

章节 1



存储入门

本章涵盖的主题：

- ✓ 信息技术的重要性
- ✓ 存储的作用
- ✓ 存储类型
- ✓ 存储如何融入更广泛的 IT 基础设施
- ✓ 闪存的影响
- ✓ 云存储
- ✓ CompTIA Storage+ 由 SNIA 考试提供支持



本章为导论,概述了技术在世界上日益重要的作用,以及这如何创造了对强大、高性能、基于技术的系统的需求,而这些系统需要优秀、知识渊博的人来设计和管理。它解释了存储在信息技术基础设施领域中占据的重要地位,以及它如何与计算和网络等其他基础设施组件交互。您将了解主要的存储形式,包括机械磁盘驱动器和新一波固态存储技术,如闪存。您还将了解云的影响。

信息的重要性 技术

信息技术 (IT) 从未像现在这样重要。它比昨天更重要,并且有迹象表明,明天它将更加重要。

从社交媒体和个人生活,一直到主要政府和企业,信息和技术是一切的中心。

在我们的个人生活中,我们越来越依赖技术设备 智能手机、智能手表、平板电脑、笔记本电脑等等。想想上次您的家庭互联网连接中断时给您带来多大的不便。突然间,您与世界隔绝,与世隔绝,能做的事情受到严重限制!即使在旅途中,我们也经常依靠 Google 地图等服务在大城市中导航或开车。如果我们开会迟到了,需要知道下一班回家的火车时间,我们该向谁求助?当然是我们的技术设备。事实上,我们在个人生活中越来越依赖技术。

企业也是如此。企业越来越依赖信息和技术。尝试切断大多数企业的互联网连接,看看会发生什么!

这不仅仅是互联网。以任何形式切断一家企业的信息和数据,看看这家企业多久会破产!

不久前,高科技对于大多数企业来说都只是锦上添花的功能。高速互联网连接是一种奢侈品,而即时访问信息则是一件好事。在当今世界,如果没有即时访问信息和互联网,企业根本无法生存。在当今世界,技术就是业务。

无技术,无生意!

由于技术对企业至关重要且具有核心性,因此企业拥有可靠、高性能的技术系统至关重要。低延迟贸易公司需要超快的 IT 系统,使他们能够尽快执行交易;如果交易速度太慢,他们就会亏损。服务公司需要可靠、高性能的 IT 系统,使他们能够为客户提供服务;如果这些服务不可用或速度太慢,客户就会转向其他地方。在线销售公司需要高性能、可靠的 IT 系统来处理客户订单;如果在线采购系统出现故障或速度太慢,客户将无法购买商品。医院需要高速、可靠地访问患者信息,以便做出准确的临床决策;如果医生不能及时访问最新的、准确的患者数据,患者可能会遭遇可怕的事情。运输公司需要高速、可靠的 IT 系统访问,以便提供安全可靠的服务,如空中交通管制和火车轨道信号。媒体公司需要高速、可靠的 IT 系统来创建和向订阅者提供高质量的内容。这个清单可以一直列下去。

IT 很重要,这一事实不会很快改变。为了让公司为了拥有可靠、高速的 IT 系统,他们需要优秀的人才来设计和管理它们,而这正是您发挥作用的地方!

存储在 IT 中的作用

在 IT 领域,存储是三大 IT 基础设施系统之一:

- 计算
- 网络
- 存储

从高层次来看,我们通常认为它们分为三层:计算层、网络层和存储层。在计算层中,Web 服务器、数据库和应用程序等应用程序可以运行。网络层提供计算节点之间的连接。例如,一种常见的设计方法是让计算机运行 Web 服务器服务,该服务通过网络层与另一台计算机上运行的数据库服务进行通信。最后,存储层是所有数据所在的位置。一个简单的类比如下:计算节点就像汽车和卡车等车辆;网络层就像道路基础设施;存储层就像存放所有商品和库存的办公室和仓库。这个比喻可能有点粗俗,但它传达了基本思想。

持久性和非持久性存储

存储有多种类型,但从高层次上讲,主要有两种类型:

- 持久性 (有时称为非易失性)
- 非持久性 (有时称为易失性)

顾名思义,持久性存储是长期存储数据的标准选择。我们称其为持久性或非易失性,因为断电后内容不会丢失。旋转磁盘驱动器可能仍然是最流行的持久性存储形式,尽管闪存越来越受欢迎。几乎每台现代个人技术设备(智能手机、智能手表、平板电脑、笔记本电脑等)都配有内置闪存。由于它是非易失性的,当该设备的电池耗尽时(这种情况经常发生),您的数据是安全的!

最常见的非持久性或易失性存储示例可能是随机存取存储器(RAM)。几乎所有形式和变体的RAM在断电时都会丢失其内容。那么为什么要使用RAM?因为它速度很快通常比许多形式的持久性存储快得多。在某些类型的操作中,RAM可以比磁盘快一百万倍,比闪存快一千倍。

一般来说,当我们提到存储时,我们指的是持久性、非易失性存储,这也是本书主要关注的内容。偶尔你可能会听到有人将RAM称为存储,但这种情况很少见。大多数情况下,存储是指持久性存储,而内存是指非持久性技术,如RAM。一个例外是闪存,它是一种持久性内存,这意味着我们认为它是存储(断电时它不会丢失其内容),但其行为很像内存,我们称之为固态。

尽管本书的大部分内容都是关于固态存储(SSS)的,但我们几乎总是使用“固态”一词来表示设备是基于半导体的,没有移动部件。另一方面,磁盘驱动器不是固态设备,因为它有许多复杂的移动部件,因此被认为是机械的。

性能和可用性

机械和固态存储设备都具有两个重要特点:

- 性能
- 可用性

在性能方面,与IT领域的其他一切一样,存储速度非常重要。但在存储领域,性能可能是一个有点复杂的主题。例如,磁盘驱动器是计算机中剩下的最后一个机械组件,因此,它总是难以跟上硅基邻居(CPU、内存等)的步伐。由于每个旋转磁盘驱动器都涉及所有机械组件,因此磁盘驱动器在随机工作负载下的表现并不好。然而,它在连续工作负载下表现优异。另一方面,闪存在随机读取工作负载方面就像闪电一样,最新的基于闪存的驱动器在连续工作负载下也越来越快(有些额定速度超过400 MBps,而最新的15K旋转驱动器额定速度接近250 MBps)。因此,存储性能不是一个简单的问题,需要对底层组件有很好的了解。当然,这本书解决了所有这些问题,所以如果这些主题现在看起来很复杂和令人生畏,那么在你读完本书的相关章节后,它们应该会像水晶一样清晰。

关于可用性,由于存储在 IT 世界中具有核心和关键性,因此至关重要的是,存储解决方案必须能够处理组件故障(例如驱动器故障)。生活中,东西出故障是常有的事,尽管旋转磁盘驱动器是一项令人惊叹的现代工程壮举,但旋转驱动器仍然会出故障。而且,当它们出故障时,它们往往会以独特的方式出故障,通常会带走所有数据。因此,了解能够让您构建高可用性、弹性存储解决方案的概念至关重要。同样,本书将贯穿整个主题。

50多岁

磁盘驱动器已经存在很长时间了。它很可能比你还老。虽然多年来它取得了很大进步,但它也没有太大的变化。

听起来上一句话是不是自相矛盾?让我解释一下。1956 年,IBM 制造并发售了 IBM RAMAC 350 磁盘存储单元。这实际上是有史以来第一个磁盘驱动器,也是所有机械磁盘驱动器之父。按照今天的标准,它是一个庞然大物。它的重量超过一吨,高 5 英尺 6 英寸,配备不少于 24 个 50 英寸的盘片(磁盘)。尽管它体积和重量如此之大,但它存储的数据不到 4 MB(在当时相当多)。从容量和物理尺寸方面来看,从那时起,事情已经突飞猛进。但从架构上讲,盘片、磁头、执行器和主轴电机的物理设计基本保持不变。

在过去 50 多年中,机械磁盘驱动器已成为高性能、高容量持久存储需求的首选介质。直到 2009/2010 年左右,它在这一类别中相对不受挑战。然而,存储领域正在发生重大变化,主要是由于以下两种技术:

- 固态存储/闪存

- 云

固态存储正在挑战旋转磁盘驱动器的霸主地位。它不仅正在强行进入手机和平板电脑等个人电子产品领域(在这些领域它已经取代了机械磁盘驱动器),而且还正在进入企业数据中心。笔记本电脑和个人电脑配备固态驱动器(SSD)而不是磁盘驱动器的情况并不少见,数据中心的许多存储阵列要么是全闪存(没有磁盘驱动器),要么是旋转磁盘和 SSD 的组合。

云既有趣又极具颠覆性,尽管它对企业和个人 IT 的颠覆性大于磁盘驱动器本身。然而,尽管用户或组织曾经自然而然地将数据存储在自己的数据中心和设备中的旋转磁盘上,但这些数据越来越多地存储在云中,其中磁盘技术等技术选择是其他人的责任。

尽管古老的磁盘驱动器已有半个多世纪的历史,并且依然强劲发展,但它在数据中心和 IT 设备中的主导地位正受到比以往更大的威胁。
这是令人兴奋的时刻!

云计算的影响

云计算已经到来,不容忽视。事实上,云计算令人兴奋,它正在改变我们设计、部署和与 IT 交互的方式。这些都不是空洞的承诺。云计算已经到来,并且正在兑现许多承诺。

在传统的 IT 基础设施中,公司的 IT 资产(服务器、存储和网络设备)由公司的 IT 部门拥有和管理。这些设备通常位于计算机房或数据中心,并通过公司租用的网络链路进行远程访问。一个典型的例子是一家销售公司,其办公室位于纽约市,连接到公司在其他地方(如新泽西、波士顿或芝加哥)的数据中心托管的服务器和应用程序。在云模型中,情况有所不同。该公司可能仍在纽约设有办公室,但不再拥有和管理其数据中心的所有 IT 设备,而是现在由云提供商拥有和管理,并位于云提供商的数据中心。该云提供商可能是公共云公司(如 Amazon 或 Rackspace)、私有云提供商或软件即服务提供商(如 Salesforce.com 或 Google Apps)。

在云模式中,云提供商负责购买 IT 设备、日常管理、升级、处理故障、管理数据中心以及执行所有传统 IT 功能。这样,销售公司就可以专注于其核心业务,而不必担心管理 IT 设备。销售公司不必预先购买设备来应对预期的增长和需求,而是可以按月支付基于云的 IT 资源的使用费用。例如,假设销售公司已将其 IT 基础设施迁移到云中,并且在一年中的大部分时间里交易稳定,但在黑色星期五、网络星期一和圣诞节假期期间销售量大幅增加。如果公司的正常交易水平需要 100 台虚拟服务器和 500 TB 的存储空间,但在假期期间,这一需求达到峰值,需要 250 台虚拟服务器和 750 TB 的存储空间,那么云是一个不错的选择。这是因为,在一年中的大部分时间里,该公司使用并支付了 100 台虚拟服务器和 500 TB 的存储空间,但在节假日期间,该公司将虚拟服务器数量增加到 250 台,存储空间增加到 750 TB。节假日结束后,该公司将关闭额外的服务器和存储空间,并停止支付费用,直到明年同一时间。

这只是云计算如何改变我们消费和支付方式的一个例子
IT 服务。本书后面将更详细地介绍此主题。

CompTIA Storage+ 由 SNIA (SGO-001) 考试提供支持

本书涵盖了 CompTIA Storage+ Powered by SNIA (SGO-001) 考试所需的主题。但是,本书绝对不仅仅是一本考试准备指南。它涵盖的内容比考试内容多得多,并且专注于为您在现实世界中工作做好准备。话虽如此,在撰写本书时,我亲自参加并通过了考试,我相信本书对考试涵盖的内容进行了公正的评价。但本书还涵盖了更多内容。

章节概要

信息技术的重要性 IT 在我们的个人生活和业务中从未如此重要。作为个人,如果没有技术设备,我们越来越感到孤立和与外界隔绝。同样,企业也越来越依赖技术。没有技术就没有企业,这个概念从未如此真实。

持久性存储 持久性存储或持久性存储介质是指断电后不会丢失其中存储的数据的存储介质。磁盘驱动器就是一个典型的例子。

您可以关闭磁盘驱动器的电源,一年后再打开驱动器的电源,数据仍会存在。本书的大部分内容侧重于持久存储的形式以及围绕持久存储设备构建的解决方案。

章节

2



存储设备

本章涵盖的主题：

- ✓ 40,000 英尺高空存储
- ✓ 机械磁盘驱动器
- ✓ 磁盘驱动器性能
- ✓ 磁盘驱动器可靠性
- ✓ 固态介质
- ✓ 固态性能
- ✓ 固态可靠性
- ✓ 胶带
- ✓ 混合驱动



本章涵盖了您在 IT 职业生涯以及 CompTIA Storage+ 考试中需要了解的有关磁盘驱动器和固态驱动器的所有内容。

因为对磁盘驱动器和固态技术的透彻理解对于存储领域的成功至关重要,所以本章深入介绍了许多基础理论。

它涵盖了旋转磁盘驱动器的所有主要机械部件,并深入探讨了闪存和固态技术的理论和内部工作原理。它向您介绍了旋转介质与固态介质截然不同的性能特征。

每种技术都有其适用之处,并不一定像说闪存和固态技术将完全取代旋转磁盘驱动器那么简单。

在了解了这些关键数据中心技术的工作原理之后,你将了解它们对整体性能和可用性的影响,这对您的 IT 环境至关重要。您还将了解容量并探索与成本相关的概念,例如每 TB 的美元数 (\$/TB) 和每 I/O 操作的美元数 (\$/I/O 操作),以及如何使用这些成本指标来制定存储支出的商业案例。

40,000 英尺高度的存储

本章的目的是介绍您需要了解的有关全球数据中心主要使用的存储设备的所有信息。这些设备如下：

- 磁盘存储
- 固态存储
- 磁带存储

这三种都是存储形式或介质。

磁盘存储是指机电硬盘驱动器。但由于这个名字有点拗口,大多数人将其称为磁盘驱动器、硬盘驱动器或硬盘驱动器 (HDD)。

我将互换术语以让你保持警惕。

固态介质是指一大批争夺数据中心市场份额的新技术。其中许多目前都是基于闪存的,但也存在其他形式的固态介质,而且还有更多形式即将出现。这是一个特别令人兴奋的存储领域。

磁带存储指的是磁带。虽然听起来很疯狂,但磁带在全球的 IT 基础设施中仍然随处可见。与磁盘驱动器一样,磁带也没有放弃其在数据中心的份额的迹象。我希望磁带能比我活得更久,我计划过上长久而快乐的生活。



不要误以为跳过本章你仍然可以成为存储专家。也不要误认为磁盘驱动器和磁带等技术是旧技术，你需要彻底了解它们。花点时间深入了解本章中包含的信息，这将成为你未来作为存储专业人士职业生涯的金矿。

我认为值得指出的是，在企业技术中，当人们提到存储时，他们几乎总是指磁盘或固态存储。偶尔你们可能会使用该术语来指光学介质或磁带，但没有人会用它来指内存。只有非技术新手才会认为他们的PC有1TB内存（它实际上有1TB的存储空间）。

磁盘、固态和磁带存储都是持久性存储或非易失性存储。这两个术语的意思是一样的。如果你在关掉电源开关，当你重新打开电源时，你的数据仍然会在那里。当然，你永远不应该直接打开电脑设备的开关！你需要小心谨慎地以合理的方式关闭电脑，因为任何在断电时只部分写入的数据在恢复供电时也只会部分写入。这个术语称为崩溃一致性。这种行为不同于内存（如DRAM）的行为。如果你关闭DRAM的电源，你就可以和它曾经存储的数据说再见了！*ten when the power is pulled will be only partially written when the power is restored—a term known as crash consistent.* This behavior differs from that of memory, such as DRAM; if you pull the power to DRAM, you can kiss goodbye the data it once stored!



誤解“崩溃一致性”这个词可能极其危险。*Crash consistent* can be extremely dangerous. 崩溃一致性的数据根本不能保证是一致的。事实上，它很有可能是损坏的。在应用程序备份方面理解这一点尤其重要。崩溃一致性甚至还远远不够好。*Understanding this is especially important when it comes to application backups.* Crash consistent is not even close to being good enough.

现在已经了解了这些基础知识，让我们近距离了解每一种重要的存储技术。

机械磁盘驱动器 Mechanical Disk Drive

嘿，我们已经进入21世纪了，对吧？那么为什么我们仍然在喋喋不休地谈论机械磁盘驱动器？它是an-
一块属于20世纪的旋转锈蚀物，对吧？*rust that belongs back in the 20th century, right?*

错误的！

是的，磁盘驱动器已经过时了。是的，它已经存在了50多年。是的，它比以往任何时候都更加被推向极限。但不管你喜欢还是不喜欢，机械磁盘驱动器并没有表现出近期自愿退役的迹象。*Mechanical disk drive is showing no signs of taking voluntary retirement any time soon.*



虽然拼写检查器可能不会将disk拼写为disk，但这样做会暴露给您，
在存储新手存储行业中没有经验的人会将disk拼写为dsk。例外情况包括捷克语
DVD或蓝光光盘等光学媒体。



The IBM 350 RAMAC disk drive from 1956 is about 1050 cm³ and has 50 platters, each 20 inches in diameter. This means it's absolutely not suitable for your smartphone or laptop. It's also not suitable for your desktop computer. You don't want to move it, it's very heavy and slow, and it's only 4 MB of capacity! Fifty 24-inch platters, a forklift truck to move it, and only 4 MB of capacity! Fast-forward to today, where you can have over 4 TB of capacity in your pocket. 4 TB is over a million times bigger than 4 MB. That's what I call progress!

As the disk drive and its quirky characteristics are so important to understanding storage, I highly recommend you know the disk drive inside and out. If you don't, it will come back to bite you later. You have been warned!

