

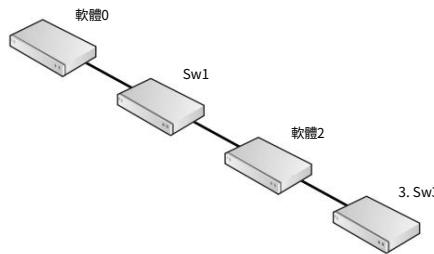


尽管 FCoE 核心边缘设计类似于以太网星型网络拓扑,但 IFGP 的性质及其对 IFSFP 路由协议的使用允许多个核心交换机及其之间的所有链路同时处于活动状态。相比之下,以太网需要生成树协议 (STP) 或热备用路由器协议 (HSRP) 来使某些路径成为备用路径,以避免出现环路。

## 级联拓扑

级联设计将交换机连接成一行,如图 5.18 所示。

图 5.18 级联拓扑



级联拓扑结构是令人厌恶的,99.999% 的时间都不应该部署,因为它不可扩展、性能差、可靠性差。如果图 5.18 中的交换机 2 (Sw2) 发生故障,交换机 3 (Sw3) 将与架构的其余部分隔离。

## 环形拓扑

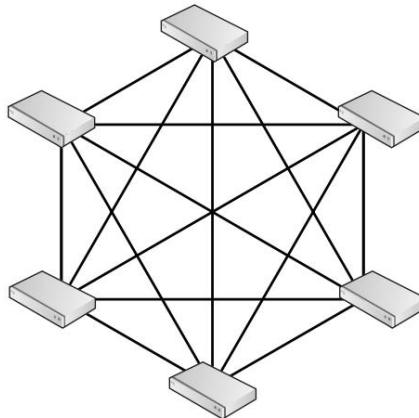
环形拓扑与级联拓扑相同,只是两个终端交换机是相连的,这使得它比级联拓扑稍微不那么丑陋。通过连接两个终端交换机,您可以通过在环上向相反方向发送帧来提供到达目的地的替代路径。显然,如果环形设计中的交换机发生故障,拓扑就会变成级联,但由于有替代路径,设备不会孤立。但是,在环形拓扑中添加交换机需要断开环。

与级联一样,它不是一种流行的设计,并且在现代存储领域几乎没有任何实际用例。

## 网状拓扑

网状拓扑结构将 SAN 中的每个交换机连接到其他每个交换机。这确保每个交换机与任何其他交换机仅相距一跳,并且具有高度的弹性和性能。但是,从 ISL 的角度来看,它的成本极高。基本上,许多交换机端口都被用作 ISL。图 5.19 显示了一个简单的网状拓扑。

图 5.19 网状拓扑



随着网状拓扑的扩展,其端口浪费越来越严重(因为它们需要用作 ISL)。

### 分层拓扑

分层拓扑是本地化的敌人。在分层结构拓扑中,您将所有类似的设备分组到同一个交换机上。例如,所有主机将位于一组交换机上,所有存储端口位于一组交换机上,所有磁带设备位于一组交换机上,所有复制端口位于一组交换机上。这种设计保证了访问存储或磁带设备的主机必须跨越至少一条 ISL。不过,只要您有足够的 ISL 带宽,这并不是世界末日。

与极端本地化相比,分层结构设计的维护和扩展非常简单。在分层设计中,通过简单地添加 ISL 来增加层间带宽相对简单。通过添加更多交换机或更多交换机端口来扩展层也很容易。

在现实世界中,分层结构拓扑相当常见,特别是如果你考虑将所有存储和磁带设备放置在核心交换机上,将所有主机放置在混合分层核心边缘拓扑中的边缘交换机上。

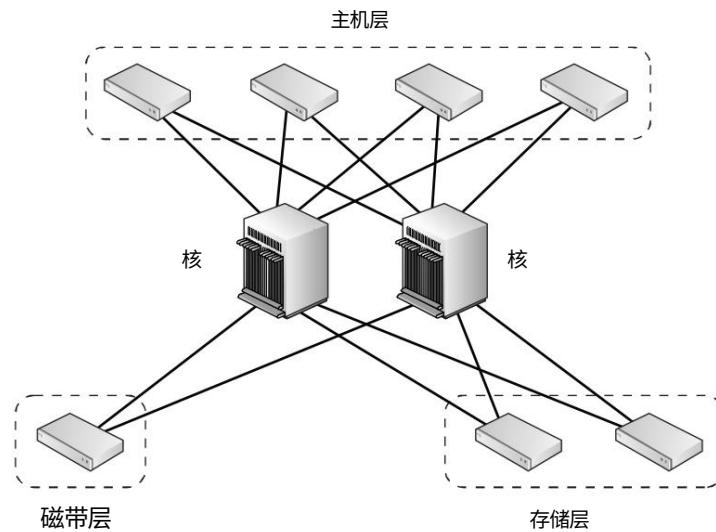
图 5.20 显示了一种采用分层形式的简单核心边缘织物设计。

尽管关于拓扑的讨论很多,但设计一个结构时并不要求必须围绕任何特定的拓扑。您可以根据自己的需求混合搭配和调整设计。

### 光纤通道路由

现在让我们看看帧传送和路由在 FC 结构中的工作方式,以及一些供应商对虚拟结构和虚拟 SAN 的实现。

图 5.20 分层结构拓扑



### 虚拟结构和 VSAN

两大 FC 交换机供应商 Brocade 和 Cisco 均拥有可将物理交换机划分到端口级别的技术,从而允许单个物理交换机中的各个端口属于离散结构。

思科发明了这项技术,并将其称为虚拟 SAN (VSAN)。ANSI T11 委员会非常喜欢它,因此将其作为虚拟结构的行业标准。

随后,Brocade 以 ANSI T11 标准为基础实施了虚拟架构。因此,思科的 VSAN 和 Brocade 的虚拟架构技术极其相似也就不足为奇了。

#### 思科VSAN

每个 VSAN 都是一个功能齐全的结构,具有自己的专用端口、设备和结构服务。这些结构服务包括常见的名称服务器、分区表、别名表等。

Cisco MDS 和 Nexus FCoE 交换机上的所有 F\_Port 都可以是一个且唯一的。一是 VSAN。但是,中继模式下的 ISL (包括 PortChannel)可以承载多个 VSAN 的流量,同时保持不同 VSAN 流量的隔离。它们通过添加到 FC 帧头以识别 VSAN 成员身份的 VSAN 标签来实现这一点。

VSAN 的一个良好用例可能包括将单个交换机上的端口划分为开发结构和暂存结构。这将允许您拥有一个运行两个不同、隔离的逻辑结构的物理平台。您可以动态移动端口

根据需要在开发和暂存结构之间进行分区,从而实现良好的灵活性。如果您弄乱了开发结构中的分区,暂存结构将不受影响。

它效果很好,如果您想隔离织物,您不再需要购买专用硬件!

在隔离逻辑结构 (VSAN) 上分离端口时,可以通过称为 VSAN 间路由 (IVR) 的 Cisco 技术在单独的 VSAN 上的设备之间路由 FC 帧。IVR 的工作原理是创建区域并将其添加到 IVR 区域集。

IVR 区域集中的区域有效地与 VSAN 重叠,从而允许位于不同 VSAN 但存在于 IVR 区域集中的设备相互通信。

### 正确使用 IVR

过于热衷于使用 IVR 部署 VSAN 技术的客户有时会犯一个错误,即以分层虚拟结构拓扑的形式将所有存储阵列放在一个 VSAN 中,将所有主机放在另一个 VSAN 中,然后使用 IVR 使主机能够跨 VSAN 与存储阵列进行通信。这不是 VSAN 的最佳用例,可能会导致 IVR 成为瓶颈。

通过将存储端口和主机端口放在同一个 VSAN 中,可以轻松解决此问题。还应注意,此问题不是 VSAN 或 IVR 技术的缺陷,而是技术使用不当的一个例子。IVR 是一项很棒的技术,但它最好用于例外情况,而不是常规情况。

### 博科虚拟网络架构

Brocade 虚拟结构与 Cisco VSAN 基本相同。它们允许您通过将端口分配给新创建的逻辑交换机来划分物理交换机。然后,多个逻辑交换机形成逻辑结构,每个逻辑结构都有自己独立的结构服务 -

名称服务器、区域划分、域 ID、主交换机等。

用例与 Cisco VSAN 相同。拿一块物理锡,将其资源划分为可以单独管理的离散逻辑实体,并保持逻辑实体之间的流量隔离。同样,弄乱 Brocade 逻辑结构上的分区数据库,所有其他逻辑结构都不会受到影响。

与思科 VSAN 技术一样,可以使用 Brocade 的光纤通道路由 (FCR) 技术和逻辑存储区域网络 (LSAN) 区域在逻辑结构之间路由流量。但是,Brocade FCR 技术需要特殊的基础设施作为主干结构,以便在两个独立的结构之间进行路由。



**应谨慎使用 Cisco IVR 和 Brocade FCR。过度使用其中任何一种,都有可能给自己带来很大的麻烦。这两种技术都很棒,但请确保正确使用它们,不要滥用它们。Both technologies are**

## FC 命名

FC SAN 上的每个终端设备都有两个重要标识符：

■世界知名品牌

■节点端口ID

作为存储人员,您与全球名称和全局端口名称交互的频率将比与 N\_Port ID 交互的频率高。

### 全球通用名称

每个节点端口（主机 HBA/CNA 端口、存储阵列端口）都有自己的 64 位（8 字节）全球唯一编号，该编号会伴随设备整个生命周期。这个 64 位名称称为全球名称 (WWN)。实际上，每个节点端口都有一个全球节点名称 (WWNN) 和一个全局端口名称 (WWPN)。您通常会使用 WWPN。



**从服务器和存储阵列的意义上来说,节点没有地址。**  
从服务器和存储阵列中的端口具有地址。这意味着您无法寻址服务器或存储阵列。您只能寻址服务器或存储阵列中的端口。虽然这看起来实际上是同一件事,但这意味着如果存储阵列有 64 个端口,您无法通过指定阵列上所有 64 个端口都响应的单个地址来寻址所有 64 个端口。

全局端口名称有时称为端口全球名称或 pWWN。本书使用术语 WWPN。

Brocade FC 交换机上 switchshow 命令的以下输出显示了交换机的 WWNN 以及连接到交换机中前两个 FC 端口的终端设备的 WWPN（如上所述,您不能通过其 WWN 将流量寻址到交换机）：

```
LegendarySw01:admin>切换显示
开关名称:LegendarySw01
交换机类型:62.3
SwitchState:在线
切换模式:本机
SwitchRole:主体
SwitchDomain:44
交换机ID:          xxx ...
SwitchWwn:        10:00:00:05:AA:BB:CC:DD
分区:            开启 (Prd_Even1)
SwitchBeacon:    关闭
```

索引端口地址媒体速度状态

原始

```
=====
0   0      22万立方米       8G    在线的      FC F_端口 50:01:43:80:be:b4:08:62
1   1      220100 立方英尺     8G    在线的      FC F_端口 50:01:43:80:05:6c:22:ae
```

以下nsshow命令显示主机 HBA 的 WWNN 和 WWPN:

```
LegendarySw01:admin> nsshow
类型 Pid          操作系统        端口名称          节点名称          TTL (秒)
否      220100;3;      50:01:43:80:05:6c:22:ae;50:01:43:80:05:6c:22:af na
FC4:FCP

NodeSymb:[43] "QMH2462 FW:v5.06.03 DVR:v8.03.07.09.05.08-k"
结构端口名称:10:00:00:05:AA:BB:CC:DD
永久端口名称:50:01:43:80:05:6c:22:af
端口索引:1
分享区域:无
设备在其他 AD 中共享:否
重定向:否
部分:否
```

WWN (WWNN 和 WWPN)由制造商指定。WWN 的书写方式如下:

为 16 个十六进制数字,每两位数字用冒号分隔。WWN 只能包含十六进制字符,因此不区分大小写,这意味着 50:01:43:80:05:6c:22:af 和 50:01:43:80:05:6C:22:AF 是相同的。

WWN 刻录在 HBA/CNA 卡中,类似于 MAC 地址刻录在 NIC 中。然而,MAC 地址类比应该到此为止!与 MAC 地址不同,WWN 不用于在结构上传输帧。24 位 N\_Port ID (本章后面将介绍)用于帧路由和交换。

WWN 主要用于与安全相关的操作,例如分区和其他设备安全,这些内容稍后也会详细介绍。



**尽管 WWN 在出厂时分配给 HBA 和 CNA 卡,但通常可以通过随附的管理工具进行更改。**

**如果您正在部署基于 WWN 的区域划分,这将特别有用,因为您不必更新别名等,而是可以简单地强制任何新的替换 HBA 或 CNA 具有与其要替换的 HBA 或 CNA 相同的 WWN。**

## N\_Port ID

正如我们前面提到的,WWN 不用于 FC 网络上的帧交换和传送。这时 N\_Port ID 就派上用场了。



*Although N\_Port ID (FCID)对于交换和路由至关重要,但 SAN 管理员很少需要与它们交互。SAN administrators rarely have to interact with them.*

在探索 N\_Port ID 之前,让我们先了解一下什么是 N\_Port。N\_Port 是 FC 术语,意为节点端口,指的是 FC 网络上终端设备中的物理 FC 端口。基本上,所有存储阵列端口和主机 HBA 端口都是端点,因此被视为 N\_Port。另一方面,FC 交换机不充当端点,因此没有 N\_Ports。N\_Ports 仅适用于 FC-P2P 和 FC-SW 架构。但无论如何,这些几乎是目前唯一部署的拓扑。

有些人将 N\_Port ID 称为光纤通道 ID,简称 FCID。您可以使用任一术语。本书使用 N\_Port ID。

N\_Port ID 是一个 24 位动态分配的地址。当 N\_Port 登录到结构时,结构会分配该地址。结构中的每个端点都有一个 N\_Port ID,并且该 N\_Port ID 被编码到 FC 帧头中,用于帧交换、路由和流控制。

让我们再看一下 switchshow 输出,并突出显示连接的 N\_Port ID。  
连接设备以粗体显示:

```
LegendarySw01:admin>切换显示
```

```
开关名称:LegendarySw01
```

```
交换机类型:72.3
```

```
SwitchState:在线
```

```
切换模式:本机
```

```
SwitchRole:主体
```

```
SwitchDomain:44
```

```
交换机ID: xxx ...
```

```
SwitchWwn: 10:00:00:05:AA:BB:CC:DD
```

```
分区: 开启 (Prd_Even1)
```

```
SwitchBeacon:关闭
```

索引端口地址媒体速度状态					原始
<hr/>					
0	0	22万立方米	8G	在线的	FC F_端口 50:01:43:80:be:b4:08:62
1	1	220100 立方英尺	8G	在线的	FC F_端口 50:01:43:80:05:6c:22:ae

N\_Port ID 比 WWPN 小,且包含在每个帧头中,这使得帧头比使用 WWN 进行帧路由和端到端传送时更小。

另外,需要明确的是,连接到 SAN 的两个物理端口的设备将获得两个 N\_Port ID,每个端口一个。

N\_Port ID 的格式和结构如下：

域 地址的前 8 位称为域 ID。它可以是 1-239 之间的值，它是端口所属交换机的域 ID。

区域 这是接下来的 8 位，交换机供应商分配并构造这部分地址。N\_Port ID 此部分的值可以是 0-255 之间的任意值。

端口 地址的最后 8 位也是供应商特定的，通常转换为交换机中端口的端口号。该值可以是 0 到 255 之间的任何值，端口号超过 256 个的交换机使用区域和端口的组合来唯一标识端口。

N\_Port ID 的这种层次化特性有助于帧的路由和传送。在帧到达目标交换机之前，只有 N\_Port ID 的域部分用于路由决策。



*ONMP 被设计为持久且永不改变的原因这正是因为如果交换机获得新的域 ID,N\_Port ID 就会改变 never change is precisely because N\_Port IDs will change if a switch gets a new domain ID.*

## N\_Port ID 虚拟化

最初，物理 N\_Port 登录到结构并从结构登录服务器获取单个 N\_Port ID。一切都很简单。然而，随着服务器虚拟化技术的出现，人们越来越希望物理 N\_Port 能够注册多个 N\_Port 并获取多个 N\_Port ID，以便将 N\_Port ID 分配给虚拟机，而 SAN 分区可以将某些资源的访问权限限制在该 VM 上。见证了 N\_Port ID 虚拟化 (NPIV) 的诞生。

在 NPIV 出现之前，N\_Port 会执行 FLOGI 以从结构中获取 N\_Port ID，然后执行 PLOGI 以向名称服务器注册其 N\_Port ID。

任务完成。快进到今天，支持 NPIV 的节点向结构注册了多个虚拟 N\_Port。不过，在结构上注册额外的 N\_Port 不是通过更多 FLOGI 操作完成的。而是通过 FDISC 操作完成的。基本上，N\_Port 可以根据需要执行任意数量的 FDISC 操作，每个 FDISC 后面都跟有获取的 N\_Port ID 的 PLOGI。图 5.21 显示了具有单个物理 HBA 的单个主机，该主机已向结构注册了另外四个 N\_Port ID。

为了获取每个虚拟 N\_Port 的额外 N\_Port ID，FDISC 操作会向结构注册一个额外的 WWPN。这意味着支持 NPIV 的设备必须具有多个 WWPN——每个 N\_Port ID 一个。此外，为了使 NPIV 正常工作，结构和交换机显然必须支持 NPIV。否则，当他们看到来自连接到单个 F\_Port 的单个主机的多个 N\_Port ID 和 WWPN 时，他们会感到困惑。



*注册状态变更通知 (RSCN) 的注册是根据WWPN/N\_Port ID 进行的，每次 FLOGI 和 PLOGI 之后也会进行注册。WWPN/N\_Port ID. Registration is also performed after each FLOGI*

图 5.21 具有四个虚拟 N\_Port ID 的单个物理 N\_Port

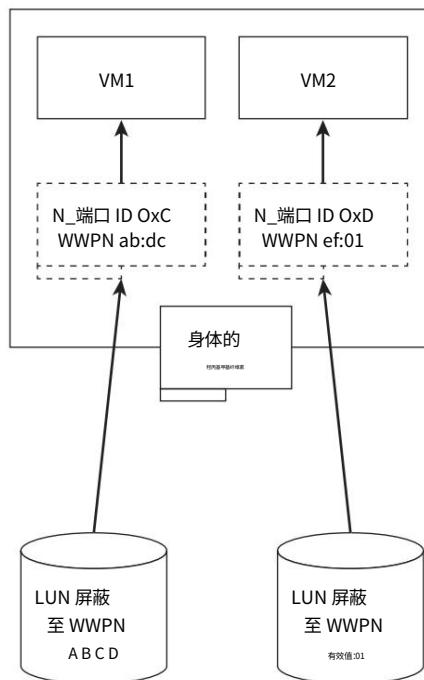


一般来说,在 HBA/CNA 方面,支持 NPIV 的 HBA 能够将自己分割成多个虚拟 HBA。每个虚拟 HBA 都有自己的 WWPN,因此可以注册自己的 N\_Port ID,就像它是一个完全边缘的物理 N\_Port。

这允许每个虚拟 HBA 在结构上直接寻址。再加上将虚拟 HBA 与 VM 关联起来的能力,您就拥有了一个可在结构上直接寻址的 VM,并且可以直接向其呈现存储。

图 5.22 显示了直接呈现给 WWPN 的两个 LUN,它们与单个物理 HBA 后面的两个虚拟机中的两个虚拟机。

图 5.22 NPIV 和直接向 VM 呈现 LUN



当然,将 LUN 直接映射到虚拟机并绕过虚拟机管理程序并不是普遍的。sally 喜欢,也不适合所有情况。但它肯定有它的位置。

## NPV 和接入网关模式

首先,NPV 与 NPIV 不同。它们看起来一样,并且基于 NPIV,但两者并不是一回事!

NPV 是 N\_Port Virtualizer 的缩写,作为一项技术,其主要目标是 NPV 旨在使来自多个供应商的交换机能够共存于同一架构中,而无需诉诸令人讨厌的互操作模式。NPV 还允许您在架构中拥有大量交换机,而不会使架构服务过载并耗尽架构中受支持的交换机的有限数量(实际上是域 ID 的有限数量)。



Cisco 发明了 NPV 而 Brocade 和 QLogic 也以真正的供应商对供应商的方式实施了同样的技术,但不得不给它取一个不同的名字。Brocade 称之为接入网关模式 (AG), QLogic 称之为智能直通。

现在我们需要了解一下 NPV 为何如此酷的背景知识。

当 FC 交换机加入结构时,它会获得自己唯一的 Domain\_ID。它还会参与在大量结构服务中,其中一些我们已经讨论过了。随着您向结构中添加越来越多的交换机,您开始面临可扩展性限制,例如在结构中最大程度地增加支持的域 ID 数量。NPV 可以提供帮助。我们稍后会解释如何做到这一点。

此外,如果您使用 Cisco、Brocade 和 QLogic 的交换机构建结构,您将不得不忍受一些古怪的互操作模式,最终得到一个功能集非常有限的结构,从第一天起,这就会成为支持难题。NPV 也在这里提供帮助。

因此,NPV 背后的秘密在于,NPV 模式下的交换机对于结构是不可见的。它不注册域 ID,也不分担协助结构服务的负担。就结构而言,交换机不是交换机;它看起来只是运行 NPIV 的 N\_Port (是的,是 NPIV,不是 NPV)。

一种常见的用例是在边缘运行 NPV 交换机,让核心交换机负责所有结构服务。这在客户向已经运行 Brocade 或 QLogic 结构的环境中添加 Cisco UCS 刀片服务器的情况下很常见。边缘到核心似乎是 NPV 的最佳选择,您不会希望在长距离的核心交换机之间使用它。

有趣的是,运行 NPV 的交换机之间的链路不是 ISL,端口也不是 E\_Port。边缘交换机上的端口有时被称为 NP\_Port,核心交换机上的端口只是普通的 F\_Port。

同样有趣的是,边缘的 NPV 开关有效地代理了所有的 FLOGI 和 PLOGI 操作。

edge 自己向核心交换机执行 FLOGI 和 PLOGI。然后,它将来自连接主机的 FLOGI 转换为 FDISC,并根据 NPIV 将每个终端设备代理注册到核心。这样,核心交换机就认为它正在通过标准 F\_Port 与 NPIV 节点通信。这种方法简单有效。

NPV 是一种基于标准的技术,它比以往的互操作模式都要好。  
结果是供应商之间提供了良好的支持。互操作模式已死;NPV 万岁!

有几件事需要注意。从边缘到核心的链路的负载平衡和 HA 可能因供应商而异。负载平衡往往是基本的循环,但未来可能会发生变化。此外,请务必咨询您的供应商是否支持 VSAN 和虚拟结构技术,尤其是在 NPV 配置中运行的边缘和核心交换机之间的连接上中继 VSAN。

要查看交换机上是否启用了 NPV/AG 模式,请发出以下命令:

```
LegendarySwitch01:管理员> ag --modeshow
已启用访问网关模式
```

Cisco MDS 结构上的以下输出显示了两个交换机配置为 VSAN 8 中的 NPV 模式:

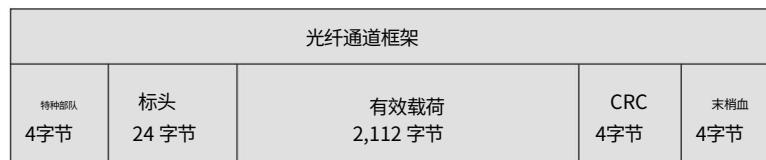
```
MDS-sw01# 显示 fcns 数据库
VSAN8:
-----
FCID 类型 PWWN (供应商)FC4-类型:功能
-----
0x010000 N 20:01:00:0d:ec:AA:BB:CC (思科) npv 0x010001 N
20:02:00:0d:ec:AA:BB:CD (思科) npv
```

## 框架

光纤通道本质上是一种基于单播帧的协议。FC SAN 中发生的大部分事情都可以看作是传统的第 2 层网络。FC 帧的长度最多可达 2,148 字节,可以包含最多 2,112 字节的有效载荷,并且必须能被 4 整除。

图 5.23 显示了一个 FC 框架。

图 5.23 FC 框架



现在让我们解释一下框架的每个组件,从左边开始向右边解释。

帧起始 所有 FC 帧都以帧起始 (SOF) 有序集作为开头,该集合充当帧分隔符的一种形式。SOF 是一个 4 字节字,指示服务类别 (COS) 以及此特定帧在集合中的哪个位置。

标头 紧跟在 SOF 后面的是 24 字节的帧标头。它包含源和目标 N\_Port ID,称为源 ID (SID) 和目标 ID (DID)。标头还定义帧是否包含用户数据 (数据帧) 或控制数据 (链路控制帧),并指示帧在交换或序列中的位置。

有效负载 紧随标头之后的是有效负载。其长度最多为 2,112 字节,如果不能被 4 整除,则必须填充。

循环冗余校验帧完整性可以通过循环冗余校验 (CRC) 实现。

帧结束 最后但并非最不重要的是帧结束 (EOF) 有序集。这通知接收节点帧已结束,并且 EOF 的规范由帧的服务类别决定。

上述内容是很好的理论,但您不太可能需要在日常存储管理中经常调用这些知识。

## 流量控制

在我们深入讨论流控制的细节之前,首先您需要了解一些背景知识。在相当高的层次上,FC SAN 上的通信由三个基本结构组成:帧、序列和交换。最小和最基本的单位是帧。多个相关帧形成一个序列,多个相关序列形成一个交换。对话经常被用作类比。在这个类比中,帧是单词,序列是句子,交换是对话。知道这一点很好,但您可能不会在日常工作中依赖这些知识。

FC SAN 中的流量控制基于信用系统。终端设备会获得信用,并且允许用其拥有的每个信用发送一帧。如果没有信用,则无法发送任何帧!这种方法对于避免拥塞和丢帧非常有效。

目前主要有两种信用体系:

■ 缓冲区到缓冲区 (B2B)

■ 端到端 (E2E)

B2B 信用适用于两个直接连接的端口,例如终端设备和交换机端口 (N\_Port 到 F\_Port)。E2E 信用适用于两个终端设备,例如发起方和目标。

## 端到端与缓冲区到缓冲区流量控制

E2E 流控制（有时称为 EE\_credit）的工作原理是目标设备向发送方返回一个 Accept (ACC),以表明它已收到帧,并且发送方可将其信用库存增加一。每次发送方发送一个帧时,它都会将其信用库存减少一,而每次收到 ACC 时,它都会将其信用库存增加一。这种基于 ACC 的补充方案适用于服务类别 1 和服务类别 2,因为它们都基于端到端连接的原理并使用确认。这些确认允许发送方维护信用库存。

交换机不参与 E2E 信用计划。



*The following “Service Classes” section will discuss different service classes. However, most FC traffic (including disk and tape I/O) uses COS 3.*

B2B 流控制（有时称为 BB\_credit）在节点端口和结构端口之间工作。由于 COS 3 使用 B2B,而 COS 3 是无连接的,没有确认,因此不使用 ACC 帧来补充信用。相反,会向发送方发回一个特殊的接收器就绪帧 (R\_RDY),以指示接收方已准备好接收更多帧。

与 E2E 类似,在 B2B 信用系统中,发送方每发送一帧就减少一信用库存,每收到一个 R\_RDY 就增加一信用库存。

E2E 和 B2B 信用系统都要求发送者拥有初始信用额度。

FLOGI 为 B2B 和 PLOGI 为 E2E 约定了此信用额度。此初始信用额度非常重要,因为如果设备在没有任何信用额度的情况下开始使用结构,它将永远无法发送帧。

既然我们谈到了这个话题,我想提一下仲裁环 (FC-AL) 环境支持一种 B2B 信用系统也没什么坏处。但是,在这种情况下,每个设备的初始信用都是零。一旦发起者仲裁了线路控制权,它就会联系目标,目标会用一定数量的 R\_RDY 来响应,这些 R\_RDY 表示它希望发起者拥有的信用数量。

## 错误恢复

FC 帧恢复在序列级别进行,这意味着如果帧丢失,则必须重新发送整个序列。这并不是世界末日,因为大多数 FC SAN 都很稳定且不拥塞,这意味着恢复并不经常需要。

## 长距离流量控制

在延伸或扩展的结构配置中,长距离链路两端的交换机端口都配置了额外的缓冲区,以允许发送更多帧并使其存在于链路上而无需等待确认。

一般来说,发送和接收端口之间的距离越长,  
为端口分配更多信用。如果没有分配额外信用,则有效

由于带宽延迟积,发送方等待 R\_RDY 帧时这些链路的带宽会减少。此外,在全双工模式下,数据可以在传输线路上发送,信用可以在接收线路上补充。

## 服务等级

为了提供不同级别的服务 (基本上是交付保证和带宽消耗),FC 协议定义了七种服务类别 (COS)。然而,不仅没有全部实现,而且绝大多数 FC 流量都是 3 类或 F 类,现实世界中偶尔会看到少量 2 类。

### 1 类

1 类是使用确认的面向连接的链路。虽然它没有得到广泛实施,但您可能会在考试中看到它。在 1 类连接中,发起方和目标占用链路的全部带宽并维持虚拟电路,直到传输序列结束。确认帧用于确保可靠且有序的帧传送。1 类连接仅使用 E2E 流控制。

### 2 级

第 2 类是使用确认的无连接链路。如果您了解 TCP/IP 理论,它有点类似于 TCP。基本上,它不是专用连接,但需要交付确认。第 2 类连接可以请求按顺序交付帧,但这不是强制性的。与以太网相比,FC SAN 中的无序帧交付很少见。

2 类连接同时使用 E2E 和 B2B 流量控制。2 类连接比 1 类,但仍然不太常见。

### 第 3 类

第 3 类是无连接服务,不使用确认帧。这是 FC 结构中最常见的服务类别,在概念上类似于 TCP/IP 世界中的 UDP。如果您了解 TCP/IP 理论,由于它与 UDP 相似,COS 3 有时被称为数据报服务。

一般来说,FC SAN 环境不会丢弃或丢弃帧,这意味着确认帧是一种不必要的开销。当然,除非您利用这些 ACC 帧来补充信用。如果帧被丢弃,上层协议有责任清理混乱并确保重新传输。

第 3 类仅使用 B2B 流控制。

### 4 类

类 4 定义了一种基于连接的服务,并带有确认功能。它与类 1 的不同之处在于,它占用部分带宽,并且可以将服务质量应用于连接。但它很少被实施。

## F 类

F 类与 2 类非常相似。它是无连接的,但使用确认帧。F 类是用于交换机到交换机通信的实际 COS。它仅在 E\_Port 和 ISL 之间运行。它用于传递控制数据,例如状态更改通知、名称服务器更新、路由信息、分区、构建结构请求和重新配置结构请求。F 类使用 B2B 流控制机制。

## 超额认购

当多个设备使用单个链路时,就会发生超额认购。ISL 经常超额认购。例如,具有 24 个 8 Gbps 端口的边缘交换机可以按以下方式配置:

- 22 台主机,每台连接速度为 8 Gbps
- 2 个 ISL 端口配置为单个 16 Gbps ISL 到核心交换机

为了让 22 台主机访问连接到核心交换机的存储阵列,所有 22 台主机都必须跨越 16 Gbps ISL。如果所有 22 台主机都充分利用了 8 Gbps,则 ISL 将成为严重的瓶颈。但是,如果您知道所有主机的总吞吐量永远不会接近其 8 Gbps 的最大理论吞吐量,则可以安全地超额订阅 ISL,如所述。

如果结构中的所有端口都以相同的速度运行,则超额订阅只是节点端口与 ISL 的比率。因此,如果您有 10 台 8 Gbps 的主机和一条 8 Gbps 的 ISL,则超额订阅率为 10:1。但是,在上例中,我们有一个以 16 Gbps 运行的逻辑 ISL 和 22 个以理论 8 Gbps 运行的主机,因此我们的超额订阅率为 11:1。

超额认购不仅仅用于 ISL。单个存储端口在扇出配置中与多个主机进行分区也很常见。例如,单个存储端口可能与 10 个主机端口进行分区,扇出率为 1:10。如果主机和存储端口以相同的速度运行,则超额认购率为 10:1。同样,这在现实世界中非常常见,因为 99% 的时间主机不会也不能充分利用其带宽。当您考虑到大多数主机将具有多个活动链路(用于冗余)时,情况尤其如此。

## 位置

局部性是指两台设备之间的通信尽可能保持接近的程度。例如,连接到同一交换机的两台通信设备具有较高的局部性。相比之下,位于相隔三跳的不同交换机上的两台设备具有非常低的局部性。

通过将存储和主机彼此靠近,局部性可以帮助避免拥塞,但不要过度依赖它。将主机连接到边缘交换机,将存储连接到核心交换机是完全可以接受的。这在刀片服务器配置中很常见,刀片服务器机箱中嵌入了边缘交换机。



在大多数 FC 结构中,基于交换机的延迟并不值得为之失眠。*worth losing any sleep over.*  
由于许多 FC 交换机采用高速直通交换机制,因此该时间通常约为每交换机约 1 微秒。*the*

与从旋转磁盘读取数据的延迟相比,任何正常的结构相关延迟(如帧切换或 ISL 跳变)都是微不足道的。当然,闪存介质等固态技术可以避免与旧旋转磁盘介质相关的许多延迟。



### Real World Scenario

#### 注意不要过度设计

请注意不要太过沉迷于局部性,因为这样很容易使设计变得过于复杂而得不到任何好处。

一家公司采取了极端做法,将彼此划分区域的存储端口和主机端口连接到共享通用交换 ASIC 的相邻交换机端口。这样做的目的是,主机和存储之间的流量将在 ASIC 上以极低的延迟进行交换,并且永远不会产生与穿越交换机背板相关的延迟,更不用说与跨越 ISL 相关的任何延迟了。

虽然设计在纸面上看起来不错,但它设计过度,并且创建了一个不可持续的配置,既繁琐又难以维护,最终无法扩展。付出了很多努力,但可能没有实现任何实际效益。简单才是王道!

## 结构重构

某些与结构相关的事件可能会导致结构重新配置。其中一些事件包括:

■向结构中添加了新的交换机。

■合并面料。

■选择了新的主交换机。

有两种类型的结构重新配置事件:

■建材面料(BF)

■重新配置结构(RCF)

构建结构事件不会造成中断,但重新配置结构事件则会造成中断。

在 RCF 事件期间,将举行主交换机选举、分配域 ID 并建立路由。完全有可能再次选举同一个主交换机,并且域 ID 不会改变。但是,如果它们确实发生了变化,这可能会很麻烦,尤其是在使用端口分区的情况下。

出于这些原因,显然最好执行 BF 而不是 RCF。

## FC SAN 故障排除

正如您所看到的,有很多方法可以处理 FC SAN。因此,在很多方面,您的配置都可能出现问题。让我们来看看一些可以排除 FC SAN 故障的方法。

### 备份交换机配置

虽然这并非真正的故障排除,但你应该定期备份你的交换机连接

以防万一您遇到无法修复的混乱情况 (FUBAR) 并且必须恢复到以前的配置。此类备份文件往往是人类可读的文件,如果您需要将损坏的配置映像与以前已知的工作配置进行比较,它们非常有用。另一种选择可能是每次进行更改时都创建一个新的区域配置,并维护以前的版本,如果提交更改后出现问题,可以回滚到以前的版本。

### 解决连接问题

您会发现自己需要排除的许多日常问题包括连接问题,例如主机无法看到新的 LUN 或无法看到 SAN 上的存储或磁带设备。连接问题往往是由于连接错误造成的。

区域划分。

每个供应商都提供了不同的工具来配置和排除分区故障,但以下是降低常见的 CLI 命令会非常有帮助。

### 平

fcping 是流行的 IP ping 工具的 FC 版本。fcping 允许您测试以下内容:

- 设备 (N\_Port) 是否处于活动状态并且响应 FC 帧
- 两个 N\_Port 之间的端到端连接
- 延迟
- 分区 (一些实现会检查两个设备之间的分区)

fcping 适用于大多数交换机平台,也是大多数操作系统和某些 HBA 的 CLI 工具。它的工作原理是向目标发送扩展链路服务 (ELS) 回显请求帧,目标用 ELS 回显响应帧进行响应。

以下 fcping 命令在 Brocade 交换机上运行,证明设备处于活动状态并且响应 FC 帧:

```
LegendarySw01:admin> fcping 50:01:43:80:05:6c:22:ae
正在 Ping 50:01:43:80:05:6c:22:ae [0x 220100],其中包含 12 个字节的数据:
收到来自 50:01:43:80:05:6c:22:ae 的回复:12 字节时间:1162 微秒
收到来自 50:01:43:80:05:6c:22:ae 的回复:12 字节,耗时:1013 微秒
收到来自 50:01:43:80:05:6c:22:ae 的回复:12 字节时间:1442 微秒
收到来自 50:01:43:80:05:6c:22:ae 的回复:12 字节,耗时:1052 微秒
收到来自 50:01:43:80:05:6c:22:ae 的回复:12 字节时间:1012 微秒
已发送 5 帧,已接收 5 帧,已拒绝 0 帧,已超时 0 帧
往返最小/平均/最大 = 1012/1136/1442 微秒
```

当设备在网络上不活跃时,将对相同的 WWPN 运行以下 fcping 命令:

```
LegendarySw01:admin> fcping 50:01:43:80:05:6c:22:ae
正在 Ping 50:01:43:80:05:6c:22:ae [0x 220100],其中包含 12 个字节的数据:
请求超时
请求超时
请求超时
请求超时
请求超时
请求超时
已发送 5 帧,已接收 0 帧,已拒绝 0 帧,5 帧超时
往返最小/平均/最大 = 0/0/0 微秒
```

以下命令显示正在对同一台主机进行 fcping,但这次是从 Linux 接口 n0.170 上的主机并指定 N\_Port ID 而不是 WWPN:

```
fcping -c 3 -h n0.170 -F 0x220100
发送回显到 0x220100
echo 1 接受 2 接受 3 接受 发 16.170 毫秒
回声 送 3 帧,接收 3 7.053 毫秒
回声 帧 0 错误,丢失 6.803 毫秒
率为 0.000%,平均 rt 时间 10.009 毫秒
```

当您在同一个 fcping 命令中指定两个 N\_Port ID 时,某些交换机供应商还会检查分区表。例如,Brocade 交换机上的以下 fcping 命令会检查分区表,以确保指定的两个 WWPN 彼此正确分区:

```
LegendarySw01:admin> fcping 50:01:43:80:be:b4:08:62 50:01:43:80:05:6c:22:ae
来源:50:01:43:80:be:b4:08:62
```

目的地:50:01:43:80:05:6c:22:ae

区域检查:未划分区域

## 跟踪

另一个以流行的 IP 网络工具为蓝本的工具是 fctrace 工具。此工具跟踪到 N\_Port 的路由/路径。以下命令显示 Cisco MDS 交换机上的 fctrace 命令：

```
MDS-sw01# fctrace fcid 0xef0010 vsan 1
```

当前路由为:0xef0010

20:00:00:05:30:00:59:ba(0xffffbee)

延迟:0 毫秒

20:00:00:05:30:00:58:1b (0xffffc6b)

时间戳无效。

20:00:00:05:30:00:59:1e (0xffffcef)

延迟:174860 毫秒

20:00:00:05:30:00:58:1b (0xffffc6b)

Brocade 有一个类似的工具,称为 pathinfo。

在讨论分区问题时,有以下一些好的事情值得考虑:

- 您的别名正确吗?
- 如果使用端口分区,您的交换机域 ID 是否已更改?
- 如果使用 WWPN 分区,任何 HBA/WWPN 是否已更改?
- 您的区域是否处于活动区域设置中?

## 重新扫描 SCSI 总线

在进行分区更改、LUN 屏蔽更改以及几乎任何其他更改主机的 LUN/卷呈现的工作后,您可能需要重新扫描该主机上的 SCSI 总线以检测新设备。以下命令显示如何使用 diskpart 工具重新扫描 Windows 服务器上的 SCSI 总线:

```
DISKPART> 列出磁盘
```

磁盘 ###	状态	尺寸	自由的	动态 Gpt
磁盘 0	在线的	68 GB	0 乙	
磁盘 1	在线的	140 GB	1024 KB	
磁盘 2	在线的	100 GB	1024 KB	

此输出显示三个卷。但是,我们刚刚向此主机提供了第四个卷,但它没有显示出来。要重新扫描 SCSI 总线,我们将发出重新扫描命令:

```
DISKPART>重新扫描
```

请等待 DiskPart 正在扫描您的配置...

DiskPart 已完成扫描您的配置。

现在我们已经发出了重新扫描,以下列出磁盘命令现在显示所有四个设备:

```
DISKPART> 列出磁盘
```

磁盘 ###	状态	尺寸	自由的	动态 Gpt
磁盘 0	在线的	68 GB	0 乙	
磁盘 1	在线的	140 GB 1024 KB		
磁盘 2	在线的	100 GB 1024 KB		
磁盘 3	在线的	250 GB 250 GB		

如果您知道 LUN 掩蔽和分区是正确的,但服务器仍然看不到第四个设备,则可能需要重新启动主机。

## 了解交换机配置转储

每个交换机供应商往往都有一个内置的命令/脚本,用于收集

gs 和日志将被发送给供应商,供其技术支持小组进行分析。这些命令/脚本的输出对于您作为存储管理员也很有用。每个供应商都有自己的这些命令/脚本版本:

思科显示技术支持

Brocade supportshow 或 supportsave

QLogic 创建支持

让我们看一下 Cisco MDS 交换机上的 show tech-support brief 的简化示例。从截断的输出中,您可以看到交换机有两个配置的 VSAN 并列出两个端口,每个 VSAN 中配置一个。您还可以看到其他有趣的信息,例如管理 IP 地址、固件版本等:

```
MDS-sw01# 显示技术支持简介
交换机名称          : MDS-sw01
开关类型            : DS-X9216-K9
Kickstart 图像       : 1.3 (2)bootflash: ///m9200-ek9-kickstart-mz.1.3.1.10.bin
系统映像           : 1.3 (2)bootflash: ///m9200-ek9-mz.1.3.1.10.bin
```

IP 地址/掩码 :10.10.10.101/24  
 交换机 WWN :20:00:00:05:30:00:75:32  
 VSAN 数量 :2  
 已配置的 VSAN :1,9

虚拟存储网络 1: 名称:VSAN0001,状态:活动,互操作模式:默认  
 域 ID:0x6d(109),WWN:20:01:00:05:30:00:84:9f [主体]  
 活动区域:VR,默认区域:拒绝

---

接口 VSAN 管理员 管理员状态	FCOT 运营 运营 港口
模式中继	模式 速度 通道
模式	(每秒千兆字节)

---

接口 VSAN 管理员 管理员状态	FCOT 运营 运营 港口
模式中继	模式 速度 通道
模式	(每秒千兆字节)

---

fc1/1	1	自动开启	fcot缺失	--	--	--
fc1/2	9	自动开启	fcot缺失	--	--	--

<输出被截断>

## 使用端口错误计数器

基于交换机的端口错误计数器是识别物理连接问题的绝佳方法,例如:

- 电缆损坏 (弯曲、扭结或其他损坏的电缆)
- 连接器损坏 (连接器上有灰尘、连接器松动)

以下示例显示了 Brocade 交换机上物理交换机端口的错误计数器:

```
LegendarySw01:admin>portshow 4/15
端口名称:LegendaryHost_HBA1
港口健康:健康
.....
端口标志:0x24b03          当前活动 F_PORT G_PORT U_PORT 逻辑联机
```

端口类型:4.1  
端口状态:1 端口物理:6 In\_Sync 在线的  
端口号:32 F\_Port

距离:正常  
端口速度:8GBps

中断:	90352	链接失败:44	价格:	0
未知:	三十二	同步丢失:3	Fbsy:	0
李:	5240	签名丢失:11		
进程:	84810	协议错误:0		
时间到:	276	无效字:0		
Rx_flushed:	0	无效的crc:0		
Tx_unavail (不可用) :	10	Delim_err:	0	
空闲缓冲区 (Free_buffer) :	2	地址错误:0		
超限:	0	入射角:	18	
暂停:	0	输出端:	50	
奇偶校验错误:	0	Ols_in:	四十五	
2_奇偶校验错误:	0	奥尔斯出局:	33	
CMI_bus_err (CMI总线错误) :	0			

这些端口计数器有时会产生误导。看到某些数值的计数过高是完全正常的，当服务器重新启动或发生类似变化时，数值增加也很常见。如果您不确定要查找什么，请查阅交换机文档，同时将计数器与某些已知良好的端口进行比较。如果您关注的某个端口上的某些计数器在增加，但某些已知良好的端口上的计数器没有增加，则说明该端口存在问题。

其他命令显示类似的错误计数器以及端口吞吐量。以下 porterrshow 命令显示端口 0 上的一些编码输出(enc out)和 3 类丢弃(disc c3)错误。这可能表示电缆损坏、端口损坏或其他硬件问题：

以下portstatsshow命令显示一个干净的端口,没有记录任何错误:

```
LegendarySw01:admin> portstatsshow 1/6
stat_wtx          0      传输 4 字节字
stat_wrx          0      接收 4 字节字
stat_ftx          0      已传输帧数
stat_frx          0      已接收帧数
stat_c2_frx       0      收到 2 类帧
stat_c3_frx       0      收到 3 类帧
stat_lc_rx        0      已接收链路控制帧
stat_mc_rx        0      已接收多播帧
stat_mc_to        0      多播超时
stat_mc_tx        0      已发送多播帧
tim_rdy_pri       0      时间R_RDY高优先级
tim_txcrd_z       0      时间 TX 信用零点 (2.5Us 刻度)
tim_txcrd_z_vc 0- 3: 0      0      0      0
tim_txcrd_z_vc 4- 7: 0      0      0      0
tim_txcrd_z_vc 8-11: 0 tim_txcrd_z_vc
12-15: 0          0      0      0
er_enc_in         0      帧内的编码错误
错误代码          0      存在 CRC 错误的帧
er_trunc          0      帧数少于最小值
er_toolong         0      帧长超过最大值
er_bad_eof         0      帧尾不良的帧
er_enc_out         0      帧外编码错误
er_bad_os          0      无效的有序集
er_rx_c3_timeout   0      3 类接收帧
由于超时而被丢弃
er_tx_c3_timeout   0      3 类传输帧
由于超时而被丢弃
er_c3_dest_unreach 0      丢弃 3 类帧
由于无法到达目的地
er_other_discard   0      其他丢弃物
er_type1_miss      0      未命中 FTB 类型 1 的帧
er_type2_miss      0      未命中 FTB 类型 2 的帧
er_type6_miss      0      未命中 FTB 类型 6 的帧
er_zone_miss       0      硬分区帧丢失
er_lun_zone_miss   0      未命中 LUN 分区的帧
er_crc_good_eof    0      具有良好 eof 的 Crc 错误
er_inv_arb         0      无效的 ARB
打开              0      loop_open
```

转移	0	loop_transfer
打开	0	FL_Port 已打开
starve_stop	0	因饥饿而停止租赁
fl_tenancy	0	FL 拥有租赁的次数
nl_tenancy	0	NL 拥有租赁的次数

## 章节概要

主机总线适配器和聚合网络适配器主机通过 HBA 或 CNA 与 FC SAN 交互。这些 PCI 设备在主机操作系统中显示为 SCSI 适配器,通过它们呈现给操作系统的任何存储卷都显示为本地连接的 SCSI 设备。这两种类型的卡还提供 FCP 操作的硬件负载,而 CNA 卡还提供其他协议 (如 iSCSI 和 TCP/IP) 的硬件负载。

交换机和控制器 FC 交换机和控制器提供主机和存储之间的连接。使用交换机进行连接可提供巨大的可扩展性以及良好的性能和改进的可管理性。交换机提供结构服务以帮助对 SAN 进行分区并使其更易于管理。

交换机间链路 交换机间链路 (ISL) 将多台交换机连接在一起,允许它们合并到可从结构中的任何交换机进行管理的公共结构中。

ISL 还可以绑定到逻辑 ISL,提供每个组件 ISL 的总带宽以及提供负载平衡和高可用性功能。

虚拟结构 每个 SAN 都可以划分为较小的虚拟结构,有时称为 VSAN。VSAN 类似于网络世界中的 VLAN,允许您将物理设备划分为多个较小的逻辑 SAN/结构。可以将流量路由到

c 通过使用特定于供应商的技术在虚拟结构之间实现。

## 概括

在本章中,我们深入讨论了无通道存储区域网络,包括我们为什么要有它们以及为什么它们在当今世界仍然有用。我们讨论了 SAN 的物理组件,包括主机总线适配器、电缆、交换机和存储阵列。我们还讨论了 SAN 中的许多逻辑组件,例如结构、名称服务器和分区,以及所有组件 (物理和逻辑组件) 如何协同工作以形成适用于关键任务存储的弹性高性能结构。我们还讨论了各种 SAN 拓扑及其优缺点,并详细讨论了命名路由和流控制。然后,我们通过讨论一些最佳实践以及故障排除提示和示例结束了本章。

# 章节

# 6



## 存储局域网

---

本章涵盖的主题：

- ✓ 40,000 英尺高空的 iSCSI
- ✓ 发起者和目标
- ✓ iSCSI 接口
- ✓ IP 网络考虑因素
- ✓ 发现
- ✓ 登录
- ✓ 会议
- ✓ 智能 SNS
- ✓ 安全
- ✓ iSCSI 网关



在本章中,您将探索日益流行的 iSCSI SAN。本章首先介绍主要概念和原理,然后讨论一些可帮助您构建和管理不同大小和复杂程度的 iSCSI SAN 的技术。由于 iSCSI SAN 建立在 IP 网络之上,因此本章介绍了重要的网络配置选项,可帮助您构建、维护和执行高可用性和安全的 iSCSI SAN。由于有很多网络选项,我会帮助您将这些选项映射到您的个人需求。

最后,本章指出了人们在部署 iSCSI SAN 时犯的一些错误,并介绍了在各种操作系统中配置 iSCSI 的一些示例。

## 40,000 英尺高空的 iSCSI

iSCSI 是一种存储网络技术,允许通过 IP 网络共享存储资源。通常,在 iSCSI SAN 上共享的存储资源是磁盘资源。但是,由于 iSCSI 是 SCSI 协议通过 TCP/IP 的映射(与 SCSI 通过光纤通道等其他传输方式的映射方式完全相同),因此理论上任何 SCSI 设备都可以通过 iSCSI SAN 共享。但实际上,在 iSCSI SAN 上共享的绝大多数设备是磁盘设备或磁带相关设备,例如磁带驱动器和磁带更换器。



iSCSI SAN 与光纤通道(FC) SAN 非常相似,两者有很多共同点。如果您已经熟悉 FC SAN,那么这是一个好消息。但如果对此并不熟悉,也不必担心。读完本章后,您将能够熟练掌握 FC SAN 和 iSCSI SANs. But don't worry if you're new to all of this. By the end of this chapter, you will be able to more than hold your own.

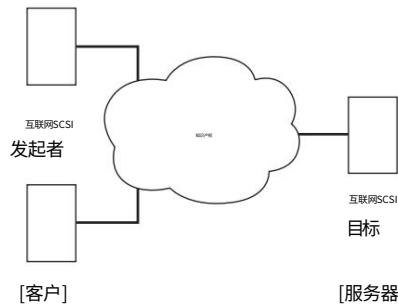
首先,让我们了解一些基础知识。The basics out of the way.

iSCSI 是 Internet 小型计算机系统接口的缩写,正确写法是小写 i 后面跟着大写 SCSI,如下所示。iSCSI 发音为 eye-SKUZ-ee。iSCSI SAN 处理块存储并通过 TCP/IP 映射 SCSI。It is pronounced eye-SKUZ-ee. An iSCSI SAN deals with block storage and maps SCSI over TCP/IP. 虽然理论上任何 SCSI 设备都可以通过 iSCSI SAN 共享,但它几乎总是用于共享主存储,例如硬盘驱动器。备份设备也相当 common. Backup devices are also quite common.

我一直都喜欢图表,所以让我们从高层次来看一下简单的 iSCSI SAN。图 6.1 介绍所有 iSCSI SAN 中的三个主要组件:

- 发起者 (服务器)
- 目标 (通过 iSCSI 共享存储的磁盘阵列或服务器)
- IP 网络

图 6.1 简单的 iSCSI SAN



在 iSCSI SAN 中,发起者通过 IP 网络向目标发出读取数据/写入数据请求。目标通过同一 IP 网络响应发起者。所有 iSCSI 通信都遵循此请求-响应机制,所有请求和响应都作为 iSCSI 协议数据单元 (PDU) 通过 IP 网络传递。

虽然我们将在本章后面详细介绍,但需要注意的是,iSCSI PDU 是 iSCSI SAN 中通信的基本单元。所有 iSCSI 通信均通过 PDU 进行:

- 设备发现
- 设备登录
- 身份验证
- SCSI 命令
- SCSI 数据
- 状态改变通知

### iSCSI 很便宜

人们经常会告诉你 iSCSI 很便宜。这既可能是真的,也可能是假的,所以你需要小心。

当人们说 iSCSI 便宜时,他们通常会将其与 FC SAN 进行比较。虽然 iSCSI 的部署成本确实只是 FC SAN 的一小部分,但这里的关键词是“可以”。事实上,iSCSI SAN 的价格可以随心所欲。但“一分钱一分货”这句格言对于 iSCSI SAN 来说非常正确。

高性能、高可用性和良好的安全性是有代价的。您可以选择超级便宜的选项,将您的软件启动器和软件目标直接放在现有的共享 IP 网络上,然后祈祷一切顺利。或者,您可以在服务器中部署专用的 iSCSI HBA,并通过专用 IP 网络将它们连接到高性能 iSCSI 存储阵列,并使用专用交换机和 IPsec 确保安全。

这些选项之一将为您提供高性能、高可用性和可靠的安全性。

另一个则不然。关键在于,iSCSI SAN 可能非常便宜,也可能非常昂贵,选择哪种取决于您的需求。

现在您已经了解了基础知识,让我们开始深入了解。

## 发起者和目标

iSCSI SAN 由通过 IP 网络连接的启动器和目标组成。虽然启动器和目标在技术上是在各自设备上运行的进程,但大多数人将服务器称为启动器,将 iSCSI 存储设备称为目标。事实上,人们不会使用这些技术术语,而是会草率地使用术语 iSCSI 主机和 iSCSI 阵列。

iSCSI 目标通常是连接有大量磁盘的服务器或专用存储阵列。在较大的环境中,目标往往是专用存储阵列,因为它们往往提供最佳性能、可用性和功能集。它们的成本通常也比具有大量共享磁盘的标准服务器高得多。

### iSCSI 接口

iSCSI 启动器和目标需要物理网络接口。这些接口通常是 PCI 设备,要么集成到服务器主板,要么作为 PCI 扩展卡包含在内。它们通过铜缆或光纤电缆连接到中间 TCP/IP 网络。

选择 iSCSI 接口时,有四种主要选项。与生活中的大多数事情一样,您付出越多,得到的越多。问题是:您得到的是什么?

