/Optimized"(活动/优化)、并且IO将继续提供服务、而不会丢失数据

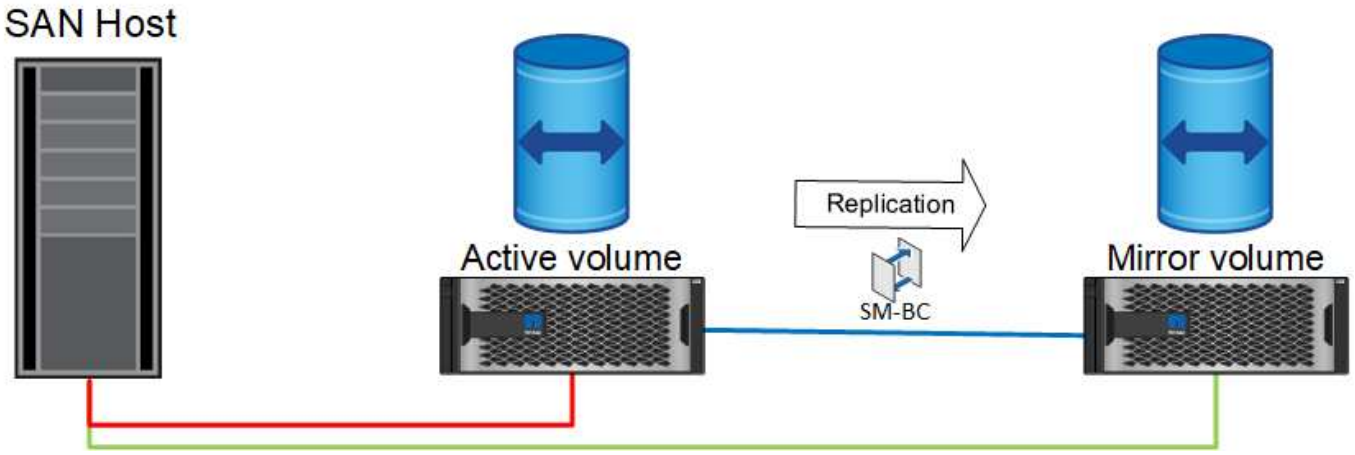
修复

源系统恢复使用后、SnapMirror活动同步可以重新同步复制、但运行的方向相反。现在、配置与起始点基本相同、只是已翻转主动镜像站点。



故障恢复

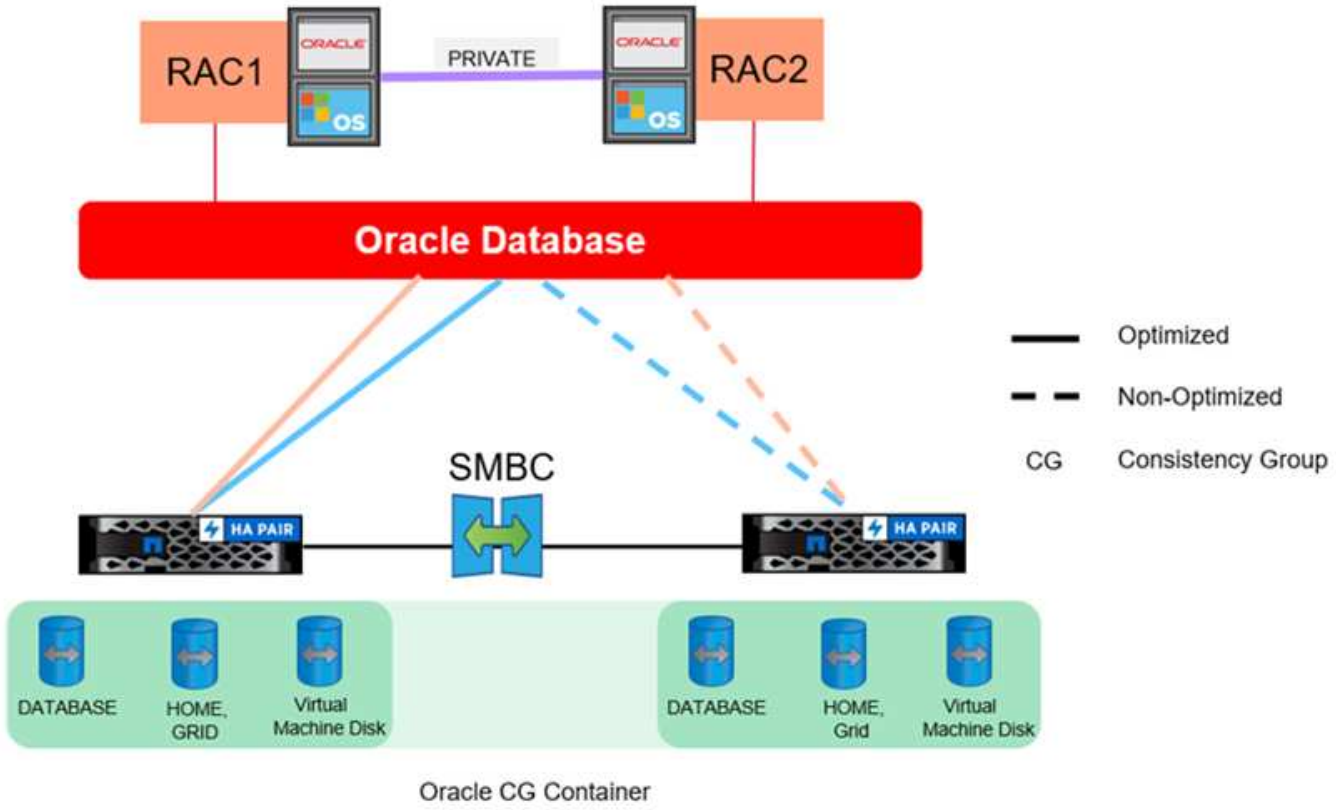
如果需要、管理员可以执行故障恢复、并将LUN的活动副本移回原始控制器。



采用**SnapMirror**活动同步的单实例**Oracle**数据库

下图显示了一种简单的部署模式、其中、存储设备会从Oracle数据库的主存储集群和远程存储集群进行分区或连接。

Oracle仅在主系统上配置。此模式可在发生存储端灾难时实现无缝存储故障转移、不会丢失任何数据、也不会造成任何应用程序停机。但是、在站点发生故障期间、此模型不会提供数据库环境的高可用性。这种类型的架构对于希望获得零数据丢失解决方案且存储服务具有高可用性的客户非常有用、但客户也认为数据库集群完全丢失需要手动操作。



这种方法还可以节省Oracle许可成本。在远程站点上预配置Oracle数据库节点需要根据大多数Oracle许可协议为所有核心授予许可。如果可以接受因安装Oracle数据库服务器和挂载无故障数据副本所需时间而导致的延迟、则此设计可能非常经济高效。

具有SnapMirror活动同步的Oracle RAC

SnapMirror主动同步可精细控制数据集复制、以实现负载平衡或单个应用程序故障转移等目的。整体架构看起来像一个扩展的RAC集群、但某些数据库专用于特定站点、整体负载是分布式的。

例如、您可以构建一个Oracle RAC集群来托管六个单独的数据库。其中三个数据库的存储主要托管在站点A上、其余三个数据库的存储将托管在站点B上此配置可最大限度地减少跨站点流量、从而确保性能最佳。此外、应用程序将配置为使用具有活动路径的存储系统本地数据库实例。这样可以最大程度地减少RAC互连流量。最后、这种整体设计可确保所有计算资源的使用均匀。随着工作负载的变化、数据库可以有选择地在各个站点之间来回故障转移、以确保负载均匀。

除了粒度之外、使用SnapMirror主动的Oracle RAC的基本原则和选项与相同 "基于MetroCluster的Oracle RAC"

Oracle数据库和SnapMirror活动同步失败情形

存在多种SnapMirror活动同步(SM-AS)故障情形、每种情形的结果各不相同。

场景	结果
复制链路故障	调解器可识别这种脑裂情形、并恢复持有主副本的节点上的I/O。当站点之间的连接恢复联机后、备用站点将执行自动重新同步。

场景	结果
主站点存储故障	调解器会启动自动计划外故障转移。 无I/O中断。
远程站点存储故障	不会造成I/O中断。由于网络导致同步复制中止、并且主节点确定它是继续提供I/O (协商一致)的合法所有者、因此会出现短暂的暂停。因此、I/O会暂停几秒钟、然后I/O将恢复。 站点联机时会自动重新同步。
调解器丢失或调解器与存储阵列之间的链路丢失	I/O将继续并保持与远程集群同步、但如果没有调解器、则无法自动执行计划外/计划内故障转移和故障恢复。
丢失HA集群中的一个存储控制器	HA集群中的配对节点尝试接管(NDo)。如果接管失败、调解器会注意到存储中的两个节点均已关闭、并自动向远程集群执行计划外故障转移。
磁盘丢失	IO最多会连续发生三次磁盘故障。这是RAID-TEC的一部分。
在典型部署中丢失整个站点	故障站点上的服务器显然将不再可用。支持集群的应用程序可以配置为在两个站点上运行、并在备用站点上继续运行、但大多数此类应用程序都需要第三个站点的Tieb破碎 机、就像SM-AS需要调解器一样。 如果没有应用程序级集群、则需要在运行正常的站点上启动应用程序。这会影响可用性、但会保留RPO = 0。不会丢失任何数据。

Oracle数据库迁移

将Oracle数据库迁移到ONTAP存储系统

利用新存储平台的功能有一个不可避免的要求：必须将数据放置在新存储系统上。ONTAP简化了迁移过程、包括ONTAP到ONTAP的迁移和升级、外部LUN导入以及直接使用主机操作系统或Oracle数据库软件的过程。

> 本文档将取代先前发布的技术报告_TR-4534：《将Oracle数据库迁移到NetApp存储系统》_

对于新的数据库项目、这不是问题、因为数据库和应用程序环境已构建到位。但是、迁移在业务中断、完成迁移所需的时间、所需的技能组合以及风险最小化方面带来了特殊挑战。

脚本

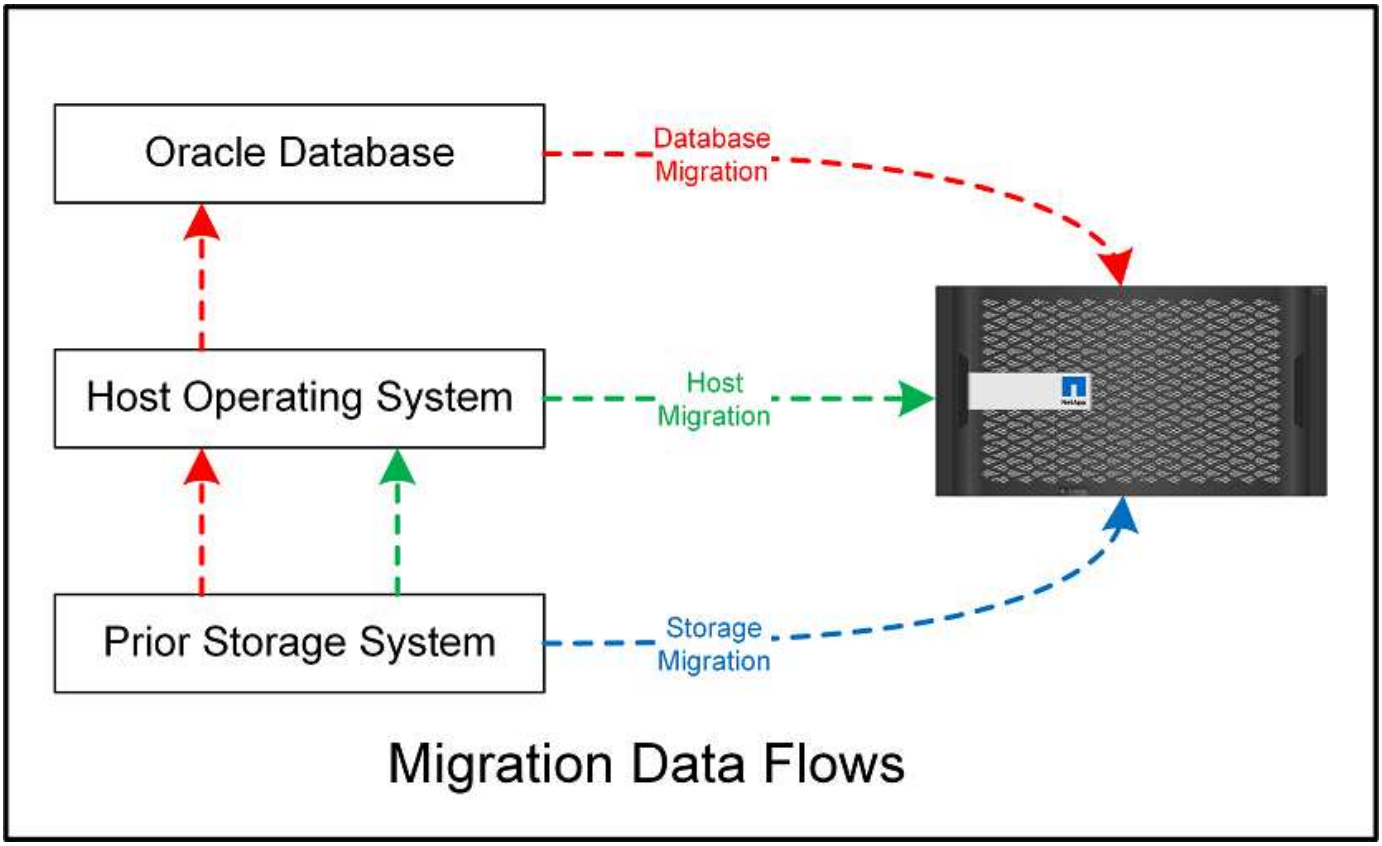
本文档提供了示例脚本。这些脚本提供了一些示例方法、用于自动执行各个方面的迁移、以降低出现用户错误的几率。这些脚本可以降低对负责迁移的IT人员的总体需求、并加快整个过程。这些脚本全部取自NetApp专业服务和NetApp合作伙伴执行的实际迁移项目。本文档通篇展示了这些参数的用法示例。

Oracle数据库迁移规划

Oracle数据迁移可在以下三个级别之一进行：数据库、主机或存储阵列。

不同之处在于、整个解决方案的哪个组件负责移动数据：数据库、主机操作系统或存储系统。

下图显示了一个迁移级别和数据流示例。在数据库级别迁移的情况下、数据会从原始存储系统通过主机层和数据库层移动到新环境中。主机级迁移类似、但数据不会通过应用层、而是使用主机进程写入新位置。最后、对于存储级别迁移、NetApp FAS系统等阵列负责数据移动。



数据库级迁移通常是指通过备用数据库进行Oracle日志传送、以便在Oracle层完成迁移。主机级迁移可通过主机操作系统配置的本机功能来执行。此配置包括使用cp、tar和Oracle Recovery Manager (RMAN)等命令执行文件复制操作、或者使用逻辑卷管理器(LVM)重新定位文件系统的底层字节。Oracle自动存储管理(Automatic Storage Management、ASM)归类为主机级功能、因为它在数据库应用程序级别以下运行。ASM取代主机上常用的逻辑卷管理器。最后、数据可以在存储阵列级别进行迁移、这意味着可以在操作系统级别以下进行迁移。

规划注意事项

迁移的最佳选择取决于多种因素的组合、包括要迁移的环境的规模、避免停机的需求以及执行迁移所需的整体工作量。大型数据库显然需要更多的时间和精力进行迁移、但这种迁移的复杂性微乎其微。小型数据库可以快速迁移、但是、如果要迁移的数据库有数千个、工作的规模可能会带来复杂性。最后、数据库越大、越有可能成为业务关键型数据库、因此需要在保持备用路径的同时最大限度地减少停机时间。

此处将讨论规划迁移策略时的一些注意事项。

数据大小

要迁移的数据库的大小显然会影响迁移规划、但大小并不一定会影响转换时间。当需要迁移大量数据时、首要考虑因素是带宽。复制操作通常使用高效的顺序I/O来执行保守地估计、假设复制操作可用网络带宽的利用率为50%。例如、一个8 GB FC端口在理论上可以传输大约800 MBps。假设利用率为50%、则可以以大约400 Mbps的速率复制数据库。因此、以这种速率在大约7小时内即可复制一个10 TB的数据库。

远距离迁移通常需要更具创意的方法、例如中所述的日志传送过程 "联机数据文件移动"。远距离IP网络的带宽很少接近LAN或SAN速度。在一个案例中、NetApp以极高的归档日志生成速率协助远程迁移220 TB数据库。选择的数据传输方法是每天运送磁带、因为这种方法可提供最大可能的带宽。

数据库计数

在许多情况下、移动大量数据的问题不在于数据大小、而在于支持数据库的配置的复杂性。仅仅知道必须迁移50 TB的数据库是不够的。它可以是一个50 TB的任务关键型数据库、4、000个原有数据库的集合、也可以是生产数据和非生产数据的混合。在某些情况下、大部分数据都由源数据库的克隆组成。这些克隆根本不需要迁移、因为它们可以轻松地重新创建、尤其是在新架构设计为利用NetApp FlexClone卷时。

对于迁移规划、您必须了解范围内有多少数据库以及必须如何确定这些数据库的优先级。随着数据库数量的增加、堆栈中的首选迁移选项往往会越来越少。例如、使用RMAN可以轻松复制单个数据库、但会短暂中断。这是主机级复制。

如果有50个数据库、则可能更容易避免设置新的文件系统结构来接收RMAN副本并将数据移动到位。可以通过利用基于主机的LVM迁移将数据从旧LUN重新定位到新LUN来完成此过程。这样、数据库管理员(Database Administrator、DBA)团队就会将职责移交给操作系统团队、因此、数据会相对于数据库透明地进行迁移。文件系统配置保持不变。

最后、如果必须迁移200个服务器中的500个数据库、则可以使用ONTAP外部LUN导入(FLI)功能等基于存储的选项来直接迁移LUN。

重新架构要求

通常、必须更改数据库文件布局才能利用新存储阵列的功能；但是、情况并非总是如此。例如、EF系列全闪存阵列的功能主要针对SAN性能和SAN可靠性。在大多数情况下、数据库可以迁移到EF系列阵列、而无需考虑数据布局的特殊注意事项。唯一的要求是高IOPS、低延迟和强大的可靠性。尽管存在与RAID配置或动态磁盘池等因素相关的最佳实践、但EF系列项目很少需要对整体存储架构进行任何重大更改才能利用这些功能。

相比之下、迁移到ONTAP通常需要更多地考虑数据库布局、以确保最终配置实现最大价值。ONTAP本身就可以为数据库环境提供许多功能、即使不需要任何特定的架构工作也是如此。最重要的是、它能够在当前硬件达到使用寿命时无故障迁移到新硬件。一般来说、迁移到ONTAP是您最后一次需要执行的迁移。后续硬件会原位升级、数据会无中断迁移到新介质。

通过一些规划、可以获得更多优势。有关快照的使用、最重要的注意事项是。快照是执行近乎即时的备份、还原和克隆操作的基础。作为快照功能的一个示例、已知的最大用途是在6个控制器上的大约250个LUN上运行一个996 TB的数据库。此数据库可以在2分钟内完成备份、在2分钟内完成还原、在15分钟内完成克隆。其他优势包括：能够根据工作负载的变化在集群中移动数据、以及应用服务质量(QoS)控制、以便在多数据库环境中提供稳定一致的良好性能。

QoS控制、数据重新定位、快照和克隆等技术几乎适用于任何配置。但是、通常需要考虑一些问题才能获得最大的益处。在某些情况下、数据库存储布局可能需要进行设计更改、才能最大程度地提高对新存储阵列的投资。此类设计更改可能会影响迁移策略、因为基于主机或基于存储的迁移会复制原始数据布局。要完成迁移并提供针对ONTAP优化的数据布局、可能还需要执行其他步骤。中所示的过程 "Oracle迁移过程概述" 之后、我们将演示一些方法、这些方法不仅可以迁移数据库、还可以轻松地将数据库迁移到最佳最终布局中。

转换时间

应确定转换期间允许的最大服务中断时间。假设整个迁移过程会造成中断、这是一个常见错误。许多任务都可以在任何服务中断开始之前完成、而且许多选项都支持在不中断或中断的情况下完成迁移。即使不可避免地发生中断、您仍必须定义允许的最大服务中断、因为转换时间的持续时间因操作步骤而异操作步骤。

例如、复制一个10 TB数据库通常需要大约7小时才能完成。如果业务需求允许中断七小时、则文件复制是一种简单安全的迁移选项。如果五个小时不可接受、则采用简单的日志传送流程(请参见 "Oracle日志传送")、只需极少的工作量即可完成设置、从而将转换时间缩短至大约15分钟。在此期间、数据库管理员可以完成此过程。如果15分钟不可接受、则可以通过脚本自动执行最终转换过程、将转换时间缩短为几分钟。您始终可以加快迁移速度、但这样做会耗费时间和精力。转换时间目标应基于业务部门可接受的内容。

回退路径

任何迁移都不是完全无风险的。即使技术运行正常、也始终存在用户错误的可能性。必须考虑与所选迁移路径相关的风险以及迁移失败的后果。例如、Oracle ASM的透明联机存储迁移功能是其主要功能之一、而这种方法是已知最可靠的方法之一。但是、使用此方法可以不可逆地复制数据。如果ASM出现问题的可能性极小、则不存在轻松的回退路径。唯一的选择是还原原始环境或使用ASM将迁移反转回原始LUN。如果原始存储系统能够执行快照类型的备份、则可以通过在该系统上执行此类操作来最大程度地降低风险、但无法消除此类风险。

排练

某些迁移过程在执行前必须进行全面验证。迁移和演练转换过程是任务关键型数据库的常见要求、对于这些数据库、迁移必须成功、停机时间必须降至最低。此外、用户验收测试通常会作为迁移后工作的一部分、整个系统只有在这些测试完成后才能恢复生产。

如果需要预演、几项ONTAP功能可以使流程更加简单。特别是、快照可以重置测试环境、并快速为数据库环境创建多个节省空间的副本。

过程

Oracle迁移过程概述

Oracle迁移数据库可以执行许多过程。选择合适的解决方案取决于您的业务需求。

在许多情况下、系统管理员和数据库管理员都有自己的首选方法来重新定位物理卷数据、镜像和脱机或利用Oracle RMAN复制数据。

这些过程主要是为不熟悉某些可用选项的IT员工提供的指导。此外、这些过程还说明了每种迁移方法的任务、时间要求和技能要求。这样、NetApp和合作伙伴专业服务或IT管理人员等其他方就可以更充分地了解每个操作步骤的要求。

制定迁移策略没有单一的最佳实践。创建计划需要首先了解可用性选项、然后选择最适合业务需求的方法。下图显示了客户的基本注意事项和典型结论、但并非普遍适用于所有情况。

例如、一个步骤可提高数据库总大小的问题描述。下一步取决于数据库是大于还是小于1 TB。建议的步骤就是根据典型客户实践提出的建议。大多数客户不会使用DataGuard复制小型数据库、但有些客户可能会使用。由于所需时间较长、大多数客户不会尝试复制50 TB的数据库、但有些客户可能有足够大的维护窗口来执行此类操作。

您可以查看迁移路径最佳注意事项类型的流程图 "此处"。

联机数据文件移动

Oracle 12cR1及更高版本可在数据库保持联机状态时移动数据文件。此外、它还适用于不同的文件系统类型。例如、可以将数据文件从xfs文件系统重新定位到ASM。由于需要执行的单个数据文件移动操作的数量众多、因此通常不会大规模使用此方法、但对于数据文件较少的小型数据库、这是一个值得考虑的选项。

此外、对于迁移现有数据库的部分内容来说、只需移动数据文件是一个很好的选择。例如、活动较少的数据文件可以重新定位到更经济高效的存储、例如FabricPool卷、该卷可以将空闲块存储在对象存储中。

数据库级迁移

数据库级别的迁移意味着允许数据库重新定位数据。具体而言、这意味着日志传送。RMAN和ASM等技术是Oracle产品、但出于迁移目的、它们在主机级别运行、在主机级别复制文件和管理卷。

日志传送

数据库级迁移的基础是Oracle归档日志、其中包含数据库更改的日志。大多数情况下、归档日志是备份和恢复策略的一部分。恢复过程首先会还原数据库、然后重播放一个或多个归档日志、以使数据库达到所需状态。这种基本技术也可用于执行迁移、操作中断极少甚至不会中断。更重要的是、此技术支持迁移、同时保持原始数据库不变、并保留一条回退路径。

迁移过程从将数据库备份还原到二级服务器开始。您可以通过多种方式执行此操作、但大多数客户都使用其常规备份应用程序来还原数据文件。还原数据文件后、用户将建立日志传送方法。目标是为主数据库生成的归档日志创建一个持续源、并在还原的数据库中重放这些日志、以使它们接近相同的状态。转换时间到后、源数据库将完全关闭、最终归档日志(在某些情况下、重做日志)将被复制并重排。重做日志也要予以考虑、这一点非常重要、因为它们可能包含已提交的某些最终事务。

在传输和回显示这些日志后、两个数据库将保持一致。此时、大多数客户都会执行一些基本测试。如果在迁移过程中发生任何错误、则日志重放应报告错误并失败。仍然建议根据已知查询或应用程序驱动型活动执行一些快速测试、以验证配置是否最佳。在关闭初始数据库之前创建一个最终测试表、以验证迁移的数据库中是否存在该表、这也是一种常见做法。此步骤可确保在最终日志同步期间不会发生任何错误。

可以针对原始数据库配置简单的日志传送迁移、这对于任务关键型数据库尤其有用。源数据库不需要更改配置、迁移环境的还原和初始配置对生产操作没有影响。配置日志传送后、它会对生产服务器提出一些I/O需求。但是、日志传送包含对归档日志的简单顺序读取、这不太可能对生产数据库性能产生任何影响。

事实证明、日志传送对于远距离、高变更率的迁移项目特别有用。在一个实例中、一个220 TB的数据库迁移到了大约500英里以外的新位置。更改率极高、而且安全限制阻止了使用网络连接。日志传送是使用磁带和信使执行的。源数据库的副本最初是使用下面概述的过程进行还原的。然后、派送员每周发送一次日志、直至转换时交付最后一组磁带并将日志应用于副本数据库。

Oracle DataGuard

在某些情况下、需要一个完整的DataGuard环境。使用术语DataGuard来指代任何日志传送或备用数据库配置是不正确的。Oracle DataGuard是一种用于管理数据库复制的全面框架、但它不是一种复制技术。在迁移过程中、完整的DataGuard环境的主要优势是可以从一个数据库透明地切换到另一个数据库。此外、如果发现问题(例如新环境的性能或网络连接问题描述)、Dataguard还可以透明地切回原始数据库。完全配置的DataGuard环境不仅需要配置数据库层、还需要配置应用程序、以便应用程序能够检测到主数据库位置的更改。一般来说、不需要使用DataGuard完成迁移、但一些客户在内部拥有丰富的DataGuard专业知识、并已依赖它来执行迁移工作。

重新构建

如前文所述、利用存储阵列的高级功能有时需要更改数据库布局。此外、存储协议的更改(例如从ASM迁移

到NFS文件系统)也必然会更改文件系统布局。

日志传送方法(包括DataGuard)的主要优势之一是、复制目标不必与源匹配。使用日志传送方法从ASM迁移到常规文件系统时没有问题、反之亦然。可以在目标位置更改数据文件的精确布局、以优化可插拔数据库(PDB)技术的使用、或者有选择地对某些文件设置QoS控制。换言之、基于日志传送的迁移过程可让您轻松安全地优化数据库存储布局。

服务器资源

数据库级迁移的一个限制是需要另一台服务器。可以通过两种方式使用第二台服务器：

1. 您可以使用第二台服务器作为数据库的永久新主目录。
2. 您可以使用第二个服务器作为临时暂存服务器。完成向新存储阵列的数据迁移并进行测试后、LUN或NFS文件系统将与暂存服务器断开连接、并重新连接到原始服务器。

第一种选择最简单、但在需要非常强大的服务器的超大型环境中使用它可能并不可行。第二个选项需要额外的工作才能将文件系统重新定位回原始位置。这可能是一个简单的操作、其中会使用NFS作为存储协议、因为文件系统可以从暂存服务器上卸载、然后重新挂载到原始服务器上。

基于块的文件系统需要额外的工作才能更新FC分区或iSCSI启动程序。使用大多数逻辑卷管理器(包括ASM)时、系统会自动检测LUN、并在原始服务器上提供这些LUN后将其置于联机状态。但是、某些文件系统和LVM实施可能需要更多的工作才能导出和导入数据。确切的操作步骤可能有所不同、但通常很容易建立一个简单、可重复的操作步骤来完成迁移并将数据重新归位到原始服务器上。