硬盘驱动器的构造 a Disk Drive

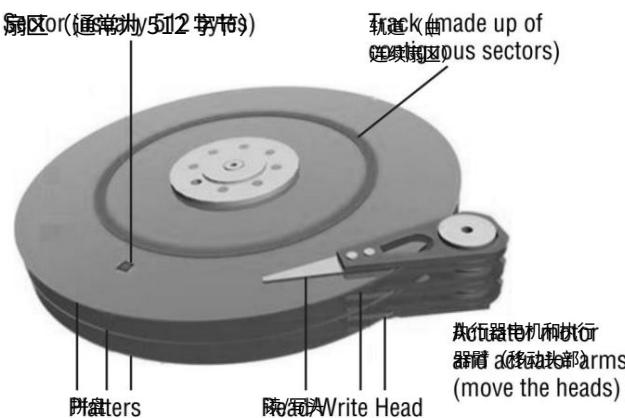
In almost all servers (racks) and notebooks and desktop PCs, the disk drive is the last remaining mechanical component. Other components are based on silicon. Understanding this is very important, because it makes the disk drive hundreds, sometimes even thousands, of times slower than everything else in a server or PC.

硬盘驱动器由四个主要机械部件组成:

- **拼盘**platters
- **读/写头**Read/write heads
- **执行器**actuator assembly
- **主轴电机**Spindle motor

Figure 2-1 shows the four components. We will also discuss other components in the following sections.

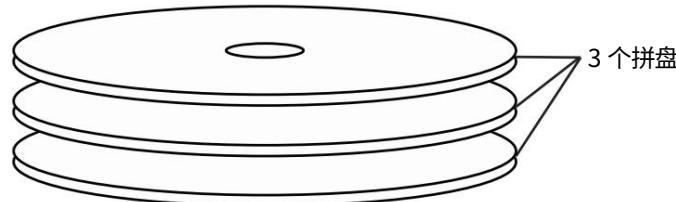
FIGURE 2-1 基本结构 anatomy of a disk drive



拼盘

盘片是存储数据的地方。盘片看起来就像任何其他刚性圆形磁盘，例如蓝光光盘或 DVD。大多数磁盘驱动器都有多个堆叠在一起的盘片，如图 2.2 所示。

图 2.2 三个堆叠的盘片



每个盘片都是刚性的（不会弯曲）、薄而圆、非常平滑，并且像风一样旋转。数据通过读/写磁头读取和写入盘片表面，读/写磁头由执行器组件控制。稍后会详细介绍。

盘片通常由玻璃或铝基板构成，每个盘片表面是磁性涂层。每个盘片的顶部和底部表面都用于存储数据。容量通常是磁盘驱动器的关键，因此绝对不会浪费任何空间！

由于所有盘片都连接到一个公共轴（主轴），因此磁盘驱动器中的所有盘片同步旋转，即它们同时开始和停止，并以相同的速度旋转。



光滑表面的重要性 The smoothness of the platter surface is unbelievably important.

任何缺陷都可能导致磁头损坏和数据丢失。 Any surface defect can result in a head crash and data loss.

为了避免磁头碰撞，磁头在盘片表面上方飞行时应保持安全距离，同时保持足够近的距离以便能够读写。光滑的表面在这里非常重要，因为它允许磁头更靠近盘片表面并获得更清晰的信号，因为干扰（称为噪音）较少。

表面不太光滑会产生更多噪音并增加碰撞风险。

读/写头

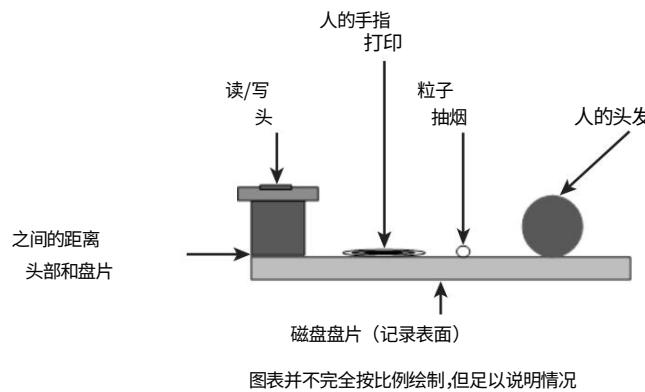
读/写磁头有时也被称为 R/W 磁头，或者简称为磁头。这些是位于盘片表面上方并读取和写入数据的部件。它们连接到执行器组件，并由磁盘驱动器控制器上的固件控制。操作系统、卷管理器和文件系统等高级系统对 R/W 磁头一无所知。

飞行高度

读/写磁头在每个盘片上方以极小的距离飞行，称为飞行高度。飞行高度以纳米为单位。如果纳米并不意味着

不管怎样,大多数磁盘驱动器的轴高都小于一粒灰尘的大小或一个指纹的深度。图 2.3 有助于理解这一点,并很好地说明了现代磁盘驱动器的设计和制造中涉及的精密工程。磁盘驱动器简直就是现代工程的奇迹!

图 2.3 头部飞行高度



不仅头部的翼高极其微小和精确;
磁头在正确扇区上的定位也非常精确。如今,数据在盘片表面上非常密集,以至于哪怕位置稍微偏离一点点,R/W 磁头就会读取或写入磁盘上的错误位置。这被称为损坏,您不想遇到这种情况!

磁头碰撞

必须注意的是,读/写磁头绝不会接触盘片表面。如果接触,则称为磁头碰撞,几乎肯定会导致驱动器上的数据无法读取,驱动器需要更换。

读取和写入盘片

当磁头在盘片表面上方移动时,它们能够感知并改变经过其下方的扇区和磁道中的位的磁性方向。

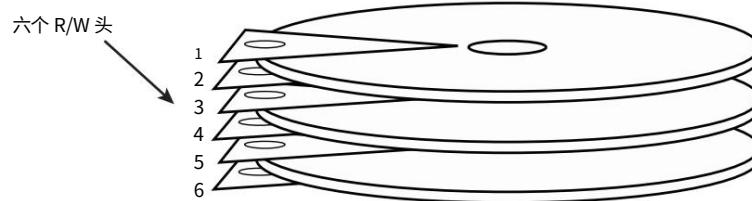
写入数据时,读/写磁头位于每个盘片表面的上方或下方,并在经过时磁化表面。磁充电以表示二进制 1 和 0 的方式进行。从盘片表面读取数据时,读/写磁头检测其下方区域的磁电荷以确定其表示二进制 1 还是 0。实际上,对磁盘扇区的读写次数没有限制。

磁头和内部驱动器寻址

每个盘片表面都有自己的 R/W 磁头。图 2.4 显示了具有三个盘片和六个 R/W 磁头的驱动器。每个盘片有两个记录表面(顶部和底部),每个表面

有自己的 R/W 磁头。三个盘片,每个盘片有两个记录面,等于六个记录面。要从这些记录面读取和写入,我们需要六个 R/W 磁头。

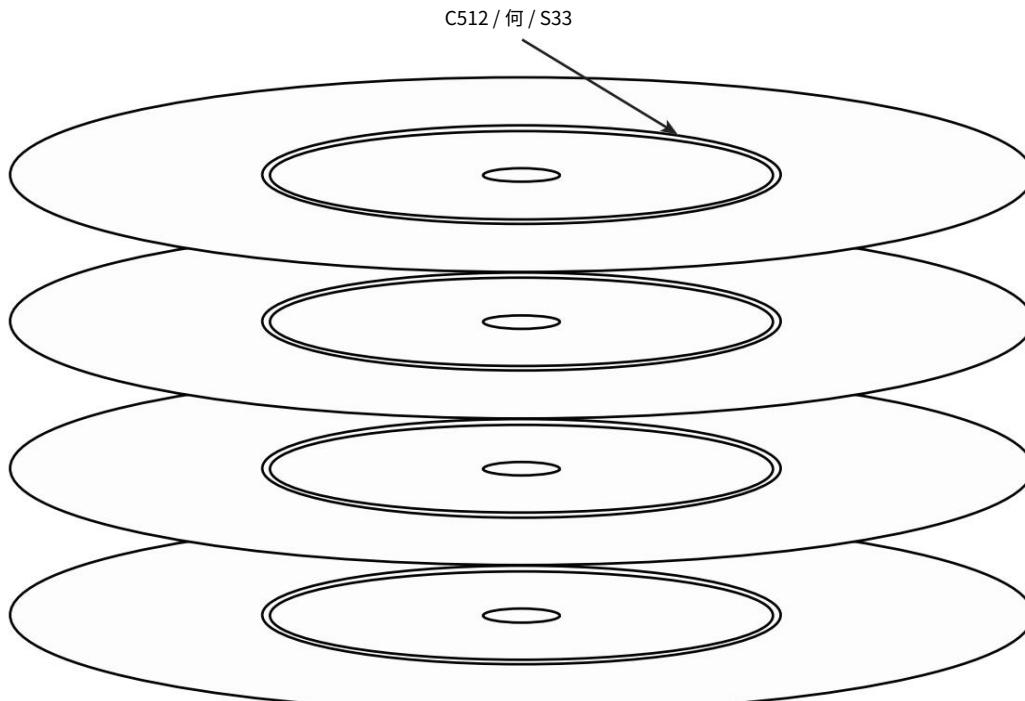
图 2.4 磁头和盘片表面



磁头和记录表面的概念在磁盘驱动器使用的柱面-磁头-扇区 (CHS) 寻址方案中非常重要,CompTIA Storage+ 考试目标中引用了该方案。要寻址磁盘驱动器中的任何扇区,您可以指定柱面号 (它为我们提供磁道)、磁头号 (它告诉我们该磁道位于哪个记录表面上) 和扇区号 (它告诉我们我们刚刚识别的磁道上的哪个扇区)。

图 2.5 显示的是柱面 512、磁头 0、扇区 33,或者换句话说,是盘片表面 0 上磁道 512 上的扇区 33。

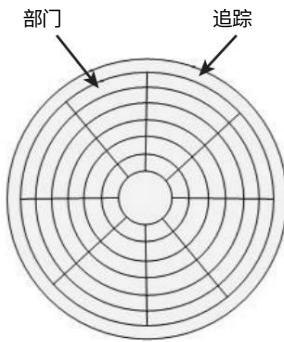
图 2.5 CHS 位置



轨道和扇区

每个盘片的表面都微观地划分为磁道和扇区。图 2.6 显示了划分为磁道和扇区的盘片示例。

图 2.6 轨道和扇区的鸟瞰图



每个盘片表面有许多磁道,每个磁道都是围绕盘片的同心环。每个磁道又被划分为扇区。

扇区是磁盘驱动器的最小可寻址单元,通常为 512 或 520 字节大小(在我们的示例中,我们将主要使用 512 字节)。因此,如果您只想将 256 字节写入磁盘,则需要整个 512 字节扇区,并且会浪费 256 字节。但不要为此感到紧张;这些值太小了,微不足道!如果要写入 1,024 个字节,则需要四个扇区。很简单!



While on the topic of sectors, it's worth mentioning the Serial Advanced Technology (SAT) interface. SAT drives have the ability to define different sector sizes and even different numbers of sectors per track. This ability is crucial for implementing data integrity technologies like End-to-End Data Protection (EDP). In EDP (also known as Data Integrity Fields or Logical Block Protection), additional data protection metadata is appended to the data, and the data is checked for consistency during transmission from the host file system to the storage medium.

EDP 允许驱动器在将数据提交到驱动器之前或将损坏的数据返回到主机/应用程序之前检测错误。EDP 添加的 8 个字节数据的一部分是保护字段,其中包括循环冗余校验 (CRC),允许从应用程序到驱动器的每个设备检查数据的完整性并确保数据没有损坏。

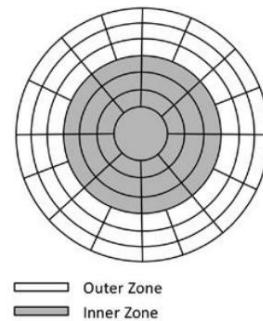
如图 2.6 所示,每个扇区的物理大小朝向盘片中心越来越小。由于每个扇区存储的数据量相同(例如 512 字节),因此外层磁道包含浪费的空间。在磁盘驱动器设计中,浪费空间/容量是不可饶恕的,因此必须发明 ax。为了更好地利用可用空间,大多数现代磁盘都实施了一种称为分区数据记录(ZDR)的系统:与内层磁道相比,更多的扇区被挤压到更靠近边缘的磁道上,从而使驱动器可以存储更多数据。有关图形描述,请参见图 2.7。

在 ZDR 类型的磁盘中(世界上几乎所有磁盘都实现了某种形式的 ZDR),每个盘片被划分为多个区域。同一区域中的轨道每条轨道的扇区数相同。但是,不同区域中的轨道每条轨道的扇区数不同。

图 2.7 显示了盘片表面分为两个区域:外区每磁道有 16 个扇区,内区每磁道有 8 个扇区。在这个简化的示例中,我们的盘片外区有 3 个磁道,内区有 3 个磁道,总共有 72 个扇区。

如果磁盘只有一个区域,每个磁道有 8 个扇区,那么同一个磁盘就只有 48 个扇区,也就是潜在容量的三分之二。显然,这是一个过于简单的例子;在现实世界中,每个磁道都有更多扇区。即便如此,这个例子也展示了如何实施 ZDR 技术可以提高空间效率。

图 2.7 分区数据记录



现代磁盘驱动器每平方英寸可以容纳超过 1TB 的数据! bit of data per square inch!
A兆位约为 128 GB! 这种称为密度。GB! This is known as areal density.

以下是与磁道和扇区相关的几个性能要点:

- 因为磁盘的外层磁道包含更多扇区,所以磁盘每次旋转时,外层磁道可以存储和检索更多数据。让我们看一个简单的例子。
在图 2.7 所示的示例盘片中,外圈磁道每转可读取或写入 16 个扇区,而内圈磁道每转只能读取或写入 8 个扇区。外圈磁道的执行速度可高达内圈磁道的两倍。因此,大多数磁盘驱动器会开始将数据写入每个盘片的外圈磁道。虽然这是

这是一个重要的理论,但你几乎无法影响这种行为或在现实世界中利用你对它的了解。我的建议是理解这个理论,但不要为此失眠!

- 如果数据可以读取或写入相同或相邻轨道上的连续扇区,则性能将优于数据随机分散在盘片上的情况。例如,读取外轨道中的 0-15 扇区(称为顺序读取)可以在盘片旋转一次后完成。相反,如果读取的 16 个扇区分散在整个盘片表面,则读取操作(在这种情况下称为随机读取)几乎肯定需要不止一次旋转。这在现实生活中非常重要!

将盘片划分为磁道和扇区的过程是通过工厂格式完成的。



在厂格式化,有时也称为**低级格式化**,是在**磁盘表面放置物理标记**的过程。这些标记构成将**写入数据(1s和0)**的**实际轨道和扇区**,以及**用于定时和精确磁头定位等的控制标记**。这种**低级工厂格式化**与大多数**IT 管理员熟悉的格式化操作**不同。

系统管理员经常执行的格式化是将文件系统写入先前格式化的磁盘驱动器的过程,这比工厂格式化更像高级操作。大多数人永远不会对磁盘驱动器执行低级格式化。



Real World Scenario

短行程

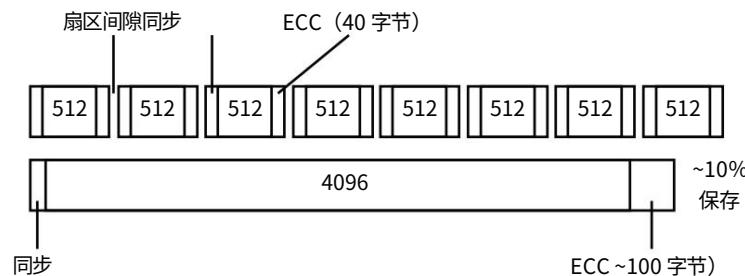
在现实世界中,人们习惯于使用短行程磁盘来提高性能。这种短行程过程只涉及使用磁盘上可用容量的一小部分,比如说 50%。通过仅使用 50% 的容量,您将只使用磁盘的最外层机架(尽管这是特定于实现的,您必须确保这一点)。您还可以减少磁盘上的整体负载,因为与使用 100% 的容量相比,它们上只有一半的数据。

然而,这种做法如今很少使用。首先,它浪费容量,而容量是要花钱的!其次,在大多数情况下,部署固态介质比短行程磁盘更能提高性能。

磁盘上的每个磁道都有一个索引标记,使读/写磁头能够保持精确定位。就像高速公路上的车道分隔线一样。相邻扇区由伺服信号分隔,有助于保持 R/W 磁头“在轨道上”。

虽然历史上扇区都是 512 字节或 520 字节,但一些更现代的磁盘驱动器正在使用更大的扇区大小进行格式化,例如日益流行的高级格式化标准。更大的扇区大小(目前为 4K)可提供更大的容量,因为它们可以实施更高效的完整性检查,例如错误纠正码(ECC)。图 2.8 将 4K 高级格式化驱动器与使用 512 字节扇区格式化的标准驱动器进行了比较。

图 2.8 4K 高级格式



在图 2.8 中,512 字节格式的驱动器需要 320 字节的 ECC,而 4K 格式的驱动器仅需要大约 100 字节。规格将会改变和改进,但这凸显了使用更大扇区的潜力。

有趣的是,4K 在 x86 计算机中是一个神奇的数字。内存页面通常为 4K,并且文件系统通常分为 4K 区段或簇。

行业内将扇区大小已预设且不可更改的磁盘类型称为固定块架构(FBA)。不过,如今这个术语很少使用,因为这是开放系统世界中唯一可以买到的磁盘类型。



当供应商宣传其驱动器的每秒输入输出操作数(IOPS)时,这通常是指 512 字节块操作。这基本上是获得数据表上令人印象深刻的数字的最佳方法。操作大块,这很好,但您可能要记住,这将比大多数实际 I/O 操作小得多。