值得你多付钱吗?以下选项按从最便宜到最昂贵的顺序列出:

**选项 1:带有软件启动器的标准以太网 NIC** 这是最便宜但可能也是最受欢迎的选项。它使用称为软件启动器的基于主机的软件将普通以太网 NIC 转换为 iSCSI 启动器。大多数现代操作系统 (包括 Linux 和 Windows) 都带有内置 iSCSI 软件启动器,通常作为基于内核的设备驱动程序实现。这使得将服务器配置为 iSCSI 启动器或目标非常便宜且非常简单。但是,它们不支持将操作系统启动分区放在 iSCSI SAN 上,并且它们会消耗主机 CPU 周期,尽管后者对于现代 CPU 来说已不再是问题。

**选项 2:带有 TCP 卸载引擎和软件启动器的以太网 NIC** 这与选项 1 类似,主要区别在于以太网 NIC 具有 TCP 卸载引擎 (TOE)。这会将 TCP 堆栈的处理从主机 CPU 卸载到 NIC,在繁忙的 NIC 上,这可能会有所不同。但是,iSCSI 处理 (例如构建 PDU 和封装/解封装) 仍由主机 CPU 处理。这在现实世界中很少见。



*无论是选项 1 还是选项 2,都不允许无盘服务器从 iSCSI SAN 启动除非使用额外的无盘启动软件 (例如 CCBoot)。从 iSCSI SAN 启动无盘服务器。*

**选项 3:iSCSI 主机总线适配器** 这将事情提升到一个新的水平。此选项将所有 TCP 和 iSCSI 处理从主机 CPU 转移到主机总线适配器 (HBA) 上的处理器。这会增加成本,但消除了对主机 CPU 执行 iSCSI 功能的依赖。此外,从功能角度来看,iSCSI HBA 通常带有一个选项 ROM,允许从 iSCSI SAN 启动无盘服务器。

**选项 4:融合网络适配器** 使用融合网络适配器 (CNA) 与选项 3 类似。它提供 iSCSI HBA 提供的所有功能 (增加成本、减少对主机 CPU 的影响以及从 SAN 启动),但还增加了可动态配置除 iSCSI 之外的协议 (例如 FCoE) 的多功能性。

选项 1 是现实世界中最常见的,并且在家中设置非常简单以及其它小型实验室环境。所有这些选项在操作系统中都显示为 SCSI 适配器。因此,操作系统 (除软件启动器外,如果有的话) 无法区分通过 iSCSI SAN 呈现的卷和服务器内部本地磁盘上的卷。这使得在现有操作系统上使用 iSCSI SAN 变得简单而轻松。

决定哪种方式最适合您的环境可能并不那么简单。这将完全取决于您的要求。例如,在非隔离 IP 网络上使用软件启动器和目标的标准 NIC 可能在实验室和小型或

远程办公室。但是,您不会在这种基础设施上运行关键业务线应用程序。为此,您需要考虑更类似于 iSCSI HBA 和专用 IP 网络的东西。

练习 6.1 展示如何配置 Windows Server 2012 软件 iSCSI 启动器。

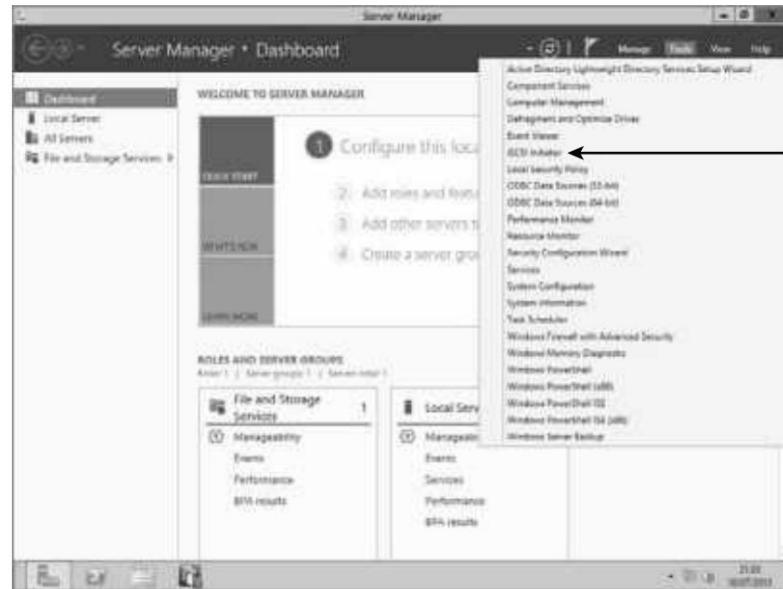
#### 练习 6.1

##### 配置 Windows Server 2012 软件 iSCSI 启动器

在此示例中,您将逐步了解启用 Windows Server 2012 操作系统中包含的软件 iSCSI 启动器的过程。

在本章的这个阶段,假设与 iSCSI 目标建立简单的未经身份验证的连接。您将在本章后面学习如何配置身份验证。

1. 在 Windows 服务器管理器界面中,从右上角选择“工具”选项,如下图所示。



2. 如果 iSCSI 服务尚未运行,系统将提示您启动它。如果提示。

3. 在出现的屏幕上,在“目标”选项卡的“目标”字段中键入 iSCSI 目标阵列的 DNS 名称的 IP 地址。然后单击“快速连接”,如下图所示。



4. “快速连接”选项返回在该 IP 地址或 DNS 名称上运行的 iSCSI 目标列表。从该列表中，选择要连接的目标。在下图中，选择了 iqn.2011-06.com.technicaldeepdive.psansyn01.sybextest 目标。选择所需目标后，单击“连接”。



5. 在以下屏幕上单击“确定”，因为此时您不需要配置任何高级功能。

6. iSCSI 启动器现在尝试连接到所选的 iSCSI 目标。如果成功，它将返回到“目标”选项卡上的“iSCSI 启动器属性”屏幕，并将所选的 iSCSI 目标显示为“已连接”，如下图所示。



现在您已配置 Windows 软件 iSCSI 启动器并将其连接到 iSCSI 目标。现在您可以使用该目标上可用的 iSCSI LUN。

## IP 网络注意事项

IP 网络对于 iSCSI SAN 至关重要，因为它对于性能、安全性和可用性至关重要。如果您的 iSCSI 环境的这一部分出现问题，那么您将面临巨大的损失！



TCP/IP 是一种出色的协议,它提供按序传输和错误恢复,适用于人类已知的几乎所有介质,并且部署极其广泛且易于认知。这使得它成为 iSCSI 传输的最佳选择。您必须了解它在 iSCSI SAN 中的作用和重要性。

## 网络选项

有很多网络选项可供选择。下面介绍的选项按从最便宜到最昂贵的顺序排列。同样,与 iSCSI 接口一样,一分钱一分货。以下列表并不详尽,还有其他网络选项可供选择,但这些是最明显和最常见的:

**选项 1:共享 Vanilla IP 网络** 这是最便宜、最简单、最不安全且性能最差的选项。在此配置中,iSCSI 流量与所有其他网络流量共享相同的 L2 广播域。它有效且简单,但通常仅适用于实验室和家庭办公室环境。如果 IP 网络不稳定或非常繁忙,则会出现性能问题,用户和同事会抱怨。例如,用户在工作日中午通过网络备份 iTunes 库可能会导致大量数据包丢失,并使 iSCSI 性能陷入低谷。

**选项 2:专用 VLAN** 该选项为每个服务器上的 iSCSI 流量提供专用网络连接,可将 iSCSI 流量与其他网络流量隔离。

这可以提高性能,如果 VLAN 不可路由,它还可以提供适度的安全性。由于此选项通常不需要任何额外的网络套件,因此它可以降低成本,同时提供相当大的性能和安全性提升。

**选项 3:专用物理网络** 这是最安全的选择。它提供最佳的安全性、性能和可用性,是现实世界中最常见的选择。

但您可能已经猜到了,它的成本最高。与 iSCSI HBA 或 CNA 一样,如果您在 iSCSI SAN 上部署任务关键型业务线应用程序,并且希望晚上能够安然入睡,那么这是您的最佳选择。增加一层安全性是使用两个具有独立路径的专用交换机,有效地创建网络 A 和网络 B,就像我们在 FC 世界中所做的那样。这个选项非常安全,您晚上可以睡得安稳,白天也可以!

这三个并不是全部的选择,但它们构成了最广泛使用可用的网络选项。

以下是关于一般网络配置的一些关键建议:首先,选择选项 2 或选项 3 最好是选项 3。它将为您省去很多麻烦。接下来,让网络配置尽可能简单;不要自作聪明。站点间连接的路由是必要的,但不要将您的 iSCSI 网络路由到您的用户网络。对于许多

供应商,禁用连接到 iSCSI 启动器和目标的端口上的生成树协议 (STP) 也是最佳实践。每个供应商都有自己的 STP 最佳实践,但禁用连接到 iSCSI 启动器和目标的端口上的 STP 是很常见的。如果您的网络需要某种形式的 STP,快速生成树协议 (RSTP) 可能是更好的选择,并在连接到 iSCSI 启动器和目标的所有端口上启用 PortFast,这样当网络事件发生时,这些端口将立即处于 STP 转发模式。BPDU Guard 还可以用于启用了 PortFast 的端口,这样连接到它们的设备就不会参与 STP。

您可能听说过,巨型帧 (大于默认最大值 1,514 字节,最多 9,000 字节的以太网帧)对于 iSCSI 是必需的。巨型帧的实施可能 (并且经常)导致问题,特别是如果网络中的所有交换机和设备未配置相同的巨型帧配置。如果您对网络有信心或拥有一支优秀的网络团队,您可能希望这样做,因为它可以带来性能优势。但是,如果您不确定自己在这方面要做什么,您可能会发现不值得这么麻烦。

## 通过 WAN 或 LAN 的 iSCSI

虽然 IP 网络可以是企业 WAN 甚至是 Internet,但绝大多数部署都是在企业 LAN 上。这种情况非常普遍,因此 iSCSI 可能更适合称为 Intranet SCSI,而不是 Internet SCSI。这种对 LAN 的偏爱既来自性能方面,也来自安全性方面。

请记住,存储流量和存储流量要求与大多数传统 TCP/IP 网络用途有很大不同。



### Real World Scenario

#### 忽略网络要求

现实世界中最常见的错误之一,尤其是对于那些缺乏 iSCSI 经验的人来说,就是忽视网络要求。

一家公司将 iSCSI 直接部署到现有的共享企业网络上,因为他们认为所需的只是任何旧 IP 网络。这导致性能不可预测且通常很差。他们浪费了数周时间调查性能问题的根本原因,一旦确定网络规格不足,在项目预算结束后,还需要额外花费购买专用网络套件。

### iSCSI 电源分配单元

既然我们一直在谈论网络,让我们快速了解一下 iSCSI PDU 以及数据如何在 iSCSI 所涉及的各个层中封装。

进一步了解一下,iSCSI PDU 由一些强制性的和一些可选段。这些段中最重要的是基本头段 (BHS),它存在于每个 iSCSI PDU 中。它是一个固定的 48 字节字段,包含 SCSI 命令描述符块 (CDB) 和相关的目标 LUN 信息。它始终存在并且始终是第一个段。通常它是 PDU 中唯一的段,因为所有其他段都是可选的,包括 PDU CRC/摘要,并且通常会被省略。

一般来说,您无需了解 iSCSI PDU 的构成。完整性通过 TCP 校验和以及以太网 CRC 来确保。

## 封装

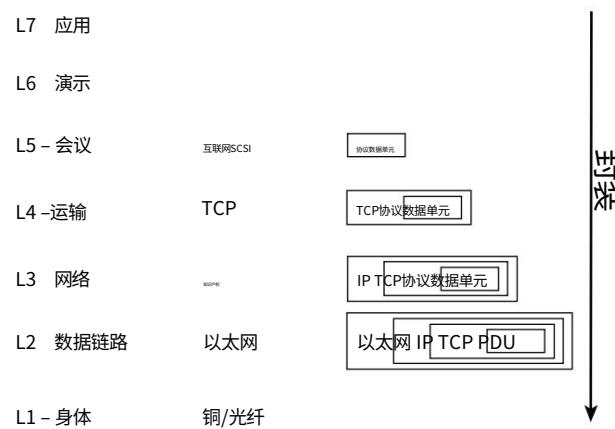
iSCSI 在开放系统互连 (OSI) 参考模型的会话层运行。

iSCSI I/O 操作的生命周期大致如下:应用程序发出 I/O 操作发送给 OS/文件系统,后者将其发送给 SCSI。SCSI 创建 CDB 并将其传递给 iSCSI 层,顺便说一句,SCSI 将其视为任何旧传输。iSCSI 将 CDB 和相关目标 LUN 数据封装到会话层的 PDU 中,并进一步向下传递到各个层,直到它们到达线路以通过网络传输。

当它们传递到不同的层时,会添加额外的封装层。

图 6.2 显示了 iSCSI PDU 如何封装在 TCP 段、IP 数据包以及最终的以太网帧中。

图 6.2 iSCSI PDU 封装



一旦封装在会话层以下,iSCSI 就不再具有任何特殊性;TCP 报头用于可靠传输,IP 报头用于数据包路由。图 6.3 显示了封装的 iSCSI PDU。

图 6.3 封装的 iSCSI PDU



iSCSI PDU 与 TCP 段之间不存在隐式的 1:1 映射。iSCSI PDU 可以跨越多个 TCP 段、IP 数据包和以太网帧。此外,由于 IP 网络的性质,帧 (因此 PDU) 可能会无序到达目标。

这不是问题,因为接收端的 TCP 堆栈负责重新排序段并确保按顺序传送到 iSCSI 层。



在 iSCSI 会话中,命令顺序通过命令序列号 (CmdSN) 进行跟踪。序列中每执行一条命令,CmdSN 就会增加一位。*In iSCSI sessions, command order is tracked using command sequence numbers (CmdSN). The CmdSN is incremented one digit for each command.*

一旦被目标接收,每个封装层都会被剥离,包括 iSCSI PDU 封装,直到只剩下 SCSI CDB。此封装/在具有 iSCSI HBA 或 CNA 的系统上,解封装将以最快的速度发生,因为封装/解封装是由 HBA 或 CNA 执行的,而不是由主机 CPU 执行的。

如果 iSCSI PDU 大小超出 TCP 层的最大段大小,则 PDU 将被分割到多个 TCP 段中。可以通过调整网络上的最大传输单元 MTU 大小来解决此问题,但要小心。只有当您知道自己在做什么时才这样做,因为它违反了“保持简单”的基本规则。



当然,IP 网络的底层传输可以是任何链路层协议,不一定是以太网。但是,它几乎总是以太网,因此在本章中,我们假设它是以太网。*Of course, IP network's underlying transport can be any link layer protocol, not necessarily Ethernet. However, it almost always uses Ethernet, so in this chapter we assume it is Ethernet.*

## iSCSI 名称

iSCSI SAN 上的每个设备都需要一个名称。没有名称,iSCSI SAN 上就没有身份。iSCSI 名称与 iSCSI 节点绑定,而不是节点中的特定接口卡。这一点很重要,因为即使 NIC 和 HBA 在设备之间交换,它也能使名称保留在节点上。

iSCSI 名称也是永久且全局唯一的。如果 IP 地址为节点或卡的地址或完全限定域名 (FQDN) 发生变化。

## iSCSI 限定名称

尽管有三种 iSCSI 命名约定可用,但 iSCSI 限定名称 (IQN) 是迄今为止最流行和部署最广泛的。考虑到这一点,本节将重点介绍 IQN,然后介绍其他两种类型。

IQN 的长度最多为 233 个字节,不区分大小写,不能包含空格,并且是全局唯一的。它们还与位置无关,这意味着如果 iSCSI 节点的 IP 地址或主机名发生变化,IQN 不会受到影响。事实上,iSCSI 设备可以在整个生命周期内保留其 IQN。但是,IQN 不像以太网 MAC 地址或 FC 全球名称那样是固定不变的。如果需要,管理员可以轻松更改它。

iSCSI 名称是分层的,在 IQN 中,三个字符具有特殊含义:

- 点。
- 结肠:
- 破折号-

这些字符划分了 iSCSI 名称。您将在本节后面看到它们的用法。

IQN 的唯一性通过两种方式实现。首先,IQN 包含域名

该域名必须由拥有 iSCSI SAN 的公司拥有。其次,为了避免一家公司可能购买了另一家公司之前使用的域名的权利,IQN 中包含一个日期。该日期应为公司部署 iSCSI SAN 时拥有该域名权利的日期。虽然可以忽略这些准则,但如果更改 iSCSI 名称,强烈建议您遵循此做法!



**通常情况下,公司不会费心更改其设备配置的默认 IQN,这是确保唯一性的最好方法之一,因为供应商已经遵循了本节中概述的指导方针。例如,客户站点的 EMC VNX 阵列在其一个 iSCSI 端口上有以下默认 IQN: iqn.1992-04**

.com.emc:cx:ckm00124XXXXXXX.a10。



**有趣的是,iSCSI 名称中使用的域名不必在 Internet 或本地企业 LAN 上可解析。DNS 解析在定义和使用 iSCSI 名称中不起作用。域名必须由 iSCSI SAN 所属的公司拥有的原因是纯粹是为了保持 iSCSI 名称的全球唯一性。**

现在您已经掌握了一些基础知识,让我们来看一个 IQN 示例:

iqn.2011-06.com.technicaldeepdive:server:lablinuxdb01

尽管乍一看这可能有点复杂,但事实并非如此。

IQN 名称分为四个字段:

- 类型
- 日期
- 命名权限
- 可自定义字符串

看看我们的示例,类型是iqn,将此名称标识为 IQN 类型名称(我们  
我们将很快讨论另外两种类型)。

日期通常是公司注册域名的年份和月份,该日期将用于名称中。日期格式必须是 YYYY-MM。正是注册域名和注册日期的组合确保了域名的全球唯一性。

命名机构是反向格式的域名。反向格式意味着扩展名在前,在本例中为com,后跟.分隔符,然后是 TechnicalDeepDive。

可自定义的字符串可以是您作为管理员决定的任何 UTF-8 文本。

但是,在选择服务器名称时,强烈建议您考虑定义一个有意义且可扩展的命名约定。

格式如下:

类型.YYYY-MM.反向域名:可自定义字符串

类型后面始终跟着句点(.) ,年份和月份始终用破折号(-) 分隔,日期和反向域名始终用句点(.) 分隔,反向域名和可自定义字符串始终用冒号(:) 分隔。

可自定义字符串中冒号后面的内容非常开放。您可以随意使用特殊字符,但从此以后,它们在名称中不再具有任何特殊含义。

现在您已经了解了构成 IQN 的主要组件和描述符,让我们再看一下前面的示例,如图 6.4 所示。

图 6.4 IQN



## 备选 iSCSI 命名约定

IQN 命名标准的替代方案包括扩展唯一标识符 (EUI) 和 T11 网络地址授权 (NAA)。两者都提供全局唯一性,但远不如 IQN 流行,而且都不是人类可读的,至少普通人无法读懂。

EUI 名称格式 (EUI-64) 由字母eui和句点(.) 组成,后面跟着以 16 位十六进制数字结尾。例如:

eui.0123456789ABCDEF

16 个 ASCII 十六进制数字的值由组织上电气和电子工程师协会 (IEEE) 为公司分配的唯一标识符 (OUI) 和公司分配的扩展标识符。两者组合形成 64 位 (16 个十六进制字符)唯一 ID。最高有效位代表 OUI,最低有效位代表扩展 ID。您不太可能会不幸遇到必须处理 EUI 名称的情况。

NAA 命名格式于 2005 年添加到 iSCSI 中,因为 FC 和 SAS 都支持 NAA 命名格式,而 iSCSI 不支持。添加 NAA 的背后考虑是与 FC 和 SAS 的互操作性。幸运的是,这种情况在部署中很少见。但是,以防万一您在考试中遇到这种情况,NAA 命名格式如下: naa,后跟句点(.),后跟所谓的 NAA 值.NAA 值可以是 64 位或 128 位.64 位 NAA 值为 16 个十六进制字符,而 128 位 NAA 值为 32 个十六进制字符.NAA 值不是人类可读的。

## iSCSI 别名

还可以将别名分配给发起程序或目标,这样您就可以为设备分配更方便用户使用的名称。但是,这些别名不能代替正确的 iSCSI 名称来进行发现、登录和身份验证等操作。它们仅可用于在管理工具中为 iSCSI 设备分配更人性化的名称以及用于类似目的。

总结命名,IQN 格式是迄今为止部署最广泛的格式,也是最人性化。EUI 和 NAA 格式非常可怕,但幸好很少部署。

# 设备发现

为了使发起方和目标能够通信,发起方需要能够发现目标。为此,发起方需要了解有关目标的三件事 :IP 地址、TCP 端口和 iSCSI 名称。这些信息是通过发现过程获得的。

有几种发现方法,旨在适应不同大小的 iSCSI SAN。以下是各种选项,从适合最小环境的选项开始,到最适合大型环境的选项结束:

手动设备发现在最小的环境中,您可以使用列出可用目标 (IP 地址、TCP 端口和 iSCSI 名称) 的文件手动配置启动器。

但这既不动态也不可扩展,因此很少使用。从技术上讲,实际上没有执行任何发现,因为目标列表是在发起者上预先配置的。

SendTargets 假设事先知道目标的网络门户使用的 IP 地址和 TCP 端口。发起者向目标上的网络门户发出 SendTargets 命令,目标会以可用目标列表进行响应 (请求-响应)。这在小型到中型 iSCSI SAN 和使用 iSCSI 网关的配置中很常见,但这种情况很少见,将在本章末尾讨论。

自动设备发现 最后一个也是最具可扩展性的选项是启动器事先不知道 SAN 上的任何目标,必须自己寻找目标。服务定位协议 (SLP) 是第一个,它要求在启动器和目标上运行 SLP 代理。启动器上的 SLP 用户代理发出多播消息来定位目标,目标上的服务代理做出响应。SLP 是中小型 iSCSI SAN 的理想选择,但不适合大型 SAN。老实说,它并不经常部署。对于大型 SAN,建议使用 iSNS。本章后面的“iSNS”部分将更详细地讨论 iSNS。

一旦启动器发现目标,下一步就是使用该目标执行 iSCSI 登录。

## iSCSI 会话

一旦 iSCSI 设备 (启动器和目标) 连接到 IP 网络,并且启动器发现了目标,它们就可以开始建立会话的过程。会话是单个 iSCSI 启动器和单个 iSCSI 目标之间的通信路径。会话包含至少一个 (可能更多) TCP 连接。命令和数据通过这些 TCP 连接以 iSCSI PDU 的形式发送。这听起来可能很难理解,但它很简单,在本章结束时,它将一目了然。让我们更详细地了解一下。



从技术角度来说,在讨论 iSCSI 通信时,应从发起方的角度描述 iSCSI 传输方向,即传出的 iSCSI 流量远离发起方,而传入的流量朝向发起方。诚然,这只是一个小次要问题,但值得了解。

## 设备登录

在创建 iSCSI 会话之前,也就是在读取和写入数据之前,发起方和目标需要执行 iSCSI 登录过程。iSCSI 登录是一个两阶段过程:

### 1. 安全谈判

### 2. 参数协商

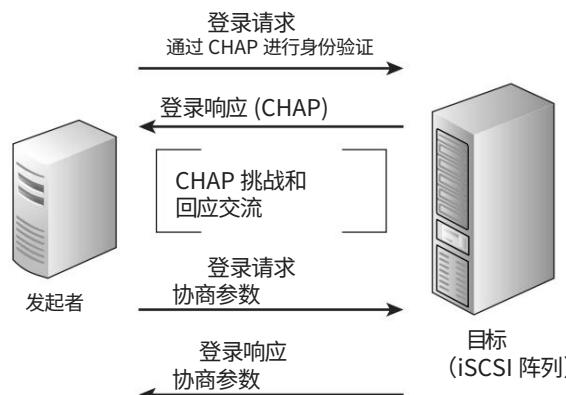
在这两个阶段中,安全协商是可选的。但是,如果安全协商是要发生,它必须在协商其他会话参数之前发生。

登录过程由发起者向 iSCSI 目标的 IP 地址和 TCP 端口发出 iSCSI 登录请求 PDU 来启动。此端口可以是默认的 IANA 分配的知名端口 3260/tcp,也可以是用户配置的端口。在 iSCSI 术语中,iSCSI 目标上的 IP 地址和侦听 TCP 端口的组合称为网络门户。

收到 iSCSI 登录请求后,目标将以 iSCSI 登录响应进行响应。正如我们现在应该可以预料到,登录请求和登录响应将形成 iSCSI PDU,并遵循整个 iSCSI 中常见的请求-响应模型。

图6.5展示了整个登录、认证、参数协商的过程。

图 6.5 高级 iSCSI 登录阶段



## 验证

虽然这不是强制性的,但强烈建议发起者和目标相互进行身份验证。毕竟,IP 网络很容易受到攻击,例如 spoofing 和中间人攻击。尽管部署专用的隔离网络可以在一定程度上避免中间人攻击。为了解决这个问题,应该使用质询握手身份验证协议 (CHAP) 进行身份验证。

尽管 CHAP 可以帮助对发起方和目标进行身份验证,但它无法保护飞行中的数据;数据仍然以明文形式发送。因此,如果您对安全性有严重的担忧,则应考虑其他网络级预防措施,例如隔离网络和 IPsec 等加密技术。然而,虽然 IPsec 非常适合为飞行中的数据提供加密服务,但它可能会带来很大的性能开销。



**Note**

虽然业界普遍认为 CHAP 是一种相对较弱的身份验证方法,但使用随机或其它强密码(共享密钥)可以大大提供帮助。如果您选择容易受到字典攻击的简单密码,则使用 CHAP 的好处会大大降低。此 iSCSI 标准要求从发起方流向目标的流量与从目标流向发起方的流量具有不同的共享密钥。这也有助于避免 CHAP 的一些众所周知的弱点。

虽然有时可能会有其他非基于 CHAP 的身份验证选项,但应谨慎考虑这些选项。由于 CHAP 是标准中指定的唯一安全身份验证机制,因此它是迄今为止最广泛实施和支持的机制。其他可能更强大的选项的小众性质使它们更难以实施和支持,并且可能存在互操作性问题。

如果您有安全意识,最好的建议是部署到隔离的不可路由网络,使用 CHAP 进行身份验证,使用强共享密钥,并使用 IPsec 加密传输中的数据。将管理接口放在单独的管理网络/VLAN 上也是常见的做法。重点是有很多选择,您可以选择适合您需求的选项。

现在让我们在 iSCSI 目标(阵列)上配置 CHAP 身份验证以及练习 6.2 和练习 6.3 中的 iSCSI 启动器。

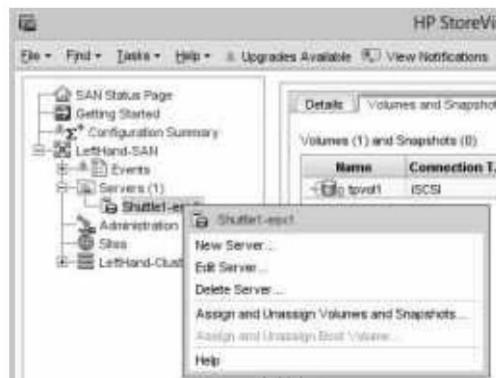
### 练习 6.2

#### 在 iSCSI 目标(阵列)上配置 CHAP 身份验证

在本练习中,您将在 HP StoreVirtual iSCSI 阵列上配置一个目标,以要求 CHAP 和相互 CHAP 身份验证。

1. 在我们的 StoreVirtual iSCSI 阵列的 GUI 中,从树窗格中选择 iSCSI 目标。

右键单击目标(服务器),然后从上下文菜单中选择“编辑服务器”,如下所示。



2. 在编辑服务器配置屏幕中,选择

屏幕底部的身份验证区域。输入目标密钥和发起者密钥的用户名和密码。请务必记住这些密码。



现在,您已为HP StoreVirtual iSCSI 阵列上的Shuttle1-esx1 iSCSI 目标配置了 CHAP 身份验证。此目标将不再接受未经身份验证的连接/登录请求。

您可以将 HP StoreVirtual VSA 的评估副本下载为 VMware ODF 文件,以便在家庭实验室中使用。

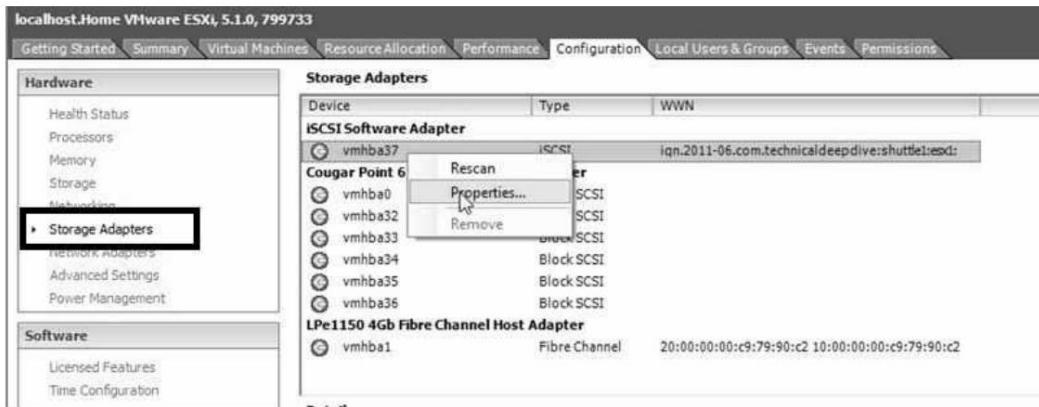
### 练习 6.3

#### 在 VMware ESXi iSCSI 启动器上配置 CHAP 身份验证

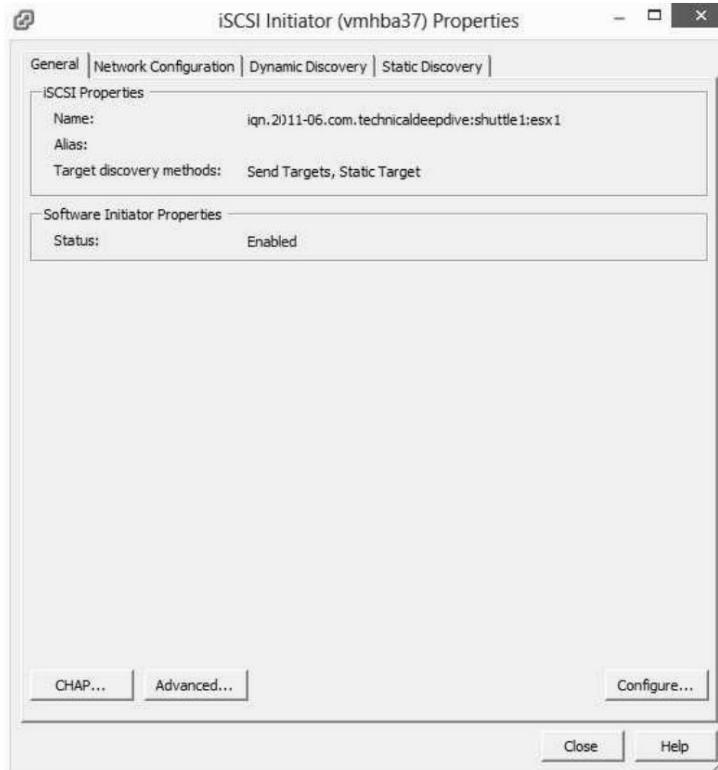
在本练习中,您将逐步完成在运行软件 iSCSI 启动器的 VMware ESXi 上配置 CHAP 身份验证的步骤。

1. 登录到 vCenter,或者直接登录到 ESXi 主机,导航到存储适配器

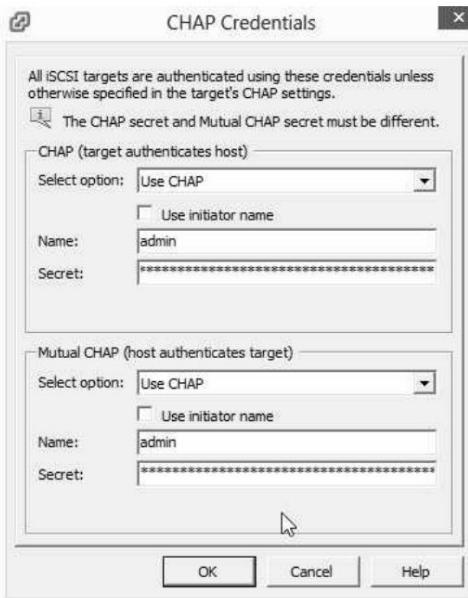
配置选项卡的部分来访问属性。



2.在出现的属性屏幕上,单击左下角的 CHAP 按钮。



3.出现 CHAP 凭证屏幕。在 CHAP (目标验证主机)区域中,从选择选项下拉菜单中选择使用 CHAP 选项。在相互 CHAP (主机验证目标)区域中,进行相同选择。输入名称和机密凭证。然后单击确定。



4.需要重新扫描 HBA。单击此屏幕上的“是”。

5.重新扫描完成后,转到 vCenter GUI 中配置选项卡的存储区域,您将看到您的 iSCSI LUN。

在图像中,您可以从 HP StoreVirtual iSCSI 阵列中看到一个新的 50 GB iSCSI LUN。如果仔细观察,您会看到该设备显示为 LEFTHAND iSCSI 磁盘。这是因为 HP 已将其 LeftHand 产品重新命名为 StoreVirtual,但在幕后,该设备仍将自己标识为 LeftHand。

Identification	Device	Drive Type	Capacity	Free
datastore1	Local ATA Disk (t10.ATA_____ST1000DM...)	Non-SSD	926.50 GB	890.45 GB
ds-iscsi-01	SYNOLOGY iSCSI Disk (naa.6001405023c...)	Non-SSD	249.75 GB	99.60 GB
ds-iscsi-02	SYNOLOGY iSCSI Disk (naa.6001405023f...)	Non-SSD	249.75 GB	94.34 GB
<b>LeftHand-50GB</b>	<b>LEFTHAND iSCSIDisk (naa.6000eb34ffe2...)</b>	<b>Non-SSD</b>	<b>49.75 GB</b>	<b>48.80 GB</b>
software (read on... 192.168.0.7:/volume1/software)		Unknown	6.05 TB	7.84 TB
VSA-DS1	SYNOLOGY iSCSI Disk (naa.6001405f123...)	Non-SSD	499.75 GB	819.00 MB
VSA-DS2	SYNOLOGY iSCSI Disk (naa.60014052705...)	Non-SSD	499.75 GB	819.00 MB

一旦 iSCSI 登录过程完成并且假设它成功,会话将进入技术上称为全功能阶段 (FFP) 的阶段。

## 全功能阶段

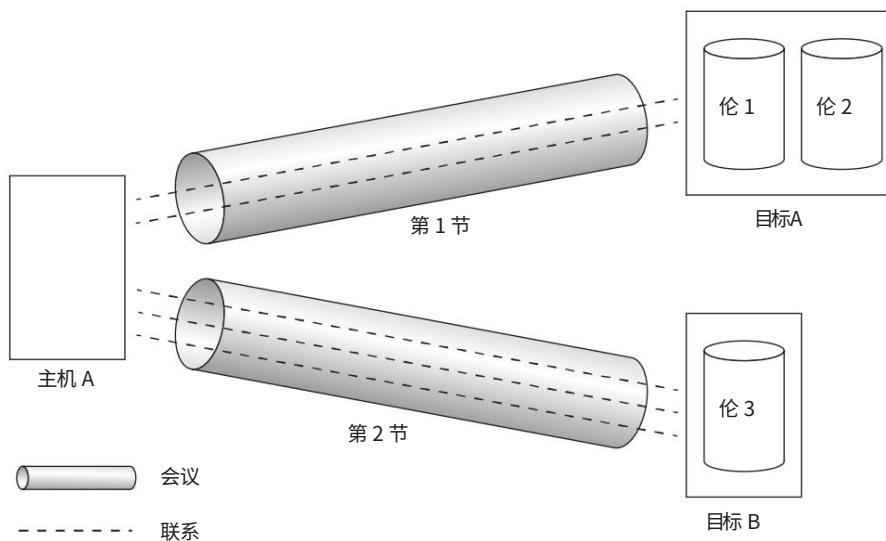
全功能阶段是操作发生的地方。一旦会话进入此阶段,就可以在发起方和目标之间发送命令和数据。在此之前,发起方和目标之间只允许发送与登录相关的 PDU。

现在是时候快速回顾一下 iSCSI 登录过程了。iSCSI 登录是协商安全性、建立 TCP 连接并定义会话进入 FFP 时将使用的所有参数的过程。登录过程由发起方和目标之间的一组请求-响应 PDU 组成。首先,建立 TCP 连接,然后建立 iSCSI 登录会话。成功登录后,会话进入 FFP,从那时起,发起方和目标就可以自由地进行日常数据读写工作了。

FFP 由一个或多个 iSCSI 会话组成。这些会话代表通信发起方和目标之间的通信。每个会话由一个或多个 TCP 连接组成,并且只有一个发起方和一个目标可以参与单个会话。如果发起方想要与多个目标(例如,两个目标)通信,则它必须至少有两个会话(每个目标一个会话)。

让我们看一个发起者与两个目标通信的简单示例。主机 A 是一台数据库服务器,具有来自目标 A 的两个 LUN,目标 A 是一个高性能 iSCSI 存储阵列。LUN1 用于数据库,LUN2 用于日志。主机 A 还具有来自目标 B 的一个 LUN,目标 B 是一个重复数据删除存储阵列。此 LUN(LUN 3)用于备份区域。主机 A 与这两个目标建立连接是完全正常的,但要做到这一点,至少需要两个会话——一个用于主机 A 与目标 A 之间的通信,另一个用于主机 A 与目标 B 之间的通信。参见图 6.6。每个会话可能都有一个或多个 TCP 连接。

图 6.6 iSCSI 会话



会话由唯一会话 ID (SSID) 标识,该 ID 包含一个称为发起方会话 ID (SID) 的发起方组件和一个称为目标门户组标记 (TPGT) 的目标组件。每个 TCP 连接在单个 iSCSI 会话上下文中由连接 ID (CID) 唯一标识。此 CID 在登录和注销以及各种其他操作期间使用。

在一个会话中,相关的请求和响应必须具有所谓的连接性。  
tion 忠诚。连接忠诚要求所有相关请求和响应都通过相同的 TCP 连接进行。例如,如果发起方通过 TCP 连接 X 发出 READ 请求,则目标对发起方的响应必须通过 TCP 连接 X。发起方发出的 WRITE 请求也是如此。如果您还没有理解,iSCSI 会话中的所有通信都是通过 iSCSI PDU 完成的。iSCSI 层 (OSI 模型的会话层)构建 PDU 并管理所有会话。

发起者负责执行注销。在 iSCSI 管理员的现实世界中,没有必要的了解这个过程的具体细节。



在现实世界中,许多流行的 iSCSI 阵列 (例如 Dell EqualLogic) 为每个 LUN 提供单独的目标,从而简化了 LUN  
屏蔽。sent a separate target for each LUN, simplifying LUN masking.

## 互联网

iSNS 代表 Internet 存储名称服务,如果您管理着一个拥有许多 iSCSI 阵列的大型数据中心,您可能想要部署 iSNS。如果您这样做,您会想要很好地了解它。对于大型 iSCSI 环境来说,它是天赐之物。

从高层次来看,iSNS 是一个简单的集中式数据库,其中包含有关 iSCSI SAN 的配置数据。启动器和目标可以向 iSNS 注册信息,也可以从中查询信息。查询 iSNS 允许 iSCSI SAN 上的设备确定 iSCSI SAN 的拓扑结构以及它可见的其他设备的功能和属性。启动器和目标还可以向 iSNS 注册属性,使其成为一个动态存储库,可以大大简化 iSCSI SAN 管理。启动器向 iSNS 注册并查询可用目标的列表。目标向 iSNS 注册,以便启动器可以发现它们。两者还可以注册 iSNS 的状态更新。



如果您了解 FC SAN 并且熟悉 FC 简单名称服务 (SNS),那么您对 iSNS 的理解就已经达到了 99%。因为 iSNS 是  
iSCSI SAN 的模型的。

iSNS 在 iSCSI SAN 中提供以下服务：

■名称服务

■将 iSCSI SAN 分区为发现域

■登录服务

■状态改变通知

**■将 FC 映射到 iSCSI 设备**

其中两个值得进一步讨论 分区和状态变化通知。

在此之前,我们先花一点时间来了解一下 iSNS 服务的一些基本组件。

iSCSI 设备与 iSNS 通信所使用的协议是 iSNSP (iSNS 协议)。

它是一种轻量级请求-响应协议,iSNS 监听的默认 TCP 端口是 3205。发起者、目标、交换机和 iSNS 之间的所有通信都通过 iSNSP 进行。当然,随后会将其封装在必需的 TCP 段、IP 数据包和以太网帧中,以便通过 IP 网络进行传输。

每个启动器或目标上运行的 iSNS 客户端是负责定位 iSNS 服务以及与 iSNS 的所有通信。

iSNS 服务器托管 iSNS 数据库并负责使该数据库可供 iSCSI SAN 使用,以便设备可以在其中注册并查询信息。

iSNS 数据库是 iSCSI SAN 配置的信息存储。这

是一个简单的轻量级数据库,不需要复杂且昂贵的独立数据库服务器。它可以与轻量级目录访问协议 (LDAP) 集成。它可以由发起者和目标动态更新。

现在您已经了解了组成 iSNS 并与其交互的主要组件,  
让我们仔细看看 iSNS 提供的服务。

## 发现域

发现域是一种将 iSCSI SAN 划分为更小、更易于管理且更安全的区域的方法。每个区域都称为发现域,简称 DD。

简而言之,发起者和目标在管理上被分配到发现域。

一旦被分配到发现域,设备就只能发现、登录并与共享公共发现域的其他设备进行通信。设备可以成为多个 DD 的成员,这种情况很常见,但需要注意的要点是,为了使发起者和目标能够相互通信,它们必须是公共 DD 的成员。

将 iSCSI SAN 划分为较小的发现域有几个优点。第一个优点是安全性。实施 DD 后,设备只能查看同一 DD 中的其他设备并与之通信。它们不仅不能登录到其发现域之外的其他设备,甚至不知道其发现域之外的设备的存在。这带来了发现域的第二个重要优势。通过限制 SAN 的视图,它们限制了可以发现和登录的设备数量。

举个例子,假设有一个开放的 iSCSI SAN (管理员的噩梦!) ,没有实施发现域。在这个开放的例子中,如果没有发现域

实施后,您可能会遇到 iSCSI SAN 上的每个启动器都尝试登录并查询 iSCSI SAN 上的其他每个设备的情况。这会耗费时间并浪费资源,更不用说安全风险了。



**发现域的实现依赖于发起者遵守规则,以及目标拒绝来自不在适当发现域中的设备的登录。发现域本身没有严格执行规则的机制。例如,没有机制丢弃从发起者到不与其共享发现域的目标的网络数据包。实际上,发现域中没有任何东西可以物理地阻止行为不当的发起者违反规则。从这个意义上说,发现域是一种通过隐蔽性实现的安全形式。**

发起者不能对他们不了解的目标进行干扰。

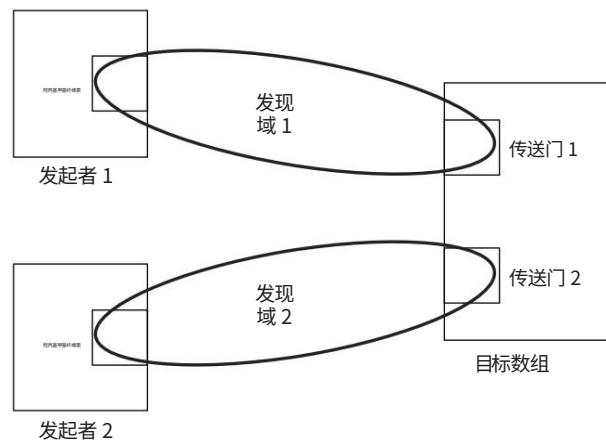
出于安全考虑,强烈建议您在发现域方面采取闭门政策。这与开放政策正好相反。在闭门政策 iSCSI SAN 中,iSCSI SAN 上的新设备会自动注册,不带发现域,直到您明确将它们配置到发现域中,它们才会真正脱离 iSCSI SAN。这是 iSCSI 和 FC SAN 中的最佳实践。



**从安全角度来看,NIS 和发现域仅限制设备发现。目标上的设备掩蔽、用于身份验证的 CHAP 和用于传输中数据隐私的 IPsec 仍然是实现终极安全性所必需的。发现域绝对不会消除对这些其他技术的需求。**

图 6.7 显示了一个小型 iSCSI SAN,它有两个启动器 (Init1 和 Init2),以及一个带有两个网络门户的目标。iSCSI SAN 被划分为两个发现域。在此示例中,Init1 不知道 Init2 的存在。

图 6.7 发现域



作为管理员,您可以将给定网络实体的一个或多个网络门户放入发现域中。将网络门户视为 iSCSI 目标上的一个或多个接口。通过将网络门户放入发现域中,您可以控制哪些接口(网络门户)将向特定发起者发送流量和从特定发起者接收流量。

回顾图 6.7,我们可以看到网络门户 1(P1) 将允许往返于 Init1 的流量,但不允许往返于 Init2 的流量。同样,网络门户 2(P2) 将仅允许往返于 Init2 的流量,而不允许往返于 Init1 的流量。



**门户组**是单个网络实体(iSCSI 目标)内的网络门户集合。这与 NIC 团队并无太大区别。会话可以有多个连接,这些连接跨越门户组中的多个门户。目标门户组标签只是用于标识目标门户组的数字字符串。

值得注意的一个怪癖是,如果你将整个网络实体放入 DD,而不是明确地将特定的网络门户放入 DD,该网络实体的所有门户都将允许该发现域的流量。

为了使发现域处于活动状态并有效(正在使用),它必须是活动发现域集(DDS)。DDS 包含许多发现域,并且作为活动 DDS 成员的发现域是唯一活动且可操作的发现域。

## 状态改变通知

状态变化通知(SCN)是一种将相关配置和拓扑信息中继到 iSCSI SAN 上的设备的方法。

SCN 服务在 iSNS 服务器上运行,这意味着您需要运行带有 iSNS 的 iSCSI SAN 才能从 SCN 中受益。对于 iSCSI SAN 的日常配置和管理,这可能就是您需要了解的全部内容。但是,在故障排除或考试时,还有更多内容可能会有所帮助。

iSNS 发出 SCN 以响应 iSCSI SAN 上的事件和更改。它是 iSNS 负责向 iSNS 注册 SCN 更新的启动器或目标上运行的客户端。如果启动器或目标未向 iSNS 注册 SCN,则不会接收 SCN。

仅存在两种类型的 SCN:

- 常规 SCN 会发给发现域内的所有已注册设备。因此,为了让设备接收常规 SCN,它必须已注册接收 SCN,并且驻留在 SCN 发往的发现域中。这种双重方法可将 SCN 流量保持在最低限度,并确保到达设备的消息是相关的。

- 管理 SCN 只能由已知控制站注册,这意味着常规启动器和目标无法注册管理 SCN。此外,管理 SCN 的分布不限于发现域的边界。作为管理员,您可以控制授权控制站(也称为管理站)的列表。

SCN 注册在 iSCSI 登录时完成。

## 管理 iSNS

管理站（有时称为控制站）是 iSCSI SAN 上具有全权访问 iSNS 的特殊实体。对 iSNS 的管理访问通常是带外（OOB），这意味着它通过与实际 iSCSI 流量分离的接口（IP 地址）进行访问，并且通常仅专用于管理流量。

管理站执行的一些常见管理功能包括：

- 创建、执行和销毁发现域
- 维护授权控制站列表
- 维护管理 SCN 设置
- 设置默认发现域和 DDS 是否启用或禁用（建议  
修补已禁用！）

有一个专用的 iSNS 管理信息库（MIB），即 iSNSMIB，用于管理通过简单网络管理协议（SNMP）控制和监控 iSNS。

由于 iSNS 对于 iSCSI 环境的日常运行至关重要，因此它需要适当的硬件资源。iSNS 过载可能会导致无法生成 SCN 和其他问题。

## 安全

由于安全是一个热门且重要的话题，值得花几分钟来快速回顾一下我们已经讨论过的一些与安全相关的原则和最佳实践，并提及一两个。

iSCSI 标准概述了使用 CHAP 作为发起者和目标的机制来相互验证身份。但是，CHAP 在网络安全领域并不是最有名的。实际情况是，如果使用弱 CHAP 共享密钥（密码），那么 CHAP 只能阻止最粗暴的攻击。但是，如果使用强密钥（不使用字典中可以找到的常用词，包括数字等），它可以成为一种有用的安全预防措施。因此，强烈建议使用 CHAP 并将其与强密钥一起使用。但是，即使使用强密钥，也要注意 CHAP 仅提供发起者和目标身份验证 - 即确保发起者和目标的身份与他们所说的身份相符。它对保护通过线路传输的数据毫无作用。

因此，即使采用配置良好的 CHAP 解决方案，iSCSI 仍然可以通过网络传输数据。以明文形式工作，完全没有本机加密服务。因此，如果您不使用 IPsec 之类的东西，您的所有数据都将以纯文本形式传输，任何监听者都可以轻松读取。如果这是一个问题，您应该认真考虑 IPsec。如果这不是问题，您可能不需要 IPsec。

作为一项额外的安全措施,为 iSCSI 流量运行专用的物理隔离网络或不可路由的 VLAN 可能是一种很好的做法。您还应该将管理接口置于 iSCSI 网络之外。

如果使用 iSNS,建议实施发现域并禁用默认发现域。这样做意味着只有通过管理添加到发现域的设备才能发现并与 SAN 上的其他设备进行通信。

大多数 iSCSI 目标还支持 LUN 屏蔽,其中目标上的资源 (LUN) 屏蔽且仅对特定启动器可见。应始终使用 LUN 屏蔽。

还有其他安全措施,包括更强大的身份验证方法,但许多措施都是小众的,在情况变得艰难时很难支持。实施上述建议足以满足大多数安全要求,并确保您可以高枕无忧,因为您知道您的 iSCSI SAN 中没有任何明显的安全漏洞。

## iSCSI 网关

当环境中存在仅具有 FC 端口的现有光纤通道存储阵列并且您想要将 iSCSI 启动器连接到该阵列时,将使用 iSCSI 网关。iSCSI 网关通过执行 iSCSI 和光纤通道协议 (FCP) 之间的协议转换,以网络网关应有的方式运行。网关的一侧是用于 iSCSI 连接的以太网端口,另一侧是用于 FCP 通信的 FC 端口。网关被配置为 iSCSI 目标,而不是 FC 存储阵列。

这有时被称为桥接拓扑或网关拓扑,并不像原生 iSCSI 环境那样常见。这种配置为 iSCSI 和 FC SAN 增加了组件和复杂性,因此不应优于原生 iSCSI SAN。

比 iSCSI 网关配置更常见的是同时具有 iSCSI 和 FC 连接(有时称为多协议阵列)。这些往往是高端、非常可靠的系统,并且不会受到网关配置所带来的额外复杂性问题的影响。图 6.8 描述了一个简单的 iSCSI 网关配置。

## 章节概要

了解 iSCSI 协议 从概念上讲,iSCSI 是一种 SAN 协议,它映射 SCSI over IP 采用简单的请求-响应模型,其中发起方和目标之间的所有消息都通过 iSCSI 协议数据单元(通常简称为 PDU 或消息)。

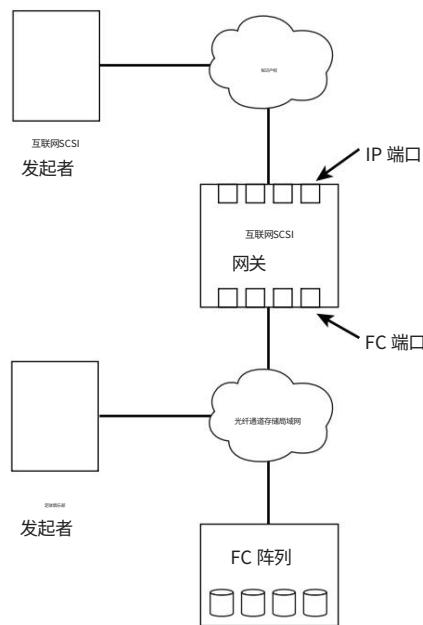
iSCSI 性能 iSCSI 性能受三个主要组件影响。最佳和最一致的启动器性能由专用 iSCSI HBA 实现。最佳和最一致的目标性能由专用 iSCSI 阵列实现。最佳和最一致的网络性能由专用网络交换机实现。

封装和 IP SCSI 命令在网络堆栈的每一层进行封装,以便最终通过 IP 网络进行传输。TCP 层负责传输可靠性按序交付,而 IP 层则提供跨网络的路由。

iSCSI 和安全 应在 iSCSI SAN 上实施多层安全性。

这些包括:用于身份验证的 CHAP、用于限制设备发现的发现域、网络隔离和用于加密飞行数据的 IPsec。

图 6.8 iSCSI 网关配置



## 概括

在本章中,您了解了 iSCSI 存储区域网络的基础知识以及一些高级功能。我们讨论了 iSCSI 启动器和可用的不同启动器选项,以及不同的网络选项。在讨论启动器和网络时,我们讨论了每个选项如何影响性能、可用性和成本。我们还讨论了一些 iSCSI SAN 的基本理论,包括 iSCSI 合格名称 (iQN)、设备发现、登录、身份验证、状态更改通知和发现域。

Machine Translated by Google

# 章节 7



## 文件、NAS 和 对象

---

本章涵盖的主题：

- ✓ NAS 协议,包括 NFS 和 SMB/CIFS
- ✓ 传统 NAS 阵列
- ✓ 统一存储阵列
- ✓ 横向扩展 NAS
- ✓ 对象存储设备
- ✓ 内容感知存储
- ✓ 合规归档



在本章中,您将仔细研究与 NAS、文件服务和对象存储环境相关的主要协议。您将首先研究传统的文件服务和 NAS 环境,然后再研究更现代、更有趣的横向扩展 NAS 技术。然后,您将进入现代且极其有趣的对象存储世界。

本章介绍 NFS 和 SMB/CIFS 文件服务协议,包括两者之间的差异以及各个版本中可用的一些功能。然后,您将比较和对比 NAS 设备与传统通用文件服务器的使用。最后,您将研究对象存储和对象存储设备,包括它们与传统文件服务的比较以及它们如何与大数据和云存储配合使用。

## 什么是文件、NAS 和对象

在存储领域,传统上有两种类型的存储:

- 块存储
- 文件存储

块存储共享原始设备,可以被认为是与位和字节一起工作,而文件存储在更高的级别运行,顾名思义,它与文件和文件夹一起工作。

如果您在 PC 上安装了新的磁盘驱动器,它将在操作系统中显示为原始块设备,您需要使用文件系统对其进行格式化,然后才能开始使用它。如果您的祖母需要在她的 PC 上安装第二个磁盘驱动器,您可能必须为她安装。文件存储则不同,而且简单得多。如果有人通过网络与您共享文件夹,一旦您连接到网络,共享文件夹即可使用。无需费力格式化或进行任何类似的技术操作。

共享文件存储通常称为网络附加存储 (NAS),使用 NFS、SMB/CIFS 等协议来共享存储。一般来说,NAS 存储通常用于非结构化数据存储,例如共享文件夹和文档存储库。

一个值得注意的例外是使用 NAS 阵列为 VMware 提供 NFS 数据存储。

SAN 和 NAS 已经存在多年。SAN 在块级别共享存储,NAS 在文件级别共享存储。两者在大多数企业数据中心中占据主导地位,并且可能在未来很多年内都会如此。然而,它们并不适合云和 Web 2.0 类型媒体所要求的规模,例如社交媒体领域中呈指数级增长的大型图像和视频文件。

这就是一种相对较新的存储类型（称为对象存储）发挥作用的地方。  
对象存储是新兴事物，仍在寻找自己的一席之地。到目前为止，它在公共云领域取得了巨大成功，但在传统的  
企业数据中心却不那么成功（尽管它在企业数据中心的使用案例和需求越来越多）。

本章将更详细地介绍文件/NAS 存储和对象存储。

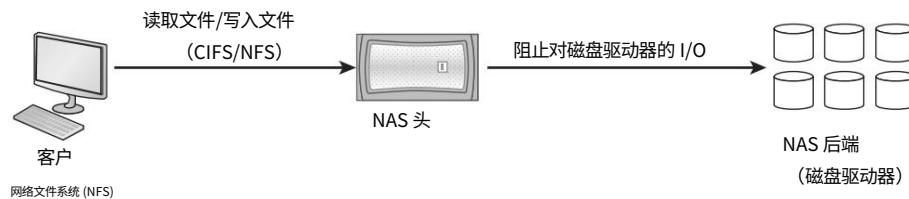
## 网络附加存储

正如本章简介中所述，NAS 不是 SAN。虽然它的首字母缩写词可能使用相同的三个字母，但两者并不相  
同！SAN 在块级别工作，而 NAS 在文件级别工作。由于本章重点介绍文件级别存储，因此我们将仅讨论文  
件，然后稍后再讨论对象。

如前所述，NAS 是网络附加存储的缩写。在企业数据中心，大多数人一想到 NAS 就会想到 NetApp。  
没错，NetApp 可以做块和文件存储，但它主要以文件存储而闻名。然而，NetApp 并不是唯一一家提供  
NAS 存储技术的供应商。

文件比块具有更高的抽象级别。尽管 NAS 客户端以文件的形式与 NAS 服务器通信，但服务器本身  
仍必须通过基于块的协议与其后端的磁盘驱动器通信。图 7.1 显示了一个简单的 NAS 设置，客户端  
安装从 NAS 设备前端导出的文件系统，而 NAS 在后端执行所需的块存储 I/O。实际上，NAS 向客户端  
隐藏了块级复杂性。

图 7.1 后端具有隐藏块访问的 NAS



企业数据中心中最常见的 NAS 存储示例是用户主目录和团队驱动器。在 Microsoft Windows 世界  
中，这些 NAS 驱动器被称为共享、网络共享或 CIFS 共享。在 UNIX 世界中，它们通常被称为导出或 NFS 导  
出。您将很快进一步探索 CIFS 和 NFS。

在 Microsoft Windows 世界中，最常见的是将共享安装到根目录  
驱动器号的路径。例如，您的用户主目录可能被挂载到 H:\，而您部门的团队驱动器可能被挂载到 F:\ 的根目录。Windows 还支持  
挂载点（技术上称为重新解析点），允许将外部文件系统映射到现有文件系统中的任意位置。类似于 Linux/UNIX 世界中的  
工作方式。

在 UNIX 世界中,NFS 导出通常安装到根(/)文件系统中的安装点,而不是驱动器号。例如,您的团队驱动器可能安装到/mnt 目录中名为 tech 的目录。



**挂载点是本地文件系统中的一个目录,新的外部文件系统可以从其中访问。用作挂载点的目录应保持为空。**例如,您的 Linux 服务器可能有一个 /teamstory that is used as the mount point should be kept empty. For example, your Linux server might have a /teams 目录位于其本地硬盘上,但使用此目录作为访问通过网络访问的外部文件系统的挂载点 point for accessing an external filesystem that is accessed over the network.