虽然可以在单个服务器环境中设置日志传送和复制数据库、但新实例必须具有不同的进程SID才能重放日志。可以使用不同的SID临时启动另一组进程ID下的数据库、并在以后进行更改。但是、这样做可能会导致许多复杂的管理活动、并使数据库环境面临用户错误的风险。

主机级迁移

在主机级别迁移数据意味着使用主机操作系统和相关实用程序完成迁移。此过程包括复制数据的任何实用程序、包括Oracle RMAN和Oracle ASM。

数据复制

不应低估简单复制操作的价值。现代网络基础架构可以按每秒千兆字节的速率移动数据、文件复制操作基于高效的顺序读写I/O与日志传送相比、主机复制操作不可避免地会造成更多中断、但迁移不仅仅是数据移动。它通常包括对网络连接、数据库重新启动时间以及迁移后测试的更改。

复制数据所需的实际时间可能不多。此外、复制操作会保留有保障的回退路径、因为原始数据不会受到影响。如果在迁移过程中遇到任何问题、可以重新激活包含原始数据的原始文件系统。

重新平台化

重新平台是指CPU类型的变化。将数据库从传统Solaris、AIX或HP-UX平台迁移到x86 Linux时、由于CPU架构发生更改、必须重新格式化数据。SPARC、IA64和POWER CPU称为大的恩第处理器、而x86和x86_64架构称为小恩第处理器。因此、根据所使用的处理器、Oracle数据文件中的某些数据的顺序会有所不同。

过去、客户一直使用DataPump跨平台复制数据。数据缓冲是一种实用程序、用于创建特殊类型的逻辑数据导出、可以在目标数据库中更快地导入。由于DataPump会为数据创建一个逻辑副本、因此会将处理器数据存储单的依赖关系置于身后。某些客户仍在使用数据缓冲区进行回滚、但Oracle 11g提供了一个速度更快的选项：跨平台可传输表空间。这种高级允许将表空间转换为不同的在位的字符格式。这是一种物理转换、其性能优于DataPump导出、DataPump导出必须先将物理字节转换为逻辑数据、然后再转换回物理字节。

有关DataPump和可传输表空间的完整讨论不在NetApp文档的讨论范围内、但NetApp根据我们在使用新CPU架构向新存储阵列日志迁移期间为客户提供帮助的经验提供了一些建议：

- 如果正在使用DataPump、则应在测试环境中测量完成迁移所需的时间。客户有时会对完成迁移所需的时间感到惊讶。这种意外的额外停机可能会导致发生原因中断。
- 许多客户误以为跨平台可传输表空间不需要数据转换。如果使用具有不同ENDE的CPU、则为RMAN `convert` 必须事先对数据文件执行操作。这不是瞬时操作。在某些情况下、可以通过在不同数据文件上运行多个线程来加快转换过程、但无法避免该转换过程。

逻辑卷管理器驱动的迁移

LVM的工作原理是、创建一组LUN (由一个或多个LUN组成)并将其拆分为通常称为块区的小单元。然后、块区池将用作源、用于创建从本质上进行虚拟化的逻辑卷。此虚拟化层可通过多种方式提供价值：

- 逻辑卷可以使用从多个LUN中绘制的块区。在逻辑卷上创建文件系统时、该文件系统可以使用所有LUN的全部性能功能。此外、它还可以均匀加载卷组中的所有LUN、从而提供更具可预测性的性能。
- 可以通过添加和在某些情况下删除块区来调整逻辑卷的大小。在逻辑卷上调整文件系统大小通常不会造成中断。
- 通过移动底层块区、可以无干扰地迁移逻辑卷。

使用LVM进行迁移的工作方式有两种：移动块区或镜像/取消块区镜像。LVM迁移使用高效的大型块顺序I/O、很少会产生任何性能问题。如果这确实成为问题描述、通常可以选择限制I/O速率。这样做不仅会增加完成迁移所需的时间、还会减轻主机和存储系统的I/O负担。

镜像和镜像

某些卷管理器(如AIX LVM)允许用户指定每个块区的副本数、并控制托管每个副本的设备。迁移的方法是：创建一个现有逻辑卷、将底层块区镜像到新卷、等待副本同步、然后删除旧副本。如果需要回退路径、则可以在删除镜像副本之前创建原始数据的快照。或者、也可以在强制删除包含的镜像副本之前短暂关闭服务器以屏蔽原始LUN。这样做会将数据的可恢复副本保留在其原始位置。

块区迁移

几乎所有卷管理器都允许迁移块区、有时还存在多个选项。例如、某些卷管理器允许管理员将特定逻辑卷的各个块区从旧存储重新定位到新存储。Linux LVM2等卷管理器提供 `pvmove` 命令、用于将指定LUN设备上的所有块区重新定位到新LUN。清空旧LUN后、可以将其删除。

> 操作面临的主要风险是从配置中删除未使用的旧LUN。更改FC分区和删除陈旧的LUN设备时必须格外小心。

Oracle自动存储管理

Oracle ASM是逻辑卷管理器和文件系统的组合。从较高层面来看、Oracle ASM会获取一组LUN、将其划分为多个小的分配单元、并将其呈现为一个称为ASM磁盘组的卷。ASM还可以通过设置冗余级别来镜像磁盘组。卷可以是未镜像(外部冗余)、镜像(正常冗余)或三向镜像(高冗余)。配置冗余级别时必须小心、因为创建后无法更改。

ASM还提供文件系统功能。尽管文件系统不会直接从主机中显示、但Oracle数据库可以在ASM磁盘组上创建、移动和删除文件和目录。此外、还可以使用asmcmd实用程序来导航此结构。

与其他LVM实施方式一样、Oracle ASM通过在所有可用LUN之间对每个文件的I/O进行条带化和负载平衡来优化I/O性能。其次、可以重新定位底层块区、以便调整ASM磁盘组的大小以及进行迁移。Oracle ASM可通过重新

平衡操作自动执行此过程。新的LUN将添加到ASM磁盘组、而旧的LUN将被丢弃、这将触发块区重新定位、并随后将清空的LUN从磁盘组中删除。此过程是经验证的迁移方法之一、ASM在提供透明迁移方面的可靠性可能是其最重要的功能。

> 由于Oracle ASM的镜像级别是固定的、因此不能与镜像和镜像迁移方法结合使用。

存储级别迁移

存储级别迁移是指在应用程序和操作系统级别以下执行迁移。过去、这有时意味着需要使用专用设备在网络级别复制LUN、但这些功能现在已在ONTAP本机提供。

SnapMirror

几乎可以使用NetApp SnapMirror数据复制软件在NetApp系统之间执行数据库迁移。此过程涉及到为要迁移的卷设置镜像关系、允许这些卷进行同步、然后等待转换窗口。到达后、源数据库将关闭、并执行一次最终镜像更新、同时镜像将断开。然后、可以通过挂载包含的NFS文件系统目录或发现包含的LUN并启动数据库来准备好使用副本卷。

在单个ONTAP集群中重新定位卷不会视为迁移、而是一项例行操作 `volume move` 操作。SnapMirror用作集群中的数据复制引擎。此过程完全自动化。当卷的属性(例如LUN映射或NFS导出权限)随卷本身一起移动时、无需执行其他迁移步骤。重新定位不会中断主机操作。在某些情况下、必须更新网络访问、以确保以尽可能最高效的方式访问新重新定位的数据、但这些任务也不会造成中断。

外部**LUN**导入**(FLI)**

FLI功能允许运行8.3或更高版本的Data ONTAP系统从另一个存储阵列迁移现有LUN。操作步骤非常简单：ONTAP系统像任何其他SAN主机一样分区到现有存储阵列。然后、Data ONTAP会控制所需的原有LUN并迁移底层数据。此外、导入过程会在迁移数据时使用新卷的效率设置、这意味着可以在迁移过程中对数据进行实时压缩和重复数据删除。

首次在Data ONTAP 8.3中实施FLI时、仅允许脱机迁移。虽然传输速度非常快、但这仍意味着在迁移完成之前LUN数据不可用。联机迁移是在Data ONTAP 8.3.1中推出的。此类迁移可使ONTAP在传输过程中提供LUN数据、从而最大限度地减少中断。重新分区主机以通过ONTAP使用LUN时、会发生短暂中断。但是、一旦进行了这些更改、数据就可以再次访问、并且在整个迁移过程中始终可以访问。

读取I/O会通过ONTAP代理、直到复制操作完成、而写入I/O会同时写入外部LUN和ONTAP LUN。这两个LUN副本将以这种方式保持同步、直到管理员执行完全转换以释放外部LUN且不再复制写入。

FLI可与FC结合使用、但如果需要更改为iSCSI、则迁移的LUN可以在迁移完成后轻松地重新映射为iSCSI LUN。

FLI的功能包括自动对齐检测和调整。在此上下文中、术语对齐是指LUN设备上的分区。要获得最佳性能、需要将I/O与4K块对齐。如果将分区放置在非4 k倍数的偏移位置、则会影响性能。

对齐的第二个方面无法通过调整分区偏移量(文件系统块大小)来更正。例如，ZFS文件系统通常默认为内部块大小512字节。使用AIX的其他客户偶尔会创建块大小为512字节或1、即1、即1、0 4字节的JFS2文件系统。尽管文件系统可能会与4 k边界对齐、但在该文件系统中创建的文件不会对齐、性能会受到影响。

在这些情况下、不应使用FLI。尽管迁移后可以访问数据、但结果是文件系统存在严重的性能限制。一般来说、在ONTAP上支持随机覆盖工作负载的任何文件系统都应使用4 k块大小。这主要适用于数据库数据文件和VDI部署等工作负载。可以使用相关的主机操作系统命令来确定块大小。

例如、在AIX上、可以使用查看块大小 `lsfs -q`。使用Linux、 `xfs_info` 和 `tune2fs` 可用于 `xfs` 和

`ext3/ext4`。使用 `zfs`，则命令为 `zdb -C`。

用于控制块大小的参数为 `ashift` 通常默认为9、表示2^9或512字节。为了获得最佳性能、 `ashift` 值必须为12 (2^12=4k)。此值在创建zpool时设置、并且无法更改、这意味着数据zpool具有 `ashift` 应通过将数据复制到新创建的zpool来迁移12以外的文件。

Oracle ASM没有基本块大小。唯一的要求是构建ASM磁盘的分区必须正确对齐。

7-模式过渡工具

7-模式过渡工具(7MTT)是一款自动化实用程序、用于将大型7-模式配置迁移到ONTAP。大多数数据库客户发现其他方法更容易、部分原因是他们通常会逐个数据库迁移环境数据库、而不是重新定位整个存储占用空间。此外、数据库通常只是大型存储环境的一部分。因此、数据库通常会单独迁移、然后可以使用7MTT移动其余环境。

有少数客户拥有专用于复杂数据库环境的存储系统、但数量相当多。这些环境可能包含许多卷、快照和大量配置详细信息、例如导出权限、LUN启动程序组、用户权限和轻型目录访问协议配置。在这种情况下、7MTT的自动化功能可以简化迁移。

7MTT可在以下两种模式之一下运行：

- *基于副本的过渡(CBT)。*采用CBT的7MTT可在新环境中从现有7-模式系统设置SnapMirror卷。数据同步后、7MTT会编排转换过程。
- *无副本过渡(CFT)。*采用CFT的7MTT基于现有7-模式磁盘架的原位转换。不会复制任何数据、现有磁盘架可以重复使用。保留现有数据保护和存储效率配置。

这两种方案之间的主要区别在于、无副本过渡是一种大爆炸方法、在这种方法中、连接到原始7-模式HA对的所有磁盘架都必须重新定位到新环境。无法移动部分磁盘架。基于副本的方法允许移动选定卷。此外、无副本过渡的转换窗口可能会更长、因为重新对磁盘架进行转换和转换元数据需要关联。根据现场经验、NetApp建议留出1小时的时间来重新定位磁盘架并重新为其接通网络、而留出15分钟到2小时的时间来进行元数据转换。

Oracle数据文件迁移

单个Oracle数据文件只需使用一个命令即可移动。

例如、以下命令将数据文件IOPST.dbf从文件系统中移动 `/oradata2` 文件系统 `/oradata3`。

```
SQL> alter database move datafile  '/oradata2/NTAP/IOPS002.dbf' to
'/oradata3/NTAP/IOPS002.dbf';
Database altered.
```

使用此方法移动数据文件可能会很慢、但通常不会产生足够的I/O、以致会干扰日常数据库工作负载。相比之下、通过ASM重新平衡进行迁移的速度会快得多、但代价是在移动数据时降低整个数据库的运行速度。

可以通过创建测试数据文件并将其移动来轻松衡量移动数据文件所需的时间。操作所用时间记录在v$session数据中：

```
SQL> set linesize 300;
SQL> select elapsed_seconds||':'||message from v$session_longops;
ELAPSED_SECONDS||':'||MESSAGE
------------------------------------------------------------------------
---------------
351:Online data file move: data file 8: 22548578304 out of 22548578304
bytes done
SQL> select bytes / 1024 / 1024 /1024 as GB from dba_data_files where
FILE_ID = 8;
        GB
----------
        21
```

在此示例中、移动的文件为数据文件8、该文件大小为21 GB、需要大约6分钟才能完成迁移。所需时间显然取决于存储系统的功能、存储网络以及迁移时发生的整体数据库活动。

通过日志传送实现**Oracle**数据库迁移

使用日志传送进行迁移的目标是、在新位置创建原始数据文件的副本、然后建立将更改传送到新环境的方法。

建立日志后、可以自动进行日志传输和重放、以使副本数据库与源数据库大致保持同步。例如、可以计划cron作业：(a)将最新日志复制到新位置、(b)每15分钟重放一次。这样做可以最大程度地减少转换时的中断、因为回写的归档日志不能超过15分钟。

下面显示的操作步骤本质上也是一个数据库克隆操作。显示的逻辑类似于NetApp SnapManager for Oracle (SMO)和NetApp SnapCenter Oracle插件中的引擎。某些客户已使用脚本或WFA工作流中显示的操作步骤执行自定义克隆操作。虽然此操作步骤比使用SMO或SnapCenter更需要手动操作、但仍可随时编写脚本、ONTAP中的数据管理API进一步简化了此过程。

日志传送-文件系统到文件系统

此示例演示了将名为waffle的数据库从普通文件系统迁移到位于不同服务器上的另一个普通文件系统的过程。同时、还展示了如何使用SnapMirror快速复制数据文件、但这并不是整个操作步骤不可或缺的一部分。

创建数据库备份

第一步是创建数据库备份。具体来说、此操作步骤需要一组数据文件、可用于归档日志重放。

environment

在此示例中、源数据库位于ONTAP系统上。创建数据库备份的最简单方法是使用快照。数据库将处于热备份模式几秒钟、而处于 `snapshot create` 在托管数据文件的卷上执行此操作。

```
SQL> alter database begin backup;
Database altered.
```

```
Cluster01::*> snapshot create -vserver vserver1 -volume jfsc1_oradata
hotbackup
Cluster01::*>
```

```
SQL> alter database end backup;
Database altered.
```

结果是在磁盘上生成一个名为的快照 `hotbackup` 该映像包含处于热备份模式时的数据文件映像。如果将此快照中的数据与相应的归档日志结合使用以使数据文件保持一致、则可以将此快照中的数据用作还原或克隆的基础。在这种情况下、它会复制到新服务器。

还原到新环境

现在、必须在新环境中还原备份。这可以通过多种方式实现、包括Oracle RMAN、从备份应用程序(如NetBackup)还原、或者对处于热备份模式的数据文件执行简单的复制操作。

在此示例中、使用SnapMirror将快照热备份复制到新位置。

1. 创建新卷以接收快照数据。从初始化镜像 `jfsc1_oradata` to `vol_oradata`。

   ```
   Cluster01::*> volume create -vserver vserver1 -volume vol_oradata
   -aggregate data_01 -size 20g -state online -type DP -snapshot-policy
   none -policy jfsc3
   [Job 833] Job succeeded: Successful
   ```

   ```
   Cluster01::*> snapmirror initialize -source-path vserver1:jfsc1_oradata
   -destination-path vserver1:vol_oradata
   Operation is queued: snapmirror initialize of destination
   "vserver1:vol_oradata".
   Cluster01::*> volume mount -vserver vserver1 -volume vol_oradata
   -junction-path /vol_oradata
   Cluster01::*>
   ```

2. 在SnapMirror设置状态(指示同步已完成)后、请根据所需的快照专门更新镜像。

   ```
   Cluster01::*> snapmirror show -destination-path vserver1:vol_oradata
   -fields state
   source-path             destination-path        state
   ----------------------- ----------------------- ------------
   vserver1:jfsc1_oradata vserver1:vol_oradata SnapMirrored
   ```

```
Cluster01::*> snapmirror update -destination-path vserver1:vol_oradata
-source-snapshot hotbackup
Operation is queued: snapmirror update of destination
"vserver1:vol_oradata".
```

3. 可以通过查看来验证同步是否成功 `newest-snapshot` 字段。

```
Cluster01::*> snapmirror show -destination-path vserver1:vol_oradata
-fields newest-snapshot
source-path             destination-path        newest-snapshot
----------------------- ----------------------- ---------------
vserver1:jfsc1_oradata vserver1:vol_oradata hotbackup
```

4. 然后、可以断开镜像。

```
Cluster01::> snapmirror break -destination-path vserver1:vol_oradata
Operation succeeded: snapmirror break for destination
"vserver1:vol_oradata".
Cluster01::>
```

5. 挂载新文件系统。对于基于块的文件系统、具体过程因使用的LVM而异。必须配置FC分区或iSCSI连接。与LUN建立连接后、可以使用Linux等命令 `pvscan` 可能需要查找哪些卷组或LUN需要正确配置才能被ASM发现。

   在此示例中、使用的是简单的NFS文件系统。可以直接挂载此文件系统。

```
fas8060-nfs1:/vol_oradata        19922944   1639360   18283584   9%
/oradata
fas8060-nfs1:/vol_logs            9961472       128    9961344   1%
/logs
```

创建控制文件创建模板

接下来必须创建控制文件模板。。 `backup controlfile to trace` 命令用于创建文本命令以重新创建控制文件。在某些情况下、此功能对于从备份还原数据库非常有用、并且通常与执行数据库克隆等任务的脚本结合使用。

1. 以下命令的输出用于为迁移的数据库重新创建控制文件。

```
SQL> alter database backup controlfile to trace as '/tmp/waffle.ctrl';
Database altered.
```

2. 创建控制文件后，将文件复制到新服务器。

```
[oracle@jfsc3 tmp]$ scp oracle@jfsc1:/tmp/waffle.ctrl /tmp/
oracle@jfsc1's password:
waffle.ctrl                                           100% 5199
5.1KB/s   00:00
```

备份参数文件

在新环境中、还需要一个参数文件。最简单的方法是从当前的spfile或pfile创建一个pfile。在此示例中、源数据库使用的是spfile。

```
SQL> create pfile='/tmp/waffle.tmp.pfile' from spfile;
File created.
```

创建**oratab**条目

要使oraenv等实用程序正常运行、必须创建oratab条目。要创建oratab条目、请完成以下步骤。

```
WAFFLE:/orabin/product/12.1.0/dbhome_1:N
```

准备目录结构

如果所需目录不存在、则必须创建它们、否则数据库启动操作步骤将失败。要准备目录结构、请满足以下最低要求。

```
[oracle@jfsc3 ~]$ . oraenv
ORACLE_SID = [oracle] ? WAFFLE
The Oracle base has been set to /orabin
[oracle@jfsc3 ~]$ cd $ORACLE_BASE
[oracle@jfsc3 orabin]$ cd admin
[oracle@jfsc3 admin]$ mkdir WAFFLE
[oracle@jfsc3 admin]$ cd WAFFLE
[oracle@jfsc3 WAFFLE]$ mkdir adump dpdump pfile scripts xdb_wallet
```

参数文件更新

1. 要将参数文件复制到新服务器、请运行以下命令。默认位置为 `$ORACLE_HOME/dbs` 目录。在这种情况下、pfile可以放置在任何位置。它仅用作迁移过程中的中间步骤。

```
[oracle@jfsc3 admin]$ scp oracle@jfsc1:/tmp/waffle.tmp.pfile
$ORACLE_HOME/dbs/waffle.tmp.pfile
oracle@jfsc1's password:
waffle.pfile                                          100%  916
0.9KB/s   00:00
```

1. 根据需要编辑文件。例如、如果归档日志位置已更改、则必须更改pfile以反映新位置。在此示例中、仅重新定位控制文件、部分目的是在日志和数据文件系统之间分布控制文件。

```
[root@jfsc1 tmp]# cat waffle.pfile
WAFFLE.__data_transfer_cache_size=0
WAFFLE.__db_cache_size=507510784
WAFFLE.__java_pool_size=4194304
WAFFLE.__large_pool_size=20971520
WAFFLE.__oracle_base='/orabin'#ORACLE_BASE set from environment
WAFFLE.__pga_aggregate_target=268435456
WAFFLE.__sga_target=805306368
WAFFLE.__shared_io_pool_size=29360128
WAFFLE.__shared_pool_size=234881024
WAFFLE.__streams_pool_size=0
*.audit_file_dest='/orabin/admin/WAFFLE/adump'
*.audit_trail='db'
*.compatible='12.1.0.2.0'
*.control_files='/oradata//WAFFLE/control01.ctl','/oradata//WAFFLE/contr
ol02.ctl'
*.control_files='/oradata/WAFFLE/control01.ctl','/logs/WAFFLE/control02.
ctl'
*.db_block_size=8192
*.db_domain=''
*.db_name='WAFFLE'
*.diagnostic_dest='/orabin'
*.dispatchers='(PROTOCOL=TCP) (SERVICE=WAFFLEXDB)'
*.log_archive_dest_1='LOCATION=/logs/WAFFLE/arch'
*.log_archive_format='%t_%s_%r.dbf'
*.open_cursors=300
*.pga_aggregate_target=256m
*.processes=300
*.remote_login_passwordfile='EXCLUSIVE'
*.sga_target=768m
*.undo_tablespace='UNDOTBS1'
```

2. 编辑完成后、根据此pfile创建一个spfile。

```
SQL> create spfile from pfile='waffle.tmp.pfile';
File created.
```

重新创建控制文件

在上一步中、是的输出 `backup controlfile to trace` 已复制到新服务器。所需输出的具体部分是 `controlfile recreation` 命令：此信息可在标记的部分下的文件中找到 `Set #1. NORESETLOGS`。它从行开始 `create controlfile reuse database` 并应包含该词 `noresetlogs`。以分号(;)字符结尾。

1. 在此示例操作步骤中、该文件如下所示。

```
CREATE CONTROLFILE REUSE DATABASE "WAFFLE" NORESETLOGS  ARCHIVELOG
    MAXLOGFILES 16
    MAXLOGMEMBERS 3
    MAXDATAFILES 100
    MAXINSTANCES 8
    MAXLOGHISTORY 292
LOGFILE
  GROUP 1 '/logs/WAFFLE/redo/redo01.log'  SIZE 50M BLOCKSIZE 512,
  GROUP 2 '/logs/WAFFLE/redo/redo02.log'  SIZE 50M BLOCKSIZE 512,
  GROUP 3 '/logs/WAFFLE/redo/redo03.log'  SIZE 50M BLOCKSIZE 512
-- STANDBY LOGFILE
DATAFILE
  '/oradata/WAFFLE/system01.dbf',
  '/oradata/WAFFLE/sysaux01.dbf',
  '/oradata/WAFFLE/undotbs01.dbf',
  '/oradata/WAFFLE/users01.dbf'
CHARACTER SET WE8MSWIN1252
;
```

2. 根据需要编辑此脚本、以反映各种文件的新位置。例如、某些已知支持高I/O的数据文件可能会重定向到高性能存储层上的文件系统。在其他情况下、更改可能纯粹出于管理员原因、例如、将给定PDB的数据文件隔离到专用卷中。
3. 在此示例中、将显示 `DATAFILE` 虽然保持不变、但重做日志会移动到中的新位置 `/redo` 而不是与归档登录共享空间 `/logs`。

```
CREATE CONTROLFILE REUSE DATABASE "WAFFLE" NORESETLOGS  ARCHIVELOG
    MAXLOGFILES 16
    MAXLOGMEMBERS 3
    MAXDATAFILES 100
    MAXINSTANCES 8
    MAXLOGHISTORY 292
LOGFILE
  GROUP 1 '/redo/redo01.log'  SIZE 50M BLOCKSIZE 512,
  GROUP 2 '/redo/redo02.log'  SIZE 50M BLOCKSIZE 512,
  GROUP 3 '/redo/redo03.log'  SIZE 50M BLOCKSIZE 512
-- STANDBY LOGFILE
DATAFILE
  '/oradata/WAFFLE/system01.dbf',
  '/oradata/WAFFLE/sysaux01.dbf',
  '/oradata/WAFFLE/undotbs01.dbf',
  '/oradata/WAFFLE/users01.dbf'
CHARACTER SET WE8MSWIN1252
;
```

```
SQL> startup nomount;
ORACLE instance started.
Total System Global Area  805306368 bytes
Fixed Size                  2929552 bytes
Variable Size             331353200 bytes
Database Buffers          465567744 bytes
Redo Buffers                5455872 bytes
SQL> CREATE CONTROLFILE REUSE DATABASE "WAFFLE" NORESETLOGS  ARCHIVELOG
  2      MAXLOGFILES 16
  3      MAXLOGMEMBERS 3
  4      MAXDATAFILES 100
  5      MAXINSTANCES 8
  6      MAXLOGHISTORY 292
  7  LOGFILE
  8    GROUP 1 '/redo/redo01.log'  SIZE 50M BLOCKSIZE 512,
  9    GROUP 2 '/redo/redo02.log'  SIZE 50M BLOCKSIZE 512,
 10    GROUP 3 '/redo/redo03.log'  SIZE 50M BLOCKSIZE 512
 11  -- STANDBY LOGFILE
 12  DATAFILE
 13    '/oradata/WAFFLE/system01.dbf',
 14    '/oradata/WAFFLE/sysaux01.dbf',
 15    '/oradata/WAFFLE/undotbs01.dbf',
 16    '/oradata/WAFFLE/users01.dbf'
 17  CHARACTER SET WE8MSWIN1252
 18  ;
Control file created.
SQL>
```

如果任何文件放错位置或参数配置错误、则会生成错误、指示必须修复的问题。数据库已挂载、但尚未打开、无法打开、因为正在使用的数据文件仍标记为处于热备份模式。必须先应用归档日志、以使数据库保持一致。

初始日志复制

要使数据文件保持一致、至少需要执行一个日志回复操作。有许多选项可用于重放日志。在某些情况下、可以通过NFS共享原始服务器上的原始归档日志位置、并且可以直接进行日志回复。在其他情况下、必须复制归档日志。

例如、一个简单的 `scp` 此操作可以将所有当前日志从源服务器复制到迁移服务器：

```
[oracle@jfsc3 arch]$ scp jfsc1:/logs/WAFFLE/arch/* ./
oracle@jfsc1's password:
1_22_912662036.dbf                                          100%   47MB
47.0MB/s   00:01
1_23_912662036.dbf                                          100%   40MB
40.4MB/s   00:00
1_24_912662036.dbf                                          100%   45MB
45.4MB/s   00:00
1_25_912662036.dbf                                          100%   41MB
40.9MB/s   00:01
1_26_912662036.dbf                                          100%   39MB
39.4MB/s   00:00
1_27_912662036.dbf                                          100%   39MB
38.7MB/s   00:00
1_28_912662036.dbf                                          100%   40MB
40.1MB/s   00:01
1_29_912662036.dbf                                          100%   17MB
16.9MB/s   00:00
1_30_912662036.dbf                                          100%  636KB
636.0KB/s   00:00
```

初始日志重放

文件位于归档日志位置后、可以发出命令来重新显示它们 `recover database until cancel` 然后是响应 `AUTO` 自动重放所有可用日志。

```
SQL> recover database until cancel;
ORA-00279: change 382713 generated at 05/24/2016 09:00:54 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_23_912662036.dbf
ORA-00280: change 382713 for thread 1 is in sequence #23
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
AUTO
ORA-00279: change 405712 generated at 05/24/2016 15:01:05 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_24_912662036.dbf
ORA-00280: change 405712 for thread 1 is in sequence #24
ORA-00278: log file '/logs/WAFFLE/arch/1_23_912662036.dbf' no longer
needed for
this recovery
...
ORA-00279: change 713874 generated at 05/26/2016 04:26:43 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_31_912662036.dbf
ORA-00280: change 713874 for thread 1 is in sequence #31
ORA-00278: log file '/logs/WAFFLE/arch/1_30_912662036.dbf' no longer
needed for
this recovery
ORA-00308: cannot open archived log '/logs/WAFFLE/arch/1_31_912662036.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
```