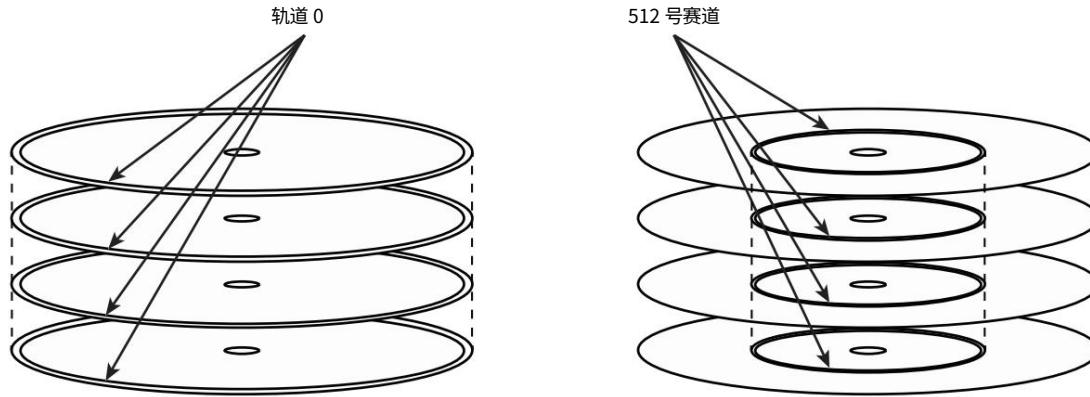
气缸

磁柱是一组磁道,它们直接堆叠在单独的盘片上。例如,假设我们的磁盘驱动器有四个盘片,每个盘片有两个记录表面,这样一共有八个记录表面。每个记录表面都有相同数量的磁道,从外部的 0 号磁道到内部的 1,023 号磁道(这只是为了说明目的而举的例子)。八个记录表面上的 0 号磁道都是外磁道。记录表面 0 上的 0 号磁道位于其他七个记录表面上的 0 号磁道的正上方。

图 2.9 显示了这一概念。左侧是一个四盘片磁盘驱动器,其中突出显示了最外层轨道 0 号轨道。盘片的 0 号轨道形成一个圆柱体,我们可以将其称为

0 柱面。在图 2.9 的右侧,我们有相同的四盘片驱动器,这次突出显示了磁道 512,这些磁道组合形成柱面 512。

图 2.9 圆柱体



逻辑块寻址

磁盘中存储和检索数据的方式可能非常复杂,并且不同磁盘之间可能会有很大差异,具体取决于磁盘的特性 -

扇区大小、盘片数量、磁头数量、区域等等。为了向操作系统隐藏这些复杂性,磁盘驱动器实现了所谓的逻辑块寻址 (LBA)。

逻辑块寻址在每个磁盘驱动器的驱动器控制器中实现,它公开磁盘容量作为简化的地址空间。这是天赐之物!考虑我们之前在图 2.7 中显示的示例盘片,每个盘片有两个区域和 72 个扇区。

假设我们的驱动器有四个盘片,总共有八个记录表面(八个 R/W 磁头)。磁盘驱动器上的位置将通过 CHS 地址在磁盘内部寻址,如下所示:

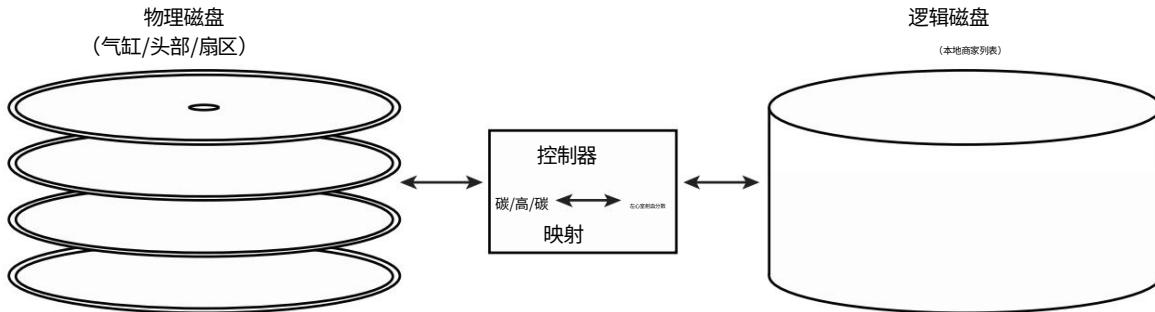
- 柱面 0,磁头 0,扇区 6
- 气缸 3,头部 6,扇区 2
- 柱面 2,磁头 0,扇区 70

这很丑陋、复杂,并且特定于每种驱动器类型。另一方面,LBA 寻址再简单不过了。磁盘控制器公开的 LBA 映射可能只是呈现了 72 个逻辑块的范围,地址为 LBA0 到 LBA71。就是这样!操作系统和文件系统只是将读写定位到 LBA 地址,而不了解磁盘驱动器的底层布局,磁盘驱动器上的控制器负责处理准确的 CHS 地址。

虽然这看起来似乎不多,但考虑到所有现代磁盘驱动器都有多个盘片,每个盘片都有两个记录表面,每个表面都有自己的读/写头。您可以看到这很快就会变得复杂。

有趣的是,由于所有开放系统操作系统和卷管理器都使用 LBA,因此它们永远不知道数据在磁盘上的准确位置!只有磁盘控制器知道这一点,因为它拥有 LBA 映射。听起来令人担忧吗?其实不必;LBA 方案已经安全且成功地使用了很多年。图 2.10 显示了 CHS 地址如何映射到磁盘驱动器控制器上的 LBA 地址。

图 2.10 磁盘控制器将 CHS 映射到 LBA



智能磁盘控制器实现的LBA方案的一个主要优点是
磁盘的操作系统驱动程序已经大大简化和标准化。
您上次为新安装的磁盘驱动器安装自定义驱动程序是什么时候?
操作系统/驱动程序可以简单地向磁盘发出读/写请求,指定 LBA 地址,控制器将这些命令和位置转换为更原始的寻道、搜索、读写命令以及 CHS 位置。

执行器组件

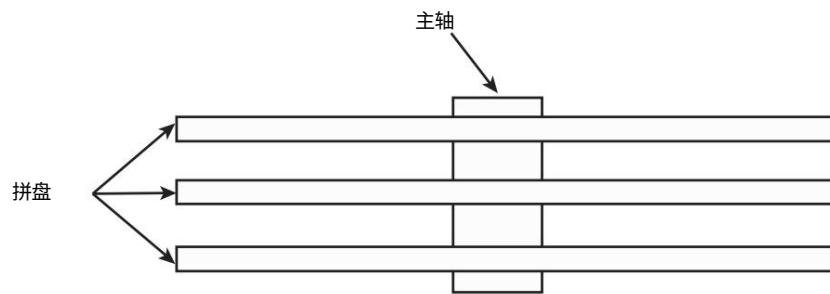
每个磁盘驱动器中下一个重要的物理组件是执行器组件。每个磁盘驱动器都有一个单独的执行器组件,该组件的作用是物理移动 R/W 磁头(在磁盘驱动器固件和控制器的指导下)。当我说移动它们时,我的意思是移动它们!如果你还没有见过 R/W 磁头在盘片表面上来回移动,那你就太没用了!它们移动速度快,精度高。由于磁盘驱动器只有一个执行器,所以所有磁头(每个盘片表面一个)都同步移动。

主轴

磁盘驱动器中的每个盘片都与主轴相连,主轴和主轴电机负责旋转磁盘(盘片)。参见图 2.11。每个磁盘驱动器只有一个主轴,主轴电机旋转盘片的速度非常快,而且非常精确。

每个盘片的中心都连接到主轴上,主轴旋转时,盘片也会旋转。由于只有一个主轴,所有盘片都连接到主轴上,因此所有盘片都同时开始和停止旋转(即主轴启动和停止时)。因此,每个盘片都以相同的速度旋转,称为每分钟转数(RPM)。

图 2.11 盘片和主轴的横截面



控制器

每个磁盘驱动器都配有自己的控制器。该控制器就像是磁盘驱动器的微型计算机。它有处理器和内存，并运行对驱动器运行至关重要的固件。

控制器掩盖了磁盘物理工作的复杂性，并将驱动器上的数据布局抽象为更简单的 LBA 形式，然后呈现给 BIOS 和操作系统等更高级别。更高级别（如操作系统）能够向控制器发出读取和写入等简单命令，然后控制器将它们转换为磁盘驱动器原生的更复杂操作。控制器固件还监控驱动器的运行状况、报告潜在问题并维护坏块图。



NOTE 坏块映射建立在隐藏扇区永远不会暴露给操作系统。当遇到坏扇区（例如无法写入的扇区）时，驱动器固件会映射这些隐藏扇区。每个驱动器都有其一些隐藏块，用于默默地重新映射坏块，而无需操作系统等更高级别知道。

接口和协议

协议可以很容易地被认为是一组命令和通信规则。

磁盘驱动器、固态介质和存储阵列都使用一种协议。（磁盘阵列通常使用多种协议。）然而，在存储领域，每种协议通常也有自己的物理接口规范，这意味着该协议也可以与驱动器的物理接口相关。例如，SATA 驱动器使用 SATA 协议，并且还具有物理 SATA 接口。SAS 和 FC 也是如此；它们各自都有自己的接口和协议。

磁盘驱动器领域最常见的四种协议和接口如下：

- 串行高级技术附件 (SATA)
- 串行连接 SCSI (SAS)
- 近线 SAS (NL-SAS)
- 光纤通道 (FC)

驱动器碎片

随着磁盘驱动器的老化,它开始遇到无法写入或读取的坏块。遇到这些坏块时,磁盘驱动器控制器和固件会将这些坏块映射到磁盘上隐藏的保留区域。这很好,操作系统和更高层永远不必知道这种情况正在发生。然而,这是计算机/磁盘驱动器老化时磁盘驱动器性能显著下降的主要原因之一。

让我们看一个简单的例子。

假设磁盘上 25% 的磁道上的单个扇区被标记为坏扇区,并且已重新映射到磁盘上的其他位置。对这 25% 的磁道中的任何一个进行连续读取都将需要 R/W 磁头转移到另一个位置,以便检索一个重新映射的扇区中的数据。这会导致额外的、以前不必要的定位延迟(寻道时间和旋转延迟)。这称为驱动器碎片,您无法在此级别上对此采取任何措施。在操作系统中运行文件系统碎片整理不会对这种情况有所帮助。唯一要做的就是更换驱动器。

所有这些都是协议和物理接口规范。后两者中,底层命令集是 SCSI,而 SATA 的底层命令集是 ATA。这些协议在磁盘控制器上实现,并决定驱动器中的物理接口类型。

SATA

SATA 根植于台式电脑和笔记本电脑等低端计算市场。

从那时起,它就强行进入高端企业技术市场,但在存储阵列中它很快就被NL-SAS所取代。

在企业技术中,SATA 硬盘是廉价、低性能和高容量的代名词。主要原因是 ATA 命令集不如 SCSI 命令集丰富,因此不太适合高性能工作负载。

因此,磁盘驱动器供应商采用成本较低的组件、较小的缓冲区等等来实现 SATA 驱动器。

然而,企业技术世界并不只追求高性能。低成本、低性能、高容量的磁盘驱动器也绝对有其一席之地。一些例子是备份和归档设备,以及占据存储阵列中自动分层解决方案的较低层。然而,这些要求中的大多数现在都由 NL-SAS 驱动器而不是 SATA 来满足。

SAS

顾名思义,SAS 是一种使用 SCSI 命令集和 SCSI 高级排队机制的串行点对点协议。SCSI 根植于高端企业技术领域,因此它拥有比 SATA 驱动器更丰富的命令集、更好的排队系统,并且物理组件质量通常也更好。所有这些因素都使得基于 SCSI 的驱动器(例如 SAS 和 FC)成为高性能、任务关键型工作负载的最佳选择。

当然,性能是有代价的。SAS 驱动器比 SATA 更贵
容量相似的驱动器。

SAS 的另一个主要优势是 SATA II 和更新的驱动器可以连接到 SAS 网络或背板,并与 SAS 驱动器并存。这使得 SAS 成为构建存储阵列的灵活选择。

SAS 驱动器也是双端口的,这使它们成为外部存储阵列的理想选择,并增加了更高的弹性。关于外部存储阵列和 SAS 驱动器,SAS 驱动器上的每个端口都可以连接到外部存储阵列中的不同控制器。这意味着,如果单个端口、端口连接甚至控制器发生故障,驱动器仍然可以通过幸存的端口访问。从故障端口到幸存端口的故障转移对用户和应用程序来说可以快速且透明。尽管 SAS 驱动器是双端口的,但这些端口以主动/被动模式工作,这意味着只有一个端口处于活动状态并在任何时间点向驱动器发出命令。

最后,SAS 驱动器可以格式化为任意扇区大小,从而使其能够轻松实现 EDP (有时称为 DIF)。使用 EDP,可以使用 520 字节扇区而不是 512 字节扇区。每个扇区的额外 8 个字节用于存储元数据,这些元数据可以确保数据的完整性,确保数据没有损坏。

足球俱乐部

FC 驱动器与 SAS 一样实现 SCSI 命令集和队列,但具有双 FC-AL 或双 FC 点对点接口。它们也适合高性能要求,每 GB 成本相对较高,但在很大程度上已被 SAS 驱动器取代。

NL-SAS

近线 SAS (NL-SAS) 驱动器是 SAS 和 SATA 驱动器的混合体。它们具有 SAS 接口并使用 SAS 协议,但也具有 SATA 驱动器的盘片和 RPM。

重点是什么?它们可以轻松插入 SAS 背板或连接器,并提供 SCSI 命令集和高级队列的优势,同时提供 SATA 常见的大容量。总而言之,它是 SAS 和 SATA 品质的完美结合。

在企业存储领域,NL-SAS 几乎已经超越了 SATA,所有主要阵列供应商都在其阵列中支持 NL-SAS。

排队

所有磁盘驱动器都采用一种称为排队的技术。排队对性能有积极的影响。

排队允许驱动器重新排序 I/O 操作,以便按照针对磁盘布局优化的顺序执行读写命令。这通常会导致重新排序 I/O 操作,以便读写命令尽可能连续,从而尽可能减少磁头移动和旋转延迟。

ATA 命令集实现了本机命令队列 (NCQ)。这提高了 SATA 驱动器的性能,尤其是在并发 IOPS 方面,但不如 SCSI 世界中的同类命令标签队列那么强大。

驱动器大小

在存储领域,大小绝对重要!但当谈到磁盘大小时,了解自己在说什么也很重要。

谈到磁盘驱动器时,您需要了解两种类型的尺寸:

- 容量
- 物理外形

通常,当提到驱动器的大小时,指的是其容量,或者

它可以存储多少数据。过去,这个单位是兆字节 (MB),但现在我们倾向于用千兆字节 (GB) 或兆兆字节 (TB) 来表示。如果你真的想给同行留下深刻印象,你可以谈论你管理的多拍字节 (PB) 资产!

但是,驱动器的大小也可以指驱动器盘片的直径,这

通常称为驱动器的外形尺寸。在现代磁盘驱动器中,有两种流行的 HDD 外形尺寸:

- 3.5 英寸
- 2.5 英寸

正如您所料,2.5 英寸盘片的驱动器在物理上比 3.5 英寸盘片的驱动器要小。而且您可能还会想到,3.5 英寸驱动器通常比 2.5 英寸驱动器可以存储更多数据。这是简单的物理原理;3.5 英寸盘片具有更大的表面积来记录数据。

所有 3.5 英寸驱动器都采用相同尺寸的外壳,这意味着所有 3.5 英寸驱动器都可以安装在服务器或磁盘阵列中的相同插槽中。2.5 英寸驱动器也是如此。所有 2.5 英寸驱动器都采用相同尺寸的外壳,这意味着任何 2.5 英寸驱动器都可以更换为另一个。因此,尽管从技术上讲,2.5 英寸和 3.5 英寸指的是盘片的直径,但当我们提到其中一个驱动器的物理尺寸时,我们通常指的是其外壳的尺寸。表 2.1 显示了这两种外形尺寸的测量值。

表 2.1 HDD 外形尺寸 (大约)

外形尺寸	高度	宽度	深度
3.5 英寸	26 毫米 (1 英寸)	101 毫米 (4 英寸)	146 毫米 (5.75 英寸)
2.5 英寸	15 毫米 (0.6 英寸)	70 毫米 (2.75 英寸)	100 毫米 (3.94 英寸)

有趣的是,固态硬盘制造商已经迅速采用了常见的 2.5-英寸和 3.5 英寸的外形尺寸。这可能是件好事,因为它使得使用固态硬盘 (SSD) 代替机械磁盘变得尽可能简单。



您可能永远都不需要知道这一点,但是 3.5 英寸和 2.5 英寸驱动器尺寸并不完全相同。3.5 英寸的盘片,例如,3.5 英寸硬盘通常具有稍大一些的盘片(约 3.7 英寸),因此之所以被称为 3.5 英寸,是因为它适合放入旧式 3.5 英寸软盘驱动器的驱动器托架中。高性能 10K 和 15K 驱动器的盘片通常比 7.2K 驱动器小,以降低功耗,因为如此快速地旋转大盘片需要更多功率。但只有极客才会喜欢这样的知识,所以要小心告诉别人。

可用容量

说到容量,值得指出的是,购买磁盘驱动器时,您通常无法获得其标称容量。例如,购买 900 GB 驱动器时,您几乎肯定无法从该驱动器获得 900 GB 的可用空间。有几个原因。

原因之一是存储容量可以表示为十进制或以 2 为基数(二进制)。例如,1 GB 在十进制中与在二十进制中略有不同:

十进制 = 1,000,000 字节

基数 2 = 1,048,576 字节



公平地说,供应商在这些问题上使用十进制和太字节,因为他们大多数客户不了解或不关心二进制数字,并且熟悉更常见的术语用法,例如千兆。这些术语也是基于十进制数字的。

你可以想象,当我们谈论很多 GB 甚至 TB 时,这两个数字之间的差异就会变得非常显著:

十进制太字节 = 1,000,000,000,000 字节

以 2 为基数的太字节 = 1,099,511,627,776 字节

在这个 TB 的例子中,差异刚刚超过 9 GB!

如果驱动器制造商以 2 为基数引用,但您的操作系统以 10 为基数引用,则您会看到您可能会感觉自己好像被欺骗了。

其他因素也会造成这种情况。例如,在操作系统中格式化磁盘时,操作系统和文件系统通常会占用一些空间作为开销。同样,在存储阵列中安装磁盘时,阵列通常会削减一些容量作为开销。

还有 RAID 和其他事情需要考虑。底线是,如果您得到的比您预期的要少,请不要感到惊讶。

行驶速度

现在您知道了容量和外形尺寸的含义,让我们继续讨论驱动器速度。

驱动器速度取决于盘片的旋转速度 而且它们旋转得非常快!但

诸如英里每小时之类的度量单位,行驶速度以转数每分钟 (RPM) 来表示。

简单地说,这就是盘片每分钟旋转的次数。

常见的驱动速度如下:

- 5,400 转/分
- 7,200 转/分
- 10,000 转/分
- 15,000 转/分

这些通常分别缩短为 5.4K、7.2K、10K 和 15K。



Earlier I mentioned that drive platters are rigid. This is extremely important when we look at rotational velocities. If the platter isn't stiff enough, RPM等极端速度可能会导致摆动。摆动就是字面意思,意味着任何摆动或不可预测的运动。这肯定会导致物理损害,头碰撞和数据丢失。

如果 RPM 对您来说不够,请这样看:一个 3.5 英寸的盘片以 15,000 RPM 的速度旋转,外缘的速度超过 150 英里/小时 (或大约 240 公里/小时)。这就是我要说的!

