Oracle数据库迁移

Enterprise applications

NetApp
May 09, 2024

This PDF was generated from https://docs.netapp.com/zh-cn/ontap-apps-dbs/oracle/oracle-migration-overview.html on May 09, 2024. Always check docs.netapp.com for the latest.

目录

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。

[错误：缺少图形映像]

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
```

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
```

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/3600a098038303044542448755657426a  10.00g sanvg
  /dev/mapper/3600a098038303044542448755657426b  10.00g sanvg
  /dev/mapper/3600a098038303044542448755657426c  10.00g sanvg
  /dev/mapper/3600a098038303044542448755657426d  10.00g sanvg
  /dev/mapper/3600a098038303044542448755657426e  10.00g sanvg
  /dev/mapper/3600a098038303044542448755657426f  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;
```

版权信息

版权所有 © 2024 NetApp, Inc.。保留所有权利。中国印刷。未经版权所有者事先书面许可，本文档中受版权保护的任何部分不得以任何形式或通过任何手段（图片、电子或机械方式，包括影印、录音、录像或存储在电子检索系统中）进行复制。

从受版权保护的 NetApp 资料派生的软件受以下许可和免责声明的约束：

本软件由 NetApp 按“原样”提供，不含任何明示或暗示担保，包括但不限于适销性以及针对特定用途的适用性的隐含担保，特此声明不承担任何责任。在任何情况下，对于因使用本软件而以任何方式造成的任何直接性、间接性、偶然性、特殊性、惩罚性或后果性损失（包括但不限于购买替代商品或服务；使用、数据或利润方面的损失；或者业务中断），无论原因如何以及基于何种责任理论，无论出于合同、严格责任或侵权行为（包括疏忽或其他行为），NetApp 均不承担责任，即使已被告知存在上述损失的可能性。

NetApp 保留在不另行通知的情况下随时对本文档所述的任何产品进行更改的权利。除非 NetApp 以书面形式明确同意，否则 NetApp 不承担因使用本文档所述产品而产生的任何责任或义务。使用或购买本产品不表示获得 NetApp 的任何专利权、商标权或任何其他知识产权许可。

本手册中描述的产品可能受一项或多项美国专利、外国专利或正在申请的专利的保护。

有限权利说明：政府使用、复制或公开本文档受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中“技术数据权利 — 非商用”条款第 (b)(3) 条规定的限制条件的约束。

本文档中所含数据与商业产品和/或商业服务（定义见 FAR 2.101）相关，属于 NetApp, Inc. 的专有信息。根据本协议提供的所有 NetApp 技术数据和计算机软件具有商业性质，并完全由私人出资开发。 美国政府对这些数据的使用权具有非排他性、全球性、受限且不可撤销的许可，该许可既不可转让，也不可再许可，但仅限在与交付数据所依据的美国政府合同有关且受合同支持的情况下使用。除本文档规定的情形外，未经 NetApp, Inc. 事先书面批准，不得使用、披露、复制、修改、操作或显示这些数据。美国政府对国防部的授权仅限于 DFARS 的第 252.227-7015(b)（2014 年 2 月）条款中明确的权利。

商标信息

NetApp、NetApp 标识和 http://www.netapp.com/TM 上所列的商标是 NetApp, Inc. 的商标。其他公司和产品名称可能是其各自所有者的商标。