最终归档日志回复报告错误、但这是正常的。日志指示 `sqlplus` 正在查找特定日志文件、但未找到它。原因很可能是日志文件尚不存在。

如果可以在复制归档日志之前关闭源数据库、则只能执行此步骤一次。归档日志会进行复制和重做、然后、该过程可以直接继续执行转换过程、以复制关键重做日志。

增量日志复制和重放

在大多数情况下、不会立即执行迁移。迁移过程可能需要几天甚至几周才能完成、这意味着必须将日志持续运送到副本数据库并进行重新显示。因此、在转换完成后、必须传输和回显示最少的数据。

这样做可以通过多种方式编写脚本、但更常见的方法之一是使用rsync、这是一个常见的文件复制实用程序。使用此实用程序的最安全方法是将其配置为守护进程。例如、 `rsyncd.conf` 下面的文件显示了如何创建名为的资源 `waffle.arch` 可通过Oracle用户凭据访问并映射到 `/logs/WAFFLE/arch`。最重要的是、资源设置为只读、这样可以读取生产数据、但不会对其进行更改。

```
[root@jfsc1 arch]# cat /etc/rsyncd.conf
[waffle.arch]
   uid=oracle
   gid=dba
   path=/logs/WAFFLE/arch
   read only = true
[root@jfsc1 arch]# rsync --daemon
```

以下命令将新服务器的归档日志目标与rsync资源同步 `waffle.arch` 在原始服务器上。。 `t` 中的参数 `rsync - potg` 根据时间戳比较文件列表、并且仅复制新文件。此过程会对新服务器进行增量更新。也可以在cron中计划定期运行此命令。

```
[oracle@jfsc3 arch]$ rsync -potg --stats --progress jfsc1::waffle.arch/*
/logs/WAFFLE/arch/
1_31_912662036.dbf
      650240 100%  124.02MB/s    0:00:00 (xfer#1, to-check=8/18)
1_32_912662036.dbf
     4873728 100%  110.67MB/s    0:00:00 (xfer#2, to-check=7/18)
1_33_912662036.dbf
     4088832 100%   50.64MB/s    0:00:00 (xfer#3, to-check=6/18)
1_34_912662036.dbf
     8196096 100%   54.66MB/s    0:00:00 (xfer#4, to-check=5/18)
1_35_912662036.dbf
    19376128 100%   57.75MB/s    0:00:00 (xfer#5, to-check=4/18)
1_36_912662036.dbf
       71680 100%  201.15kB/s    0:00:00 (xfer#6, to-check=3/18)
1_37_912662036.dbf
     1144320 100%    3.06MB/s    0:00:00 (xfer#7, to-check=2/18)
1_38_912662036.dbf
    35757568 100%   63.74MB/s    0:00:00 (xfer#8, to-check=1/18)
1_39_912662036.dbf
      984576 100%    1.63MB/s    0:00:00 (xfer#9, to-check=0/18)
Number of files: 18
Number of files transferred: 9
Total file size: 399653376 bytes
Total transferred file size: 75143168 bytes
Literal data: 75143168 bytes
Matched data: 0 bytes
File list size: 474
File list generation time: 0.001 seconds
File list transfer time: 0.000 seconds
Total bytes sent: 204
Total bytes received: 75153219
sent 204 bytes  received 75153219 bytes  150306846.00 bytes/sec
total size is 399653376  speedup is 5.32
```

收到日志后、必须对其进行重新显示。前面的示例显示了如何使用sqlplus手动运行 `recover database until cancel`，一个可以轻松实现自动化的过程。此处显示的示例使用中所述的脚本 "重放数据库上的日志"。这些脚本接受一个参数、用于指定需要重放操作的数据库。这样就可以在多数据库迁移工作中使用相同的脚本。

```
[oracle@jfsc3 logs]$ ./replay.logs.pl WAFFLE
ORACLE_SID = [WAFFLE] ? The Oracle base remains unchanged with value
/orabin
SQL*Plus: Release 12.1.0.2.0 Production on Thu May 26 10:47:16 2016
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit
Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
SQL> ORA-00279: change 713874 generated at 05/26/2016 04:26:43 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_31_912662036.dbf
ORA-00280: change 713874 for thread 1 is in sequence #31
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
ORA-00279: change 814256 generated at 05/26/2016 04:52:30 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_32_912662036.dbf
ORA-00280: change 814256 for thread 1 is in sequence #32
ORA-00278: log file '/logs/WAFFLE/arch/1_31_912662036.dbf' no longer
needed for
this recovery
ORA-00279: change 814780 generated at 05/26/2016 04:53:04 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_33_912662036.dbf
ORA-00280: change 814780 for thread 1 is in sequence #33
ORA-00278: log file '/logs/WAFFLE/arch/1_32_912662036.dbf' no longer
needed for
this recovery
...
ORA-00279: change 1120099 generated at 05/26/2016 09:59:21 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_40_912662036.dbf
ORA-00280: change 1120099 for thread 1 is in sequence #40
ORA-00278: log file '/logs/WAFFLE/arch/1_39_912662036.dbf' no longer
needed for
this recovery
ORA-00308: cannot open archived log '/logs/WAFFLE/arch/1_40_912662036.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
```

转换

准备好转换到新环境后、必须执行一次最终同步、其中包括归档日志和重做日志。如果原始重做日志位置尚不已知、则可按如下方式进行标识：

```
SQL> select member from v$logfile;
MEMBER
------------------------------------------------------------------------
------
/logs/WAFFLE/redo/redo01.log
/logs/WAFFLE/redo/redo02.log
/logs/WAFFLE/redo/redo03.log
```

1. 关闭源数据库。
2. 使用所需的方法在新服务器上对归档日志执行一次最终同步。
3. 必须将源重做日志复制到新服务器。在此示例中、重做日志已重新定位到的新目录中 `/redo`。

   ```
   [oracle@jfsc3 logs]$ scp jfsc1:/logs/WAFFLE/redo/* /redo/
   oracle@jfsc1's password:
   redo01.log
   100%   50MB  50.0MB/s   00:01
   redo02.log
   100%   50MB  50.0MB/s   00:00
   redo03.log
   100%   50MB  50.0MB/s   00:00
   ```

4. 在此阶段、新数据库环境包含将其恢复到与源完全相同状态所需的所有文件。归档日志必须最后一次重新显示。

```
SQL> recover database until cancel;
ORA-00279: change 1120099 generated at 05/26/2016 09:59:21 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_40_912662036.dbf
ORA-00280: change 1120099 for thread 1 is in sequence #40
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
AUTO
ORA-00308: cannot open archived log
'/logs/WAFFLE/arch/1_40_912662036.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
ORA-00308: cannot open archived log
'/logs/WAFFLE/arch/1_40_912662036.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
```

5. 完成后、必须重做日志。如果消息 `Media recovery complete` 将返回、此过程将成功、数据库将同步并可打开。

```
SQL> recover database;
Media recovery complete.
SQL> alter database open;
Database altered.
```

日志传送- ASM到文件系统

此示例演示了如何使用Oracle RMAN迁移数据库。它与前面的文件系统到文件系统日志传送示例非常相似、但主机无法识别ASM上的文件。迁移ASM设备上的数据的唯一方法是重新定位ASM LUN或使用Oracle RMAN执行复制操作。

虽然从Oracle ASM复制文件时需要使用RMAN、但RMAN的使用并不限于ASM。RMAN可用于从任何类型的存储迁移到任何其他类型。

此示例显示了将名为pancake的数据库从ASM存储重新定位到位于路径不同服务器上的常规文件系统 `/oradata` 和 `/logs`。

创建数据库备份

第一步是为要迁移到备用服务器的数据库创建备份。由于源使用Oracle ASM、因此必须使用RMAN。可以按如下所示执行简单的RMAN备份。此方法会创建一个带标记的备份、稍后可通过RMAN在操作步骤中轻松识别该备份。

第一个命令用于定义备份的目标类型以及要使用的位置。第二个选项仅启动数据文件的备份。

```
RMAN> configure channel device type disk format '/rman/pancake/%U';
using target database control file instead of recovery catalog
old RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT   '/rman/pancake/%U';
new RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT   '/rman/pancake/%U';
new RMAN configuration parameters are successfully stored
RMAN> backup database tag 'ONTAP_MIGRATION';
Starting backup at 24-MAY-16
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=251 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001 name=+ASM0/PANCAKE/system01.dbf
input datafile file number=00002 name=+ASM0/PANCAKE/sysaux01.dbf
input datafile file number=00003 name=+ASM0/PANCAKE/undotbs101.dbf
input datafile file number=00004 name=+ASM0/PANCAKE/users01.dbf
channel ORA_DISK_1: starting piece 1 at 24-MAY-16
channel ORA_DISK_1: finished piece 1 at 24-MAY-16
piece handle=/rman/pancake/1gr6c161_1_1 tag=ONTAP_MIGRATION comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 24-MAY-16
channel ORA_DISK_1: finished piece 1 at 24-MAY-16
piece handle=/rman/pancake/1hr6c164_1_1 tag=ONTAP_MIGRATION comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 24-MAY-16
```

备份控制文件

稍后需要在的操作步骤中为备份控制文件 `duplicate database` 操作。

```
RMAN> backup current controlfile format '/rman/pancake/ctrl.bkp';
Starting backup at 24-MAY-16
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
channel ORA_DISK_1: starting piece 1 at 24-MAY-16
channel ORA_DISK_1: finished piece 1 at 24-MAY-16
piece handle=/rman/pancake/ctrl.bkp tag=TAG20160524T032651 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 24-MAY-16
```

备份参数文件

在新环境中、还需要一个参数文件。最简单的方法是从当前的spfile或pfile创建一个pfile。在此示例中、源数据库使用spfile。

```
RMAN> create pfile='/rman/pancake/pfile' from spfile;
Statement processed
```

ASM文件重命名脚本

移动数据库时，控制文件中当前定义的几个文件位置会发生变化。以下脚本将创建一个RMAN脚本、以便于执行此过程。此示例显示了一个数据文件数量非常少的数据库、但数据库通常包含数百甚至数千个数据文件。

此脚本可在中找到 "ASM到文件系统名称转换" 它做了两件事。

首先、它会创建一个参数来重新定义重做日志位置、该位置称为 `log_file_name_convert`。它本质上是一个交替字段的列表。第一个字段是当前重做日志的位置、第二个字段是新服务器上的位置。然后、重复执行此模式。

第二个功能是为数据文件重命名提供模板。该脚本循环显示数据文件、提取名称和文件编号信息、并将其格式化为RMAN脚本。然后、它会对临时文件执行相同的操作。结果是生成一个简单的RMAN脚本、可以根据需要进行编辑、以确保文件还原到所需位置。

```
SQL> @/rman/mk.rename.scripts.sql
Parameters for log file conversion:
*.log_file_name_convert = '+ASM0/PANCAKE/redo01.log',
'/NEW_PATH/redo01.log','+ASM0/PANCAKE/redo02.log',
'/NEW_PATH/redo02.log','+ASM0/PANCAKE/redo03.log', '/NEW_PATH/redo03.log'
rman duplication script:
run
{
set newname for datafile 1 to '+ASM0/PANCAKE/system01.dbf';
set newname for datafile 2 to '+ASM0/PANCAKE/sysaux01.dbf';
set newname for datafile 3 to '+ASM0/PANCAKE/undotbs101.dbf';
set newname for datafile 4 to '+ASM0/PANCAKE/users01.dbf';
set newname for tempfile 1 to '+ASM0/PANCAKE/temp01.dbf';
duplicate target database for standby backup location INSERT_PATH_HERE;
}
PL/SQL procedure successfully completed.
```

捕获此屏幕的输出。。 `log_file_name_convert` 参数将按如下所述放置在pfile中。必须相应地编辑RMAN数据文件重命名和重复脚本、才能将数据文件放置在所需位置。在此示例中、它们全部置于中 `/oradata/pancake`。

```
run
{
set newname for datafile 1 to '/oradata/pancake/pancake.dbf';
set newname for datafile 2 to '/oradata/pancake/sysaux.dbf';
set newname for datafile 3 to '/oradata/pancake/undotbs1.dbf';
set newname for datafile 4 to '/oradata/pancake/users.dbf';
set newname for tempfile 1 to '/oradata/pancake/temp.dbf';
duplicate target database for standby backup location '/rman/pancake';
}
```

准备目录结构

这些脚本几乎已准备就绪、可以执行、但首先必须设置好目录结构。如果所需目录不存在、则必须创建它们、否则数据库启动操作步骤将失败。以下示例反映了最低要求。

```
[oracle@jfsc2 ~]$ mkdir /oradata/pancake
[oracle@jfsc2 ~]$ mkdir /logs/pancake
[oracle@jfsc2 ~]$ cd /orabin/admin
[oracle@jfsc2 admin]$ mkdir PANCAKE
[oracle@jfsc2 admin]$ cd PANCAKE
[oracle@jfsc2 PANCAKE]$ mkdir adump dpdump pfile scripts xdb_wallet
```

创建**oratab**条目

要使oraenv等实用程序正常运行、需要使用以下命令。

```
PANCAKE:/orabin/product/12.1.0/dbhome_1:N
```

参数更新

必须更新保存的pfile、以反映新服务器上的任何路径更改。数据文件路径更改由RMAN复制脚本进行更改、几乎所有数据库都需要对进行更改 `control_files` 和 `log_archive_dest` parameters此外、还可能需要更改审核文件位置以及参数、例如 `db_create_file_dest` 在ASM之外可能不相关。经验丰富的DBA应在继续操作之前仔细查看建议的变更。

在此示例中、主要更改包括控制文件位置、日志归档目标以及的添加 `log_file_name_convert` 参数。

```
PANCAKE.__data_transfer_cache_size=0
PANCAKE.__db_cache_size=545259520
PANCAKE.__java_pool_size=4194304
PANCAKE.__large_pool_size=25165824
PANCAKE.__oracle_base='/orabin'#ORACLE_BASE set from environment
PANCAKE.__pga_aggregate_target=268435456
PANCAKE.__sga_target=805306368
PANCAKE.__shared_io_pool_size=29360128
PANCAKE.__shared_pool_size=192937984
PANCAKE.__streams_pool_size=0
*.audit_file_dest='/orabin/admin/PANCAKE/adump'
*.audit_trail='db'
*.compatible='12.1.0.2.0'
*.control_files='+ASM0/PANCAKE/control01.ctl','+ASM0/PANCAKE/control02.ctl
'
*.control_files='/oradata/pancake/control01.ctl','/logs/pancake/control02.
ctl'
*.db_block_size=8192
*.db_domain=''
*.db_name='PANCAKE'
*.diagnostic_dest='/orabin'
*.dispatchers='(PROTOCOL=TCP) (SERVICE=PANCAKEXDB)'
*.log_archive_dest_1='LOCATION=+ASM1'
*.log_archive_dest_1='LOCATION=/logs/pancake'
*.log_archive_format='%t_%s_%r.dbf'
'/logs/path/redo02.log'
*.log_file_name_convert = '+ASM0/PANCAKE/redo01.log',
'/logs/pancake/redo01.log', '+ASM0/PANCAKE/redo02.log',
'/logs/pancake/redo02.log', '+ASM0/PANCAKE/redo03.log',
'/logs/pancake/redo03.log'
*.open_cursors=300
*.pga_aggregate_target=256m
*.processes=300
*.remote_login_passwordfile='EXCLUSIVE'
*.sga_target=768m
*.undo_tablespace='UNDOTBS1'
```

确认新参数后、必须将这些参数生效。虽然存在多个选项、但大多数客户都会根据文本pfile创建spfile。

```
bash-4.1$ sqlplus / as sysdba
SQL*Plus: Release 12.1.0.2.0 Production on Fri Jan 8 11:17:40 2016
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to an idle instance.
SQL> create spfile from pfile='/rman/pancake/pfile';
File created.
```

启动非挂载

复制数据库前的最后一步是启动数据库进程、但不挂载文件。在此步骤中、spfile可能会出现明显问题。如果 `startup nomount` 命令因参数错误而失败、关闭、更正pfile模板、将其重新加载为spfile并重试非常简单。

```
SQL> startup nomount;
ORACLE instance started.
Total System Global Area  805306368 bytes
Fixed Size                  2929552 bytes
Variable Size             373296240 bytes
Database Buffers          423624704 bytes
Redo Buffers                5455872 bytes
```

复制数据库

与此过程中的其他步骤相比、将先前的RMAN备份还原到新位置所需的时间更长。必须在不更改数据库ID (DBID)或不重置日志的情况下复制数据库。这样可以防止应用日志、而这是完全同步副本所必需的步骤。

使用在上一步中创建的脚本、使用RMAN作为aux连接到数据库、并使用问题描述the DUKATE DATABASE命令。

```
[oracle@jfsc2 pancake]$ rman auxiliary /
Recovery Manager: Release 12.1.0.2.0 - Production on Tue May 24 03:04:56
2016
Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights
reserved.
connected to auxiliary database: PANCAKE (not mounted)
RMAN> run
2> {
3> set newname for datafile 1 to '/oradata/pancake/pancake.dbf';
4> set newname for datafile 2 to '/oradata/pancake/sysaux.dbf';
5> set newname for datafile 3 to '/oradata/pancake/undotbs1.dbf';
6> set newname for datafile 4 to '/oradata/pancake/users.dbf';
7> set newname for tempfile 1 to '/oradata/pancake/temp.dbf';
8> duplicate target database for standby backup location '/rman/pancake';
9> }
executing command: SET NEWNAME
```

```
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
Starting Duplicate Db at 24-MAY-16
contents of Memory Script:
{
   restore clone standby controlfile from  '/rman/pancake/ctrl.bkp';
}
executing Memory Script
Starting restore at 24-MAY-16
allocated channel: ORA_AUX_DISK_1
channel ORA_AUX_DISK_1: SID=243 device type=DISK
channel ORA_AUX_DISK_1: restoring control file
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:01
output file name=/oradata/pancake/control01.ctl
output file name=/logs/pancake/control02.ctl
Finished restore at 24-MAY-16
contents of Memory Script:
{
   sql clone 'alter database mount standby database';
}
executing Memory Script
sql statement: alter database mount standby database
released channel: ORA_AUX_DISK_1
allocated channel: ORA_AUX_DISK_1
channel ORA_AUX_DISK_1: SID=243 device type=DISK
contents of Memory Script:
{
   set newname for tempfile  1 to
 "/oradata/pancake/temp.dbf";
   switch clone tempfile all;
   set newname for datafile  1 to
 "/oradata/pancake/pancake.dbf";
   set newname for datafile  2 to
 "/oradata/pancake/sysaux.dbf";
   set newname for datafile  3 to
 "/oradata/pancake/undotbs1.dbf";
   set newname for datafile  4 to
 "/oradata/pancake/users.dbf";
   restore
   clone database
   ;
}
executing Memory Script
executing command: SET NEWNAME
```

```
renamed tempfile 1 to /oradata/pancake/temp.dbf in control file
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
Starting restore at 24-MAY-16
using channel ORA_AUX_DISK_1
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00001 to
/oradata/pancake/pancake.dbf
channel ORA_AUX_DISK_1: restoring datafile 00002 to
/oradata/pancake/sysaux.dbf
channel ORA_AUX_DISK_1: restoring datafile 00003 to
/oradata/pancake/undotbs1.dbf
channel ORA_AUX_DISK_1: restoring datafile 00004 to
/oradata/pancake/users.dbf
channel ORA_AUX_DISK_1: reading from backup piece
/rman/pancake/1gr6c161_1_1
channel ORA_AUX_DISK_1: piece handle=/rman/pancake/1gr6c161_1_1
tag=ONTAP_MIGRATION
channel ORA_AUX_DISK_1: restored backup piece 1
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:07
Finished restore at 24-MAY-16
contents of Memory Script:
{
   switch clone datafile all;
}
executing Memory Script
datafile 1 switched to datafile copy
input datafile copy RECID=5 STAMP=912655725 file
name=/oradata/pancake/pancake.dbf
datafile 2 switched to datafile copy
input datafile copy RECID=6 STAMP=912655725 file
name=/oradata/pancake/sysaux.dbf
datafile 3 switched to datafile copy
input datafile copy RECID=7 STAMP=912655725 file
name=/oradata/pancake/undotbs1.dbf
datafile 4 switched to datafile copy
input datafile copy RECID=8 STAMP=912655725 file
name=/oradata/pancake/users.dbf
Finished Duplicate Db at 24-MAY-16
```

初始日志复制

现在、您必须将更改从源数据库发送到新位置。这样做可能需要多个步骤。最简单的方法是让源数据库上

的RMAN将归档日志写出到共享网络连接。如果共享位置不可用、另一种方法是使用RMAN写入本地文件系统、然后使用rcp或rsync复制文件。

在此示例中、将显示 `/rman` 目录是一个NFS共享、可供原始数据库和迁移的数据库使用。

其中一个重要的问题描述是 `disk format` 条款。备份的磁盘格式为 `%h_%e_%a.dbf`，表示必须使用数据库的线程编号、序列号和激活ID格式。尽管字母不同、但这与匹配 `log_archive_format='%t_%s_%r.dbf` 参数。此参数还以线程编号、序列号和激活ID的格式指定归档日志。最终结果是、源上的日志文件备份会采用数据库预期的命名约定。这样做会执行等操作 `recover database` 更简单、因为sqlplus可以正确地预测要回显的归档日志的名称。

```
RMAN> configure channel device type disk format
'/rman/pancake/logship/%h_%e_%a.dbf';
old RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'/rman/pancake/arch/%h_%e_%a.dbf';
new RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'/rman/pancake/logship/%h_%e_%a.dbf';
new RMAN configuration parameters are successfully stored
released channel: ORA_DISK_1
RMAN> backup as copy archivelog from time 'sysdate-2';
Starting backup at 24-MAY-16
current log archived
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=373 device type=DISK
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=54 RECID=70 STAMP=912658508
output file name=/rman/pancake/logship/1_54_912576125.dbf RECID=123
STAMP=912659482
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=41 RECID=29 STAMP=912654101
output file name=/rman/pancake/logship/1_41_912576125.dbf RECID=124
STAMP=912659483
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:01
...
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=45 RECID=33 STAMP=912654688
output file name=/rman/pancake/logship/1_45_912576125.dbf RECID=152
STAMP=912659514
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=47 RECID=36 STAMP=912654809
output file name=/rman/pancake/logship/1_47_912576125.dbf RECID=153
STAMP=912659515
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:01
Finished backup at 24-MAY-16
```

初始日志重放

文件位于归档日志位置后、可以发出命令来重新显示它们 `recover database until cancel` 然后是响应 `AUTO` 自动重放所有可用日志。参数文件当前正在将归档日志定向到 `/logs/archive`，但这与使用RMAN保存日志的位置不匹配。在恢复数据库之前、可以按如下所示临时重定向此位置。

```
SQL> alter system set log_archive_dest_1='LOCATION=/rman/pancake/logship'
scope=memory;
System altered.
SQL> recover standby database until cancel;
ORA-00279: change 560224 generated at 05/24/2016 03:25:53 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_49_912576125.dbf
ORA-00280: change 560224 for thread 1 is in sequence #49
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
AUTO
ORA-00279: change 560353 generated at 05/24/2016 03:29:17 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_50_912576125.dbf
ORA-00280: change 560353 for thread 1 is in sequence #50
ORA-00278: log file '/rman/pancake/logship/1_49_912576125.dbf' no longer
needed
for this recovery
...
ORA-00279: change 560591 generated at 05/24/2016 03:33:56 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_54_912576125.dbf
ORA-00280: change 560591 for thread 1 is in sequence #54
ORA-00278: log file '/rman/pancake/logship/1_53_912576125.dbf' no longer
needed
for this recovery
ORA-00308: cannot open archived log
'/rman/pancake/logship/1_54_912576125.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
```

最终归档日志回复报告错误、但这是正常的。此错误指示sqlplus正在查找特定日志文件、但未找到该文件。原因很可能是日志文件尚不存在。

如果可以在复制归档日志之前关闭源数据库、则只能执行此步骤一次。归档日志会进行复制和重做、然后、该过程可以直接继续执行转换过程、以复制关键重做日志。

增量日志复制和重放

在大多数情况下、不会立即执行迁移。迁移过程可能需要几天甚至几周时间才能完成、这意味着必须将日志持续运送到副本数据库并进行重新显示。这样可以确保在转换到达时传输和回调的数据最少。

可以轻松编写此过程的脚本。例如、可以在原始数据库上计划以下命令、以确保用于日志传送的位置持续更新。

```
[oracle@jfsc1 pancake]$ cat copylogs.rman
configure channel device type disk format
'/rman/pancake/logship/%h_%e_%a.dbf';
backup as copy archivelog from time 'sysdate-2';
```

```
[oracle@jfsc1 pancake]$ rman target / cmdfile=copylogs.rman
Recovery Manager: Release 12.1.0.2.0 - Production on Tue May 24 04:36:19
2016
Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights
reserved.
connected to target database: PANCAKE (DBID=3574534589)
RMAN> configure channel device type disk format
'/rman/pancake/logship/%h_%e_%a.dbf';
2> backup as copy archivelog from time 'sysdate-2';
3>
4>
using target database control file instead of recovery catalog
old RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'/rman/pancake/logship/%h_%e_%a.dbf';
new RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'/rman/pancake/logship/%h_%e_%a.dbf';
new RMAN configuration parameters are successfully stored
Starting backup at 24-MAY-16
current log archived
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=369 device type=DISK
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=54 RECID=123 STAMP=912659482
RMAN-03009: failure of backup command on ORA_DISK_1 channel at 05/24/2016
04:36:22
ORA-19635: input and output file names are identical:
/rman/pancake/logship/1_54_912576125.dbf
continuing other job steps, job failed will not be re-run
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=41 RECID=124 STAMP=912659483
RMAN-03009: failure of backup command on ORA_DISK_1 channel at 05/24/2016
04:36:23
ORA-19635: input and output file names are identical:
/rman/pancake/logship/1_41_912576125.dbf
continuing other job steps, job failed will not be re-run
...
channel ORA_DISK_1: starting archived log copy
```

```
input archived log thread=1 sequence=45 RECID=152 STAMP=912659514
RMAN-03009: failure of backup command on ORA_DISK_1 channel at 05/24/2016
04:36:55
ORA-19635: input and output file names are identical:
/rman/pancake/logship/1_45_912576125.dbf
continuing other job steps, job failed will not be re-run
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=47 RECID=153 STAMP=912659515
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03009: failure of backup command on ORA_DISK_1 channel at 05/24/2016
04:36:57
ORA-19635: input and output file names are identical:
/rman/pancake/logship/1_47_912576125.dbf
Recovery Manager complete.
```

收到日志后、必须对其进行重新显示。前面的示例显示了如何使用sqlplus手动运行 `recover database until cancel`，可以轻松实现自动化。此处显示的示例使用中所述的脚本 "在备用数据库上重放日志"。该脚本接受一个参数、用于指定需要重放操作的数据库。此过程允许在多数据库迁移工作中使用相同的脚本。

```
[root@jfsc2 pancake]# ./replaylogs.pl PANCAKE
ORACLE_SID = [oracle] ? The Oracle base has been set to /orabin
SQL*Plus: Release 12.1.0.2.0 Production on Tue May 24 04:47:10 2016
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit
Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
SQL> ORA-00279: change 560591 generated at 05/24/2016 03:33:56 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_54_912576125.dbf
ORA-00280: change 560591 for thread 1 is in sequence #54
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
ORA-00279: change 562219 generated at 05/24/2016 04:15:08 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_55_912576125.dbf
ORA-00280: change 562219 for thread 1 is in sequence #55
ORA-00278: log file '/rman/pancake/logship/1_54_912576125.dbf' no longer
needed for this recovery
ORA-00279: change 562370 generated at 05/24/2016 04:19:18 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_56_912576125.dbf
ORA-00280: change 562370 for thread 1 is in sequence #56
ORA-00278: log file '/rman/pancake/logship/1_55_912576125.dbf' no longer
needed for this recovery
...
ORA-00279: change 563137 generated at 05/24/2016 04:36:20 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_65_912576125.dbf
ORA-00280: change 563137 for thread 1 is in sequence #65
ORA-00278: log file '/rman/pancake/logship/1_64_912576125.dbf' no longer
needed for this recovery
ORA-00308: cannot open archived log
'/rman/pancake/logship/1_65_912576125.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
```

转换

准备好转换到新环境后、必须执行一次最终同步。使用常规文件系统时、可以轻松确保迁移的数据库与原始数据库100%同步、因为原始重做日志会被复制和重做。使用ASM无法实现此目的。只能轻松地重新复制归档日志。为了确保不会丢失任何数据、必须谨慎地最终关闭原始数据库。