RPM 直接转换为驱动器性能。RPM 越高 = 性能越高 (当然价格也越高)。

RPM 和容量之间也存在很强的相关性。通常,驱动器的 RPM 越高,驱动器的容量越低。相反,驱动器的 RPM 越低,容量越高。

另外值得注意的是,10K 和 15K 驱动器通常实现 SCSI 命令集并具有 SAS 或 FC 接口。因此,速度更快的驱动器会实现性能更优化的协议和接口。这是有道理的。此外,5.4K 和 7.2K 驱动器通常实现 ATA 命令集并具有 SATA 接口。

如果你从一开始就阅读这篇文章,那么以下内容现在应该完全有意义:

2.5 英寸,900 GB,10K SAS

3.5 英寸,4 TB,7,200 RPM SATA



Don't expect to see disks with an RPM faster than 15K. The emergence of solid-state media and the difficulty and increased costs required to develop

现在您已经熟悉了磁盘驱动器的结构和主要特性,让我们看看各个部分是如何共同影响驱动器性能的。

磁盘驱动器性能

有多种因素会影响磁盘驱动器的性能,以下部分将介绍其中的一些重要因素。我们已经讨论过其中一些因素,但现在我们将进行更详细的讨论。

影响磁盘驱动器性能的两个主要因素是:

■ 寻道时间

■ 旋转延迟

这两种情况都属于定位延迟类别 基本上就是任何需要磁头移动或等待盘片旋转到位的情况。由于这些都是机械操作,因此这些操作带来的延迟可能非常大。

在讨论其他磁盘驱动器之前,让我们仔细看看它们性能考虑因素,例如 IOPS 和传输速率。

寻道时间

寻道时间是将读写头移动到磁盘上的正确位置(将其定位在正确的磁道和扇区上)所需的时间。寻道时间越快越好,寻道时间以毫秒(ms)表示,因此数字越低越好。



A millisecond is one thousandth of a full second and is expressed as ms.

从计算的角度来看,一毫秒似乎是毫秒级操作(微秒(百万分之一秒)或有时甚至纳秒(百万亿分之一秒)为单位进行测量。例如,3 GHz 处理器上的内存访问可以是 1 纳秒。因此与这些相比,一毫秒确实是很长的时间。

因此,寻道时间会对随机工作负载产生重大影响,过度寻道通常被称为磁头抖动,有时简称为抖动。抖动绝对会降低磁盘驱动器的性能!

在非常随机的小块工作负载下,磁盘花费 90% 以上的时间进行寻道并不罕见。您可以想象,这会严重减慢速度!例如,磁盘在良好的连续工作负载下可能达到 100 MBps,但如果给它一个令人讨厌的随机工作负载,您的速度很容易就会下降到不到 1 MBps!



通常情况下改善小块随机工作负载的最简单和最有效的方法是使用固态介质来解决问题。用固态硬盘替换传统随机工作负载的旋转硬盘。

这里列出了从不同 RPM 驱动器的规格表中获取的一些实际平均寻道时间:

■ 15K 驱动:3.4/3.9 毫秒

■ 7.2K 驱动:8.5/9.5 毫秒

在此列表中,每行的第一个数字表示读取,第二个数字表示写入。



Real World Scenario

磁盘抖动

现实世界中常见的错误是配置错误的备份,这会导致磁头抖动,进而影响性能。应始终将备份配置为备份位置与正在备份的数据位于不同的磁盘上。

一家公司将备份与正在备份的数据存储在同一个磁盘上。通过这样做,他们迫使磁盘对同一个磁盘执行密集的读写操作。这涉及大量磁头移动,从而严重降低了性能。在花费大量时间找到性能问题的根源后,他们将备份区域移至单独的磁盘。这样,一组盘片和磁头读取源数据,而另一组写入备份,一切都运行得更顺畅。

从数据保护的角度来看,将数据备份到不同位置也是备份 101。如果您将数据备份到与源数据相同的位置,而该共享位置发生故障,则您将丢失源数据和备份数据。这可不是什么好事。

旋转延迟

旋转延迟是定位延迟的另一种主要形式。旋转延迟是指磁头定位在正确轨道上之后,正确扇区到达 R/W 磁头下方所需的时间。

旋转延迟与驱动器的 RPM 直接相关。RPM 越高的驱动器旋转延迟越短(即旋转延迟越短)。虽然寻道时间仅在随机工作负载中才有意义,但旋转延迟在随机和连续工作负载中都很重要。

磁盘驱动器供应商通常将旋转延迟列为平均延迟,并以毫秒为单位表示。行业接受的平均延迟是磁盘旋转半圈(即旋转半圈)所需的时间。以下是从实际磁盘驱动器规格表中引用的几个平均延迟数字示例:

- 15K 驱动 = 2.0 毫秒
- 7.2K 驱动 = 4.16 毫秒



首先需要明确的是,所有性能指标都高度依赖于测试中的变量。例如,在测试磁盘能够产生的 IOPS 数量时,结果将在很大程度上取决于 I/O 操作的大小。大型 I/O 操作通常比小型 I/O 操作需要更长的时间。

了解 IOPS

IOPS 是“每秒输入/输出操作数”的首字母缩写,发音为 eye-ops。

IOPS 用于衡量和表达机械磁盘驱动器、固态驱动器和存储阵列的性能。作为性能指标,IOPS 最适用于衡量数据库 I/O 等随机工作负载的性能。

不幸的是,IOPS 可能是最常用和最被滥用的存储性能指标。

供应商,尤其是存储阵列供应商,往往过于热衷于大肆宣传超高的 IOPS 数字,希望您对此印象深刻,以至于忍不住将现金交给他们以换取他们的产品。因此,请谨慎对待存储阵列供应商的 IOPS 数字!不过,别担心。在本节结束时,您应该有足够的知识来揭穿他们的虚张声势。



通常 IOPS 数值,或者供应商提供的任何性能数值,与其说是他们告诉你的,不如说是他们没有告诉你的一切。它们必须成为你理解行业的关键数据。

MPG 是一回事,但当您意识到这个数字是在一辆油箱几乎空了、没有乘客、所有座椅和多余重量都被拆除、使用在普通道路上不合法的专用轮胎、迎着强大的尾风、在油箱中加有非法燃料的情况下从陡峭的山坡上驶下的汽车上实现的,那就完全是另一回事了!您会发现,尤其是对于存储阵列供应商而言,他们的数字通常不能反映现实世界!

那么什么是 I/O 操作?I/O 操作基本上是磁盘或磁盘阵列响应主机(通常是服务器)的请求而执行的读取或写入操作。



您可能会发现有些人将 IOP 操作称为 I/O 操作,这在行业中很常见,但在书面交流中则较少使用。最正确的术语是 IOP。这可能与某些人对比较随意在提到单个通信时,经常会使用缩写版本。当然,如果您将 IOP 分解为输入/输出操作,那就没有意义了。使用 I/O 会更好一些,因为操作暗示得更清楚。无论如何,您可能会听到人们在指 I/O 操作时使用这些术语。

如果真的那么简单就好了!不幸的是,I/O 操作有多种类型。以下是最重要的几种:

- 阅读
- 写
- 随机
- 顺序

- 缓存命中
- 缓存未命中

让问题更加复杂的是,I/O 操作的大小也会产生很大的影响。



每当出现复杂情况时(例如IOPS供应商就会试图迷惑你,希望他们能给你留下深刻印象)-这可能是一个枯燥的话题,但除非你想被欺骗否则你需要学习这点)be a dry subject,

读写 IOPS

读写 IOPS 应该很容易理解。单击正在处理的文档上的“保存”,计算机就会通过一个或多个写入 IOPS 将该文件保存到磁盘。稍后,当您想要打开该文档进行打印时,计算机会通过一个或多个读取 IOPS 从磁盘将其读回。

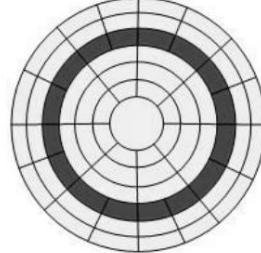
随机 IOPS 与顺序 IOPS

众所周知,机械磁盘驱动器在处理连续工作负载时表现良好,但在处理随机工作负载时表现不佳。如果不进行复杂的配置,你几乎无能为力,而这在固态硬盘如此盛行的世界中毫无意义。

另一方面,固态介质喜欢随机工作负载,并且不能很好地处理顺序工作负载。让我们快速看一个极其简单但有用的例子,它说明了为什么磁盘驱动器更喜欢顺序工作负载而不是随机工作负载。

图 2.12 显示了磁盘驱动器盘片的简单俯视图。单个磁道 (磁道 2,这是从外部开始的第三条轨道 (因为计算机从 0 开始计算轨道,而不是 1),它以十六个扇区 (0-15)突出显示。

图 2.12 单磁道 16 扇区



对该磁道上的八个扇区进行顺序写入将导致扇区 0-7 按顺序写入。这可以在磁盘旋转一次时完成 - 请记住磁盘每分钟旋转数千次。如果需要从磁盘读回相同的数据,则该读取操作也可以通过磁盘旋转一次来满足。不仅

磁盘只需旋转一次,但读/写磁头无需执行任何寻道。因此,定位延迟为零。但是,在等待正确的扇区到达 R/W 磁头下方时,仍然可能存在一些旋转延迟。太棒了!

现在让我们假设一个随机工作负载。为了保持比较公平,我们将读取或写入另外八个扇区的数据,并且我们将只使用第 2 轨,就像我们之前的示例一样。这次,需要按以下顺序读取扇区:5、2、7、8、4、0、6、3。如果 R/W 磁头从扇区 0 上方开始,则此读取模式将需要磁盘旋转五圈:

- Rev 1 将拾取第 5 扇区。
- Rev 2 将接手第 2、7 和 8 区。
- Rev 3 将拾取第 4 扇区。
- Rev 4 将拾取扇区 0 和 6。
- Rev 5 将拾取第 3 扇区。

这会导致更多的工作量和更多的时间来处理读取请求!请记住,这个例子过于简单。当我们把搜索也考虑进去时,事情就变得复杂多了,更不用说多个盘片了!



不过地说,硬盘驱动器及其制造商实际上比我们这个简单的示例聪明得多,它们有一个缓冲区在其中缓存所有 I/O 操作的队列。当 I/O 操作(例如我们的随机工作负载)进入缓冲区时,将根据 I/O 操作以确定是否可以以更有效的顺序进行服务。在我们的示例中,最佳读取顺序将是 0、2、3、4、5、6、7、8。在缓冲区中,每个 I/O 操作都被赋予一个标签,然后每个 I/O 操作都按标签顺序进行服务。显然,队列越大越好。在我们的示例中,将读取命令标记为最佳顺序只需要一次旋转。

缓存命中和缓存未命中

缓存命中 IOPS(有时称为缓存 IOPS)是指从缓存而不是磁盘满足的 IOPS。由于缓存通常是 DRAM,因此与必须从磁盘提供的 I/O 操作相比,由缓存提供的 I/O 操作就像闪电一样迅速。

当查看大型磁盘阵列供应商提供的 IOPS 数字时,了解所引用的 IOPS 数字是缓存命中还是未命中至关重要。缓存命中 IOPS 将比缓存未命中 IOPS 高得多,也更令人印象深刻。在现实世界中,您将同时遇到缓存命中和缓存未命中,尤其是在读取工作负载中,因此纯缓存中的读取 IOPS 数字是无用的,因为它们与现实世界毫无相似之处。除非附带更多信息,例如 70% 读取、30% 写入、100% 随机、4K I/O 大小、避免缓存,否则 600,000 IOPS 之类的声明是无用的。

蒙蔽你的双眼

在现实世界中,供应商会试图让您购买他们的套件。以下是供应商以前使用的两个常见技巧:

众所周知,存储阵列供应商会使用 Iometer 等知名的行业性能工具,但会配置它们来执行极其不切实际的工作负载,以使套件看起来比实际更好。他们曾经玩过的一个花招是只读取或写入零,而系统经过优化,可以很好地处理以零为主的工作负载。具体细节并不重要;重要的是,尽管使用的工具在行业中众所周知,但用于测量其套件的工作负载极其人为,与任何实际工作负载都毫无相似之处。

最后一点极其重要!

另一个老把戏是将示例工作负载设计成所有 I/O 都由缓存满足,而物理磁盘驱动器从不使用。同样,这些测试得出了令人难以置信的高 IOPS 数字,但与该套件在现实世界中预期服务的工作负载类型完全不同。

在读取基准测试数据时,请务必将进行测试的逻辑卷大小与被测系统的缓存量进行比较。如果卷大小不是缓存大小的至少两倍,则在现实世界中您将看不到基准测试所建议的相同性能。

关键是,要警惕营销和销售材料上的绩效数字。如果它看起来好得令人难以置信,那么它很可能就是假的。

计算磁盘驱动器 IOPS

估算磁盘驱动器应能够提供的 IOPS 数量的一种快速简便的方法是使用以下公式:

$$1 / (x + y) \times 1,000$$

其中 x = 平均寻道时间, y = 平均旋转延迟

一个简单的例子,一个驱动器的平均寻道时间为 3.5 毫秒,平均旋转延迟为 2 毫秒,它的 IOPS 值为 181 IOPS:

$$1 / (3.5 + 2) \times 1,000 = 181.81$$

这仅适用于旋转磁盘,而不适用于固态介质或具有大缓存的磁盘阵列。

每秒

每秒兆字节数 (MBps) 是另一个性能指标。该指标用于表示磁盘驱动器或存储阵列每秒可执行的兆字节数。如果 IOPS

最适合测量随机工作负载,而 MBps 在测量连续工作负载 (例如媒体流或大型备份作业) 时最有用。



MBps 与 Mbps 不同 MBps 是每秒兆字节, 而 Mbps 是每秒 megabits per second, whereas Mbps is megabits per second. 1 MBps is equal to 1,000 kilo-

最大和持续传输速率

最大传输速率是您在谈论磁盘驱动器性能时会听到的另一个术语。它通常用于表示在最佳条件下磁盘驱动器的绝对最高可能传输速率 (吞吐量或 MBps)。它通常指驱动器无法维持的突发速率。对这些数字持保留态度。更好的指标是持续传输速率 (STR)。

STR 是磁盘驱动器读取或写入分布在多个磁道上的连续数据的速率。由于它包括分布在多个磁道上的数据,因此它包含了一些相当现实的开销,例如 R/W 磁头切换和磁道切换。这使它成为衡量磁盘驱动器性能的最有用指标之一。

以下是最小传输速率和
来自真实磁盘驱动器规格表的 15K 驱动器的持续传输速率:

- 最大:600 MBps
- 持续:198 MBps

磁盘驱动器可靠性

毫无疑问,磁盘驱动器比以前更加可靠。但是,磁盘驱动器确实会出现故障,而且一旦出现故障,往往会造成灾难性的后果。因此,您绝对必须设计和构建能够应对磁盘驱动器故障的存储解决方案。

值得注意的是,企业级硬盘比便宜的硬盘更可靠,
消费级硬盘。原因如下:

- 企业级驱动器通常采用质量更好的零件按照更高的规格制造。

这总是有帮助的!

- 企业级硬盘通常用于旨在提高

磁盘驱动器和其他计算机设备的可靠性。例如,企业级磁盘驱动器通常安装在高端磁盘阵列中专门设计的笼子中。这些笼子和阵列旨在最大限度地减少振动,并具有优化的空气冷却和受保护的电源馈送。它们还位于温度和湿度受控的数据中心,灰尘和类似问题水平极低。将其与您桌子下面的旧 PC 中的磁盘驱动器进行比较!

平均故障间隔时间

谈到磁盘驱动器的可靠性时,一个常见的统计数据是平均故障间隔时间,简称 MTBF。这是一个令人厌恶的统计数据,可能非常复杂,而且被广泛误解。举例来说,一个流行的 2.5 英寸 1.2 TB 10K SAS 驱动器的 MTBF 值为 200 万小时!这绝对不意味着您应该指望这个驱动器在发生故障之前运行 228 年 (200 万小时)。这个统计数据可能会告诉您,如果您的环境中 228 个这样的磁盘驱动器,那么第一年可能会发生一次故障。除此之外,它并不是一个有用的统计数据。

年故障率

另一个可靠性规范是年故障率 (AFR)。这试图估计磁盘驱动器在一年的充分使用期间发生故障的可能性。上例中的同一 1.2 TB 驱动器的 AFR 小于 1%。

抛开这些统计上的胡言乱语,残酷的现实是磁盘驱动器会出故障。而且当它们出故障时,它们往往会以独特的方式出故障——这意味着当它们坏掉时,它们通常会带走所有的数据!所以你最好做好准备!这时 RAID 之类的东西就会发挥作用。第 4 章“RAID:保证数据安全”中讨论了 RAID。

固态媒体

让我们把所有机械的东西抛在身后,迈向 21 世纪。固态介质领域广阔、令人兴奋,而且发展速度惊人!新技术和创新以惊人的速度涌现。尽管固态介质,甚至闪存,已经存在了几十年,但由于容量、可靠性和成本问题,其普及受到了阻碍。幸运的是,这些问题不再是问题,固态存储真正占据了中心地位。

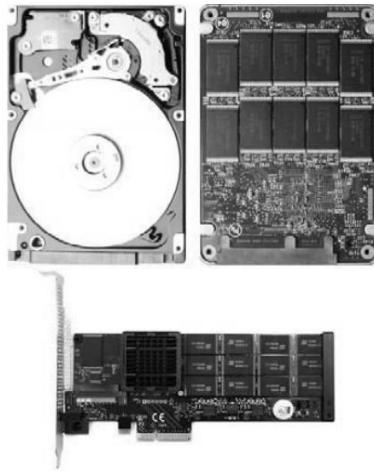
我可能应该首先说一下固态介质有各种形状,尺寸。但我们最感兴趣的是这两个:

固态硬盘 (SSD) 从各方面来看,SSD 的外观和感觉都与硬盘 (HDD) 类似。它们具有相同的形状、大小和颜色。这意味着 SSD 采用常见的 2.5 英寸和 3.5 英寸外形尺寸,并采用相同的 SAS、SATA 和 FC 接口和协议。您几乎可以将它们放入任何当前支持磁盘驱动器的系统中。

PCIe 卡/固态卡 (SSC) 这是 PCIe 扩展卡形式的固态介质。在这种形式下,固态设备可以插入任何服务器或存储阵列中的任何 PCIe 插槽。它们在服务器中很常见,可以为服务器应用程序提供极高的性能和低延迟。它们在高端存储阵列中也越来越受欢迎,用作二级缓存和阵列缓存和分层的扩展。

图 2.13 显示了机械驱动器与固态驱动器以及 PCIe 固态驱动器的并排情况。下方的状态设备。HDD 和 SSD 是按比例缩放的,但 PCIe 卡不是。

图 2.13 HDD (左上)、SSD (右上) 和 PCIe 闪存卡 (底部)



还有不同类型的固态介质,例如闪存、相变存储器 (PCM)、忆阻器、铁电 RAM (FRAM) 等等。不过,本章重点介绍闪存,因为它在业界被广泛采用。在讨论存储的未来时,其他章节会介绍其他技术。

与机械磁盘驱动器不同,固态介质没有机械部件。与计算中的其他所有事物一样,固态设备基于硅/半导体。呼!然而,这使得固态设备与旋转磁盘驱动器具有截然不同的特征和行为模式。以下是一些重要的高级差异: ■ 固态是随机工作负载之王,尤其是随机读取!