在 Microsoft Windows 世界中,安装网络驱动器的过程称为映射驱动器。Mapping a network drive is referred to as mapping a drive.

在 Microsoft Windows 和 UNIX 世界中,一些远程文件系统被挂载到本地命名空间 (Windows 世界中的驱动器字母或 UNIX 世界中的挂载点)。您就可以读取和写入文件就好像这些文件和文件夹是您机器本地的 mount point in the UNIX world), you read and write files to them as if the files and folders were local to your machine.

尽管 NAS 的工作级别可能比 SAN (块) 更高且它很简单 than SAN (block), it is by no means simple. NAS 必须处理以下 SAN 所需处理的问题: concerns that SAN does not have to deal with:

- 文件级锁定
- 用户和组权限 group permissions
- 身份验证 authentication
- 防病毒软件 virus software

现在让我们仔细看看最常见的 NAS 和文件共享协议 common NAS and file-sharing protocols.

## NAS Protocols

在 NAS 存储领域,有多种协议。然而,最突出和最重要的协议 (也是您在参加 CompTIA Storage+ 考试时最需要了解的协议) 是网络文件系统和服务器消息块。that you need to know the most about for the CompTIA Storage+ exam—are Network File System and Server Message Block.

### NFS

我们从网络文件系统 (NFS) 开始是正确的。当 Sun Microsystems 发明 NFS 时,它几乎发明了我们今天所知道的网络文件共享。NFS, it pretty much invented network file sharing as we know it today.

NFS 是一种客户端-服务器协议,在 TCP/IP 网络上运行,并且根植于 UNIX (双类语义)。然而,它也在 Microsoft Windows 环境中实现,尽管比在 UNIX 环境中实现的频率要低得多。The two main versions of NFS in use today:

- NFSv3
- NFSv4 (包括 4.1 和 4.2)

在撰写本书时,NFSv3 的部署仍然远远超过 NFSv4 的部署,尽管 NFSv3 在 20 世纪 90 年代中期就已标准化,而 NFSv4 在 21 世纪初也已标准化。

NFS 配置中的主要组件是 NFS 服务器和 NFS 客户端。NFS 服务器通过网络将特定目录导出到特定客户端。NFS 客户端挂载 NFS 导出以从中读取和写入。NFS 服务器可以是通用 UNIX/Linux 服务器,也可以是运行 NFS 服务的 NAS 设备。

### NFSv3 概述

NFSv3 是一种通过 UDP 或 TCP 运行的无状态协议,并且基于开放网络计算 (ONC) 远程过程调用 (RPC) 协议。

NFSv3 的安全性相当糟糕。基本上,NFSv3 的标准规定安全性如下:  
可选的,导致几乎没有人实施任何推荐的安全功能。因此,一些组织,尤其是金融机构,试图避免使用 NFSv3。

NFSv3 在防火墙后配置和连接也比较棘手。这是因为它利用端口映射器服务来确定要监听和连接哪些网络端口。基本上,当 NFS 服务器启动时,它会告诉端口映射器服务它正在监听哪些 TCP/IP 端口号以获取特定的 RPC 号码。这意味着 NFS 客户端必须询问 NFS 服务器上的端口映射器服务,对于给定的 RPC 号码,要通过哪个 TCP 或 UDP 端口号联系 NFS 服务器。如果重新启动 NFS 守护程序,此映射可能会发生变化。在防火墙环境中,这可能会变得有点混乱。

尽管存在这些缺陷,NFSv3 仍然是一个功能强大且流行的文件服务协议。  
tocol,只要您能够忽略巨大的安全漏洞。

### NFSv4 概述

NFSv4 是 NFS 协议向前迈出的一大步,使其与其他网络文件共享协议 (如 SMB/CIFS) 相媲美。然而,尽管在 21 世纪初被批准为标准,但实施它的供应商并不多,部署它的客户就更少了。除此之外,NFSv4 提供了一些很好的功能,而且越来越受欢迎。一些主要改进包括:

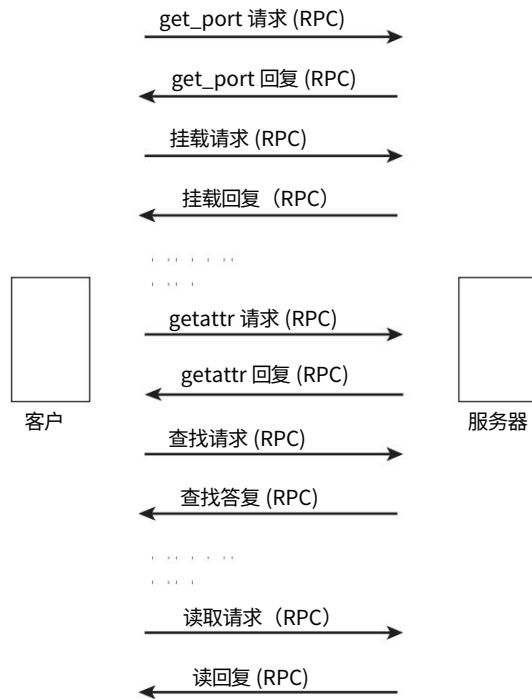
- 类似于 Windows ACL 的访问控制列表
- 强制实施强有力的安全措施
- 复合 RPC
- 委托和客户端缓存
- 通过众所周知的 TCP 端口 2049 进行操作
- NFSv4.1 还带来了并行 NFS (pNFS)

让我们更详细地了解一下其中的一些方面。

### NFSv4 复合 RPC

NFSv3 是一种聊天协议 不是最差的,但也不是最好的。基本上,NFSv3 中的所有操作都是简短、清晰的 RPC 交换,NFS 客户端和 NFS 服务器会通过这些 RPC 相互传递大量 RPC。图 7.2 显示了使用 NFSv3 的客户端和服务器之间的典型对话。

图 7.2 闲聊 NFSv3



通常，简单地挂载导出文件并随后读取导出文件系统上的文件需要 NFS 客户端和服务器之间进行 20 次或更多次交换。

这在 LAN 上并不理想,而在 WAN 上更是噩梦,因为 WAN 中常见的高延迟。NSv4 中的复合 RPC 就是这样。原理很简单。NFS 客户端批量处理操作,并将它们作为更少的 RPC 发送到 NFS 服务器。来自服务器的响应也会被批量处理。这会产生一种更适合 WAN 和 LAN 的更少干扰的协议。

## 定义 WAN、LAN 和 MAN

广域网 (WAN) 是覆盖大片地理距离的网络。无论该网络是私有网络还是通过互联网连接,都没有关系。

例如，一家公司的伦敦和纽约办事处可能通过专用线路连接，而纽约和开曼群岛办事处可能使用互联网上的 VPN 隧道。两者都可以称为 WAN。

局域网 (LAN) 与 WAN 相反。LAN 通常仅限于一栋建筑或非常短的距离（大约 100 米左右）。在 LAN 中，您可以获得较高的数据速率（例如 40 GB 或更高），同时延迟仍然相对较低。

城域网 (MAN) 介于 LAN 和 WAN 之间,通常在城市/大都市区域等中等距离内运行。

### NFSv4 安全性

NFSv4 在安全性方面比 NFSv3 做得好得多。尽管 NFSv3 支持强大的安全性,但人们 (包括供应商)经常无法实现它。NFSv4 通过强制实施强大的安全性改变了游戏规则。

其中一个主要的安全增强功能是使用 Kerberos v5 提供常见的加密服务:

■身份验证

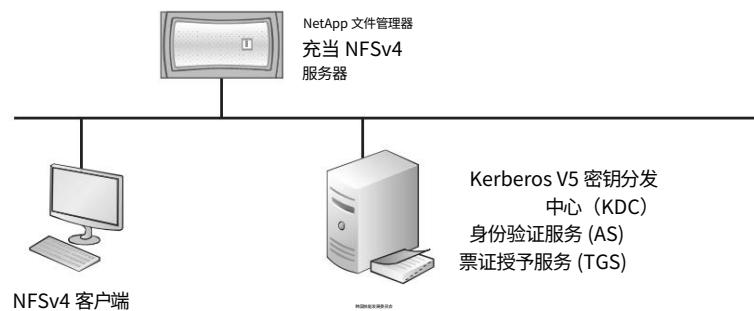
■完整性 ■隐

私 (加密)

支持基于 UNIX 和 Windows 的 Kerberos 密钥分发中心 (KDC)。

图 7.3 显示了具有 Kerberos KDC 的 NFSv4 配置示例。

图 7.3 简单的 NFSv4 配置



使用 Kerberos 在 Linux 主机上挂载 NFSv4 导出需要此处显示的挂载选项:

```
安装 -t nfs4 uber-netapp01:/vol/nfs4_test_vol /mnt/nfs4 -o sec=krb5
```

### NFSv4 访问控制列表和权限

在 NFSv3 中,挂载请求是根据发出挂载请求的客户端的 IP 地址来允许或拒绝的。没有考虑发出挂载请求的用户。NFSv4 完全颠覆了这一点。在 NFSv4 中,一切都是为了对发出挂载请求的用户进行身份验证。

NFSv4 也赶上了 SMB/CIFS,因为它使用了基于 NTFS 模型的适当访问控制列表 (ACL)。这很好,因为 NTFS 安全模型远优于 UNIX 世界中常见的传统基于 POSIX 的读/写/执行权限。

NFSv4 世界中的 ACL 填充了访问控制条目 (ACE)。该模型是一种“除非特别授予否则拒绝”的模型,因此您不必明确拒绝对文件和目录的访问。遵守此原则 (不明确拒绝访问)

与使用明确拒绝相比,可以更简单地解决 ACL 问题。

## 综合起来

现在您已经了解了 NFS 的原理,让我们来做一个创建、导出和挂载 NFSv3 目录的练习。参见练习 7.1。

### 练习 7.1

#### 导出并挂载 NFS 共享

在 UNIX 和 Linux 服务器上, /etc/exports 文件包含所有 NFS 导出文件系统的列表。如果此文件存在并配置,则当 UNIX/Linux 服务器启动时, nfsd 守护进程将启动并导出列出的文件系统。当然,我使用的是我系统上的目录和客户端名称,如果您想主动尝试此操作,则需要替换系统原生的名称。

1. 登录到您的 Linux 服务器 LegendaryLinux01, 使用 exportfs 命令

将 /home/npoulton/test\_export 目录导出到名为 uberlinux01 的 Linux 客户端:

```
#exportfs uberlinux01:/home/npoulton/test_export/
```

2. 上述命令在 /etc/exportfs 文件中创建了一个条目。使用以下命令验证这一点

exportfs 命令:

```
# 导出文件系统
```

```
:/home/npoulton/test_export/uberlinux01
```

3. 现在您的目录已导出到名为 uberlinux01 的 Linux 主机,让我们切换到 legendlinux01 客户端并将导出内容挂载到 /mnt/test 的挂载点:

```
#安装 legendlinux01:/home/npoulton/test_export//mnt/test
```

4. 将目录更改为 /mnt/test 并查看是否可以看到任何文件:

```
# cd /mnt/test
```

```
# ls -l
```

```
-rw-r--r-- 1 npoulton 是 11 月 9 日 10:58 uber_file
```

5. 使用 umount 命令卸载导出:

```
# 光盘 /
```

```
# 卸载 /mnt/test
```

6. 切换回 Linux 导出服务器(legendlinux01)并取消导出目录 -

要执行此操作,请在 exportfs 命令中使用 -u 开关:

```
# exportfs -u uberlinux01:/home/npoulton/test_export/
```



**强烈建议在通过 NFS 导出资源时使用 root squashing。root squashing 将任何从其本地计算机 (*nfs* 客户端) 的 `root` 用户下访问可 NFS 导出的用户的用户设置为 NFS 服务器的 `nfsnobody` 账户的值这有效地确保了他们不会获得 NFS 导出的 `root` 权限。除非您绝对确定自己知道自己在做什么,否则请不要关闭此行为。**

## SMB/CIFS

首先让我们先搞清楚术语。不过这是一个棘手的问题。从技术上讲,该协议是 SMB 协议服务器消息块的缩写。但是几乎每个人都将其称为通用互联网文件系统 (CIFS, 隆音为 cifs)。也最初被称为 CIFS, 但现在正式称为 SMB。我知道这很无聊, 但归根结底, 很少有人关心, 因此通常使用 SMB 而不是 CIFS。它最初被称为 CIFS, 但现在正式称为 SMB。Boring, I know! At the end of the day, few people care, but you're usually safe going with the flow, and *the flow* is calling it CIFS.

同样在术语方面, 您经常会听到人们提到诸如 CIFS 共享、CIFS 客户端和 CIFS 服务器之类的东西。CIFS 共享是使用 SMB/CIFS 协议在网络中共享的任何文件夹。这通常是您使用 UNC 路径连接的任何东西, 例如 red on the network using the SMB/CIFS protocol. This generally is anything you connect to using a UNC path such as this:

```
\\\LegendaryServer\sys\book\book\
```



**UNC 是 Universal Naming Convention (通用命名约定) 的缩写, 它是用于通过网络寻址 CIFS 共享的语法。UNC 路径的格式是 \\ 由 UNC 用于地址的共享, 因此它在 Microsoft Windows 环境中占主导地位 over a network. The format for UNC paths is \\ 服务器名称\ 共享名\rename. Because UNC is used to address CIFS shares, it is predominant in Microsoft Windows environments.**

CIFS 是一种基于客户端-服务器的协议。CIFS 服务器是提供 CIFS 的任何文件服务器。CIFS 共享。这可能是共享其部分本地目录的 Windows 服务器, 也可能是指 NAS 阵列中的文件服务器实例。CIFS 客户端是通过网络访问 CIFS 共享的任何设备。所有 Microsoft Windows 计算机都具有 CIFS 客户端, 无需其他软件。Linux 服务器需要安装 Samba 客户端才能访问 CIFS 共享。All Microsoft Windows machines have a CIFS client, and no additional software is required. Linux servers require a Samba client installed in order to access CIFS shares.

## The Strengths of SMB/CIFS

CIFS 是 Microsoft Windows 世界中事实上的文件服务协议。它已经存在多年, 并且经过调整和改进, 现已成为同类中最好的文件服务协议。CIFS 在 TCP/IP 网络上的 TCP 端口 445 上运行, now a best-of-breed file-serving protocol. CIFS operates natively over TCP/IP networks on TCP port 445.

CIFS 还被许多第三方供应商广泛部署, 因为它是一个公开记录的协议, 尽管它是 Microsoft 特有的。但请注意, 并非所有第三方实现都实现了所有功能, despite the fact that it is Microsoft proprietary. Beware, though: not all third-party implementations implement all features.

CIFS 是一种功能非常丰富的文件服务协议。它支持以下所有功能,此外还支持许多其他功能:

- 身份验证
- 加密
- 配额
- 分支缓存
- 高级权限
- 请求配给
- 请求流水线



**SMB/CIFS 可用于共享和访问文件以外的设备,如打印机和其他权限,并可用于服务器之间的进程间通信(IPC)。**

SMB/CIFS 最初名声不佳。它非常繁琐,历史复杂,没有采用 TCP/IP。后来,随着 Windows 2000 的推出,微软突然接受了 TCP/IP,就好像它发明了 TCP/IP 一样。然而,尽管它在早期阶段很丑陋,但现在看起来好多了!

SMB 2.0 是一次大规模修订,真正提高了标准。主要改进包括大幅减少与网络相关的闲聊、减少协议套件中的命令总数、引入命令复合以及添加命令流水线。

命令复合允许将命令分批发送,并以更少的命令形式发送。响应也是如此。流水线技术更进一步,允许在收到 ACK 之前发送多个复合命令。它还支持 TCP 窗口缩放。最终结果是更优秀、更少干扰的协议,更适合 WAN 用例,包括 Internet。



**如果在网络上使用 SMB/CIFS,请多加小心,因为有很多方法可以利用该协议。您需要小心谨慎,确保您的服务器和  
NFS 设备已修补到最新级别,如果可能,您可能希望使用 Kerberos 进行身份验证和加密。**

SMB 3.0 随 Windows Server 2012 推出,在已经非常出色的 SMB 2.0 的基础上进行了改进,增加了其他各种改进,并引入了 Microsoft 称为 SMB Direct 的一项有趣功能。SMB Direct 允许通过高性能、低延迟的远程直接内存访问 (RDMA) 技术 (如 InfiniBand) 发送 SMB 流量。

在安全方面,CIFS 利用 Kerberos 提供身份验证和加密等加密服务,并且设置起来非常简单。但是,并非所有 NAS 供应商实施的 SMB/CIFS 都实施基于 Kerberos 的加密服务。购买前请务必检查。

总而言之,SMB/CIFS 协议的现代版本非常可靠。Microsoft 正在大力推动该协议,而许多 NFS 商店(客户和供应商)在实施和部署 NFSv4 中已存在 10 多年的许多新功能方面进展缓慢。

微软现在支持将 SQL Server 2012 数据库文件存储在 SMB/CIFS 共享上,这是 SMB/CIFS 最新版本质量的证明。如果协议不可靠且性能不高,则不支持此功能。

映射远程 CIFS 共享很重要但很简单。参见练习 7.2。

## 练习 7.2

### 挂载远程 CIFS 共享

在这里,您将了解如何映射并检查 CIFS 共享的内容。显然,如果您按照这些步骤操作,则需要替换您自己的共享和服务器的名称。

1. 使用 net use 命令将服务器上的远程 CIFS 共享映射到使用 UNC 命名约定的挂载点。在这里,我将 TeamDrive 映射到服务器 W2012-01 到挂载点 Z:\, 但您可以使用任何您喜欢的远程 CIFS 共享、服务器和挂载点。

```
net use z: \\w2012-01\teamdrive
```

2. 现在 CIFS 共享已成功安装在 PC 上,请运行 dir 命令以列出共享的内容:

```
C:\>cd z:  
Z:\>dir  
驱动器 Z 中的卷是 143。  
卷序列号 D2B5-52F
```

```
Z:\ 目录  
2011年09月21日 07:18 <目录> 用户
```

从 dir 输出中可以看到,CIFS 共享包含一个名为 Users 的目录。

---

### SMB/CIFS 权限

SMB/CIFS 文件和文件夹安全基于访问控制列表,允许对文件和文件夹权限进行极其精细的控制。

处理 CIFS 共享上的权限时需要注意一些事项。

首先,CIFS 共享有两组不同的权限：

■共享权限

■ NTFS 权限

共享权限只能应用于共享的根目录,而不能应用于子文件夹或

单个文件。共享权限也仅适用于网络连接,这意味着本地登录的任何人（包括终端服务）都会自动绕过任何共享权限。另一方面,NTFS 权限可以应用于任何文件或文件夹,并且同样适用于网络或本地登录的用户。

NTFS 权限比共享权限更加灵活和细化,并且您所做的大多数权限工作都将使用 NTFS 权限。

由于每个 CIFS 共享都必须具有两种类型的权限,那么它们如何才能有效地协同工作呢?一种非常常见的方法是配置共享权限以授予 Everyone 组完全控制权,然后开始使用 NTFS 权限锁定。这有效地绕过了共享权限。

共享和 NTFS 权限有几个共同的重要点：

■两者都采用隐式拒绝模型。如果没有明确授予访问权限,它将被拒绝。

■适用最严格的权限。

第二点基本上意味着拒绝权限胜过允许权限。例如,考虑这样一种情况,用户 npoulton 是 IT-All 安全组的成员,也是 IT-Contractors 安全组的成员。如果 IT-All 组被允许访问特定文件夹,但 ITContractors 组被拒绝访问,则安全主体 npoulton 将不被允许访问该文件夹。

然而,事情并没有那么简单。显式权限优于继承的权限,甚至是继承的拒绝权限。这意味着,如果某个文件夹从其父文件夹继承了 IT-All 组的拒绝权限,但该组被明确授予对该文件夹的访问权限,则明确授予的访问权限将覆盖继承的拒绝权限。

在枚举共享和 NTFS 权限时,此方法略有不同。如果共享权限允许用户 npoulton 完全控制共享,但 NTFS 权限只允许同一用户进行只读访问,因此 npoulton 用户的有效权限将是只读访问。



*Make sure you know what you are doing before giving NTFS full control rights. This allows the user to change or delete files and their contents.*



运行不同版本 SMB 协议的客户端和服务器据说使用不同的 SMB 方言。当客户端和服务器发起通信时,它们会协商支持的最高方言。

## FTP

FTP 是文件传输协议的缩写。FTP 是一种基于客户端-服务器的协议,在 TCP/IP 网络上运行。FTP 通常用于通过 Internet 传输大型文件,顾名思义,FTP 实际上是在客户端和服务器之间传输(移动)文件。这与 NFS 和 SMB 不同,后者允许文件在被访问和更新时保留在服务器上。大多数 Web 浏览器都支持 FTP,并且还有各种各样的 FTP 客户端可用。

Vanilla FTP 在安全性方面较弱,并以明文形式发送密码。如果需要身份验证和其他安全功能,则应使用带 SSL 的 FTP (FTPS) 或安全 Shell FTP (SFTP)。

除了安全漏洞之外,FTP 还是一种相当原始的协议,与使用最新版本的 NFS 和 SMB/CIFS。因此,FTP 正在逐渐被 BitTorrent 等协议取代。

## NAS 阵列

NAS 阵列是充当文件服务器的专用计算机系统。它们有各种形状和大小,从小型家用系统开始,一直到大规模可扩展的横向扩展 NAS 系统。大多数组织都在中高端系统上开展业务。这些通常是专用的定制计算机,可以汇集和共享大量驱动器。一些 NAS 阵列具有专用的定制硬件,例如 ASIC 和 FPGA,尽管这些硬件越来越不受欢迎,而且现在大多数都是使用商用硬件构建的。在软件方面,情况有所不同。大多数 NAS 阵列运行专有软件,这些软件被精简到最基本的,并经过高度调优,专门用于

cally 负责服务。

运行小型、专用的 NAS 操作系统有几个优点,包括:

- 高性能
- 攻击面小
- 简单的升级和补丁管理
- 更少的错误

高性能源于 NAS 操作系统不必满足其他通用服务器要求,并且可以专门针对文件服务进行调整。这也使得升级和修补变得更容易,因为您拥有更少的移动部件。较小的攻击面源于 NAS 操作系统仅运行代码

特定于文件服务,因此不会受到针对与文件服务无关的代码漏洞的黑客攻击的风险。



专用集成电路 (ASIC) 和现场可编程门阵列 (FPGA) 实际上是定制晶片,是为特定任务而设计的专用处理硬件。两者之间有一些细微的差别,但就我们而言,它们是定制硬件,与英特尔 x86 和 x64 CPU 等商用硬件完全相反。业界正在采取重大举措,摆脱定制硬件,尽可能多地部署商用硬件,在软件中实现智能。这种软件万能的方法成本更低、开发周期更快,几乎是云技术和软件定义技术(如软件定义网络 (SDN) 和软件定义存储 (SDS))的唯一出路。仍有少量定制硬件,ASIC 和 FPGA 进入更传统的存储技术,如 SAN 和 NAS 磁盘阵列。

NAS 阵列的主要目标是汇集存储并通过 IP 网络将其作为文件和文件夹提供给其他系统。通常,用于共享文件的协议是 NFS 和 SMB/CIFS。有时还包括其他协议,例如文件传输协议 (FTP) 和安全复制 (SCP)。但是,NAS 阵列上的协议套件往往落后于已发布的标准,并且没有实现给定协议的所有功能。例如,NAS 设备上的 CIFS 落后于 Microsoft Windows 服务器上的最新 SMB/CIFS 实现是很常见的。这是由于多种原因造成的。

首先,NAS 供应商将始终落后于微软,因为微软拥有并开发了该协议。NAS 供应商实际上复制了微软的行为。其次,并非所有 NAS 客户都需要所有功能,供应商只会实现其客户要求的功能以及他们可以证明开发费用合理的功能。

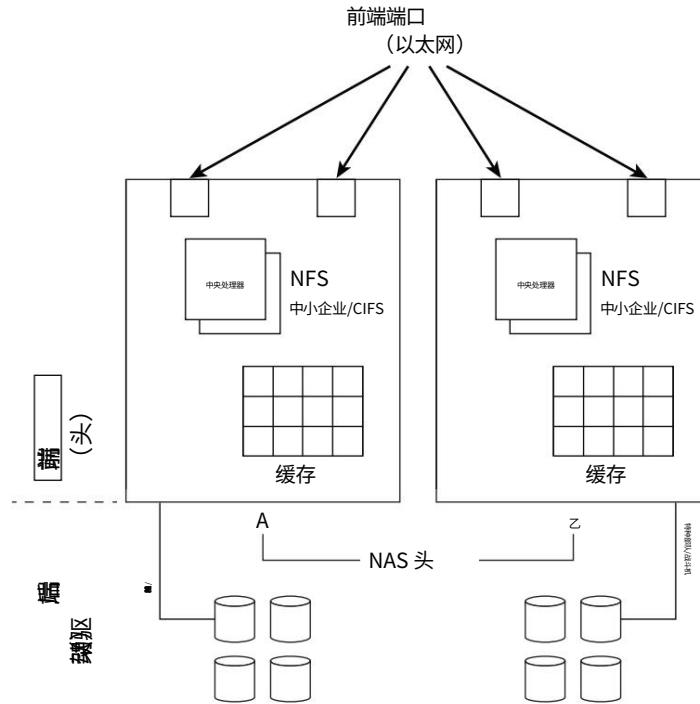
尽管大多数 NAS 阵列都是基于商用英特尔架构构建的,而且通常运行 Linux 内核版本,您无法在其上运行自己的软件。常见的例外是,一些供应商允许您运行专门的 NAS 版本的防病毒软件。

图 7.4 显示了典型 NAS 阵列的逻辑框图。  
NAS 阵列的后端磁盘驱动器可以是套装 NAS 解决方案的一部分,也可以它们可以是来自外部存储阵列的 LUN。后者称为 NAS 网关,本章后面将对此进行介绍。



第 3 章“存储阵列”也讨论了 NAS 阵列 throughout Chapter 3, “Storage Arrays.”

图 7.4 NAS 阵列框图



## NAS 阵列与通用文件服务器

NAS 阵列在现代数据中心中很受欢迎,作为文件共享的整合点和非结构化数据的存储。



结构化数据是指以已知方式高度组织的数据,例如数据库。另一方面,非结构化数据是指大多数其他类型的数据,例如文档存储库和富媒体文件(如图片和视频)。NAS 可以用于这两种存储方式,但更常用于非结构化数据存储。

如前所述,标准的 Windows 或 Linux 服务器可以配置为文件服务器,并执行与 NAS 阵列相同的功能。事实上,有时 Windows 和 Linux 服务器支持更多功能,例如 Microsoft 分支缓存和 Microsoft 分布式文件系统 (DFS) 的一些高级功能。那么,为什么要花钱购买专用的 NAS 阵列呢?

### NAS 优点

从积极的一面来看,NAS 阵列往往在四个关键领域提供优势:

- 简化管理
- 提高性能
- 提高数据可用性
- 高级功能

在简化管理方面,NAS 阵列通常具有专门的 GUI 和 CLI 使文件系统、网络连接和高级功能的管理变得简单很多。将多个通用文件服务器整合到单个 NAS 阵列中是很常见的,这大大减少了环境中的管理点数量。

此外,修补和代码升级可以大大简化,并且通常无需关闭系统即可完成,更不用说硬件升级通常很简单且易于扩展。

从性能角度来看,NAS 阵列倾向于将大量驱动器集中到后端。这会导致性能提高和性能降低的热点。NAS 阵列还可以配备大型 DRAM 和闪存以提高性能。

大多数 NAS 阵列支持多个控制器(有时称为磁头),可提供更高的性能和数据可用性。这些磁头通常采用 n+1 配置,即使单个磁头发生故障,NAS 系统仍可正常运行。

NAS 阵列支持的一些高级功能包括高级 RAID 保护、快照、远程复制、重复数据删除、压缩和虚拟机管理程序集成。

### NAS 缺点

缺点是,NAS 阵列并不总是支持给定协议的所有功能。例如,并非所有 NAS 阵列都支持 NFSv4。此外,并非所有 NAS 阵列都支持 SMB/CIFS 的所有功能。通常,分支缓存和 DFS 的某些方面等功能通常不受支持。



### Real World Scenario

#### 整合至 NAS 阵列

一家零售行业的中小型企业拥有一个杂乱无章的文件服务器,这些服务器基于旧版 Windows(Windows Server 2003)构建,管理起来越来越困难。文件服务器最初使用物理文件服务器中的本地磁盘,但开始超出本地连接磁盘的范围,需要为文件服务器提供 SAN 存储,以应对文件服务器的增长。这种 SAN 存储并不便宜,但文件服务器中没有更多新磁盘插槽来容纳更多本地连接磁盘。因此,决定将文件服务器迁移到 NetApp Filer(NAS 阵列)。最后,公司所有八个

物理文件服务器被迁移并整合到单个 NetApp Filer。此 Filer 是单点管理,能够轻松增加容量,并利用复制和快照技术大幅改善文件服务器资产的备份解决方案。将文件服务器迁移到 NetApp Filer 取得了巨大成功,资产从失控和杂乱无章变为可控,具有增长空间,管理、可用性和备份得到重大改进。

## NAS 与 DAS

在将 NAS 与直接连接存储 (DAS) 进行比较时,重要的是要记住 DAS 提供块存储,而 NAS 提供文件存储 - 因此这不是一个同类的比较。

假设您不关心块存储或文件存储之间的差异,NAS 的主要优势往往是可扩展性和高级功能,例如快照、远程复制、重复数据删除等。但是,NAS 的延迟和每 TB 成本比 DAS 更高。

有时 NAS 就是你想要的,特别是当你在寻找整合的文件时服务,而对于其他要求 (例如廉价的哑存储),DAS 可能是更好的选择。

## NAS 与 SAN

这更像是同类的比较,尽管 iSCSI 协议和块协议之间仍然存在差异。

从技术上讲,差异归结为以下几点。SAN 存储仅通过专用网络共享原始容量,而 NAS 还提供网络感知文件系统(权限、文件锁定等)。这两种方式各有利弊。SAN 存储需要由主机上的卷管理器层进行管理,并写入文件系统,但 NAS 的共享则不需要。

在现实世界中,NAS 有时被视为办公文档和富媒体文件的垃圾场。虽然确实如此,但这对 NAS 造成了极大的伤害。更公平的说法可能是,NAS 天生适合作为非结构化数据的共享存储,而结构化数据集更常见于基于 SAN 的块存储上。

## 传统 NAS 阵列

传统的 NAS 阵列 (如最初的 NetApp 驱动器和 EMC VNX 设备)已经过时,难以满足当今许多非结构化数据需求。它们往往不具备很好的可扩展性,在大型环境中很容易成为管理难题,中型组织拥有 10 个或更多驱动器的情况并不罕见,大型组织在数据中心楼层拥有数百个驱动器,每个驱动器都需要单独管理。

除此之外,他们在小规模上做得还不错,并提供了常见的 NFS 和 SMB/CIFS 协议。它们还提供远程复制、快照、重复数据删除和压缩等功能。

## 统一 NAS 阵列

Unified NAS 阵列将传统 NAS 阵列与块 SAN 阵列整合到一个系统中。这些混合阵列有时被称为多协议阵列,因为它们同时支持文件和块协议。对某些人来说,它们是 Frankenstorage,

而对于其他人来说,它们只是医生开的处方。一般来说,它们适用于小型和有时中型组织,这些组织需要块和文件存储,但无法证明进行两次单独购买是合理的。通常,这些统一存储阵列会为块和文件存储维护单独的驱动器池,以防止两种协议类型的 I/O 相互干扰并降低性能。



*For more information on how to manage a hybrid array, see the sidebar "Unified Storage" later in this chapter.*

## 横向扩展 NAS 阵列

任何了解传统 NAS 阵列的人都会告诉你,它们无法扩展。如果你有大量 NAS 需求,你就不得不部署多个独立的 NAS 阵列,每个阵列都单独管理。这导致了通常所说的 NAS 蔓延。

横向扩展 NAS 竭尽全力为 NAS 带来可扩展性。同时,它还利用了大数据浪潮,企业正在寻求挖掘和利用大量非结构化数据。

虽然横向扩展 NAS 与传统 NAS 有很多共同之处(例如支持以太网上的 NFS 和 SMB/CIFS 协议),但横向扩展 NAS 也遵循一些关键原则。具体如下:

- 水平可扩展性
- 单一全局命名空间

### 水平可扩展性

水平可扩展性只是横向扩展的一个花哨术语。让我们快速定义一下这个术语。考虑在服务器上运行的应用程序。向运行该应用程序的服务器添加更多内存、更多存储或更多 CPU 称为纵向扩展或垂直扩展。向应用程序添加更多服务器(每台服务器都具有 CPU、内存和存储)称为横向扩展或水平扩展。

NAS 世界中的原理是一样的。支持纵向扩展的 NAS 系统将允许您向其添加更多内存和存储,这与横向扩展不同,横向扩展可以添加更多节点/头/控制器。另一方面,横向扩展 NAS 系统将允许您向其添加更多节点,每个节点都会为 NAS 系统带来自己的 CPU、内存和存储。这些横向扩展 NAS 系统通常称为集群或 NAS 集群。



*思考水平扩展的一个好方法是将工作负载分散到更多节点上。 spreading the workload over more nodes.*

横向扩展 NAS 集群支持可动态添加到集群的多个节点。每个节点都有自己的 CPU、内存和存储,它们构成整个 NAS 集群。通常需要高速、低延迟的后端网络,以便读取存储在连接到不同节点的驱动器上的数据不会很慢。10 GB 以太网和 InniBand 是流行的后端互连技术。

编写支持横向扩展的软件和文件系统是一件棘手的事情,这是为什么不是所有的 NAS 系统都是横向扩展的。

### 单一全局命名空间

通常,横向扩展 NAS 集群将拥有一个无限可扩展的文件系统。文件系统有时被称为命名空间。横向扩展 NAS 集群中的所有节点都运行文件系统并对其具有完全的读/写访问权限。

由于只有一个文件系统（命名空间）,并且集群中的所有节点都可以访问它,因此可以将其称为单个全局命名空间。拥有这样的单个全局命名空间自然会导致后端数据放置均匀,这不仅会导致大规模并行 I/O 操作,而且还会减少驱动器和节点热点。

除了规模庞大和全球化之外,这些文件系统还允许无缝地在集群中添加和删除节点。这在解决技术更新问题时非常有用,例如用新硬件替换旧硬件时。例如,运行较新硬件的新节点可以加入集群,运行旧版本硬件的节点可以无缝地被逐出,从而使硬件技术更新比传统 NAS 阵列简单得多。

像这样的横向扩展文件系统通常被描述为分布式和集群感知的。  
通常,当数据写入横向扩展 NAS 系统时,数据将分布在整个后端。

任何管理过大型传统 NAS 阵列（其中多个文件系统仅锁定在两个节点）的人都知道,这是多么令人头疼的管理问题。单一全局命名空间对于简化管理和提高可扩展性大有裨益。

图 7.5 显示了传统的双头 NAS,后端有多个文件系统,每个文件系统主要由其所属/连接的节点访问。图 7.6 显示了小型、五节点横向扩展 NAS,后端有一个文件系统（全局命名空间）,集群中的所有节点都可以平等地访问它。

图 7.5 具有多个后端文件系统的传统双节点 NAS

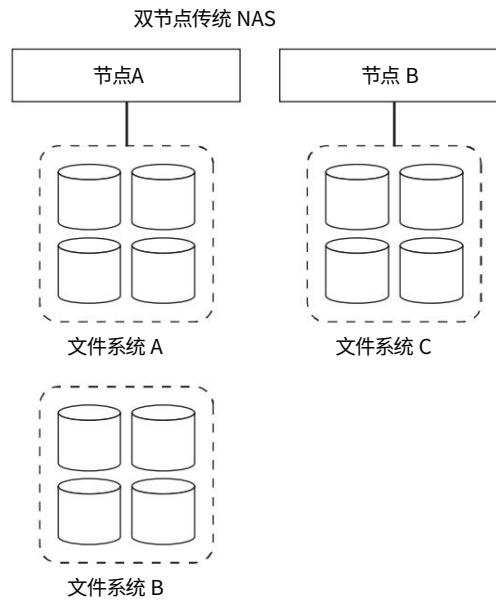
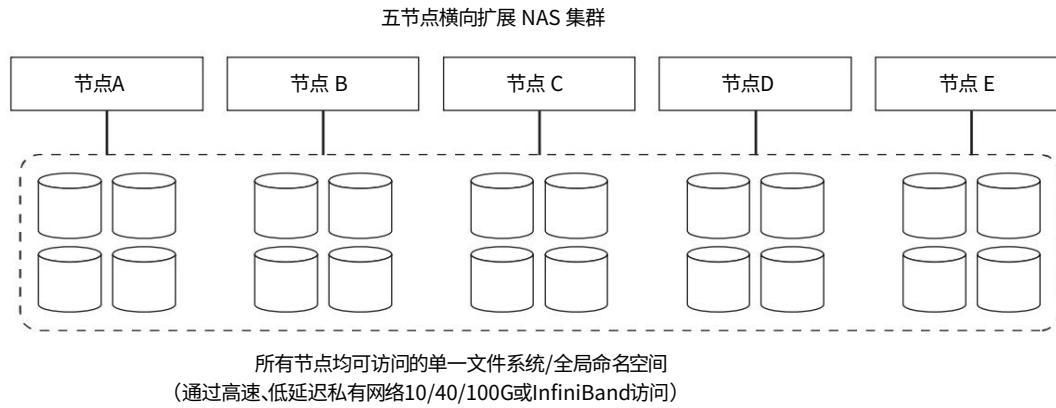


图 7.6 具有单个全局命名空间的五节点横向扩展 NAS



一般来说，横向扩展架构更适合大型环境，因为它们防止 NAS 蔓延。您无需部署许多传统的 NAS 阵列（每个阵列都有多个需要单独管理的文件系统），而是可以部署一个运行单个多 PB 级文件系统的横向扩展 NAS 群集。

### 其他横向扩展 NAS 注意事项

横向扩展 NAS 还存在一些其他常见问题。横向扩展 NAS 往往建立在商用硬件之上。很少看到横向扩展 NAS 运行 ASIC 和 FPGA。此外，横向扩展 NAS 架构通常提供在文件级别工作的灵活保护方案。例如，可以为单个文件分配不同的保护级别。例如，您可能希望保留某些文件的两个副本和另一些文件的三个副本。文件的副本通常保存在集群中的不同节点上。

有时这被称为网络级 RAID 或 RAIN 的一种形式。

### 独立节点冗余阵列

RAIN 是独立节点冗余阵列的缩写，有时也称为独立节点可靠阵列。如果您已经阅读了第 4 章“RAID：保护您的数据安全”，您就会知道 RAID 在驱动器级别提供保护。RAIN 与 RAID 非常相似，但提供节点级别的保护。基本上，在受 RAIN 保护的系统中，您可以丢失集群中的整个节点而不会丢失数据。

RAIN 架构通常通过奇偶校验、纠错码（如 Reed-Solomon 码）或简单地制作文件和对象的额外副本来自提供保护。

RAIN 在分布式计算和云级存储领域非常流行，包括 Amazon S3 等主要对象存储服务。它还是一种允许您在商用硬件之上构建高弹性解决方案的技术。

横向扩展 NAS 集群通常还支持更特殊的协议，例如并行 NFS（pNFS）。

## 网关 NAS 设备

NAS 网关是传统的 NAS 前端，后端没有自己的本机驱动器。基本上，NAS 网关是运行 NAS 操作系统和文件共享协议的 NAS 头（节点/控制器），但在后端没有任何自己的容量。NAS 网关没有自己的驱动器，而是连接到外部 FC 存储阵列以获取其容量。如图 7.7 所示。

外部存储可以是第三方存储，既可以专用于 NAS 网关，也可以与其他设备共享。但是，如果外部存储与其他系统共享，则通常会将特定驱动器专用于 NAS 网关，如图 7.8 所示。

图 7.7 NAS 网关连接 FC 存储阵列

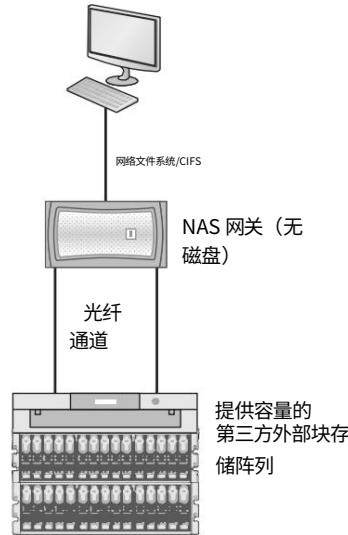
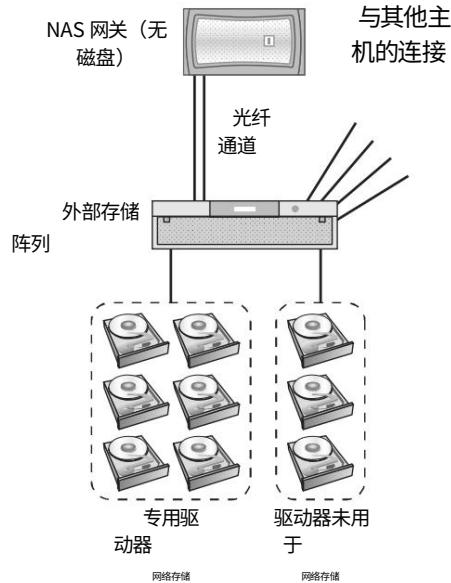


图 7.8 共享外部阵列上的专用 NAS 网关驱动器



注意 NAS 网关配置的复杂性，并确保您遵守供应商支持的指南，以便如果出现问题并且您需要依赖供应商，他们会为您提供支持。

显然，NAS 客户端并不知道它正在与传统 NAS 进行通信，自己的专属存储后端，或者是否正在与 NAS 网关对话。

## 其他 NAS 阵列优点

正如第 3 章所述,NAS 阵列通常提供一些重要的技术,例如:

- 重复数据删除
- 压缩
- 远程复制
- 快照
- 虚拟机管理程序集成和卸载

## NAS 性能

除了磁盘驱动器速度和缓存大小等常见的存储问题外,还有许多其他因素会严重影响 NAS 性能。让我们花点时间重点介绍其中的一些。

### 网络对 NAS 性能的影响

由于 NAS 技术通过 IP 网络运行,因此它们会受到网络性能的极大影响,无论是正面影响还是负面影响。与所有存储相关的事情一样,延迟通常是最重要的性能特征。高延迟会导致用户愤怒,如果情况变得非常糟糕,应用程序可能会变得毫无用处,甚至被认为无法运行。所以请认真对待这个问题!