1. 首先、必须将数据库静机、以确保不会进行任何更改。此暂停可能包括禁用计划的操作、关闭侦听器和/或关闭应用程序。
2. 执行此步骤后、大多数数据库配置协议都会创建一个虚拟表、用作关闭标记。
3. 强制进行日志归档、以确保在归档日志中记录虚拟表的创建。为此、请运行以下命令：

```
SQL> create table cutovercheck as select * from dba_users;
Table created.
SQL> alter system archive log current;
System altered.
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
```

4. 要复制最后一个归档日志、请运行以下命令。数据库必须可用、但未打开。

```
SQL> startup mount;
ORACLE instance started.
Total System Global Area  805306368 bytes
Fixed Size                  2929552 bytes
Variable Size             331353200 bytes
Database Buffers          465567744 bytes
Redo Buffers                5455872 bytes
Database mounted.
```

5. 要复制归档日志、请运行以下命令：

```
RMAN> configure channel device type disk format
'/rman/pancake/logship/%h_%e_%a.dbf';
2> backup as copy archivelog from time 'sysdate-2';
3>
4>
using target database control file instead of recovery catalog
old RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'/rman/pancake/logship/%h_%e_%a.dbf';
new RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'/rman/pancake/logship/%h_%e_%a.dbf';
new RMAN configuration parameters are successfully stored
Starting backup at 24-MAY-16
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=8 device type=DISK
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=54 RECID=123 STAMP=912659482
RMAN-03009: failure of backup command on ORA_DISK_1 channel at
05/24/2016 04:58:24
ORA-19635: input and output file names are identical:
/rman/pancake/logship/1_54_912576125.dbf
continuing other job steps, job failed will not be re-run
...
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=45 RECID=152 STAMP=912659514
RMAN-03009: failure of backup command on ORA_DISK_1 channel at
05/24/2016 04:58:58
ORA-19635: input and output file names are identical:
/rman/pancake/logship/1_45_912576125.dbf
continuing other job steps, job failed will not be re-run
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=47 RECID=153 STAMP=912659515
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03009: failure of backup command on ORA_DISK_1 channel at
05/24/2016 04:59:00
ORA-19635: input and output file names are identical:
/rman/pancake/logship/1_47_912576125.dbf
```

6. 最后、在新服务器上重放其余归档日志。

```
[root@jfsc2 pancake]# ./replaylogs.pl PANCAKE
ORACLE_SID = [oracle] ? The Oracle base has been set to /orabin
SQL*Plus: Release 12.1.0.2.0 Production on Tue May 24 05:00:53 2016
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit
Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
SQL> ORA-00279: change 563137 generated at 05/24/2016 04:36:20 needed
for thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_65_912576125.dbf
ORA-00280: change 563137 for thread 1 is in sequence #65
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
ORA-00279: change 563629 generated at 05/24/2016 04:55:20 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_66_912576125.dbf
ORA-00280: change 563629 for thread 1 is in sequence #66
ORA-00278: log file '/rman/pancake/logship/1_65_912576125.dbf' no longer
needed
for this recovery
ORA-00308: cannot open archived log
'/rman/pancake/logship/1_66_912576125.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
```

7. 在此阶段、复制所有数据。数据库已准备好从备用数据库转换为活动操作数据库、然后再打开。

```
SQL> alter database activate standby database;
Database altered.
SQL> alter database open;
Database altered.
```

8. 确认是否存在假表、然后将其放下。

```
SQL> desc cutovercheck
 Name                                      Null?    Type
 ----------------------------------------- --------
----------------------------
 USERNAME                                  NOT NULL VARCHAR2(128)
 USER_ID                                   NOT NULL NUMBER
 PASSWORD                                           VARCHAR2(4000)
 ACCOUNT_STATUS                            NOT NULL VARCHAR2(32)
 LOCK_DATE                                          DATE
 EXPIRY_DATE                                        DATE
 DEFAULT_TABLESPACE                        NOT NULL VARCHAR2(30)
 TEMPORARY_TABLESPACE                      NOT NULL VARCHAR2(30)
 CREATED                                   NOT NULL DATE
 PROFILE                                   NOT NULL VARCHAR2(128)
 INITIAL_RSRC_CONSUMER_GROUP                        VARCHAR2(128)
 EXTERNAL_NAME                                      VARCHAR2(4000)
 PASSWORD_VERSIONS                                  VARCHAR2(12)
 EDITIONS_ENABLED                                   VARCHAR2(1)
 AUTHENTICATION_TYPE                                VARCHAR2(8)
 PROXY_ONLY_CONNECT                                 VARCHAR2(1)
 COMMON                                             VARCHAR2(3)
 LAST_LOGIN                                         TIMESTAMP(9) WITH
TIME ZONE
 ORACLE_MAINTAINED                                  VARCHAR2(1)
SQL> drop table cutovercheck;
Table dropped.
```

无中断重做日志迁移

有时、除了重做日志之外、数据库整体组织正确。发生这种情况的原因有很多、其中最常见的原因是与快照有关。SnapManager for Oracle、SnapCenter和NetApp Snap Creator存储管理框架等产品可以近乎即时地恢复数据库、但前提是您还原数据文件卷的状态。如果重做日志与数据文件共享空间、则无法安全地执行还原、因为它会导致重做日志被销毁、这可能意味着数据丢失。因此、必须重新定位重做日志。

此操作步骤非常简单、可以无干扰地执行。

当前重做日志配置

1. 确定重做日志组的数量及其相应的组编号。

```
SQL> select group#||' '||member from v$logfile;
GROUP#||''||MEMBER
--------------------------------------------------------------------------
--------
1 /redo0/NTAP/redo01a.log
1 /redo1/NTAP/redo01b.log
2 /redo0/NTAP/redo02a.log
2 /redo1/NTAP/redo02b.log
3 /redo0/NTAP/redo03a.log
3 /redo1/NTAP/redo03b.log
rows selected.
```

2. 输入重做日志的大小。

```
SQL> select group#||' '||bytes from v$log;
GROUP#||''||BYTES
--------------------------------------------------------------------------
--------
1 524288000
2 524288000
3 524288000
```

创建新日志

1. 对于每个重做日志、创建一个大小和成员数量匹配的新组。

```
SQL> alter database add logfile ('/newredo0/redo01a.log',
'/newredo1/redo01b.log') size 500M;
Database altered.
SQL> alter database add logfile ('/newredo0/redo02a.log',
'/newredo1/redo02b.log') size 500M;
Database altered.
SQL> alter database add logfile ('/newredo0/redo03a.log',
'/newredo1/redo03b.log') size 500M;
Database altered.
SQL>
```

2. 验证新配置。

```
SQL> select group#||' '||member from v$logfile;
GROUP#||''||MEMBER
------------------------------------------------------------------------
--------
1 /redo0/NTAP/redo01a.log
1 /redo1/NTAP/redo01b.log
2 /redo0/NTAP/redo02a.log
2 /redo1/NTAP/redo02b.log
3 /redo0/NTAP/redo03a.log
3 /redo1/NTAP/redo03b.log
4 /newredo0/redo01a.log
4 /newredo1/redo01b.log
5 /newredo0/redo02a.log
5 /newredo1/redo02b.log
6 /newredo0/redo03a.log
6 /newredo1/redo03b.log
12 rows selected.
```

丢弃旧日志

1. 丢弃旧日志(组1、2和3)。

```
SQL> alter database drop logfile group 1;
Database altered.
SQL> alter database drop logfile group 2;
Database altered.
SQL> alter database drop logfile group 3;
Database altered.
```

2. 如果遇到错误、导致您无法删除活动日志、请强制切换到下一个日志以释放锁定并强制执行全局检查点。请参见以下此过程的示例。删除位于旧位置的日志文件组2的尝试被拒绝、因为此日志文件中仍有活动数据。

```
SQL> alter database drop logfile group 2;
alter database drop logfile group 2
*
ERROR at line 1:
ORA-01623: log 2 is current log for instance NTAP (thread 1) - cannot
drop
ORA-00312: online log 2 thread 1: '/redo0/NTAP/redo02a.log'
ORA-00312: online log 2 thread 1: '/redo1/NTAP/redo02b.log'
```

3. 日志归档后加上检查点可用于删除日志文件。

```
SQL> alter system archive log current;
System altered.
SQL> alter system checkpoint;
System altered.
SQL> alter database drop logfile group 2;
Database altered.
```

4. 然后从文件系统中删除日志。执行此过程时应格外小心。

Oracle数据库主机数据复制

与数据库级迁移一样、在主机层进行迁移也是一种独立于存储供应商的方法。

换言之、有时"只复制文件"是最佳选择。

虽然这种低技术方法可能看起来过于简单、但它确实具有显著优势、因为无需使用特殊软件、而且在该过程中、原始数据始终保持安全不变。主要限制是、文件复制数据迁移过程会造成系统中断、因为在复制操作开始之前、必须关闭数据库。没有好的方法可以同步文件中的更改、因此、在开始复制之前、必须完全将文件置于静状态。

如果不希望执行复制操作所需的关闭操作、则基于主机的下一个最佳选项是利用逻辑卷管理器(LVM)。存在许多LVM选项、包括Oracle ASM、所有这些选项都具有相似的功能、但也有一些必须考虑的限制。在大多数情况下、可以在不发生停机和中断的情况下完成迁移。

文件系统到文件系统复制

不应低估简单复制操作的有用性。此操作需要在复制过程中停机、但这是一个高度可靠的过程、无需具备有关操作系统、数据库或存储系统的专业知识。此外、它非常安全、因为它不会影响原始数据。通常、系统管理员会将要挂载的源文件系统更改为只读、然后重新启动服务器以确保任何内容都不会损坏当前数据。可以为复制过程编写脚本、以确保其尽可能快地运行、而不会出现用户错误的风险。由于I/O类型是简单的顺序数据传输、因此具有高带宽效率。

以下示例演示了安全快速迁移的一个选项。

environment

要迁移的环境如下：

- 当前文件系统

```
ontap-nfs1:/host1_oradata       52428800  16196928  36231872  31%
/oradata
ontap-nfs1:/host1_logs          49807360    548032  49259328   2% /logs
```

- 新文件系统

```
ontap-nfs1:/host1_logs_new      49807360       128  49807232   1%
/new/logs
ontap-nfs1:/host1_oradata_new   49807360       128  49807232   1%
/new/oradata
```

概述

数据库可以由数据库管理机构进行迁移、只需关闭数据库并复制文件即可、但如果必须迁移多个数据库、或者最短的停机时间至关重要、则可以轻松编写该过程的脚本。使用脚本还可以降低用户出错的几率。

显示的示例脚本可自动执行以下操作：

- 正在关闭数据库
- 将现有文件系统转换为只读状态
- 将源文件系统中的所有数据复制到目标文件系统、从而保留所有文件权限
- 卸载新旧文件系统
- 使用与先前文件系统相同的路径重新挂载新文件系统

操作步骤

1. 关闭数据库。

```
[root@host1 current]# ./dbshut.pl NTAP
ORACLE_SID = [oracle] ? The Oracle base has been set to /orabin
SQL*Plus: Release 12.1.0.2.0 Production on Thu Dec 3 15:58:48 2015
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit
Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
SQL> Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
NTAP shut down
```

2. 将文件系统转换为只读。可以使用脚本更快地完成此操作、如所示 "将文件系统转换为只读"。

```
[root@host1 current]# ./mk.fs.readonly.pl /oradata
/oradata unmounted
/oradata mounted read-only
[root@host1 current]# ./mk.fs.readonly.pl /logs
/logs unmounted
/logs mounted read-only
```

3. 确认文件系统现在为只读。

```
ontap-nfs1:/host1_oradata on /oradata type nfs
(ro,bg,vers=3,rsize=65536,wsize=65536,addr=172.20.101.10)
ontap-nfs1:/host1_logs on /logs type nfs
(ro,bg,vers=3,rsize=65536,wsize=65536,addr=172.20.101.10)
```

4. 将文件系统内容与同步 `rsync` 命令：

```
[root@host1 current]# rsync -rlpogt --stats --progress
--exclude=.snapshot /oradata/ /new/oradata/
sending incremental file list
./
NTAP/
NTAP/IOPS.dbf
 10737426432 100%  153.50MB/s    0:01:06 (xfer#1, to-check=10/13)
NTAP/iops.dbf.zip
    22823573 100%   12.09MB/s    0:00:01 (xfer#2, to-check=9/13)
...
NTAP/undotbs02.dbf
  1073750016 100%  131.60MB/s    0:00:07 (xfer#10, to-check=1/13)
NTAP/users01.dbf
     5251072 100%    3.95MB/s    0:00:01 (xfer#11, to-check=0/13)
Number of files: 13
Number of files transferred: 11
Total file size: 18570092218 bytes
Total transferred file size: 18570092218 bytes
Literal data: 18570092218 bytes
Matched data: 0 bytes
File list size: 277
File list generation time: 0.001 seconds
File list transfer time: 0.000 seconds
Total bytes sent: 18572359828
Total bytes received: 228
sent 18572359828 bytes  received 228 bytes  162204017.96 bytes/sec
total size is 18570092218  speedup is 1.00
```

```
[root@host1 current]# rsync -rlpogt --stats --progress
--exclude=.snapshot /logs/ /new/logs/
sending incremental file list
./
NTAP/
NTAP/1_22_897068759.dbf
    45523968 100%   95.98MB/s    0:00:00 (xfer#1, to-check=15/18)
NTAP/1_23_897068759.dbf
    40601088 100%   49.45MB/s    0:00:00 (xfer#2, to-check=14/18)
...
NTAP/redo/redo02.log
    52429312 100%   44.68MB/s    0:00:01 (xfer#12, to-check=1/18)
NTAP/redo/redo03.log
    52429312 100%   68.03MB/s    0:00:00 (xfer#13, to-check=0/18)
Number of files: 18
Number of files transferred: 13
Total file size: 527032832 bytes
Total transferred file size: 527032832 bytes
Literal data: 527032832 bytes
Matched data: 0 bytes
File list size: 413
File list generation time: 0.001 seconds
File list transfer time: 0.000 seconds
Total bytes sent: 527098156
Total bytes received: 278
sent 527098156 bytes  received 278 bytes  95836078.91 bytes/sec
total size is 527032832  speedup is 1.00
```

5. 卸载旧文件系统并重新定位复制的数据。可以使用脚本更快地完成此操作、如所示 "替换文件系统"。

```
[root@host1 current]# ./swap.fs.pl /logs,/new/logs
/new/logs unmounted
/logs unmounted
Updated /logs mounted
[root@host1 current]# ./swap.fs.pl /oradata,/new/oradata
/new/oradata unmounted
/oradata unmounted
Updated /oradata mounted
```

6. 确认新文件系统已就位。

```
ontap-nfs1:/host1_logs_new on /logs type nfs
(rw,bg,vers=3,rsize=65536,wsize=65536,addr=172.20.101.10)
ontap-nfs1:/host1_oradata_new on /oradata type nfs
(rw,bg,vers=3,rsize=65536,wsize=65536,addr=172.20.101.10)
```

7. 启动数据库。

```
[root@host1 current]# ./dbstart.pl NTAP
ORACLE_SID = [oracle] ? The Oracle base has been set to /orabin
SQL*Plus: Release 12.1.0.2.0 Production on Thu Dec 3 16:10:07 2015
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to an idle instance.
SQL> ORACLE instance started.
Total System Global Area  805306368 bytes
Fixed Size                  2929552 bytes
Variable Size             390073456 bytes
Database Buffers          406847488 bytes
Redo Buffers                5455872 bytes
Database mounted.
Database opened.
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
NTAP started
```

完全自动化转换

此示例脚本接受数据库SID的参数、后跟共同分隔的文件系统对。对于上面显示的示例、命令的发出方式如下：

```
[root@host1 current]# ./migrate.oracle.fs.pl NTAP /logs,/new/logs
/oradata,/new/oradata
```

执行此示例脚本时、此示例脚本将尝试执行以下序列。如果在任何步骤中遇到错误、则会终止：

1. 关闭数据库。
2. 将当前文件系统转换为只读状态。
3. 使用以逗号分隔的每对文件系统参数、并将第一个文件系统同步到第二个文件系统。
4. 卸载先前的文件系统。
5. 更新 `/etc/fstab` 文件、如下所示：
   a. 在创建备份 `/etc/fstab.bak`。

b. 注释掉先前和新文件系统的先前条目。

c. 为使用旧装载点的新文件系统创建一个新条目。

6. 挂载文件系统。
7. 启动数据库。

以下文本提供了此脚本的执行示例：

```
[root@host1 current]# ./migrate.oracle.fs.pl NTAP /logs,/new/logs
/oradata,/new/oradata
ORACLE_SID = [oracle] ? The Oracle base has been set to /orabin
SQL*Plus: Release 12.1.0.2.0 Production on Thu Dec 3 17:05:50 2015
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit
Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
SQL> Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
NTAP shut down
sending incremental file list
./
NTAP/
NTAP/1_22_897068759.dbf
    45523968 100%  185.40MB/s    0:00:00 (xfer#1, to-check=15/18)
NTAP/1_23_897068759.dbf
    40601088 100%   81.34MB/s    0:00:00 (xfer#2, to-check=14/18)
...
NTAP/redo/redo02.log
    52429312 100%   70.42MB/s    0:00:00 (xfer#12, to-check=1/18)
NTAP/redo/redo03.log
    52429312 100%   47.08MB/s    0:00:01 (xfer#13, to-check=0/18)
Number of files: 18
Number of files transferred: 13
Total file size: 527032832 bytes
Total transferred file size: 527032832 bytes
Literal data: 527032832 bytes
Matched data: 0 bytes
File list size: 413
File list generation time: 0.001 seconds
```

```
File list transfer time: 0.000 seconds
Total bytes sent: 527098156
Total bytes received: 278
sent 527098156 bytes  received 278 bytes  150599552.57 bytes/sec
total size is 527032832  speedup is 1.00
Succesfully replicated filesystem /logs to /new/logs
sending incremental file list
./
NTAP/
NTAP/IOPS.dbf
 10737426432 100%  176.55MB/s    0:00:58 (xfer#1, to-check=10/13)
NTAP/iops.dbf.zip
    22823573 100%    9.48MB/s    0:00:02 (xfer#2, to-check=9/13)
... NTAP/undotbs01.dbf
   309338112 100%   70.76MB/s    0:00:04 (xfer#9, to-check=2/13)
NTAP/undotbs02.dbf
  1073750016 100%  187.65MB/s    0:00:05 (xfer#10, to-check=1/13)
NTAP/users01.dbf
     5251072 100%    5.09MB/s    0:00:00 (xfer#11, to-check=0/13)
Number of files: 13
Number of files transferred: 11
Total file size: 18570092218 bytes
Total transferred file size: 18570092218 bytes
Literal data: 18570092218 bytes
Matched data: 0 bytes
File list size: 277
File list generation time: 0.001 seconds
File list transfer time: 0.000 seconds
Total bytes sent: 18572359828
Total bytes received: 228
sent 18572359828 bytes  received 228 bytes  177725933.55 bytes/sec
total size is 18570092218  speedup is 1.00
Succesfully replicated filesystem /oradata to /new/oradata
swap 0 /logs /new/logs
/new/logs unmounted
/logs unmounted
Mounted updated /logs
Swapped filesystem /logs for /new/logs
swap 1 /oradata /new/oradata
/new/oradata unmounted
/oradata unmounted
Mounted updated /oradata
Swapped filesystem /oradata for /new/oradata
ORACLE_SID = [oracle] ? The Oracle base has been set to /orabin
SQL*Plus: Release 12.1.0.2.0 Production on Thu Dec 3 17:08:59 2015
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
```

```
Connected to an idle instance.
SQL> ORACLE instance started.
Total System Global Area  805306368 bytes
Fixed Size                  2929552 bytes
Variable Size             390073456 bytes
Database Buffers          406847488 bytes
Redo Buffers                5455872 bytes
Database mounted.
Database opened.
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
NTAP started
[root@host1 current]#
```

Oracle ASM spfile和passwd迁移

完成涉及ASM的迁移的一个困难是ASM专用的spfile和密码文件。默认情况下、这些关键元数据文件是在定义的第一个ASM磁盘组上创建的。如果必须清空并删除特定ASM磁盘组、则必须重新定位用于管理该ASM实例的spfile和密码文件。

可能需要重新定位这些文件的另一个用例是在部署数据库管理软件(如SnapManager for Oracle或SnapCenter Oracle插件)期间。这些产品的功能之一是、通过还原托管数据文件的ASM LUN的状态来快速还原数据库。执行此操作需要先使ASM磁盘组脱机、然后再执行还原。只要给定数据库的数据文件隔离在专用ASM磁盘组中、就不会出现此问题。

如果该磁盘组还包含ASM spfile/passwd文件、则使该磁盘组脱机的唯一方法是关闭整个ASM实例。此过程会造成系统中断、这意味着需要重新定位spfile/passwd文件。

environment

1. 数据库SID = TOAST
2. 上的当前数据文件 `+DATA`
3. 上的当前日志文件和控制文件 `+LOGS`
4. 新的ASM磁盘组建立为 `+NEWDATA` 和 `+NEWLOGS`

ASM spfile/passwd文件位置

可以无系统地重新定位这些文件。但是、为了安全起见、NetApp建议关闭数据库环境、以便确保文件已重新定位、并且配置已正确更新。如果服务器上存在多个ASM实例、则必须重复执行此操作步骤。

确定ASM实例

根据中记录的数据确定ASM实例 `oratab` 文件ASM实例用+符号表示。

```
-bash-4.1$ cat /etc/oratab | grep '^+'
+ASM:/orabin/grid:N             # line added by Agent
```

此服务器上有一个名为+ASM的ASM实例。

确保所有数据库均已关闭

唯一可见的SMON进程应该是正在使用的ASM实例的SMON。如果存在另一个SMON进程、则表示数据库仍在运行。

```
-bash-4.1$ ps -ef | grep smon
oracle     857     1  0 18:26 ?        00:00:00 asm_smon_+ASM
```

唯一的SMON进程是ASM实例本身。这意味着没有其他数据库在运行、并且可以安全地继续运行、而不会造成数据库操作中断的风险。

找到文件

使用确定ASM spfile和密码文件的当前位置 `spget` 和 `pwget` 命令

```
bash-4.1$ asmcmd
ASMCMD> spget
+DATA/spfile.ora
```

```
ASMCMD> pwget --asm
+DATA/orapwasm
```

这两个文件都位于的底部 `+DATA` 磁盘组。

复制文件

使用将文件复制到新的ASM磁盘组 `spcopy` 和 `pwcopy` 命令如果新磁盘组是最近创建的、并且当前为空、则可能需要先挂载它。

```
ASMCMD> mount NEWDATA
```

```
ASMCMD> spcopy +DATA/spfile.ora +NEWDATA/spfile.ora
copying +DATA/spfile.ora -> +NEWDATA/spfilea.ora
```

```
ASMCMD> pwcopy +DATA/orapwasm +NEWDATA/orapwasm
copying +DATA/orapwasm -> +NEWDATA/orapwasm
```

文件现在已从复制 `+DATA` to `+NEWDATA`。

更新ASM实例

现在、必须更新ASM实例以反映位置更改。。 `spset` 和 `pwset` 命令用于更新启动ASM磁盘组所需的ASM元数据。

```
ASMCMD> spset +NEWDATA/spfile.ora
ASMCMD> pwset --asm +NEWDATA/orapwasm
```

使用更新的文件激活ASM

此时、ASM实例仍会使用这些文件的先前位置。必须重新启动实例、才能强制从文件的新位置重新查看这些文件、并释放对先前文件的锁定。

```
-bash-4.1$ sqlplus / as sysasm
SQL> shutdown immediate;
ASM diskgroups volume disabled
ASM diskgroups dismounted
ASM instance shutdown
```

```
SQL> startup
ASM instance started
Total System Global Area 1140850688 bytes
Fixed Size                  2933400 bytes
Variable Size            1112751464 bytes
ASM Cache                  25165824 bytes
ORA-15032: not all alterations performed
ORA-15017: diskgroup "NEWDATA" cannot be mounted
ORA-15013: diskgroup "NEWDATA" is already mounted
```

删除旧的spfile和密码文件

如果已成功执行操作步骤、则先前的文件将不再锁定、现在可以删除。

```
-bash-4.1$ asmcmd
ASMCMD> rm +DATA/spfile.ora
ASMCMD> rm +DATA/orapwasm
```

Oracle ASM到ASM副本

Oracle ASM本质上是一个轻型组合卷管理器和文件系统。由于文件系统不易显示、因此必须使用RMAN执行复制操作。虽然基于副本的迁移过程既安全又简单、但会造成一些中断。可以最大限度地减少中断、但不能完全消除中断。

如果您希望无中断迁移基于ASM的数据库、最佳选择是利用ASM的功能、在删除旧LUN的同时、将ASM块区重新平衡到新LUN。这样做通常是安全的、不会造成操作中断、但不会提供回退路径。如果遇到功能或性能问题、唯一的选择是将数据迁移回源。

可以通过将数据库复制到新位置而不是移动数据来避免此风险、从而使原始数据保持不变。数据库可以在上线之前在其新位置进行全面测试、如果发现问题、原始数据库可作为回退选项使用。

此操作步骤是涉及RMAN的许多选项之一。它支持一个分两步进行的过程、即创建初始备份、然后通过日志重放进行同步。为了最大限度地减少停机时间、需要使用此过程、因为它可以使数据库在初始基线复制期间保持正常运行并提供数据。

复制数据库

Oracle RMAN会为当前位于ASM磁盘组上的源数据库创建一个级别0 (完整)副本 `+DATA` 到上的新位置 `+NEWDATA`。

```
-bash-4.1$ rman target /
Recovery Manager: Release 12.1.0.2.0 - Production on Sun Dec 6 17:40:03
2015
Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights
reserved.
connected to target database: TOAST (DBID=2084313411)
RMAN> backup as copy incremental level 0 database format '+NEWDATA' tag
'ONTAP_MIGRATION';
Starting backup at 06-DEC-15
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=302 device type=DISK
channel ORA_DISK_1: starting datafile copy
input datafile file number=00001
name=+DATA/TOAST/DATAFILE/system.262.897683141
...
input datafile file number=00004
name=+DATA/TOAST/DATAFILE/users.264.897683151
output file name=+NEWDATA/TOAST/DATAFILE/users.258.897759623
tag=ONTAP_MIGRATION RECID=5 STAMP=897759622
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting incremental level 0 datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 06-DEC-15
channel ORA_DISK_1: finished piece 1 at 06-DEC-15
piece
handle=+NEWDATA/TOAST/BACKUPSET/2015_12_06/nnsnn0_ontap_migration_0.262.89
7759623 tag=ONTAP_MIGRATION comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 06-DEC-15
```

强制执行归档日志切换

您必须强制执行归档日志切换、以确保归档日志包含使副本完全一致所需的所有数据。如果不使用此命令、重做日志中可能仍会显示关键数据。

```
RMAN> sql 'alter system archive log current';
sql statement: alter system archive log current
```

关闭源数据库

此步骤会导致中断、因为数据库已关闭并置于访问受限的只读模式。要关闭源数据库、请运行以下命令：

```
RMAN> shutdown immediate;
using target database control file instead of recovery catalog
database closed
database dismounted
Oracle instance shut down
RMAN> startup mount;
connected to target database (not started)
Oracle instance started
database mounted
Total System Global Area     805306368 bytes
Fixed Size                     2929552 bytes
Variable Size                390073456 bytes
Database Buffers             406847488 bytes
Redo Buffers                   5455872 bytes
```

控制文件备份

如果必须中止迁移并还原到原始存储位置、则必须备份控制文件。备份控制文件的副本并非100%必需、但它确实可以使将数据库文件位置重置回原始位置的过程更加轻松。

```
RMAN> backup as copy current controlfile format '/tmp/TOAST.ctrl';
Starting backup at 06-DEC-15
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=358 device type=DISK
channel ORA_DISK_1: starting datafile copy
copying current control file
output file name=/tmp/TOAST.ctrl tag=TAG20151206T174753 RECID=6
STAMP=897760073
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
Finished backup at 06-DEC-15
```

参数更新

当前spfile包含对控制文件在旧ASM磁盘组中当前位置的引用。必须对其进行编辑、编辑中间的pfile版本即可轻松完成编辑。

```
RMAN> create pfile='/tmp/pfile' from spfile;
Statement processed
```

更新**pfile**

更新引用旧ASM磁盘组的所有参数、以反映新ASM磁盘组名称。然后保存更新后的pfile。确保 `db_create` 参数存在。

在以下示例中、引用了 `+DATA` 已更改为 `+NEWDATA` 以黄色突出显示。两个关键参数是 `db_create` 用于在正确位置创建任何新文件的参数。

```
*.compatible='12.1.0.2.0'
*.control_files='+NEWLOGS/TOAST/CONTROLFILE/current.258.897683139'
*.db_block_size=8192
*. db_create_file_dest='+NEWDATA'
*. db_create_online_log_dest_1='+NEWLOGS'
*.db_domain=''
*.db_name='TOAST'
*.diagnostic_dest='/orabin'
*.dispatchers='(PROTOCOL=TCP) (SERVICE=TOASTXDB)'
*.log_archive_dest_1='LOCATION=+NEWLOGS'
*.log_archive_format='%t_%s_%r.dbf'
```