■ 由于完全没有定位延迟 (如寻道时间),固态设备的性能更加可预测。

■ 固态介质比机械磁盘需要更少的电力和冷却。

■ 遗憾的是,固态介质的资本支出成本往往要高得多
比机械磁盘更.

固态设备通常还具有小型 DRAM 缓存,用作加速器。
缓存还用于存储元数据 (例如块目录和媒体相关统计信息)。它们通常带有一个小电池或电容器,以便在突然断电时刷新缓存的内容。

不再需要 RPM 或寻道时间

有趣的是,固态介质没有位置或机械延迟的概念。显然,由于没有旋转盘片,因此不存在旋转延迟,同样,由于没有 R/W 磁头,因此不存在寻道时间。正因为如此,在随机工作负载方面,固态介质完全胜过旋转磁盘。对于随机读取尤其如此,其中固态介质的性能通常比旋转磁盘更接近 DRAM。

这无异于改变游戏规则!

闪存

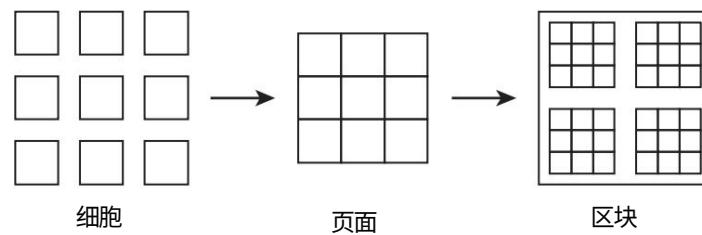
闪存是一种固态存储形式。我的意思是，它基于半导体，没有移动部件，并提供持久存储。我们使用的特定类型的闪存称为 NAND 闪存，它有一些您需要注意的怪癖。另一种形式的闪存是 NOR 闪存，但这在存储和企业技术中并不常用。

写入闪存

写入 NAND 闪存是特别奇怪的事情，您需要确保自己理解这一点。

首先简单介绍一下背景知识。闪存由分组为页面的闪存单元组成。页面大小通常为 4K、8K 或 16K。然后这些页面被分组为更大的块，通常为 128K、256K 或 512K。参见图 2.14。

图 2.14 闪光灯的剖析



全新的 ash 设备出厂时所有单元都设置为 1。闪存单元只能从 1 编程到 0。如果要将单元的值改回 1，则需要通过块擦除操作将其擦除。现在来看看小字：擦除 ash 内存只能在块级别进行！是的，您没看错。闪存只能在块级别擦除。您可能还记得，块可能非常大。如果您刚接触 ash，我建议您再读一遍本段内容，以便理解。



注意：这种块擦除过程（将高电压施加到整个块上并擦除所有内容重置为 1）是由闪存的固有技术（也可能来自快照的 EEPROM）（用于擦除可编程闪存存储器）使用 UV 紫外线进行擦除。

这种奇特的编程/写入灰烬的方式意味着几件重要的事情：

- 第一次写入闪存驱动器时，写入性能将非常快。
块中的每个单元都预设为 1，并且可以单独编程为 0。没有更多了。它很简单，而且速度快如闪电，尽管仍然不如读取性能快。

■ 如果闪存块的任何部分已被写入，则对该块的任何部分的所有后续写入都将需要一个更加复杂且耗时的过程。这个过程是读取/擦除/编程。基本上，控制器将块的当前内容读入缓存，擦除整个块，然后将新内容写入整个块。这显然比写入干净块要慢得多。不过幸运的是，这种情况不应该经常发生，因为在正常运行状态下，闪存驱动器将具有处于预擦除状态的隐藏块，它会将写入重定向到这些块。只有当闪存设备用完这些预擦除的块时，它才必须恢复执行完整的读取/擦除/编程循环。这种情况称为写入悬崖。



读取操作是闪存可以执行的最快操作,擦除操作比读取操作慢近 10 倍,编程操作是所有操作中最糟糕的,比读取操作慢约 100 倍。这就是为什么执行读取/擦除/编程循环不会降低闪存性能的原因。read. And a

为什么这样工作？经济！这个过程需要更少的晶体管和显著降低制造成本。一切都是为了钱！

因此，总结一下，这对您来说可能是新的：第一次写入闪存块时，您可以写入单个单元或页面，这很快。但是，一旦您更改了块中的单个单元，对该块中任何单元的任何后续更新都会慢得多，因为它需要读取/擦除/编程操作。这是一种写入放大的形式，本章后面将进一步讨论。

NAND 闪存的剖析

让我们回到单元、页面和块的话题。这显然与旋转磁盘的柱面、磁头和扇区有很大不同。幸运的是，ash 和其他固态控制器向更广泛的世界呈现了熟悉的 LBA 样式的内容视图。

这意味着操作系统、驱动程序和卷管理器不需要担心这些低级差异。

您需要了解四种主要的NAND灰分类型：

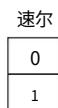
- 单层单元 (SLC)
- 多层单元 (MLC)
- 企业级 MLC (eMLC)
- 三层单元 (TLC)

正如您所看到的，每种方法都有其优点和缺点。

单层单元

最开始的是单层单元 (SLC)。在 SLC 世界中,单个存储单元可以存储 1 位数据;施加到存储单元的电荷可以是开或关(因此可以有二进制 0 或二进制 1)。图 2.15 显示了 SLC 存储单元。

图 2.15 SLC 闪存单元



在三种可用的灰分电池选项中,SLC 提供最高的性能和使用寿命最长。但是,它也是最昂贵的,并且密度/容量最低。



NAND 闪存的写入耐久性常以写入次数来描述。写入耐久性通常指 P/E 周期 (P/E cycle), 即单个闪存单元可以在不损坏的情况下被编程和擦除 100,000 次, 然后可靠性才会受到质疑。这是因为编程和擦除闪存单元的内容会导致物理单元磨损。

由于 SLC 具有高性能和良好的写入耐久性,它非常适合高端计算应用。可悲的是,世界并不太关心高端计算市场。它还不够大,无法赚很多钱。世界关心的是消费市场。因此,市面上的 SLC 灰分越来越少。而且由于它的每 GB 成本最高,也许它不会因为太糟糕而消失。虽然 SLC 确实是所有灰分单元选项中耗电量最少的,但这在现实世界中并不是一个重要的因素。

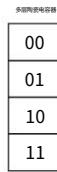
多层单元

多层单元 (MLC) 每个存储单元存储 2 位数据 (00、01、10 或 11),因此每个存储单元存储的数据量是 SLC 的两倍。太棒了!

至少从容量角度来看,它非常出色。从性能和写入耐久性角度来看,它就没那么出色了。传输速率明显较低,并且 P/E 周期通常在 10,000 左右,而不是 SLC 的 100,000。但是,从随机读取性能角度来看,它仍然比旋转磁盘好得多!MLC 已经占领了整个 SLC 市场,以至于 MLC 在高端计算和存储环境中占据主导地位。

有趣的是,MLC 能够通过向单元施加不同级别的电压来在每个存储单元中存储 2 位。因此,电压不是高或低,而是可以是高、中高、中低或低。如图 2.16 所示。

图 2.16 MLC 闪存单元



企业级 MLC

企业级 MLC 闪存,通常称为 eMLC,是一种 MLC 闪存存储器,与标准 MLC 闪存相比,具有更强的纠错能力,因此错误率更低。

基于 eMLC 技术的闪存驱动器和设备也往往配置过多,并可能配备更高规格的闪存控制器,以及在控制器上运行的固件更智能。例如,400 GB eMLC 驱动器实际上可能有 800 GB 的 eMLC 闪存,但隐藏了 400 GB,以便为驱动器提供更高的性能和更好的整体可靠性。

在性能方面,eMLC 使用过度配置技术(本章后面将讨论)将写入数据重定向到隐藏区域中的预擦除单元。这些单元的写入速度比必须经过读取/编程/擦除周期的单元更快。

在可靠性方面,eMLC 驱动器能够在 800 GB 而非 400 GB 的存储单元上平衡写入工作负载,将每个存储单元的工作负载减半,从而有效地使驱动器的使用寿命加倍。

三层电池

三层单元(TLC)完全由消费者驱动,只能通过高度复杂的障眼法在企业级计算环境中发挥作用。应用复杂的固件并提供大量隐藏容量会造成一种错觉,即底层存储比实际情况更可靠。

我确信你已经猜到了:TLC 每个单元存储 3 位,提供以下二进制选项:000、001、010、011、100、101、110、111。正如你所料,它还提供最低的吞吐量和最低的写入耐久性水平。TLC 的写入耐久性通常约为 5,000 个 P/E 周期或更低。

因此,TLC 容量高,性能和可靠性低。这对于消费市场来说很好,但对于企业技术市场来说却不是那么好。图 2.17 显示了 TLC 灰分单元。

图 2.17 TLC 闪存单元

薄层色谱
000
001
010
011
100
101
110
111

Real World Scenario

闪存故障

虽然作为专业人士您需要了解媒体的可靠性,但实际上制造商或供应商应该通过维护合同来承保这种可靠性。

我们来看几个例子。

存储阵列供应商可能会为您提供 MLC 或 TLC 闪存设备,供您在其阵列中使用。这些阵列的维护合同应确保供应商更换任何发生故障的组件。因此,您无需担心可靠性,只需确保您的设计考虑到设备会发生故障并能够无缝应对此类故障(例如通过 RAID)。您不应该关心您的阵列中的 SSD 在去年被更换了两次;供应商会免费为您更换两次。

在 PCIe 闪存卡领域,虽然闪存卡故障可能在保修或维护合同范围内,但不太可能受到 RAID 保护。这意味着虽然故障卡会被更换,但在更换故障部件之前,您的服务可能会中断或性能严重下降。在这样的配置中,可靠的灾难恢复和高可用性(HA)集群非常重要,因为更换这些部件可能需要 4 到 24 小时。

Flash 过度配置

您可能会遇到企业级闪存盘和消费级闪存盘这两个术语。一般来说,闪存盘的质量是一样的。主要区别通常是企业级闪存盘具有控制器保留的大量隐藏容量

以备不时之需。这是一种称为过度配置的技术。一个简单的例子是，一个具有 240 GB 存储容量的闪存驱动器，但控制器只公布 200 GB 可用。隐藏的 40 GB 被控制器用于后台管理过程，如垃圾收集和静态磨损均衡。这可以提高性能并延长设备的使用寿命。在这种情况下，驱动器供应商应始终将驱动器的容量列为 200 GB，而不是 240 GB。

企业级固态硬盘

企业级固态硬盘这个术语没有标准定义。但是，企业级固态硬盘通常比消费级硬盘具有以下优势：

- 更多过度配置的容量
- 更多缓存
- 更复杂的控制器和固件
- 更多渠道
- 电容器可在突然断电时保存 RAM 内容
- 更全面的保修

垃圾收集

垃圾收集是由固态设备控制器执行的后台过程。

通常，垃圾收集涉及清理以下内容：

- 已标记为删除的块
- 仅稀疏填充的块

已标记为删除的块可在后台擦除，这样它们便可随时接收写入，而无需执行令人生畏的读取/擦除/编程循环（写入放大）。这显然可提高写入性能。

可以将稀疏的块合并为更少的块，从而释放更多可擦除且可供快速写入的块。例如，可以将三个仅占用 30% 的块合并并缩减为一个占用 90% 的块，从而在此过程中释放两个块。

这些技术可以提高性能并延长设备的使用寿命。但是，垃圾收集过多或过少之间需要保持平衡。如果垃圾收集过多，则会对存储单元进行过多的写入，并且会不必要地将宝贵的 P/E 周期用于后台任务。如果垃圾收集过少，性能可能会很差。

写入放大

当写入介质的次数高于主机发出的写入次数时，就会发生写入放大。一个简单的例子是当 4K 主机写入 256K 闪存时

已经包含数据的块。如前所述,对已包含数据的块进行任何写入都需要以下步骤:

1.将块的现有内容读入缓存。

2.擦除块的内容。

3.对整个块的新内容进行编程。

这显然会导致向灰烬介质中写入超过 4K 的数据。

写入放大会降低闪存性能并不必要地磨损闪存单元。

必须不惜一切代价避免这种情况。

通道和并行性

通道对于闪存设备性能至关重要。通道越多 = 内部吞吐量越大。如果您可以将足够的 I/O 操作排队到设备,那么通过多个通道同时使用闪存设备内的多个 NAND 芯片可以显著提高性能(并行性)。闪存驱动器通常具有 16 位或 24 位同步接口。

写入耐久性

前面提到过,写入耐久性是指闪存单元额定的编程/擦除 (P/E) 周期数。例如,SLC 闪存单元的额定 P/E 周期数可能为 100,000,而 MLC 则为 10,000。

编程和擦除灰分单元的过程会导致

电池(通过燃烧或擦掉的方式)。一旦电池达到其额定 P/E 循环值,控制器就会将其标记为不可靠,并且不再信任它。

了解和管理固态设备的写入耐久性可以有计划地淘汰和妥善更换此类设备。这样可以减少它们发生灾难性故障的次数。例如,带有固态设备的存储阵列将监控所有固态设备的 P/E 周期数据,并提醒供应商在它们磨损之前主动更换它们。EMC 和 HDS 以及大多数其他供应商会监控旋转磁盘上的错误,并在磁盘损坏之前主动更换磁盘。这可以防止长时间的 RAID 重建等。



大多数存储阵列都会监控旋转硬盘驱动器的运行状况,有时能够预测故障并在其实际发生故障之前主动更换它们。

损耗均衡

磨损均衡是将工作负载分配到所有碎片整理单元的过程,以便一些碎片整理单元不会保持年轻和活泼,而其他碎片整理单元则变得衰老和憔悴。

有两种常见的磨损均衡类型:

■后台磨损均衡

■动态磨损均衡

实时磨损均衡技术会悄悄地将写入内容重定向到未使用的灰烬区域,这些区域所经历的 P/E 循环次数低于原始目标单元。其目的是确保某些单元不会比其他单元经历更多的 P/E 循环次数。

后台磨损均衡,有时也称为静态磨损均衡,是一项基于控制器的日常管理任务。此过程负责处理包含不经常更改的用户数据的灰分单元(这些单元通常不适用于后台磨损均衡)。

后台磨损均衡是必须精心平衡的任务之一,以便磨损均衡本身不会使用过多的 P/E 周期并缩短设备的预期寿命。

固态设备中的缓存

如果没有缓存,存储世界就无法生存。即使是所谓的超快固态设备也能从少量 DRAM 缓存注入中获益(就像向现有的 V8 引擎中添加一氧化二氮一样)。

然而,除了缓存似乎总是带来的明显性能优势之外,在固态世界中,缓存至少提供了两种有助于延长介质寿命的重要功能:

■ 写入合并

■ 书写合并

写入合并是将写入保留在缓存中,直到可以写入整页或整块的过程。假设一个 ash 设备具有 128K 页面大小。32K 写入进入并保留在缓存中。不久之后,64K 写入进入,之后不久,几个 16K 写入进入。总而言之,这些写入总计 128K,现在可以通过单个编程(写入)操作将其写入 ash 块。这比将初始 32K 写入立即提交到 ash 中,然后必须对每个后续写入进行读取/擦除/编程循环要高效得多,也更容易处理 ash 单元。

当特定 LBA 地址在短时间内被反复写入时,就会发生写入合并。如果这些写入(对同一 LBA)中的每一个都保存在缓存中,则只有最新版本的数据才需要写入 ash。这再次减少了对底层 ash 的 P/E 影响。

闪存和固态的成本

无法避免的是:与机械磁盘相比,开箱即用的闪存和固态存储的每 GB 成本相对较高。因此,固态介质的支持者热衷于强调性价比的成本指标。

一个常用的性价比指标是每 I/O 操作花费的美元数。每 I/O 操作花费的美元数是通过将驱动器成本除以其可执行的 IOPS 数量得出的。因此,执行 25,000 IOPS 且成本为 10,000 美元的 SSD 的每 I/O 操作成本为 0.40 美元/I/O 操作,即每 I/O 操作 40 美分。相比之下,执行 200 IOPS 且成本为 2,000 美元的 600 GB 磁盘驱动器。该磁盘驱动器的每 I/O 操作成本为 10 美元/I/O 操作。要从这 600 GB 磁盘驱动器中获得 25,000 IOPS,您需要 125 个驱动器,总共花费 250,000 美元!虽然这是一大笔钱,但 125 个驱动器将需要 125 个驱动器托架,需要电源和冷却,并为您提供大约 75 TB

容量。如果您只需要 2 TB 的容量来实现 10,000 IOPS,那么这简直是过度且笨重的!您可以看到为什么 SSD 人员更喜欢这个成本指标!