因此,应该认真考虑 10 GB 以太网和专用 LAN。

10 GB 以太网提供更高的带宽,专用网络可消除拥塞,而拥塞是延迟的致命因素。此外,如果延迟确实令人担忧,而您又无法使用 SAN 或本地连接存储,则应认真考虑使用低延迟交换机,并至少将网络跳数保持在最低限度。

巨型帧还可以提高性能。在低层,客户端和服务器之间的通信通过以太网帧进行。巨型帧可以减少客户端和服务器之间发送的帧数,从而提高整体网络性能。

使用巨型帧时,请确保数据路径中的所有设备 (NIC、交换机和 NAS 设备)都支持并启用它们。

链路聚合技术 (有时称为绑定)也可能有帮助。

链路聚合可以以 NIC 组合的形式应用于主机 (客户端),也可以通过链路聚合控制协议 (LACP) 等技术应用于网络。这些技术可以帮助提高吞吐量以及多路径故障转移。



链路聚合类似于 FC 世界中的 ISL 中继端口通道 / port channels in the FC world.  
基本概念是将两个交换机之间的多个链路组合成一个逻辑链路以提高吞吐量并改善故障转移。

TCP 窗口缩放也会影响性能,尤其是在 WAN 等高延迟网络上。TCP 窗口框架的工作原理是服务器通告它可以接受多少未确认数据,然后停止接受更多数据。较大的 TCP 窗口允许服务器接受更多未确认数据,这可以显著提高 WAN 等高延迟网络的性能,因为在 WAN 等高延迟网络上,从一个主机发送的数据可能需要很长时间才能到达目的地。将两个主机之间的链接视为管道;较大的 TCP 窗口允许您填满管道。

### 与网络无关的 NAS 性能影响因素

远离网络,NAS 和基于文件的存储具有较高的元数据开销:元数据与用户数据之比很高,这意味着等待缓慢的元数据操作完成可能会使 NAS 性能下降。因此,一些 NAS 设计(包括横向扩展 NAS 集群)确保元数据始终存储在闪存介质上以便快速访问。

如果文件扫描开销过高,防病毒(AV)解决方案也会影响 NAS 性能。在 NAS 部署中实施 AV 解决方案时,请务必遵循供应商的最佳实践。

在 NAS 环境中实施安全措施也会增加开销并降低性能。早期在隐私模式(端到端加密)下实施 Kerberized NFSv4 会给操作带来显著的 CPU 开销。但是,现代更快的 CPU 已大大改善了这一方面。在实施这些措施之前,您仍应测试这些措施对性能的影响。

## 对象存储

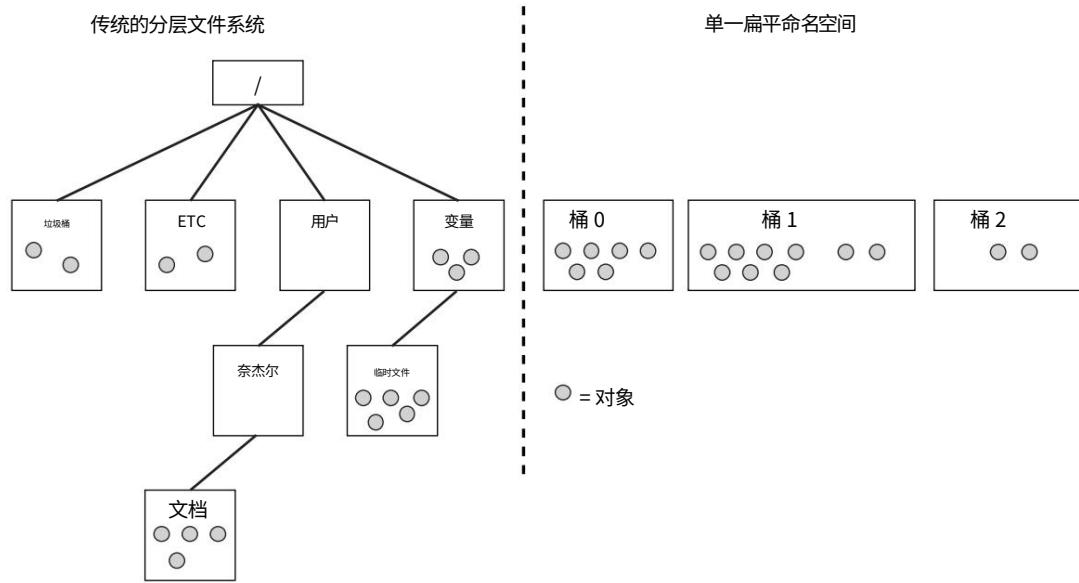
让我们先快速地进行一下高层次的概述,然后再进行更深入的了解。对象存储对于某些人来说需要一点思维上的转变。请记住,它旨在用于云规模和云用例,并且设计为通过 RESTful API 进行访问。您很快就会更仔细地研究所有这些。

如果我们将对象存储与 SAN 或 NAS 存储进行比较,那么可以说对象存储与 NAS 的共同点多于与 SAN 的共同点。这主要是因为对象更像文件而不是块。事实上,可以说在很多情况下文件就是对象,对象就是文件。但是,对象存储设备(OSD)与 NAS 系统之间存在一些关键差异。

首先,对象并不总是具有人性化的名称。它们通常由非常长(例如 64 位)的唯一标识符来标识,这些标识符是从对象的内容加上一些任意哈希方案派生出来的。

接下来,对象存储在单个大型的 at 命名空间中。扁平意味着没有传统文件系统那样的层次结构或树形结构。此 at 命名空间是对象存储系统固有的大规模可扩展性的关键因素。即使是具有大规模 PB 级文件系统的横向扩展 NAS 系统,也会限制文件系统或文件系统内各个目录中的文件数量。图 7.9 显示了传统的分层文件系统与对象存储的 at 命名空间的比较。

图 7.9 文件系统层次结构与平面对象存储



对象存储和 NAS 之间的另一个主要区别是对象存储设备

通常不像传统的 NAS 文件系统那样通过网络安装。对象存储设备不是通过 NFS 和 SMB/CIFS 等协议安装和访问，而是通过 REST、SOAP 和 XAM 等 API 访问。通常，它会是一个与 OSD 而不是人类对话的应用程序。



存储网络行业协会(SNIA)开发了XAM API旨在为对象存储和CAS系统提供基于标准的接口,从而避免供应商锁定。XAM是可扩展访问方法(extensible Access Method)的缩写 standards-based interface for object stores

让我们花点时间看一下 NAS 与对象存储的一个非常有用类比。  
年龄。如果将 NAS 比作外出就餐时停车，您必须亲自将车开进停车场，选择楼层，然后选择停车位。然后您有责任记住停车的楼层和停车位。对象存储更像是代客泊车，您将车开到餐厅前面，下车，换取车票，将停车任务留给餐厅工作人员。当您在餐厅时，您不知道车停在哪里。您唯一关心的是，当您吃完饭和社交后想把车还回来时，您可以交出车票，您的车将和您离开时一模一样地归还。使用对象存储，您（或您的应用程序）创建一些数据并将其提供给 OSD 以换取唯一的对象 ID (OID)。您（和您的应用程序）不关心对象存储在哪里，只要它受到保护并在您下次向 OSD 出示您的唯一 OID 时返回给您即可。这应该非常简单。

对象存储不是为高性能和高变化要求而设计的,也不是它是为存储结构化数据(如数据库)而设计的。这是因为对象存储通常不允许就地更新。对于变化很大的数据,它也不一定是最佳选择。它非常适合存储和检索富媒体和其他 Web 2.0 类型的内容,如照片、视频、音频和其他文档。

现在是时候仔细研究一下对象存储了。

## 对象和对象 ID

对象存储设备用于存储对象。我们所说的对象实际上指的是文件,通常是富媒体文件,例如图像、视频和音频文件。由于它们通常是富媒体文件,所以对象通常很大,但并不一定如此。虽然数据库在技术上也是文件,但我们不会将数据库文件称为对象。

存储在对象存储中的每个对象都分配有自己唯一的对象 ID,称为 OID。OID 通常是一个丑陋的 64 位哈希值,对您来说毫无意义。

但这绝对是至关重要的。

每个对象都存储在对象存储中,同时还存储有描述该对象的大量元数据。

如果我们回到代客泊车的类比,我们提到我们不关心

对象存储是我们存储对象的存储方式。嗯,这只是在一定程度上是正确的。您需要考虑确定诸如对对象应用什么保护级别以及对象在哪些位置可用的策略。在小范围内,位置可以指对象存储在同一数据中心内的哪些节点中;在更大的范围内,它可以指对象的副本存储在哪个国家或大陆。这是因为对象存储在云规模上工作,并且对象存储可以跨越国家和大陆。您可能拥有来自澳大利亚、英国和美国的同一对象存储的节点。

在这三个位置都拥有对象的副本自然会带来保护(类似于 RAID)以及本地数据访问。如果澳大利亚的节点发生故障,您仍然可以拥有英国和美国对象的副本。此外,如果您位于美国,那么从美国节点访问对象将比访问澳大利亚或英国的远程副本更快。话虽如此,与 SAN 或 NAS 系统相比,对象访问仍然不够快。

在定义对象保护策略时,您保留的副本越多,对象受到的保护就越好,但代价是空间。如果您没有明确规定将保存对象的额外副本的位置,OSD 将确保将副本保存在提供最高可用性级别的位置。例如,OSD 不会将三个副本保存在连接到同一节点的同一磁盘上,因为这会使该节点及其磁盘成为单点故障。

另一个考虑因素是对象的数量和大小。理论上,一些对象存储设备支持几乎无限数量的对象和无限大小的对象。实际上,拥有数十亿个对象的对象存储很常见。关于对象大小,这取决于您选择的对象存储技术,但对象可能非常大。我们说的是 TB 级!

## 元数据

如果您认为元数据在 NAS 中很重要,那么请等到看到对象存储。广泛的元数据是基于对象的存储的标志。

忘记尝试记住在哪个目录中,在哪个目录中,在您不再记得名称的目录中,您六周前保存了那个文件。对象存储就是基于元数据进行搜索和索引。一个常见的例子是医学图像,例如 X 射线和 MRI 图像。这些可以用强大的元数据标记,包括患者姓名、患者年龄、性别、种族背景、身高、体重、吸烟者、饮酒者、处方药、现有医疗状况等。

所有这些都可以搜索和利用。另一个常见的例子是交通管理和监控系统,其中保存的视频图像可以用大量元数据标记,例如道路类型、一天中的时间、一年中的时间、天气、能见度、拥堵指标、缩略图视图等。

元数据以标准键值对的形式与对象一起存储。一般来说,  
OSD 将使用基本元数据标记每个对象,例如大小、创建时间、上次修改时间  
编辑时间等等。此后,您或您的应用程序可以定义几乎无限的自定义元数据。OSD 可能不会理解此用户定义的自定义元  
数据,但这并不重要。重要的是它存在并且可以被您的应用程序搜索和利用。

一旦将元数据添加到对象,数据有效负载和元数据就会一起存储在 OSD 中,尽管 OSD 可能会将对象分块以便  
在后端更有效地存储。

## API 访问

如前所述,并非所有 OSD 都提供 NAS 接口。访问 OSD 通常仅通过 RESTful HTTP、SOAP 等 API,或者有时通过特定语言的库和 API (如 Python、PHP、Java 等) 进行。这意味着对象存储绝对不是 NAS 的直接替代品。

RESTful HTTP API 意味着非常简单的PUT/GET/POST/DELETE操作,这些操作在 HTTP 和 Web 中很常见。  
下面显示了GET和PUT命令的示例。GET将列出npoulton帐户中的所有容器,而PUT将创建一个名为uber\_container的新容器:

获取<http://swift.nigelpoulton.com/v1/npoulton/>  
放置 [http://swift.nigelpoulton.com/v1/npoulton/uber\\_container](http://swift.nigelpoulton.com/v1/npoulton/uber_container)

这些示例有些过于简单,因为请求始终需要身份验证令牌。因此,创建容器的PUT命令可能看起来更像这样:

```
curl -X PUT -H "X-Auth-Token:AUTH_tk3...bc8c" http://swift.nigelpoulton.com/v1/  
AUTH_admin/uber_container
```

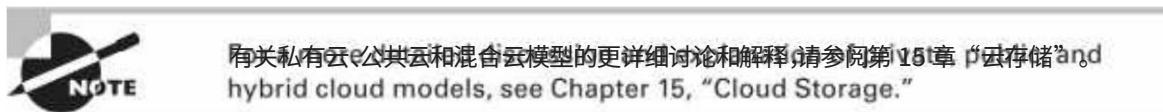
由于对象存储中的对象可以通过轻量级 HTTP Web 服务进行操作,因此每个对象都可以 (并且通常会) 拥有自己唯一的 URL,可以通过该 URL 进行访问。例如:

[http://swift.nigelpoulton.com/v1/nigelpoulton/uber\\_container/lily\\_in\\_a\\_tree.eps](http://swift.nigelpoulton.com/v1/nigelpoulton/uber_container/lily_in_a_tree.eps)

一些对象存储确实提供了 NAS 接口,允许它们通过网络并通过 NFS 和 CIFS 等进行访问。

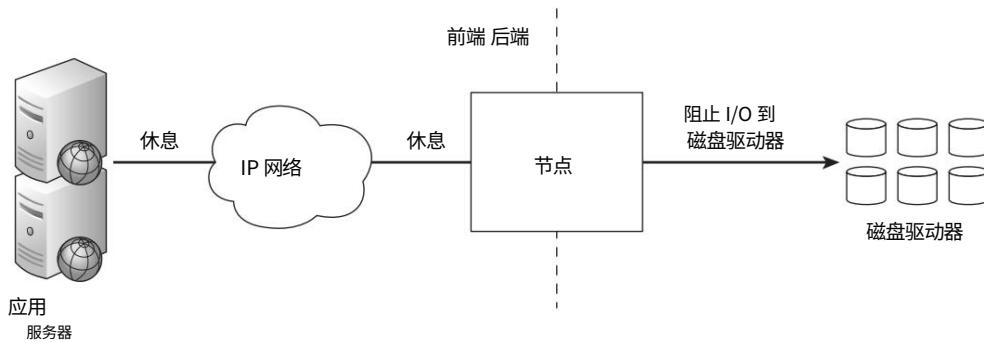
## 建筑学

对象存储的设计充分考虑了云的特点,包括大规模可扩展性、超低成本和相对较低的性能等因素。它还设计为与 REST 等标准 Web API 配合使用。



从架构上讲,对象存储设备非常简单。它们有一个前端和后端。前端是实现巧妙功能的地方。它实际上是使用 REST 和 SOAP API 的客户端与使用较低级别块存储协议的后端之间的代理。前端通常也处理大量元数据,并通过运行 TCP/IP 的以太网网络工作。还有一个用于节点到节点通信的后端专用网络。图 7.10 显示了一个简单的对象存储架构。

图 7.10 简单的对象存储架构



与通过高速 10 GB 和 40 GB 以太网 (有时是 InfiniBand) 运行后端的横向扩展 NAS 技术不同,对象存储中的私有网络既不高速也不低延迟。事实上,它们通常必须穿越 WAN 和互联网 因此,它们通常通过低速、高延迟链路运行,而不是高速和低延迟。

虽然命名空间可以划分为分区（有时称为容器或存储桶），但这些容器和存储桶不能嵌套。这与传统的文件系统有很大不同，在传统的文件系统中，目录嵌套在目录中。对象存储命名空间位于。对象存储的命名空间也是完全分布式的，即使在遍布全球的对象存储中也是如此。

## 公共云或私有云

对象存储相对较新，设计时就考虑到了云。许多公司没有构建自己的内部对象存储系统，而是选择使用基于云的对象存储服务，例如 Amazon S3、Windows Azure 和 Rackspace Cloud Files。这些基于云的存储服务（称为公共云）通常采用按需付费的定价模式，使公司可以避免购买自己的硬件和软件所涉及的前期资本成本，从而降低部署基于对象的存储所涉及的初始成本。

使用基于云的模型，您可以向提供商创建一个帐户，向他们发送您的数据

通过互联网或将载有数据的 USB 硬盘实际寄给他们，然后您就可以开始使用该服务了。从那时起，您通常按月付费，具体取决于您使用的容量和带宽。您可以通过云存储提供商支持的 API 或他们为您提供的控制面板类型的工具来访问和管理您的数据。一些云存储提供商还提供内容分发网络（CDN）提供商或与他们合作，以提供对数据的更快访问。CDN 通过将基于 Web 的服务器放置在世界各地的战略位置来缓存您的数据，从而加快对您数据的访问速度。例如，如果您的云存储提供商在美国和英国设有数据中心，但您的运营地点在香港，那么放置在香港附近并缓存您的数据集的 CDN 将极大地改善您的整体云存储体验。

公共云的主要替代方案是私有云，而 OpenStack 是一种流行的私有云选项。OpenStack 是一套云平台，包括计算、网络、存储和管理功能。OpenStack 的主要存储产品是 OpenStack 对象存储，通常以其代号 OpenStack Swift 来指代。OpenStack 软件最初是 Rackspace 和 NASA 的合资企业，旨在通过将基础设施服务作为位于廉价商品服务器、网络和存储硬件之上的软件来运行，从而降低运营 IT 基础设施的成本。

OpenStack 现已得到 HP、IBM、戴尔、Red Hat 和思科等许多传统大型 IT 公司的支持，并且正在获得大量希望不再从传统存储供应商处购买昂贵存储技术的公司的支持。

OpenStack Swift 是支持 Rackspace Cloud Files 的技术，也是许多其他可用公共云服务的基础。它具有广泛的 API 访问权限，并提供对象存储的所有主要功能，包括身份验证、容器、不同级别的冗余、加密等。

## 安全

与所有涉及 IP 网络（尤其是互联网）的事物一样，安全性至关重要。因此，您需要确保您选择使用的任何对象存储解决方案都要求所有操作都包含来自授权子系统的有效授权令牌。幸运的是，大多数都这样做。

## 对象存储选项

对象存储的例子有很多。有些是公共云，有些是私有云，有些提供混合模式。让我们简单看一下其中的一些。

### 亚马逊 S3

如果您认为 Amazon.com 是一家图书销售公司，那么您还生活在 90 年代，远远落后于时代。亚马逊是一家科技公司，也是一家庞大的基于云的数据存储公司。亚马逊简单存储服务 (S3) 可能是世界上最大、最知名的对象存储系统。Amazon S3 中存储了数万亿个对象。是的，数万亿！这意味着 Amazon S3 中的对象比我们认为的银河系中的星星还要多。话虽如此，S3 距离存储比地球上的沙粒还多的对象还有很长的路要走！

S3 的简单性是其受欢迎程度和广泛实施的关键。您可以立即注册并试用它，就像使用其他公共云对象存储一样。由于它基于 Web，因此只要您有互联网连接，您就可以在地球上的任何地方访问它并从中存储和检索数据。

您可以存储和检索互联网连接和银行余额允许的任意数量的数据。S3 使用公用设施式计费系统，这在大多数云服务中很常见。

您还可以对对象应用策略，以便将其删除或存档到更便宜、更慢的对象存档（称为 Amazon Glacier）中。

S3 将唯一 OID 称为键，并将对象存储在存储桶中。当您存储对象上传到 S3 后，它将在多个 Amazon 设施中存储和保护，直到您收到已成功存储的通知。存储和检索对象时还会执行基于网络的校验和，以确保网络上没有发生任何损坏。

### OpenStack Swift

Swift 是 OpenStack 的对象存储产品。Swift 可用于构建公共或私有对象存储。Swift 是一款开源软件，可让您在商用硬件之上构建自己的水平可扩展 OSD。它具有经过身份验证的 RESTful HTTP API 接口。

与大多数对象存储一样，Swift 可以使用常见的 Linux curl 命令进行操作。您可以从小处着手，将 Swift OSD 扩展到数百甚至数千个节点。

尽管是开源的,Swift 是一个成熟且功能丰富的对象存储系统。这些功能包括通过 REST 进行广泛的 API 访问,并绑定 Python、PHP、Java 等。它还提供批量导入、身份验证、加密和对象复制。

OpenStack 被定位为云的主要操作平台,它提供的不仅仅是 Swift。在存储领域,OpenStack 还提供用于块存储的 Cinder。除了存储之外,还有用于计算、网络、自动化等的组件。请访问 [www.openstack.org](http://www.openstack.org) 了解更多信息。

## 合规性档案和内容可寻址存储

许多组织都受政府和行业强制规定的约束,这些规定要求他们保留不可变的数据副本。不可变意味着一旦创建了数据,就必须能够保证它不能被更改。与数据不可变性相关的另一个行业术语是一次写入,多次读取 (WORM)。在 WORM 技术中,您可以一次写入一个对象并根据需要多次读取它,但您不能更新或更改该对象。WORM 的一个常见示例是 CD-ROM 技术。



数据保留监管要求的 **ex** 著名例子是 2002 年的《萨班斯-奥克斯利法案》(简称 SOX)。其他还包括 HIPAA 和 PCI。

对象存储的一个特定分支,称为内容可寻址存储 (CAS),作为合规性存档而广受欢迎。在 CAS 系统中,所有对象都使用哈希进行指纹识别,该哈希决定了对象在对象存储中的位置。对象的内容决定了它在存储中的位置,这一概念可能会改变许多人的思维方式,但它是 CAS 工作方式的核心。

当一个对象被放置在 CAS 对象存储中时,该对象的内容会被读取,然后通过散列系统运行以生成分配给该对象的指纹。

此指纹决定了对象在 CAS 存储中的位置。这是重要的部分:如果您更改该对象的任何部分,则将更改其指纹,这也会更改其在存储中的位置!将来再次读取对象时,将评估其哈希值,您将知道该对象是否已更改。

因此,CAS 在合规性归档方面非常自然,而且许多 CAS 系统旨在强制执行严格的数据不变性,并保证保留策略下的对象在保留期内不会被篡改。



This guarantees that the object has not changed since it was stored in the store—because its fingerprint and address have not changed in the store—because its fingerprint and address have not changed.

因此,据说 CAS 系统可以提供内容保证或内容真实性。

CAS 的另一项优势是重复数据删除。如果将新对象添加到存储中,如果某个对象与现有对象相同,它将获得相同的指纹和存储中的相同位置。发生这种情况时,无法创建该对象的第二个副本,并且该对象自然会被重复删除。虽然这是一种重复删除形式,但应该注意的是,它更正确地称为 le 级单实例,并且远不如块级重复删除那么强大。尽管如此,它仍然是 CAS 系统的一个不错的附带好处。

在考虑使用哈希函数的 CAS 系统时,使用强哈希函数(如 SHA-1)至关重要。如果使用弱哈希函数,则会增加冲突的风险,当两个不同的对象返回相同的哈希值时就会发生冲突。

发生这种情况时,CAS 系统会认为这两个对象是相同的,您会丢失数据!这是因为 CAS 系统认为它已经有了新对象的副本,因此不需要存储提交到存储的新对象。避免这种情况的唯一方法是使用强哈希函数来消除冲突的可能性,或者在每次发现哈希匹配时执行逐位比较。这两种方法都有相关的开销。强哈希函数比弱哈希函数的计算量更大,而逐位比较需要后端存储操作来读取现有对象。逐位比较的工作原理是将新提交的对象中的每个位与已存储的对象中的每个位进行比较。幸运的是,大多数现代 CAS 系统都采用强哈希(如 SHA-1),这些哈希极不可能产生冲突。

CAS 不擅长处理频繁更改的数据。不断更改大量频繁更新对象的指纹和存储位置的开销对 CAS 系统来说是一个真正的负担。

## 对象存储摘要

对象存储几乎完美适用于云级存储,用于存储不经常更改且不需要特别快速访问的数据。使用 NAS 和 SAN 等传统存储技术,存储用移动设备拍摄并上传到社交网站和私有云存储的图片和视频文件是不可能的。对象存储可以帮上忙!

对象存储还为长期数据存档提供了一个简洁的解决方案,尤其是符合  
ance 档案馆必须存储数据的不可变副本。

由于对象存储本身速度较慢,并且不适合高变化率的数据,因此它不会完全接管企业数据中心。但是,它们已经拥有了互联网。在企业数据中心,对象存储将在未来许多年内与 NAS 和 SAN 技术并驾齐驱。

在寻找大量非结构化数据的解决方案时,对象存储值得仔细研究。它们还提供了一个机会,可以摆脱传统的、基于供应商的、预先打包的现成产品,并有机会拥抱具有软件驱动智能的商品硬件。

## 章节概要

NAS 阵列 NAS 阵列是专门构建的存储系统,旨在通过共享 IP 网络提供共享文件服务。它们可用于整合多个通用文件服务器,例如 Windows 和 Linux 服务器。它们实现 NFS 和 SMB/CIFS 协议提供高性能、高可用性以及快照和远程复制等高级功能。

传统 NAS 阵列传统 NAS 阵列最适合中小型企业,但难以满足大型企业的非结构化数据需求。

它们提供不错的性能和高级数据服务,如快照、远程复制以及数据效率技术(如压缩和重复数据删除)。但它们的扩展性不佳。

横向扩展 NAS 阵列 横向扩展 NAS 阵列试图提供传统 NAS 阵列提供的所有功能,并具有巨大的可扩展性。它们仍然使用传统的文件服务协议,例如 NFS 和 SMB/CIFS。它们有助于避免大型环境中的 NAS 蔓延,非常适合大数据项目。

对象存储 对象存储是一种专为云级可扩展性而设计的新型存储系统。对象通过基于 Web 的 API(如 REST 和 SOAP)存储和检索到对象存储中。每个对象都可以用可搜索和索引的大量元数据进行标记。对象存储非常适合不经常更改且不需要高性能的丰富内容数据。它在公共云模型中很受欢迎。

内容感知存储 内容感知存储(CAS)是一种对象存储形式,非常适合用作合规性存档,以提供数据不变性,满足政府和监管标准。在 CAS 系统中,对象的内容决定了其在对象存储中的位置,对对象内容的任何更改都会改变其在存储中的位置。CAS 不太适合频繁更改的数据。

## 概括

在本章中,我们介绍了文件存储和对象存储的主题。我们首先学习了主要的文件服务协议(SMB 和 NFS)及其使用方法以及它们具有哪些安全功能。然后,我们了解了网络附加存储(NAS)阵列,将其与本地存储和 SAN 存储进行了比较,并解释了它们通常具有一些高级和企业级功能。我们还讨论了传统 NAS 阵列的一些可扩展性限制以及横向扩展 NAS 解决方案如何解决这一限制。然后,我们转到令人兴奋的对象存储主题,并解释了它与更传统的块和文件存储解决方案有何不同,以及这些差异如何使其成为大多数云存储需求的理想选择。我们讨论了对象存储架构及其主要通过 API 访问的方式,以及元数据在对象存储中的重要性。



# 章节 8



## 复制 技术

---

本章涵盖的主题：

- ✓ 同步复制
- ✓ 业务连续性
- ✓ 异步复制
- ✓ 基于阵列的复制
- ✓ 数据库日志传送
- ✓ 逻辑卷管理器快照
- ✓ 虚拟机管理程序复制和快照
- ✓ 多站点复制拓扑
- ✓ 基于阵列的快照
- ✓ 基于阵列的克隆



本章介绍复制和快照技术以及它们如何应用于业务连续性和灾难恢复规划和执行。您将了解远程复制技术的类型,例如同步和异步,以及每种技术对支持它们所需的网络链路的影响。您还将了解如何在堆栈的不同层(例如在存储阵列、虚拟机管理程序、卷管理器以及应用程序和数据库中)实现复制和快照技术。最后,您将了解如何确保应用程序和文件系统的一致性,以及一致性为何如此重要。

## 业务连续性

复制和快照技术是几乎所有业务连续性(BC)计划的基本组成部分。业务连续性就是在发生灾难时保持业务正常运行。由于我们使用灾难一词,因此业务连续性通常被称为灾难恢复(DR)。当我们使用灾难一词时,我们指的是影响平稳运行的业务IT系统的任何事件。被视为灾难的事件的常见示例包括:

- 系统故障和崩溃
- 电源故障
- 漏水
- 自然灾害
- 人为错误

例如,即将来临的飓风或大风暴将影响运行您实时业务应用程序的站点。当风暴来临时,您需要移动这些应用程序,让它们在不受风暴影响的其他站点上运行。

另一个例子是大型共享存储阵列发生故障(这种事情确实会发生),需要您将受影响的应用程序备份到另一个阵列(可能在另一个站点)。

所有这些事情都需要仔细的计划和大量的练习!



虽然存储领域确实很少出错,但是与网络世界相比,可以公平地说,当存储世界出现问题时,它们往往会出现大问题!尽管存储问题很少发生,但一旦发生,问题往往很大。

**Note!** Although storage issues are few and far between, when they happen they tend to be big.

我们必须明白,在当今世界,大多数企业都如此依赖技术,以至于说技术就是业务,这是对的。没有技术就没有业务!不幸的是,要让 C 级管理层相信这一点,事实往往非常困难,尽管当系统出现故障并且企业正在亏损时,他们会毫不犹豫地来对你大喊大叫,很难说服 C 级管理层这一点,即使系统已经宕机,企业正在亏损!

尽管 99.9% 的企业无需担心飓风,但许多企业都计划了需要他们在维护工作期间将应用服务转移到替代站点的活动。这些计划中的活动可能包括年度电力维护或其他维护和工程(M&E)工作,当维护工作正在进行时。这些计划中的活动可能包括年度电力维护或其他维护和工程(M&E)工作。

但业务连续性不仅仅是确保 IT 系统正常运行,它还确保 IT 系统是 up and running. It's all about ensuring that the business can still operate effectively during a disaster. For example, if people cannot access those services or if performance is so poor that the business cannot effectively operate.

本章重点介绍业务连续性的数据复制组件,并讨论卷快照。但是,业务连续性计划中的其他关键因素包括备份、工作场所恢复(总部停运时员工的办公桌)、在家办公、灾难期间的有效沟通、使服务器和应用程序恢复在线状态等。

recovery (desks for staff to work from if HQ is down), working from home, effective communications during the disaster, bringing servers and applications back online, and so on.

归根结底,每个部门的每个人都对企业及其客户负有责任,以确保在发生灾难时企业能够正常运转。customers of the business, to ensure that the business can operate in the event of a disaster. If your infrastructure fails, and you have no plan on how to get your business back up and running, then you will face losing your job, other people's jobs, and potentially the entire business. So, no pressure!

制定恢复业务运营的计划并不仅仅意味着在笔记本上写下一些想法。我们谈论的是适当的计划和适当的演练 (有时称为排练)。演练对于管理有效和可靠的业务连续性至关重要。测试您的计划,从未来按计划进行的事情中吸取教训。相信我,事情永远不会完全按照计划进行。然后,再次地测试!您绝对不想在严重程度为 S1 的灾难中第一次测试您的 BCs 计划! Run from the things that didn't go to plan—and trust me, things never go quite according to plan—and then test again and again! You absolutely do not want to be testing your BC plans for the first time in the heat of a severity 1 disaster!

了解应用程序的业务重要性也很重要。成熟的组织会将所有应用程序映射到优先级或层级。然后,在确定发生灾难时恢复应用程序的顺序时,这些优先级将成为决策过程的一部分。例如,所有优先级 1 的应用程序可能会首先恢复在线,并且您可能已经与管理层达成协议,您将在 2 小时内让所有优先级 1 的应用程序恢复并运行。优先级 3 的应用程序对业务不太重要,并且您可能已经与管理层达成协议,applications might be brought back online first, and you might have agreed with management that you will have all priority 1 applications back up and running within 2 hours. Priority 3 applications will be less important to the business, and you might have agreed with management

这些系统将在灾难发生后 10 小时内恢复运行。成熟的组织会定期演练其 BC 计划——这一点再怎么强调也不为过！



**不要误以为 IT 部门内部使用的系统不如优先级 1 关键任务、业务线应用程序在发生灾难时那么重要。**如果您无法登录恢复关键任务应用程序所需的系统，您恢复关键任务应用程序的机会将大大降低。允许员工远程登录和管理服务器以运行脚本和工具的 IT 系统（例如远程访问网关）非常重要，通常需要先启动它们，然后才能开始恢复业务应用程序。如果您无法登录，就无法运行脚本来启动业务应用程序。*Failed to recover your mission-critical applications. IT systems such as remote-access gateways that allow staff to log on remotely and management servers to run scripts and tools are important and often need to be up before you can even start recovering business applications. You can't run scripts to start business applications if you can't get logged on in the first place!*



一些经常被忽视的基础设施类型的应用程序包括电子邮件和企业聊天工具。这些工具在发生灾难时对于协调工作至关重要。在灾难期间，沟通往往是首要任务。如果每个人都朝着不同的方向努力，那么每个人都忙于从灾难中恢复是没有任何意义的。*Communication is often the number one priority. There is no point in everyone in IT busily working on recovering from the disaster if everyone is working in different directions.*

OK，这涵盖了背景和重要的基础知识。在您开始详细了解这章的基础技术之前，让我们快速浏览一下与业务连续性相关的那些语言 detail at the technologies underpinning it all, let's take a quick look at some of the language related to business continuity:

**恢复点目标 (RPO)** 这是服务恢复到的状态。This is the point in time to which a service is restored/恢复。换句话说，如果您的 RPO 为 15 分钟，则您需要能够将应用程序或服务恢复到灾难发生前 15 分钟内的状态。为此，您可以安排每 15 分钟进行一次备份。例如，在 8:30 进行的数据准备可以将数据库的状态恢复到 8:15 时（即 8:30 之前 15 分钟）的状态。但是如果灾难发生在 8:44，则备份无法将数据库恢复到 8:30 和 8:44 之间的状态。备份只能将应用程序恢复到 8:30 时的状态。即备份的准确时间。需要注意的是，备份至关重要，但通常不能提供最佳的 RPO。对于更精细的 RPO，频繁使用持续数据保护 (CDP) 并非快照通常是最好的选择。*If the backup (8:30 a.m.). However, if a disaster strikes at 8:44 a.m., the backup cannot recover the database to how it was between 8:30 and 8:44. The backup can recover the application only to the state it was in at exactly 8:30—the exact time the backup was taken. As a note, backups are vital, but do not normally provide the best RPOs. For a more granular RPO, frequent snapshots of continuous data protection (CDP) products are often a better choice.*

**恢复时间目标 (RTO)** 这指的是应用程序或服务恢复运行所需的时间。如果我们 8:44 遇到同样的灾难，并且您需要 2 小时才能将数据库恢复到灾难发生前 15 分钟的状态，那么您的 RTO 不应少于 2 小时。就 RTO 而言，这是您和业务经理预先商定的恢复应用程序或服务所需的时间段。因此，如果您知道恢复服务最多需要 1 小时，那么您应该同意更现实的 RTO，例如 3 小时。当然，如果遇到严重的意外灾难，并且必须恢复 20 个应用程序（每个应用程序的 RTO 为 2 小时），那么您可能无法在 2 小时内全部恢复它们。*Business manager pre-agree that it will take to recover an application or service. So if you know that at best it will take you 2 hours to recover a service, you should agree to a more realistic RTO of something like 3 hours. Of course, if you encounter a severe unexpected disaster and you have to recover 20 applications, each with an RTO of 2 hours, you will probably not be able to recover them all within the 2 hours.*

服务级别协议 (SLA) SLA 是您与客户或业务部门同意维持的服务级别,SLA 随处可见。您可能在高性能存储阵列上有一个 SLA,它将提供少于 30 毫秒的 I/O 响应时间。在业务连续性领域,SLA 往往包含 RPO 和 RTO。关于 SLA 和 RTO 需要注意的一点是,尽管您可能能够在 2 小时内恢复电子邮件、财务系统、交易或报告系统,但如果它们都受到同一灾难的影响,您可能无法在 2 小时内恢复所有系统。换句话说,您可以在 2 小时内恢复电子邮件、财务系统或交易系统,但您无法在同一个 2 小时内恢复电子邮件、财务和交易!对于业务管理而言,提前了解这些事情至关重要。

服务 谈到 BC 时,重要的是从服务或应用程序的角度来谈论,即服务器和其他技术资源的集合。例如,如果您尚未恢复 Web 服务器需要与之通信的任何必需后端数据库服务器,那么企业将不会关心您是否可以恢复所有前端 Web 服务器。应从为企业提供的服务的角度来考虑和执行保护和恢复。

调用 DR 调用 DR 通常意味着在远程站点启动应用程序或服务。例如,数据库服务器上的硬件出现问题导致数据库服务器启动和关闭,这可能会导致您为该应用程序调用 DR/

通过将所需的所有组件移至 DR 站点中的备用系统来恢复服务。可以为单个服务器、整个应用程序、整个服务甚至整个资产调用 DR。为整个资产调用 DR 通常绝非易事!

弹性降低 弹性降低,在 BC 术语中,通常指丧失调用 DR 的能力或丧失高可用性。丧失调用 DR 能力的一个例子是 DR 站点断电,这意味着如果您的主要生产站点出现问题,您将无法恢复该站点的服务。丧失高可用性的一个例子是丢失一个 SAN 结构;在正常操作情况下,您可能会运行双冗余结构,而丢失其中一个结构将降低您的弹性。关于弹性降低需要注意的是,您的业务应用程序通常不会受到影响;您只是面临更高的风险,即故障可能导致比您所能容忍的更长的停机时间。

高可用性 (HA) HA 是指 IT 以弹性方式设计和部署解决方案。存储领域的一个常见示例是部署双独立结构,每个服务器都连接到两个结构,这样如果一个结构发生故障,服务可以在幸存的结构上继续。只有当两个结构同时发生故障时,才需要调用 DR。HA 的主要目标是避免在绝对必要的情况下调用 DR。这是因为 HA 通常不需要任何服务中断。在双结构示例中,当一个结构发生故障时,另一个结构会立即接管,而不会造成任何服务中断。调用 DR 时通常不会出现这种情况。调用 DR 几乎总是会导致短暂的服务停机。



谈到业务连续性,如果您不定期测试业务连续性计划,则需要考虑很多事情,而且可能会错过很多内容。请定期测试。Don't test your business continuity plans regularly. Test

# 复制

复制是任何 BC 和 DR 策略的核心组成部分。然而,它只是其中一个组成部分。复制必须与有效的数据备份等相结合,才能形成可行的业务连续性策略。



*In this chapter, the term “replication” refers specifically to remote replication—将数据从主站点复制到远程辅助站点。有时这种远程复制技术被称为镜像。通过这种方式可以创建远程镜像。*

您可以使用远程复制将数据的副本保留在一个或多个远程站点。这些远程副本的两个主要用途如下:

- 业务连续性
- 报告和测试/开发

对于 BC,数据的远程副本通常可用于在发生系统崩溃、断电、漏水和自然灾害等中断时恢复应用程序和服务。对于报告和测试/开发用例,远程副本通常用作您针对其运行报告的数据的只读副本。

使用远程副本运行报告的一个优点是,施加在远程副本上的任何工作负载 (例如报告导致的大量读取工作负载) 都会被隔离到远程副本,并且不会对数据的实时生产副本的性能产生任何影响。

复制可以在堆栈的各个层上执行。以下是最常见的复制层:

- 应用/数据库层
- 主机层 (基于Linux世界的逻辑卷管理器)
- 存储层 (基于阵列)

你将会看到这些内容,但首先让我们看几个重要的概念通常与复制有关,无论它在堆栈中的什么位置实现。



*Although replication techniques can provide protection from natural disasters, they’re usually no good at protecting your data from logical corruption. Logical corruptions such as viruses will be faithfully replicated. If data is deleted on the original site, it will be faithfully replicated to your remote replicas, ensuring that data is wiped from your remote site, too.*

## 同步复制

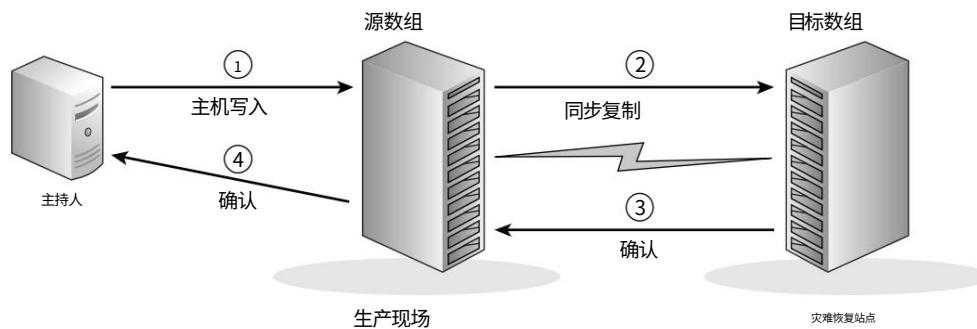
同步复制技术可确保远程副本与源数据副本保持同步,我们有时称之为零数据丢失。但要实现这一点,同步复制技术会对性能产生负面影响。原因是,在写入被视为完成之前,同步复制卷的所有写入都会提交给源和远程副本。这可确保源和目标始终保持同步,但这也意味着写入数据必须一直传输到远程副本,然后远程副本必须返回确认,写入才被视为完成。往返远程副本可能需要相当长的时间(几毫秒),并且取决于源和目标之间的距离。



**同步复制可实现零数据丢失,从而使 RPO 为零。**  
但是,副本的一致性可能需要您执行额外的工作才能使其达到一致状态。这些额外的工作可能会影响您与业务管理层商定的 RTO。

图 8.1 显示了基于阵列的同步远程复制的故事板。

图 8.1 同步复制情节提要



## 复制距离和延迟

复制延迟受多种因素影响,包括所涉及的技术和设备、途中必须经过的路由器数量以及源和目标之间的距离。关于延迟和距离的一个好的经验法则是假设源和目标之间每 100 英里的延迟约为 1 毫秒。当然,如果您在旅途中必须经过许多路由器,则延迟会增加,因为每个路由器都必须处理您的数据包。

可以使用ping命令(有点粗略地)测试网络延迟。以下ping命令显示从伦敦的一台机器到另一台机器的往返时间

在纽约,通过企业多协议标签交换 (MPLS) 网络进行连接。伦敦和纽约之间的距离约为 3500 英里,平均往返时间 (RTT) 为 77 毫秒 往返约 38.5 毫秒,再返回 38.5 毫秒。假设每 100 英里延迟 1 毫秒,那么伦敦系统和纽约系统之间的距离为 3850 英里。计算结果为  $77 \text{ 毫秒} / 2 = 38.5 \text{ 毫秒}$  (因为 ping)

命令告诉我们数据包从伦敦到纽约再返回所需的时间。然后我们取 38.5 毫秒并乘以 100 将毫秒转换为英里,得到 3,850 英里。

以下输出中显示的名称和 IP 地址已被修改,因此它们不代表真实的名称和 IP 地址:

```
C:\users\nigelpoulton>ping nydc01
```

正在 Ping nydc01.company-internal-network.com [10.11.12.13],数据为 32 字节:

```
来自 10.11.12.13 的回复:字节=32 时间=77ms TTL=52
来自 10.11.12.13 的回复:字节=32 时间=76ms TTL=52
来自 10.11.12.13 的回复:字节=32 时间=77ms TTL=52
来自 10.11.12.13 的回复:字节=32 时间=77ms TTL=52
```

10.11.12.13 的 Ping 统计数据:

```
数据包:已发送 = 4,已接收 = 4,丢失 = 0 (0% 丢失) ,
近似往返时间 (以毫秒为单位) :
    最小值 = 76 毫秒,最大值 = 77 毫秒,平均值 = 77 毫秒
```

一般来说,100 英里是同步复制技术覆盖距离的上限。最好少于 100 英里。无论如何,在做出此类决策时,需要考虑的关键群体如下:

- 存储供应商
- 应用程序供应商
- 网络团队

您的存储供应商和应用程序供应商将拥有配置和使用复制的最佳实践,您肯定会希望遵守他们的最佳实践。您的网络应该能够提供有关复制流量将通过的网络的准确数据。不要试图独自完成所有工作并希望获得最好的结果。

做好你的研究并与合适的人交谈!

## 复制链接注意事项

对于复制而言,复制流量的传输网络至关重要。网络连接不稳定,时断时续,会对复制解决方案产生重大影响。

同步复制解决方案往往最容易受到网络链路故障的影响。对于例如,在极端情况下,不可靠的网络链接会阻止应用程序更新。如果正在复制的卷配置为在复制链接断开时阻止写入,则可能出现这种情况;您可能拥有某些应用程序和用例,其中源卷和目标卷不同步是不可接受的。在现实世界中,99.9%的企业选择在网络中断时不停止写入。但是,如果您在复制链接断开时不停止对源卷的写入,则目标卷将很快不同步。并且,如果您对这些卷的 SLA 规定它们的 RPO 为零(意味着您可以将应用程序或服务恢复到中断的准确时间),那么您将违反 SLA。

但是同步复制的问题可能是由链接中断以外的其他因素引起的。慢速链接或大量丢包的链接可能会减慢对源卷的写入响应,因为对卷的新写入必须复制到远程系统,然后才能向主机/应用程序确认完成。复制过程中的任何延迟都会直接影响系统的性能。



同步复制配置始终需要强大而可靠的网络链接。但是,如果您恰好有应用程序和用例需要在复制链接断开时隔离(B(停止写入)卷,那么支持复制链接的网络电路是否强大就显得更为重要。在这些情况下,可能需要高可用性配置中的多个网络链接。

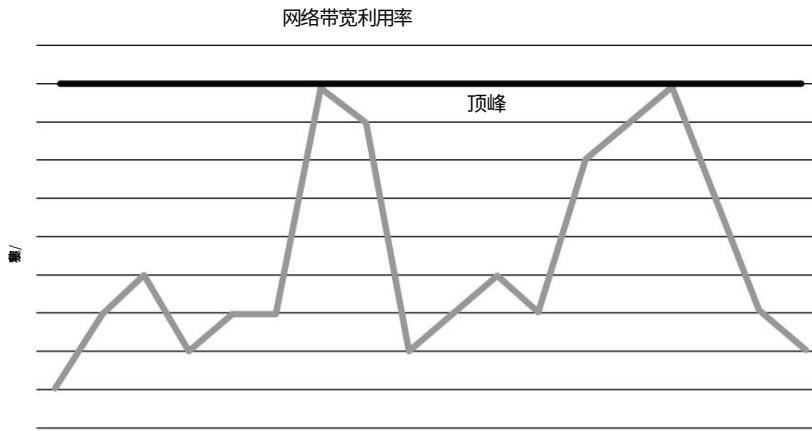
由于复制网络在同步复制环境中非常重要,因此在源站点和目标站点之间部署多个不同的复制网络是很常见的。根据所使用的技术,这些不同的复制网络(通常通过不同的电信提供商的不同路由)可以以主动/主动或主动/备用配置。设计原则是,如果一个复制网络发生故障,另一个可以以最小的中断处理复制流量。

复制网络(有时称为网络链路)的大小在同步复制配置中也至关重要。如果大小选择错误,您要么付出高昂的代价,要么陷入无法满足 RPO SLA 的尴尬境地。同步复制配置的一个重要设计原则是确定网络带宽的大小以适应峰值流量(包括突发流量)。例如,如果在高流量突发期间复制网络饱和,并且复制过程出现延迟,则在源卷等待目标阵列(复制链路的另一端)确认写入事务完成时,每个写入事务都会增加额外的延迟。

图 8.2 显示了网络带宽利用率。如果此链路支持同步复制配置,需要调整其大小以应对流量的峰值。

另一个需要注意的要点是,尤其是在共享 IP 网络上,复制流量会消耗带宽并降低使用共享网络的其他应用程序的性能。确保与网络人员合作,实施监管或其他形式的网络带宽限制。

图 8.2 同步远程复制带宽要求



## 异步复制

在许多设计方面,异步复制与同步复制相反。虽然同步复制提供零数据丢失 (RPO 为零),但会增加写入事务的延迟,但异步复制不会增加写入事务的延迟,但如果您需要从远程副本恢复服务,则几乎是会丢失数据。

异步复制不会给写入事务增加任何开销,因为所有写入事务在提交到源卷后都会立即发出完成信号。无需等待事务通过复制网络发送到目标系统,然后再发回完成信号。在事务提交到源后,通过复制网络将事务复制到目标的过程是异步执行的。当然,虽然这消除了同步远程复制的性能损失,但它带来了数据丢失的可能性。

之所以会出现这种数据丢失的可能性,是因为目标卷几乎总是落后于源卷。具体落后多少取决于所涉及的技术。在某些技术中,目标卷落后源卷 15 分钟或更长时间的情况并不少见 - 这意味着如果您丢失了源卷并且必须从目标卷恢复服务,您可能会丢失最多 15 分钟的数据。

异步复制技术的另一个优点是,它们允许源卷和目标卷 (或阵列)之间的距离比同步复制技术大得多。这是因为源和目标之间的 RTT 不那么重要。毕竟,写入性能不再受复制网络延迟的影响。理论上,源卷和目标卷之间的距离几乎是无限的。

然而在实践中,许多技术在超过 3,000 英里的距离内都难以发挥作用。这意味着您可能很难将数据从澳大利亚悉尼的办公室复制到纽约的数据中心 (距离约 10,000 英里)。在这种情况下,更常见的是,在亚太地区设立一个数据中心,作为亚太地区所有办公室复制数据的区域中心。

异步复制也无需指定网络连接  
配置以应对峰值需求。因此，异步复制的网络链路可以更便宜，从而节省大量成本。

当然，只有当您的服务、应用程序和用户能够应对灾难发生时丢失数据的可能性时，所有这些才有用。



术语 *DR*（灾难恢复）是指使用其复制的存储启动应用程序。该过程通常涉及将整个应用程序故障转移到远程备援系统。该过程通常具有破坏性，需要很多人参与，并且如果有大量应用程序和服务需要故障转移到 DR，则可能需要长达数小时的时间。

关于同步和异步复制的内容到此就够了。让我们从应用层开始，看看如何在堆栈的不同层上执行复制。图 8.3 显示了存储技术堆栈的一些组件，其中应用层位于顶部。当我们讨论如何在堆栈的不同层上执行复制时，我们将从顶部的应用层开始，然后沿着堆栈向下到较低的层。

图 8.3 存储堆栈的一些主要层



## 应用层复制

应用程序位于存储堆栈的顶部。它们是堆栈中的最高级别，高于逻辑卷管理器、虚拟机管理程序和存储阵列。

应用层复制在应用服务器上执行复制的所有繁重工作。无需将复制任务和复制管理转移到存储阵列。这项繁重的工作会消耗应用服务器的 CPU 和内存资源，并且

尽管这曾经被视为大忌,但最近 CPU 技术的进步以及当今大多数服务器中常见的大量内存意味着这种艰苦的复制工作通常可以由应用服务器轻松处理。

应用程序层复制的主要优势无疑是应用程序感知或应用程序智能。术语“应用程序感知复制”意味着复制技术与应用程序紧密集成,通常作为应用程序本身的组件。复制引擎和应用程序的这种紧密集成会生成处于应用程序一致状态的副本,这意味着这些副本非常适合快速有效地恢复应用程序。这些类型的应用程序一致性副本与应用程序不感知的副本(不在应用程序层执行或与应用程序没有任何集成的副本)形成鲜明对比,后者通常处于不一致的状态,并且可能需要额外的工作才能成功从中恢复应用程序。

应用层复制的常见示例包括 Oracle Data Guard 等技术以及 Microsoft 应用程序附带的本机复制技术(例如 Microsoft SQL Server 和 Microsoft Exchange Server)。

让我们看一下几种流行的基于应用程序的复制技术:

- Oracle 数据卫士
- Microsoft Exchange 服务器

顺便说一下,这两者都是通过 IP 进行复制的。

### Oracle 数据卫士

Oracle 数据库可以通过基于存储阵列的复制技术或 Oracle Data Guard(Oracle 自己的应用程序层远程保护套件)进行复制。这两种类型的复制在现实世界中都得到了广泛使用,但最近出现了一种趋势,即更多地使用 Data Guard,而不是基于阵列的复制。让我们快速了解一下 Data Guard。

Oracle Data Guard(有时缩写为 DG)是一种基于应用程序的远程复制技术。由于它是由 Oracle 专门为 Oracle 编写的,因此它具有许多基于阵列的复制所不具备的应用程序感知能力。最终结果是 Data Guard 副本在事务上是一致的。

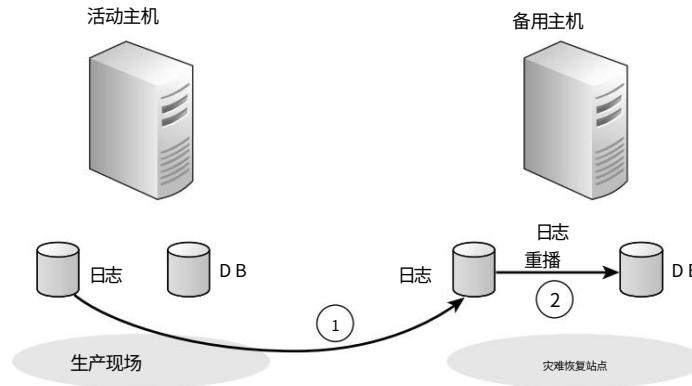
在 DG 配置中,生产数据库称为主数据库,数据库的每个远程灾难恢复副本称为备用数据库。主数据库可以有多个备用副本,但在 DG 配置中只能有一个主数据库。备用数据库可以且应该位于地理位置遥远的位置,这样如果托管主数据库的站点发生故障,备用数据库不会受到影响,并且可以转换为主数据库。

DG 是一种日志传送技术;主服务器上的重做日志被传送到备用数据库,它们应用于确保事务一致的备用数据库。图 8.4 显示了一个简单的日志传送配置。

应用层复制还可以比在堆栈较低层(如阵列)执行的复制更节省带宽。例如,在复制数据库时,基于阵列的复制技术必须将所有更改复制到数据库和日志。

文件卷。相比之下,基于日志传送技术的应用层复制仅复制对日志文件的更改。应用层复制还可以比基于快照技术的基于阵列的复制更节省带宽。基于快照的复制技术往往以范围为单位进行复制,这可能远远大于它们需要复制的更改数据。例如,对 1 MB 范围进行 4 KB 更改将需要复制整个 1 MB 范围。

图 8.4 基于日志传送应用程序的复制



#### Data Guard 支持两种类型的备用数据库：

**物理** 物理备用数据库确保是主数据库的精确物理副本。这包括相同的磁盘块布局和相同的数据库配置（包括逐行精确性）,这意味着备用数据库将具有与主数据库相同的 ROWID 和其他信息。这种类型的备用数据库通过从主数据库发送重做日志并将其作为重做事务直接应用于备用数据库来工作。在 DG 配置中,物理备用数据库可以以只读方式安装。

**逻辑** 逻辑备用数据库包含与主数据库完全相同的数据,但磁盘结构和数据库布局会有所不同。这是因为重做日志被转换为 SQL 语句,然后针对备用数据库执行。这种差异使逻辑备用数据库比物理备用数据库更灵活,从而使其不仅可用于 DR 和只读 (R/O) 访问。

两种备用数据库类型都保证事务一致性。并且,在提交复制数据之前,它们都会在备用数据库上执行数据完整性检查。

Oracle Data Guard 还允许切换和故障转移。切换是手动将备用数据库从备用角色转换为主角色的过程。

切换通常与计划维护任务相关,可保证零数据丢失。

故障转移通常是在主数据库发生故障时自动发生的进程。故障转移也可以配置为保证零数据丢失。

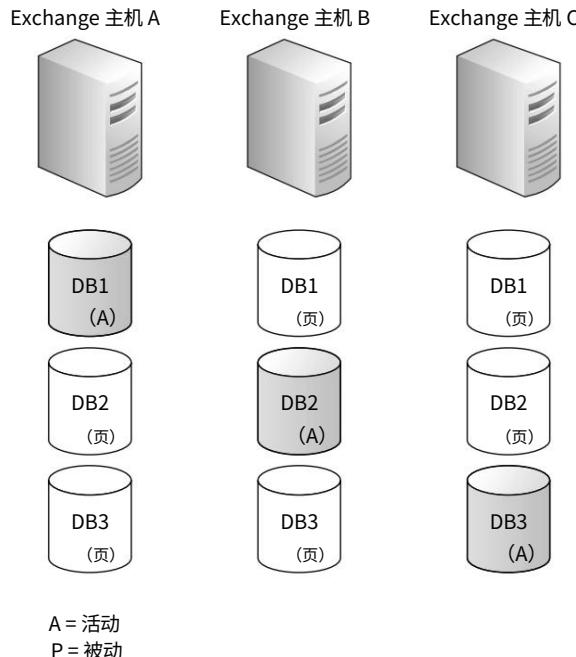
## Microsoft Exchange 服务器复制

Microsoft Exchange Server 是另一个具有自己的本机复制引擎的流行应用程序。您可以进行基于阵列的复制,也可以进行 Exchange 本机复制。选择权在您手中。最近,人们真正倾向于使用 Exchange 本机复制而不是基于阵列的复制。两者都有效,但 Exchange 自己的应用程序层复制往往更便宜,并且具有更多的应用程序感知能力。在故障转移时,您也不需要让存储团队参与。不过,缺点是它不支持同步复制。

与 Oracle Data Guard 和许多其他与数据库相关的应用程序层复制技术一样,Exchange 复制基于主动/被动架构,允许您拥有一个数据库的多个备用副本。它还利用日志传送;您将事务日志传送到远程 Exchange 服务器,然后应用它们。与大多数基于应用程序的复制技术一样,它支持切换和故障转移。

从 Exchange 2010 开始,复制的基本单位是数据可用性组(简称 DAG)。DAG 是托管邮箱数据库的主动和备用副本的 Exchange 服务器的集合,如图 8.5 所示。

图 8.5 Microsoft Exchange 数据可用性组



在图 8.5 中,DAG 中的任何服务器都可以托管来自 DAG 中任何其他服务器的邮箱数据库副本。DAG 中的每台服务器都必须拥有配置为活动或备用的任何邮箱数据库的自己的副本。不允许在共享存储上拥有多个邮箱服务器可见和共享的数据库的单个副本!

在 DAG 中,恢复是在数据库级别执行的,并且可以配置为自动执行。由于恢复是在数据库级别执行的,因此可以对单个邮箱数据库进行故障转移或切换,而不必在服务器级别进行故障转移或切换(这会影响该服务器上的所有数据库)。

此外,Exchange 复制在后台使用 Microsoft 故障转移群集  
从日常运营来看,这一点并不明显。



With the introduction of DAG in Microsoft Exchange Server 2010, replication no longer goes through TCP port 445 or SMB. This makes it easier for those concerned about security to work in environments where data is sensitive. Today, no organization can afford to compromise on security.