更新**init.ora**文件

大多数基于ASM的数据库都使用 `init.ora` 文件位于中 `$ORACLE_HOME/dbs` 目录、即指向ASM磁盘组上的spfile。此文件必须重定向到新ASM磁盘组上的某个位置。

```
-bash-4.1$ cd $ORACLE_HOME/dbs
-bash-4.1$ cat initTOAST.ora
SPFILE='+DATA/TOAST/spfileTOAST.ora'
```

按如下所示更改此文件：

```
SPFILE=+NEWLOGS/TOAST/spfileTOAST.ora
```

重新创建参数文件

现在、可以使用已编辑的pfile中的数据填充spfile。

```
RMAN> create spfile from pfile='/tmp/pfile';
Statement processed
```

启动数据库以开始使用新的**spfile**

启动数据库、确保它现在使用新创建的spfile、并正确记录对系统参数所做的任何进一步更改。

```
RMAN> startup nomount;
connected to target database (not started)
Oracle instance started
Total System Global Area     805306368 bytes
Fixed Size                     2929552 bytes
Variable Size                373296240 bytes
Database Buffers             423624704 bytes
Redo Buffers                   5455872 bytes
```

还原控制文件

RMAN还可以将RMAN创建的备份控制文件直接还原到新spfile中指定的位置。

```
RMAN> restore controlfile from
'+DATA/TOAST/CONTROLFILE/current.258.897683139';
Starting restore at 06-DEC-15
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=417 device type=DISK
channel ORA_DISK_1: copied control file copy
output file name=+NEWLOGS/TOAST/CONTROLFILE/current.273.897761061
Finished restore at 06-DEC-15
```

挂载数据库并验证新控制文件的使用情况。

```
RMAN> alter database mount;
using target database control file instead of recovery catalog
Statement processed
```

```
SQL> show parameter control_files;
NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
control_files                        string
+NEWLOGS/TOAST/CONTROLFILE/cur
                                                 rent.273.897761061
```

日志重放

数据库当前使用旧位置的数据文件。在使用副本之前、必须对其进行同步。初始复制过程经过了一段时间、所做的更改主要记录在归档日志中。这些更改复制如下：

1. 执行包含归档日志的RMAN增量备份。

```
RMAN> backup incremental level 1 format '+NEWLOGS' for recover of copy
with tag 'ONTAP_MIGRATION' database;
Starting backup at 06-DEC-15
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=62 device type=DISK
channel ORA_DISK_1: starting incremental level 1 datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001
name=+DATA/TOAST/DATAFILE/system.262.897683141
input datafile file number=00002
name=+DATA/TOAST/DATAFILE/sysaux.260.897683143
input datafile file number=00003
name=+DATA/TOAST/DATAFILE/undotbs1.257.897683145
input datafile file number=00004
name=+DATA/TOAST/DATAFILE/users.264.897683151
channel ORA_DISK_1: starting piece 1 at 06-DEC-15
channel ORA_DISK_1: finished piece 1 at 06-DEC-15
piece
handle=+NEWLOGS/TOAST/BACKUPSET/2015_12_06/nnndn1_ontap_migration_0.268.
897762693 tag=ONTAP_MIGRATION comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting incremental level 1 datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 06-DEC-15
channel ORA_DISK_1: finished piece 1 at 06-DEC-15
piece
handle=+NEWLOGS/TOAST/BACKUPSET/2015_12_06/ncsnn1_ontap_migration_0.267.
897762697 tag=ONTAP_MIGRATION comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 06-DEC-15
```

2. 重放日志。

```
RMAN> recover copy of database with tag 'ONTAP_MIGRATION';
Starting recover at 06-DEC-15
using channel ORA_DISK_1
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: specifying datafile copies to recover
recovering datafile copy file number=00001
name=+NEWDATA/TOAST/DATAFILE/system.259.897759609
recovering datafile copy file number=00002
name=+NEWDATA/TOAST/DATAFILE/sysaux.263.897759615
recovering datafile copy file number=00003
name=+NEWDATA/TOAST/DATAFILE/undotbs1.264.897759619
recovering datafile copy file number=00004
name=+NEWDATA/TOAST/DATAFILE/users.258.897759623
channel ORA_DISK_1: reading from backup piece
+NEWLOGS/TOAST/BACKUPSET/2015_12_06/nnndn1_ontap_migration_0.268.8977626
93
channel ORA_DISK_1: piece
handle=+NEWLOGS/TOAST/BACKUPSET/2015_12_06/nnndn1_ontap_migration_0.268.
897762693 tag=ONTAP_MIGRATION
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
Finished recover at 06-DEC-15
```

激活

恢复的控制文件仍引用原始位置的数据文件、并且还包含复制的数据文件的路径信息。

1. 要更改活动数据文件、请运行 `switch database to copy` 命令：

```
RMAN> switch database to copy;
datafile 1 switched to datafile copy
"+NEWDATA/TOAST/DATAFILE/system.259.897759609"
datafile 2 switched to datafile copy
"+NEWDATA/TOAST/DATAFILE/sysaux.263.897759615"
datafile 3 switched to datafile copy
"+NEWDATA/TOAST/DATAFILE/undotbs1.264.897759619"
datafile 4 switched to datafile copy
"+NEWDATA/TOAST/DATAFILE/users.258.897759623"
```

   活动数据文件现在是复制的数据文件、但最终重做日志中可能仍包含更改。

2. 要重放所有剩余日志、请运行 `recover database` 命令：如果消息 `media recovery complete` 显示、表示此过程已成功。

```
RMAN> recover database;
Starting recover at 06-DEC-15
using channel ORA_DISK_1
starting media recovery
media recovery complete, elapsed time: 00:00:01
Finished recover at 06-DEC-15
```

此过程仅更改了普通数据文件的位置。临时数据文件必须重命名、但不需要复制、因为它们只是临时文件。数据库当前已关闭、因此临时数据文件中没有活动数据。

3. 要重新定位临时数据文件、请首先确定其位置。

```
RMAN> select file#||' '||name from v$tempfile;
FILE#||''||NAME
------------------------------------------------------------------------
--------
1 +DATA/TOAST/TEMPFILE/temp.263.897683145
```

4. 使用RMAN命令为每个数据文件设置新名称来重新定位临时数据文件。使用Oracle Managed Files (OMF)时、无需完整名称；ASM磁盘组就足够了。打开数据库后、OMF会链接到ASM磁盘组上的相应位置。要重新定位文件、请运行以下命令：

```
run {
set newname for tempfile 1 to '+NEWDATA';
switch tempfile all;
}
```

```
RMAN> run {
2> set newname for tempfile 1 to '+NEWDATA';
3> switch tempfile all;
4> }
executing command: SET NEWNAME
renamed tempfile 1 to +NEWDATA in control file
```

重做日志迁移

迁移过程已接近完成、但重做日志仍位于原始ASM磁盘组上。重做日志无法直接重新定位。相反、系统会创建一组新的重做日志并将其添加到配置中、然后是一组旧日志。

1. 确定重做日志组的数量及其相应的组编号。

```
RMAN> select group#||' '||member from v$logfile;
GROUP#||''||MEMBER
---------------------------------------------------------------------------
--------
1 +DATA/TOAST/ONLINELOG/group_1.261.897683139
2 +DATA/TOAST/ONLINELOG/group_2.259.897683139
3 +DATA/TOAST/ONLINELOG/group_3.256.897683139
```

2. 输入重做日志的大小。

```
RMAN> select group#||' '||bytes from v$log;
GROUP#||''||BYTES
---------------------------------------------------------------------------
--------
1 52428800
2 52428800
3 52428800
```

3. 对于每个重做日志、使用匹配的配置创建一个新组。如果不使用OMF、则必须指定完整路径。此示例也使用 `db_create_online_log` parameters如前所示、此参数设置为+NEWLOGS。通过此配置、您可以使用以下命令创建新的联机日志、而无需指定文件位置、甚至无需指定特定ASM磁盘组。

```
RMAN> alter database add logfile size 52428800;
Statement processed
RMAN> alter database add logfile size 52428800;
Statement processed
RMAN> alter database add logfile size 52428800;
Statement processed
```

4. 打开数据库。

```
SQL> alter database open;
Database altered.
```

5. 丢弃旧日志。

```
RMAN> alter database drop logfile group 1;
Statement processed
```

6. 如果遇到错误、导致您无法删除活动日志、请强制切换到下一个日志以释放锁定并强制执行全局检查点。下面显示了一个示例。删除位于旧位置的日志文件组3的尝试被拒绝、因为此日志文件中仍有活动数据。通过

检查点后的日志归档、您可以删除日志文件。

```
RMAN> alter database drop logfile group 3;
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of sql statement command at 12/08/2015 20:23:51
ORA-01623: log 3 is current log for instance TOAST (thread 4) - cannot
drop
ORA-00312: online log 3 thread 1:
'+LOGS/TOAST/ONLINELOG/group_3.259.897563549'
RMAN> alter system switch logfile;
Statement processed
RMAN> alter system checkpoint;
Statement processed
RMAN> alter database drop logfile group 3;
Statement processed
```

7. 查看环境以确保所有基于位置的参数均已更新。

```
SQL> select name from v$datafile;
SQL> select member from v$logfile;
SQL> select name from v$tempfile;
SQL> show parameter spfile;
SQL> select name, value from v$parameter where value is not null;
```

8. 以下脚本演示了如何简化此过程：

```
[root@host1 current]# ./checkdbdata.pl TOAST
TOAST datafiles:
+NEWDATA/TOAST/DATAFILE/system.259.897759609
+NEWDATA/TOAST/DATAFILE/sysaux.263.897759615
+NEWDATA/TOAST/DATAFILE/undotbs1.264.897759619
+NEWDATA/TOAST/DATAFILE/users.258.897759623
TOAST redo logs:
+NEWLOGS/TOAST/ONLINELOG/group_4.266.897763123
+NEWLOGS/TOAST/ONLINELOG/group_5.265.897763125
+NEWLOGS/TOAST/ONLINELOG/group_6.264.897763125
TOAST temp datafiles:
+NEWDATA/TOAST/TEMPFILE/temp.260.897763165
TOAST spfile
spfile                                   string
+NEWDATA/spfiletoast.ora
TOAST key parameters
control_files +NEWLOGS/TOAST/CONTROLFILE/current.273.897761061
log_archive_dest_1 LOCATION=+NEWLOGS
db_create_file_dest +NEWDATA
db_create_online_log_dest_1 +NEWLOGS
```

9. 如果ASM磁盘组已完全清空、则现在可以使用卸载这些磁盘组 `asmcmd`。但是、在许多情况下、属于其他数据库的文件或ASM spfile/passwd文件可能仍存在。

```
-bash-4.1$ . oraenv
ORACLE_SID = [TOAST] ? +ASM
The Oracle base remains unchanged with value /orabin
-bash-4.1$ asmcmd
ASMCMD> umount DATA
ASMCMD>
```

Oracle ASM到文件系统的副本

Oracle ASM到文件系统副本操作步骤与ASM到ASM副本操作步骤非常相似、但具有类似的优势和限制。主要区别在于使用可见文件系统时与使用ASM磁盘组时不同命令和配置参数的语法。

复制数据库

Oracle RMAN用于为当前位于ASM磁盘组上的源数据库创建级别0 (完整)副本 `+DATA` 到上的新位置 `/oradata`。

```
RMAN> backup as copy incremental level 0 database format
'/oradata/TOAST/%U' tag 'ONTAP_MIGRATION';
Starting backup at 13-MAY-16
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=377 device type=DISK
channel ORA_DISK_1: starting datafile copy
input datafile file number=00001 name=+ASM0/TOAST/system01.dbf
output file name=/oradata/TOAST/data_D-TOAST_I-2098173325_TS-SYSTEM_FNO-
1_01r5fhjg tag=ONTAP_MIGRATION RECID=1 STAMP=911722099
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:07
channel ORA_DISK_1: starting datafile copy
input datafile file number=00002 name=+ASM0/TOAST/sysaux01.dbf
output file name=/oradata/TOAST/data_D-TOAST_I-2098173325_TS-SYSAUX_FNO-
2_02r5fhjo tag=ONTAP_MIGRATION RECID=2 STAMP=911722106
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:07
channel ORA_DISK_1: starting datafile copy
input datafile file number=00003 name=+ASM0/TOAST/undotbs101.dbf
output file name=/oradata/TOAST/data_D-TOAST_I-2098173325_TS-UNDOTBS1_FNO-
3_03r5fhjt tag=ONTAP_MIGRATION RECID=3 STAMP=911722113
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:07
channel ORA_DISK_1: starting datafile copy
copying current control file
output file name=/oradata/TOAST/cf_D-TOAST_id-2098173325_04r5fhk5
tag=ONTAP_MIGRATION RECID=4 STAMP=911722118
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting datafile copy
input datafile file number=00004 name=+ASM0/TOAST/users01.dbf
output file name=/oradata/TOAST/data_D-TOAST_I-2098173325_TS-USERS_FNO-
4_05r5fhk6 tag=ONTAP_MIGRATION RECID=5 STAMP=911722118
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting incremental level 0 datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 13-MAY-16
channel ORA_DISK_1: finished piece 1 at 13-MAY-16
piece handle=/oradata/TOAST/06r5fhk7_1_1 tag=ONTAP_MIGRATION comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 13-MAY-16
```

强制执行归档日志切换

要确保归档日志包含使副本完全一致所需的所有数据、需要强制执行归档日志开关。如果不使用此命令、重做日志中可能仍会显示关键数据。要强制执行归档日志切换、请运行以下命令：

```
RMAN> sql 'alter system archive log current';
sql statement: alter system archive log current
```

关闭源数据库

此步骤会导致中断、因为数据库已关闭并置于访问受限的只读模式。要关闭源数据库、请运行以下命令：

```
RMAN> shutdown immediate;
using target database control file instead of recovery catalog
database closed
database dismounted
Oracle instance shut down
RMAN> startup mount;
connected to target database (not started)
Oracle instance started
database mounted
Total System Global Area     805306368 bytes
Fixed Size                   2929552 bytes
Variable Size             331353200 bytes
Database Buffers          465567744 bytes
Redo Buffers                5455872 bytes
```

控制文件备份

备份控制文件、以防您必须中止迁移并还原到原始存储位置。备份控制文件的副本并非100%必需、但它确实可以使将数据库文件位置重置回原始位置的过程更加轻松。

```
RMAN> backup as copy current controlfile format '/tmp/TOAST.ctrl';
Starting backup at 08-DEC-15
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile copy
copying current control file
output file name=/tmp/TOAST.ctrl tag=TAG20151208T194540 RECID=30
STAMP=897939940
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
Finished backup at 08-DEC-15
```

参数更新

```
RMAN> create pfile='/tmp/pfile' from spfile;
Statement processed
```

更新pfile

应更新引用旧ASM磁盘组的任何参数、在某些情况下、如果这些参数不再相关、则应将其删除。更新它们以反映新的文件系统路径并保存更新后的pfile。确保列出了完整的目标路径。要更新这些参数、请运行以下命令：

```
*.audit_file_dest='/orabin/admin/TOAST/adump'
*.audit_trail='db'
*.compatible='12.1.0.2.0'
*.control_files='/logs/TOAST/arch/control01.ctl','/logs/TOAST/redo/control
02.ctl'
*.db_block_size=8192
*.db_domain=''
*.db_name='TOAST'
*.diagnostic_dest='/orabin'
*.dispatchers='(PROTOCOL=TCP) (SERVICE=TOASTXDB)'
*.log_archive_dest_1='LOCATION=/logs/TOAST/arch'
*.log_archive_format='%t_%s_%r.dbf'
*.open_cursors=300
*.pga_aggregate_target=256m
*.processes=300
*.remote_login_passwordfile='EXCLUSIVE'
*.sga_target=768m
*.undo_tablespace='UNDOTBS1'
```

禁用原始init.ora文件

此文件位于中 `$ORACLE_HOME/dbs` 目录中、通常位于一个pfile中、用作指向ASM磁盘组上spfile的指针。要确保原始spfile不再使用、请对其重命名。但是、请勿将其删除、因为如果必须中止迁移、则需要此文件。

```
[oracle@jfsc1 ~]$ cd $ORACLE_HOME/dbs
[oracle@jfsc1 dbs]$ cat initTOAST.ora
SPFILE='+ASM0/TOAST/spfileTOAST.ora'
[oracle@jfsc1 dbs]$ mv initTOAST.ora initTOAST.ora.prev
[oracle@jfsc1 dbs]$
```

重新创建参数文件

这是spfile重新定位的最后一步。不再使用原始spfile、数据库当前已使用中间文件启动(但未挂载)。此文件的内容可以按如下所示写出到新的spfile位置：

```
RMAN> create spfile from pfile='/tmp/pfile';
Statement processed
```

启动数据库以开始使用新的**spfile**

您必须启动数据库以释放中间文件上的锁定、并仅使用新的spfile文件启动数据库。启动数据库还可以证明新的spfile位置正确且其数据有效。

```
RMAN> shutdown immediate;
Oracle instance shut down
RMAN> startup nomount;
connected to target database (not started)
Oracle instance started
Total System Global Area     805306368 bytes
Fixed Size                     2929552 bytes
Variable Size                331353200 bytes
Database Buffers             465567744 bytes
Redo Buffers                   5455872 bytes
```

还原控制文件

在路径上创建了一个备份控制文件 `/tmp/TOAST.ctrl` 在操作步骤中的早期版本。新的spfile将控制文件位置定义为 `/logfs/TOAST/ctrl/ctrlfile1.ctrl` 和 `/logfs/TOAST/redo/ctrlfile2.ctrl`。但是、这些文件尚不存在。

1. 此命令会将控制文件数据还原到spfile中定义的路径。

   ```
   RMAN> restore controlfile from '/tmp/TOAST.ctrl';
   Starting restore at 13-MAY-16
   using channel ORA_DISK_1
   channel ORA_DISK_1: copied control file copy
   output file name=/logs/TOAST/arch/control01.ctl
   output file name=/logs/TOAST/redo/control02.ctl
   Finished restore at 13-MAY-16
   ```

2. 问题描述挂载命令、以便正确发现控制文件并包含有效数据。

   ```
   RMAN> alter database mount;
   Statement processed
   released channel: ORA_DISK_1
   ```

   以验证 `control_files` 参数中、运行以下命令：

```
SQL> show parameter control_files;
NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
control_files                        string
/logs/TOAST/arch/control01.ctl
                                                 ,
/logs/TOAST/redo/control02.c
                                                 tl
```

日志重放

数据库当前正在使用旧位置的数据文件。必须先同步数据文件、然后才能使用副本。初始复制过程经过了一段时间、所做的更改主要记录在归档日志中。这些更改将通过以下两个步骤进行复制。

1. 执行包含归档日志的RMAN增量备份。

```
RMAN>  backup incremental level 1 format '/logs/TOAST/arch/%U' for
recover of copy with tag 'ONTAP_MIGRATION' database;
Starting backup at 13-MAY-16
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=124 device type=DISK
channel ORA_DISK_1: starting incremental level 1 datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001 name=+ASM0/TOAST/system01.dbf
input datafile file number=00002 name=+ASM0/TOAST/sysaux01.dbf
input datafile file number=00003 name=+ASM0/TOAST/undotbs101.dbf
input datafile file number=00004 name=+ASM0/TOAST/users01.dbf
channel ORA_DISK_1: starting piece 1 at 13-MAY-16
channel ORA_DISK_1: finished piece 1 at 13-MAY-16
piece handle=/logs/TOAST/arch/09r5fj8i_1_1 tag=ONTAP_MIGRATION
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 13-MAY-16
RMAN-06497: WARNING: control file is not current, control file
AUTOBACKUP skipped
```

2. 重放日志。

```
RMAN> recover copy of database with tag 'ONTAP_MIGRATION';
Starting recover at 13-MAY-16
using channel ORA_DISK_1
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: specifying datafile copies to recover
recovering datafile copy file number=00001 name=/oradata/TOAST/data_D-
TOAST_I-2098173325_TS-SYSTEM_FNO-1_01r5fhjg
recovering datafile copy file number=00002 name=/oradata/TOAST/data_D-
TOAST_I-2098173325_TS-SYSAUX_FNO-2_02r5fhjo
recovering datafile copy file number=00003 name=/oradata/TOAST/data_D-
TOAST_I-2098173325_TS-UNDOTBS1_FNO-3_03r5fhjt
recovering datafile copy file number=00004 name=/oradata/TOAST/data_D-
TOAST_I-2098173325_TS-USERS_FNO-4_05r5fhk6
channel ORA_DISK_1: reading from backup piece
/logs/TOAST/arch/09r5fj8i_1_1
channel ORA_DISK_1: piece handle=/logs/TOAST/arch/09r5fj8i_1_1
tag=ONTAP_MIGRATION
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
Finished recover at 13-MAY-16
RMAN-06497: WARNING: control file is not current, control file
AUTOBACKUP skipped
```

激活

恢复的控制文件仍引用原始位置的数据文件、并且还包含复制的数据文件的路径信息。

1. 要更改活动数据文件、请运行 `switch database to copy` 命令：

```
RMAN> switch database to copy;
datafile 1 switched to datafile copy "/oradata/TOAST/data_D-TOAST_I-
2098173325_TS-SYSTEM_FNO-1_01r5fhjg"
datafile 2 switched to datafile copy "/oradata/TOAST/data_D-TOAST_I-
2098173325_TS-SYSAUX_FNO-2_02r5fhjo"
datafile 3 switched to datafile copy "/oradata/TOAST/data_D-TOAST_I-
2098173325_TS-UNDOTBS1_FNO-3_03r5fhjt"
datafile 4 switched to datafile copy "/oradata/TOAST/data_D-TOAST_I-
2098173325_TS-USERS_FNO-4_05r5fhk6"
```

2. 尽管数据文件应完全一致、但要重放联机重做日志中记录的其余更改、需要执行最后一步。使用 `recover database` 命令以重放这些更改并使副本与原始副本完全相同。但是、该副本尚未打开。

```
RMAN> recover database;
Starting recover at 13-MAY-16
using channel ORA_DISK_1
starting media recovery
archived log for thread 1 with sequence 28 is already on disk as file
+ASM0/TOAST/redo01.log
archived log file name=+ASM0/TOAST/redo01.log thread=1 sequence=28
media recovery complete, elapsed time: 00:00:00
Finished recover at 13-MAY-16
```

重新定位临时数据文件

1. 确定原始磁盘组上仍在使用的临时数据文件的位置。

```
RMAN> select file#||' '||name from v$tempfile;
FILE#||''||NAME
------------------------------------------------------------------------
--------
1 +ASM0/TOAST/temp01.dbf
```

2. 要重新定位数据文件、请运行以下命令。如果存在许多临时文件、请使用文本编辑器创建RMAN命令、然后将其剪切并粘贴。

```
RMAN> run {
2> set newname for tempfile 1 to '/oradata/TOAST/temp01.dbf';
3> switch tempfile all;
4> }
executing command: SET NEWNAME
renamed tempfile 1 to /oradata/TOAST/temp01.dbf in control file
```

重做日志迁移

迁移过程已接近完成、但重做日志仍位于原始ASM磁盘组上。重做日志无法直接重新定位。相反、系统会创建一组新的重做日志并将其添加到配置中、然后删除旧日志。

1. 确定重做日志组的数量及其相应的组编号。

```
RMAN> select group#||' '||member from v$logfile;
GROUP#||''||MEMBER
---------------------------------------------------------------------------
--------
1 +ASM0/TOAST/redo01.log
2 +ASM0/TOAST/redo02.log
3 +ASM0/TOAST/redo03.log
```

2. 输入重做日志的大小。

```
RMAN> select group#||' '||bytes from v$log;
GROUP#||''||BYTES
---------------------------------------------------------------------------
--------
1 52428800
2 52428800
3 52428800
```

3. 对于每个重做日志、使用与当前重做日志组相同的大小并使用新文件系统位置创建一个新组。

```
RMAN> alter database add logfile '/logs/TOAST/redo/log00.rdo' size
52428800;
Statement processed
RMAN> alter database add logfile '/logs/TOAST/redo/log01.rdo' size
52428800;
Statement processed
RMAN> alter database add logfile '/logs/TOAST/redo/log02.rdo' size
52428800;
Statement processed
```

4. 删除仍位于先前存储上的旧日志文件组。

```
RMAN> alter database drop logfile group 4;
Statement processed
RMAN> alter database drop logfile group 5;
Statement processed
RMAN> alter database drop logfile group 6;
Statement processed
```

5. 如果遇到阻止删除活动日志的错误、请强制切换到下一个日志以释放锁定并强制执行全局检查点。下面显示了一个示例。删除位于旧位置的日志文件组3的尝试被拒绝、因为此日志文件中仍有活动数据。日志归档后加上检查点可以删除日志文件。

```
RMAN> alter database drop logfile group 4;
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of sql statement command at 12/08/2015 20:23:51
ORA-01623: log 4 is current log for instance TOAST (thread 4) - cannot
drop
ORA-00312: online log 4 thread 1:
'+NEWLOGS/TOAST/ONLINELOG/group_4.266.897763123'
RMAN> alter system switch logfile;
Statement processed
RMAN> alter system checkpoint;
Statement processed
RMAN> alter database drop logfile group 4;
Statement processed
```

6. 查看环境以确保所有基于位置的参数均已更新。

```
SQL> select name from v$datafile;
SQL> select member from v$logfile;
SQL> select name from v$tempfile;
SQL> show parameter spfile;
SQL> select name, value from v$parameter where value is not null;
```

7. 以下脚本演示了如何简化此过程。

```
[root@jfsc1 current]# ./checkdbdata.pl TOAST
TOAST datafiles:
/oradata/TOAST/data_D-TOAST_I-2098173325_TS-SYSTEM_FNO-1_01r5fhjg
/oradata/TOAST/data_D-TOAST_I-2098173325_TS-SYSAUX_FNO-2_02r5fhjo
/oradata/TOAST/data_D-TOAST_I-2098173325_TS-UNDOTBS1_FNO-3_03r5fhjt
/oradata/TOAST/data_D-TOAST_I-2098173325_TS-USERS_FNO-4_05r5fhk6
TOAST redo logs:
/logs/TOAST/redo/log00.rdo
/logs/TOAST/redo/log01.rdo
/logs/TOAST/redo/log02.rdo
TOAST temp datafiles:
/oradata/TOAST/temp01.dbf
TOAST spfile
spfile                                   string
/orabin/product/12.1.0/dbhome_
                                                        1/dbs/spfileTOAST.ora
TOAST key parameters
control_files /logs/TOAST/arch/control01.ctl,
/logs/TOAST/redo/control02.ctl
log_archive_dest_1 LOCATION=/logs/TOAST/arch
```

8. 如果ASM磁盘组已完全清空、则现在可以使用卸载这些磁盘组 `asmcmd`。在许多情况下、仍然存在属于其他数据库的文件或ASM spfile/passwd文件。

```
-bash-4.1$ . oraenv
ORACLE_SID = [TOAST] ? +ASM
The Oracle base remains unchanged with value /orabin
-bash-4.1$ asmcmd
ASMCMD> umount DATA
ASMCMD>
```

数据文件清理操作步骤

迁移过程可能会导致数据文件的语法较长或比较隐秘、具体取决于Oracle RMAN的使用方式。在此处显示的示例中、备份是使用的文件格式执行的 `/oradata/TOAST/%U`。 `%U` 指示RMAN应为每个数据文件创建一个默认唯一名称。结果与以下文本中所示结果类似。数据文件的传统名称嵌入在名称中。可以使用中所示的脚本化方法来清除此问题 "ASM迁移清理"。

```
[root@jfsc1 current]# ./fixuniquenames.pl TOAST
#sqlplus Commands
shutdown immediate;
startup mount;
host mv /oradata/TOAST/data_D-TOAST_I-2098173325_TS-SYSTEM_FNO-1_01r5fhjg
/oradata/TOAST/system.dbf
host mv /oradata/TOAST/data_D-TOAST_I-2098173325_TS-SYSAUX_FNO-2_02r5fhjo
/oradata/TOAST/sysaux.dbf
host mv /oradata/TOAST/data_D-TOAST_I-2098173325_TS-UNDOTBS1_FNO-
3_03r5fhjt /oradata/TOAST/undotbs1.dbf
host mv /oradata/TOAST/data_D-TOAST_I-2098173325_TS-USERS_FNO-4_05r5fhk6
/oradata/TOAST/users.dbf
alter database rename file '/oradata/TOAST/data_D-TOAST_I-2098173325_TS-
SYSTEM_FNO-1_01r5fhjg' to '/oradata/TOAST/system.dbf';
alter database rename file '/oradata/TOAST/data_D-TOAST_I-2098173325_TS-
SYSAUX_FNO-2_02r5fhjo' to '/oradata/TOAST/sysaux.dbf';
alter database rename file '/oradata/TOAST/data_D-TOAST_I-2098173325_TS-
UNDOTBS1_FNO-3_03r5fhjt' to '/oradata/TOAST/undotbs1.dbf';
alter database rename file '/oradata/TOAST/data_D-TOAST_I-2098173325_TS-
USERS_FNO-4_05r5fhk6' to '/oradata/TOAST/users.dbf';
alter database open;
```