虽然向你的首席财务官推销它不容易,但它仍然是一个很好的成本指标,高性能环境中的价值。

写悬崖

在查看固态介质的性能时,在现实世界中需要注意的一点是写入悬崖的概念,即一旦每个闪存单元被写入至少一次,闪存的性能就会显著下降。

当供应商向您展示他们的套件并引用性能数字时,您需要知道这些数字是否是稳定状态数字。稳定状态数字 (IOPS 和 MBps)是在设备填满并且每个块至少被写入一次后实现的。

如果固态设备从未被填充到合理的容量(大约 70% 到 80%),您将很难从性能角度对设备施加压力。设备将始终有足够的可用空间来执行实时 I/O 重定向,从而将 I/O 重定向到已擦除的单元,并且性能比写入必须更新尚未擦除的块时的性能要好得多。

与现实情况相比,供应商经常引用不稳定且不切实际的数字。

混合驱动

混合硬盘是一种相对较新的产品。它们既有旋转盘片,又有固态内存(通常是固态硬盘)。这是两全其美的办法,对吧?不过,目前尚无定论,混合硬盘肯定不是万能的解决方案。

大多数混合硬盘都采用简单的缓存系统;访问最多的数据集数据从旋转磁盘区域转移到驱动器的固态区域,在那里可以更快地访问它们。这对于许多数据集来说是必要的,但在新写入的数据方面,它有一个明显的弱点。新写入的数据通常首先写入旋转磁盘,然后在频繁访问一段时间后,将逐渐进入固态内存。但这通常需要时间。一些混合驱动器会尝试将新数据放入固态,然后如果不经常访问,则将其分层。这两种方法各有利弊。

总而言之,混合硬盘试图兼具容量和速度。哪些内容进入固态硬盘,哪些内容进入旋转磁盘,由硬盘上的固件决定。作为管理员,您不必担心管理该过程。混合硬盘在某些台式电脑和笔记本电脑中很受欢迎,但在企业级服务器和存储阵列中不太可能得到广泛采用。

磁带

天哪!如果说旋转磁盘已经过时了,那么磁带肯定是史前时代的东西了!是的,磁带已经存在很久了,而且它还会继续存在下去。

磁带仍然是备份等长期存储需求的热门选择和存档数据。尽管它很古老,但通常容量大,价格也合理。

从性能角度来看,它非常适合顺序访问,但不适合随机访问。访问磁带上随机位置的数据需要磁带驱动器执行多次快进和倒带操作。不太好。第 11 章“备份和恢复”更详细地讨论了磁带。

进一步的存储考虑

除了本章中已经学到的内容之外,还有一些主题值得你花时间和精力去了解。现在让我们来探讨一下。

缓存

存储领域喜欢缓存,老实说,如果没有缓存,存储性能会很糟糕。但什么是缓存?缓存很简单,只需在磁盘前面放置少量内存,这样就可以从缓存内存中访问经常访问的数据,而不必“转到磁盘”。缓存内存通常基于 DRAM,因此比旋转磁盘快得多。

磁盘驱动器中的缓存内存非常重要,因此所有企业级磁盘驱动器都配备了少量的 DRAM 缓存。在高性能、低容量驱动器中,缓存大小约为 16 MB 是相当常见的,而在低性能、高容量驱动器中,缓存大小则更多在 128 MB 左右。尽管与驱动器的容量相比,缓存很小,但它对随机工作负载很有帮助。通常使用电容器或电池保护缓存,以免断电。

缓存内存也广泛用于存储阵列,高端存储阵列配备 1 TB 缓存的情况并不罕见。虽然这听起来可能很多,但当你仔细观察并发现它高达 1 PB (PB 级) 的磁盘存储时,你就会意识到这是一个相对较小的量。

缓存通常具有如此高的每 GB 成本,以至于存储人员经常使用短语“cache is cash”,暗示 cache 并不便宜。

聪明的

自我监测、分析和报告技术,通常称为 SMART 或 SMART,是硬盘驱动器的行业范围监测标准,旨在预测故障。

预测驱动器故障可让您采取一些措施，并避免驱动器实际发生故障时所需的额外工作。如果您可以预测驱动器即将发生故障，则可以将其内容复制到另一个称为热备用的磁盘。但是，如果驱动器发生故障，则必须从奇偶校验重建其内容。与直接数据复制相比，从奇偶校验重建需要大量计算，并且需要很长时间。

SMART 的规格以及它是否优秀对于存储管理或设计来说并不重要。它是一种默默地开展业务的技术，希望能够帮助您的环境顺利运行。

碎片化

我的电脑运行缓慢。我知道 它需要碎片整理。我相信你一定说过或者听别人说过。但是运行碎片整理能加快你的电脑速度吗？

让我们来谈谈碎片。碎片是存储领域最容易被误解的现象之一，可能是因为碎片有多种类型。本节介绍文件系统、磁盘和 SSD 碎片。

文件系统碎片

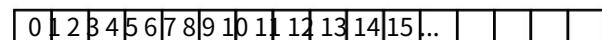
基于操作系统的碎片整理工具（例如 Microsoft Windows 中的工具）可以解决的碎片称为文件系统碎片。当文件在文件系统中不占据连续地址时，就会发生文件系统碎片。这可能不是一个有用的描述，所以让我们看一个简单的例子。



需要说明的是：我经常听到 Linux 管理员说基于 Linux 的文件系统不会产生数据碎片。但事实并非如此。Linux don't fragment data. That is not true.

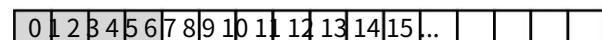
假设你有一个空文件系统，其地址空间为 16 个区（0-15），如下如图2.18所示。

图 2.18 逻辑文件系统地址空间



现在让我们假设您编写了一个名为uber-file的新文件，它需要四个范围。文件系统会将此文件写入范围 0 到 3，如图 2.19 中阴影范围所示。

图 2.19 部分写入的文件系统



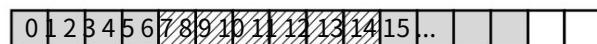
现在假设您写入另一个新文件,这次名为legend-file,它占用六个区段。文件系统将该文件写入区段 4 至 9,如图 2.20 中的对角线所示。

图 2.20 传奇文件写入文件系统



现在假设你想向uber-file 添加更多内容,将其大小增加一倍,它有八个区段。保存文件时,需要另外四个区段来保存它,但文件末尾没有空闲区段,并且文件系统中的其他任何地方都没有八个空闲的连续区段。文件系统必须将uber-file的附加内容写入区段 10 到 13,如图 2.21 所示。

图 2.21 uber-file中添加了更多内容



最终结果是超级文件在文件系统中被分割成碎片。



此时重要的要点是文件伸系统和操作系统不知道文件和文件系统在磁盘驱动器中的布局方式。OS have no knowledge of how the files and filesystem are laid out in the disk drive. 它们只能看到驱动器控制器提供的 LBA 地址空间,无法确保文件占用连续的扇区和磁道。

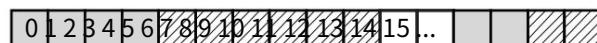
这意味着操作系统或文件系统所做的任何碎片整理工作都不一定会重新组织磁盘布局以使其更加连续。

整理文件系统

让我们继续使用 16 区文件系统并从图 2.21 中停止的地方继续。

要对这个文件系统进行碎片整理,碎片整理工具将使用两个空闲扩展区作为工作区或旋转空间,重新排列扩展区,直到文件在文件系统中尽可能连续地显示。例如,在我们的简单示例中,您可以轻松地将扩展区 4 和 5 排列到空闲扩展区 14 和 15。现在 4 和 5 空闲,您可以将 10 和 11 移动到 4 和 5。此后,文件系统将如图 2.22 所示。

图 2.22 对文件系统进行碎片整理



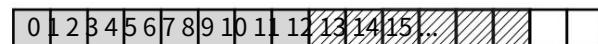
下一步是将扩展区 6 和 7 移动到空闲扩展区 10 和 11。然后将 12 和 13 移动到新释放的范围 6 和 7。此时，您的文件系统将如图 2.23 所示。

图 2.23 继续对文件系统进行碎片整理



此时，uber-file 不再有碎片。您可以看到，将扩展区 14 和 15 移动到 12 和 13 将导致 legend-file 也不再有碎片，如图 2.24 所示。

图 2.24 整理后的文件系统



太棒了。现在您的文件系统已没有碎片，性能无疑会有所提高。但这并不是全部！磁盘碎片仍然存在，这通常会对性能产生更大的影响，而且您对此无能为力。

磁盘碎片

磁盘碎片是指文件在系统中可能完全没有碎片，但在磁盘上却有碎片。该文件不占用连续的扇区和磁道，因此，访问该文件会产生旋转延迟和寻道时间，如果该文件在磁盘上连续分布，则不会产生这些延迟和寻道时间。

磁盘碎片自然会产生，特别是当磁盘驱动器老化并开始遇到必须重新映射的坏块时。假设您对磁盘进行了 8K 写入，并且有 16 个空闲的连续扇区（每个 512 字节）。通常，对该磁盘进行 8K 写入会导致数据写入这 16 个连续扇区。但是，如果在将数据写入磁盘时发现其中一个扇区是坏的（即无法写入数据），则必须将该扇区重新映射到磁盘上的其他位置。最终，您将得到一个 8K 文件，该文件被写入 15 个连续扇区和磁盘上某个其他随机位置的一个不连续扇区。这意味着读取或写入该文件需要额外的寻道时间和额外的旋转。真可惜！

随着磁盘驱动器的老化，它们必须重新映射越来越多的坏块。这至少是磁盘驱动器和计算机似乎随着年龄增长而变慢的原因之一。

整理文件系统碎片对缓解这种碎片化现象没有多大作用。当这种情况发生时，你能做的最好的事情就是换一个新的磁盘驱动器或一台新电脑。

为什么不应该对 SSD 进行碎片整理

一般来说,你不应该对固态硬盘上的数据进行碎片整理。原因如下:

- 对文件系统进行碎片整理会导致大量的读写操作。不建议将不必要的数据写入固态介质,因为每次写入都会缩短闪存单元的使用寿命。别忘了,每个闪存单元只能被编程/擦除有限次。定期对闪存设备进行碎片整理会很快耗尽有限的周期数,并严重缩短设备的使用寿命。许多现代操作系统不允许您对闪存驱动器进行碎片整理。
- 对固态硬盘进行碎片整理可能会打乱控制器的工作。请记住,固态设备上的控制器在后台工作以优化数据布局,以便将来的写入可以避免写入放大和其他问题的影响。 ■ 固态设备没有移动部件,因此不会受到机械和定位延迟(如寻道时间)的影响。碎片化布局不会像旋转磁盘驱动器那样严重影响性能。

存储成本

谈到成本,有几件事需要考虑。固态介质和机械磁盘涉及两种一般类型的成本。您需要考虑存储的购买价格。您还需要考虑使用存储的运营成本。要做到这一切,您需要知道如何衡量成本。

收购成本

购买存储时最常用的成本指标是每 TB 美元 (\$/TB) 或每 GB 美元 (\$/GB)。这衡量了成本的一小部分——采购成本,而且仅针对容量,不考虑性能。但是,它非常简单且易于理解,因此仍然是购买存储时使用最广泛的成本指标。一个简单的例子是 100 GB 的 SSD (或磁盘驱动器),价格为 1,000 美元。该驱动器的采购成本为每 GB 10 美元。如果同样的 100 GB 驱动器价格为 5,000 美元,则其每 GB 成本为 50 美元。成本/容量:再简单不过了。

一般而言,与旋转磁盘相比,固态介质的每 TB 成本较低。因此,固态硬盘供应商正在推行另一项采购成本指标:每 I/O 操作美元数 (\$/I/O 操作)。该指标衡量成本与性能,在这场竞争中,固态硬盘胜过旋转磁盘。不幸的是,对于固态硬盘供应商来说,每 I/O 操作美元数的使用范围较窄,理解程度也较低,部分原因是 I/O 操作不像直接容量那样容易理解和向财务总监 (FD) 解释。

运营成本

固态介质的运行/操作成本往往比机械磁盘低很多,因为固态介质没有移动部件。因此,无需电力来旋转盘片和移动磁头,因此产生的热量也少得多。在大型 IT 环境中,电力、热量和冷却会大大增加数据中心的运营费用。这意味着部署固态介质可以降低服务器、存储阵列或数据中心的总体 TCO。

衡量成本

谈到存储,衡量成本的方法很少。资本支出 (cap-ex) 和运营支出 (op-ex) 是显而易见的方法。当我们说高转速驱动器通常比低转速驱动器成本更高时,我们指的是资本支出和运营支出成本。在资本支出方面,高转速驱动器具有更高的\$/GB 比率,这意味着您每 GB 支付的费用更高。在运营支出方面,使盘片以 10K 或 15K 的速度旋转比使其以仅仅 5.4K 的速度旋转需要更多的能量。而且 15K 驱动器会比 5.4K 驱动器散发出更多的热量。这两项因素在大型数据中心都可能很重要。

除了这些因素之外,众所周知,存储阵列供应商还会对性能更高的驱动器收取更高的维护费用,这让情况雪上加霜。最终,您可能会为高性能付出高昂的代价。

概括

在本章中,我们深入讨论了机械和固态硬盘的结构和内部工作原理,以便为您学习其他存储知识奠定坚实的基础。我们讨论了影响性能、容量和可靠性的机械和固态硬盘技术的各个方面。我们介绍了缓存对性能的重要性;讨论了自我监控分析、报告技术和碎片化。我们在本章的最后讨论了存储成本。

章节概要

磁盘驱动器性能 磁盘驱动器擅长处理连续工作负载,但难以处理大量随机工作负载。增加服务器或存储阵列可处理的 IOPS 数量的常用方法通常包括添加更多磁盘驱动器。更现代的方法包括添加固态介质。

磁盘驱动器可靠性 尽管磁盘驱动器比以前更加可靠,但它们仍然会发生故障!请确保以处理磁盘驱动器故障的方式设计存储环境。

固态设备性能 固态设备在随机工作负载（尤其是随机读取工作负载）下速度惊人，其性能更接近 DRAM 而非旋转磁盘。但是，如果设备上没有足够的可用空间来运行垃圾收集等后台任务，固态设备的写入性能可能会显著降低。

存储性能指标 了解任何已发布的性能指标背后的背景和细节至关重要。与所有性能指标一样，它们只有在代表真实场景的情况下才有用和有效。

章节 3



存储阵列

本章涵盖的主题：

- ✓ 企业级和中端存储阵列
- ✓ 块存储阵列和 NAS 存储阵列
- ✓ 基于闪存的存储阵列
- ✓ 存储阵列架构
- ✓ 基于阵列的复制和快照技术
- ✓ 精简配置
- ✓ 空间效率
- ✓ 自动分层
- ✓ 存储虚拟化
- ✓ 主机集成技术



本章以第 2 章“存储设备”中介绍的基础知识为基础，展示了各种驱动器技术如何汇集在一起并在存储阵列中使用。它涵盖了全球大多数企业使用的存储阵列的主要类型，包括存储区域网络 (SAN) 和网络附加存储 (NAS) 阵列。您将了解它们如何与主机操作系统、虚拟机管理程序和应用程序集成，以及存储阵列中常见的高级功能。这些功能包括在业务连续性规划中发挥重要作用的技术，例如远程复制和本地快照。它们还包括有助于提高存储利用率并降低运行存储环境成本的空间效率技术 - 例如精简配置、自动分层、重复数据删除和压缩等技术。

存储阵列 设定场景

问：天哪！数据中心角落里那块巨大的旧锡片是干什么的？

A. 这就是存储阵列！

传统上，存储阵列很大，由旋转的磁盘组成，需要占用大量数据中心地面空间，消耗的电力足以供给一个小国。最大的可能超过 10 个机柜长，容纳数千个磁盘驱动器、数百个前端端口和数 TB 的缓存。它们往往装在专用机柜中，这是数据中心经理最可怕的噩梦。最重要的是，它们价格昂贵，管理复杂，而且几乎和花岗岩一样不灵活。如果你想作为顾问赚很多钱，那这很好，但如果你想经营一家能够快速响应业务需求的精简高效的 IT 商店，那就不太好。

图 3.1 显示了一个 11 帧 EMC VMAX 阵列。

图 3.1 11 机架 EMX VMAX 存储阵列



值得庆幸的是,这一切都在改变,这在很大程度上是因为固态存储和云的影响。这两种技术都具有巨大的颠覆性,迫使存储供应商提高水平。存储阵列变得更节能、更易于管理、更能感知应用程序和虚拟机管理程序、更小、更标准化,在某些情况下还更便宜!

在了解详细信息之前,我们先来简单了解一下术语:存储阵列、存储子系统、存储框架和 SAN 阵列这些术语通常用于指代同一事物。

虽然从技术角度来说可能不是最准确的,但本书主要使用术语“存储阵列”或“阵列”,因为它们可能是业界使用最广泛的术语。



有时你会听到人们将存储阵列称为 SAN! 这是完全错误的。SAN 是用于连接存储阵列和主机的专用存储网络。*Right wrong. A SAN is a dedicated storage network used for connecting storage arrays and hosts.*

谈谈细节 *the details.*

What is a Storage Array?

存储阵列是一种计算机系统,专门用于为外部连接的计算机提供存储,通常通过存储网络;这种存储传统上是旋转硬盘,但我们看到存储阵列中固态介质的数量正在增加。大型存储阵列的存储量超过 1 PB 的情况并不罕见,been spinning disk, but we are seeing an increasing number of solid-state media in storage arrays. It is not uncommon for a large storage array to have more than a petabyte (PB) of storage.

存储阵列通过共享网络连接到主机,通常提供高级可靠性和增强功能。存储阵列主要有三种类型: 1. 提供连接到主机的块存储协议 (SCSI 和 SAS)。2. 提供连接到主机的文件系统 (NFS 和 CIFS)。3. 提供连接到主机的对象存储协议 (Amazon S3 和 OpenStack Swift)。

■ 存储 SAN

■ 网络存储

■ 统一 SAN (SANs and NAS)

SAN 存储阵列 (有时称为块存储阵列) 通过基于块的协议提供连接,例如光纤通道 (FC)、以太网光纤通道 (iSCSI)、互联网小型计算机系统接口 (iSCSI) 或串行连接 SCSI (SAS)。块存储阵列通过 SAN 发送低级磁盘驱动器访问命令 (称为 SCSI 命令描述符块 (CDB))。例如 READ(S)、WRITE(S) 块和 READ CAPACITY (SAS)。块存储阵列 send low-level disk-drive access commands called SCSI command descriptor blocks (CDBs) such as READ block, WRITE block, and READ CAPACITY over the SAN.