Exchange Server 2010 和 2013 都提供了同步复制 API,第三方存储供应商可以利用该 API 为 Exchange 2010 和 2013 DAG 提供同步复制功能。它们有效地用自己的基于阵列的同步块复制取代了本机日志传送复制引擎。

还有许多其他基于数据库和应用程序的复制技术,但这里提到的两个产品 Oracle Data Guard 和 Microsoft Exchange Server 很好地展示了它们的工作原理。它们的优势包括以下几个关键因素:

- 成本
- 应用程序一致性
- 简单

## 基于逻辑卷管理器的复制

卷管理器在技术堆栈中的位置比应用程序更靠下,但比虚拟机管理程序和存储阵列的位置更高。

基于主机的卷管理器(如 Linux LVM)能够使用长距离 LVM 镜像进行远程复制。在 Linux LVM 方面,大多数人仍然认为这既是一种穷人的复制技术,也对生产环境有一定的风险。这对于实验室和开发环境来说是必要的,但它不是一项可以押注业务的技术。话虽如此,它还是很有用的,而且就 Linux LVM 而言,它是免费的。



由于 LVM 不了解底层存储技术,许多公司都利用 LVM 镜像来执行存储迁移,即服务器的存储从一个供应商更改为另一个供应商,而主机保持不变。有些迁移是在服务器运行但应用程序和数据库关闭时执行的。然而,有时 LVM 镜像已成功用于将使用率较低的数据库迁移到新的存储阵列。后者应极其小心地进行,因为没有人能保证它会成功。

在 Windows 世界中,通常使用在卷管理层之上实现为过滤驱动程序的第三方产品来提供基于服务器的复制。

一个常见的例子,并且已经存在了一段时间了,就是 Double-Take。

## 基于虚拟机管理程序的复制

在技术堆栈中,虚拟机管理程序的位置比应用程序和逻辑卷管理器更靠下,但比存储阵列的位置更高。

随着虚拟机管理程序技术的成熟,它们开始提供越来越多的高级功能。其中一项功能就是复制。尽管有时被视为穷人的复制选择,但基于虚拟机管理程序的复制技术正在快速成熟,并且比基于阵列的复制技术具有一些潜在优势 - 更不用说更低的成本了。

到目前为止,中小型企业 (SMB) 对基于虚拟机管理程序的复制技术的兴趣最浓厚,但即使您的公司规模不仅限于 SMB,也绝对值得一看。VMware vSphere Replication 是目前最先进的虚拟机管理程序复制技术,因此我们以它为例进行分析。

### vSphere 复制

VMware vSphere 现在提供原生虚拟机管理程序级复制引擎,可供 Site Recovery Manager (SRM) 使用。它被称为 vSphere Replication。

vSphere Replication 的优点包括降低成本和简化

配置。这两个因素在 99% 的 IT 环境中都很重要。

通过无需购买昂贵的基于阵列的复制许可证,可以降低成本,从而有效地将 SRM 复制的高级功能带给更多人。vSphere Replication 还通过允许存储平台独立性并减少存储层上任何潜在的供应商锁定,为存储层带来了极大的灵活性。基本上,您可以在底层使用任何存储,并且您选择的存储技术不必在源和目标上相同。



**谨慎使用任何存储,不要费力匹配源和目标的存储。虽然它可以降低成本并允许更灵活的 h- 复制配置,但不匹配源和目标存储可能会增加配置的一些风险。这些风险包括可用性和性能,如果在发生灾难时您不得不在性能较低的存储上运行业务,则可能会出现性能不佳的情况。**

vSphere Replication 是一种基于 IP 的复制技术,可在虚拟机磁盘 (VMDK) 级别运行。在 VMDK 级别运行可让您以非常精细的级别配置复制 - 甚至比虚拟机级别还要精细。例如,假设您有一台具有四个虚拟磁盘的虚拟机,如下所示:

VMDK 1: 操作系统分区

VMDK 2: Page-le 分区

VMDK 3:数据库分区

VMDK 4:临时分区

现在想象一下,你的数据中心之间有一条细长的网络管道,并且想要

确保只复制发生灾难时需要的数据。您可以选择只复制 VMDK 1 和 VMDK 3,而不复制页面文件和临时 VMDK 文件。这可以显著减轻站点之间复制网络的负载。

在效率方面,vSphere Replication 足够智能,可以只复制每个 VMDK 的更改块,使用与用于增量备份的更改块跟踪(CBT)引擎类似的技术。

vSphere Replication 还存在于 ESX 堆栈中的存储层之上,这意味着您可以自由使用任何您想要的存储作为底层存储 - 只要它存在于 VMware 的硬件兼容性列表 (HCL) 中。这与使用基于阵列的复制技术相反,后者通常带有关于哪些底层存储平台可以相互复制的严格规则。(这些规则通常会限制源和目标使用相同的供应商和相同的型号。)例如,在 vSphere Replication 配置中,您的 VM 主副本可能在来自 Dell EqualLogic iSCSI 存储阵列的 iSCSI LUN 上运行,但您的远程离线 VM 可能位于本地磁盘上的数据存储中!

vSphere Replication 将已启动的虚拟机 (如果主虚拟机已关闭,则无需复制)从一个 ESXi 主机复制到另一个 ESXi 主机。这些 ESXi 主机可以位于同一站点的同一群集中,也可以位于远程站点的不同群集中。此外,默认情况下会安装基于 ESXi 主机的代理,这意味着无需基于虚拟机的代理即可使用 vSphere Replication。vSphere Replication 还具有 Microsoft 前端应用程序感知功能,可对 Microsoft Exchange Server 和 Microsoft SQL Server 等 Microsoft 卷影复制服务 (VSS) 进行 API 调用,确保远程副本一致且可以安全轻松地恢复。

如您在练习 8.1 中看到的,标准 vCenter 工具用于配置 vSphere Replication 以及 VM 恢复。RPO 也可以配置为最短 15 分钟到最长 24 小时之间的任何时间。

### 练习 8.1

#### 配置 vSphere Replication

本练习重点介绍通过 vSphere Web Client 配置 vSphere Replication (VR) 所需的步骤。作为先决条件,本示例假设您已下载并正在运行 vSphere Replication Manager (VRM) 虚拟设备,并在 VMkernel NIC 上启用了 vSphere Replication 流量。

1.在 vCenter Web Client 中,找到要复制的已启动的虚拟机。

右键单击虚拟机并选择所有 vSphere Replication 操作配置复制。

### 练习 8.1 (续)

只需在 vCenter 中配置一个站点即可配置 vSphere Replication。

显然,这种复制无法保护您免受站点故障的影响,但它非常适合在实验室中练习或防止数据存储丢失。

2.选择您想要复制到的站点并单击下一步。

3.选择要复制到的目标数据存储 (如果您只有一个站点,请确保目标数据存储与您的虚拟机当前所在的数据存储不同)。

点击下一步。

4.使用滑块选择 15 分钟到 24 小时之间的 RPO。越小

通常更好。

5.如果您的客户操作系统支持,您还可以选择让虚拟机静止。

6.单击下一步。

7.检查您的选择并单击“完成”。

vSphere 现在将进行必要的更改 (例如将 vSCSI 驱动程序注入客户机以管理更改块跟踪),然后执行初始完全同步。初始完全同步完成后,将仅复制更改的块。

---

在当前版本中,您可以使用 vSphere Replication 配置的最小 RPO 为 15 分钟。这对于大多数用例来说都是合适的,但并非全部。

尽管 vSphere Replication 具有许多吸引人的功能,但它是基于主机的,因此会消耗基于主机的 CPU 和内存。这并非总是世界末日,但在一个要求在拥有大量服务器整合的平台上高效利用资源的世界里,它可能不是最有效的解决方案。如果您已经在存储阵列技术方面进行了投资,情况尤其如此。

## 将 SRM 与基于存储阵列的复制相集成

VMware Site Recovery Manager (SRM) 可以 (并且经常) 与基于阵列的复制技术集成,从而提供企业级业务连续性解决方案。

这种配置允许通过常见的 VMware 工具进行复制和站点恢复,但具有坚如磐石的基于阵列的复制引擎的强大功能,这些引擎可以减轻虚拟机管理程序的所有复制开销。

仅受支持的存储阵列通过 vCenter 存储复制与 VMware SRM 配合使用  
适配器 (SRA) 插件允许 vCenter 和存储阵列相互理解和通信。

## 基于阵列的复制

就复制技术而言,基于阵列的复制技术位于堆栈的底部,低于虚拟机管理程序、逻辑卷管理器和应用程序。

在基于阵列的复制技术中,存储阵列会完成所有繁重的复制工作,从而释放主机资源来为应用程序提供服务。但是,普通的基于阵列的复制技术(没有特殊软件使其能够感知应用程序)并不了解应用程序。因此,我们认为基于阵列的复制不够智能,或者更重要的是,它创建的副本卷通常最多只能达到崩溃一致性。这意味着远程副本卷中的应用程序数据状态对于应用程序的恢复来说并不理想,并且可能需要额外的工作才能从副本数据中恢复应用程序。这是基于阵列的复制逐渐失去流行度并经常被基于应用程序的复制技术取代的主要原因之一。



可以将基于阵列的复制技术与应用程序集成。Microsoft 在 Exchange Server 2010 和 2013 中提供的同步复制 API 就是一个例子,它允许存储阵列供应商将其复制技术与 Exchange Server 集成,从而有效地用存储阵列中的同步基于块的复制引擎替换本机 Exchange Server 日志传送应用程序层复制引擎。这提供了两全其美的效果:应用程序感知以及将复制开销卸载到存储阵列。

尽管如此,基于阵列的复制仍然很可靠并且非常流行,而且它可以  
通过 API 与应用程序和管理程序集成。

存储阵列倾向于采用以下两种方式之一实现异步复制:

- 基于快照
- 基于期刊

让我们看一下这两者。

## 基于异步快照的复制

许多存储阵列在快照技术之上实现异步复制。

在这种设置中,源阵列会定期对源卷进行快照:

根据您配置的计划,然后通过复制网络将快照中的数据发送到目标阵列,然后将其应用于副  
本卷。显然,为了做到这一点,需要将源卷的初始副本复制到目标系统,然后从此时起,将源卷的  
快照发送到目标系统。

由于您配置了计划,因此您可以将其与您的 SLA 和 RPO 相匹配。因此,例如,如果您同意  
某个特定卷的 RPO 为 10 分钟,则可以配置该卷的复制计划,每 5 分钟拍摄一次快照,并每 5  
分钟将快照数据复制到目标卷。这将为快照数据通过复制网络传输并应用于副本卷提供充足  
的时间,并让您轻松满足该特定 SLA 的 10 分钟 RPO。

需要注意的是,如果您已同意 10 分钟的 RPO,则配置 10 分钟的复制间隔不足以实现 10 分钟的 RPO。毕竟,快照中包含的数据必须通过复制网络发送,然后应用于远程站点,所有这些都需要额外的时间。因此,某些存储阵列允许您为卷指定 RPO,然后阵列会配置适当的复制间隔,以便可以安全地实现 RPO。这种方法还为阵列提供了优先复制接近其 RPO 阈值的卷的可能性,而不是那些未接近其 RPO 阈值的卷。



如果阵列的异步复制引擎基于快照技术,那么了解用于复制的快照是否计入系统支持的最大快照数也很重要。例如,如果阵列最多支持 256 个快照并且您有 32 个复制卷,每个卷在复制设置中维护两个并发快照,则这 64 个基于复制的快照可能会计入系统的 256 个快照。在配置基于快照驱动的复制的解决方案时,请仔细检查这一点。本章末尾对基于阵列的快照的讨论将进一步介绍这一点。



在使用基于快照技术构建的异步复制解决方案时,请确保您了解底层快照扩展大小,因为它会对网络带宽利用率产生重大影响。例如,较大的扩展大小(例如 1 MB)每次更新该扩展中的一个字节时都必须通过复制网络复制 1 MB 的数据。另一方面,较小的扩展大小(例如 64 KB)每次更新扩展中的一个字节时只需通过复制网络复制 64 KB。因此,在复制快照扩展大小方面,越小越好。

基于快照的复制的一个好处是,只要您的阵列具有良好的快照技术,它就会合并写入,这意味着如果您的应用程序自上次复制间隔以来已对同一数据块更新了一千次,则只有该数据块的最新内容将与下一组增量(用于指代已更改的块的术语)一起通过复制网络发送,而不是所有一千次更新。

### 基于日志的异步复制

另一种流行的实现基于阵列的异步复制的方法是使用日志卷。基于日志的异步复制技术将新写入内容缓冲到缓存或专用卷(称为日志卷或写入意图日志),然后再通过复制网络异步将其发送到目标阵列。

让我们仔细看看它的内部工作原理。

在基于日志的异步复制配置中,当新的写入数据到达阵列,它遵循正常程序,即被保护到镜像缓存,然后向主机/应用程序发出完成信号。但是,作为此过程的一部分,写入 I/O 被标记为复制卷的一部分,以确保它在下一个复制间隔到达时被复制到日志卷并通过复制网络复制到远程目标阵列。复制间隔之间的间隔完全取决于您的阵列复制技术以及您如何配置它。但是,它通常是每隔几秒钟,这意味着基于日志的异步复制解决方案提供的 RPO 类型通常低于基于快照的复制配置提供的 RPO 类型 -

其往往提供最短 5 分钟的 RPO。

基于日志的异步复制可以应用写入排序元数据,以便当日志卷的内容被复制到远程系统时,写入将按照提交到源卷的相同顺序提交到目标卷。

这有助于保持远程副本卷的一致性。但是,如果出于某种原因系统无法再将更新写入日志卷 (假设日志卷已满或发生故障),系统将恢复为维护已更改块的位图。这种操作模式称为位图,它允许您跟踪源卷中已更改的块,以便仅将更改发送到远程阵列,但它不允许您维护写入顺序。



一些阵列在内存中实现异步复制日志,这种结构通常称为内存传输缓冲区或侧文件。*memory, in a construct often referred to as either a memory transit buffer or sidefile.* 显然,可用于复制日志的内存量是有限的,并且在复制网络长时间停机期间,这些系统通常会恢复使用日志卷,或者切换到不维护写入顺序的位映射模式。

实现方式因源阵列是否将更新推送到目标阵列或目标阵列是否提取更新而异。对于您的日常工作,您不必担心这些事情,当然不必像确保您满足 SLA 和 RPO 那样担心。

这就是基于日志的异步复制的工作原理。让我们看看其中的一些要点。

显然,这些期刊的容量足够大、性能足够好是至关重要的以满足缓冲复制流量的需求。在容量方面,当网络链路或远程阵列长时间处于关闭状态时,这一点尤其重要。与所有调整大小的练习一样,这是一个权衡:如果日志卷的大小过大,则资源利用不足并浪费金钱,但是如果日志卷的大小过小,当复制网络或远程阵列处于关闭状态时,它们可能会被填满,并导致复制的卷在问题解决后需要完全同步。完全同步不是您想要经常执行的操作。

例如,假设正在复制一个 2 TB 的卷。当复制网络出现故障时 (比如维护承包商在数据中心外挖掘道路时撞到了电缆),对源卷的所有更改都将缓冲到日志卷中,直到复制网络恢复。如果复制链接在之前已修复

日志卷已满,您需要复制到目标阵列的只是日志卷中包含的更改。但是,如果日志卷在网络链路中断期间已满,则复制的卷几乎肯定会被标记为失败,并需要您执行完全同步 - 将整个 2 TB 卷通过复制网络复制到远程阵列。这可能远不止日志卷中包含的更改。即使相距不到 100 英里的两个站点之间的带宽超过 8 Gbps,复制数 TB 的数据也可能需要数小时。

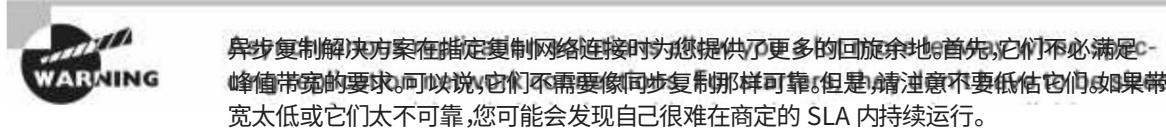
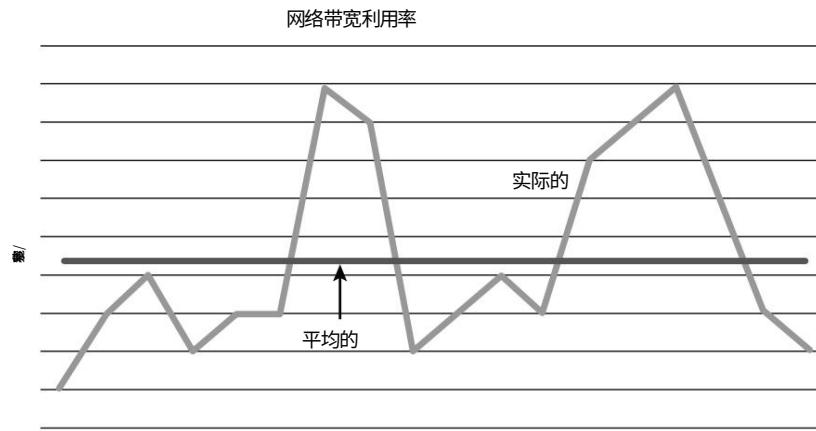


图 8.6 显示了异步复制的网络带宽要求。  
每小时捕获一次带宽利用率,包括平均线路。

图 8.6 异步复制的带宽要求



## 复制拓扑

这里介绍的复制拓扑不限于基于阵列的复制。但是,本节重点介绍如何使用基于阵列的复制实现不同的复制拓扑,因为最复杂的复制拓扑往往是围绕基于阵列的复制实现的。

不同的存储阵列支持不同的复制拓扑。并非每个阵列都支持但是,每个复制拓扑都存在问题。在尝试实施复杂的复制拓扑之前,请务必进行研究,因为您可能会发现自己处于不受支持的情况。这些不同的拓扑通常是同步和异步复制的混合,具有两个、三个甚至四个阵列,位于相距不同距离的多个站点。

让我们看一些复制拓扑的示例。

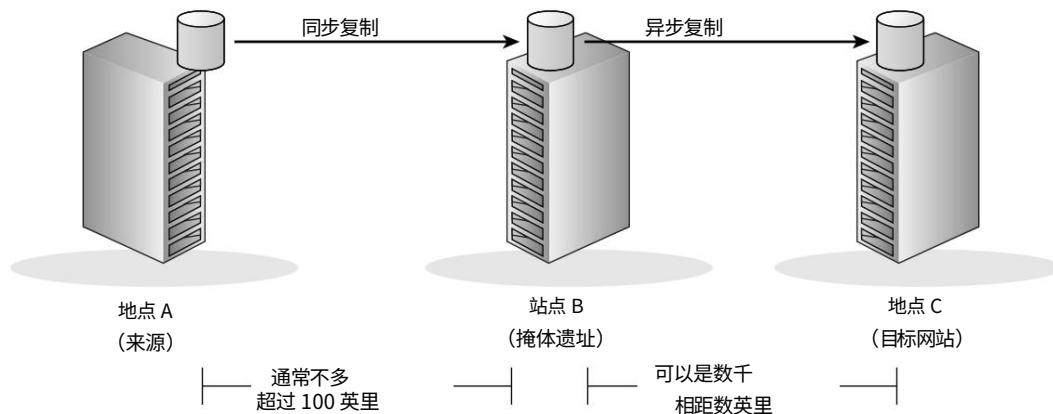


本节介绍的复制拓扑绝不是详尽无遗的。还有许多其他配置是可能的，其中大多数配置和维护起来都很复杂。  
Plenty of other configurations are possible. Most of

### 三站点级联

顾名思义，三站点级联基于三个站点：源站点、目标站点和两者之间的掩体站点。此配置如图 8.7 所示。

图 8.7 三站点级联拓扑



源站点和目标站点的阵列无法直接相互通信。任何想要从源站点的阵列复制到目标站点的阵列的数据都必须先转到堡垒站点的阵列。一个常见的例子是，源站点和堡垒站点相距不到 100 英里，并配置了同步复制。而目标站点则相距 100 多英里，并异步复制到目标站点。

由于目标站点通常距离源站点和掩体站点很远，因此这种拓扑结构非常适合高性能要求，具有较低的 RPO，但能够承受大规模本地灾难。花点时间让它适应一下。首先，该解决方案可以承受大规模本地灾难，因为目标站点可能距离源站点数千英里。它也可以具有高性能，因为源站点和掩体站点中的卷之间的同步复制可以足够接近，以至于对复制流量施加的网络延迟很低。并且 RPO 可以很低（实际上可以为零），因为同步复制提供零数据丢失。当然，如果您确实经历了大规模本地灾难并丢失了源站点和掩体站点，您将不得不在异步复制到的目标站点上恢复应用程序，因此您将落后于源站点，并且很有可能丢失一些数据。

三站点级联可让您的应用程序即使丢失源站点也能保持受保护。这是因为堡垒和目标站点之间仍有复制链接，这意味着这两个站点之间的复制仍可进行。在简单的双站点

复制模型,如果您丢失了源站点并将应用程序故障转移到目标站点,则目标站点将“独立运行”,直到您恢复源站点并将复制从目标站点重新配置回原始源站点。

三站点级联模型至少有一个主要缺点,那就是一旦堡垒站点丢失,源站点和目标站点之间的连接就会被切断。源站点的数据无法复制到目标站点,目标站点就会开始越来越落后。

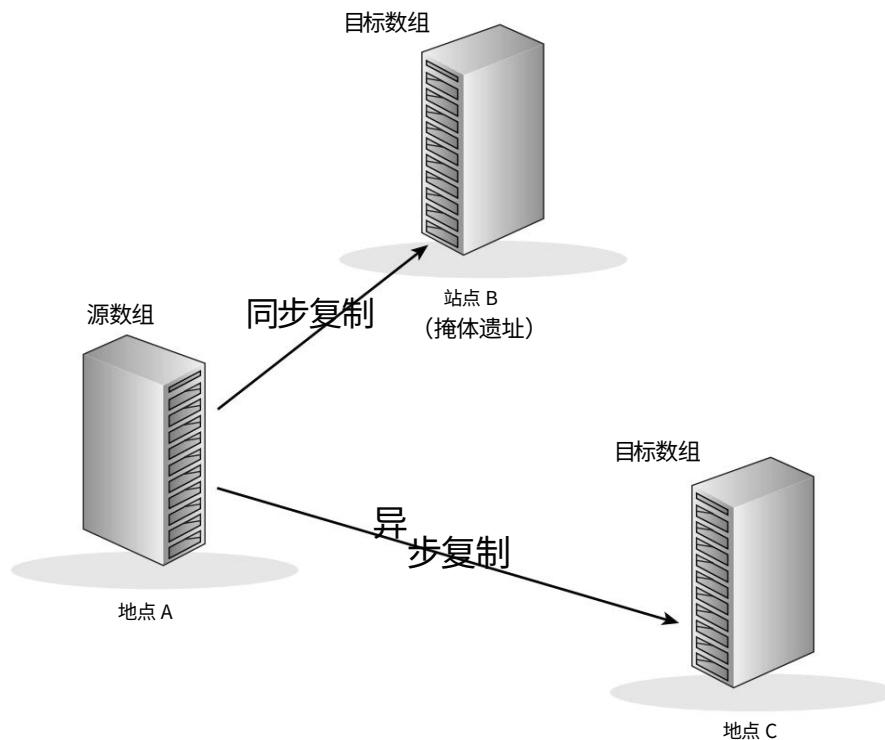
### 三站点多目标

三站点多目标将源阵列同时复制到两个目标阵列。

这种拓扑结构可能比三站点级联更具弹性,因为掩体站点不再是解决方案中的薄弱环节。源阵列可以看到配置中的其他两个阵列,并配置为复制到它们两个,而无需依赖掩体站点中的阵列。但是,三站点多目标的主要缺点是两个目标阵列之间通常没有连接。如果源站点丢失,则幸存的两个阵列无法相互复制。

如图 8.8 所示,在三站点多目标拓扑中,其中一个远程站点通常在同步复制距离内,而另一个则远得多。这种设置允许拓扑能够提供零数据丢失(通过本地同步复制链路),并且由于异步复制到远得多的阵列,因此能够安然度过大规模本地灾难。

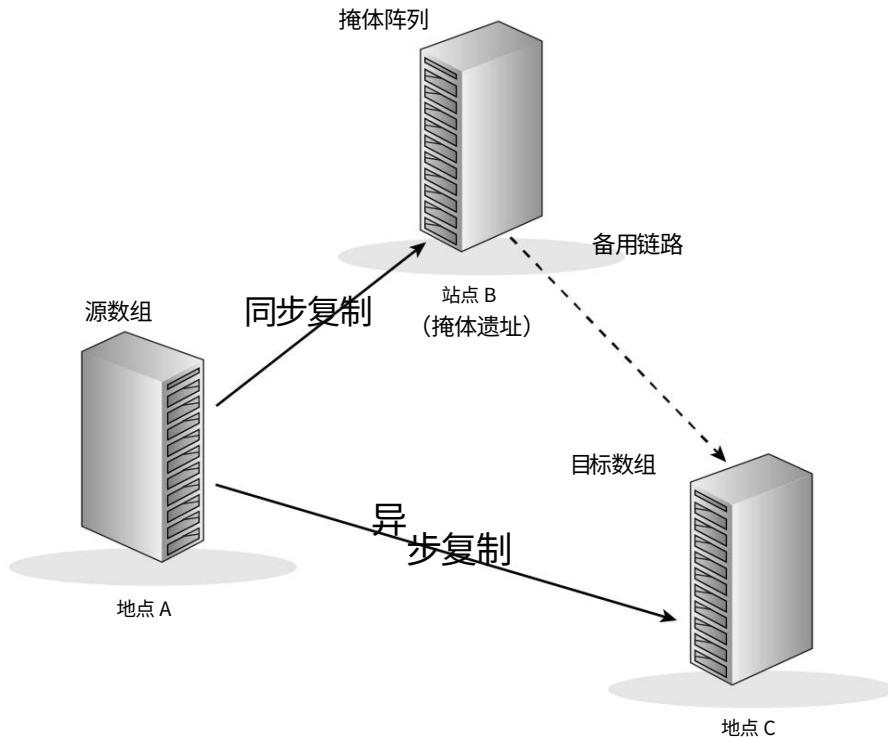
图 8.8 三站点多目标拓扑



### 三地点三角

如图 8.9 所示,三站点三角拓扑与三站点多目标几乎完全相同。唯一的区别是站点 B 和站点 C 中的两个目标阵列之间存在额外的复制链接。如图 8.9 所示,两个目标阵列之间的这个额外复制链接通常处于待机模式,这意味着在正常操作情况下,它不用于发送数据。但是,如果源站点或源阵列丢失,它可以用于复制数据。这有助于消除三站点多目标的主要弱点,即如果源阵列丢失,虽然您有两个幸存的阵列,但它们无法相互复制,您只剩下两份数据,但无法将两个幸存的阵列配置为灾难恢复对。

图 8.9 三站点三角复制拓扑



只要您的阵列的复制技术支持,那么在发生导致您丢失源阵列的灾难时,剩下的两个目标阵列完全可以相互通信,确定它们所持有的数据副本之间的差异,并且只需将差异部分发送给对方,而不需要完全同步。

## 本地快照

首先,让我们澄清一下术语。快照是数据的本地副本(副本)。本地意味着快照与源卷位于同一主机或存储阵列上,如果您无法访问源阵列,快照将无法用作恢复工具。如果您无法访问源阵列,则显然无法访问源卷和快照。

快照就是我们所说的系统卷的时间点(PIT)副本。它们提供了某一特定时间点(例如下午5点)的体积或系统状况的图像。这些快照可以用于很多事情,例如回滚点、备份源、用于执行测试的临时卷等等。快照非常灵活,在现实世界中被广泛使用。

无论如何,让我们回到术语上来。让我们准确定义一下我们使用术语“时间点”和“本地”。

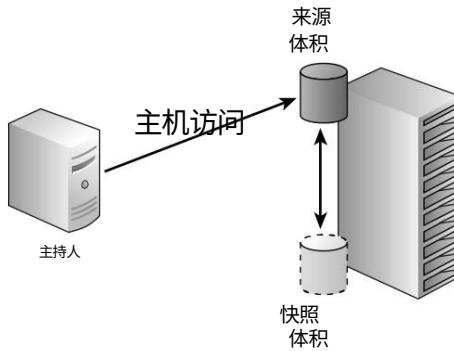
本地是指创建的副本与源卷存在于同一系统上。

如果是基于主机的快照,则快照与源卷位于同一主机上。如果是基于阵列的快照,则快照与源卷位于同一阵列上。

时间点快照是卷在某一特定时间的副本。例如,PIT快照可能是卷在早上9点的副本。这意味着它们不与源保持同步。这与远程副本完全相反,远程副本通常与源卷保持同步或半同步。

源卷和主卷这两个术语指的是实时生产卷,术语“源阵列”和“主阵列”是指托管源卷的实时生产阵列。图8.10显示了一个简单的快照配置,其中的一些术语映射到图表中。

图8.10 本地快照



根据您使用的快照技术,您将能够创建只读快照、读写快照,或者两者兼有。



关于快照需要注意的要点是它们不是备份!请务必牢记这点!快照与源卷位于同一系统上!如果丢失源卷或源系统,快照也会丢失!但是,如果您将数据复制到远程目标卷,然后对远程目标卷进行快照,则这些快照可以作为源卷的备份副本,因为它们位于单独的系统上,可能位于单独的站点中。如果丢失源系统或源站点,您仍然可以访问远程快照。

## 基于虚拟机管理程序的快照

现在,大多数现代虚拟机管理程序都提供本机快照技术。与基于虚拟机管理程序和主机的复制技术一样,这些快照会消耗基于主机的资源,并且不会从将快照过程转移到存储阵列中受益。尽管如此,它们正在不断改进,并且越来越受欢迎。

由于 VMware 是虚拟机管理程序领域的主要参与者,并且拥有相对成熟的集成快照堆栈,让我们仔细看看虚拟机快照在 VMware 环境中的工作原理。

### VMware 快照

VMware vSphere 提供本机虚拟机 (VM) 快照功能以及第三方供应商可以用来创建和操作 VM 快照的 API。

VM 快照是 PIT 快照,可保留 VM 当时的状态和数据快照拍摄的时间。例如,如果您在下午 5 点拍摄虚拟机的快照,则您保留了该虚拟机在下午 5 点时的状态图像。此快照包括虚拟机的状态 (包括拍摄快照时虚拟机是否已启动) 以及虚拟机内存的内容。

创建虚拟机快照时,有一个选项可以让虚拟机处于静止状态。让虚拟机处于静止状态一切都是为了确保你拍摄的照片一致。

对于 Microsoft 服务器 只要在要快照的 VM 上安装了 VMware Tools 就会调用 VSS 来使 VM 静止。VSS 确保刷新所有文件系统缓冲区,并暂停文件系统的新 I/O,直到快照完成。还可以使任何 VSS 感知应用程序静止,例如 Microsoft Exchange Server、Microsoft SQL Server、Oracle Database、Microsoft SharePoint Server 和 Microsoft Active Directory。这不仅确保文件系统一致的快照,而且确保应用程序一致的快照。

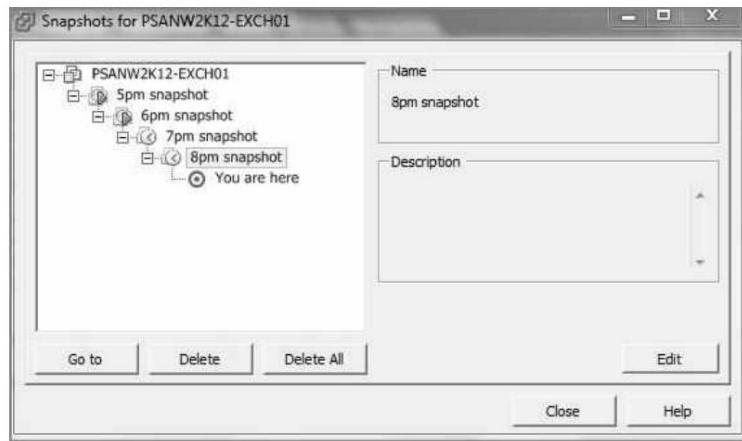
如果无法使用 VSS 来使系统静止 (例如,在非 Microsoft 操作系统上),则 VMware 专有 SYNC 驱动程序将管理系统的静止。这通常只能使文件系统静止。SYNC 驱动程序有效地保留系统上的传入写入,并将脏数据刷新到磁盘,以尝试使文件系统保持一致。

虚拟机可以有多个快照。这些快照形成一个快照树或快照链。您可以删除树中的任何快照并恢复到树中的任何快照。但是,如果快照树结构较大或复杂,则

性能开销。对变化率较高的虚拟机进行大量快照也会产生影响。因此，创建大量快照时要小心。你可以这样做并不意味着你应该这样做。

图 8.11 显示了快照链。

图 8.11 VMware Snapshot Manager 快照链

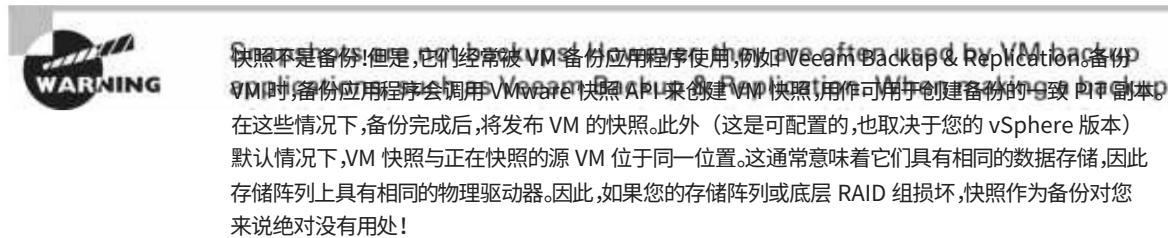


再看下一图 8.11，其中显示了 VMware Snapshot Manager 中的快照链。请注意，您现在位于快照链的最底部，默认情况下，VMware 快照的工作方式是使用链中的最后一个快照。如果想删除快照，则需要在 VMware 称为整合的过程中合并其内容。假设您有一个父（源）VM 和一个快照。您的 VM 正在使用快照运行，除非您想丢失自拍摄快照以来对 VM 所做的所有更改，否则在删除快照时，您需要将快照的内容与父卷合并。

这会将自创建快照以来的所有更改汇总到父卷。如果不合并内容，您将丢失自创建快照以来所做的所有更改。

在 VMware 环境中，当创建虚拟机快照时，与快照虚拟机关联的每个 VMDK 都会获得一个增量 VMDK 文件，该文件实际上是一个重做文件。还会创建一两个元数据文件，用于存储虚拟机内存的内容以及快照管理。所有这些增量文件都是稀疏文件（节省空间），其工作方式类似于传统的写时复制快照，当您进行更改时，这些快照会开始填充数据。

相对于虚拟机的变化率，Delta 文件显然会消耗空间，并且可以增长到与父磁盘相等的最大大小。



正如我提到的备份Microsoft服务器和Microsoft应用程序,公平地指出,Microsoft Hyper-V虚拟机管理程序还为所有支持Microsoft VSS的应用程序提供完全一致的应用程序备份:Exchange Server、SQL Server、Windows上的Oracle数据库、SharePoint Server、Active Directory等等。

现在让我们快速看一下如何拍摄虚拟机快照。练习8.2展示了如何使用vSphere Client执行此操作。

## 练习8.2

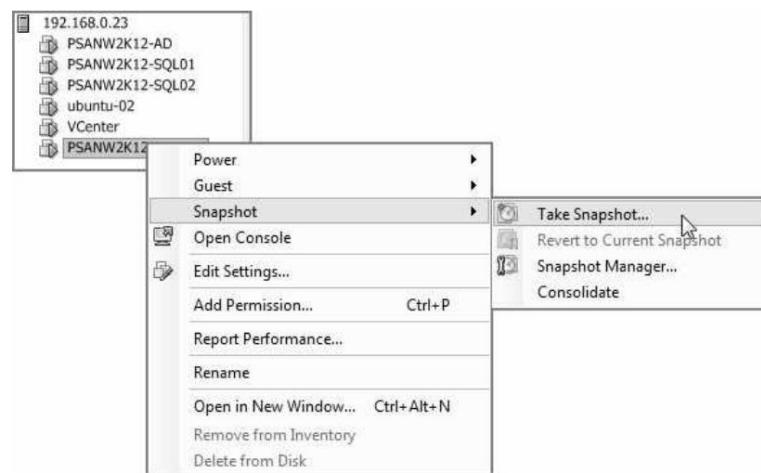
### 拍摄手动虚拟机快照

在本练习中,您将了解如何使用vSphere Client手动拍摄虚拟机快照。

1.在vSphere Client中,找到要拍摄快照的虚拟机

并单击鼠标右键。

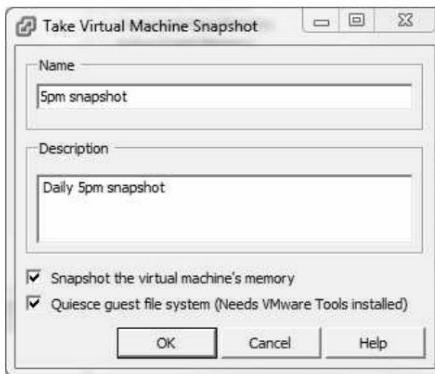
2.选择快照拍摄快照。



## 练习 8.2 (续)

3. 在出现的屏幕上,为快照命名并提供可选描述。

确保选中两个复选框,即“快照虚拟机内存”和“静默客户文件系统(需要安装 VMware Tools)”。



4. 单击确定。

5. vSphere Client 屏幕底部的“近期任务”视图显示快照操作的进度。

Recent Tasks						
Name	Target	Status	Details	Initiated by	Requested Start Ti...	Start Time
Create virtual machine snapshot	PSANW2K12-E...	24%	<div style="width: 24%;"></div>	root	01/06/2013 22:26:07	01/06/2013 22:26:07

## 基于阵列的快照

与基于阵列的复制一样,基于阵列的快照的优点是可以将快照工作从主机系统转移到阵列。这可以释放主机上的 CPU 和内存,并提供更快的快照,并可能更快地从快照中恢复。

但是,它也会增加配置的复杂性。与所有事物一样,如果可能,您应该先试用再购买。

与基于虚拟机管理程序的快照一样,所有基于阵列的快照技术都可以通过与 Microsoft VSS 等产品集成来实现应用程序感知,从而提供应用程序一致性快照。此外,当与备份软件结合使用时,备份软件和基于阵列的快照的组合可以通知应用程序它们已备份,从而使它们可以截断日志文件并执行通常与备份相关的其他整理工作。

在接下来的几页中,您将仔细了解基于阵列的快照技术和专注于它们的实施方式以及每种方法的优缺点。

通常使用两种主要类型的基于阵列的快照技术：

- 节省空间的快照
- 完整克隆

这些技术有时会有不同的名称,因此我将尽量保持一致

在本章中。有时我可能会将节省空间的快照称为快照,有时我会将完整克隆称为克隆。此外,在谈到白话时,我将使用术语源卷来指代快照所取自的原始卷。我还将在图中用虚线绘制快照,以表明它们是节省空间的。

## 节省空间的快照

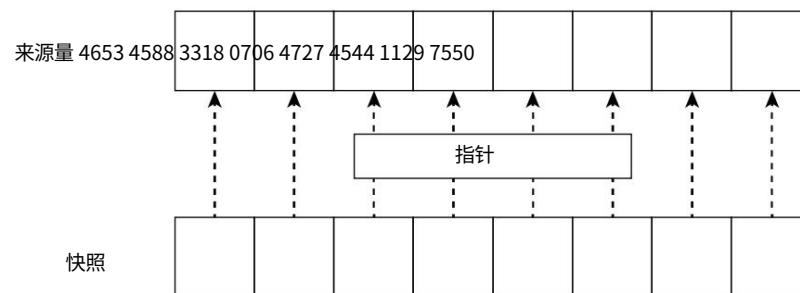
节省空间的快照是基于指针的。处理此类快照时需要考虑两个重要概念：

空间效率告诉我们,快照仅包含自拍摄快照以来对源卷所做的更改。这些更改有时称为增量。

基于指针告诉我们,当创建快照时,快照的整个地址空间由将 I/O 请求引回源卷的指针组成。

图 8.12 显示了指针的概念。

图 8.12 快照中指针的使用



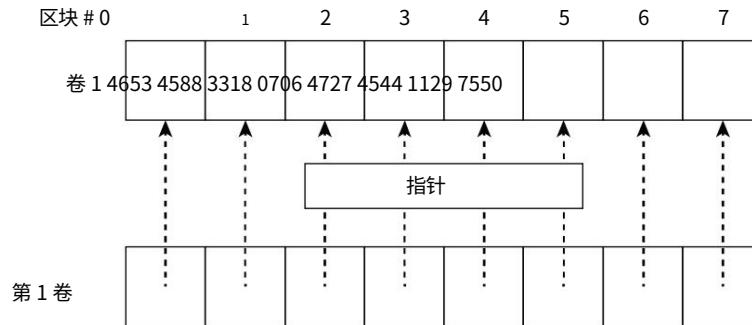
让我们看一个简单的例子。假设我们有一个名为Vol1 的源卷。

Vol1有八个数据块,编号为 0-7。周一下午 1 点,我们对Vol1进行节省空间的快照。此时, Vol1中的所有块如图 8.13 所示。Vol1 的快照称为Vol1.snap。当我们创建快照时, Vol1.snap不占用磁盘空间。如果主机挂载此快照并读取其内容,则所有读取 I/O 都将重定向回Vol1 中包含的原始数据。Vol1.snap基本上是一个空卷,仅包含指针 (通常在内存中) ,这些指针将读取 I/O 重定向回包含数据的主卷。

接下来,我们假设到下午 2 点, Vol1已更新了块 0 和 3。现在,如果主机想要读取Vol1.snap 中的数据,它将被重定向回源卷(Vol1) ,但读取块 0 和 3 时除外。由于块 0 和 3 自创建快照以来已发生变化,因此原始内容 (4653 和 0706) 已复制到Vol1.snap

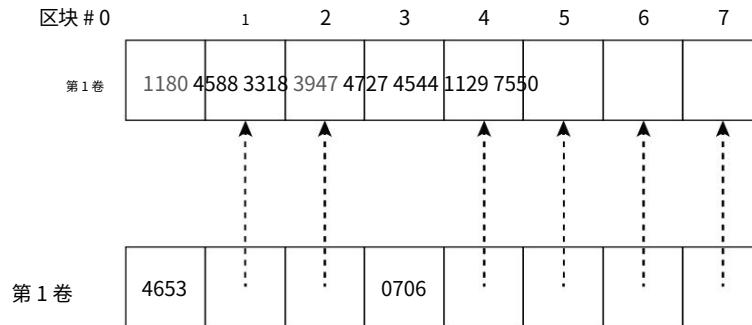
为了保存下午 1 点Vol1的图像

图 8.13 下午 1 点的源卷和快照卷



下午 2 点,快照配置如图 8.14 所示,快照卷仍然代表了下午 1 点时主要体积的精确图像

图 8.14 下午 2 点的源卷和快照卷



快照也是即时可用的,这意味着只要您单击“创建”按钮,从概念上讲,快照已经可供挂载和使用。

快照也可以是读/写的,这使得它们非常灵活,并可用于许多用例。但是,如果您将快照安装为读/写并向其发出写入操作,它显然不再代表拍摄快照时源卷的状态。

这是可以的,只要您不打算使用快照回滚到拍摄快照的时间点!

如果您刚接触快照技术,并且仍然不清楚这一点,请去喝杯咖啡,排除干扰,然后重新阅读上述内容。了解节省空间的快照的工作原理至关重要。

### 完整克隆快照

完整克隆快照(有时也称为完整克隆或克隆)不节省空间。它们不使用指针,因此更容易理解。然而,它们仍然是源卷的 PIT 副本。

在创建克隆时,将创建源卷的精确的逐块副本。  
如果源卷为 2 TB,则克隆也将为 2 TB,这将消耗 4 TB 的存储空间。  
您可以看到完整克隆如何快速消耗存储空间。

与快照一样,克隆可以读取/写入,一旦您写入克隆(或快照),它就不再代表创建克隆时源卷的状态。这使得它无法作为回滚到创建克隆时源卷状态的工具。但是,它可用于测试和其他目的。

### 瞬时克隆的工作原理

大多数基于阵列的克隆技术都提供即时克隆。与节省空间的快照一样,这意味着单击“创建”后即可立即使用它们。

但是如何才能立即提供大容量多 TB 卷的克隆呢?将 2 TB 数据复制到克隆卷肯定需要很长时间吧?

即时克隆通常按如下方式工作。创建克隆后,阵列开始尽可能快地将数据从源卷复制到后端的克隆,但会小心不要减慢前端的用户 I/O。如果您读取或写入已复制的克隆区域,显然这是可以的。但是,如果您想读取尚未复制的克隆区域,阵列会将读取 I/O 重定向到源卷以获取相关块,或者立即将这些块从源复制到克隆,这样读取 I/O 就不会失败。

无论哪种方式,操作都是由后台的阵列执行的,主机和应用程序不会注意到任何差异。

如果您的克隆是读/写克隆,则对尚未从源复制的克隆区域的任何写入 I/O 都可以直接写入克隆卷,并且不再需要复制源卷中的相应块。

图 8.15 比较了 2 TB 卷的空间高效快照与进行快照/克隆时 2 TB 卷的完整克隆。

显然,随着时间的推移,快照会开始消耗空间。消耗多少空间完全取决于您的 I/O 配置文件。对源卷的大量写入和更新将触发快照卷中的大量空间消耗。

因此,如果完整克隆会占用大量磁盘空间,那为什么要费心使用它们呢?为什么不总是想要节省空间的快照吗?有几个常见原因:

- 读取和写入完整克隆不会影响源卷的性能。您可以不断修改克隆而不会影响源卷的性能。这对于需要在繁忙时进行备份的繁忙数据库非常有用;您可以进行克隆并备份克隆。您也可以使用

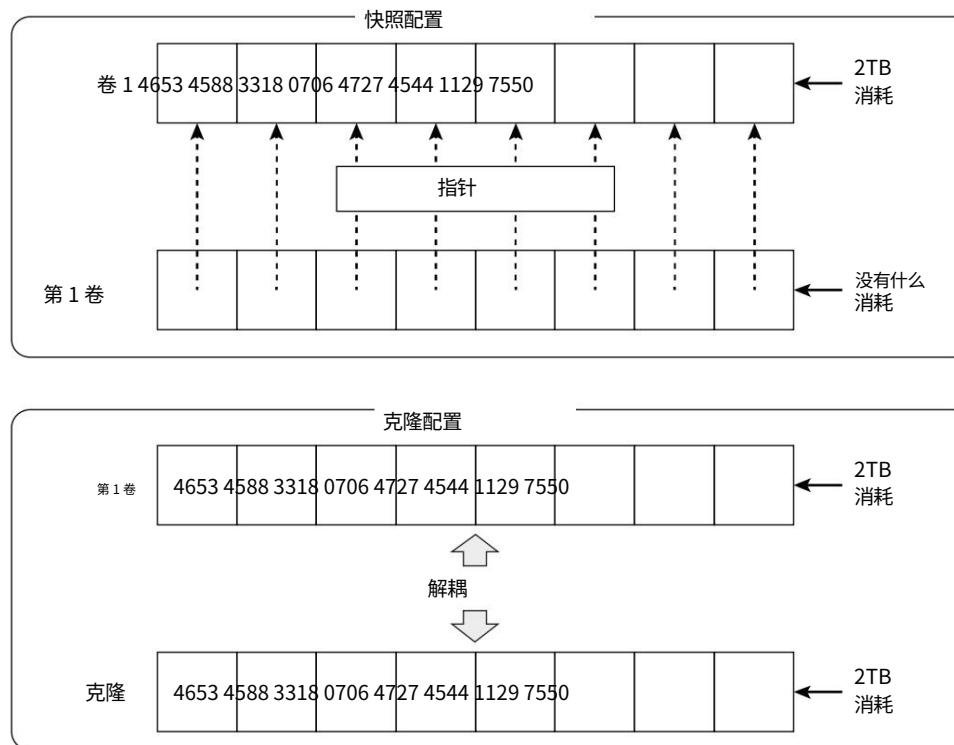
将同一个繁忙数据库用作报告数据库,这样针对克隆运行的报告就不会影响数据库的性能。这是因为完整克隆完全独立于源,应在单独的主轴上创建,并且不包含指针。相反,节省空间的快照会与其源卷共享大量数据。如果您使用密集的读写 I/O 来敲击快照,您将引用与源共享的一些数据块,从而敲击源卷。

#### ■ 完整克隆完全独立于源卷。如果要克隆的物理驱动器

如果源卷发生故障,克隆将不会受到影响(只要您不将源卷和克隆卷放在同一驱动器上)。

由于节省空间的快照依赖于源卷,因此如果主卷后面的磁盘发生故障,则源卷和节省空间的快照都将发生故障。

图 8.15 比较节省空间的快照和完整克隆的空间消耗



在现代 IT 世界中,预算紧张且不断缩减,快照往往比完整克隆更受欢迎,因此我们将花费更多时间详细了解基于阵列的、节省空间的快照。

#### 快照范围大小

快照本质上是节省空间的。它们的一大优势是占用的空间非常小。因此,说到节省空间的快照,我们可以节省的空间越多

保存得越多越好。快照范围越小,快照范围越大越好。

因此,快照扩展大小为 16 KB 比 256 KB 要好。虽然这些数字看起来可能很小,但在拥有数百个快照的大型系统上,这些数字很快就会累积起来。

依赖于阵列底层的异步复制技术也是如此。

快照技术。在这些情况下,具有较大快照范围大小的阵列必须将大量数据复制到远程系统。

## 保留快照空间

某些存储阵列(尤其是较旧的阵列)要求您留出容量作为快照保留空间。然后,此空间用于存储与快照相关的数据。如果您预计源卷会发生很大变化,则必须为快照数据保留大量空间,而如果您的源卷不会发生很大变化,则不必为快照数据保留那么多空间。这种方法的问题有两个方面:

- 通常很难知道您的源卷是否会受到高利率或低利率的影响的變化。
- 即使您预计源卷将经历较高的变化率,但仍然几乎不可能知道到底要留出多少空间。

这通常会导致快照保留空间过大,这意味着您的阵列中有一些容量专门用于存储快照数据,但可能从未使用过。这可不是什么好事。

幸运的是,大多数现代存储阵列都不以这种方式工作。大多数动态分配空间可根据需要分配给快照,这些空间来自所有卷都可以访问的通用存储池(而不是专用的隔离存储池)。

在这些更现代的架构中,通过拥有多个池,仍然可以确保快照空间永远不会与源卷的空间来自同一个池(有时出于可用性和性能原因,这被视为一种良好做法)。重要的是,这两个池都可以用于源卷和快照数据;您可能只是选择不从同一个池中为源和相关快照留出空间。

例如,池 A 可能用于源卷 1,而池 B 用于源卷 1 的快照。池 B 可能用于源卷 2,而池 A 用于源卷 2 的快照。在此配置中,两个池均用于源卷和快照。

如果空间不足,源卷和快照数据共享池可能会引发问题,大多数阵列都允许您配置策略,规定容量不足时如何处理空间和快照。例如,当容量达到 90% 时,您可以停止创建任何新快照,而当容量达到 95% 时,您可以开始删除最旧的快照,以释放数据以用于源卷的增长。

## 快照和分层

由于快照通常被认为不如其关联的生产源卷重要,因此将它们存储在较低层存储中很有用。大多数存储阵列都允许您指定从较低层存储创建快照卷。

但是,许多现代阵列允许通过阵列的自动分层算法来管理快照。这样可以在较高层中创建快照,然后如果未使用,则随着时间的推移逐渐下降到较低层。这在快照在创建后的最初几天内被访问但在接下来的几周内未被访问并最终被删除的情况下非常有用。如果由阵列自动分层算法控制,快照可以在高性能存储上开始其生命并在使用时停留在那里,但在没有被访问时,其生命最后几天会在较慢、较便宜的存储上度过。

### 崩溃一致性和应用程序感知快照

与标准的基于阵列的复制一样,基于阵列的快照技术在其开箱即用状态下不具备应用程序感知能力,这意味着快照中的数据将是我们所说的崩溃一致性。



**术语 “崩溃一致性”表示数据的一致性,就像服务器意外崩溃一样。The term “crash consistency” refers to data consistency, just as if the server had unexpectedly crashed.**

快照无需任何特殊软件与应用程序集成,因此在崩溃时仍能保持一致,因为大多数应用程序和数据库在服务器的本地缓冲区中缓存数据,并延迟将这些更新同步到磁盘。在任何时间点,磁盘上的数据都不能准确表示应用程序的当前状态。理解这一点非常重要。想象一下,你发一封电子邮件回家,说你明天上午 10 点到达火车站,然后又发一封电子邮件说你改变了计划,现在要回去,并于上午 9 点到达机场。如果你的第二封电子邮件一直没有到达,机场就不会有人来接你,你回家的火车之旅也将白白浪费。应用程序写入磁盘也是一样:磁盘上的数据可能不是应用程序状态的最新副本,因此,除非你先将应用程序的状态刷新到磁盘(这是称为静止的过程的一部分),否则对磁盘内容进行快照是没有用的。

幸运的是,大多数基于阵列的快照都可以被赋予应用程序智能,因此能够创建应用程序一致的快照。这通常是通过应用程序框架或应用程序集成软件实现的。例如,如前所述,Microsoft VSS 是一个 API 和框架,允许第三方快照技术(包括阵列快照技术)与 Microsoft 应用程序集成并生成应用程序一致的快照。