Oracle ASM重新平衡

如前文所述、可以通过重新平衡过程将Oracle ASM磁盘组透明地迁移到新存储系统。总之、重新平衡过程需要先向现有LUN组添加大小相等的LUN、然后再删除之前的LUN。Oracle ASM会以最佳布局自动将底层数据重新定位到新存储、然后在完成后释放旧LUN。

迁移过程使用高效的顺序I/O、通常不会发生原因发生任何性能中断、但可以根据需要对迁移速率进行控制。

确定要迁移的数据

```
SQL> select name||' '||group_number||' '||total_mb||' '||path||'
'||header_status from v$asm_disk;
NEWDATA_0003 1 10240 /dev/mapper/3600a098038303537762b47594c315864 MEMBER
NEWDATA_0002 1 10240 /dev/mapper/3600a098038303537762b47594c315863 MEMBER
NEWDATA_0000 1 10240 /dev/mapper/3600a098038303537762b47594c315861 MEMBER
NEWDATA_0001 1 10240 /dev/mapper/3600a098038303537762b47594c315862 MEMBER
SQL> select group_number||' '||name from v$asm_diskgroup;
1 NEWDATA
```

创建新**LUN**

创建大小相同的新LUN、并根据需要设置用户和组成员资格。LUN应显示为 `CANDIDATE` 磁盘。

```
SQL> select name||' '||group_number||' '||total_mb||' '||path||'
'||header_status from v$asm_disk;
 0 0 /dev/mapper/3600a098038303537762b47594c31586b CANDIDATE
 0 0 /dev/mapper/3600a098038303537762b47594c315869 CANDIDATE
 0 0 /dev/mapper/3600a098038303537762b47594c315858 CANDIDATE
 0 0 /dev/mapper/3600a098038303537762b47594c31586a CANDIDATE
NEWDATA_0003 1 10240 /dev/mapper/3600a098038303537762b47594c315864 MEMBER
NEWDATA_0002 1 10240 /dev/mapper/3600a098038303537762b47594c315863 MEMBER
NEWDATA_0000 1 10240 /dev/mapper/3600a098038303537762b47594c315861 MEMBER
NEWDATA_0001 1 10240 /dev/mapper/3600a098038303537762b47594c315862 MEMBER
```

添加新**LUN**

虽然可以同时执行添加和删除操作、但通过两个步骤添加新LUN通常更容易。首先、将新LUN添加到磁盘组。此步骤会将一半的块区从当前ASM LUN迁移到新LUN。

重新平衡功率表示数据的传输速率。数量越多、数据传输的并行性就越高。迁移过程采用高效的顺序I/O操作来执行、这些操作不太可能会出现发生原因性能问题。但是、如果需要、可以使用调整正在进行的迁移的重新平衡能力 `alter diskgroup [name] rebalance power [level]` 命令：典型迁移使用的值为5。

```
SQL> alter diskgroup NEWDATA add disk
'/dev/mapper/3600a098038303537762b47594c31586b' rebalance power 5;
Diskgroup altered.
SQL> alter diskgroup NEWDATA add disk
'/dev/mapper/3600a098038303537762b47594c315869' rebalance power 5;
Diskgroup altered.
SQL> alter diskgroup NEWDATA add disk
'/dev/mapper/3600a098038303537762b47594c315858' rebalance power 5;
Diskgroup altered.
SQL> alter diskgroup NEWDATA add disk
'/dev/mapper/3600a098038303537762b47594c31586a' rebalance power 5;
Diskgroup altered.
```

监控操作

可以通过多种方式监控和管理重新平衡操作。在此示例中、我们使用了以下命令。

```
SQL> select group_number,operation,state from v$asm_operation;
GROUP_NUMBER OPERA STAT
------------ ----- ----
           1 REBAL RUN
           1 REBAL WAIT
```

迁移完成后、不会报告重新平衡操作。

```
SQL> select group_number,operation,state from v$asm_operation;
no rows selected
```

丢弃旧**LUN**

迁移现已完成一半。可能需要执行一些基本性能测试、以确保环境运行状况良好。确认后、可以通过删除旧LUN来重新定位其余数据。请注意、这不会导致立即释放LUN。删除操作会通知Oracle ASM先重新定位块区、然后再释放LUN。

```
sqlplus / as sysasm
SQL> alter diskgroup NEWDATA drop disk NEWDATA_0000 rebalance power 5;
Diskgroup altered.
SQL> alter diskgroup NEWDATA drop disk NEWDATA_0001 rebalance power 5;
Diskgroup altered.
SQL> alter diskgroup newdata drop disk NEWDATA_0002 rebalance power 5;
Diskgroup altered.
SQL> alter diskgroup newdata drop disk NEWDATA_0003 rebalance power 5;
Diskgroup altered.
```

监控操作

可以通过多种方式监控和管理重新平衡操作。在此示例中、我们使用了以下命令：

```
SQL> select group_number,operation,state from v$asm_operation;
GROUP_NUMBER OPERA STAT
------------ ----- ----
           1 REBAL RUN
           1 REBAL WAIT
```

迁移完成后、不会报告重新平衡操作。

```
SQL> select group_number,operation,state from v$asm_operation;
no rows selected
```

删除旧**LUN**

在从磁盘组中删除旧LUN之前、应对标头状态执行一次最终检查。从ASM释放LUN后、该LUN不再具有列出的名称、而标头状态将列为 `FORMER`。这表示可以从系统中安全删除这些LUN。

```
SQL> select name||' '||group_number||' '||total_mb||' '||path||'
'||header_status from v$asm_disk;
NAME||''||GROUP_NUMBER||''||TOTAL_MB||''||PATH||''||HEADER_STATUS
--------------------------------------------------------------------------
------
 0 0 /dev/mapper/3600a098038303537762b47594c315863 FORMER
 0 0 /dev/mapper/3600a098038303537762b47594c315864 FORMER
 0 0 /dev/mapper/3600a098038303537762b47594c315861 FORMER
 0 0 /dev/mapper/3600a098038303537762b47594c315862 FORMER
NEWDATA_0005 1 10240 /dev/mapper/3600a098038303537762b47594c315869 MEMBER
NEWDATA_0007 1 10240 /dev/mapper/3600a098038303537762b47594c31586a MEMBER
NEWDATA_0004 1 10240 /dev/mapper/3600a098038303537762b47594c31586b MEMBER
NEWDATA_0006 1 10240 /dev/mapper/3600a098038303537762b47594c315858 MEMBER
8 rows selected.
```

LVM迁移

此处提供的操作步骤显示了对名为的卷组执行基于LVM的迁移的原则 `datavg`。这些示例取自Linux LVM、但这些原则同样适用于AIX、HP-UX和VLVM。具体命令可能有所不同。

1. 确定中当前的LUN `datavg` 卷组。

```
[root@host1 ~]# pvdisplay -C | grep datavg
  /dev/mapper/3600a098038303537762b47594c31582f datavg lvm2 a--  10.00g
10.00g
  /dev/mapper/3600a098038303537762b47594c31585a datavg lvm2 a--  10.00g
10.00g
  /dev/mapper/3600a098038303537762b47594c315859 datavg lvm2 a--  10.00g
10.00g
  /dev/mapper/3600a098038303537762b47594c31586c datavg lvm2 a--  10.00g
10.00g
```

2. 创建物理大小相同或略大的新LUN、并将其定义为物理卷。

```
[root@host1 ~]# pvcreate /dev/mapper/3600a098038303537762b47594c315864
  Physical volume "/dev/mapper/3600a098038303537762b47594c315864"
successfully created
[root@host1 ~]# pvcreate /dev/mapper/3600a098038303537762b47594c315863
  Physical volume "/dev/mapper/3600a098038303537762b47594c315863"
successfully created
[root@host1 ~]# pvcreate /dev/mapper/3600a098038303537762b47594c315862
  Physical volume "/dev/mapper/3600a098038303537762b47594c315862"
successfully created
[root@host1 ~]# pvcreate /dev/mapper/3600a098038303537762b47594c315861
  Physical volume "/dev/mapper/3600a098038303537762b47594c315861"
successfully created
```

3. 将新卷添加到卷组。

```
[root@host1 tmp]# vgextend datavg
/dev/mapper/3600a098038303537762b47594c315864
  Volume group "datavg" successfully extended
[root@host1 tmp]# vgextend datavg
/dev/mapper/3600a098038303537762b47594c315863
  Volume group "datavg" successfully extended
[root@host1 tmp]# vgextend datavg
/dev/mapper/3600a098038303537762b47594c315862
  Volume group "datavg" successfully extended
[root@host1 tmp]# vgextend datavg
/dev/mapper/3600a098038303537762b47594c315861
  Volume group "datavg" successfully extended
```

4. 问题描述 `pvmove` 命令将每个当前LUN的块区重新定位到新LUN。。 `- i [seconds]` 参数用于监控操作的进度。

```
[root@host1 tmp]# pvmove -i 10
/dev/mapper/3600a098038303537762b47594c31582f
/dev/mapper/3600a098038303537762b47594c315864
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 0.0%
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 14.2%
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 28.4%
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 42.5%
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 57.1%
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 72.3%
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 87.3%
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 100.0%
[root@host1 tmp]# pvmove -i 10
/dev/mapper/3600a098038303537762b47594c31585a
/dev/mapper/3600a098038303537762b47594c315863
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 0.0%
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 14.9%
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 29.9%
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 44.8%
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 60.1%
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 75.8%
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 90.9%
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 100.0%
[root@host1 tmp]# pvmove -i 10
/dev/mapper/3600a098038303537762b47594c315859
/dev/mapper/3600a098038303537762b47594c315862
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 0.0%
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 14.8%
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 29.8%
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 45.5%
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 61.1%
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 76.6%
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 91.7%
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 100.0%
[root@host1 tmp]# pvmove -i 10
/dev/mapper/3600a098038303537762b47594c31586c
/dev/mapper/3600a098038303537762b47594c315861
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 0.0%
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 15.0%
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 30.4%
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 46.0%
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 61.4%
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 77.2%
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 92.3%
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 100.0%
```

5. 此过程完成后、使用从卷组中删除旧LUN `vgreduce` 命令：如果成功、现在可以从系统中安全地删除此LUN。

```
[root@host1 tmp]# vgreduce datavg
/dev/mapper/3600a098038303537762b47594c31582f
Removed "/dev/mapper/3600a098038303537762b47594c31582f" from volume
group "datavg"
[root@host1 tmp]# vgreduce datavg
/dev/mapper/3600a098038303537762b47594c31585a
  Removed "/dev/mapper/3600a098038303537762b47594c31585a" from volume
group "datavg"
[root@host1 tmp]# vgreduce datavg
/dev/mapper/3600a098038303537762b47594c315859
  Removed "/dev/mapper/3600a098038303537762b47594c315859" from volume
group "datavg"
[root@host1 tmp]# vgreduce datavg
/dev/mapper/3600a098038303537762b47594c31586c
  Removed "/dev/mapper/3600a098038303537762b47594c31586c" from volume
group "datavg"
```

外部LUN导入

使用FLI迁移Oracle—规划

NetApp中介绍了使用FLI迁移SAN资源的过程 "TR-4380：《使用外部LUN导入进行SAN迁移》"。

从数据库和主机的角度来看、不需要执行任何特殊步骤。更新FC分区并使LUN在ONTAP上可用后、LVM应能够从LUN中读取LVM元数据。此外、卷组已准备就绪、无需执行其他配置步骤。在极少数情况下、环境可能会包含使用先前存储阵列的引用进行硬编码的配置文件。例如、包含的Linux系统 `/etc/multipath.conf` 必须更新引用给定设备的WWN的规则、以反映FLI所做的更改。

> 有关支持的配置的信息、请参见NetApp兼容性列表。如果您的环境未包括在其中、请与NetApp代表联系以获得帮助。

此示例显示了Linux服务器上托管的ASM和LVM LUN的迁移。FLI在其他操作系统上受支持、尽管主机端命令可能不同、但原则相同、ONTAP过程相同。

确定LVM LUN

准备工作的第一步是确定要迁移的LUN。在此处显示的示例中、两个基于SAN的文件系统挂载在上 `/orabin` 和 `/backups`。

```
[root@host1 ~]# df -k
Filesystem                        1K-blocks      Used Available Use%
Mounted on
/dev/mapper/rhel-root              52403200   8811464  43591736  17% /
devtmpfs                           65882776         0  65882776   0% /dev
...
fas8060-nfs-public:/install      199229440 119368128  79861312  60%
/install
/dev/mapper/sanvg-lvorabin         20961280  12348476   8612804  59%
/orabin
/dev/mapper/sanvg-lvbackups        73364480  62947536  10416944  86%
/backups
```

可以从设备名称中提取卷组的名称、该名称采用格式(卷组名称)-(逻辑卷名称)。在这种情况下、卷组称为 `sanvg`。

。 `pvdisplay` 命令可按如下所示来确定支持此卷组的LUN。在这种情况下、包含10个LUN `sanvg` 卷组。

```
[root@host1 ~]# pvdisplay -C -o pv_name,pv_size,pv_fmt,vg_name
  PV                                            PSize   VG
  /dev/mapper/3600a0980383030445424487556574266  10.00g sanvg
  /dev/mapper/3600a0980383030445424487556574267  10.00g sanvg
  /dev/mapper/3600a0980383030445424487556574268  10.00g sanvg
  /dev/mapper/3600a0980383030445424487556574269  10.00g sanvg
  /dev/mapper/3600a0980383030445424487556574 26a  10.00g sanvg
  /dev/mapper/3600a0980383030445424487556574 26b  10.00g sanvg
  /dev/mapper/3600a0980383030445424487556574 26c  10.00g sanvg
  /dev/mapper/3600a0980383030445424487556574 26d  10.00g sanvg
  /dev/mapper/3600a0980383030445424487556574 26e  10.00g sanvg
  /dev/mapper/3600a0980383030445424487556574 26f  10.00g sanvg
  /dev/sda2                                     278.38g rhel
```

确定ASM LUN

此外、还必须迁移ASM LUN。要以sysasm用户身份从sqlplus获取LUN和LUN路径数、请运行以下命令：

```
SQL> select path||' '||os_mb from v$asm_disk;
PATH||''||OS_MB
--------------------------------------------------------------------------
------
/dev/oracleasm/disks/ASM0 10240
/dev/oracleasm/disks/ASM9 10240
/dev/oracleasm/disks/ASM8 10240
/dev/oracleasm/disks/ASM7 10240
/dev/oracleasm/disks/ASM6 10240
/dev/oracleasm/disks/ASM5 10240
/dev/oracleasm/disks/ASM4 10240
/dev/oracleasm/disks/ASM1 10240
/dev/oracleasm/disks/ASM3 10240
/dev/oracleasm/disks/ASM2 10240
10 rows selected.
SQL>
```

FC网络更改

当前环境包含20个要迁移的LUN。更新当前SAN、以便ONTAP可以访问当前LUN。尚未迁移数据、但ONTAP必须从当前LUN中读取配置信息、才能为该数据创建新的主目录。

至少必须将AF/FAS系统上的一个HBA端口配置为启动程序端口。此外、必须更新FC分区、以便ONTAP可以访问外部存储阵列上的LUN。某些存储阵列配置了LUN屏蔽、用于限制哪些WWN可以访问给定LUN。在这种情况下、还必须更新LUN屏蔽以授予对ONTAP WWN的访问权限。

完成此步骤后、ONTAP应能够使用查看外部存储阵列 `storage array show` 命令：它返回的关键字段是用于标识系统上的外部LUN的前缀。在以下示例中、是外部阵列上的LUN `FOREIGN_1` 在ONTAP中显示、并使用前缀 `FOR-1`。

确定外部阵列

```
Cluster01::> storage array show -fields name,prefix
name          prefix
------------- ------
FOREIGN_1     FOR-1
Cluster01::>
```

确定外部**LUN**

通过传递、可以列出这些LUN `array-name` 到 `storage disk show` 命令：在迁移操作步骤期间、系统会多次引用返回的数据。

```
Cluster01::> storage disk show -array-name FOREIGN_1 -fields disk,serial
disk     serial-number
-------- -------------
FOR-1.1  800DT$HuVWBX
FOR-1.2  800DT$HuVWBZ
FOR-1.3  800DT$HuVWBW
FOR-1.4  800DT$HuVWBY
FOR-1.5  800DT$HuVWB/
FOR-1.6  800DT$HuVWBa
FOR-1.7  800DT$HuVWBd
FOR-1.8  800DT$HuVWBb
FOR-1.9  800DT$HuVWBc
FOR-1.10 800DT$HuVWBe
FOR-1.11 800DT$HuVWBf
FOR-1.12 800DT$HuVWBg
FOR-1.13 800DT$HuVWBi
FOR-1.14 800DT$HuVWBh
FOR-1.15 800DT$HuVWBj
FOR-1.16 800DT$HuVWBk
FOR-1.17 800DT$HuVWBm
FOR-1.18 800DT$HuVWBl
FOR-1.19 800DT$HuVWBo
FOR-1.20 800DT$HuVWBn
20 entries were displayed.
Cluster01::>
```

将外部阵列**LUN**注册为候选导入阵列

外部LUN最初归类为任何特定的LUN类型。在导入数据之前、必须将LUN标记为外部LUN、从而使其成为导入过程的候选LUN。此步骤可通过将序列号传递到来完成 `storage disk modify` 命令、如以下示例所示。请注意、此过程仅会将LUN标记为ONTAP中的外部LUN。不会向外部LUN本身写入任何数据。

```
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBW} -is
-foreign true
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBX} -is
-foreign true
...
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBn} -is
-foreign true
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBo} -is
-foreign true
Cluster01::*>
```

创建卷以托管迁移的**LUN**

托管迁移的LUN需要一个卷。确切的卷配置取决于利用ONTAP功能的整体计划。在此示例中、ASM LUN放置在一个卷中、而LVM LUN放置在另一个卷中。这样、您就可以将LUN作为独立的组进行管理、以实现分层、创建快照或设置QoS控制等目的。

设置 `snapshot-policy `to `none`。迁移过程中可能会涉及大量的数据周转。因此、如果由于在快照中捕获不需要的数据而意外创建快照、则空间消耗可能会大幅增加。

```
Cluster01::> volume create -volume new_asm -aggregate data_02 -size 120G
-snapshot-policy none
[Job 1152] Job succeeded: Successful
Cluster01::> volume create -volume new_lvm -aggregate data_02 -size 120G
-snapshot-policy none
[Job 1153] Job succeeded: Successful
Cluster01::>
```

创建**ONTAP LUN**

创建卷后、必须创建新的LUN。通常、创建LUN需要用户指定LUN大小等信息、但在这种情况下、外部磁盘参数会传递到命令。因此、ONTAP会从指定序列号复制当前LUN配置数据。它还会使用LUN几何结构和分区表数据来调整LUN对齐并建立最佳性能。

在此步骤中、必须对照外部阵列交叉引用序列号、以确保正确的外部LUN与正确的新LUN匹配。

```
Cluster01::*> lun create -vserver vserver1 -path /vol/new_asm/LUN0 -ostype
linux -foreign-disk 800DT$HuVWBW
Created a LUN of size 10g (10737418240)
Cluster01::*> lun create -vserver vserver1 -path /vol/new_asm/LUN1 -ostype
linux -foreign-disk 800DT$HuVWBX
Created a LUN of size 10g (10737418240)
...
Created a LUN of size 10g (10737418240)
Cluster01::*> lun create -vserver vserver1 -path /vol/new_lvm/LUN8 -ostype
linux -foreign-disk 800DT$HuVWBn
Created a LUN of size 10g (10737418240)
Cluster01::*> lun create -vserver vserver1 -path /vol/new_lvm/LUN9 -ostype
linux -foreign-disk 800DT$HuVWBo
Created a LUN of size 10g (10737418240)
```

创建导入关系

LUN现在已创建、但尚未配置为复制目标。在执行此步骤之前、必须先将LUN置于脱机状态。这一额外步骤旨在保护数据免受用户错误的影响。如果ONTAP允许对联机LUN执行迁移、则会存在一个风险、即因出现输入错误而可能会覆盖活动数据。强制用户首先使LUN脱机这一额外步骤有助于验证是否将正确的目标LUN用作迁移目标。

```
Cluster01::*> lun offline -vserver vserver1 -path /vol/new_asm/LUN0
Warning: This command will take LUN "/vol/new_asm/LUN0" in Vserver
         "vserver1" offline.
Do you want to continue? {y|n}: y
Cluster01::*> lun offline -vserver vserver1 -path /vol/new_asm/LUN1
Warning: This command will take LUN "/vol/new_asm/LUN1" in Vserver
         "vserver1" offline.
Do you want to continue? {y|n}: y
...
Warning: This command will take LUN "/vol/new_lvm/LUN8" in Vserver
         "vserver1" offline.
Do you want to continue? {y|n}: y
Cluster01::*> lun offline -vserver vserver1 -path /vol/new_lvm/LUN9
Warning: This command will take LUN "/vol/new_lvm/LUN9" in Vserver
         "vserver1" offline.
Do you want to continue? {y|n}: y
```

LUN脱机后、您可以通过将外部LUN序列号传递到来建立导入关系 `lun import create` 命令：

```
Cluster01::*> lun import create -vserver vserver1 -path /vol/new_asm/LUN0
-foreign-disk 800DT$HuVWBW
Cluster01::*> lun import create -vserver vserver1 -path /vol/new_asm/LUN1
-foreign-disk 800DT$HuVWBX
...
Cluster01::*> lun import create -vserver vserver1 -path /vol/new_lvm/LUN8
-foreign-disk 800DT$HuVWBn
Cluster01::*> lun import create -vserver vserver1 -path /vol/new_lvm/LUN9
-foreign-disk 800DT$HuVWBo
Cluster01::*>
```

建立所有导入关系后、可以将LUN重新置于联机状态。

```
Cluster01::*> lun online -vserver vserver1 -path /vol/new_asm/LUN0
Cluster01::*> lun online -vserver vserver1 -path /vol/new_asm/LUN1
...
Cluster01::*> lun online -vserver vserver1 -path /vol/new_lvm/LUN8
Cluster01::*> lun online -vserver vserver1 -path /vol/new_lvm/LUN9
Cluster01::*>
```

创建启动程序组

启动程序组(igroGroup)是ONTAP LUN屏蔽架构的一部分。除非先授予主机访问权限、否则无法访问新创建的LUN。为此、可创建一个igrop、其中列出应授予访问权限的FC WWN或iSCSI启动程序名称。编写此报告时、

只有FC LUN支持FLI。但是、迁移后转换为iSCSI是一项简单的任务、如所示 "协议转换"。

在此示例中、创建了一个igrop、其中包含两个WWN、分别对应于主机HBA上的两个可用端口。

```
Cluster01::*> igroup create linuxhost -protocol fcp -ostype linux
-initiator 21:00:00:0e:1e:16:63:50 21:00:00:0e:1e:16:63:51
```

将新LUN映射到主机

创建igroup后、LUN将映射到定义的igroup。这些LUN仅可供此igrop中包含的WWN使用。在迁移过程的这一阶段、NetApp会假定主机尚未分区到ONTAP。这一点非常重要、因为如果将主机同时分区到外部阵列和新的ONTAP系统、则可能会在每个阵列上发现具有相同序列号的LUN。这种情况可能会导致多路径故障或数据损坏。

```
Cluster01::*> lun map -vserver vserver1 -path /vol/new_asm/LUN0 -igroup
linuxhost
Cluster01::*> lun map -vserver vserver1 -path /vol/new_asm/LUN1 -igroup
linuxhost
...
Cluster01::*> lun map -vserver vserver1 -path /vol/new_lvm/LUN8 -igroup
linuxhost
Cluster01::*> lun map -vserver vserver1 -path /vol/new_lvm/LUN9 -igroup
linuxhost
Cluster01::*>
```

使用FLI迁移Oracle—转换

由于需要更改FC网络配置、外部LUN导入期间不可避免地会发生某些中断。但是、中断的持续时间不必比重新启动数据库环境并更新FC分区以将主机FC连接从外部LUN切换到ONTAP所需的时间长。

此过程可概括如下：

1. 将外部LUN上的所有LUN活动置于静噪状态。
2. 将主机FC连接重定向到新的ONTAP系统。
3. 触发导入过程。
4. 重新发现LUN。
5. 重新启动数据库。

您无需等待迁移过程完成。给定LUN的迁移开始后、该LUN便可在ONTAP上使用、并可在数据复制过程继续期间提供数据。所有读取都会传递到外部LUN、所有写入都会同步写入到两个阵列。复制操作速度非常快、重定向FC流量的开销也非常小、因此对性能的任何影响都应该是瞬时的、并且最小化。如果有问题、您可以延迟重新启动环境、直到迁移过程完成并删除导入关系之后。

关闭数据库

在此示例中、静音环境的第一步是关闭数据库。

```
[oracle@host1 bin]$ . oraenv
ORACLE_SID = [oracle] ? FLIDB
The Oracle base remains unchanged with value /orabin
[oracle@host1 bin]$ sqlplus / as sysdba
SQL*Plus: Release 12.1.0.2.0
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit
Production
With the Partitioning, Automatic Storage Management, OLAP, Advanced
Analytics
and Real Application Testing options
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
```

关闭网格服务

要迁移的基于SAN的文件系统之一还包括Oracle ASM服务。暂停底层LUN需要卸载文件系统、这反过来意味着停止此文件系统上具有已打开文件的所有进程。

```
[oracle@host1 bin]$ ./crsctl stop has -f
CRS-2791: Starting shutdown of Oracle High Availability Services-managed
resources on 'host1'
CRS-2673: Attempting to stop 'ora.evmd' on 'host1'
CRS-2673: Attempting to stop 'ora.DATA.dg' on 'host1'
CRS-2673: Attempting to stop 'ora.LISTENER.lsnr' on 'host1'
CRS-2677: Stop of 'ora.DATA.dg' on 'host1' succeeded
CRS-2673: Attempting to stop 'ora.asm' on 'host1'
CRS-2677: Stop of 'ora.LISTENER.lsnr' on 'host1' succeeded
CRS-2677: Stop of 'ora.evmd' on 'host1' succeeded
CRS-2677: Stop of 'ora.asm' on 'host1' succeeded
CRS-2673: Attempting to stop 'ora.cssd' on 'host1'
CRS-2677: Stop of 'ora.cssd' on 'host1' succeeded
CRS-2793: Shutdown of Oracle High Availability Services-managed resources
on 'host1' has completed
CRS-4133: Oracle High Availability Services has been stopped.
[oracle@host1 bin]$
```

卸载文件系统

如果所有进程均已关闭、则卸载操作将成功。如果权限被拒绝、则文件系统上必须存在一个具有锁定的进程。。`fuser` 命令有助于识别这些进程。

```
[root@host1 ~]# umount /orabin
[root@host1 ~]# umount /backups
```

停用卷组

卸载给定卷组中的所有文件系统后、可以停用该卷组。

```
[root@host1 ~]# vgchange --activate n sanvg
  0 logical volume(s) in volume group "sanvg" now active
[root@host1 ~]#
```

FC网络更改

现在、可以更新FC分区、以删除主机对外部阵列的所有访问权限、并建立对ONTAP的访问权限。

启动导入过程

要启动LUN导入过程、请运行 `lun import start` 命令：

```
Cluster01::lun import*> lun import start -vserver vserver1 -path
/vol/new_asm/LUN0
Cluster01::lun import*> lun import start -vserver vserver1 -path
/vol/new_asm/LUN1
...
Cluster01::lun import*> lun import start -vserver vserver1 -path
/vol/new_lvm/LUN8
Cluster01::lun import*> lun import start -vserver vserver1 -path
/vol/new_lvm/LUN9
Cluster01::lun import*>
```