NAS 存储阵列 (有时称为 file) 通过基于文件的协议 (例如网络文件系统 (NFS) 和 SMB/CIFS) 提供连接。基于文件的协议比低级块命令更在更高的级别。它们使用执行诸如创建文件、重命名文件、锁定文件内的字节范围、关闭文件等操作的命令来操作文件和目录。They manipulate files and directories with commands that do things such as create files, rename files, lock a byte range within a file, close a file, and so on.



尽管该协议实际上是服务器消息块 (SMB),但更多时候它被称为其旧名称通用互联网文件系统 (CIFS)。它的发音为 *sif(s)ed to by its legacy name, Common Internet File System (CIFS). It is pronounced sifs.*

统一存储阵列（有时称为多协议阵列）通过块和文件协议提供共享存储。两全其美，对吧？有时是，有时不是。

所有存储阵列（SAN 和 NAS）的目的都是汇集存储资源，并使这些资源可供通过存储网络连接的主机使用。除此之外，大多数存储阵列还提供以下高级特性和功能：

- 复制
- 快照
- 卸载
- 高可用性和弹性
- 高性能
- 空间效率

虽然存储阵列种类繁多，但每个存储阵列都有一个非功能性目标，即为磁盘驱动器和固态驱动器的生存和发展提供最佳环境和生态系统。这些阵列经过精心设计和调整，可提供最佳冷却气流、减震和清洁的受保护电源，以及执行定期清理磁盘和其他健康检查等任务。基本上，如果您是磁盘驱动器，您会希望生活在存储阵列中！



SCSI 机箱服务 (SES) 是存储阵列为磁盘驱动器提供的服务类型之一。SES 是一种背景技术，您可能永远不需要了解它，但它提供了许多至关重要的服务。SES 监控存储阵列中的电源和电压、风扇和冷却、中板和其他与环境相关的因素。在无人看管的数据中心，SES 可以提醒您，您的一级生产存储阵列的温暖后端可能有一窝鸽子在筑巢。

SAN 存储

如前所述，SAN 存储阵列通过基于块的协议（如 FC、FCoE 和 iSCSI）提供与存储资源的连接。存储资源以 SCSI 逻辑单元号 (LUN) 的形式呈现给主机。将 LUN 视为原始磁盘设备，就操作系统或虚拟机管理程序而言，它的外观和行为与本地安装的磁盘驱动器完全相同 - 基本上是一块原始容量。因此，操作系统/虚拟机管理程序知道它需要格式化 LUN 并向其写入文件系统。

虽然存在一些争议，但与 NAS 存储相比，SAN 存储阵列具有
SAN 存储阵列历来被认为性能更高，成本也更高。它们提供 NAS 阵列上提供的所有快照和复制服务，但由于 SAN 存储阵列不拥有或不了解卷上的文件系统，因此在某些方面，它们与 NAS 同类产品相比略显笨拙。



SAN 阵列被认为比 NAS 阵列性能更高的一个主要原因是 SAN 阵列通常需要专用的光纤通道网络。光纤通道网络通常专用于存储流量,通常采用直通交换技术,并以高吞吐量运行。因此它们的连接带宽通常比共享 1 Gigabit 甚至 10 Gigabit 以太网网络低得多,而这在使用 NAS 阵列时很常见。

NAS 存储

NAS 存储阵列使用文件而不是块。这些阵列通过基于 TCP/IP 的文件共享协议 (如 NFS 和 SMB/CIFS) 提供连接。它们通常用于整合 Windows 和 Linux 文件服务器,其中主机从 NAS 安装导出和共享的方式与从 Linux 或 Windows 文件服务器安装 NFS 或 CIFS 共享的方式完全相同。因此,主机知道这些导出和共享不是本地卷,这意味着无需将文件系统写入已安装的卷,因为这是 NAS 阵列的工作。

Windows 主机想要从 NAS 阵列映射 SMB/CIFS 共享,其映射方式与使用通用命名约定 (UNC) 路径 (例如 \legendary-file-server\shares\tech) 从 Windows 文件服务器映射共享的方式完全相同。

由于 NAS 协议在共享以太网上运行,因此它们通常会受到以下问题的影响:网络延迟比 SAN 存储更高,更容易出现网络相关问题。此外,由于 NAS 存储阵列使用文件和目录,因此它们必须处理文件权限、用户帐户、Active Directory、网络信息服务 (NIS)、文件锁定和其他文件相关技术。一个常见的挑战是将病毒检查与 NAS 存储集成。毫无疑问:NAS 存储与 SAN 存储完全不同。

NAS 阵列通常使用 NetApp 术语来指代,因此这并不奇怪。

经常听到有人说:“我们这个周末要升级生产服务器上的代码。”NAS 控制器通常被称为磁头或 NAS 磁头。因此,诸如“供应商正在现场更换纽约 NAS 上发生故障的磁头”之类的陈述指的是更换发生故障的 NAS 控制器。

NAS 阵列历来被认为比 SAN 阵列更便宜、性能更低。事实并非如此。事实上,由于 NAS 阵列拥有并了解导出卷上使用的底层文件系统,因此了解文件和元数据,因此它们通常比 SAN 阵列更具优势。与大多数东西一样,您可以购买便宜、性能低下的 NAS 设备,也可以深入挖掘并购买昂贵的高性能 NAS。当然,大多数供应商会告诉您他们的 NAS 设备成本低、性能高。在花费公司辛苦赚来的钱之前,请务必确保您知道自己要购买什么。

统一存储

统一阵列提供基于块和文件的存储 SAN 和 NAS。不同的供应商以不同的方式实现统一阵列,但最终结果是网络存储阵列允许主机以块协议上的块 LUN 或文件共享协议上的网络共享的形式访问存储资源。这些阵列非常适合那些需要基于块的存储的应用程序 (例如 Microsoft Exchange Server) 但可能还希望整合文件服务器并使用 NAS 阵列的客户。

有时,这些所谓的统一阵列实际上是一个 SAN 阵列和一个 NAS 阵列,每个阵列都有自己的专用磁盘,用螺栓固定在一起,前面有一个闪亮的门,以隐藏后面的大杂烩。这样的阵列被亲切地称为 Frankenstorage。

Frankenstorage 设计的替代方案是拥有一个单一阵列,该阵列具有一个公共驱动器池,运行一个可处理块和文件的单一微码程序。

每种设计方法都有利弊。Frankenstorage 设计往往更浪费资源,更难管理,但它可以提供更好、更可预测的性能,因为如果你仔细观察,就会发现 SAN 和 NAS 组件实际上是气隙隔离的,并且作为独立系统运行。一般来说,仅使用通用 GUI 将 SAN 阵列和 NAS 阵列连接在一起并不是一个好的设计,表明它采用了传统技术。

存储阵列架构快速导览

在本节中,您将研究大多数存储阵列架构所共有的主要组件,以便为本章的其余部分奠定基础。

图 3.2 显示了存储阵列 (SAN 或 NAS) 的高级框图,概述了主要部件。

从左侧开始,我们有前端端口。这些端口连接到存储网络并允许主机访问和使用导出的存储资源。前端端口通常是 FC 或以太网 (iSCSI、FCoE、NFS、SMB)。希望使用存储阵列共享资源的主机必须通过与存储阵列相同的协议连接到网络。因此,如果您想通过光纤通道访问块 LUN,则需要在服务器中安装光纤通道主机总线适配器 (HBA)。

如果我们移到前端端口的右侧,我们会看到处理器。这些处理器现在几乎都是英特尔 CPU。它们运行阵列固件并控制前端端口以及通过它们进出的 I/O。

如果我们继续往前走,处理器后面就是缓存。这用于加速阵列性能,在基于机械磁盘的阵列中,缓存对于良好的性能至关重要。没有缓存,就没有性能!

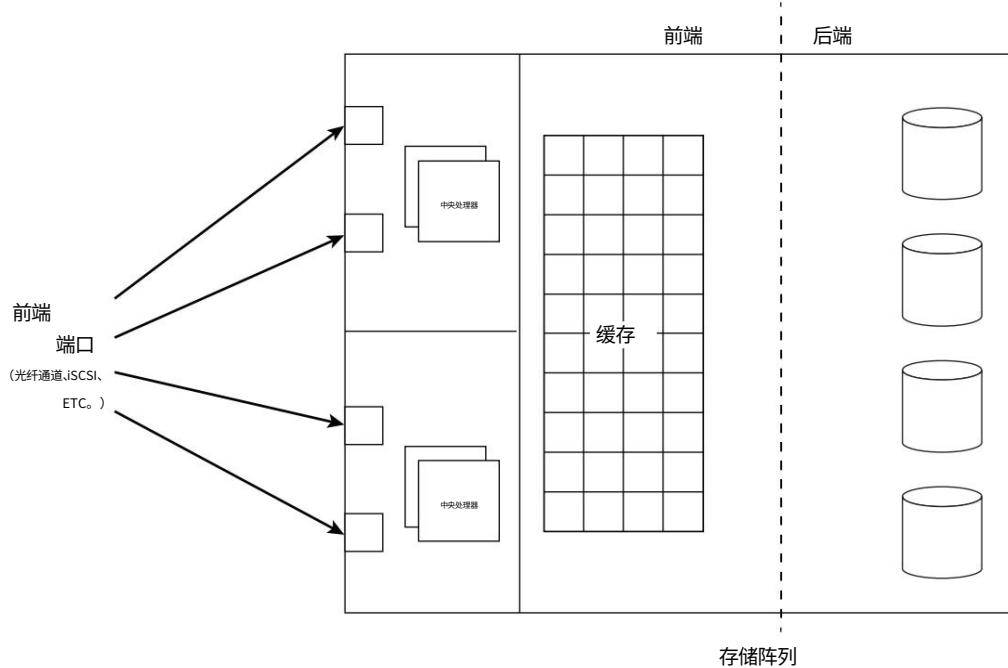
一旦离开缓存,您就进入了后端领域。这里可能有更多的 CPU 和连接到构成后端主要部分的驱动器的端口。

有时控制前端的相同 CPU 也控制后端。



本章后面的“存储阵列的解剖”部分将更详细地讨论存储阵列架构。in the “The Anatomy of a Storage Array” section later in this chapter.

图 3.2 基本存储阵列框图



建筑原则

现在让我们看一下存储阵列常见的一些主要架构原则。

冗余

IT 设计中的冗余原则是每个组件都配备多个,这样当 (不是如果) 组件发生故障时,您的系统能够保持正常运行并继续提供服务。冗余设计通常是 $N+1$,这意味着每个组件 (N) 都有一个备用组件,如果 N 发生故障,该组件可以承担负载。冗余 IT 设计是数字时代企业永不停机的基石,即使是少量停机也会对企业造成灾难性的影响。

不同的存储阵列具有不同的冗余级别,因此您可以获得您所支付的费用。廉价阵列将配备最少的冗余部件,而企业级阵列将为几乎每个组件 (电源、前端端口、CPU、内部路径、缓存模块、驱动器架和驱动器) 内置冗余。

这些组件大部分都是热插拔的。最终目标是让阵列能够应对各种情况,即使出现多个组件故障,也能继续提供 I/O 服务。



热插拔是指无需关闭电源即可更换计算机系统内的物理组件的能力。一个常见的例子是硬盘驱动器,它应该始终能够在系统启动并进行 I/O 服务时进行更换。A common example



Real World Scenario

在系统仍在线的情况下执行重大维护的示例

一家公司花费了很长时间来排除其存储阵列后端间歇性故障。虽然问题不大,但足以进行调查。供应商努力找出问题的根本原因,并制定了更换可能是根本原因的多个组件的系统计划。他们从更换最简单的组件开始,然后更换所有可能的组件,直到只剩下下一个组件。他们更换的组件包括:磁盘驱动器、磁盘驱动器盒、光纤电缆、后端端口和光纤通道仲裁环 (FC-Al) 模块。所有这些组件都是冗余的,并且在系统在线时进行更换。连接的主机均不知道维护,并且没有受到任何服务影响。更换所有这些组件后,故障仍然存在,必须更换驱动器机箱中的中板。由于此过程需要关闭整个驱动器机箱的电源、移除所有驱动器、更换中板并重新组装,因此该过程被认为具有高风险,并在周末进行。尽管整个驱动器机箱断电、被移除、更换组件,然后重新通电,但服务并未中断。整个系统保持运行,就连接的系统而言,没有任何异常。这一切对连接的主机和应用程序都是透明的。

双控制器架构

双控制器架构正如其名称所示:具有两个控制器的存储阵列。



控制器通常以其他名称来称呼,例如节点或引擎,在 NAS 世界中通常称为头端 *nodes or engines*, and commonly as *heads* in the NAS world.

在几乎所有双控制器配置中,虽然两个控制器同时处于活动状态,但它们并不是真正的活动/活动状态。每个 LUN 仅由其中一个控制器拥有。奇数编号的 LUN 通常由一个控制器拥有,而偶数编号的 LUN 由另一个控制器拥有。只有拥有 LUN 的控制器才能直接对其进行读写。从这个意义上讲,双控制器阵列在每个 LUN 上都是活动/被动的 - 一个控制器拥有 LUN,因此对该 LUN 处于活动状态,而另一个控制器处于被动状态,不会直接对其进行读写。这称为非对称逻辑单元访问 (ALUA)。

如果主机通过非所属控制器访问 LUN，则非所属控制器必须将请求转发给所属控制器，从而增加该过程的延迟。

如果这种情况持续存在，则应更改 LUN 的所有权（拥有并可以直接读写 LUN 的控制器）。安装在主机上的特殊软件（称为多路径软件）可帮助管理此问题，并且控制器所有权通常由系统自动更改，无需用户干预。

让我们看一个实施 ALUA 的双控制器系统示例。假设您有一个双控制器系统，其中一个控制器名为 CTL0，另一个控制器名为 CTL1。

该系统有 10 个 LUN。CTL0 拥有所有奇数编号的 LUN，而 CTL1 拥有所有偶数编号的 LUN。当我说控制器拥有一个 LUN 时，我的意思是它对该 LUN 具有独占的读写能力。在此配置中，您可以看到两个控制器都是活动的——这就是有些人将它们称为活动/活动的原因。但是，在任何时间点，只有一个控制器对任何 LUN 具有读/写访问权限。

以下是小提示：如果双控制器系统中的其中一个控制器发生故障，则幸存的控制器将接管故障控制器之前拥有的 LUN。这似乎很好，直到您意识到您可能已经使幸存控制器的工作量翻倍。如果两个控制器都已经很忙，幸存控制器最终可能会承担比其所能处理的工作更多的工作。这绝不是一个好地方！这个故事的寓意是做好故障计划，不要让系统超负荷。

双控制器系统已经存在多年，并且实施起来相对简单，购买成本低。因此，它们在中小型环境中很受欢迎，通常被称为中端阵列。

双控制器的缺点

以下是您在使用双控制器架构时应该注意的主要问题：

■ 当一个控制器发生故障时，另一个控制器将承担发生故障的控制器的工作负载。显然，这会增加幸存控制器的工作负载。很容易看出，如果一个控制器发生故障，则两个控制器均以设计容量的 80% 运行的双控制器阵列将出现性能问题。

■ 当双控制器系统中的某个控制器发生故障时，系统必须进入称为直写模式。在此模式下，在向主机发出确认 (ACK) 之前，必须确保 I/O 已安全传输到后端磁盘。这一点很重要，因为如果在数据位于缓存中（但未镜像到其他控制器，因为该控制器已关闭）时发出 ACK，则如果幸存的控制器也发生故障，则会导致数据丢失。

因为当 I/O 命中缓存时，写通模式不会发出 ACK，所以性能受到严重影响！

■ 双控制器体系结构的可扩展性受到严重限制，并且不能拥有超过两个控制器节点。



We've seen how performance issues arise in production environments that push dual-controller arrays to their limit, only to have the surviving

我曾多次看到,当底层存储阵列被迫在单个控制器上运行时,电子邮件系统无法及时发送和接收邮件。即使更换了发生故障的控制器,邮件队列也可能需要数小时,有时甚至数天才能处理积压邮件并恢复正常服务。当这种情况发生时,电子邮件或存储管理员/架构师的工作就不妙了!

网格存储架构

网格存储架构(有时称为集群或横向扩展架构)可以解决双控制器架构的局限性。它们由两个以上的控制器组成,比传统的双控制器架构更加现代,更适合当今的需求。

真正的主动/主动 LUN 所有权

与大多数双控制器架构相比,主要的设计差异在于,基于网格的阵列中的所有控制器都充当单个逻辑单元,因此以真正的主动/主动配置运行。当我说真正的主动/主动时,我的意思是它不限于我们在双控制器架构中受到限制的任何 ALUA 诡计。在网格架构中,多个节点可以拥有并写入任何和所有 LUN。

这意味着主机可以通过其拥有的任何可用路径向 LUN 发出 I/O,而不会增加非拥有控制器必须将请求转发给拥有控制器时发生的延迟。相比之下,对于双控制器阵列,主机只能通过拥有控制器的路径向特定 LUN 发出 I/O,而不会产生额外的延迟。



The true active-active means that you can independently add more nodes, which expand system without adding GPUs, more ports to the system, as well as more

让我们重新回顾“双控制器架构”部分中的简单示例,其中我们的阵列上有 10 个 LUN,这次是四控制器阵列。由于这是一个网格架构,因此所有四个节点都可以对所有 10 个 LUN 处于活动状态并进行写入。这可能意味着可以显著提高性能。

无缝控制器故障

网格存储阵列还可以更好地处理控制器故障。这是因为它们不应该进入缓存直写模式 我说不应该,是因为

所有阵列都足够聪明,可以做到这一点。如果一个四节点阵列只有一个控制器发生故障,那么仍然会有三个控制器正常运行。除了设计不佳之外,没有理由认为这三个幸存的控制器在更换故障控制器时无法继续提供镜像(受保护)缓存,尽管可用容量较小。

这里有一个简单的例子:假设我们的四控制器阵列中的每个控制器都有 10 GB 的缓存,系统总共有 40 GB 的缓存。为简单起见,我们假设缓存受镜像保护,这意味着 40 GB 中的一半是可用的 - 20 GB。当一个控制器发生故障时,可用的总缓存将减少到 30 GB。这仍然可以在幸存的三个控制器上进行镜像保护,留下约 15 GB 的可用缓存。重要的一点是,虽然缓存量减少了,但系统不必进入缓存直写模式。这会导致节点故障期间的性能下降显著。

此外,四节点阵列如果丢失一个控制器节点,其性能(CPU、缓存和前端端口数量)仅减少了 25%,而不是 50%。

同样,八节点阵列的性能仅会降低 12.5%。再加上无需进入缓存直写模式,您将拥有一个更加可靠和容错的系统。

横向扩展能力

网格阵列的扩展性也远优于双控制器阵列。这是因为您可以添加 CPU、缓存、带宽和磁盘。双控制器架构通常只允许您添加驱动器。

假设您有一个可以处理每秒 10,000 次输入/输出操作(IOPS)的控制器。基于这些控制器的双控制器阵列最多可以处理 20,000 IOPS 尽管如果其中一个控制器发生故障,这个数字将下降回 10,000。即使您在控制器后面添加足够的磁盘驱动器以能够处理 40,000 IOPS,前端的控制器也永远无法将驱动器推至其 40,000 IOPS 的潜力。如果这是基于网格的架构,您还可以向前端添加更多控制器,以便前端足够强大,可以将驱动器推至极限。

提高可扩展性、性能和弹性是有代价的!基于网格的阵列通常被认为是企业级的,价格也相对较高。

企业级阵列

您经常会听到人们将阵列称为企业级。这是什么意思呢?