如果您计划依靠快照来恢复应用程序,您最好计划将快照技术与应用程序集成,以便获得应用程序一致的快照。如果不这样做,您将面临快照无法恢复应用程序的风险,或者即使可以,也需要做更多工作才能实现这一点。

### 写时复制快照

写时复制(CoW)快照是一种节省空间的快照,目前使用较少。它们保持主卷的连续布局,

这很棒,但不幸的是它们也具有首次写入性能的开销。

更仔细地考虑一下这两点：

- 拍摄快照后,首次写入源卷上的任何块时,需要将该块的原始内容复制到快照卷以进行保存。只有完成此操作后,写入操作才能覆盖源卷上块的原始内容。这可能会产生相关的性能开销。这在基于 SSD 的阵列或 SSD 卷上不是什么大问题,因为无需移动磁头等操作,因此不会产生定位延迟。此外,许多阵列可以通过在缓存中执行操作并异步更新磁盘内容来隐藏此性能损失,尽管如果您计划保留大量快照,这种方法的扩展性往往不是很好。

- CoW 快照保留主卷的原始连续布局。这是在下一节中将更详细地讨论。

## 写入时重定向快照

写入时重定向 (RoW) 快照有时也称为写入时分配快照。它们与写入时复制快照有很大不同。当受快照保护的卷收到写入 I/O 时,源卷中的原始数据不会更改或复制到快照卷。数据被冻结并通过非常快速的元数据操作成为快照的一部分。进入源卷的新写入只是通过一种重定向操作写入阵列上的新位置,并且源卷的元数据表会更新以反映新位置。

让我们看一个简单快捷的例子。假设您有一个名为Vol1的卷,它包含 100 个连续的块,即 0-99。您在下午 1 点快照 Vol1,然后在下午 2 点更新块 50 的内容。使用写入时重定向,块 50 的内容将冻结在其原始状态并成为快照卷地址空间的一部分;我们将其称为Vol1.snap。原本要写入块 50 的写入数据被重定向并写入其他地方;我们称之为块 365。Vol1 的元数据已更新,因此逻辑块 50 现在映射到磁盘上的物理块 365。



快照写入速度非常快,不会影响卷性能。*They don't impact volume performance.*

重定向写入快照的缺点是它们往往会使源卷的地址空间碎片化。在我们的简单示例中, Vol1最初在磁盘上有 100 个连续的块,即块 0-99。在更新块 50 的内容后, Vol1在系统其他位置的随机位置上有一个块(其内容的 1%)。如表 8.1 所示。

表 8.1 重定向写快照行为对卷地址空间的影响

逻辑地址	实际地址
0	0
1	1
2	2
3	3
...	...
四十八	四十八
49	49
50	365
51	51
52	52
...	...
97	97
98	98
99	99

源卷地址空间的碎片会累积起来,最终对卷性能产生负面影响。碎片严重的卷无法再从顺序读写中获益,迫使所有访问模式都变成随机的。

对于全灰系统或含有大量灰分的系统来说,这个问题不大。

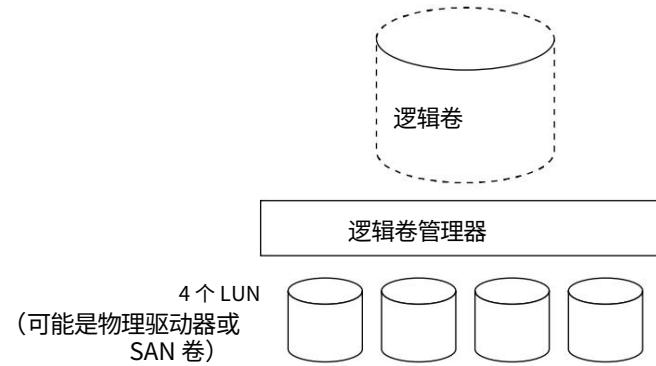
## 基于主机的快照和卷管理器

制作和管理快照的另一种选择是通过基于主机的卷管理器。卷管理器是一种提供高级基于主机的卷管理功能的软件,例如:

- 从多个底层物理 LUN 创建条带、镜像或 RAID 保护卷。例如,主机从存储阵列获得多个 RAID 保护的 LUN,然后让基于主机的卷管理器在卷顶部添加一层条带,这种情况仍然很常见。如图 8.16 所示。

- 动态添加和删除LUN 以及相关的卷增长。
- 快照创建和管理。

图 8.16 卷管理器在基于阵列的 LUN 上创建条带



与其他类型的快照一样,卷管理器快照可供备份应用程序使用:对卷进行一致性快照,然后备份一致性快照。关于快照技术,卷管理器快照属于基于主机的阵营,因此会消耗主机资源。它们不会将快照管理所涉及的任何工作转移到存储阵列上。

从好的方面来说,它们可以提供文件系统级的一致性,并且通常可以配置为读/写快照。但是,这两种功能都依赖于所涉及的卷管理器技术,有时还依赖于所使用的 Linux 内核。它与基于阵列的快照一样复杂。无论您选择哪种技术,您都需要在实际生产环境中部署之前进行研究和测试。

通常,为了使卷管理器快照应用程序一致,您需要手动(可能通过脚本)使应用程序静止。

缺点是,卷管理器快照通常具有相关的性能开销。卷管理器快照通常只对运行卷管理器的主机可见且可导出。

然而,它们在小型商店和开发环境中被广泛使用。

## 概括

在本章中,我们讨论了复制技术如何融入业务连续性和灾难恢复的大局,然后深入探讨了当今存储领域常见的不同类型的复制。我们比较了同步和异步复制,以及它们对恢复点目标、应用程序性能、网络带宽要求和其他因素的影响。我们讨论了复制技术如何在堆栈的不同层级实现,包括应用程序、操作系统和

存储阵列。我们还讨论了常见的复制拓扑以及它们如何保护企业及其数据免受不同类型灾难的影响。然后，我们结束了讨论快照技术和应用程序一致性的章节。

## 章节概要

**同步复制** 同步复制提供零数据丢失，在所有不同类型的复制中提供最佳的恢复点目标。但是，它会带来性能损失，并且会随着源卷和目标卷之间的距离增加而增加。

**异步复制** 异步复制实际上肯定会丢失部分数据。但是，它不会像同步复制那样造成性能损失。异步复制允许您将源和目标置于几乎无限远的距离，并且通常可以配置恢复点目标。

**应用层复制** 在应用层执行的复制将复制负担转嫁给主机，但提供了出色的应用程序感知能力，确保远程副本在事务上是一致的。

**节省空间的快照** 节省空间的快照是卷或虚拟机的即时副本，可在特定时间提供卷或虚拟机的映像。这些快照可用作回滚点、备份源或测试。它们绝对不能替代备份！

# 章节 9



## 存储虚拟化

---

本章涵盖的主题：

- ✓ 什么是存储虚拟化
- ✓ 基于主机的存储虚拟化
- ✓ 基于网络的存储虚拟化
- ✓ 基于控制器的存储虚拟化
- ✓ 带内与带外存储虚拟化
- ✓ 存储虚拟化的用例
- ✓ 如何配置基于控制器的虚拟化
- ✓ 软件定义存储



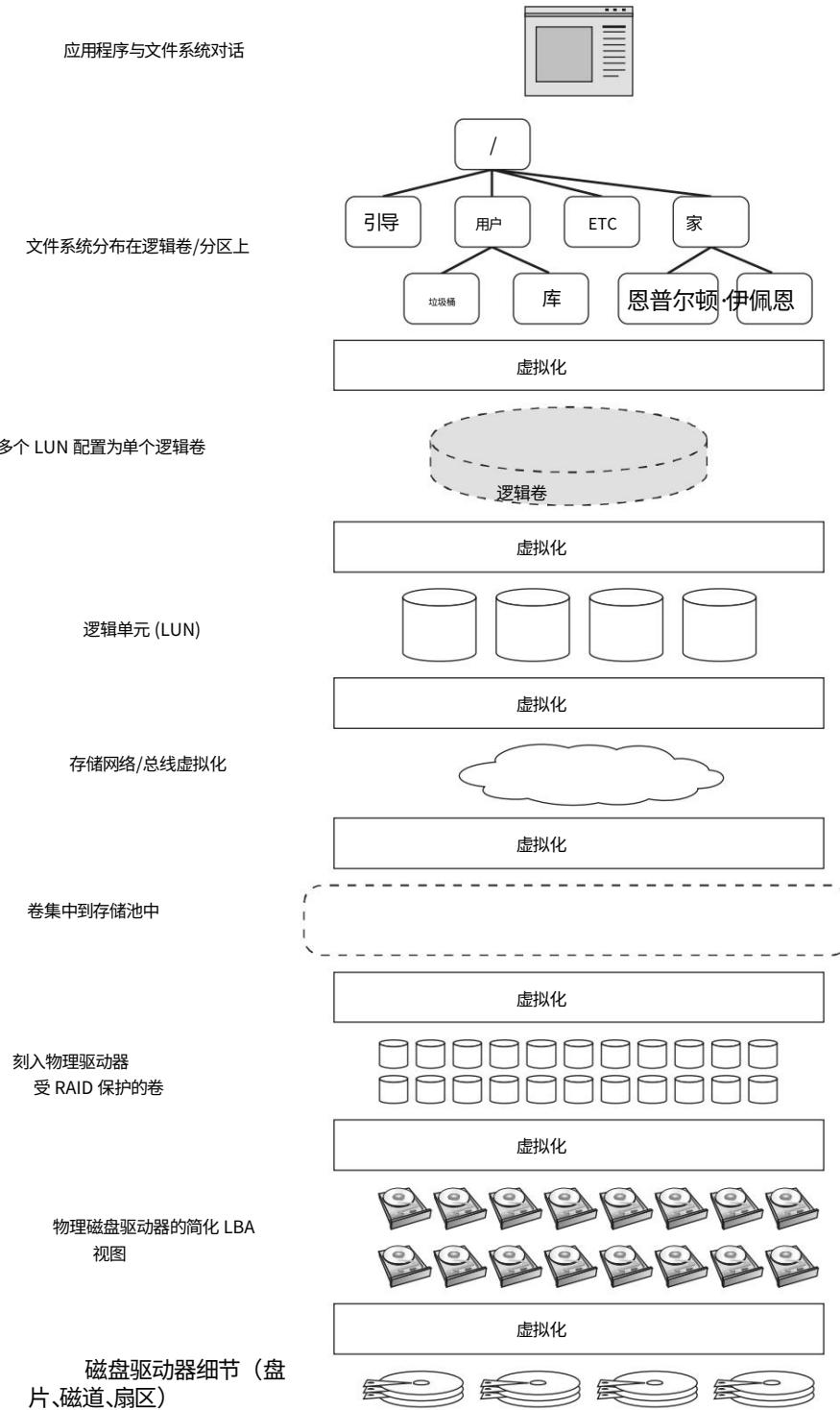
本章介绍了现实世界中常见和实施的存储虚拟化形式。我们首先将解释典型存储堆栈中存在的许多不同虚拟化层，这些虚拟化层已经存在多年，但通常不被认为是虚拟化本身。我们将介绍 SNIA 共享存储模型 (SSM)，并以此为参考框架，将不同类型的存储虚拟化映射到它们在 SSM 堆栈中的位置。我们将花大量时间讨论基于控制器的虚拟化，这是迄今为止供应商和技术公司实施的最流行的存储虚拟化形式，也是客户在现实世界中部署的。我们将回顾一些潜在的用例，研究存储虚拟化的常见实现方式，并考虑一些潜在的陷阱。我们将通过探索最新上市的存储虚拟化形式：软件定义存储来结束本章。我们将研究软件定义存储的一些示例、优点、缺点和潜在未来，它有望成为一种颠覆性的架构。

## 什么是存储虚拟化

存储虚拟化是那些对许多人来说含义各异的术语之一。问五个人它的含义，你很容易得到五个不同的答案。这并不是因为存储虚拟化不成熟或尚未成为一项技术。更可能是因为虚拟化几乎发生在存储堆栈的每一层，而且新形式的存储虚拟化一直在不断出现。

从最高层次来看，虚拟化是将物理设备抽象为逻辑或虚拟设备。但是，存储虚拟化（以及其他形式的虚拟化）通常更进一步，虚拟化已经虚拟化的设备。考虑到这一点，让我们快速查看一个常见的存储堆栈，其中应用程序位于顶部，物理存储介质位于最底部，然后通过应用程序 I/O 所采用的路线向下移动堆栈。在此过程中，我将指出虚拟化发生的许多领域中的一些。图 9.1 有助于使事情更清楚一些。

图 9.1 传统 SAN 存储的虚拟化级别



应用程序通常与位于逻辑卷之上的文件系统或数据库通信。这些逻辑卷通常是通过将多个离散 LUN 虚拟化为单个虚拟/逻辑卷而创建的。底层 LUN 可以通过 SAN 访问,SAN 本身被划分为区域,这些区域基本上是虚拟 SCSI 总线。在存储阵列上,LUN 本身是缓存内存中定义的虚拟设备。这些虚拟 LUN 下方是存储池,存储池是有时由多个 RAID 组创建的虚拟存储池。RAID 组下方是多个物理驱动器,但即使是这些物理驱动器也被其板载控制器虚拟化为简单的逻辑块地址 (LBA) 空间,以隐藏内部工作的复杂性。这里描述的每个步骤都可以视为虚拟化。

如图 9.1 所示,堆栈中的较低层通常会随着我们向上移动而虚拟化堆栈,从而导致一层又一层的虚拟化。最终结果是,尽管您的应用程序、数据库或文件系统可能认为它正在直接与专用的本地连接的物理磁盘驱动器通信,但事实并非如此!它正在与一个虚拟设备通信,该虚拟设备已在堆栈的多层上进行了多次虚拟化,以提供更高的性能和更高的可用性。



由于为了向应用程序提供持久存储而出现多层次虚拟化,因此任何应用程序、文件系统、卷管理器甚至操作系统都不知道其数据存储在磁盘的那个位置。它们可能在本地文件系统中有十一个地址,但该地址映射到逻辑卷上的地址,该地址映射到 LUN 上的地址,该地址映射到池中的页面,该页面映射到 RAID 组上逻辑设备上的块,该块映射到多个物理驱动器上的 LBA 地址,最终映射到驱动器盘片上的扇区和磁道。关键是应用程序、文件系统和操作系统工具完全不知道其数据位于底层存储介质上的哪些扇区上!

*to blocks on a logical device on a RAID group, which map to LBA addresses on multiple physical drives, which finally map to sectors and tracks on drive platters. The point is that application, filesystem, and operating system tools have absolutely no idea on which sectors their data lives on the underlying storage media!*

不要因为担心而彻夜难眠!这些虚拟化层已经使用多年,经过了充分的现场测试,并且被认为可靠。你会发现,这些虚拟化层为游戏带来了巨大的好处! and truly field tested and known to be sound. And as you will find out, these layers of virtualization bring massive benefits to the game.

尽管前面描述了虚拟化的许多级别,但是这些级别往往已经建立并且根深蒂固,以至于我们常常忘记它们的存在;而且几乎从未将它们视为存储虚拟化。we often forget that they exist and hardly ever consider them as storage virtualization.

## SNIA 共享存储模型 Storage Model

在我们深入了解市场上各种类型的存储虚拟化技术之前,最好先快速了解一下 SNIA 定义的共享存储模型(Shared Storage Model, SSM), probably a good idea to have a quick look at the SNIA-defined Shared Storage Model (SSM).

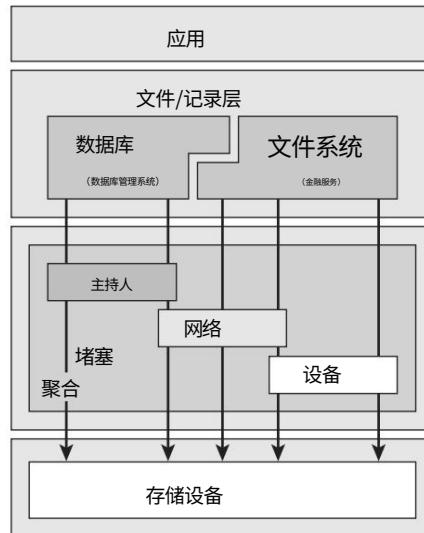


*As mentioned in the book, SNIA is the Storage Networking Industry Association, a non-profit industry association dedicated to advancing storage technology and standards.*

SNIA SSM 与网络世界中经常提到的更著名的开放系统互连 (OSI) 七层网络模型非常相似。尽管 SNIA SSM 远不如 OSI 模型出名且引用较少,但它在描述存储概念和架构时非常有用。在本章中,我们将使用 SSM 作为参考点来描述不同类型的存储虚拟化。

图 9.2 显示了 SNIA 共享存储模型。

图 9.2 SNIA 共享存储模型



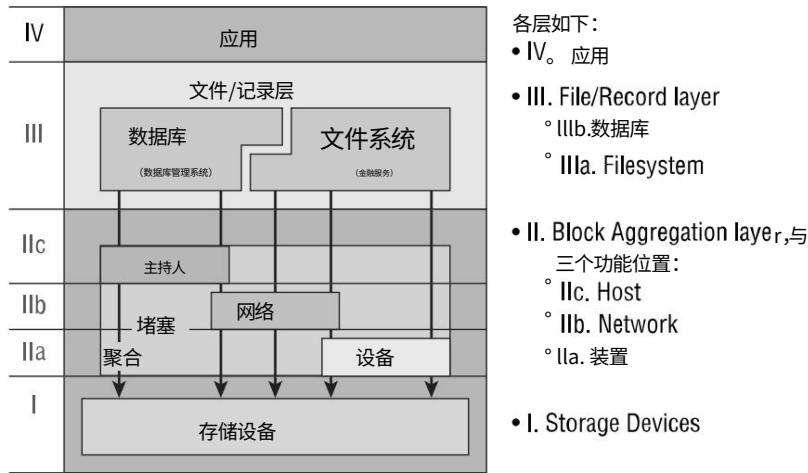
如图 9.2 所示,SNIA SSM 是一个非常简单的分层模型。底层最底层是磁盘驱动器、固态介质和磁带等最低级别的组件。最顶层是文件系统和数据库等高级结构。

从技术上讲,SSM 在应用层下分为三个有用的层:

- 文件/记录层
- 块聚合层
- 存储设备层

过去,大多数有趣的存储虚拟化操作都发生在块聚合层。但如您所见,该层内有三个独立的子层。图 9.3 更详细地介绍了 SSM,并将这些层进一步细分。

图 9.3 SNIA 共享存储模型的进一步细分



各层如下：

- **IV. 应用**
- **III. File/Record layer**
  - **IIIa. Filesystem**
- **II. Block Aggregation layer**, 与三个功能位置：
  - **IIc. Host**
  - **IIb. Network**
  - **IIa. 聚合**
- **I. Storage Devices**



SNIA 使用罗马数字来标识 SSM 堆栈的各层。这样做是为了避免与 OSI 网络模型的各层混淆。This is done so that they are never confused with the layers of OSI

基于SSM分层模型,SNIA定义了三种类型的存储虚拟化:

- 基于主机的虚拟化
- 基于网络的虚拟化
- 基于存储的虚拟化

所有这些都发生在 SSM 的块聚合层。

让我们快速地看一下每一个。

## 基于主机的虚拟化

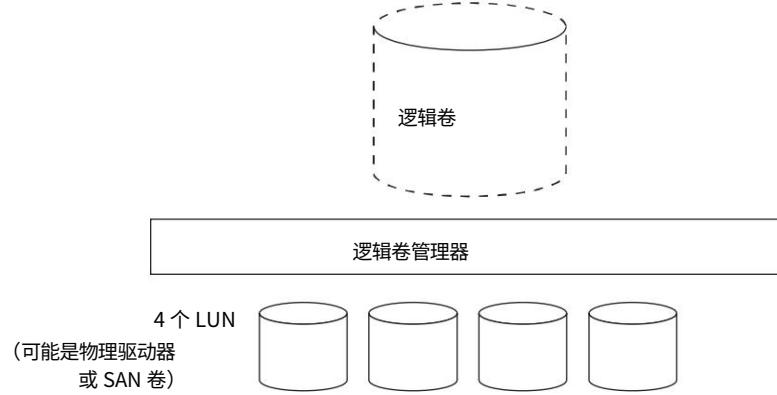
基于主机的存储虚拟化通常通过逻辑卷管理器 (LVM) 实现,通常简称为卷管理器。卷管理器与块存储 (DAS 或 SAN) 配合使用,并将多个块 I/O 设备虚拟化为逻辑卷。

然后,这些逻辑卷将被提供给堆栈中的更高层,文件系统通常会写入其中。

逻辑卷可以进行切片、条带化、串联,有时甚至可以使用软件 RAID 进行保护。从存储虚拟化的角度来看,它们从下层获取设备并创建新的虚拟设备(称为逻辑卷)。这些逻辑卷通常具有卓越的性能、更高的可用性,或两者兼而有之。

图 9.4 显示了一个简单的卷管理器,它从四个卷中创建一个逻辑卷  
从 SAN 连接的存储阵列向其提供的 LUN。

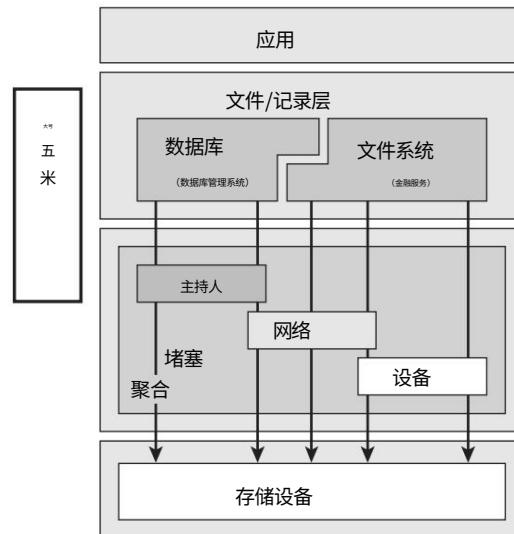
图 9.4 逻辑卷管理器创建逻辑卷



尽管卷管理器确实可行,有时也很有用,但并不经常使用  
创建受软件 RAID 保护的逻辑卷。这通常是由于创建基于奇偶校验的软件 RAID 可能带来性能  
开销。由基于主机的卷管理器创建的逻辑卷也往往由于以主机为中心而具有有限的可扩展性  
它们无法在多个主机之间轻松共享。

图 9.5 显示了基于主机的卷管理器如何映射到 SNIA SSM。卷管理器是基于主机的软  
件,它操纵块存储设备并将它们聚合为新的块设备 (例如,通过获取两个块设备并将它们连  
接在一起以形成单个镜像卷)。然后数据库和文件系统使用这些卷。

图 9.5 LVM 映射到 SNIA 共享存储模型



在现实世界中,人们通常不认为 LVM 功能是存储虚拟化。公平地说,这可能是因为卷管理器已经存在并执行这种基于主机的虚拟化,比我们使用存储虚拟化这一术语的时间要长得多。

## 基于网络的虚拟化

基于网络 (或基于 SAN) 的存储虚拟化已经尝试过,但都失败了。当时有几种解决方案,最知名的可能是 EMC Invista,但大多数 (如果不是全部的话) 都已消亡。

从概念上讲,网络层的存储虚拟化需要智能  
网络交换机或基于 SAN 的设备,可执行如下功能:

- 存储阵列的聚合和虚拟化
- 将异构阵列中的 LUN 组合成单个 LUN
- 结构级别的异构复制 (在不同的阵列之间进行复制  
技术)

所有这些方案都得到了供应商和 SNIA 的大力推广和赞扬。所有这些方案都曾被尝  
试过,但大多数都失败了。这些方案在纸面上看起来不错,但大多数都从未受到客户的欢迎。

据记录显示,这些设备往往是带外异步解决方案。

## 基于存储的虚拟化

大多数人将基于存储的虚拟化称为基于控制器的虚拟化。而基于控制器的虚拟化是目前现实世界中存储虚拟化的主要形式。因此,您将在本章后面深入研究基于控制器的虚拟化。但简而言之,高级存储阵列能够连接到  
其他下游存储阵列,发现它们的卷,并以与使用自己的内部磁盘相同的方式使用它们,从而有效地虚拟化其后面的  
下游阵列。

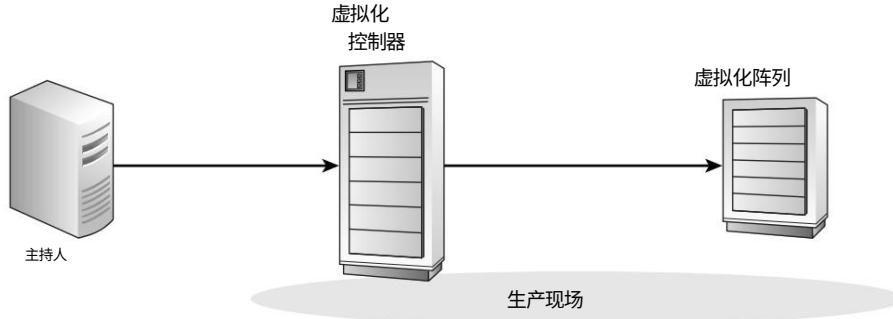
## 带内和带外虚拟化

SNIA 将存储虚拟化分为带内或带外。

在带内虚拟化中,执行虚拟化的技术直接位于数据路径中。这意味着所有 I/O (用户数据和控制数据) 都通  
过执行虚拟化的技术。现实世界中的一个常见示例是位于其他存储控制器前面并对其进行虚拟化的存储控  
制器。执行虚拟化的控制器直接位于数据路径中 - 位于发出

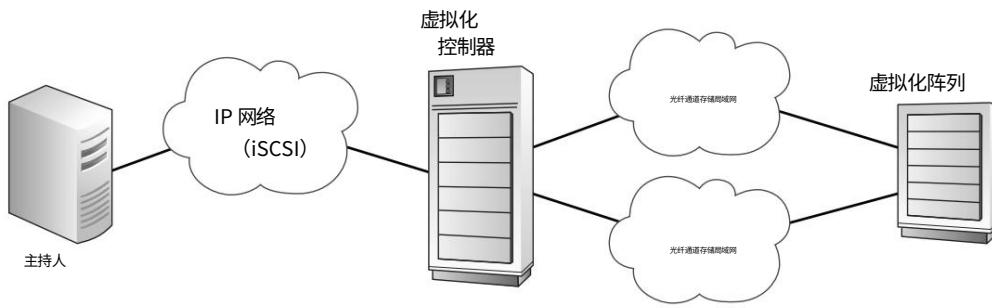
I/O 和它正在虚拟化的存储阵列。就主机而言,执行虚拟化的控制器是其目标,看起来和感觉起来都像一个存储阵列(在这种情况下,它是,只是它不是数据的最终目的地)。而从正在虚拟化的存储阵列的角度来看,它看起来像一个主机。图 9.6 给出了这种配置类型的高级视图。

图 9.6 带内存存储虚拟化



这种带内虚拟化方法有几个优点。一个主要优点是主机上不需要特殊驱动程序,在现实世界中,这可能是一个巨大的好处。另一个优点是带内虚拟化可用于桥接协议,例如让执行虚拟化的设备连接到后端的 FC 存储阵列,但在前端显示 iSCSI LUN。图 9.7 显示了执行带内虚拟化的设备在 iSCSI 和 FC 之间桥接的配置。

图 9.7 带内存存储虚拟化设备执行桥接



带外虚拟化,有时也称为非对称虚拟化,通常具有元数据通过虚拟化设备或设备传输,通常需要在主机上部署特殊的 HBA 驱动程序和代理软件。与带内解决方案相比,它不太流行,部署也少得多。

# 为什么要进行存储虚拟化？

那么，推动存储虚拟化流行的的因素有哪些呢？存储虚拟化试图解决以下挑战：

- 管理
- 功能性
- 性能
- 可用性
- 技术更新/迁移
- 成本

在管理方面，存储虚拟化可以通过虚拟化多种资产来提供帮助：

无论是磁盘驱动器还是存储阵列，都位于单点虚拟化之后。此单点虚拟化随后成为大多数日常任务的单点管理。

例如，在 Hitachi VSP 虚拟化控制器后面虚拟化三个存储阵列，可使用 Hitachi VSP 管理工具管理所有四个阵列。与所有优秀的虚拟化技术一样，来自多个供应商的多种技术可通过单一管理界面进行虚拟化和管理。

在功能方面，可以为正在虚拟化的存储设备添加功能。一个常见的例子是虚拟化位于较高层阵列后面的低层存储阵列。较高层阵列（执行虚拟化的阵列）的特性和功能（例如快照和复制）通常会扩展到较低层阵列提供的容量。

在性能方面，存储虚拟化可以通过多种方式提高性能。一种方法是通过在单个虚拟 LUN 后面虚拟化更多磁盘驱动器。例如，如果您的 LUN 后面有 16 个驱动器并且需要更高的性能，则只需增加虚拟 LUN 后面的驱动器数量即可。存储虚拟化通常协助服务器虚拟化的另一种方法是在具有大缓存的高性能企业级阵列后面虚拟化较小、性能较低的阵列，从而有效地将大缓存的性能扩展到正在虚拟化的阵列。

在可用性方面，RAID 组处于堆栈的较低位置，用于创建受 RAID 保护的 LUN，这些 LUN 看起来和感觉起来像物理块访问磁盘驱动器，但实际上由多个物理磁盘驱动器组成，形成一个弹性 RAID 集。基于阵列的复制和快照技术也增加了虚拟化提供的保护。

在技术更新方面，基于控制器的虚拟化通常用于平滑存储迁移。这有助于将呈现给特定主机的卷从一个存储阵列无中断地迁移到另一个存储阵列，而无需主机停机或 I/O 中断。

从成本角度来看，存储虚拟化可以通过将更便宜、更低端的存储阵列虚拟化到更高性能的阵列来提供帮助。这将更高性能阵列的性能优势带到了虚拟化阵列的容量中。它还将高端阵列的功能（如复制、快照、重复数据删除等）扩展到低成本阵列的容量中。



所有这些虚拟化的好处都值得怀疑。它们都是可行的，而且在现实世界中也得到了实现，但也都存在一些问题。举一个简单的例子，在高性能阵列后面虚拟化低端存储阵列可以将高性能阵列的性能扩展到低端阵列提供的容量，但只能达到一定程度。如果你试图把它推得太远，高端阵列的性能就会开始受到影响。

*In extend the performance of the high-performance array to the capacity provided by the low-end array, but only up to a point. If you try to push it too far, performance of the high-end array can start to suffer.*



如以往，所有成本节省都取决于您支付多少。虚拟化设备和许可证很少便宜或免费。

*As always, all cost savings depend on how much you pay. Virtualization devices and licenses are seldom cheap or free.*

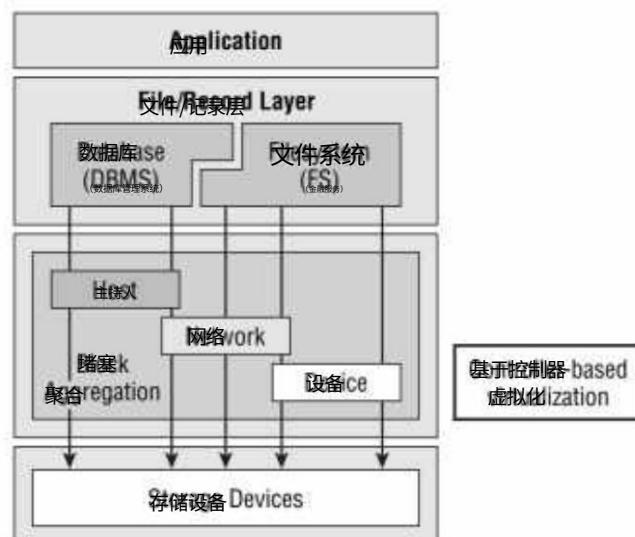
理论已经讲得够多了，让我们仔细看看最流行的存储虚拟化形式：基于控制器的虚拟化。

## 基于控制器的虚拟化 Controller-based Virtualization

首先，让我们将其映射到 SSM 模型。基于控制器的虚拟化发生在块聚合层内的设备层；是一种基于块的带内虚拟化形式，如图 9.8 所示。

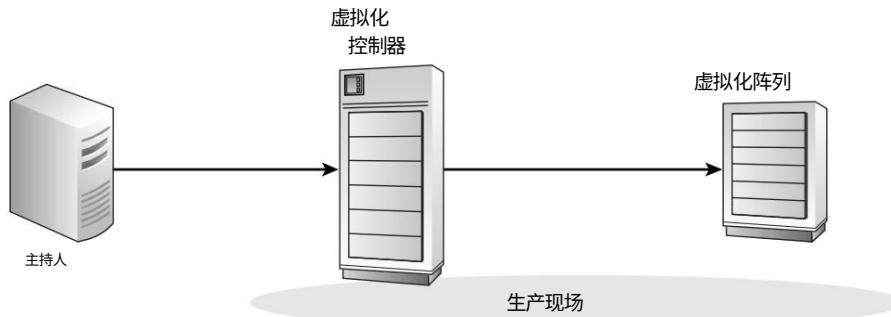
*the block aggregation layer, and is a form of *block-based in-band virtualization*. This is shown in Figure 9.8.*

图 9.8 基于控制器的虚拟化映射到 SSM



从高层次上讲，基于控制器的虚拟化是单个存储阵列的虚拟化 the virtualization of one storage array 在两个存储阵列后面如图 9.9 所示 as shown in Figure 9.9.

图 9.9 基于控制器的虚拟化



**在控制器虚拟化设置中,一个阵列充当主阵列,另一个充当从阵列。我们将主阵列称为虚拟化控制器,从阵列称为虚拟化阵列或被虚拟化的阵列。**

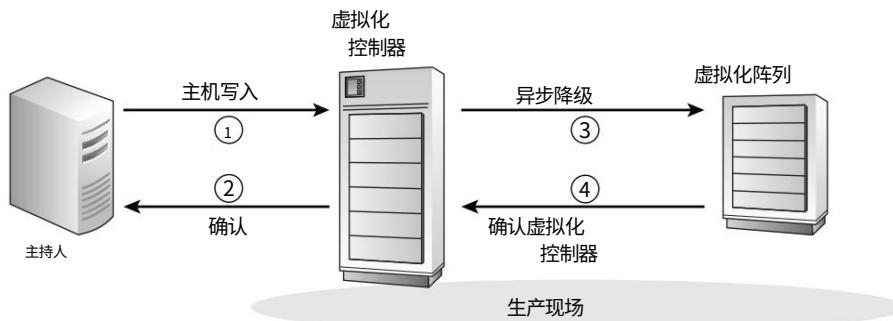
在基于控制器的虚拟化配置中,虚拟化控制器位于被虚拟化的阵列。就发出 I/O 的主机而言,虚拟化控制器是目标,是数据被发送的最终端点。主机不需要特殊的驱动程序或代理软件,并且完全不知道它正在与之通信的阵列实际上是一个中间人,而它的数据实际上被发送到另一个存储阵列进行持久存储。

所有智能和功能均位于虚拟化控制器中。被虚拟化的阵列仅提供受 RAID 保护的容量,并在其前面放置一些缓存。虚拟化阵列中可能本身可用的其他特性和功能均未使用,因此无需获得许可。

在大多数基于控制器的虚拟化设置中,当主机发出写入 I/O 时,它会将 I/O 发送到虚拟化控制器,虚拟化控制器会缓存 I/O 并将 ACK 返回给主机。然后,根据虚拟化控制器的缓存转储算法,写入 I/O 被延迟传输到正在虚拟化的阵列。虚拟化控制器通常在与其后面的虚拟化阵列通信时模拟主机,并向虚拟化阵列发出相同的写入数据,虚拟化阵列又根据其自己的缓存转储算法将其缓存并提交到磁盘。如图 9.10 所示。

由于虚拟化阵列在与其后面的虚拟化阵列通信时通常会模仿主机,因此虚拟化阵列上不需要特殊的软件或选项。就它们而言,它们只是连接到另一台主机,后者向其发出读写 I/O。例如,HDS VSP 平台支持基于控制器的虚拟化,并在与其后面虚拟化的阵列通信时模拟 Windows 主机。

图 9.10 向虚拟化存储阵列写入 I/O



HDS 和 NetApp 等供应商采用了存储虚拟机管理程序这一术语来描述他们的基于控制器的虚拟化平台。该描述在一定程度上是有效的。

接下来我们来看看一个典型的基于控制器的存储虚拟化配置并逐步介绍配置它的高级步骤。

## 典型的控制器虚拟化配置

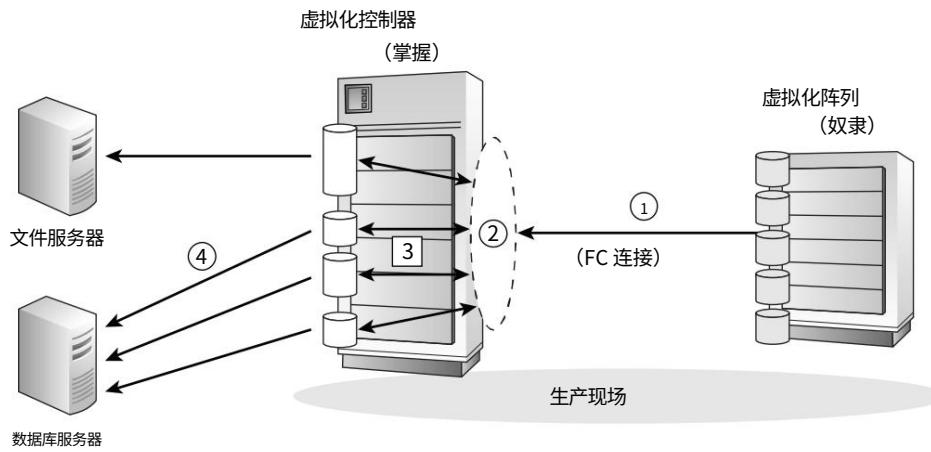
在本节中,您将了解配置基于控制器的虚拟化所需的典型高级步骤。这些步骤基于在 Hitachi VSP 阵列上配置基于控制器的虚拟化。尽管这些步骤是高级的,但它们可以巩固您对如何配置典型基于控制器的虚拟化设置的理解。

### 配置正在虚拟化的阵列

通常,要做的第一件事是配置要虚拟化的阵列。在此阵列上,您可以创建通常配置的任何 RAID 保护卷,并将它们作为 SCSI LUN 显示在前端端口上。您需要配置 LUN 掩码,以便虚拟化控制器的 WWPN 可以访问 LUN。这些 LUN 不需要任何特殊的配置参数,通常可以使用正常的缓存设置。根据虚拟化控制器的要求为这些 LUN 和前端端口配置任何主机模式选项非常重 要。例如,如果虚拟化控制器模拟 Windows 主机,则需要确保 LUN 和端口配置为预期 Windows 主机。

例如,图 9.11 显示了一个具有 100 TB 存储的虚拟化阵列,配置为五个 2 TB LUN,提供给四个前端端口。前端端口配置为 Windows 主机模式,LUN 掩码设置为允许虚拟化控制器端口的 WWPN。

图 9.11 配置要虚拟化的阵列



1. 虚拟化阵列上的 LUN 呈现给虚拟化控制器的 WWPN。
2. 虚拟化控制器认领这些 LUN 并将其用作存储。在本例中，它们形成了一个池子。
3. 从步骤 2 中创建的池中创建卷。
4. 步骤 3 中创建的卷将作为 LUN 从前端端口呈现给主机。



**至关重要的是**,没有其他主机访问从虚拟化阵列呈现到虚拟化控制器的LUN。如果其他主机在虚拟化控制器访问这些LUN时访问它们,这些LUN上的数据将被损坏。最常见的存储虚拟化配置是将虚拟化阵列中的所有存储呈现给虚拟化控制器。这样,任何主机都无需直接连接到虚拟化阵列。但是,可以将虚拟化阵列上的某些LUN配置为正常主机访问,而其他LUN由虚拟化控制器专用。重要的是,任何单个LUN都不应被两者访问。

## 配置虚拟化控制器

配置要虚拟化的阵列后,接下来配置虚拟化控制器。在这里,您将配置两个或更多前端端口(是的,前端端口)进入虚拟化模式。此虚拟化模式将端口从其正常目标模式转换为特殊形式的启动器模式,允许端口连接到正在虚拟化的阵列并发现和使用其LUN。为了保持简单,这些端口通常模拟标准Windows或Linux主机,以便正在虚拟化的阵列不需要进行任何特殊的配置更改。这使互操作性和支持变得非常简单。

## 连接虚拟化控制器和虚拟化阵列

连接通常为 FC,可以是直接连接或 SAN 连接。由于这些连接的重要性,许多人选择直接连接,以便尽可能保持两个阵列之间的路径简单干净。

当虚拟化控制器连接到正在虚拟化的阵列时,它会执行标准 PLOGI。由于其 WWPN 已添加到正在虚拟化的阵列上的 LUN 屏蔽列表中,因此它将发现并声明提供给它的 LUN,就像它使用本地安装的驱动器的容量一样。常见的例外是虚拟化控制器通常不会将 RAID 应用于发现的 LUN。RAID 等低级功能仍由虚拟化阵列执行。

从此时起,虚拟化控制器可以使用发现的 LUN,方式与使用其自己的内部卷完全相同。也就是说,外部卷可用于创建池和 LUN。

图 9.11 显示了向虚拟化控制器呈现五个 LUN 的虚拟化阵列。虚拟化控制器登录到虚拟化阵列,发现并声明 LUN,然后将它们组成一个池。然后,该池用于为四个新创建的卷提供容量,这些卷从虚拟化控制器的前端呈现给两台主机作为 SCSI LUN。这两台主机并不知道 LUN 的存储在它正在通信的阵列后面虚拟化。事实上,主机认为它正在与物理安装在它正在运行的服务器硬件中的物理磁盘驱动器通信!

虚拟化控制器中的本地存储称为内部存储,而被虚拟化的阵列中的存储通常称为外部存储。

## 综合起来

练习 9.1 以更简洁的格式回顾了前面的部分。

### 练习 9.1

#### 配置存储虚拟化

这些步骤概述了配置存储虚拟化的相当标准的过程。

您的环境中的具体过程可能有所不同,因此请务必查阅您的阵列文档。如有疑问,请咨询您的阵列供应商或渠道合作伙伴。我们在此介绍一个高级示例,以帮助您了解正在发生的事情。

1. 在被虚拟化的阵列上,执行以下任务:

1. 将后端存储划分为受 RAID 保护的卷。通常,会创建较大的卷(例如 2 TB),并将其作为 FC LUN 呈现在前端。

2. 在前端端口上显示这些 LUN,并将它们屏蔽到虚拟化控制器。

3. 为这些 LUN 配置任何标准缓存设置。

4. 配置虚拟化控制器所需的任何主机模式设置。例如,如果您的虚拟化控制器模拟 Windows 主机,请确保正确配置主机模式设置。

2. 在虚拟化控制器上, 将多个前端端口配置为外部端口 (虚拟化模式)。您应至少配置两个以实现冗余。
3. 既然虚拟化控制器以及被虚拟化的阵列上已经执行了基本任务, 那么就需要连接阵列。这些连接可以是直接连接, 也可以是 SAN 连接。重要的是它们可以互相看到并进行通信。
4. 一旦阵列连接并且可以互相看到, 请使用虚拟化控制器的 GUI 或 CLI 来发现从虚拟化阵列呈现给它的 LUN。

然后可以像虚拟化控制器中的任何其他磁盘资源一样配置和使用这些新发现的卷, 例如使用它们来创建池或导出的 LUN。

现在我们已经描述了基于控制器的虚拟化的工作原理, 让我们进一步了解一下仔细研究一些潜在的好处, 其中包括:

- 延长现有资产的使用寿命
- 为现有资产添加高级功能
- 增强自动分层等功能
- 简化迁移

## 延长现有阵列的使用寿命

虚拟化现有存储阵列的一大好处是可以延长其使用寿命。其理念是, 与其关闭旧阵列 (因为它已经过时, 不支持您所需的最新高级功能), 不如决定对其进行一次大脑移植, 将其虚拟化到较新的阵列中, 使其焕发新生。

这个想法在纸面上看起来不错, 但现实往往要复杂一些。  
即使在困难的经济环境下, IT 预算不断减少, 阵列也很少通过虚拟化来延长其使用寿命。

一些并发症包括:

- 必须保持虚拟化阵列的合同维护与  
它会继续服务于新的环境。如果旧的虚拟化阵列仍在为您的一级关键任务应用程序提供服务, 那么让旧的虚拟化阵列享受下一工作日 (NBD) 支持合同并不是最好的主意, 即使它是在另一个较新的阵列后面虚拟化的。
- 多供应商支持可能会很混乱, 而且会充满相互指责, 每个供应商都会指责对方, 而您很快就会一事无成! 如果您要虚拟化的阵列很旧, 那么这种情况会变得更具挑战性, 因为供应商通常不希望您长期维护旧产品, 而且他们通常会停止发布固件更新等。

经验表明, 虽然虚拟化阵列可以延长其寿命, 但在理论上, 现实往往更加复杂。

## 向虚拟化阵列添加功能

客户更常见的做法是购买新的一级企业级阵列和新的二级/三级中端阵列，并在企业级阵列后面虚拟化中端阵列。这种配置允许将一级阵列的高级功能和智能扩展到虚拟化二级/三级阵列提供的容量。

因为执行虚拟化的阵列对待虚拟化阵列的容量的方式与对待内部磁盘的方式相同，所以虚拟化阵列上的卷可以进行精简配置、重复数据删除、复制、快照、分层、虚拟机管理程序负载支持等等。

这种配置在现实世界中相当流行。

## 存储虚拟化和自动分层

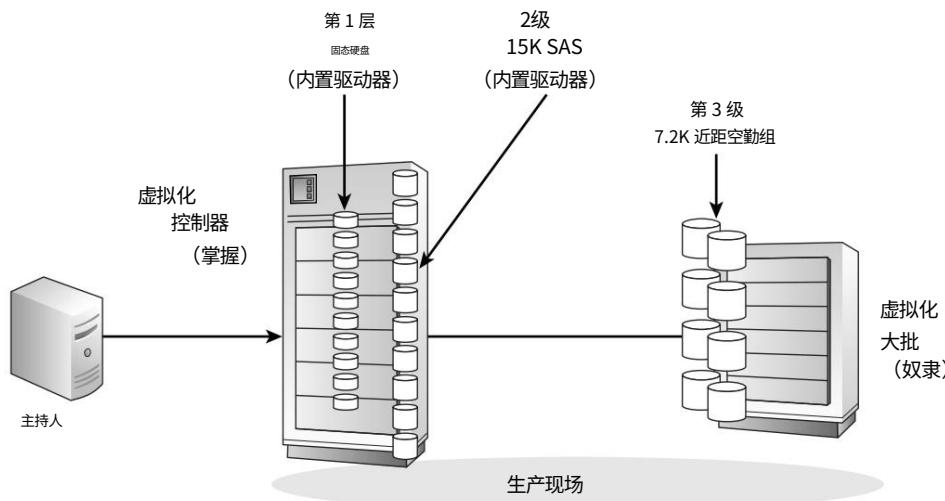
所有支持基于控制器的虚拟化的优质存储阵列都将允许虚拟化阵列中的容量成为可由其自动分层算法使用的存储层。

如果您的阵列支持此功能，那么使用虚拟化阵列的容量作为低层存储具有财务和技术上的合理性。

因为将物理磁盘放入第 1 层阵列的成本可能会让你眼花缭乱即使它是 4 TB NL-SAS 等低性能磁盘，仅用高性能驱动器填充第 1 层阵列的内部驱动器插槽也具有可靠的经济价值。除了填充内部驱动器插槽的高成本外，这些内部驱动器插槽还提供比在虚拟化阵列中访问的容量更低的延迟。因此，保留用于高性能驱动器的内部插槽并使用虚拟化容量作为廉价且深度的层，从财务和技术角度来看都是有利的。

这很好地导致了感觉自然的多层配置，如图 9.12 所示。

图 9.12 存储虚拟化和自动分层



将卷的热范围放在内部灰存储上、将暖范围放在内部高性能 SAS 上、将最冷和访问最不频繁的范围放在外部虚拟化阵列上的 7.2 K 或 5.4 K NL-SAS 上,这样做没有任何问题。

这种配置在现实世界中被广泛部署。

当然,这一切都假设你的虚拟化许可证成本不会高到破坏存储虚拟化的任何商业案例。

## 其他虚拟化注意事项

这是无法避免的:基于控制器的虚拟化会为您的配置增加一层复杂性。这不是可以逃避的事情,但在深入研究之前,您一定要意识到这一点。



### Real World Scenario

#### 存储虚拟化可能引起的并发症

说到复杂性,一家公司一直乐于使用存储虚拟化,直到一位高级存储管理员犯了一个简单的错误,导致了一场重大事故。存储管理员删除了旧的未使用的外部卷 - 正在虚拟化的卷。但是,虚拟化控制器为卷提供了十六进制 ID,而虚拟化阵列为卷提供了十进制 ID。长话短说,管理员混淆了十六进制和十进制数字,删除了一大堆错误的卷。这导致许多系统丢失了卷,导致许多人整夜辛苦工作。如果没有使用存储虚拟化,环境中就不会存在这种十六进制和十进制数字的轻微复杂性。

根据阵列实现存储虚拟化的方式以及配置方式,您可能会将资产投入一条难以退出的单行道。如果您后来决定存储虚拟化不适合您,请小心不要将自己锁定在一个难以更改的设计中。

如果设计和大小不合适,性能可能会成为问题。例如,如果您的虚拟化控制器中没有足够的缓存来处理工作负载和其后的任何虚拟化阵列提供的额外容量,您可能会陷入性能问题,随后需要进行昂贵的升级。此外,如果您正在虚拟化的阵列没有足够的驱动器来处理虚拟化控制器向其发送的数据,则很快就会导致性能问题。



## Real World Scenario

### 正确规划存储虚拟化的重要性

我之前曾与一家公司合作，该公司没有适当地确定其实时生产存储虚拟化解决方案的规模或配置。他们犯了两个根本性错误。

首先，他们没有在虚拟化的阵列中提供足够的性能资源（这次是磁盘驱动器不足）。这意味着在高峰时段，虚拟化阵列无法接收虚拟化控制器发送的大量数据。这导致数据在虚拟化控制器的缓存中备份，并迅速导致虚拟化控制器及其连接的所有主机陷入停滞状态。

他们已将虚拟化控制器上的缓存配置为虚拟化卷的写回缓存，但这并没有帮助他们解决问题。这意味着，当主机写入虚拟化阵列时，它们会很快确认，缓存开始填满。然而，随着缓存越来越满，将数据从缓存中取出并放到磁盘上变得很重要，但在这种情况下，磁盘位于虚拟化阵列上。正如我们前面提到的，虚拟化阵列规模太小，数据无法足够快地从缓存中抽出，导致巨大的缓存写入待处理情况，使虚拟化控制器陷入瘫痪。

最后，如果你不知道自己需要什么，虚拟化设置的成本可能会成为一个问题。让自己陷入其中。虚拟化许可证很少是免费的，并且它们的成本必须考虑到您为基于控制器的虚拟化制定的任何业务案例中！

## 软件定义存储

软件定义存储 (SDS) 是存储虚拟化的另一种形式。然而，让事情变得更加复杂的是，软件定义存储没有正式的定义，许多人都说“没有比软件定义更糟糕的术语了。”

对于很多供应商来说，这不过是一个营销热词，每个供应商都在拼命证明他们的产品是软件定义的。然而，尽管缺乏一个公认的定义，但仍有大量的设计和架构变化正在进行中，

承诺不仅重新定义存储,还重新定义整个数据中心!而且肯定有供应商在销售真正的软件定义存储产品!

我们将讨论一些下一代存储虚拟化技术,在我看来,它们就是软件定义的存储。无论如何,它们都是存储虚拟化的形式,与我们已经讨论过的不同。它们非常酷!

## SDS 解释

为了更好地解释软件定义存储,首先看一下传统的非软件定义存储会很有用。这样我们就可以进行比较。

### 传统非软件定义存储

在传统的非 SDS 存储阵列中,智能与物理硬件紧密耦合。它与硬件控制器紧密耦合,而硬件控制器又通常与物理驱动器和后端的其余部分紧密耦合。换句话说,提供智能和所有数据服务的固件只能在同一供应商的特定硬件控制器上运行。您无法将一个供应商的阵列中的微代码拿来在另一个供应商的阵列的控制器硬件上运行。您甚至无法拿来在行业标准的服务器硬件平台(如 HP ProLiant 和 Cisco UCS)上运行它。

在传统的非软件定义存储中,软件和硬件是结合在一起的  
在严格的一夫一妻制的关系中!



当谈到智能时,我们指的是 Intelliogl、高可用性、快照、复制、RAID、重复数据删除、精简配置、虚拟机管理程序卸载等等。在传统的非软件定义存储架构中,所有这些都以微代码实现,  
只能在特定硬件上运行。在软件定义存储中,这种智能可以在任何行业标准的硬件或虚拟机  
管理程序上运行。

许多传统阵列通过双控制器架构提供高可用性(HA)。如果去掉双控制器(特定硬件与特定控制器互连),您就无法再实现 HA。

对于传统阵列而言,智能与硬件的集成似乎还不够紧密,有些阵列更进一步,将部分智能实现在  
硬件,如 ASIC 和 FPGA 等定制硅片。这是最紧密的硬件和软件集成。

基本上,对于传统的存储阵列来说,没有硬件,软件就是无用的,  
没有软件,硬件就毫无用处。两者在工厂里就焊接在一起了!