监控导入进度

可以使用监控导入操作 `lun import show` 命令：如下图所示、所有20个LUN的导入正在进行中、这意味着现在可以通过ONTAP访问数据、即使数据复制操作仍在进行。

```
Cluster01::lun import*> lun import show -fields path,percent-complete
vserver   foreign-disk path              percent-complete
--------- ------------ ----------------- ----------------
vserver1  800DT$HuVWB/ /vol/new_asm/LUN4 5
vserver1  800DT$HuVWBW /vol/new_asm/LUN0 5
vserver1  800DT$HuVWBX /vol/new_asm/LUN1 6
vserver1  800DT$HuVWBY /vol/new_asm/LUN2 6
vserver1  800DT$HuVWBZ /vol/new_asm/LUN3 5
vserver1  800DT$HuVWBa /vol/new_asm/LUN5 4
vserver1  800DT$HuVWBb /vol/new_asm/LUN6 4
vserver1  800DT$HuVWBc /vol/new_asm/LUN7 4
vserver1  800DT$HuVWBd /vol/new_asm/LUN8 4
vserver1  800DT$HuVWBe /vol/new_asm/LUN9 4
vserver1  800DT$HuVWBf /vol/new_lvm/LUN0 5
vserver1  800DT$HuVWBg /vol/new_lvm/LUN1 4
vserver1  800DT$HuVWBh /vol/new_lvm/LUN2 4
vserver1  800DT$HuVWBi /vol/new_lvm/LUN3 3
vserver1  800DT$HuVWBj /vol/new_lvm/LUN4 3
vserver1  800DT$HuVWBk /vol/new_lvm/LUN5 3
vserver1  800DT$HuVWBl /vol/new_lvm/LUN6 4
vserver1  800DT$HuVWBm /vol/new_lvm/LUN7 3
vserver1  800DT$HuVWBn /vol/new_lvm/LUN8 2
vserver1  800DT$HuVWBo /vol/new_lvm/LUN9 2
20 entries were displayed.
```

如果需要脱机进程、请将重新发现或重新启动服务延迟到 `lun import show` 命令表示所有迁移均已成功完成。然后、您可以按照中所述完成迁移过程 "外部LUN导入—完成"。

如果需要联机迁移、请继续在新主目录中重新发现LUN并启动服务。

扫描**SCSI**设备更改

在大多数情况下、重新发现新LUN的最简单方法是重新启动主机。这样做会自动删除旧的陈旧设备、正确发现所有新LUN并构建关联的设备、例如多路径设备。此处的示例显示了一个完全联机的流程、用于演示目的。

注意：重新启动主机之前、请确保中的所有条目都已启用 `/etc/fstab` 此参考迁移的SAN资源已被注释掉。如果不执行此操作、并且LUN访问出现问题、则操作系统可能无法启动。这种情况不会损坏数据。但是、启动到救援模式或类似模式并更正可能会非常不方便 `/etc/fstab` 以便可以启动操作系统以启用故障排除。

可以使用重新扫描此示例中使用的Linux版本上的LUN `rescan-scsi-bus.sh` 命令：如果命令成功、则输出中应显示每个LUN路径。输出可能难以解释、但如果分区和igrop配置正确、则应显示许多LUN包含 `NETAPP` 供应商字符串。

```
[root@host1 /]# rescan-scsi-bus.sh
Scanning SCSI subsystem for new devices
Scanning host 0 for  SCSI target IDs  0 1 2 3 4 5 6 7, all LUNs
 Scanning for device 0 2 0 0 ...
OLD: Host: scsi0 Channel: 02 Id: 00 Lun: 00
      Vendor: LSI      Model: RAID SAS 6G 0/1  Rev: 2.13
      Type:   Direct-Access                    ANSI SCSI revision: 05
Scanning host 1 for  SCSI target IDs  0 1 2 3 4 5 6 7, all LUNs
 Scanning for device 1 0 0 0 ...
OLD: Host: scsi1 Channel: 00 Id: 00 Lun: 00
      Vendor: Optiarc  Model: DVD RW AD-7760H  Rev: 1.41
      Type:   CD-ROM                           ANSI SCSI revision: 05
Scanning host 2 for  SCSI target IDs  0 1 2 3 4 5 6 7, all LUNs
Scanning host 3 for  SCSI target IDs  0 1 2 3 4 5 6 7, all LUNs
Scanning host 4 for  SCSI target IDs  0 1 2 3 4 5 6 7, all LUNs
Scanning host 5 for  SCSI target IDs  0 1 2 3 4 5 6 7, all LUNs
Scanning host 6 for  SCSI target IDs  0 1 2 3 4 5 6 7, all LUNs
Scanning host 7 for  all SCSI target IDs, all LUNs
 Scanning for device 7 0 0 10 ...
OLD: Host: scsi7 Channel: 00 Id: 00 Lun: 10
      Vendor: NETAPP   Model: LUN C-Mode       Rev: 8300
      Type:   Direct-Access                    ANSI SCSI revision: 05
 Scanning for device 7 0 0 11 ...
OLD: Host: scsi7 Channel: 00 Id: 00 Lun: 11
      Vendor: NETAPP   Model: LUN C-Mode       Rev: 8300
      Type:   Direct-Access                    ANSI SCSI revision: 05
 Scanning for device 7 0 0 12 ...
...
OLD: Host: scsi9 Channel: 00 Id: 01 Lun: 18
      Vendor: NETAPP   Model: LUN C-Mode       Rev: 8300
      Type:   Direct-Access                    ANSI SCSI revision: 05
 Scanning for device 9 0 1 19 ...
OLD: Host: scsi9 Channel: 00 Id: 01 Lun: 19
      Vendor: NETAPP   Model: LUN C-Mode       Rev: 8300
      Type:   Direct-Access                    ANSI SCSI revision: 05
0 new or changed device(s) found.
0 remapped or resized device(s) found.
0 device(s) removed.
```

检查多路径设备

LUN发现过程还会触发多路径设备的重新创建、但已知Linux多路径驱动程序偶尔会出现问题。的输出 `multipath - ll` 应进行检查、以验证输出是否如预期。例如、以下输出显示了与关联的多路径设备 `NETAPP` 供应商字符串。每个设备都有四个路径、其中两个路径的优先级为50、两个路径的优先级为10。尽管不同版本的Linux的确切输出可能会有所不同、但此输出看起来与预期一致。

请参考用于验证的Linux版本的Host Utilities文档 `/etc/multipath.conf` 设置正确。

```
[root@host1 /]# multipath -ll
3600a098038303558735d493762504b36 dm-5 NETAPP  ,LUN C-Mode
size=10G features='4 queue_if_no_path pg_init_retries 50
retain_attached_hw_handle' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| |- 7:0:1:4  sdat 66:208 active ready running
| `- 9:0:1:4  sdbn 68:16  active ready running
`-+- policy='service-time 0' prio=10 status=enabled
  |- 7:0:0:4  sdf  8:80   active ready running
  `- 9:0:0:4  sdz  65:144 active ready running
3600a098038303558735d493762504b2d dm-10 NETAPP  ,LUN C-Mode
size=10G features='4 queue_if_no_path pg_init_retries 50
retain_attached_hw_handle' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| |- 7:0:1:8  sdax 67:16  active ready running
| `- 9:0:1:8  sdbr 68:80  active ready running
`-+- policy='service-time 0' prio=10 status=enabled
  |- 7:0:0:8  sdj  8:144  active ready running
  `- 9:0:0:8  sdad 65:208 active ready running
...
3600a098038303558735d493762504b37 dm-8 NETAPP  ,LUN C-Mode
size=10G features='4 queue_if_no_path pg_init_retries 50
retain_attached_hw_handle' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| |- 7:0:1:5  sdau 66:224 active ready running
| `- 9:0:1:5  sdbo 68:32  active ready running
`-+- policy='service-time 0' prio=10 status=enabled
  |- 7:0:0:5  sdg  8:96   active ready running
  `- 9:0:0:5  sdaa 65:160 active ready running
3600a098038303558735d493762504b4b dm-22 NETAPP  ,LUN C-Mode
size=10G features='4 queue_if_no_path pg_init_retries 50
retain_attached_hw_handle' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| |- 7:0:1:19 sdbi 67:192 active ready running
| `- 9:0:1:19 sdcc 69:0   active ready running
`-+- policy='service-time 0' prio=10 status=enabled
  |- 7:0:0:19 sdu  65:64  active ready running
  `- 9:0:0:19 sdao 66:128 active ready running
```

重新激活**LVM**卷组

如果已正确发现LVM LUN、则 `vgchange --activate y` 命令应成功。这是一个很好的逻辑卷管理器价值示例。更改LUN的WWN甚至序列号并不重要、因为卷组元数据会写入LUN本身。

操作系统扫描了LUN、发现写入LUN上的少量数据、将其标识为属于的物理卷 `sanvg volumegroup`。然后构建所有必需的设备。只需重新激活卷组即可。

```
[root@host1 /]# vgchange --activate y sanvg
  Found duplicate PV fpCzdLTuKfy2xDZjai1NliJh3TjLUBiT: using
/dev/mapper/3600a098038303558735d493762504b46 not /dev/sdp
  Using duplicate PV /dev/mapper/3600a098038303558735d493762504b46 from
subsystem DM, ignoring /dev/sdp
  2 logical volume(s) in volume group "sanvg" now active
```

重新挂载文件系统

重新激活卷组后、可以在挂载文件系统时保持所有原始数据完好无损。如前文所述、即使数据复制在后端组中仍处于活动状态、文件系统也能完全正常运行。

```
[root@host1 /]# mount /orabin
[root@host1 /]# mount /backups
[root@host1 /]# df -k
Filesystem                        1K-blocks      Used Available Use%
Mounted on
/dev/mapper/rhel-root              52403200   8837100  43566100  17% /
devtmpfs                           65882776         0  65882776   0% /dev
tmpfs                               6291456        84   6291372   1%
/dev/shm
tmpfs                              65898668      9884  65888784   1% /run
tmpfs                              65898668         0  65898668   0%
/sys/fs/cgroup
/dev/sda1                            505580    224828    280752  45% /boot
fas8060-nfs-public:/install      199229440 119368256  79861184  60%
/install
fas8040-nfs-routable:/snapomatic   9961472     30528   9930944   1%
/snapomatic
tmpfs                              13179736        16  13179720   1%
/run/user/42
tmpfs                              13179736         0  13179736   0%
/run/user/0
/dev/mapper/sanvg-lvorabin         20961280  12357456   8603824  59%
/orabin
/dev/mapper/sanvg-lvbackups        73364480  62947536  10416944  86%
/backups
```

重新扫描**ASM**设备

重新扫描SCSI设备时、应已重新发现ASMlib设备。可以通过重新启动ASMlib并扫描磁盘来联机验证重新发现。

此步骤仅与使用ASMlib的ASM配置相关。

注意：如果未使用ASMlib、则为 `/dev/mapper` 设备应已自动重新创建。但是、权限可能不正确。如果没有ASMlib、则必须在底层设备上为ASM设置特殊权限。通常通过任一中的特殊条目来完成此操作 `/etc/multipath.conf` 或 `udev` 规则、或者可能同时位于这两个规则集中。可能需要更新这些文件、以反映环境中的WWN或序列号变化、从而确保ASM设备仍具有正确的权限。

在此示例中、重新启动ASMlib并扫描磁盘会显示与原始环境相同的10个ASM LUN。

```
[root@host1 /]# oracleasm exit
Unmounting ASMlib driver filesystem: /dev/oracleasm
Unloading module "oracleasm": oracleasm
[root@host1 /]# oracleasm init
Loading module "oracleasm": oracleasm
Configuring "oracleasm" to use device physical block size
Mounting ASMlib driver filesystem: /dev/oracleasm
[root@host1 /]# oracleasm scandisks
Reloading disk partitions: done
Cleaning any stale ASM disks...
Scanning system for ASM disks...
Instantiating disk "ASM0"
Instantiating disk "ASM1"
Instantiating disk "ASM2"
Instantiating disk "ASM3"
Instantiating disk "ASM4"
Instantiating disk "ASM5"
Instantiating disk "ASM6"
Instantiating disk "ASM7"
Instantiating disk "ASM8"
Instantiating disk "ASM9"
```

重新启动网格服务

现在LVM和ASM设备已联机且可用、可以重新启动网格服务。

```
[root@host1 /]# cd /orabin/product/12.1.0/grid/bin
[root@host1 bin]# ./crsctl start has
```

重新启动数据库

重新启动网格服务后、可以启动数据库。在尝试启动数据库之前、可能需要等待几分钟、以便ASM服务完全可用。

```
[root@host1 bin]# su - oracle
[oracle@host1 ~]$ . oraenv
ORACLE_SID = [oracle] ? FLIDB
The Oracle base has been set to /orabin
[oracle@host1 ~]$ sqlplus / as sysdba
SQL*Plus: Release 12.1.0.2.0
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to an idle instance.
SQL> startup
ORACLE instance started.
Total System Global Area 3221225472 bytes
Fixed Size                  4502416 bytes
Variable Size            1207962736 bytes
Database Buffers         1996488704 bytes
Redo Buffers               12271616 bytes
Database mounted.
Database opened.
SQL>
```

使用**FLI**迁移**Oracle—**完成

从主机角度来看、迁移已完成、但仍会从外部阵列提供I/O、直到删除导入关系为止。

在删除关系之前、必须确认所有LUN的迁移过程均已完成。

```
Cluster01::*> lun import show -vserver vserver1 -fields foreign-
disk,path,operational-state
vserver   foreign-disk path              operational-state
--------- ------------ ----------------- -----------------
vserver1 800DT$HuVWB/ /vol/new_asm/LUN4 completed
vserver1 800DT$HuVWBW /vol/new_asm/LUN0 completed
vserver1 800DT$HuVWBX /vol/new_asm/LUN1 completed
vserver1 800DT$HuVWBY /vol/new_asm/LUN2 completed
vserver1 800DT$HuVWBZ /vol/new_asm/LUN3 completed
vserver1 800DT$HuVWBa /vol/new_asm/LUN5 completed
vserver1 800DT$HuVWBb /vol/new_asm/LUN6 completed
vserver1 800DT$HuVWBc /vol/new_asm/LUN7 completed
vserver1 800DT$HuVWBd /vol/new_asm/LUN8 completed
vserver1 800DT$HuVWBe /vol/new_asm/LUN9 completed
vserver1 800DT$HuVWBf /vol/new_lvm/LUN0 completed
vserver1 800DT$HuVWBg /vol/new_lvm/LUN1 completed
vserver1 800DT$HuVWBh /vol/new_lvm/LUN2 completed
vserver1 800DT$HuVWBi /vol/new_lvm/LUN3 completed
vserver1 800DT$HuVWBj /vol/new_lvm/LUN4 completed
vserver1 800DT$HuVWBk /vol/new_lvm/LUN5 completed
vserver1 800DT$HuVWBl /vol/new_lvm/LUN6 completed
vserver1 800DT$HuVWBm /vol/new_lvm/LUN7 completed
vserver1 800DT$HuVWBn /vol/new_lvm/LUN8 completed
vserver1 800DT$HuVWBo /vol/new_lvm/LUN9 completed
20 entries were displayed.
```

删除导入关系

迁移过程完成后、删除此迁移关系。完成此操作后、I/O将专门从ONTAP上的驱动器提供。

```
Cluster01::*> lun import delete -vserver vserver1 -path /vol/new_asm/LUN0
Cluster01::*> lun import delete -vserver vserver1 -path /vol/new_asm/LUN1
...
Cluster01::*> lun import delete -vserver vserver1 -path /vol/new_lvm/LUN8
Cluster01::*> lun import delete -vserver vserver1 -path /vol/new_lvm/LUN9
```

取消注册外部**LUN**

最后、修改磁盘以删除 `is-foreign` 名称。

```
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBW} -is
-foreign false
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBX} -is
-foreign false
...
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBn} -is
-foreign false
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBo} -is
-foreign false
Cluster01::*>
```

使用**FLI**迁移**Oracle—**协议转换

更改用于访问LUN的协议是一项常见要求。

在某些情况下、将数据迁移到云是整体战略的一部分。TCP/IP是云协议、从FC更改为iSCSI可以更轻松地迁移到各种云环境。在其他情况下、iSCSI可能是利用IP SAN降低的成本的理想选择。有时、迁移可能会使用不同的协议作为临时措施。例如、如果外部阵列和基于ONTAP的LUN不能同时位于同一HBA上、则可以使用iSCSI LUN足够长的时间来从旧阵列复制数据。从系统中删除旧LUN后、您可以将其转换回FC。

以下操作步骤演示了从FC到iSCSI的转换、但总体原则适用于从iSCSI到FC的反向转换。

安装**iSCSI**启动程序

默认情况下、大多数操作系统都包含软件iSCSI启动程序、但如果未包含、则可以轻松安装。

```
[root@host1 /]# yum install -y iscsi-initiator-utils
Loaded plugins: langpacks, product-id, search-disabled-repos,
subscription-
              : manager
Resolving Dependencies
--> Running transaction check
---> Package iscsi-initiator-utils.x86_64 0:6.2.0.873-32.el7 will be
updated
--> Processing Dependency: iscsi-initiator-utils = 6.2.0.873-32.el7 for
package: iscsi-initiator-utils-iscsiuio-6.2.0.873-32.el7.x86_64
---> Package iscsi-initiator-utils.x86_64 0:6.2.0.873-32.0.2.el7 will be
an update
--> Running transaction check
---> Package iscsi-initiator-utils-iscsiuio.x86_64 0:6.2.0.873-32.el7 will
be updated
---> Package iscsi-initiator-utils-iscsiuio.x86_64 0:6.2.0.873-32.0.2.el7
will be an update
--> Finished Dependency Resolution
Dependencies Resolved
=============================================================================
```

```
===
 Package                        Arch   Version              Repository
Size
================================================================================
===
Updating:
 iscsi-initiator-utils          x86_64 6.2.0.873-32.0.2.el7 ol7_latest 416
k
Updating for dependencies:
 iscsi-initiator-utils-iscsiuio x86_64 6.2.0.873-32.0.2.el7 ol7_latest  84
k
Transaction Summary
================================================================================
===
Upgrade  1 Package (+1 Dependent package)
Total download size: 501 k
Downloading packages:
No Presto metadata available for ol7_latest
(1/2): iscsi-initiator-utils-6.2.0.873-32.0.2.el7.x86_6 | 416 kB   00:00
(2/2): iscsi-initiator-utils-iscsiuio-6.2.0.873-32.0.2. |  84 kB   00:00
--------------------------------------------------------------------------------
---
Total                                           2.8 MB/s | 501 kB
00:00Cluster01
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Updating   : iscsi-initiator-utils-iscsiuio-6.2.0.873-32.0.2.el7.x86
1/4
  Updating   : iscsi-initiator-utils-6.2.0.873-32.0.2.el7.x86_64
2/4
  Cleanup    : iscsi-initiator-utils-iscsiuio-6.2.0.873-32.el7.x86_64
3/4
  Cleanup    : iscsi-initiator-utils-6.2.0.873-32.el7.x86_64
4/4
rhel-7-server-eus-rpms/7Server/x86_64/productid       | 1.7 kB     00:00
rhel-7-server-rpms/7Server/x86_64/productid           | 1.7 kB     00:00
  Verifying  : iscsi-initiator-utils-6.2.0.873-32.0.2.el7.x86_64
1/4
  Verifying  : iscsi-initiator-utils-iscsiuio-6.2.0.873-32.0.2.el7.x86
2/4
  Verifying  : iscsi-initiator-utils-iscsiuio-6.2.0.873-32.el7.x86_64
3/4
  Verifying  : iscsi-initiator-utils-6.2.0.873-32.el7.x86_64
4/4
```

```
Updated:
  iscsi-initiator-utils.x86_64 0:6.2.0.873-32.0.2.el7
Dependency Updated:
  iscsi-initiator-utils-iscsiuio.x86_64 0:6.2.0.873-32.0.2.el7
Complete!
[root@host1 /]#
```

确定**iSCSI**启动程序名称

在安装过程中会生成一个唯一的iSCSI启动程序名称。在Linux上、它位于中 `/etc/iscsi/initiatorname.iscsi` 文件此名称用于标识IP SAN上的主机。

```
[root@host1 /]# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.1992-05.com.redhat:497bd66ca0
```

创建新启动程序组

启动程序组(igroGroup)是ONTAP LUN屏蔽架构的一部分。除非先授予主机访问权限、否则无法访问新创建的LUN。完成此步骤的方法是创建一个igrop、其中列出了需要访问的FC WWN或iSCSI启动程序名称。

在此示例中、创建了一个igrop、其中包含Linux主机的iSCSI启动程序。

```
Cluster01::*> igroup create -igroup linuxiscsi -protocol iscsi -ostype
linux -initiator iqn.1994-05.com.redhat:497bd66ca0
```

关闭环境

在更改LUN协议之前、必须将LUN完全置于静状态。要转换的LUN之一上的任何数据库都必须关闭、文件系统必须卸载、卷组必须停用。如果使用ASM、请确保已卸载ASM磁盘组并关闭所有网格服务。

取消**LUN**与**FC**网络的映射

在LUN完全静置后、从原始FC igrop中删除映射。

```
Cluster01::*> lun unmap -vserver vserver1 -path /vol/new_asm/LUN0 -igroup
linuxhost
Cluster01::*> lun unmap -vserver vserver1 -path /vol/new_asm/LUN1 -igroup
linuxhost
...
Cluster01::*> lun unmap -vserver vserver1 -path /vol/new_lvm/LUN8 -igroup
linuxhost
Cluster01::*> lun unmap -vserver vserver1 -path /vol/new_lvm/LUN9 -igroup
linuxhost
```

将LUN重新映射到IP网络

将对每个LUN的访问权限授予新的基于iSCSI的启动程序组。

```
Cluster01::*> lun map -vserver vserver1 -path /vol/new_asm/LUN0 -igroup
linuxiscsi
Cluster01::*> lun map -vserver vserver1 -path /vol/new_asm/LUN1 -igroup
linuxiscsi
...
Cluster01::*> lun map -vserver vserver1 -path /vol/new_lvm/LUN8 -igroup
linuxiscsi
Cluster01::*> lun map -vserver vserver1 -path /vol/new_lvm/LUN9 -igroup
linuxiscsi
Cluster01::*>
```

发现iSCSI目标

iSCSI发现分为两个阶段。第一种方法是发现目标、这与发现LUN不同。。 `iscsiadm` 下面显示的命令用于探测由指定的门户组 `-p argument` 和用于存储提供iSCSI服务的所有IP地址和端口的列表。在这种情况下、有四个IP地址在默认端口3260上提供iSCSI服务。

> 如果无法访问任何目标IP地址、则此命令可能需要几分钟才能完成。

```
[root@host1 ~]# iscsiadm -m discovery -t st -p fas8060-iscsi-public1
10.63.147.197:3260,1033 iqn.1992-
08.com.netapp:sn.807615e9ef6111e5a5ae90e2ba5b9464:vs.3
10.63.147.198:3260,1034 iqn.1992-
08.com.netapp:sn.807615e9ef6111e5a5ae90e2ba5b9464:vs.3
172.20.108.203:3260,1030 iqn.1992-
08.com.netapp:sn.807615e9ef6111e5a5ae90e2ba5b9464:vs.3
172.20.108.202:3260,1029 iqn.1992-
08.com.netapp:sn.807615e9ef6111e5a5ae90e2ba5b9464:vs.3
```

发现iSCSI LUN

发现iSCSI目标后、重新启动iSCSI服务以发现可用的iSCSI LUN并构建关联设备、例如多路径或ASMlib设备。

```
[root@host1 ~]# service iscsi restart
Redirecting to /bin/systemctl restart  iscsi.service
```

重新启动环境

通过重新激活卷组、重新挂载文件系统、重新启动RAC服务等方式重新启动环境。作为预防措施、NetApp建议您在转换过程完成后重新启动服务器、以确保所有配置文件均正确无误、并且所有陈旧设备均已删除。

注意：重新启动主机之前、请确保中的所有条目都已启用 `/etc/fstab` 此参考迁移的SAN资源已被注释掉。如果未执行此步骤、并且LUN访问出现问题、则可能会导致操作系统无法启动。此问题描述不会损坏数据。但是、启动到救援模式或类似模式并进行更正可能非常不方便 `/etc/fstab` 以便可以启动操作系统、以便开始故障排除工作。

Oracle迁移操作步骤示例脚本

提供的脚本是如何为各种操作系统和数据库任务编写脚本的示例。它们按原样提供。如果特定操作步骤需要支持、请联系NetApp或NetApp经销商。

数据库关闭

以下Perl脚本仅使用Oracle SID的一个参数、并关闭数据库。它可以作为Oracle用户或root用户运行。

```
#! /usr/bin/perl
use strict;
use warnings;
my $oraclesid=$ARGV[0];
my $oracleuser='oracle';
my @out;
my $uid=$<;
if ($uid == 0) {
@out=`su - $oracleuser -c '. oraenv << EOF1
77 Migration of Oracle Databases to NetApp Storage Systems © 2021 NetApp,
Inc. All rights reserved
$oraclesid
EOF1
sqlplus / as sysdba << EOF2
shutdown immediate;
EOF2
'
`;}
else {
@out=`. oraenv << EOF1
$oraclesid
EOF4
sqlplus / as sysdba << EOF2
shutdown immediate;
EOF2
`;};
print @out;
if ("@out" =~ /ORACLE instance shut down/) {
print "$oraclesid shut down\n";
exit 0;}
elsif ("@out" =~ /Connected to an idle instance/) {
print "$oraclesid already shut down\n";
exit 0;}
else {
print "$oraclesid failed to shut down\n";
exit 1;}
```

数据库启动

以下Perl脚本仅使用Oracle SID的一个参数、并关闭数据库。它可以作为Oracle用户或root用户运行。

```
#! /usr/bin/perl
use strict;
use warnings;
my $oraclesid=$ARGV[0];
my $oracleuser='oracle';
my @out;
my $uid=$<;
if ($uid == 0) {
@out=`su - $oracleuser -c '. oraenv << EOF1
$oraclesid
EOF1
sqlplus / as sysdba << EOF2
startup;
EOF2
'
`;}
else {
@out=`. oraenv << EOF3
$oraclesid
EOF1
sqlplus / as sysdba << EOF2
startup;
EOF2
`;};
print @out;
if ("@out" =~ /Database opened/) {
print "$oraclesid started\n";
exit 0;}
elsif ("@out" =~ /cannot start already-running ORACLE/) {
print "$oraclesid already started\n";
exit 1;}
else {
78 Migration of Oracle Databases to NetApp Storage Systems © 2021 NetApp,
Inc. All rights reserved
print "$oraclesid failed to start\n";
exit 1;}
```

将文件系统转换为只读

以下脚本采用文件系统参数、并尝试卸载它、然后将其重新挂载为只读。在迁移过程中、这样做非常有用、因为在迁移过程中、文件系统必须保持可用性以复制数据、同时必须防止意外损坏。

```
#! /usr/bin/perl
use strict;
#use warnings;
my $filesystem=$ARGV[0];
my @out=`umount '$filesystem'`;
if ($? == 0) {
 print "$filesystem unmounted\n";
 @out = `mount -o ro '$filesystem'`;
 if ($? == 0) {
 print "$filesystem mounted read-only\n";
 exit 0;}}
else {
 print "Unable to unmount $filesystem\n";
 exit 1;}
print @out;
```

替换文件系统

以下脚本示例用于将一个文件系统替换为另一个文件系统。由于它会编辑`/etc/fstab `文件、因此必须以root用户身份运行。它接受新旧文件系统的一个逗号分隔参数。

1. 要替换文件系统、请运行以下脚本：

```
#! /usr/bin/perl
use strict;
#use warnings;
my $oldfs;
my $newfs;
my @oldfstab;
my @newfstab;
my $source;
my $mountpoint;
my $leftover;
my $oldfstabentry='';
my $newfstabentry='';
my $migratedfstabentry='';
($oldfs, $newfs) = split (',',$ARGV[0]);
open(my $filehandle, '<', '/etc/fstab') or die "Could not open
/etc/fstab\n";
while (my $line = <$filehandle>) {
 chomp $line;
 ($source, $mountpoint, $leftover) = split(/[ , ]/,$line, 3);
 if ($mountpoint eq $oldfs) {
 $oldfstabentry = "#Removed by swap script $source $oldfs $leftover";}
 elsif ($mountpoint eq $newfs) {
```

```
 $newfstabentry = "#Removed by swap script $source $newfs $leftover";
 $migratedfstabentry = "$source $oldfs $leftover";}
 else {
 push (@newfstab, "$line\n")}}
79 Migration of Oracle Databases to NetApp Storage Systems © 2021
NetApp, Inc. All rights reserved
push (@newfstab, "$oldfstabentry\n");
push (@newfstab, "$newfstabentry\n");
push (@newfstab, "$migratedfstabentry\n");
close($filehandle);
if ($oldfstabentry eq ''){
 die "Could not find $oldfs in /etc/fstab\n";}
if ($newfstabentry eq ''){
 die "Could not find $newfs in /etc/fstab\n";}
my @out=`umount '$newfs'`;
if ($? == 0) {
 print "$newfs unmounted\n";}
else {
 print "Unable to unmount $newfs\n";
 exit 1;}
@out=`umount '$oldfs'`;
if ($? == 0) {
 print "$oldfs unmounted\n";}
else {
 print "Unable to unmount $oldfs\n";
 exit 1;}
system("cp /etc/fstab /etc/fstab.bak");
open ($filehandle, ">", '/etc/fstab') or die "Could not open /etc/fstab
for writing\n";
for my $line (@newfstab) {
 print $filehandle $line;}
close($filehandle);
@out=`mount '$oldfs'`;
if ($? == 0) {
 print "Mounted updated $oldfs\n";
 exit 0;}
else{
 print "Unable to mount updated $oldfs\n";
 exit 1;}
exit 0;
```