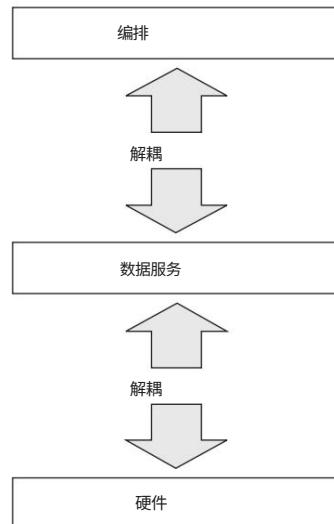
这种紧密的集成使得很难采用这些架构并将其重新设计为更灵活的 SDS 解决方案。但是,嘿,谁说他们需要这样做呢?这些紧密耦合的架构仍然在世界各地的企业中运行。

## SDS 方式

SDS 与传统阵列及其紧密耦合的智能和硬件完全相反。SDS 就是要将硬件和智能分离。它从硬件中抽取所有价值并将其转移到软件中。这使得硬件成为一种几乎无关紧要的廉价商品。

从架构角度来看,软件定义存储将存储分为三个层级 编排、数据服务和硬件,如图 9.13 所示。这些层级之间没有紧密集成,这意味着您可以将这三个层级混合搭配,从而让您在选择硬件、数据服务和编排技术时拥有更大的灵活性。

图 9.13 软件定义存储的三层



您可能会听到人们将软件定义存储 (SDS) 称为将控制平面与数据平面分离。这个术语更流行,更适合网络世界,因为控制平面和数据平面的概念在网络世界中得到更广泛的应用。此外,在网络世界中,由于来自不同供应商的设备之间的交互频率远高于存储世界,因此控制平面具有更多的通用性。

然而,其原理是一样的,都是指智能与硬件的分离。

SDS 有几个关键方面。最突出的功能之一是使用虚拟存储设备 (VSA) 来减轻硬件压力。API 是使用 SDS 的便捷功能。SDS 易于扩展,并与云和其他创新兼容。让我们更仔细地看一下这些要点。

一个流行的例子就是将智能从硬件中剥离出来。当然，一种软件定义的存储形式是虚拟存储设备 (VSA)。VSA 是完全智能存储控制器的一个实例，以虚拟机 (VM) 的形式实现为纯软件，运行在行业标准的虚拟机管理程序上，而虚拟机管理程序又运行在行业标准的 x64 硬件上。

VSA 可以利用其运行的物理服务器中的本地磁盘。它们还可以利用 DAS 磁盘托盘或甚至 iSCSI 和 FC 等共享存储的容量。许多 VSA 供应商建议将大型、非智能 JBOD（只是一堆磁盘）存储连接到它们以获得容量。

将存储控制器（智能）作为虚拟机运行，开辟了一个全新的可能性世界。例如，如果您需要向 VSA 添加资源，只需让虚拟机管理程序为其分配更多 CPU (vCPU) 或 RAM (vRAM)。当您不再需要分配给 VSA 的额外资源时，只需减少分配的 vCPU 和 vRAM 数量。我们甚至可以说，整个堆栈（计算、存储、网络、应用程序等）可以足够智能，在 VSA 需求高时增加 VSA 的资源，然后在 VSA 不再需要时再次将其撤走。一个简单的例子可能是 CPU 密集型的夜间后台处理重复数据删除任务。当作业开始时，运行相关脚本以添加更多 vRAM 和 vCPU，或允许系统动态分配资源。然后，当重复数据删除任务完成时，收回资源。这就是我所说的动态！

这些 VSA 可以在与您的主机和应用程序相同的虚拟机管理程序实例上运行，从而使存储更靠近应用程序。

惠普 (HP) 等老牌供应商以及 Nutanix、Nexenta、ScaleIO、SimpliVity 等初创供应商都拥有利用 VSA 软件定义存储模型的技术。

SDS 也是 API 驱动的。事实上，它不仅如此，还是开放 API 驱动的，包括 RESTful API。RESTful API 使用以下简单动词来执行功能：GET、POST、PUT 和 DELETE。SDS 的主要目标是让用户通过 API 与系统交互，以及让系统本身使用 API 进行自己的内部调用。这都是关于软件定义存储的 API。与更传统的非 SDS 存储相比，后者的管理是通过 CLI 和通常非常差的 GUI 进行的。

SDS 本质上是横向扩展的。再次以简单的 VSA 为例。如果您需要更多存储控制器，只需启动更多 VSA。这些可以通过前面提到的通用行业标准 API 进行管理，从而实现更简单、更集中的管理。不再需要登录到每个阵列来配置它。

SDS 拥抱云。由于存储层已经抽象出来，并且从物理上与智能分离，因此将云作为存储层或目标实现起来更为简单。传统阵列往往设计为将存储介质靠近控制器，控制器上的软件智能可以了解有关存储介质的细微细节。SDS 采用完全不同的方法，其中存储硬件

抽象化,使云集成变得更加简单和基础。SDS 解决方案采用 REST 等云 API,并提供 Amazon S3 和 OpenStack Swift 等集成。

软件定义模型的另一个好处是软件创新的速度

远高于硬件创新。基本上,软件创新比硬件创新更容易、更快、更便宜。那么为什么要把创新固定在硬件升级和发布上来减慢你的创新周期呢?

## 软件定义存储会统治世界吗?

虽然 SDS 的概念和一些实现确实很有趣而且确实有其适用之处,但它们很可能会长期与传统存储阵列共存很长一段时间。

毕竟,传统的、紧密耦合的架构

风靡一时,电器、交钥匙和预包装等术语也风靡一时。现在,如果你听那些在 SDS 领域耕耘农业的人说话,你会发现这些术语简直是亵渎神明。

但请不要忘记,传统方法,即全面测试、磨合,并在传统阵列的侧面贴上“专为 XYZ 设计”的标志,也有其优势。首先,您知道供应商已经对解决方案进行了极为细致的测试,这让您有信心在其上运行一级业务关键型应用程序。当出现问题时,您也可以只与一家供应商打交道,而不是让多家供应商互相指责。



有趣的是,有些供应商仅以软件形式提供 SDS 解决方案。Software-only SDS solutions as software-only. 其他公司将 SDS 解决方案作为软件和硬件包出售,其中包括控制器智能和硬件。这样做的主要原因是可支持性。虽然软件(智能)和硬件在这里作为单个包出售,但不必如此。这纯粹是为了简化支持,因为供应商将对其硬件上运行的软件进行广泛测试,并将其运行在实验室中以协助支持呼叫。

归根结底,世界各地的知名公司都不会放弃多年来一直为他们服务的东西。他们可能会先试用 SDS 解决方案,并在实验室和开发环境中实施,然后再将关键任务系统托付给他们。较新的公司更有可能在现有存储技术上零投资的情况下实现高度虚拟化,因此更有可能采取更大胆的方法并信任基于 SDS 的解决方案。

与所有事情一样,购买前先做好功课并试用!

## 章节概要

基于主机的存储虚拟化 基于主机的虚拟化通常采用主机上的逻辑卷管理器的形式,它将卷聚合并抽象为虚拟卷（称为逻辑卷）。卷管理器还提供快照和复制等高级存储功能,但它们的可扩展性有限,因为它们与单个主机绑定。大多数人并不认为基于主机的存储虚拟化是存储虚拟化的真正形式。

基于网络（基于 SAN）的存储虚拟化 基于网络的虚拟化如今更多的是一种概念而不是现实,因为该领域的大多数产品均未流行且已退役。

基于控制器的存储虚拟化 这是迄今为止最常见的存储虚拟化形式,由可以虚拟化下游阵列资源的智能存储阵列（或存储控制器）组成。基于控制器的存储虚拟化是一种带内、基于块的虚拟化形式。

软件定义存储 (SDS) 是任何仅以软件实现的存储控制器智能,可以安装在运行于行业标准服务器硬件上的行业标准虚拟机管理程序上,该虚拟机管理程序可以虚拟化底层商品存储并为其添加智能。

## 概括

在本章中,我们介绍了现实世界中常见的所有存储虚拟化实现,并集中介绍了最流行的形式——基于控制器的虚拟化。我们介绍了存储虚拟化的一些主要用例,以及如果操作不当可能会遇到的一些潜在问题。然后,我们在本章的最后讨论了软件定义存储 (SDS) 这一热门新主题,并将其与将存储智能与存储硬件紧密集成的更传统方法进行了比较和对比。

# 章节 10



## 容量 优化 技术

---

本章涵盖的主题：

- ✓ 精简配置
- ✓ 过度配置
- ✓ 数据压缩
- ✓ 重复数据删除
- ✓ 自动分层
- ✓ 子 LUN 自动分层



过去几年,存储行业见证了高质量容量优化技术的发展。这些技术包括精简配置、压缩、重复数据删除和自动分层。这一趋势源于多种因素,包括 IT 预算缩减、需要用更少的资源做更多的事情、存储技术的自然成熟,以及来自存储供应商的大力推动,这些供应商需要强大的容量优化技术,以便使其产品在价格上与基于旋转磁盘的解决方案具有竞争力。最终结果是,有大量高质量的容量优化技术可供选择。

本章介绍了所有这些主要的容量优化技术。它涵盖了诸如在线和后处理数据重复数据删除和数据压缩以及精简配置、过度配置和分层等主题。按照我们通常的方法,您将深入了解这些技术的工作原理和在现实世界中的实现方式,以及将一些技术映射到用例,例如备份和归档以及主存储。您将了解其中一些技术可能不是最好的,这样您就不会犯本来可以避免的错误。本章讨论了容量优化技术对性能、可用性和成本等关键原则的影响。阅读本章后,您应该能够在正确的位置部署正确的数据效率技术。

## 容量优化技术 概述

SNIA 使用术语“容量优化方法”来指代用于减少存储持久数据所需空间的所有方法。这几乎概括了这一点。容量优化技术就是用更少的资源做更多的事情:在更少的硬件上存储更多数据、充分利用存储资产、管理数据增长、从介质中榨干每一寸容量,等等。

虽然可能还有一些其他好处,比如性能改进,但这些都是附带好处和奖励。容量优化技术存在的理由是提高容量利用率,使您能够在更小的空间内存储更多数据。

在讨论提高产能利用率时,我们自然会讨论技术这将对投资回报率 (ROI) 和总拥有成本 (TCO) 等财务问题产生积极影响。这包括 \$/TB 采购成本、减少数据中心占用空间、减少网络带宽 (网络成本)、减少电力和冷却,以及更精简、更环保的数据中心。

## 性能影响

让我们花点时间来看看实施容量优化技术可能带来的一些性能影响。与大多数技术一样,容量优化技术可以提高或降低整体存储和系统性能。

例如,分层 (我们将其视为一种容量优化技术)可能会将数据下移到较低的层级,从而降低性能,或上移到较高的层级,从而提高性能。重复数据删除同样可以提高或降低性能。对基于旋转磁盘的主存储阵列上的数据进行在线重复数据删除通常会降低读写性能,而对具有大量读取缓存的虚拟机进行后处理重复数据删除则可以显著提高性能。精简配置 (TP) 也可以通过将数据分散到存储阵列后端的大量驱动器上来提高性能,但它也可能导致后端布局分散并对顺序性能产生负面影响。

在讨论每项容量优化技术时,我们将大量讨论性能问题,但目前重点应该是容量优化技术可以提高或降低性能。这完全取决于技术的实施方式。所以要小心。

## 容量优化技术的类型

有许多容量优化技术可用 事实上,数量太多,可能会让人感到困惑。为了澄清起见,让我们首先快速概述一下这些技术的类型,您将在本章中进一步探索这些技术:

精简配置是根据需要分配存储容量的过程,而不是预先保留大量空间以备两年后可能需要。

压缩是另一种较为知名的容量优化技术,即对数据进行重新编码以减小其大小。这与重复数据删除有着微妙但根本的区别。

重复数据删除是另一种更流行和更知名的容量优化技术,它是用对单个共享副本的引用替换相同数据的多个副本,最终目的是节省空间。

分层技术旨在将数据放置在适当的存储介质上,通常基于访问数据的频率,例如将不经常访问的数据移动到更便宜、更慢的存储中,将更频繁访问的数据放在更昂贵、性能更高的存储中。虽然这并不严格符合提高容量利用率的描述,但它确实优化了容量的利用方式。

容量优化技术还可以在各种地方实施:基于源、基于设备、网关、基于目标等等。它们也可以以内联、后处理甚至介于两者之间的方式实施。我们将在本章中介绍这些选项中的大多数。

## 容量优化技术的用例

容量优化技术的使用案例几乎是无限的。由于 CPU、固态存储和其他技术的进步，曾经因性能和其他开销而被视为禁忌的技术现在变得可行。

在接下来的几年里，我们将看到容量优化技术在几乎每个堆栈级别上得到实施，并包含在大多数主流存储技术中。

备份、归档、主存储、复制、快照、云，只要您说得出，如果还没有，那么很快就会有。

有很多内容需要讲，所以我们继续吧。

## 精简配置

在存储领域，精简配置 (TP) 是一项相对较新的技术，但却得到了热烈的欢迎和广泛采用。TP 迅速被采用的主要原因是它有助于节省资金，或者至少避免花钱。它通过按需分配模型来实现这一点，在该模型中，只有当实际写入数据时才将存储分配给卷，而不是预先分配大量存储。您很快就会更仔细地了解它的工作原理。

本章使用了术语“厚卷”和“薄卷”，但这些概念也称为传统卷和虚拟卷。

### 厚卷

由于厚卷是最古老和最常见的卷类型，让我们花点时间来回顾一下它们，为我们讨论更新、更复杂的薄卷主题做好准备。

厚卷很简单。你选择一个尺寸，卷立即保留 / 消耗大小定义的全部容量。假设您从具有 10 TB 可用空间的阵列创建 1 TB 厚卷。即使您没有向其中写入任何数据，该厚卷也会立即消耗 1 TB 并将阵列上的可用容量减少到 9 TB。这很容易理解，也很容易实现，但如果从不向该 1 TB 厚卷写入任何数据，这确实是一种空间浪费。在一个对有限存储池有着无尽需求的世界里，这种潜在的空间浪费是一个问题。而这是厚卷的主要问题，不能不解决。

厚卷让用户浪费存储空间的问题变得更加严重

人们通常高估自己所需的存储量，以防将来某个时候需要它。存储阵列通常被完全配置 将所有容量划分为厚卷并提供给主机 但实际上主机使用的容量不到一半。

另一半则闲置在那里。想象一下一个 200 TB 的存储阵列，其中所有 200 TB 都作为厚卷分配给主机，但所有主机写入的总数据总量

这些厚卷大约有 80 TB。这浪费了超过 100 TB 的存储空间。这种配置与使用容量的比率在很多地方仍然很常见。

然而,在很大程度上得益于精简配置,这个问题已经不再那么严重。

## 精简卷

精简卷的工作方式与厚卷有很大不同。精简卷在最初创建时占用很少的容量,有时甚至不占用容量。只有当数据写入其中时,它们才会开始占用空间。



**在大多数系统中,精简卷在创建时确实会占用少量空间。这些空间用于在主机写入时为数据准备一个空间区域,而无需系统经过查找可用空间并将其分配给精简卷的过程。并非所有系统都这样工作,但许多系统都是这样工作的。** without the system having to go through the process of locating free space and allocating it to the thin volume. Not all systems work like this, but many do.

让我们快速了解一下 1 TB 精简卷的工作原理。当我们创建这个 1 TB 精简卷时,它最初占用的空间很少或根本不占用空间。如果我们将其导出到的主机从不写入它,它将永远不会再占用任何空间。假设一位热切的开发人员认为她的新业务应用程序需要一个新卷来处理它将生成的大量数据并请求一个 2 TB 的卷作为存储管理员,您从具有 10 TB 可用空间的阵列中分配一个新的 2 TB 卷。但是,该应用程序最多只生成 200 GB 的数据(远远达不到预期的数据量)并请求一个 2 TB 的卷。当该应用程序写入 200 GB 时,使用的 1.8 TB 空间仍可供阵列分配给其他卷。这显然是一个强大的概念。

### 文件系统并不总是删除数据

必须了解的是,由于许多应用程序和文件系统的运作方式,文件系统报告的已使用空间量并不总是与存储阵列认为已使用的空间量相匹配。例如,如果您有 100 GB 的精简卷,并且应用程序或用户向该卷上的文件系统写入 100 GB,这将显示为在您的 1 TB 精简卷中消耗了大约 100 GB 的数据。现在假设您的用户或应用程序删除了 100 GB 数据中的 50 GB,文件系统将报告仅使用了 50 GB,而精简卷通常仍会显示它消耗了 100 GB。如果该用户或应用程序继续写入另外 50 GB 的数据,则精简卷将报告消耗了 150 GB,即使文件系统仅显示使用了 100 GB。也就是说,如果您的用户或应用程序然后删除 50 GB 的 100 GB 的数据,则精简卷将报告消耗了 150 GB,即使文件系统仅显示使用了 100 GB。

原因是,当用户或应用程序删除数据时,大多数文件系统实际上并不会删除数据。相反,它们只是将删除的数据标记为不再需要,但将数据保留在原处。在这方面,存储阵列不知道任何数据已被删除,这意味着它无法回收和重新使用文件系统中已释放的任何空间。此外,大多数文件系统会将新数据写入可用区域,并且只有在文件系统中没有剩余可用空间时才会覆盖已删除的数据。此行为在文件系统中称为向前移动,并导致 TP 感知存储阵列将新范围分配给 TP 卷。

幸运的是,大多数存储阵列和大多数现代文件系统(例如最新版本的 NTFS 和 ext)都具有精简配置意识并实施了 T10 UNMAP 等标准,以便存储阵列能够感知文件系统删除操作并释放存储阵列内已删除的空间。

当谈到精简配置时,需要理解的一个重要概念是扩展区大小。  
扩展大小是将数据写入薄卷时应用于薄卷的增长单位。

例如,假设系统的 TP 区大小为 768 KB。每次主机将任何数据写入 TP 卷的未使用区域时,都会以 768 KB 的倍数为 TP 卷分配空间。如果区大小为 42 MB,情况也是如此;任何时候将数据写入 TP 卷的新区域时,都会为该卷分配 42 MB 的区。

如果您的主机只向 TP 卷写入了 4 KB 的数据,那么 42 MB 的扩展可能看起来有点过度,而且从容量角度来看,非常大的扩展大小肯定不如小的扩展大小高效。但是,较小的扩展大小可能会带来相关的性能开销,因为系统在将数据写入卷时必须执行更多的扩展分配。假设主机向新的 TP 卷写入 8 MB 的数据。如果扩展大小为 42 MB,系统只需分配一个扩展,而如果扩展大小为 768 KB,则必须分配 11 个扩展,这可能是 11 倍的工作量。在大多数情况下,较小扩展大小的容量相关优势被认为比使用较小扩展大小的潜在性能开销更为显著,因为在现实世界中,任何潜在的性能开销都非常小,以至于很少甚至从未被注意到。

有时您可能会听到将 TP 范围称为页面。



### Real World Scenario

确保在进行重大更改之前进行测试

一家大型零售公司正在部署一款支持 TP 的全新存储阵列。作为部署的一部分,该公司正在将主机从几个较旧的低端存储阵列迁移到新的支持 TP 的阵列上。在迁移过程中,他们注意到数据库备份和刷新作业似乎占用了

在崭新的阵列上花费的时间更长。乍一看,情况不应该如此。新阵列拥有更快的处理器、更多缓存、更多磁盘和宽条带 TP 卷。

在有确凿证据证明新存储阵列上的备份和刷新作业比旧存储阵列上花费的时间更长后,我们打电话给供应商解释原因。

根本原因是 TP 范围的分配方式是随机的。

由于 TP 区是按需分配给 TP 卷的,因此它们随机分散在后端的所有主轴上。这种随机布局对顺序备份和刷新操作的性能产生了负面影响。针对多种工作负载类型进行了广泛的测试,结果证明(在这种特定的 TP 实现上),对于大量顺序工作负载,TP 卷的性能不如传统的厚卷。

然后实施了一种解决方法,即所有用于备份和刷新操作的 TP 卷都将预先分配。使用此选项(在创建新的 TP 卷时可用),所有扩展区都预先确定并且是连续的,尽管在将数据写入其中之前仍不会分配。这种预分配与使用厚卷略有不同,因为尽管构成预分配 TP 卷的所有扩展区都是预先保留的,但在将数据写入其中之前仍不会分配。但是,这种解决方法解决了该问题。也可以通过将备份和刷新卷创建为厚卷来解决该问题。

## 过度配置

大多数基于精简配置的存储阵列都允许管理员对阵列进行过度配置。让我们看一个例子来解释过度配置的工作原理以及它的优缺点。

让我们从一个精简配置存储阵列开始,该阵列具有 100 TB 的物理容量(扣除 RAID 开销等)和 50 台连接主机。现在让我们假设为这 50 台主机配置的总存储容量为 150 TB。显然,这些卷必须是精简配置的。如果不是,则可以从阵列配置的最大容量将是 100 TB。但是因为我们已经配置了 150 TB 的容量,而阵列只安装了 100 TB 的实际物理容量,所以我们对阵列进行了过度配置。从过度配置的角度来看,这个阵列是 50% 的过度配置,或者 150% 的配置。



大多数精简配置阵列(支持精简配置并能够创建精简配置卷和过度配置的阵列)也支持创建传统厚卷的功能。因此,阵列通常可以同时具有厚卷和薄卷。这些阵列通常允许您在线在厚卷和薄卷之间进行转换,而不会影响或导致卷所呈现到的主机停机。

过度配置有一些明显的好处,但也有一些缺点。

过度配置允许阵列假装其容量大于实际容量。这可以提高容量利用率。在我们之前的示例中,阵列具有 100 TB 的物理可用存储,并且过度配置了 50%。该阵列可能仍严重未得到充分利用。完全有可能,尽管已为连接的主机配置了 150 TB,但这些主机可能只向阵列写入了 60 TB 到 70 TB 的数据。

现在,我们假设该阵列的存储成本为 2,000 美元/TB,这意味着

阵列的成本为 200,000 美元。如果我们将阵列进行了密集配置,它将只能配置 100 TB 的容量。为了满足额外 50 TB 的请求,我们将不得不购买至少 50 TB 的额外容量,成本为 100,000 美元。但是,由于阵列进行了精简配置,我们可以零成本对其进行过度配置,以满足 50 TB 的额外需求。因此,过度配置具有明显的经济效益。

然而,过度配置的缺点是,您的阵列将面临容量不足的风险,无法满足所连接主机的存储需求。例如,假设我们的阵列只有 100 TB 的物理可用存储,而我们已配置了 150 TB。如果所连接的主机需要超过 100 TB 的存储,则阵列将无法满足这一需求。因此,主机将无法写入卷的新区域,这可能会导致主机和应用程序崩溃。您不希望发生这种情况,避免这种情况需要仔细规划和预见。只有当您拥有可用的工具和历史数字来了解容量使用趋势时,才应进行过度配置。

您还应该了解获得采购订单批准和供应商提供额外容量所需的前置时间。如果您的公司需要很长时间才能签署采购订单,并且您的供应商也需要很长时间才能运送新容量,那么您就需要将这一点纳入您的超额供应政策中。

由于刚才提到的风险,最终负责 IT 服务的人员必须意识到并同意承担这些风险。话虽如此,如果管理得当,过度配置可能很有用,而且在现实世界中越来越常见。

## 热门内容

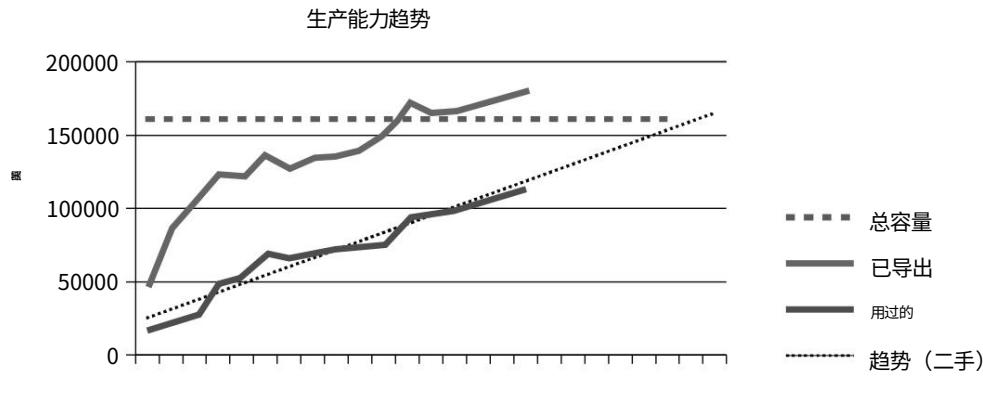
正如我所提到的,在过度配置阵列的环境中,趋势分析的重要性,让我们花点时间来看看一些对趋势分析很重要的指标。此列表绝不是详尽无遗的,但它涵盖了过度配置的存储管理员通常关注的指标:

- 物理可用容量
- 已配置/已分配容量
- 已使用容量

作为趋势分析的一部分,您需要至少每月跟踪一次这些指标。

图 10.1 显示了示例图表,该图表跟踪了过去 16 个月的指标,并添加了一条趋势线,预测未来 6 个月使用容量预计如何增长。

图 10.1 过度配置跟踪图



正如你所见,图 10.1 中的数组将破坏银行 (已用容量将超过大约在 2015 年 2 月,您将能够看到所有可用容量 (即物理可用容量)。当然,您也希望在 2014 年 11 月之前将更多容量安装到阵列中。这些图表和趋势图在所有过度配置环境中都至关重要。定期与管理层分享它们。将它们打印出来、压成薄片并钉在您的桌子上,这样您就不会忽略它们了!

## 采购订单的触发点

如果您计划在环境中过度配置,则需要确定购买新容量的触发点。这些触发点是促使您购买新容量的条件。

常见的触发点基于已用容量。例如,您可能有一个触发点,规定一旦已用容量达到物理可用容量的 90%,您就开始订购更多存储。如果您使用 90% 的容量作为触发点,并且您有一个具有 200 TB 物理可用存储的阵列,则一旦已用容量达到 180 TB,您就会开始安装新容量的过程。

至关重要的是,不要等到为时已晚才获得更多容量。在某些组织中,安装新容量的过程可能相当漫长。您可能包括获取正式报价、提出采购订单 (PO)、遵循采购订单签收流程、向供应商下订单和采购订单、等待零件交付、等待零件安装等等。

以下是大多数组织使用的三个最常见的触发点:

- 可用的未使用容量
- 可用容量百分比
- 过度配置百分比

## 确定容量购买的触发点

在本分步指南中,您将了解决定容量购买触发点的场景。

1. 您部署一个具有 200 TB 可用存储空间的存储阵列并配置所有卷  
作为精简配置卷 (TP 卷)。

2. 使用阵列六个月后,您为连接的主机配置了 150 TB 的存储空间,但发现只有 50% 的存储空间得到了使用 (已配置 150 TB, 已使用 75 TB)。

3. 阵列现在处于稳定状态,这意味着您不再向其中迁移大量新主机,因此您预计从现在开始它的增长将是稳定的。

4. 经过另外三个月的使用,您已经配置了 180 TB, 但只使用了 95 TB (仍然接近使用了 50%)。

按照目前的速度,您每月平均配置 10 TB, 因此两个月后将需要一个新的存储阵列。但是,您决定不购买新的存储阵列,而是超额配置阵列。

毕竟,它仍有 85 TB 的容量尚未使用。

5. 此时,您决定获得高级管理层的认可和批准,以启用超额配置。您提供显示增长率和可用空间 (当前未使用的物理可用容量) 的相关统计数据和图表。作为批准的一部分,您同意管理层在您超过 25 TB 可用空间界限时批准购买新的存储容量。决定使用 25 TB 这个数字是因为您知道您当前的稳定运行率是每月增加约 5 TB 的使用空间,并且公司需要长达六周的时间来批准大型 IT 基础设施支出。您的存储供应商还需要另外两周的时间来运送额外的存储空间。您还向管理层解释说,当您的可用空间越来越少时,每月使用存储量异常增加的影响可能会很严重。例如,如果您有 50 TB 的未使用空间,而开发人员决定使用 15 TB 的意外空间,那么您可以接受,但如果只有 10 TB 的未使用可用存储空间,而同一个开发人员决定使用 15 TB,那么就会发生巨大的混乱。

6. 在管理层同意对阵列进行过度配置后,您可以对阵列进行必要的配置更改,并持续监控和趋势利用率。

7. 又过了六个月,您从仅剩 180 TB 物理可用存储空间的阵列中配置了 225 TB, 这导致阵列的配置过度了 25%。

您的已用容量现在也为 125 TB。

8.现在您可以看到,在过去 9 个月中,您使用的容量每月增加约 5.5 TB。按照此速度,您将在 12 到 13 个月内用尽可用的物理空间,因此阵列中仍有足够的增长空间。

9.再过九个月,你的数组看起来如下:已用空间现在为

175 TB,这意味着您只剩下 25 TB 的可用空间,而且至关重要的是,阵列中只剩下大约四到五个月的增长空间,之后空间就会用完,您就会丢掉工作。因此,您按下按钮为阵列购买新的容量,而且由于您不想今年不得不去乞求更多的预算资金,因此您购买的容量足以再使用 12 个月多一点(可能可用容量为 96 TB)。

选择不同的触发器指标甚至使用多个触发器的原因有很多。例如,对于具有高达 200 TB 物理可用存储的中型阵列,20 TB 的可用空间可能效果很好。20 TB 是 200 TB 阵列的 10%。但是,如果您的阵列具有 800 TB 的可用物理存储,那么 20 TB 突然就只占总阵列容量的 2.5%。而且,较大的阵列往往连接了更多的主机,因此增长速度比较小的阵列更快,这意味着 20 TB 可能在连接了少量主机的小型阵列上持续一段时间,但在连接了数百台主机的大型阵列上不会持续那么长时间。

## 空间回收

尽管精简配置具有巨大的优势,但随着时间的推移,精简卷会变得越来越薄,这被称为膨胀。如果没有办法消除膨胀,最终会导致精简卷用尽所有可能的空间,使它们比厚卷好不了多少。这时,空间回收就可以派上用场了,它为我们提供了一种保持精简卷薄的机制。

造成臃肿的主要原因之一是大多数传统文件系统不理解

精简配置,在删除数据时不通知存储阵列。让我们看一个简单的例子。假设一个 100 GB 的文件系统位于一个 100 GB 的精简配置卷上。如果主机将 90 GB 的数据写入文件系统,阵列会看到这一点并将 90 GB 的数据分配给卷。如果主机随后删除了 80 GB 的数据,文件系统将显示 10 GB 的已用数据和 90 GB 的可用空间。但是,阵列不知道主机刚刚删除了 80 GB 的数据,并且仍会显示精简卷已使用 90%。这种膨胀远非理想。

现在让我们看看空间回收如何帮助避免这种情况。

## 零空间回收

空间回收(通常称为零空间回收或零页面回收)

是一种识别已删除数据的技术,可以将保存已删除数据的范围释放回阵列上的空闲池,在那里它们可以用作其他卷的空间。



*This is the traditional space recovery technology that is still used today.*  
传统上,空间回收技术要求将已删除的数据清零(用二进制零覆盖),然后阵列可以扫描连续的二进制零区域并确定这些区域代表已删除的数据。然而,更现代的方法使用TRIM或UNMAP等技术,允许主机/文件系统向阵列发送信号,告诉它哪些区可以回收。

## 传统空间回收

过去,操作系统和文件系统在删除数据时很懒惰。

文件系统不会将删除的数据所在的空间清零,而是将该区域标记为已删除,但将数据保留在那里。从操作系统的角度来看,这是件好事,因为它比实际将数据清零的工作量要少,而且它使恢复已删除的数据变得容易得多。然而,这对精简配置存储阵列来说毫无用处。就阵列而言,数据仍然在那里。在这些情况下,需要使用特殊工具或脚本将二进制零写入磁盘,以便阵列可以识别已删除的空间并将其回收用于其他地方。

一旦将二进制零写入文件系统的已删除区域,该数组将要么动态识别这些零并释放阵列内的空间,要么需要管理员手动在阵列上运行空间回收作业。这两种方法的结果相同——增加了阵列上的可用空间——但其中一种方法需要管理干预,另一种则不需要。

回收主机上的零空间并不太难。练习 10.1 将指导您如何执行此操作。

### 练习 10.1

#### 回收非 TP 感知主机和文件系统上的零空间

假设我们向主机提供了一个 250 GB 的 TP 卷,并使用单个 250 GB 文件系统进行格式化。随着时间的推移,主机向文件系统写入了 220 GB 的数据,但删除了 60 GB,从而使文件系统剩下 160 GB 的已用空间和 90 GB 的可用空间。

但是,由于文件系统不支持 TP,因此支持 TP 的存储阵列无法识别这 60 GB 的已删除数据。为了让存储阵列识别并释放空间,您需要执行以下步骤:

1. 用二进制零填充 90 GB 的可用文件系统空间。执行此操作时,您可能希望留出几 GB 的空闲空间,以防主机在您用零填充文件系统的同时尝试写入数据。无论出于何种目的,文件系统都将零视为普通数据,因此如果文件系统被零填充,其行为与被普通用户数据填充时完全相同;它会填满并且不允许写入更多数据。
2. 将零写入文件系统后,立即删除您创建的全是零的文件,以便文件系统有空间供新用户写入。

- 3.在存储阵列上,启动手动零空间回收操作。此操作扫描卷以查找与阵列的 TP 扩展大小相匹配的连续零组。例如,如果阵列的 TP 扩展大小为 8 MB,则阵列将需要找到 8 MB 的连续零。每次找到 100% 为零的 TP 扩展时,它都会从 TP 卷中释放 TP 扩展,并将其提供给系统中的其他卷。

整个 TP 区必须充满零,才能从 TP 卷释放并提供给其他卷。因此,如果您的 TP 区大小为 1 GB,则需要 1 GB 的连续零才能释放区。如果您的区大小为 768 KB,则只需要 768 KB 的连续零即可释放区。

### UNMAP SCSI 标准

如果所有这些看起来都是释放空间的大量工作,那么您是对的,尤其是在拥有数千台主机的大型环境中。幸运的是,存在一种更巧妙的方法。大多数现代操作系统都包含旨在与基于阵列的精简配置配合使用的文件系统。

T10 (INCITS 标准机构内负责制定 SCSI 标准的技术委员会)制定了一项称为 UNMAP 的 SCSI 标准,该标准专门用于允许主机和存储阵列相互传达与精简配置相关的信息。例如,当主机操作系统从文件系统中删除数据时,它会向阵列发出 SCSI UNMAP 命令,通知它已删除的范围。然后,阵列可以对这些数据采取行动,从卷中释放这些范围,并将它们放回阵列空闲池中,其他主机可以使用它们。

这比手动运行脚本来写零要容易得多!

幸运的是,大多数现代操作系统(如 RHEL 6、Windows 2012 和 ESXi 5.x)都附带精简配置友好的文件系统并实施 T10 UNMAP。



*In certain situations, T10 UNMAP may cause host performance degradation on hosts. You should test performance with T10 UNMAP enabled or disabled. Your environment. An ext4 filesystem allows you to mount a filesystem in either mode—T10 UNMAP enabled or T10 UNMAP disabled. Following your testing, you should decide whether there will be instances in your estate where you want T10 UNMAP disabled.*

### Inline Space Reclamation

如前所述,一些存储阵列能够监视传入的二进制零流并执行以下操作之一 incoming streams of binary zeros and do either of the following:

- 如果它们针对的是卷中的新扩展区,则不将它们写入磁盘(主机可能正在对卷执行完全格式化) if the zero is targeted to a new extent on the volume (the host may be performing a full format on the volume)
- 如果范围已分配,则释放它们所针对的范围 if (主机正在运行零空间回收脚本)ed (the host is running a zero space reclaim script)

这是内联空间回收的一种形式。它通常只能通过以下方式启用和禁用：存储管理员，因为它可能会影响性能。

较旧的阵列往往需要管理员手动运行空间回收作业来释放主机上已清零的空间。这是后处理回收的一种形式。虽然两种方法都有效，但从管理角度来看，内联方法要简单得多。

## 精简配置问题

就像生活中的大多数事情一样，没有完美的解决方案可以解决所有问题，精简配置也不例外。虽然它带来了很多好处，但也有一两点需要注意：

- 与厚读相比，大量顺序读取工作负载的性能较差卷。
- 每次添加新空间时，都可能产生少量的写入性能开销分配至精简卷。

正如我一直提到的，如果你确实打算过度配置，请确保保持你的趋势并更新预测图表。否则，您可能会面临空间不足的风险，并导致整个庄园一片混乱。

请注意：过度配置不会简化容量管理或存储管理。它会让管理变得更加困难，并给您作为存储管理员带来压力。要获得潜在的好处，精简配置和过度配置需要付出一些努力。

## 压缩

压缩是一种容量优化技术，它使您能够使用更少的存储容量来存储更多数据。例如，10 MB 的 PowerPoint 演示文稿可以压缩到 5 MB，这样您就可以仅使用 5 MB 的磁盘空间来存储该 10 MB 演示文稿。压缩还允许您更快地通过网络传输数据，同时消耗更少的带宽。总而言之，它是一种很好的容量优化技术。

压缩通常基于文件，许多人都熟悉 WinZip 等技术，它允许您压缩和解压缩单个文件和文件夹，对于通过通信网络传输大型文件特别有用。其他类型的压缩包括文件系统压缩、基于存储阵列的压缩和备份压缩。文件系统压缩的工作原理是压缩写入文件系统的所有文件。

基于存储阵列的压缩通过压缩写入存储阵列中卷的所有数据来实现。而备份压缩则通过压缩正在备份或写入磁带的数据来实现。

压缩主要有两种类型：

- 无损压缩
- 有损压缩

顾名思义，无损压缩不会丢失任何信息，而有损压缩则会丢失一些信息。您很可能想知道，如果有损压缩会丢失信息，它还有什么用呢？一种流行的有损压缩形式是 JPEG 图像压缩。大多数人都熟悉 JPEG 图像，但我们都不认为它们是质量低劣的图像，尽管 JPEG 图像是使用有损压缩算法进行压缩的。此类有损压缩技术的工作原理是删除不会影响数据整体质量的细节。（对于 JPEG 而言，这是图像的整体质量。）人眼通常无法识别压缩文件和非压缩文件之间的差异，即使压缩图像丢失了很多细节。人眼和大脑就是注意不到这种差异。当然，有损压缩并不是万能的。例如，您当然不希望您的银行余额小数点前丢失任何零 - 当然，除非您的余额恰好是赤字！

## 压缩的工作原理

压缩的工作原理是减少文件或数据流中的字节数，之所以能做到这一点，是因为大多数文件和数据流都会一遍又一遍地重复相同的数据模式。压缩的工作原理是重新编码数据，这样生成的数据使用的位数就会比源数据少。

一个简单的例子是，一个角落里有一个大的蓝色方块的图形。在文件中，这个蓝色区域可以定义如下：

矢量 0,0 = 蓝色

矢量 0,1 = 蓝色

矢量 0,2 = 蓝色

矢量 0,3 = 蓝色

矢量 0,4 = 蓝色

矢量 0,5 = 蓝色

矢量 0,6 = 蓝色

矢量 0,7 = 蓝色

...

矢量 53,99 = 蓝色

一个简单的压缩算法会采用这种重复模式并将其更改为类似下面的内容：

矢量 0,0 至 53,99 = 蓝色

这很短但却告诉我们同样的事情，而不会丢失数据（因此是无损的）。

我们所做的只是确定一个模式，并将该模式重新编码为告诉我们完全相同的事情的其他内容。我们在压缩过程中没有丢失任何东西，但文件占用的位数和字节数更少。

当然,这只是举个例子来说明这一点。实际上,压缩算法节奏比这先进得多。



**压缩已压缩的数据通常不会导致输出变小。事实上,压缩已压缩的数据有时会导致输出稍大。**音频、视频和其他富媒体文件通常使用内容感知压缩技术进行本地压缩。因此,尝试压缩 MP3、MP4、JPEG 或类似文件通常不会产生有价值的结果。

主存储压缩并没有真正引起轰动。它确实存在,并且有一些用例,但其采用速度至少可以说很慢。



**主存储用于存储正在使用的数据。主存储的示例包括 DAS、SAN 和 NAS 存储,它们为实时应用程序、数据库和文件服务器提供服务。非主存储的示例(通常称为近线存储或离线存储)包括备份和归档存储。**

性能是主存储压缩尚未风靡全球的主要原因。一些压缩技术会产生不良副作用,即降低存储性能。但是,如果操作得当,压缩可以提高旋转磁盘和固态介质的性能。性能在主存储领域非常重要,因此在您的资产中部署压缩技术之前务必小心谨慎。确保对其进行测试,并确保让数据库管理员、Windows 和 Linux 管理员,甚至开发人员参与其中。

除了主存储之外,压缩在备份和归档领域也非常流行。

有两种常见的基于阵列的压缩方法(对于主存储,以及备份和存档):

■ 内联

■ 后期处理

让我们看一下这两者。

## 在线压缩

内联压缩会在数据被发送到磁盘之前,先在缓存中对其进行压缩。这会消耗 CPU 和缓存资源,如果您碰巧遇到特别高的 I/O,则压缩和解压缩操作可能会增加 I/O 延迟。再次提醒,购买前先试用,测试、测试、再测试!

从积极的一面来看,内联压缩减少了写入后端的数据量,这意味着所需的容量更少,无论是内部总线还是后端驱动器。存储内部总线所需的带宽量

执行内联压缩的阵列比不执行内联压缩的系统占用的空间要小得多。这是因为数据在通过内部总线发送到后端驱动器之前在缓存中进行了压缩。

## 后处理压缩

在后处理压缩领域,数据必须以未压缩的形式写入磁盘,然后在稍后进行压缩。这避免了压缩操作对性能的任何潜在负面影响,但也避免了对性能的任何潜在正面影响。此外,后处理压缩的主要缺点是它要求您有足够的存储容量来以未压缩的形式存储数据,直到稍后对其进行压缩。

无论压缩是在线完成还是在后期处理中完成,解压缩显然必须实时在线完成。

## 性能问题

不可避免地,性能在主存储系统中非常重要。与压缩和主存储相关的一个主要性能考虑因素是解压已压缩数据所需的时间。例如,当主机发出对某些数据的读取请求时,如果该数据已在缓存中,则从缓存中提供该数据,这称为缓存命中。缓存命中速度极快。但是,如果读取的数据尚未在缓存中,则必须从磁盘读取。这称为缓存未命中。缓存未命中速度很慢,尤其是在基于旋转磁盘的系统上。

现在让我们将解压缩加入其中。每次发生缓存未命中时,我们不仅必须从后端的磁盘请求数据,还必须解压缩数据。根据您的压缩算法,解压缩数据可能会给该过程增加显着的延迟。因此,许多存储技术在容量和性能之间进行权衡,并使用基于 Lempel-Ziv (LZ) 的压缩技术。LZ 压缩并不是世界上最好的压缩率,但它的开销相对较低,并且可以快速解压缩数据!

## 压缩的未来

压缩在主存储领域前景越来越光明。现代 CPU 现在具有原生压缩功能和足够的空闲周期来处理压缩等任务。此外,闪存等固态技术正在推动压缩的采用。毕竟,压缩允许在闪存中存储更多数据,而闪存有助于避免压缩技术带来的某些性能影响。

尽管主存储中压缩技术的前景光明,但请记住在部署到生产环境之前进行测试。压缩技术改变了事物的工作方式,您需要在部署之前了解这些变化的影响。

## 重复数据删除

如果将重复数据删除归结为最基本的要素,您会发现它就是识别重复数据并确保仅存储一次。实施重复数据删除技术的存储系统通过检查数据并检查系统中是否已存在数据的副本来实现这一点。如果此数据的副本已经存在,则使用指针指向数据的副本,而不是存储其他副本。这在图 10.2 中进行了描述。

图 10.2 用指针替换重复数据

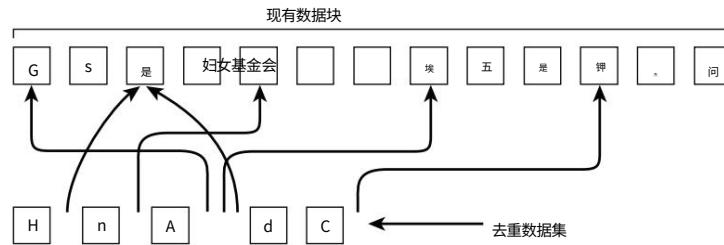


图 10.2 的顶部是现有的存储数据池。该池下方是新的数据已写入系统的数据集。此数据集已进行重复数据删除,因为数据集中任何已存在于现有数据池中的块都会从新数据集中删除,并替换为指针。

当用单个副本以及每个额外副本的指针替换相同数据的多个副本时,神奇的是指针几乎不占用任何空间,尤其是与它们要替换的数据相比时。如果图 10.2 中示例中的每个方块大小为 16 KB,每个指针为 1 KB,则我们节省了 90 KB。六个指针中的每一个仅消耗 1 KB,总共有 6 KB 可用于存储指针,而不是存储六个数据块所需的  $6 \times 16$  KB。

重复数据删除与压缩类似,但又有细微的不同。重复数据删除是指将之前见过的数据模式联系起来,但与压缩不同,这些模式不必在文件或数据集中彼此接近。重复数据删除适用于文件和目录,甚至可以适用于 PB 级大小的整个系统。

下面是一个简单快捷的例子。在本章的手稿中,单词 storage 已被使用了 204 次(本章投入生产时可能会发生变化)。一个简单的重复数据删除算法可以取单词 storage 的一个实例并删除其他所有实例,用指向原始实例的指针替换它们。在这个例子中,我们假设单词 storage 消耗 4 KB,而指针仅消耗 1 KB。如果我们不进行重复数据删除,并坚持使用该单词的 204 个实例,我们将消耗 816 KB ( $204 \times 4$  KB)。但如果我们将进行重复数据删除,对该单词的 204 次引用将仅消耗 207 KB(初始副本为 4 KB,加上其他 203 个指针的  $204 \times 1$  KB)。现在让我们假设我们的重复数据删除适用于本书的所有章节(每个章节都是一个单独的 Microsoft Word 文档);我们将节省更多空间!