作为此脚本用法的示例、假设中的数据 `/oradata` 将迁移到 `/neworadata` 和 `/logs` 将迁移到 `/newlogs`。执行此任务的最简单方法之一是、使用简单的文件复制操作将新设备重新定位回原始装载点。

2. 假设中存在新旧文件系统 `/etc/fstab` 文件、如下所示：

```
cluster01:/vol_oradata /oradata nfs rw,bg,vers=3,rsize=65536,wsize=65536
0 0
cluster01:/vol_logs /logs nfs rw,bg,vers=3,rsize=65536,wsize=65536 0 0
cluster01:/vol_neworadata /neworadata nfs
rw,bg,vers=3,rsize=65536,wsize=65536 0 0
cluster01:/vol_newlogs /newlogs nfs rw,bg,vers=3,rsize=65536,wsize=65536
0 0
```

3. 运行时、此脚本会卸载当前文件系统并将其替换为新的：

```
[root@jfsc3 scripts]# ./swap.fs.pl /oradata,/neworadata
/neworadata unmounted
/oradata unmounted
Mounted updated /oradata
[root@jfsc3 scripts]# ./swap.fs.pl /logs,/newlogs
/newlogs unmounted
/logs unmounted
Mounted updated /logs
```

4. 该脚本还会更新 `/etc/fstab` 相应地归档。在此处所示的示例中、它包括以下更改：

```
#Removed by swap script cluster01:/vol_oradata /oradata nfs
rw,bg,vers=3,rsize=65536,wsize=65536 0 0
#Removed by swap script cluster01:/vol_neworadata /neworadata nfs
rw,bg,vers=3,rsize=65536,wsize=65536 0 0
cluster01:/vol_neworadata /oradata nfs
rw,bg,vers=3,rsize=65536,wsize=65536 0 0
#Removed by swap script cluster01:/vol_logs /logs nfs
rw,bg,vers=3,rsize=65536,wsize=65536 0 0
#Removed by swap script cluster01:/vol_newlogs /newlogs nfs
rw,bg,vers=3,rsize=65536,wsize=65536 0 0
cluster01:/vol_newlogs /logs nfs rw,bg,vers=3,rsize=65536,wsize=65536 0
0
```

自动化数据库迁移

此示例说明了如何使用关闭、启动和文件系统替换脚本来完全自动执行迁移。

```
#! /usr/bin/perl
use strict;
#use warnings;
my $oraclesid=$ARGV[0];
```

```
my @oldfs;
my @newfs;
my $x=1;
while ($x < scalar(@ARGV)) {
   ($oldfs[$x-1], $newfs[$x-1]) = split (',',$ARGV[$x]);
   $x+=1;}
my @out=`./dbshut.pl '$oraclesid'`;
print @out;
if ($? ne 0) {
   print "Failed to shut down database\n";
   exit 0;}
$x=0;
while ($x < scalar(@oldfs)) {
   my @out=`./mk.fs.readonly.pl '$oldfs[$x]'`;
   if ($? ne 0) {
      print "Failed to make filesystem $oldfs[$x] readonly\n";
      exit 0;}
   $x+=1;}
$x=0;
while ($x < scalar(@oldfs)) {
   my @out=`rsync -rlpogt --stats --progress --exclude='.snapshot'
'$oldfs[$x]/' '/$newfs[$x]/'`;
   print @out;
   if ($? ne 0) {
      print "Failed to copy filesystem $oldfs[$x] to $newfs[$x]\n";
      exit 0;}
   else {
      print "Succesfully replicated filesystem $oldfs[$x] to
$newfs[$x]\n";}
   $x+=1;}
$x=0;
while ($x < scalar(@oldfs)) {
   print "swap $x $oldfs[$x] $newfs[$x]\n";
   my @out=`./swap.fs.pl '$oldfs[$x],$newfs[$x]'`;
   print @out;
   if ($? ne 0) {
      print "Failed to swap filesystem $oldfs[$x] for $newfs[$x]\n";
      exit 1;}
   else {
      print "Swapped filesystem $oldfs[$x] for $newfs[$x]\n";}
   $x+=1;}
my @out=`./dbstart.pl '$oraclesid'`;
print @out;
```

显示文件位置

此脚本会收集大量关键数据库参数、并以易于阅读的格式打印这些参数。此脚本在查看数据布局时非常有用。此外、还可以修改此脚本以供其他用途。

```
#! /usr/bin/perl
#use strict;
#use warnings;
my $oraclesid=$ARGV[0];
my $oracleuser='oracle';
my @out;
sub dosql{
        my $command = @_[0];
        my @lines;
        my $uid=$<;
        if ($uid == 0) {
        @lines=`su - $oracleuser -c "export ORAENV_ASK=NO;export
ORACLE_SID=$oraclesid;. oraenv -s << EOF1
EOF1
sqlplus -S / as sysdba << EOF2
set heading off
$command
EOF2
"
        `;}
        else {
        $command=~s/\\\\\\/\\/g;
        @lines=`export ORAENV_ASK=NO;export ORACLE_SID=$oraclesid;. oraenv
-s << EOF1
EOF1
sqlplus -S / as sysdba << EOF2
set heading off
$command
EOF2
        `;};
return @lines}
print "\n";
@out=dosql('select name from v\\\\\$datafile;');
print "$oraclesid datafiles:\n";
for $line (@out) {
        chomp($line);
        if (length($line)>0) {print "$line\n";}}
print "\n";
@out=dosql('select member from v\\\\\$logfile;');
print "$oraclesid redo logs:\n";
for $line (@out) {
```

```
        chomp($line);
        if (length($line)>0) {print "$line\n";}}
print "\n";
@out=dosql('select name from v\\\\\$tempfile;');
print "$oraclesid temp datafiles:\n";
for $line (@out) {
        chomp($line);
        if (length($line)>0) {print "$line\n";}}
print "\n";
@out=dosql('show parameter spfile;');
print "$oraclesid spfile\n";
for $line (@out) {
        chomp($line);
        if (length($line)>0) {print "$line\n";}}
print "\n";
@out=dosql('select name||\' \'||value from v\\\\\$parameter where
isdefault=\'FALSE\';');
print "$oraclesid key parameters\n";
for $line (@out) {
        chomp($line);
        if ($line =~ /control_files/) {print "$line\n";}
        if ($line =~ /db_create/) {print "$line\n";}
        if ($line =~ /db_file_name_convert/) {print "$line\n";}
        if ($line =~ /log_archive_dest/) {print "$line\n";}}
        if ($line =~ /log_file_name_convert/) {print "$line\n";}
        if ($line =~ /pdb_file_name_convert/) {print "$line\n";}
        if ($line =~ /spfile/) {print "$line\n";}
print "\n";
```

ASM迁移清理

```
#! /usr/bin/perl
#use strict;
#use warnings;
my $oraclesid=$ARGV[0];
my $oracleuser='oracle';
my @out;
sub dosql{
        my $command = @_[0];
        my @lines;
        my $uid=$<;
        if ($uid == 0) {
        @lines=`su - $oracleuser -c "export ORAENV_ASK=NO;export
ORACLE_SID=$oraclesid;. oraenv -s << EOF1
EOF1
```

```
sqlplus -S / as sysdba << EOF2
set heading off
$command
EOF2
"
        `;}
        else {
        $command=~s/\\\\\\/\\/g;
        @lines=`export ORAENV_ASK=NO;export ORACLE_SID=$oraclesid;. oraenv
-s << EOF1
EOF1
sqlplus -S / as sysdba << EOF2
set heading off
$command
EOF2
        `;}
return @lines}
print "\n";
@out=dosql('select name from v\\\\\$datafile;');
print @out;
print "shutdown immediate;\n";
print "startup mount;\n";
print "\n";
for $line (@out) {
        if (length($line) > 1) {
                chomp($line);
                ($first, $second,$third,$fourth)=split('_',$line);
                $fourth =~ s/^TS-//;
                $newname=lc("$fourth.dbf");
                $path2file=$line;
                $path2file=~ /(^.*.\/)/;
                print "host mv $line $1$newname\n";}}
print "\n";
for $line (@out) {
        if (length($line) > 1) {
                chomp($line);
                ($first, $second,$third,$fourth)=split('_',$line);
                $fourth =~ s/^TS-//;
                $newname=lc("$fourth.dbf");
                $path2file=$line;
                $path2file=~ /(^.*.\/)/;
                print "alter database rename file '$line' to
'$1$newname';\n";}}
print "alter database open;\n";
print "\n";
```

ASM到文件系统名称转换

```
set serveroutput on;
set wrap off;
declare
    cursor df is select file#, name from v$datafile;
    cursor tf is select file#, name from v$tempfile;
    cursor lf is select member from v$logfile;
    firstline boolean := true;
begin
    dbms_output.put_line(CHR(13));
    dbms_output.put_line('Parameters for log file conversion:');
    dbms_output.put_line(CHR(13));
    dbms_output.put('*.log_file_name_convert = ');
    for lfrec in lf loop
        if (firstline = true) then
            dbms_output.put('''' || lfrec.member || ''', ');
            dbms_output.put('''/NEW_PATH/' ||
regexp_replace(lfrec.member,'^.*./','') || '''');
        else
            dbms_output.put(',''' || lfrec.member || ''', ');
            dbms_output.put('''/NEW_PATH/' ||
regexp_replace(lfrec.member,'^.*./','') || '''');
        end if;
        firstline:=false;
    end loop;
    dbms_output.put_line(CHR(13));
    dbms_output.put_line(CHR(13));
    dbms_output.put_line('rman duplication script:');
    dbms_output.put_line(CHR(13));
    dbms_output.put_line('run');
    dbms_output.put_line('{');
    for dfrec in df loop
        dbms_output.put_line('set newname for datafile ' ||
            dfrec.file# || ' to ''' || dfrec.name ||''';');
    end loop;
    for tfrec in tf loop
        dbms_output.put_line('set newname for tempfile ' ||
            tfrec.file# || ' to ''' || tfrec.name ||''';');
    end loop;
    dbms_output.put_line('duplicate target database for standby backup
location INSERT_PATH_HERE;');
    dbms_output.put_line('}');
end;
/
```

重放数据库上的日志

此脚本接受处于挂载模式的数据库的Oracle SID的一个参数、并尝试重放所有当前可用的归档日志。

```
#! /usr/bin/perl
use strict;
my $oraclesid=$ARGV[0];
my $oracleuser='oracle';
84 Migration of Oracle Databases to NetApp Storage Systems © 2021 NetApp,
Inc. All rights reserved
my $uid = $<;
my @out;
if ($uid == 0) {
@out=`su - $oracleuser -c '. oraenv << EOF1
$oraclesid
EOF1
sqlplus / as sysdba << EOF2
recover database until cancel;
auto
EOF2
'
`;}
else {
@out=`. oraenv << EOF1
$oraclesid
EOF1
sqlplus / as sysdba << EOF2
recover database until cancel;
auto
EOF2
`;
}
print @out;
```

重放备用数据库上的日志

此脚本与前面的脚本相同、只是它是为备用数据库设计的。

```
#! /usr/bin/perl
use strict;
my $oraclesid=$ARGV[0];
my $oracleuser='oracle';
my $uid = $<;
my @out;
if ($uid == 0) {
@out=`su - $oracleuser -c '. oraenv << EOF1
$oraclesid
EOF1
sqlplus / as sysdba << EOF2
recover standby database until cancel;
auto
EOF2
'
`;}
else {
@out=`. oraenv << EOF1
$oraclesid
EOF1
sqlplus / as sysdba << EOF2
recover standby database until cancel;
auto
EOF2
`;
}
print @out;
```

附加说明

Oracle数据库性能优化和基准测试过程

准确测试数据库存储性能是一个极其复杂的主题。它需要了解以下问题：

- IOPS和吞吐量
- 前台和后台I/O操作之间的区别
- 延迟对数据库的影响
- 许多操作系统和网络设置也会影响存储性能

此外、还需要考虑执行非存储数据库任务。有时、优化存储性能不会带来任何有用的优势、因为存储性能不再是性能的限制因素。

现在、大多数数据库客户都选择全闪存阵列、这就需要考虑一些额外的注意事项。例如、考虑在双节点AFF A900系统上进行性能测试：

- 如果读/写比率为80/20、则两个A900节点可以在延迟甚至超过150μs微秒之前提供超过100万次的随机数据库IOPS。这远远超出了大多数数据库当前的性能需求、因此很难预测预期的改进。存储将作为瓶颈在很大程度上被消除。
- 网络带宽日益成为性能限制的常见来源。例如、旋转磁盘解决方案通常会成为数据库性能的瓶颈、因为I/O延迟非常高。当全闪存阵列消除延迟限制后、障碍往往会转移到网络上。在虚拟化环境和刀片式系统中、这一点尤为明显、因为它们的真正网络连接很难直观地呈现出来。如果由于带宽限制而无法充分利用存储系统本身、则可能会使性能测试复杂化。
- 由于全闪存阵列的延迟显著缩短、因此通常无法将全闪存阵列与包含旋转磁盘的阵列进行性能比较。测试结果通常没有意义。
- 将峰值IOPS性能与纯闪存阵列进行比较通常不是一项有用的测试、因为数据库不受存储I/O的限制例如、假设一个阵列可以承受50万次随机IOPS、而另一个阵列可以承受30万次随机IOPS。如果数据库将99%的时间花在CPU处理上、则这种差异在实际环境中无关紧要。这些工作负载从不充分利用存储阵列的全部功能。相反、在整合平台中、峰值IOPS功能可能至关重要、在该平台中、存储阵列应加载到其峰值功能。
- 在任何存储测试中、始终考虑延迟和IOPS。市场上的许多存储阵列都声称IOPS达到了极高水平、但延迟会使这些IOPS在这种水平下毫无用处。全闪存阵列的典型目标为1毫秒标记。更好的测试方法不是测量可能的最大IOPS、而是确定在平均延迟超过1毫秒之前存储阵列可以承受的IOPS数。

Oracle自动工作负载存储库和基准测试

Oracle性能比较的黄金标准是Oracle自动工作负载存储库(Automatic Workload Repository、AWR)报告。

AWR报告有多种类型。从存储角度来看、是指通过运行生成的报告 `awrrpt.sql` 命令功能最全面、最有价值、因为它针对特定数据库实例、并包含一些详细的直方图、这些直方图可按延迟细分存储I/O事件。

比较两个性能阵列时、理想情况下需要在每个阵列上运行相同的工作负载、并生成一个准确针对该工作负载的AWR报告。如果工作负载运行时间非常长、则可以使用一个AWR报告、其中经过的时间包含开始和停止时间、但最好将AWR数据细分为多个报告。例如、如果批处理作业从午夜运行到早上6点、请创建一系列从午夜到凌晨1点、从凌晨1点到凌晨2点的一小时AWR报告、依此类推。

在其他情况下、应优化非常短的查询。最佳选择是基于查询开始时创建的AWR快照和查询结束时创建的第二个AWR快照创建AWR报告。否则、数据库服务器应保持安静、以最大限度地减少后台活动、因为后台活动会掩盖正在分析的查询的活动。

> 如果AWR报告不可用、则Oracle statspack报告是一个很好的替代方案。它们包含与AWR报告大部分相同的I/O统计信息。

Oracle AWR和故障排除

AWR报告也是分析性能问题的最重要工具。

与基准测试一样、性能故障排除要求您精确测量特定工作负载。如果可能、请在向NetApp支持中心报告性能问题或与NetApp或合作伙伴客户团队合作购买新的解决方案时提供AWR数据。

提供AWR数据时、请考虑以下要求：

- 运行 `awrrpt.sql` 命令以生成报告。输出可以是文本或HTML。
- 如果使用Oracle Real Application Clusters (RAC)、请为集群中的每个实例生成AWR报告。
- 确定问题存在的具体时间。AWR报告的最长可接受用时通常为一小时。如果问题持续数小时或涉及多小时操作(例如批处理作业)、请提供多个涵盖要分析的整个期间的一小时AWR报告。

- 如果可能、将AWR快照间隔调整为15分钟。此设置允许执行更详细的分析。这还需要执行更多的 `awrrpt.sql` 以提供每15分钟间隔的报告。
- 如果问题是运行时间非常短的查询、请根据操作开始时创建的AWR快照和操作结束时创建的第二个AWR快照提供AWR报告。否则、数据库服务器应保持安静、以最大限度地减少后台活动、因为后台活动会掩盖所分析操作的活动。
- 如果在特定时间报告了性能问题、但在其他时间未报告、请提供其他证明性能良好的AWR数据以供比较。

CALIBRAT_IO

。 `calibrate_io` 切勿使用命令测试、比较存储系统或对其进行基准测试。如Oracle文档中所述、此操作步骤会校准存储的I/O功能。

校准与基准测试不同。此命令的目的是通过问题描述I/O来帮助校准数据库操作、并通过优化向主机发出的I/O级别来提高其效率。因为执行的I/O类型 `calibrate_io` 操作不代表实际的数据库用户I/O、结果不可预测、而且经常甚至无法重现。

SLOB2

SOLB2 (Song Little Oracle基准)已成为评估数据库性能的首选工具。它由Kevin Clsson开发、可从获取 "https://kevinclosson.net/slob/"。安装和配置只需几分钟、它会使用实际的Oracle数据库在用户可定义的表空间上生成I/O模式。它是少数几个可以使全闪存阵列的I/O饱和的测试选项之一此外、它还有助于生成低得多的I/O级别、以模拟IOPS低但对延迟敏感的存储工作负载。

Swingbench

Swingbench可用于测试数据库性能、但要以对存储造成压力的方式使用Swingbench、则极为困难。NetApp尚未从Swingbench中检测到任何测试产生足够的I/O来为任何AFF阵列带来大量负载。在有限情况下、可以使用订单输入测试(Order Entry Test、OOT)从延迟角度评估存储。如果数据库对特定查询具有已知的延迟依赖关系、则此功能可能会很有用。必须注意确保主机和网络配置正确、以实现全闪存阵列的潜在延迟。

HammerDB

HAMmerDB是一款数据库测试工具、用于模拟TPC-C和TPC-H基准测试等。构建一个足够大的数据集可能需要花费大量时间才能正确执行测试、但它可以作为一个有效的工具来评估OLTP和数据仓库应用程序的性能。

猎户座

Oracle ORION工具通常与Oracle 9一起使用、但尚未对其进行维护、以确保与各种主机操作系统中的更改兼容。由于与操作系统和存储配置不兼容、因此很少与Oracle 10或Oracle 11结合使用。

Oracle重新编写了该工具、默认情况下会随Oracle 12c一起安装。虽然此产品已得到改进、并使用了与实际Oracle数据库相同的许多调用、但它使用的代码路径或I/O行为与Oracle不同。例如、大多数Oracle I/O都是同步执行的、这意味着数据库会暂停、直到I/O完成、因为I/O操作在前台完成。简单地将随机I/O充斥存储系统并不是真正的Oracle I/O、也不提供比较存储阵列或衡量配置更改影响的直接方法。

尽管如此、也有一些适用于ORION的用例、例如、对特定主机-网络-存储配置的最大可能性能进行常规测量、或者对存储系统的运行状况进行评估。通过仔细测试、可以设计出可用的ORION测试来比较存储阵列或评估配置更改的影响、前提是这些参数包括考虑IOPS、吞吐量和延迟、并尝试忠实地复制真实的工作负载。

过时的NFSv3锁定和Oracle数据库

如果Oracle数据库服务器崩溃、则在重新启动时、陈旧的NFS锁定可能会出现问题。通过仔细注意服务器上的名称解析配置、可以避免此问题。

出现此问题的原因是、创建锁定和清除锁定使用两种略有不同的名称解析方法。其中涉及两个进程、即网络锁定管理器(Network Lock Manager、NLM)和NFS客户端。NLM使用 `uname n` 要确定主机名、请使用 `rpc.statd` 流程使用 `gethostbyname()`。这些主机名必须匹配、操作系统才能正确清除陈旧锁定。例如、主机可能正在查找属于的锁定 `dbserver5`，但主机已将锁定注册为 `dbserver5.mydomain.org`。条件 `gethostbyname()` 返回的值与不相同 `uname -a`，则锁定释放过程未成功。

以下示例脚本将验证名称解析是否完全一致：

```
#! /usr/bin/perl
$uname=`uname -n`;
chomp($uname);
($name, $aliases, $addrtype, $length, @addrs) = gethostbyname $uname;
print "uname -n yields: $uname\n";
print "gethostbyname yields: $name\n";
```

条件 `gethostbyname` 不匹配 `uname`，可能是陈旧的锁定。例如、此结果揭示了一个潜在问题：

```
uname -n yields: dbserver5
gethostbyname yields: dbserver5.mydomain.org
```

通常、可以通过更改主机在中的显示顺序来查找解决方案 `/etc/hosts`。例如、假设主机文件包含以下条目：

```
10.156.110.201  dbserver5.mydomain.org dbserver5 loghost
```

要解析此问题描述、请更改完全限定域名和短主机名的显示顺序：

```
10.156.110.201  dbserver5 dbserver5.mydomain.org loghost
```

`gethostbyname()` 现在返回短 `dbserver5` 主机名、与的输出匹配 `uname`。因此、锁定会在服务器崩溃后自动清除。

Oracle数据库的WAFL对齐验证

正确对齐WAFL对于获得良好性能至关重要。尽管ONTAP以4 KB单位管理块、但这并不意味着ONTAP以4 KB单位执行所有操作。事实上、ONTAP支持不同大小的块操作、但底层记帐由WAFL以4 KB单位进行管理。

术语"对齐"是指Oracle I/O与这些4 KB单位的对应关系。要获得最佳性能、需要将一个Oracle 8 KB块驻留在驱动

器上的两个4 KB WAFL物理块上。如果块偏移2 KB、则此块位于一个4 KB块的一半、一个单独的完整4 KB块、然后是第三个4 KB块的一半。这种排列会导致性能下降。

对齐不是NAS文件系统的问题。Oracle数据文件会根据Oracle块的大小与文件开头对齐。因此、8 KB、16 KB和32 KB的块大小始终对齐。所有块操作都会与文件开头偏移、以4 KB为单位。

与此相反、LUN通常在开始时包含某种类型的驱动程序标头或文件系统元数据、以创建偏移。在现代操作系统中、对齐很少会成为问题、因为这些操作系统专为可能使用本机4 KB扇区的物理驱动器而设计、这也需要将I/O与4 KB边界对齐以获得最佳性能。

但也有一些例外情况。数据库可能是从未针对4 KB I/O进行优化的旧版操作系统迁移的、或者分区创建期间的用户错误可能导致偏移量大小不以4 KB为单位。

以下示例是Linux专用的、但操作步骤可适用于任何操作系统。

已对齐

以下示例显示了对具有单个分区的单个LUN的对齐检查。

首先、创建使用驱动器上所有可用分区的分区。

```
[root@host0 iscsi]# fdisk /dev/sdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF
disklabel
Building a new DOS disklabel with disk identifier 0xb97f94c1.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.
The device presents a logical sector size that is smaller than
the physical sector size. Aligning to a physical sector (or optimal
I/O) size boundary is recommended, or performance may be impacted.
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-10240, default 1):
Using default value 1
Last cylinder, +cylinders or +size{K,M,G} (1-10240, default 10240):
Using default value 10240
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
[root@host0 iscsi]#
```

可以使用以下命令以数学方式检查对齐情况：

```
[root@host0 iscsi]# fdisk -u -l /dev/sdb
Disk /dev/sdb: 10.7 GB, 10737418240 bytes
64 heads, 32 sectors/track, 10240 cylinders, total 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 65536 bytes
Disk identifier: 0xb97f94c1
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1              32    20971519    10485744   83  Linux
```

输出显示单位为512字节、分区起始单位为32。这是总共32 x 512 = 16、834字节、是4 KB WAFL块的整数倍。此分区已正确对齐。

要验证是否正确对齐、请完成以下步骤：

1. 确定LUN的通用唯一标识符(UUID)。

```
FAS8040SAP::> lun show -v /vol/jfs_luns/lun0
              Vserver Name: jfs
                  LUN UUID: ed95d953-1560-4f74-9006-85b352f58fcd
                    Mapped: mapped`                 `
```

2. 在ONTAP控制器上输入节点Shell。

```
FAS8040SAP::> node run -node FAS8040SAP-02
Type 'exit' or 'Ctrl-D' to return to the CLI
FAS8040SAP-02> set advanced
set not found.  Type '?' for a list of commands
FAS8040SAP-02> priv set advanced
Warning: These advanced commands are potentially dangerous; use
         them only when directed to do so by NetApp
         personnel.
```

3. 对第一步中确定的目标UUID启动统计收集。

```
FAS8040SAP-02*> stats start lun:ed95d953-1560-4f74-9006-85b352f58fcd
Stats identifier name is 'Ind0xffffff08b9536188'
FAS8040SAP-02*>
```

4. 执行一些I/O使用非常重要 `iflag` 用于确保I/O是同步的且不缓冲的参数。

> 使用此命令时请格外小心。反转 `if` 和 `of` 参数会销毁数据。

```
[root@host0 iscsi]# dd if=/dev/sdb1 of=/dev/null iflag=dsync count=1000
bs=4096
1000+0 records in
1000+0 records out
4096000 bytes (4.1 MB) copied, 0.0186706 s, 219 MB/s
```

5. 停止统计信息并查看对齐直方图。所有I/O都应位于中 `.0` 存储分段、表示I/O与4 KB块边界对齐。

```
FAS8040SAP-02*> stats stop
StatisticsID: Ind0xffffff08b9536188
lun:ed95d953-1560-4f74-9006-85b352f58fcd:instance_uuid:ed95d953-1560-
4f74-9006-85b352f58fcd
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.0:186%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.1:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.2:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.3:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.4:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.5:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.6:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.7:0%
```

未对齐

以下示例显示了未对齐的I/O：

1. 创建不与4 KB边界对齐的分区。这不是现代操作系统上的默认行为。

```
[root@host0 iscsi]# fdisk -u /dev/sdb
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First sector (32-20971519, default 32): 33
Last sector, +sectors or +size{K,M,G} (33-20971519, default 20971519):
Using default value 20971519
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

2. 创建分区时使用的是33扇区偏移、而不是默认的32扇区偏移。重复中所述的操作步骤 "已对齐"。直方图显示

如下：

```
FAS8040SAP-02*> stats stop
StatisticsID: Ind0xffffff0468242e78
lun:ed95d953-1560-4f74-9006-85b352f58fcd:instance_uuid:ed95d953-1560-
4f74-9006-85b352f58fcd
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.0:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.1:136%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.2:4%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.3:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.4:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.5:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.6:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.7:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_partial_blocks:31%
```

未对齐情况很明显。I/O大部分落在*中 `*.1` 存储分段、与预期偏移匹配。创建分区时、该分区会比优化默认值更远地移动到设备中512字节、这意味着直方图偏移512字节。

此外、还可以使用 `read_partial_blocks` 统计信息不为零、这意味着执行的I/O未填满整个4 KB块。

重做日志记录

此处介绍的过程适用于数据文件。Oracle重做日志和归档日志具有不同的I/O模式。例如、重做日志记录是对单个文件的循环覆盖。如果使用默认的512字节块大小、则写入统计信息如下所示：

```
FAS8040SAP-02*> stats stop
StatisticsID: Ind0xffffff0468242e78
lun:ed95d953-1560-4f74-9006-85b352f58fcd:instance_uuid:ed95d953-1560-4f74-
9006-85b352f58fcd
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.0:12%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.1:8%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.2:4%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.3:10%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.4:13%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.5:6%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.6:8%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.7:10%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_partial_blocks:85%
```

I/O将分布在所有直方图分段中、但这不是性能问题。但是、使用4 KB块大小可能会有利于极高的重做日志记录速率。在这种情况下、需要确保重做日志记录LUN正确对齐。但是、这对于获得良好性能并不像数据文件对齐那样重要。