显然这是一个过于简单的例子,但它有助于传达这个概念。

重复数据删除与压缩并无太大区别。它已广泛应用于备份和存档用例，并且从历史上看，基于旋转磁盘的主存储系统采用重复数据删除的速度很慢。然而，目前市场上的许多全闪存存储阵列都实现了数据重复数据删除。



**NOTE** 了解已删除重复数据的数据也可以进行压缩对您太有裨益。但是，已压缩的数据通常无法进行删除。重复数据，这在设计解决方案时非常重要。*that has been compressed cannot normally be deleted.*

例如，如果您压缩数据库备份，然后将其存储在重复数据删除备份设备上，则您不应该期望看到压缩备份实现良好的重复数据删除率。

## 基于块的重复数据删除

基于块的重复数据删除旨在在系统中仅存储唯一块。虽然重复数据删除的粒度各不相同，但粒度越小通常越好——至少从容量优化的角度来看是这样。

在重复数据删除技术中，通常提到两大类粒度：

- 文件级别
- 块级别

文件级粒度，有时称为单实例存储 (SIS)，寻找在 le 级别上的精确重复，实际上不应该称为重复数据删除。

块级重复数据删除的粒度比整个文件小得多，有时被 SNIA 称为子块。由于其粒度比文件细得多，因此可以实现更高的重复数据删除率和效率。

文件级重复数据删除（单实例）是一种非常低级的重复数据删除技术，您应该清楚自己购买的是什么！另一方面，块级重复数据删除（块粒度约为 4 KB）通常被认为是最好的。



**NOTE** 重复数据删除的计算方式为删除前数据量 ÷ 删除后数据量，并以比率表示。例如，如果我们的数据集在删除重复数据之前为 10 GB，删除重复数据之后减少到 1 GB，则我们的重复数据删除比率计算是  $10 \div 1 = 10$ ，表示为 10:1。这意味着原始数据集是删除重复数据/存储数据大小的 10 倍。换句话说，我们已实现 90% 的空间减少。

让我们仔细看看单实例和块级重复数据删除。

## 文件级单实例

说实话,文件级重复数据删除(也称为单实例)是一种相当粗糙的重复数据删除形式。为了将两个文件视为重复,它们必须完全相同。例如,假设本章的 Microsoft Word 文档为 1.5 MB。

如果我保存该文件的另一个副本,但删除该副本中该句子末尾的句号,则这两个文件将不再是完全相同的,需要将两个文件单独保存到磁盘,占用 3 MB 空间。但是,如果不删除句号,则这两个文件将完全相同,并且可以完全删除重复数据,这意味着只会将一个副本存储到磁盘,仅占用 1.5 MB 空间。

这个例子并不是很极端。但是,如果我在周一创建一个 10 MB 的 PowerPoint 文件,然后在周二更改一张幻灯片并将其保存到不同的文件夹中,会发生什么情况?这样,我将有两个文件无法进行重复数据删除,并且总共占用大约 20 MB 的空间,尽管这两个文件除了一张幻灯片的微小更改外完全相同。现在假设我在周三编辑同一个演示文稿中的另一张幻灯片并将其保存到另一个文件夹中。现在我有三个几乎相同的文件,但都单独存储,占用大约 30 MB 的存储空间。这很快就会失控,并凸显出文件单实例的主要弱点。它虽然有效,但很粗糙,效率不高。

## 定长块重复数据删除

在块级别进行重复数据删除比在文件级别进行重复数据删除要好得多。

块级重复数据删除比文件级重复数据删除更细粒度,从而能够获得更好的效果。

块级重复数据删除的最简单形式是固定块,有时也称为固定长度段。固定块重复数据删除采用预定的块大小(例如 16 KB),并将数据流切分为该块大小的段(在我们的示例中为 16 KB)。

然后它检查这些 16 KB 段并检查是否已经有它们的副本。

让我们将其付诸实践。我们将坚持使用 16 KB 的固定块大小,并使用我用来撰写本章的 1.5 MB Word 手稿。1.5 MB Word 文档为我们提供了 96 个 16 KB 块。现在让我们保存两份几乎相同的手稿副本。

两个副本之间唯一的区别是我在其中一个副本中删掉了这个句子末尾的句号。为简单起见,我们假设这个缺失的句号出现在第 65 个 16 KB 块中。使用固定块重复数据删除技术后,两个文件中的前 64 个 16 KB 块(1,024 KB)将完全相同,因此将被删除重复数据。因此,我们只需为 64 个 16 KB 块分别存储一份副本 - 这样可节省 1 MB 的存储空间。但是,其余 32 个 16 KB 块各不相同,因此我们无法对它们进行重复数据删除(用指针替换它们)。最终结果是,我们可以保存手稿文件的两个几乎相同的副本,这样只占用 2 MB 的存储空间,而不是 3 MB。这很好,但可以更好。

那么,之所以所有省略句点的块都会被识别为唯一块而不是重复块,是因为固定块重复数据删除的性质。固定块重复数据删除方法通常情况下,只要您更改

数据集中的一个块发生偏移后,所有后续块都会发生偏移,不再进行重复数据删除。这是固定块重复数据删除的主要缺点,如图 10.3 所示。

图 10.3 定长重复数据删除

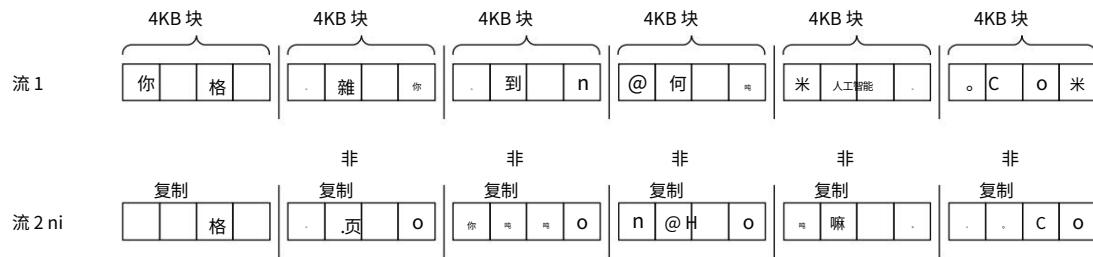


图 10.3 中的示例显示了两个几乎相同的电子邮件地址。唯一不同的是  
ence 是流 2 中电子邮件本地部分的句点。如您所见,在包含句点之后的每个块都会偏移  
一个位置,并且接下来的块都不相同,这意味着无法进行重复数据删除。

### 可变块大小

可变块(也称为可变长度段)重复数据删除方法解决了固定块开始遇到的问题。可变块方法在分割数据集时应用动态浮动边界,从而进一步减少对块进行更改的影响。它们还不仅根据大小来决定每个块的开始和结束位置,还通过识别数据本身的重复模式来决定。这意味着当插入或删除数据时(如我们的周期示例),可变块方法在看到重复模式时将与现有数据重新对齐。大多数情况下,可变块大小重复数据删除技术将比固定块和单实例产生更高的存储效率。

### 哈希和指纹识别

让我们看一下索引和确定块重复的最常见方法。

发送到重复数据删除系统并存储在其中的每个数据块都会被赋予一个数字指纹(唯一哈希值)。这些指纹会被编入索引并存储起来,因此可以非常快速地查找。它们通常存储在缓存中。如果索引丢失,通常可以重新创建,尽管重新创建过程可能很长,并且在重新创建期间系统可能会停止服务。

当新的写入进入重复数据删除系统时,这些新的写入会被分段,然后对每个分段进行指纹识别。指纹会与系统中已知指纹的索引进行比较。如果找到匹配项,系统就会知道传入的数据是重复(或潜在重复)块,并使用指向系统中已存储的块副本的更小的指针替换传入的块。



根据数字指纹识别系统的强度,系统可以选择在每次识别出重复块时执行逐位比较。在逐位比较中,块中所有可能重复的位都会与系统中已存储的块中的位进行比较。执行此比较是为了确保数据确实是重复的,而不是所谓的冲突,即两个不同的数据块产生相同的指纹。采用弱指纹识别的系统需要执行更多的逐位比较操作,因为弱指纹识别系统会产生更高的冲突率。从积极的一面来看,弱指纹识别系统通常具有较低的性能开销,只要您可以忍受执行更多逐位比较的开销,就没问题。全闪存存储阵列就是执行逐位比较的开销非常低的一个例子,因为从闪存介质读取数据的速度非常快。

## 重复数据删除的位置

数据可以在不同的地方进行重复数据删除。在典型的备份用例中,重复数据删除通常采用以下三种方式中的一种或多种:

- 来源
- 目标
- 联合

让我们依次看一下每个选项。

### 基于源的重复数据删除

基于源的重复数据删除(有时称为客户端重复数据删除)需要在主机上安装代理软件,并且重复数据删除的繁重工作在主机上完成,消耗主机资源,包括CPU和RAM。但是,基于源的重复数据删除可以显著减少源和目标之间消耗的网络带宽量 存储更少,移动更少!这在备份场景中尤其有用,因为大量数据在源和目标之间通过网络传输。

由于其消耗基于主机的资源,且其可进行重复数据删除的数据池有限(它只能对同一主机上存在的数据进行重复数据删除),因此它通常被视为一种轻量级的首次重复数据删除技术,可有效减少网络带宽。因此,它通常与基于目标的重复数据删除技术相结合,以提高重复数据删除率。

## 基于目标的重复数据删除

基于目标的重复数据删除,有时也称为基于硬件的重复数据删除,在备份领域已经流行了很长时间。在基于目标的重复数据删除中,重复数据删除过程发生在目标机器上,例如重复数据删除备份设备。

这些设备往往是专用设备,具有自己的 CPU、RAM 和持久存储(磁盘)。这种方法减轻了主机(源)的重复数据删除负担,但对减少源和目标之间的网络带宽消耗毫无作用。

重复数据删除备份设备还常常用于对重复数据删除备份设备对之间复制的数据进行重复数据删除或压缩。市场上有来自不同供应商的大量重复数据删除备份设备。

## 联合重复数据删除

联合重复数据删除是一种分布式重复数据删除方法。它实际上是基于源和目标的组合,可为您提供两全其美的效果——最佳重复数据删除率以及减少源和目标之间的网络带宽。

这种联合方法,即源和目标都执行重复数据删除,可以  
通过减少源和目标之间的网络带宽来提高性能,同时潜在地提高目标重复数据删除设备的吞吐能力。后者是通过  
在数据到达目标之前在源处有效地进行预重复数据删除来实现的。

Symantec OpenStorage (OST) 等技术可实现备份软件和备份目标设备之间的紧密集成,包括在源和目标  
处进行重复数据删除。

现在,许多备份解决方案都提供完全联合的方法来  
重复数据删除,重复数据删除同时发生在源和目标上。

## 何时进行重复数据删除

我已经提到过,重复数据删除过程可以发生在源端或目标端,或者两者兼而有之。但就基于目标端的重复数据删除而言,重复数据删除过程可以是以下任一方式:

- 内联
- 后期处理

让我们看一下这两种方法。

### 内联重复数据删除

在内联重复数据删除中,数据在被提取时和在后端提交到磁盘之前都会被实时删除。内联重复数据删除通常在缓存中对数据进行重复数据删除。我们有时将此称为同步重复数据删除。

内联重复数据删除要求所有哈希（指纹）和潜在的逐位比较操作必须在数据提交到磁盘之前进行。轻量级哈希更简单、执行速度更快，但需要更多逐位比较，而强哈希计算量更大，但需要更少的逐位比较操作。

内联重复数据删除的主要优势在于它只需要足够的存储空间来存储已删除重复的数据。无需大型着陆区来存储未删除重复的数据，直到稍后进行重复数据删除。

内联重复数据删除的潜在缺点是它可能会对数据摄取率产生负面影响，即设备处理传入数据的速率（以 MBps 为单位）。如果系统必须对进入系统的所有数据进行分段、指纹识别和重复检查，则很有可能会对性能产生负面影响。

虽然内联重复数据删除对于小型、中型和大型企业都非常有效，但它对于小型企业和小型站点尤其有益，因为以下情况属实：

- 您希望将硬件占用空间保持在最低限度，这意味着您希望部署尽可能少的存储硬件。
- 您的备份窗口 您可以执行备份操作的时间窗口 ups 不是问题，这意味着您有充足的时间执行备份。

在这些情况下，内联重复数据删除可能是一个不错的选择。首先，您只需部署有足够的存储空间来保存重复数据删除后的数据。此外，由于您不必将备份作业压缩到很短的时间窗口内，因此无需担心内联重复数据删除过程对备份吞吐量的潜在影响。

## 后处理重复数据删除

后处理重复数据删除是异步执行的。基本上，数据在写入磁盘时不会进行重复数据删除，然后在稍后的某个时间点对数据运行重复数据删除过程以删除重复数据。

在后处理模型中，所有哈希和逐位比较操作都在数据被提取后进行，这意味着数据重复数据删除过程不会直接影响设备提取数据的能力（MBps 吞吐量）。然而，这种方法的缺点是需要更多的存储空间，因为您需要足够的存储容量来容纳未重复数据删除形式的数据，同时等待稍后进行重复数据删除。

大多数备份相关的重复数据删除设备都执行内联重复数据删除，因为大多数现代重复数据删除设备都具有大量的 CPU、RAM 和网络带宽，能够相对好地处理内联重复数据删除。



使用后处理重复数据删除技术作为备份解决方案时要小心，因为后台（后处理）重复数据删除操作可能无法在下一轮备份开始之前完成。在大型环境中尤其如此，因为大型环境的备份窗口很小，而用户必须备份大量数据。

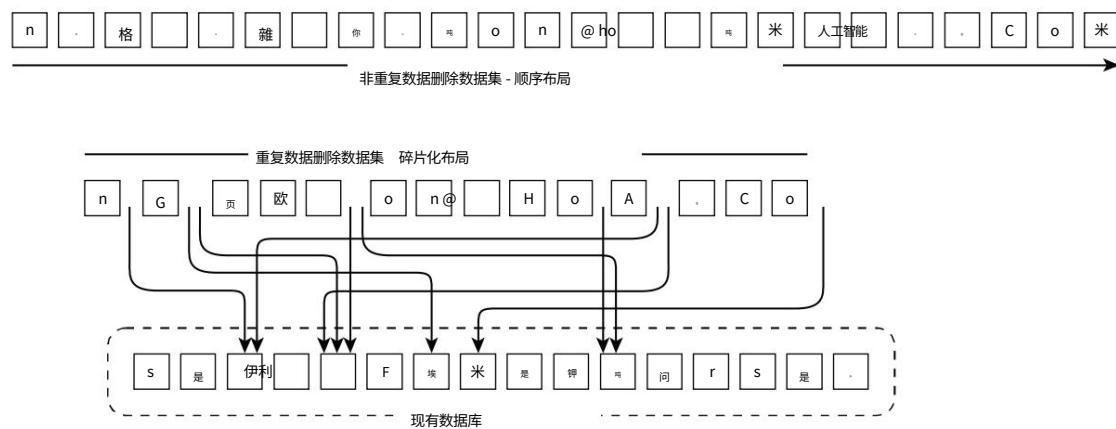
## 备份用例

重复数据删除技术现在已成为几乎所有良好备份解决方案的支柱。它们与大多数备份计划所包含的数据类型和重复数据模式非常吻合。在一个月内,大多数人会一次又一次地备份相同的文件和数据。例如,如果您的主目录每月进行四次完整备份,则每个月至少会备份相同的文件四次,这还不考虑上述四次完整备份之间可能发生的任何增量备份。将这样的数据发送到重复数据删除设备通常会产生出色的重复数据删除效果。

对备份相关数据进行重复数据删除还可以延长数据保留时间,因为您可以从存储中获得更多可用容量。您的重复数据删除设备中可能拥有 100 TB 的物理存储空间,但如果获得的重复数据删除率为 5:1,则您实际上拥有 500 TB 的可用容量。此外,一般来说,您保留数据(备份保留)的时间越长,您的重复数据删除率就越好。这是因为您可能会备份越来越多的相同数据,从而使您能够获得越来越好的重复数据删除率。

但是,重复数据删除技术往往会使文件布局变得碎片化。重复数据由指向后端其他位置的指针引用。图 10.4 显示了重复数据删除之前数据集的潜在顺序布局,以及重复数据删除之后的同一数据集。虽然重复数据删除版本的文件系统占用的空间要少得多,但读回重复数据删除数据集的全部内容将比读回未重复数据删除数据集的全部内容困难得多,而且需要的时间也多得多。对于基于旋转磁盘的系统尤其如此,但对于固态系统来说,问题要小得多。

图 10.4 重复数据删除造成的碎片



数据布局的碎片化对备份相关技术(如合成完整备份)有直接影响。合成完整备份是一种不以正常方式将所有数据从源发送到目标的完整备份。相反,合成完整备份

备份的创建方式是获取目标上已存在的上一次完整备份,然后应用自该完整备份以来发生的任何增量备份,从而有效地创建新的完整备份,而无需读取源上的所有数据并通过网络将其发送到目标。虽然合成完整备份如今很流行,但它们对重复数据删除系统来说是一个挑战。为了创建合成完整备份,必须从重复数据删除设备读取上一次完整备份以及此后进行的任何增量备份,同时重复数据删除设备必须写出新的合成完整备份。所有这些读取和写入(包括大量繁重的随机读取,因为重复数据删除后的数据在系统中是碎片化的)可能会严重影响重复数据删除设备的性能。

## 虚拟化用例

重复数据删除非常适合虚拟化技术,例如服务器和桌面虚拟化。这些环境往往能获得良好的重复数据删除率,因为 x 个虚拟 Windows 服务器将共享大量相同的核心文件和块。Linux 等其他操作系统也是如此。

例如,如果您拥有一个 10 TB 的 Windows 服务器映像,并从该映像部署 100 台服务器,那么这 100 台服务器将共享大量相同的文件和块。这些文件和块显然可以很好地进行重复数据删除。如果您拥有大型读取缓存,这样的重复数据删除还可以带来潜在的性能改进,因为许多共享文件和块将被所有虚拟机高度引用(频繁读取),从而使它们保留在读取缓存中,从而可以非常快速地访问它们。如果这 100 台虚拟服务器中的每一台都必须访问我们存储阵列中磁盘上各个位置自己的核心文件,那么这将要求我们的存储阵列在后端磁盘上进行大量查找和读取工作。这将导致性能不佳。但是,如果对这些核心文件和块进行了重复数据删除,并且只频繁访问主副本,则可以从缓存内存中以比每次读取都必须转到后端磁盘快得多的速度访问它们。

## 主存储和云用例

全闪存阵列的兴起已开始将重复数据删除技术推到主存储用例的前沿。传统上,由于性能影响,主存储系统避免使用重复数据删除。然而,这种性能影响主要是由于旋转磁盘的性质及其较差的随机读取性能。毕竟,良好的随机读取性能对于以下两个与重复数据删除相关的操作至关重要:

■ 执行逐位比较操作 ■ 对重复数据删除后的数据执行读取

### 操作

如果逐位比较操作需要很长时间,那么它们将影响系统的整体性能。此外,由于重复数据删除过程通常会导致数据碎片化,因此读取重复数据删除数据往往需要大量随机读取。全灰阵列不会出现随机读取性能问题,这使得主存储中的重复数据删除技术非常可行。事实上,大多数全灰阵列都支持内联

重复数据删除，并且它与一些全灰阵列集成得非常紧密，以至于它们不允许您关闭该功能！

第 3 层和云也是重复数据删除技术大有裨益的地方，因为从较低层存储（尤其是从云）访问数据的速度通常很慢。在此基础上增加重复数据删除的开销并不是性能方面的主要因素，但它可以显著降低容量利用率。

## 重复数据删除摘要

重复数据删除在某些方面与压缩类似，但并不完全相同。也就是说，重复数据删除和压缩可以结合使用以节省空间，但您需要了解以下几点：

- 重复数据删除后的数据可以进行压缩。
- 压缩数据通常无法进行重复数据删除。

如果要结合使用这两者，则先进行重复数据删除，然后再进行压缩。



### Real World Scenario

#### 混合压缩和重复数据删除

一家大型银行购买了几台大型重复数据删除备份设备，以增强其现有的备份和恢复流程。其想法是先将备份发送到重复数据删除备份设备，然后再复制到磁带，以便更快地完成恢复操作。当数据库备份在发送到重复数据删除备份设备之前开始被压缩时，该解决方案就失效了。最初，数据库备份在复制到重复数据删除备份设备之前已写入 SAN 卷，但正如通常的情况一样，随着数据库的增长，为其分配了更多的存储空间。但是，预算中没有足够的资金来扩展数据库备份写入的 SAN 卷。

相反，我们决定压缩数据库备份，以便它们仍然能够容纳未增加大小的 SAN 卷。

问题在于，该组织规模庞大，而且结构混乱，数据库人员没有与备份人员讨论这一决定，他们自己也不知道压缩数据不会进行重复数据删除。经过一段时间的运行，重复数据删除设备无法再对数据库备份进行重复数据删除，人们注意到重复数据删除设备上的重复数据删除率正在下降，空间比预期的要早得多。

通过在数据库服务器上部署数据库备份代理解决了这种情况，允许它们直接备份到重复数据删除备份设备，而无需写入中间 SAN 卷，因此无需压缩。

某些数据类型的重复数据删除效果会比其他数据类型更好;这就是生活。

重复数据删除往往会导致数据布局随机/碎片化,这对旋转磁盘驱动器的性能有负面影响,但对于闪存和其他固态介质来说这不是问题。

文件级重复数据删除是一种低效的重复数据删除技术,无法产生良好的重复数据删除效果。能够在较小块尺寸下工作的可变块重复数据删除技术往往是最好的。

## 自动分层

几乎所有部署多层旋转磁盘(有时还部署少量灰存储)的基于磁盘的存储阵列都支持子 LUN 自动分层。虽然它不能像重复数据删除和压缩技术那样增加可用容量,但它绝对可以优化存储资源的使用。

自动分层有助于确保数据位于正确的存储层。其目的通常是将频繁访问的数据放在闪存或 15K 驱动器等快速介质上,将不频繁访问的数据放在速度较慢、更便宜的介质上,例如高容量 SATA 和 NL-SAS 驱动器,甚至可能放在云端。

最终目标是将昂贵的高性能资源用于最活跃的数据,并且希望最重要的是数据,同时将不常访问的数据保存在速度较慢、更便宜的介质上。



如果您的存储阵列是全闪存阵列,则几乎肯定不会使用自动分层技术,除非它在不同等级的闪存(例如 SLC、MLC 和 TLC)之间分层数据。然而,市场上很少有全闪存阵列这样做,尽管一些供应商正在积极考虑这样做。

## 子 LUN 自动分层

自动分层的早期实现是在卷级别上进行的;分层操作(例如在层上上下移动)作用于整个卷。例如,如果某个卷的一部分被频繁访问并且需要提升到更高的存储层,则必须将整个卷上移。例如,如果某个卷为 200 GB,并且只有 2 GB 被频繁访问并且需要移动到第 1 层,但自动分层导致整个 200 GB 都被移动,则这种方法可能会极大地浪费更高层中的昂贵资源。

这时,子 LUN 自动分层技术就派上用场了。子 LUN 分层无需在层中上下移动整个卷。它的工作原理是将卷切片并切分为多个较小的区段。然后,当只有部分卷需要向上或向下移动层时,只有与这些区域相关的区段

卷需要移动。假设我们的 200 GB 卷被分成多个区,每个区为 42 MB (大多数高端日立阵列都采用这种方式)。这次,如果我们的 200 GB 卷有 2 GB 的热数据 (经常访问),这些数据可以通过移动到第 1 层获得好处,那么阵列将只移动 42 MB 的热区,而不必移动冷区 (不经常访问) 42 MB 区。这是一个更好的解决方案!

显然,较小的扩展区大小往往比较大的扩展区大小更高效,至少在有效利用我们每个层中的容量方面是如此。一些中档阵列具有较大的扩展区大小,在 1 GB 的范围内。移动这样的 1GB 扩展区比移动整个卷要好,但如果您需要在层上或层下移动的只是 10 MB 的卷,那么仍然可以改进。在这种情况下,更小的扩展区大小 (例如 42 MB 甚至更小) 显然应该更有效。较小的扩展区大小在层间移动时也需要更少的系统资源。例如,在层上或层下移动 1 GB 的数据比移动少量 42 MB 扩展区需要更多的时间和资源。

但是,较小的区段大小也有其弊端。它们需要更多的系统资源 (如内存) 来跟踪它们。

## 存储层

在分层领域,三似乎是一个神奇的数字。绝大多数分层存储配置 (也许除了专用混合阵列) 都是由三层存储构建的。

具体要将哪种类型的存储放入每个层级,但常见的方法是将第 1 层级设为固态存储 (闪存),将第 2 层级设为快速旋转磁盘 (例如 10K 或 15K SAS 驱动器),将第 3 层级设为大型、慢速 SATA 或 NL-SAS 驱动器。但是,您拥有的每个层级的数量以及类似的问题高度依赖于您的环境和工作负载。如果可能,您应该与供应商和技术合作伙伴密切合作,以确定哪种存储最适合您的环境。

图 10.5 显示了现实世界中常用的三层设计。百分比值表示每个层占整个系统容量的多少。它们不是指每个层占整个磁盘驱动器的百分比。

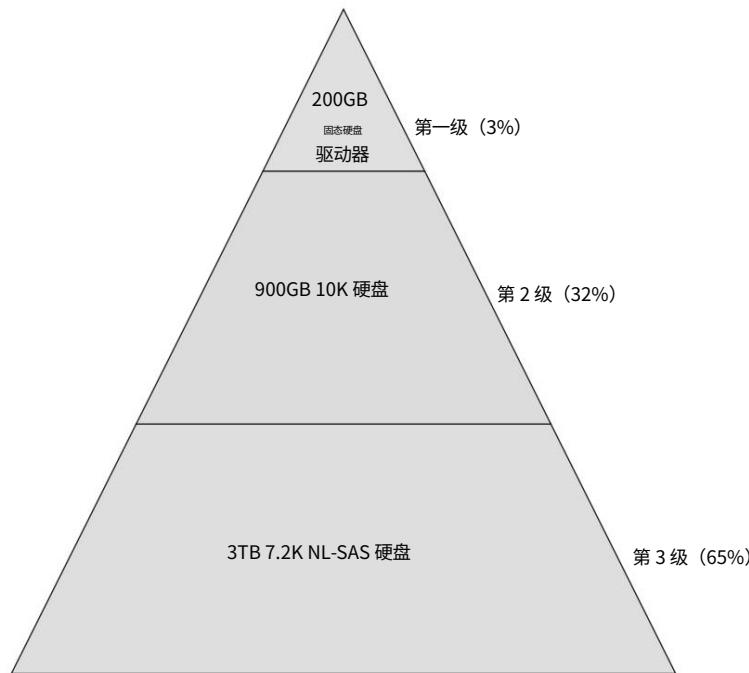
## 活动监控

为了做出与分层相关的决策 (例如将哪些数据区移上层以及将哪些数据区移下层),存储阵列需要监视阵列上的活动。此监视通常涉及跟踪后端哪些数据区访问最多以及哪些数据区访问最少。但是,监视此活动的时间对于获得所需的分层结果至关重要。例如,许多组织选择仅在核心工作时间 (即阵列为业务线应用程序提供服务的时间) 监视数据。通过这样做,您可以确保移至最高层的数据区是您的关键业务线应用程序使用的数据区。相反,如果您在工作时间之外监视系统 (例如,当您的备份运行时),则最终可能会得到

阵列将与备份操作相关的范围移动到最高层的情况。

大多数情况下,您不希望备份相关的卷和范围占用昂贵的高性能存储层,而您的业务线应用程序必须使用更便宜、性能较低的存储层。

图 10.5 层级数量金字塔图



允许您微调监控周期的阵列和系统比那些控制力较弱的系统要好。您对环境及其支持的业务了解得越多,您就越能更好地调整监控周期等因素,您的分层操作结果也会越好。

## 分层时间表

与微调和自定义调度监控窗口类似,也可以对数据在层之间移动的时间进行调度。与创建自定义监控周期一样,定义数据在层之间移动的周期也应根据您的业务进行定制。毕竟,在层之间上下移动数据会消耗系统资源,您可能不希望在工作日的中间时段将这些资源用于分层操作,因为您的应用程序需要这些资源来为客户销售提供服务并执行类似的功能。

安排分层操作(在各层之间上下移动数据)的常见方法是让它们在工作时间之外进行 - 例如,从凌晨 3 点到凌晨 5 点 - 当您的大多数系统处于安静状态或正在备份时。

## 配置分层策略

除了自定义监控和移动窗口外,通常还需要

分层策略。一些常见且有效的策略包括:

■除外责任

■限制

■优先事项

排除功能允许您执行诸如从所有分层操作中排除某些卷之类的操作,从而有效地将这些卷固定到单个层中。它们还允许您执行诸如排除某些卷以使其不占用某些层中的空间之类的操作。例如,您可能不希望与备份相关的卷占用最高层。

限制允许您限制特定卷或卷集在特定层级中可以占用的量。

当某一特定层级存在争用时,优先级可让您优先处理某些卷。例如,您的主要销售应用程序可能与研究应用程序争夺第 1 层灰烬存储。在这种情况下,您可以为与销售应用程序相关的卷分配更高的优先级,使这些卷优先于其他卷,并增加与销售应用程序相关的卷获得第 1 层空间的机会。

当然,并非所有阵列和系统都支持所有功能。在购买之前,请先做好研究  
假设您选择的技术可以完成所有上述任务。



### Real World Scenario

#### 分阶段测试性能和部署变更的重要性

一家大型公司是子 LUN 自动分层技术的早期先驱,尽管在部署该技术之前做了大量规划,但仍学到了很多惨痛的教训。

其中一个教训是单个卷可以占用的一级闪存空间量。该公司在使用子 LUN 自动分层部署其第一个阵列时遇到了重大性能问题。他们花了几个周末的时间将应用程序小心地迁移到新的存储阵列。然后,在另一个计划的迁移周末之前的星期五,几个重要的应用程序的性能显著下降。性能下降足以推迟下个周末的计划迁移。调查开始了,长话短说,根本原因是最近添加的一个应用程序对性能要求很高,以至于它占用了大部分可用的一级闪存空间。

但由于占用了太多的一级闪存空间,它迫使其他重要的应用程序下降到较低的层级。

通过限制新迁移的应用程序可以占用的一级闪存存储量,解决了这个问题。

根据我的经验,这种微调很少需要,但这个例子凸显了自动分层技术的潜在危险:性能可能每天都在变化,这取决于每个应用程序在各个层中的移动方式。如果今天您的卷中有 10% 位于第 1 层闪存中,但明天您只有 1% 位于第 1 层闪存中,那么明天您的性能很可能会下降。

## 数组大小问题

不幸的是,自动分层解决方案并不适合所有情况。在阵列和存储系统规模较小的情况下,自动分层解决方案可能不是最佳选择。主要原因是每个存储层具有的主轴(驱动器)数量。例如,安装了 32 个驱动器的小型存储阵列可能非常适合配置为单个池的单个存储层,这意味着所有卷都可以访问所有 32 个主轴。如果您尝试将这样一个小型系统划分为三个存储层(例如,四个闪存驱动器、十二个 15K 驱动器和十六个 SATA 驱动器),您可能会发现每个层的主轴太少,无法使任何层的性能足够好,并且闪存存储的好处可能会被在层上上下移动数据所需的后端操作所抵消。此外,自动分层有时是一项许可功能,这会增加阵列的成本。通常,这种额外的成本对小型阵列来说并不划算。

作为一般规则,与在较小的阵列上使用自动分层相比,您最好使用单层中高性能驱动器(例如 10K SAS 驱动器),可能辅以闪存缓存,其中闪存驱动器用作第二级缓存而不是持久存储层。

## 远程复制

自动分层的一个经常被忽视的副作用是复制的源阵列和目标阵列之间可能存在不平衡。例如,在复制配置中,一个阵列被视为主阵列(已连接活动主机),另一个阵列被视为辅助阵列(已连接备用主机),主阵列将看到所有读写工作负载。由于这种读写工作负载,阵列的自动分层算法会在层上上下移动扩展区,以确保最佳布局。但是,目标阵列看不到相同的工作负载。目标阵列看到的都是复制的 I/O,只有写入 I/O 会从源复制到目标。读取 I/O 不会被复制。因此,卷在目标阵列上的层上的分布方式通常与在源阵列上的层上的分布方式大不相同。这通常会导致卷位于目标阵列上性能较低的层上。如果您必须将系统和应用程序迁移到目标阵列,那么这可能会成为一个问题,因为它们的性能可能低于在源阵列上的性能。这只是自动分层需要考虑的另一个问题。

## 概括

在本章中,我们介绍了现代存储领域中所有流行的容量优化技术。我们首先从容量和性能的角度讨论了精简配置以及厚卷和精简卷之间的区别。

然后,我们讨论了精简配置如何实现过度配置,以及这可能带来的潜在成本节约。我们通过讨论零空间回收技术以及这项技术如何成熟并帮助我们保持精简卷的精简来结束关于精简配置的讨论。然后,我们讨论了内联和后处理压缩重复数据删除,并仔细指出了每种技术的区别和用例,以及在并行使用这两种技术时需要注意的事项。我们在本章的最后讨论了自动分层,以及它如何通过将正确的数据放在正确的存储层上来使我们更有效地利用我们的容量。

## 章节概要

**精简配置** 精简配置 (TP) 技术允许您借出比实际拥有的更多的存储空间,从而提高存储资源的利用率。这称为过度配置,必须进行管理和监控,以免超出存储阵列中的物理容量。

**压缩** 压缩技术对数据进行重新编码,以便使用更少的位和字节来存储和传输数据,使您能够以更少的物理容量存储更多数据,并通过消耗更少带宽的网络更快地传输数据。压缩技术是所有良好备份解决方案的一部分,并且在全灰存储阵列中越来越受欢迎。

**重复数据删除** 重复数据删除技术可让您从存储系统中挤出更多可用容量。重复数据删除的工作原理是识别重复的数据块,并用指针替换重复的数据块。可变块重复数据删除被认为是一种领先的重复数据删除方法。重复数据删除技术是所有良好备份解决方案的一部分,并且在全灰存储阵列中越来越受欢迎。

**子 LUN 自动分层** 虽然子 LUN 自动分层技术不会增加可用容量,但可以让您更有效地利用存储资源。子 LUN 自动分层的工作原理是将经常访问的数据块放在高性能介质(如固态介质)上,而将不经常访问的数据块放在速度较慢、更便宜的存储介质上。使用子 LUN 自动分层技术,较小的扩展区大小可实现更高的效率。

Machine Translated by Google

# 章节 11



## 备份和恢复

---

本章涵盖的主题：

- ✓ 我们为何备份
- ✓ 备份方法
- ✓ 备份类型
- ✓ 备份目标
- ✓ 备份至云端
- ✓ 备份保留
- ✓ 归档



在本章中,您将了解服务器和应用程序的备份和恢复。虽然您将主要关注服务器和应用程序的备份和恢复,但许多相同的原则

适用于笔记本电脑、平板电脑等设备的备份。

本章的主要教训是备份就是为了恢复。这一点再怎么强调也不为过!把这个事实牢记在心,确保你永远不会忘记。如果你每天都备份数据,但无法从这些备份中恢复数据,那么你的备份就是无用的,你在浪费时间。

您还将了解备份方法,例如热备份和无 LAN 备份,以及以及它们对网络和应用程序性能和可恢复性的影响。您将了解备份的类型,例如完整备份、增量备份、差异备份和合成完整备份,以及每种备份对恢复、应用程序和网络性能的影响。本章还介绍了备份设备,例如磁带、磁带驱动器、虚拟磁带库,甚至其他不太常见的备份媒体类型,例如蓝光盘。

您将了解容量优化技术如何影响备份性能、成本和潜在的恢复性能。

本章还介绍了归档以及它与备份的不同之处和对备份的补充。此外,还有日益严格立法和监管审计要求的影响,这些要求管理着现代世界中大量的数据归档。您将了解非常适合归档的技术,例如内容可寻址存储和一次写入多次读取技术。

但是,本章的主要教训是,备份就是能够恢复丢失或损坏的数据 - 并且及时有效地恢复它,尽管不一定是超快的方式。

## 为何我们要备份

备份并不吸引人。它们要花钱。管理起来很麻烦。而且会影响应用程序性能。那么我们为什么要备份呢?首先,备份可以挽救您的数据,甚至挽救您的工作和公司 这一点毫不夸张。

基本上,没有数据就没有生意!

数据可能会被损坏或意外删除,这也不是什么秘密。当这种情况发生时,备份突然就成了世界上最重要的事情,每个人都坐在一起紧张地咬着手指,希望数据最近被备份并可以恢复。

以下是与备份的重要性相关的正确的陈述：

- 如果灾难发生后企业无法后退,就无法前进。
- 如果未备份,则表示不存在。
- 如果它不存在于至少两个地方,则不会备份。

这些陈述构成了备份的一些基本原则。

现在让我们讨论一下备份在灾难恢复和业务连续性规划中的作用,以及定义一些备份和恢复的基本术语。

## 备份和业务连续性

备份和恢复的主要作用之一是作为业务连续性计划 (BCP) 的一部分。在 BCP 中,备份与其他存储相关技术 (如复制和快照)并存,但我们应该注意,复制数据绝对不同于备份,快照本身并不构成备份。无论如何,备份应该成为几乎每个业务连续性计划的一部分。

## 恢复点目标

恢复点目标 (RPO) 是服务可以恢复到的时间点/

恢复。因此,如果您的 RPO 为 24 小时,您将能够将服务或应用程序恢复到不超过 24 小时前的状态。例如,每天凌晨 2 点备份数据库的备份计划可以将该数据库恢复到每天凌晨 2 点的状态。在任何给定时间点,您都将拥有不超过 24 小时的数据库有效备份 - 在您规定的 24 小时 RPO 内。

RPO 是 IT 和企业之间更广泛的服务水平协议 (SLA) 的一部分。

在 IT 领域,SLA 是一种合同形式,概述了 IT 将向客户提供的服务级别。如果您是内部 IT 部门,则客户通常是企业或业务部门。例如,SLA 可能规定所有数据将每天备份,并可在请求后的 6 小时内恢复。几乎所有 SLA 都包含恢复点目标。

说到备份,恢复点目标是一个主要考虑因素。如果企业要求某个应用程序的 RPO 为 5 分钟,那么该系统的每日备份将无法满足该 RPO,您需要使用快照等不同技术。

对于传统的基于磁带的备份,正常的最小 RPO 为一天 (24 小时),因为每天将传统磁带备份的频率超过一次通常是不现实的。这主要有两个原因:

- 许多备份方法都会影响被备份系统的性能。因此  
频繁进行传统的磁带备份可能会对备份系统的性能产生负面影响。
- 许多企业备份系统 (媒体服务器、磁带驱动器、备份网络)已经非常繁忙,每天仅进行一次备份。

在这种情况下,如果需要更精细的 RPO,可以使用快照等技术来增强传统备份。例如,这些快照可以每 15 分钟拍摄一次,但只保留一天。然后,每个完整的传统磁带备份可以每天拍摄一次,并保留 12 个月。这种配置将允许将应用程序恢复到过去 24 小时内的任何 15 分钟点。

有趣的是,在备份的上下文中,还有保留的概念。例如,如果您每天备份公司数据库服务器,将这些备份保留 12 个月,然后在超过 12 个月时立即将其过期(清除),那么您将无法将公司数据库系统恢复到 12 个月前的状态。在此示例中,公司数据库系统的保留期为 12 个月(365 天)。虽然这似乎不是问题,但法律和监管要求可能规定,给定系统的数据必须可供恢复过去 25 年的数据。

## 恢复时间目标

恢复时间目标(RTO)与恢复点目标有显著不同。

恢复点目标表示您可以在特定时间范围内将系统或应用程序恢复到其原始状态,而恢复时间目标则表示您需要多长时间才能完成此操作。例如,您可能需要 8 小时才能将系统恢复到 24 小时前的状态。

在定义和决定 RTO 时,请确保考虑以下因素:

- 从异地存储调出磁带所需的时间
- 召集员工、登录并开展恢复工作所需的时间

例如,如果您周六早上没有员工在工作,并且所有磁带都存放在安全的存储设施中,则您需要将这两个事实都考虑在 RTO 中,以处理用户在周六早上 6 点登录请求以恢复某些数据的可能性。这并不像说您可以每分钟恢复 x GB 那么简单,因此恢复 y GB 需要 z 分钟。还有其他重要因素需要考虑。

练习 11.1 向您展示如何创建一个简单的计划备份。

### 练习 11.1

#### 创建简单的计划备份

此分步活动将引导您完成创建研究文件夹每小时计划备份的过程,该备份将备份到本地安装为 Windows 服务器上 F: 驱动器的 SAN 磁盘。计划将在上午 7 点至下午 7 点之间每小时备份服务器的 c:\research 文件夹

1. 在 Windows 桌面上,单击开始 → 管理工具 → Windows Server 备份。

2. 在 Windows Server Backup 工具中,选择“操作”→“备份计划”。

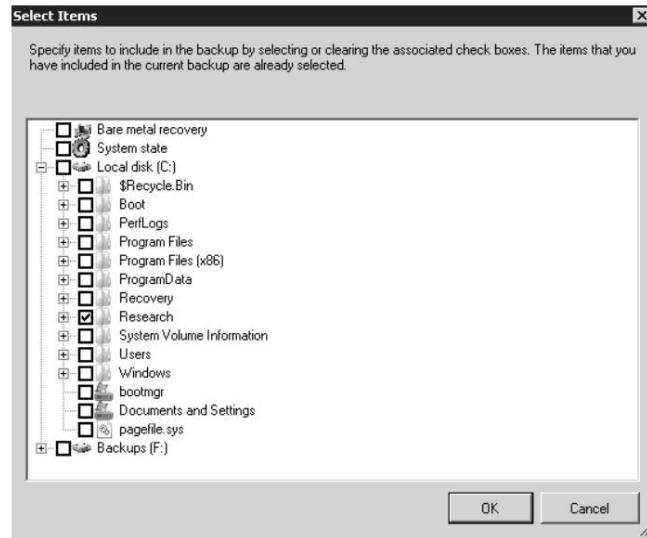
3. 在“开始使用”屏幕上,单击“下一步”。

4.在“选择备份配置”屏幕上,选择“自定义”。



5.在“选择备份项目”屏幕上,单击“添加文件”按钮。

6.在出现的“选择项目”屏幕上,选择要备份的文件夹,然后单击“确定”。在下图中,选择了“研究”文件夹。

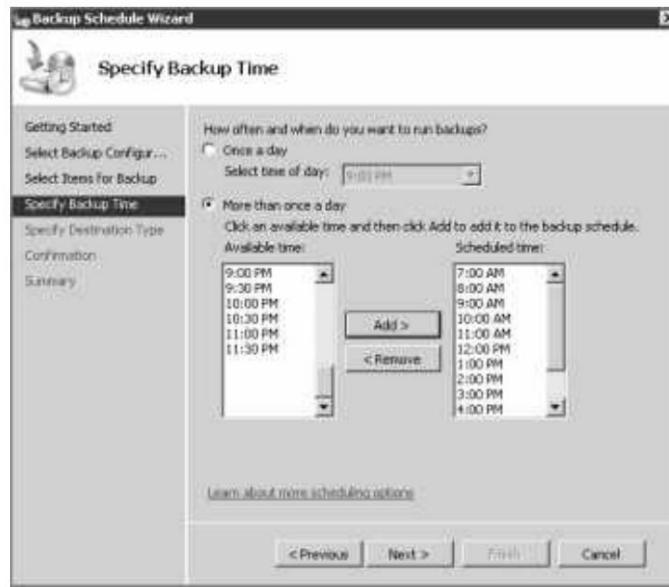


7.验证在“选择要备份的项目”中是否选择了正确的项目进行备份  
屏幕,然后单击“下一步”。

## 练习 11.1（继续）

8. 在“指定备份时间”屏幕上，选择“每天多次”单选按钮

然后选择要执行备份的时间。以下屏幕截图显示了上午 7 点到晚上 8 点之间选择的每小时备份



9. 在“选择目标类型”屏幕上，选择备份数据的目的地

存储，然后单击“下一步”。

10. 在“选择目标磁盘”屏幕上，选择要将备份存储在的磁盘

复制到并单击下一步。这可能会导致出现警告，告诉您所选磁盘的所有内容都将被格式化。这意味着磁盘上的所有现有数据都将被删除），因此请确保选择正确的磁盘作为备份目标。



11.在确认屏幕上,验证所有选项均正确无误,然后单击“完成”。

这样就完成了使用 Windows Server Backup 工具在 Windows 服务器上安排自定义数据集的定期每日备份所需的步骤。值得注意的是,此工具仅提供简单的计划,不允许您为一周中的不同日子(例如周末)指定不同的计划。

---

## 备份窗口

备份窗口是与备份和业务连续性相关的另一项内容。备份窗口是一段可以备份数据的时间段(有开始和停止时间)。例如,备份窗口可能包括某个企业的所有非核心业务时间,例如晚上 11 点到早上 9 点。如果是这样,则所有备份都必须在该时间窗口内开始和完成。但是,许多企业针对不同的应用程序有不同的备份窗口,具体取决于这些应用程序的核心业务时间。

备份窗口的主要原因是备份通常会对备份系统的性能产生负面影响,因此在核心工作日备份关键系统通常不是一个好主意。在确定备份窗口时,从业务相关人员那里获取意见非常重要。

# 一切都是为了恢复

没人关心备份;人们只关心恢复。备份通常被认为是 IT 中最不吸引人的领域。在考虑应用程序时,没人会考虑备份,也没人愿意自掏腰包购买磁带等备份基础设施。但是,每个人都希望备份数据,如果在需要恢复操作时没有备份,每个人都会非常生气!相信我,如果业务用户需要恢复数据,但没有该数据的备份,那么您将陷入困境。

备份的明确目的是能够在用户错误/删除、设备故障、逻辑损坏等情况发生后恢复数据。考虑到这一点,能够恢复您备份的数据绝对至关重要。如果您无法恢复数据,那么备份数据是毫无意义的,并且会让您在客户眼中显得无能。因此,定期演练恢复数据至关重要。您应该定期测试恢复已备份数据的能力。



## Real World Scenario

### 测试恢复操作的重要性

我曾工作过的一家公司一直愉快地备份其 Microsoft Exchange 环境,直到有一天,有人要求恢复一个关键的 Exchange 相关配置文件。当尝试从前一天晚上的备份中恢复文件时,备份中找不到该文件。这被认为是该特定备份作业中遗漏的文件,并尝试从前一天晚上的备份中恢复文件。文件再次丢失。重复此过程,尝试从周末和月末的备份作业中恢复文件;每次文件都丢失了。结果,无法恢复文件,不得不重建 Exchange 服务器。

事实证明,该组织的 Exchange 服务器备份是作为标准文件系统备份执行的,而不是使用与 Microsoft Exchange 协同工作的 Microsoft Exchange 备份代理来执行的,以确保应用程序一致的备份。

虽然这个问题立即被纠正,但是对于这个特定的恢复请求来说已经太晚了。

这次经历的教训是,该公司从未测试过从备份中恢复 Exchange。当公司需要依赖备份时,它们就毫无用处了!所以要练习恢复操作!



**除了能够恢复备份数据的重要性之外,有些应用程序(如数据库)需要经常备份才能截断日志文件。如果不定期备份这些系统,日志文件可能会填满并迫使数据库停止运行。**

**Except for being able to recover backed up data, some applications (such as databases) need to be backed up frequently in order to be able to truncate log files. If you don't back these systems up regularly, the log files can fill up and force the database to stop.**

## 备份架构 Architecture

最常见的备份架构基于久经考验的客户端/服务器架构。大多数备份架构由以下四个组件组成:client-server architecture. Most backup architectures are composed of the following four components:

- 备份服务器 servers
- 备份客户端 clients
- 媒体服务器 servers
- 备份目标 targets

备份服务器有时也称为主服务器或单元管理器,是备份环境的大脑。此服务器负责所有工作 server or cell manager, is the brains of the backup environment. This server holds all the job

定义、作业计划、备份数据库和目录以及与备份环境有关的所有其他元数据。

备份客户端是安装在备份服务器上的软件代理,负责将数据从备份的客户端计算机传输到存储介质上。特定于应用程序的备份代理 (例如 Microsoft Exchange 或 Microsoft SQL Server) 还可以协调备份的应用程序一致性,但稍后会详细介绍。



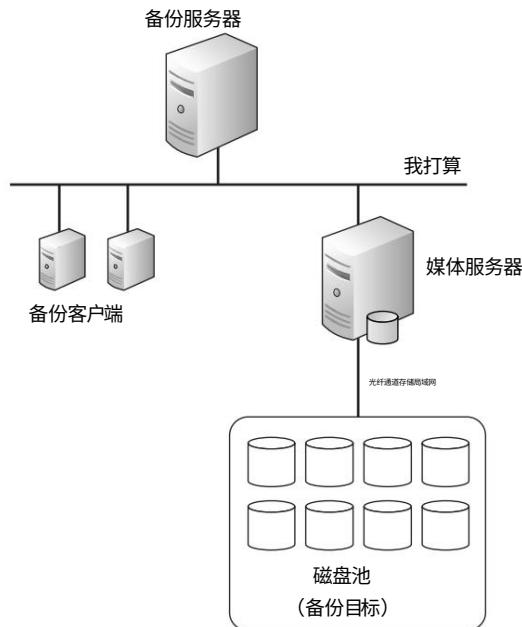
公司其他部门的人可能没有意识到备份客户端代理实际上是在服务器上,而不是在客户端上。在大多数情况下,这种区别可以忽略不计,但有时您可能需要向同事澄清这一点,以避免混淆。

媒体服务器连接到备份目标,并使其可供备份客户端使用,以便它们可以将数据发送到备份目标。例如,媒体服务器可能通过 FC 网络连接到存储池,并通过 IP LAN 使该存储可供备份客户端使用。

备份目标包括磁带驱动器和 DVD,甚至是大型磁盘存储池或大型磁带库。传统的备份方法主要使用磁带作为备份目标。更现代的方法倾向于使用基于磁盘的介质,包括通过基于 LAN 的协议 (如 SMB/CIFS 和 NFS) 共享的磁盘池,甚至是模仿磁带库的专用磁盘阵列 (称为虚拟磁带库)。

图 11.1 显示了具有单个备份服务器、两个备份客户端、单个介质服务器以及磁盘池。

图 11.1 备份环境示例



如图 11.1 所示,高级备份过程如下:

1.备份服务器监控备份计划。

2.当计划指示特定客户端需要备份时,备份

服务器指示备份客户端上的代理软件执行备份。作为此指令的一部分,备份服务器告诉代理要备份哪些数据以及将其备份到何处。

3.备份服务器通过IP将数据发送到媒体服务器。

4.媒体服务器通过 FC 将传入的备份数据传输到备份目标所连接的 SAN。

## 备份方法

备份是业界的一项老技术。事实上,备份的历史非常悠久,它与大型计算机一起成长,这两种技术至今仍是密不可分的。直到最近,备份方法和技术一直没有改变。事实上,在许多方面,备份看起来仍然与近 50 年前一样。然而,备份方法最终还是屈服于变化,用于备份服务器和应用程序的方法越来越多。本节介绍一些更流行的方法和方法。

### 热备份

在现代 IT 世界中,几乎所有备份都是热备份。热备份 (有时称为在线备份)是在被备份的应用程序和服务器保持在线并为用户提供服务时进行的。

热备份非常适合当今对 IT 资源的需求,因为服务器和应用程序需要始终保持在线状态。热备份还有助于减少管理开销,并允许备份环境扩展,因为它们不需要为了满足备份而协调和管理应用程序的管理停止和重新启动。

大多数现代应用程序 (例如 Oracle Database 和大多数 Microsoft 服务器产品 (Microsoft SQL Server、Microsoft Exchange Server、Microsoft SharePoint Server 等)) 都支持热备份。为了提供此热备份功能,大多数应用程序都会将自身置于热备份模式,并利用某种形式的时间点 (PIT) 技术来创建应用程序数据的冻结映像。然后使用此冻结映像进行备份。通常,快照技术 (包括基于阵列的快照技术) 被用作 PIT 冻结映像创建过程的一部分。

热备份领域中另一个极其重要的问题是执行备份完整性检查的能力。幸运的是,大多数支持热备份的优秀应用程序都具有强大的功能来执行校验和、基于应用程序的完整性检查以及备份映像验证。这样做有助于确保备份是良好的备份,并且可以用作恢复映像。

## 离线备份

离线备份与在线热备份相反,希望它们成为过去。离线备份要求应用程序和数据库在备份期间处于离线状态。显然,这在现代世界中并不理想,因为企业通常全天候运营,在执行常规备份时无法承受应用程序和服务的停机时间。

## 基于 LAN 的备份

基于 LAN 的备份已经存在多年,并且被认为便宜且方便,但通常性能较低,并且存在影响网络上其他流量的风险。

基于 LAN 的备份通过 LAN 发送备份数据来工作,此 LAN 可以是您的主生产网络或专用于备份流量的网络。专用网络可以是虚拟 LAN (VLAN),其中为备份流量创建专用的 L2 广播域,也可以是物理专用网络,在主机和网络交换机中具有自己的网卡。后一种选择显然比简单的 VLAN 花费更多,但它提供了最佳和最可预测的性能,并且最容易排除故障,并且对其他非备份相关网络流量产生负面影响的可能性最小。

在典型的基于 LAN 的备份配置中,备份客户端将通过 LAN 将数据从正在备份的主机发送到媒体服务器或直接发送到存储媒体。

重要的一点是备份数据总是通过 LAN 发送。

图 11.2 显示数据从客户端通过专用备份 LAN 发送到媒体服务器。然后,媒体服务器通过光纤通道存储区域网络 (FC SAN) 连接到备份目标。在图 11.2 中,每台服务器连接到两个 LAN:

- 管理局域网
- 备份局域网 (专用于备份流量)

## 无 LAN 备份 (基于 SAN)

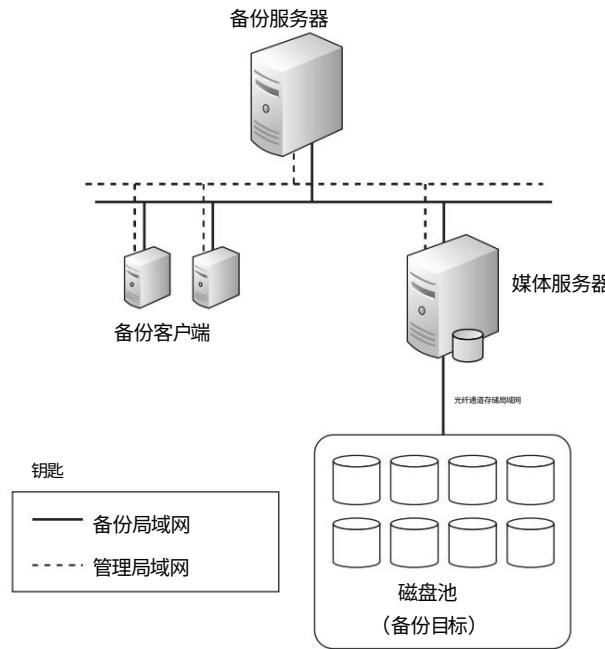
无 LAN 备份架构,顾名思义,就是无需通过 LAN 即可备份数据。通常,数据通过 SAN 从备份客户端传输到存储介质。



有时,无 LAN 备份被称为 SAN 备份,因为它们倾向于通过 SAN 而不是 LAN 备份数据。它们也经常被称为基于块的备份,因为它们通常不理解或不关心文件。它们只是备份它们被指示备份的所有块。当您尝试备份数百万个小文件时,这会非常快。然而,它有一个固有的缺点。基于块的备份通常不提供恢复单个文件的顺畅而简单的方法。

如果您想要恢复单个文件,则可能需要将整个备份恢复到临时区域,然后找到所需的文件并将其复制回原始位置。

图 11.2 基于 LAN 的备份



与基于 LAN 的备份相比,无 LAN 备份往往提供更高的性能和可靠性,但成本也更高。这是一个真正一分钱一分货的例子。然而,随着 10G 网络技术的出现,基于 LAN 的备份也可以提供高性能,但成本也很高。

在以下环境中,能够以更高的速度备份数据非常重要:  
备份窗口是一个问题。如果您很难在分配的备份窗口内完成所有备份,您可能需要考虑无 LAN 备份,甚至可能考虑通过 10G 以太网进行基于 LAN 的备份。

图 11.3 显示了无 LAN 备份配置,其中被备份的客户端与基于 SAN 的磁带库有直接连接。为了实现这一点,被备份的客户端通常必须具有可以直接与磁带库通信的备份代理软件或安装备份介质服务器软件。

在图 11.3 中,备份服务器通知备份客户端上的代理软件执行到磁带库备份目标的备份。然后,备份数据通过 SAN 而不是 LAN 直接发送到磁带库。这减轻了 LAN 上的负载,并可能使已投资 FC SAN 技术的组织能够部署强大的高性能备份环境,而无需部署专用的备份 LAN。

## 无服务器备份

无服务器备份是一种无 LAN 备份,但它们将这一概念提升到了一个新的高度。它们不仅无 LAN,这意味着它们不使用 LAN 来传输备份数据,而且无服务器,这意味着它们也不利用服务器资源。图 11.4 显示了这一概念。

图 11.3 无 LAN 备份

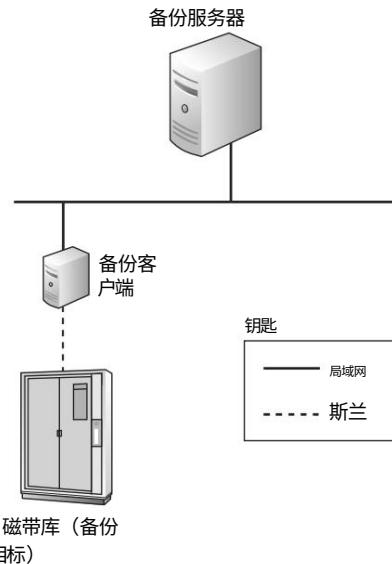
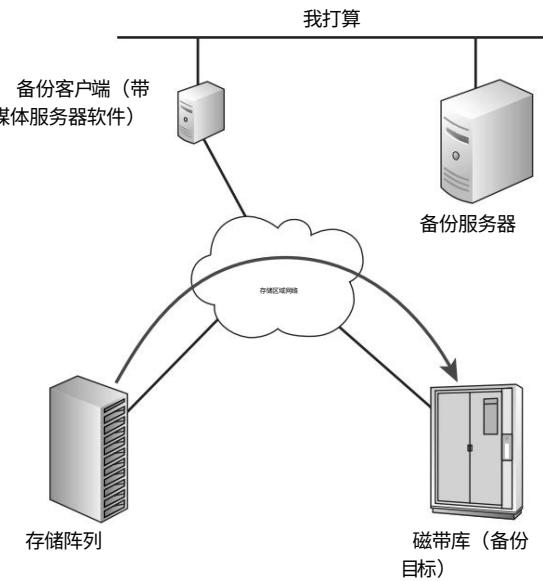


图 11.4 无服务器备份



在底层,无服务器备份使用基于 SCSI 的 EXTENDED COPY 命令,该命令允许将数据直接从源 LUN 复制到目标 LUN。这里重要的一个词是直接。在无服务器备份配置中,数据通过 SAN 从源 LUN 直接发送到目标 LUN,但不通过拥有数据的服务器。

由于无服务器备份与基于 SAN 的 LUN 一起工作，并且 SAN 上的磁带设备作为 LUN 呈现，因此无服务器备份配置中的目标 LUN 通常基于磁带。



**EXTENDED COPY** 命令不必将整个 LUN 复制到另一个 LUN。它将指定范围的块从源 LUN 复制到目标 LUN。这些范围是包含正在备份的数据的块。*from the source LUN to*

图 11.4 显示了典型的无服务器备份配置，其中备份数据副本所采用的路径以粗线突出显示，从存储阵列通过 SAN 到磁带库。该过程大致如下：

1. 备份服务器告诉备份客户端机器上的备份代理软件以指定的磁带库为目标启动无服务器备份。
2. 使用 SCSI EXTENDED COPY 命令将数据通过 SAN 直接从存储阵列复制到磁带库。任何备份数据都不会经过备份客户端。
3. 有关作业的元数据通过 IP LAN 传递到备份服务器。此元数据包括作业详细信息，例如开始时间、完成时间、完成代码等。

虽然备份的数据不通过 LAN 发送，但有关备份仍然通过 LAN 发送到备份服务器和备份客户端/代理。这样，作业完成代码、开始时间、结束时间等元数据就可以存储在备份服务器上。

无服务器备份的优势与无 LAN 备份的优势相同（高性能和高可靠性，但成本可能高于基于 LAN 的备份），另外还具有减少对备份服务器的负面影响的优势。毕竟，它们被称为无服务器备份，因此可以预料到它们不会触及或影响服务器。

#### 国家信息管理计划

网络数据管理协议 (NDMP) 是一种专为基于标准的高效 NAS 备份而设计的协议。NDMP 与无服务器备份类似，备份数据可以直接从 NAS 设备发送到备份设备，而无需通过备份媒体服务器。如果没有 NDMP，则必须将基于 NAS 的文件共享安装在备份媒体服务器上才能进行备份。此配置会将数据通过网络 (LAN) 从 NAS 拖到备份媒体服务器，然后可能再通过网络拖回到备份设备。使用 NDMP，无需在媒体服务器上安装基于 NAS 的网络共享。相反，数据可以直接从 NAS 设备发送到备份设备，只需接触网络一次。这减少了网络负载。

非 NAS 相关的网络文件服务器（例如 Windows 和 Linux 服务器）不一定需要 NDMP，因为它们可以安装备份代理软件，以便直接备份到基于网络的备份设备。但是，NAS