遗憾的是,没有官方定义,供应商很容易滥用该术语。但通常情况下,企业级一词具有以下含义:

- 多控制器
- 控制器发生故障时影响最小
- 在线不中断升级(NDU)
- 扩展到超过 1,000 个驱动器

■高可用性

■高性能

■可扩展

■始终开启

■可预测的性能

■很贵！

这个清单还可以继续列下去,但基本上,企业级就是你每次都会买的,如果你可以负担得起!

大多数存储供应商都会提供中端和企业级阵列。一般来说

一般来说,基于网格的架构属于企业级类别,而双控制器架构通常被认为是中端。

坚不可摧的企业级阵列

企业级阵列是存储阵列领域的佼佼者,供应商常常使用坚不可摧或坚不可摧等极致形容其企业级产品。

有一次,惠普甚至在实验室中安装了一款企业级 XP 阵列,并用子弹穿透它来证明它是防弹的。子弹穿过阵列,没有造成任何问题。当然,子弹只破坏了阵列一半的控制器逻辑,并且工作负载经过特殊设计,不会受到损坏组件的影响。但无论如何,这还是一个不错的营销手段。

还有一次,惠普将两台 XP 企业级阵列配置为复制配置,并用炸药炸毁了其中一个阵列。幸存的阵列承担了工作负载并继续为连接的应用程序提供服务。

这两部影片都有些极端,但在 YouTube 上观看还很有趣。

有些人认为,阵列要真正成为企业级,就必须支持大型机连接。确实,所有支持大型机的阵列都是企业级的,但并非所有企业级阵列都支持大型机!

中音阵列

企业阵列拥有所有的附加功能并且针对的是财力雄厚的客户,而中端阵列则更针对注重成本的客户。

因此,中端阵列通常与双控制器架构相关联。

虽然中端阵列并不差,但它们肯定落后于企业阵列
在以下领域:

- 性能
- 可扩展性
- 可用性

然而,它们的成本往往要低得多(无论是资本支出还是
和运营支出)。

将阵列称为中端阵列表明它并不低端。尽管这些阵列可能与企业阵列不在同一年级,但它们通常具有良好的性能、可用性和不错的功能集,价格也更实惠。

全闪存阵列

对于存储和IT管理员而言,全闪存阵列的运行和行为与传统存储阵列非常相似,只是速度更快!或者至少这是目标。

说它们的行为相同,我的意思是它们有前端端口,通常是一些DRAM缓存、内部总线、后端驱动器等。它们获取闪存驱动器,将它们汇集在一起,将它们分成卷,并通过RAID或其他类似的保护方案保护它们。许多甚至提供快照和复制服务、精简配置、重复数据删除和压缩。全闪存阵列可以是双控制器、单控制器或横向扩展(毫无疑问,未来还会有其他任何控制器)。

然而,你需要了解其背后的细微而显著的差异
意识到了。我们来看看。

固态技术正在撼动存储行业并改变大部分规则!
这意味着您不能简单地采用经过多年设计和调整的、可以很好地与旋转磁盘配合使用的旧式架构,并将其与固态技术紧密结合。实际上,您可以这样做,而且一些供应商也这样做了,但您绝对不会从阵列或您放入其中的固态技术中获得最大收益。

固态存储改变了大部分规则,因为它的行为方式与旋转磁盘截然不同。这意味着过去20多年来对存储阵列设计进行的所有微调(大量前端缓存、优化的数据放置、预取算法、后端布局、后端路径性能等)都将被抛到九霄云外。

全闪存阵列中的缓存

缓存的主要目的是隐藏旋转磁盘的机械延迟(缓慢)。
好吧,ash不会遭受任何这些弊病,这使得大型、昂贵的缓存变得不那么有用。全ash阵列可能仍会有一个DRAM缓存,因为不管ash有多快,DRAM还是更快。然而,全ash阵列中的DRAM缓存量将更小,主要用于缓存元数据。

缓存算法还需要了解何时与 ash 后端对话。
不再需要执行大量预取和预读。无论如何,从灰烬中读取速度非常快,并且无需等待磁头移动或盘片旋转到位。

闪存支持重复数据删除

说真的。我知道,全灰阵列非常适合一项历来被视为阻碍性能的技术:主存储,这听起来很荒谬。

但请记住,灰烬正在改变规则。

首先,全闪存阵列的每 TB 成本要求数据缩减技术(如重复数据删除和压缩)以实现具有竞争力的\$/TB 成本。如果在现代架构(专为固态存储而设计)上正确实施,主存储的内联重复数据删除可能对性能没有任何影响。它可以这样工作:现代 CPU 带有 ofload 函数,可用于对传入数据执行极低开销的轻量级哈希。如果哈希表明数据可能之前已经见过,因此是重复数据删除的候选,则可以对实际数据执行逐位比较。这就是神奇之处:在闪存介质上,逐位比较速度极快,因为它们本质上是读取操作。在旋转磁盘上,它们很慢!在旋转磁盘架构上实施内联重复数据删除需要强大的哈希,以减少对逐位比较的需求。

这些强哈希在 CPU 周期方面非常昂贵,因此它们会给旋转磁盘架构带来开销。对于基于哈希的阵列来说,这不是问题!

此外,如果在灰分存储前有一个适度的缓存,这些哈希运算就可以执行
异步 在向主机发出 ACK 之后。

重复数据删除还会导致后端数据布局不连续。这会对旋转磁盘架构产生重大性能影响,因为它会导致大量的寻道和旋转延迟。对于基于 ash 的架构来说,这些都不是因素。

尽管如此,重复数据删除对于面向超高性能 0 级市场的全闪存阵列来说可能仍然不太合适。但对于其他任何阵列(包括 1 级),提供重复数据删除功能几乎是强制性的!



在本章中,我使用术语“缓存”、“分层”概念来指代许多不同的东西。例如,组组内的重要的业务线应用程序可能被称为第1层。For example, an important line-of-

应用程序,而我也将高端存储阵列称为 1 级存储阵列。我也用这个术语来指代驱动器。1 级驱动器是速度最快、性能最高的驱动器,而 3 级驱动器是速度最慢、性能最低的驱动器。关于分层的最后一点:业界已习惯使用术语 0 级来指代超高性能驱动器和存储阵列。我偶尔会使用这个术语。

适用于虚拟服务器环境的 Flash

虚拟机管理程序技术往往会产生业界所称的 I/O 混合器效应。这是在单个物理服务器上运行数十台虚拟机的不良副作用。在单个物理服务器上运行如此多的工作负载会导致该服务器的所有 I/O 高度随机化 - 有点像将所有东西扔进食物搅拌机搅拌均匀！我们知道旋转磁盘不喜欢随机 I/O。

另一方面，固态存储实际上就是为此而生，使得全闪存阵列成为高度虚拟化工作负载的理想选择。

0 级或 1 级性能

某些全闪存阵列的目标存在明显分歧。有些阵列力图开拓新市场，即所谓的 0 级市场，在这个市场中，超高性能和每 IOP 成本 (\$/IOP) 是关键。这些阵列是存储领域的佼佼者，拥有数百万的 IOPS。另一方面，全闪存阵列瞄准的是当前的 1 级存储市场。这些阵列比现有的 1 级旋转磁盘阵列具有更高且更可预测的性能，但它们的性能并不在 0 级市场的 I/O 佼佼者范围内。相反，它们更注重特性、功能和实施节省空间的技术（如重复数据删除），以便以具有竞争力的每 GB 美元 (\$/GB) 成本进入市场。

存储阵列的优点

存储阵列允许您集中存储资源，从而更有效地利用容量和性能。从性能角度来看，通过并行使用资源，您可以从套件中榨取更多性能。这让您更好地利用容量和性能。

容量

从容量角度来看，存储阵列有助于防止闲置容量（也称为容量孤岛）的累积。让我们看一个简单的例子：10 台服务器，每台都有 1 TB 的本地存储。其中五台服务器已使用超过 900 GB 并需要添加更多存储，而其他五台服务器仅使用了 200 GB。由于此存储是本地连接的，因此无法让容量不足的五台服务器使用仅使用了 200 GB 的五台服务器的备用容量。这会导致容量搁浅。

现在我们假设 10 TB 的容量汇集在一个存储阵列上，所有 10 台服务器都已连接。所有可用容量可在 10 台连接的服务器之间动态共享。不存在闲置容量。

表现

存储阵列的性能优势与容量优势类似。继续使用 10 台服务器的示例，我们假设每台服务器都有两个磁盘，每个磁盘能够提供 100 IOPS。现在假设某些服务器需要的 IOPS 超过本地安装的磁盘可以提供的 200 IOPS，而其他服务器几乎没有使用其可用的 IOPS。使用本地连接的存储，每个磁盘驱动器的 IOPS 仅供安装该驱动器的服务器使用。通过将所有驱动器集中到一个能够并行使用它们的存储阵列中，所有 IOPS 都可能可供任何或所有服务器使用。

管理

存储阵列使容量和性能管理更加简单。继续使用我们的 10 台服务器示例，如果所有存储都本地连接到 10 台服务器，则将有 10 个管理点。但是，如果您使用存储阵列，则可以从单个点管理这些服务器的存储。这在具有数百台甚至数千台服务器的大型环境中可能非常重要，否则这些服务器需要单独管理其存储。

高级功能

存储阵列往往提供高级功能，包括复制、快照、精简配置、重复数据删除、压缩、高可用性以及操作系统/虚拟机管理程序负载降低。当然，云现在正在威胁存储阵列的霸主地位，迫使存储阵列供应商在许多方面提高自己的竞争力。

提高可靠性

只要存储阵列至少有两个控制器（例如双控制器系统），它就可以承受任何单个组件（包括整个控制器）的故障，而不会丢失数据。当然，在更换故障组件之前，性能可能会受到影响；但是，您不会丢失数据。但对于直接附加存储 (DAS) 方法，情况并非如此，因为存储直接安装在服务器中。在 DAS 方法中，服务器主板或内部存储控制器的故障可能会导致系统丢失数据。

存储阵列的缺点

尽管存储阵列有很多优点，但也存在一些缺点。在本节中，您将了解其中的一些缺点。

潜伏

延迟是存储阵列无法解决的一个问题。即使使用速度最快、延迟最低的存储网络，存储阵列的读写延迟也总是比通过内部 PCIe 总线读写本地磁盘的延迟要大。因此，

对于需要超低延迟的利基用例,本地存储选项可能更适合您。如果您需要低延迟和高随机 IOPS,那么本地连接的基于 PCIe 的存储可能是最佳选择。但是,绝大多数应用程序存储要求都可以通过存储阵列来满足。只有超低延迟要求才需要本地连接的存储。

锁定

如果不小心,存储阵列可能会让您陷入困境。您需要投资于重要的基础设施,并投入大量的前期资本支出和持续的维护。在四到五年内,您需要将其作为技术更新周期的一部分进行替换。将服务迁移 to 新的存储阵列作为技术更新的一部分通常本身就是一个小项目。虽然新兴技术使迁移变得更容易,但这仍然是一个很大的难题。购买存储时,请务必考虑技术更新要求。虽然五年似乎还很遥远,但当您更新套件时,如果您必须花费大量精力迁移到新的存储平台,您会后悔的。

成本

无法避免的是:存储阵列通常价格昂贵。但是,一分钱一分货,对吧?您需要与存储供应商讨价还价。如果他们给您的价格不够优惠,那么这是一个自由市场,有许多其他供应商非常乐意向您出售他们的套件。不要犯这样的错误:只相信销售人员而不做调查。

存储阵列的剖析

现在让我们深入了解一下。图 3.3 显示了存储阵列的主要组件 (与之前在图 3.2 中看到的相同)。

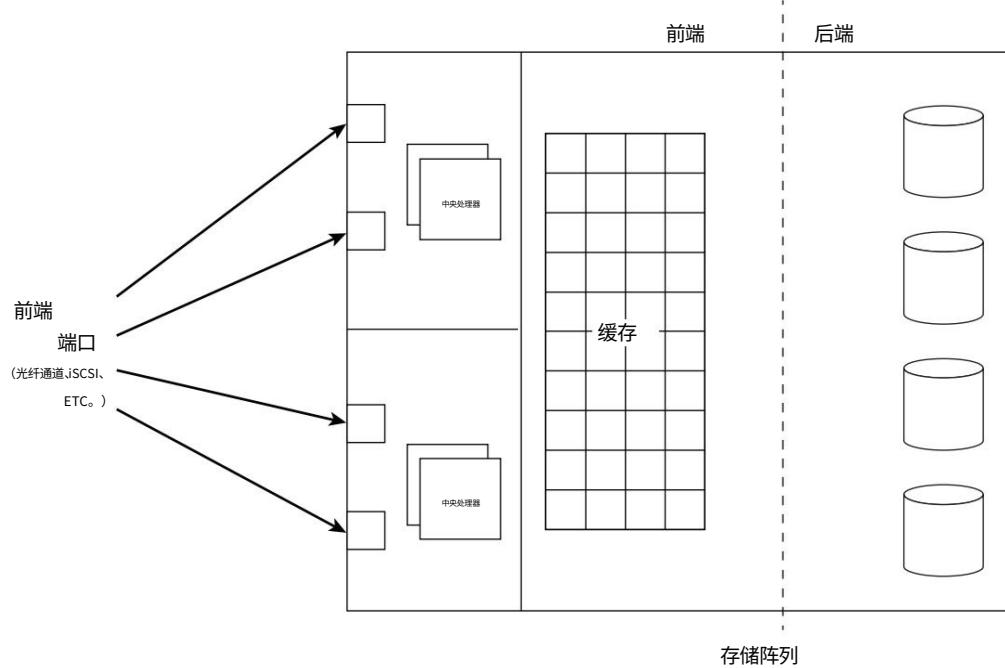
我们将从前端开始,然后按照数组的方式一直到后端 (图 3.3 从左到右)。

前端

存储阵列的前端是阵列与存储网络和主机交互的地方,也可以称之为阵列的进出网关。这也是大多数神奇的事情发生的地方。

如果您了解 SCSI,SAN 存储阵列上的前端端口将充当主机 I/O 的 SCSI 目标,而主机中的 HBA 将充当 SCSI 启动器。如果您的阵列是 NAS 阵列,则前端端口是具有 IP 地址和域名系统 (DNS) 主机名的网络端点。

图 3.3 存储阵列的主要组件



端口和连接

服务器通过前端的端口与存储阵列通信,这些端口通常称为前端端口。前端端口的数量和类型取决于存储阵列的类型和大小。大型企业阵列可以有数百个前端端口,并且根据存储阵列的类型,这些端口有以下几种类型:FC、SAS 和以太网 (FCoE 和 iSCSI 协议)。

大多数高端存储阵列都具有可热插拔的前端端口,这意味着当端口发生故障,可以使用仍处于在线状态且提供 I/O 服务的阵列进行替换。

主机可以通过不同的方式连接到存储阵列,例如,以直接连接的方式将电缆直接从服务器连接到存储阵列,或者通过网络连接。主机和存储阵列之间也可能存在多个连接或路径,所有这些都需要管理。让我们更仔细地研究一下这些概念。

直接连接

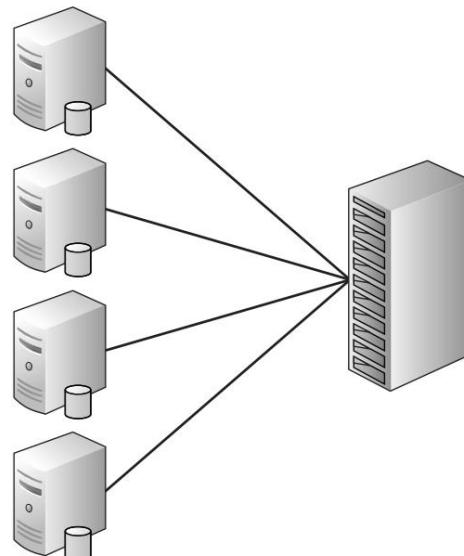
主机可以直接连接到前端端口,而无需中间的 SAN 交换机,这种连接模式称为直接连接。直接连接在小型部署中占有一席之地,但扩展性不佳。在直接连接配置中,主机与存储端口只能一一映射。如果您的存储阵列有八个前端端口,则最多可以连接八台服务器。事实上,由于大多数人遵循行业最佳实践,即拥有多条存储路径 (每台主机至少连接到

两个存储阵列端口),这会将主机与存储端口的比例从一比一降低到一比二,这意味着八端口存储阵列只能满足四台主机的需求,每台主机都有两条通往存储阵列的路径。无论如何,这都是不可扩展的。

SAN 连接

SAN 连接比直接连接更受欢迎且更具可扩展性。SAN 连接在服务器和存储阵列之间放置一个交换机,并允许多台服务器共享同一个存储端口。这种多台服务器共享单个前端端口的方法通常被称为扇入,因为它在图中绘制时类似于手持风扇。扇入如图 3.4 所示。

图 3.4 扇入



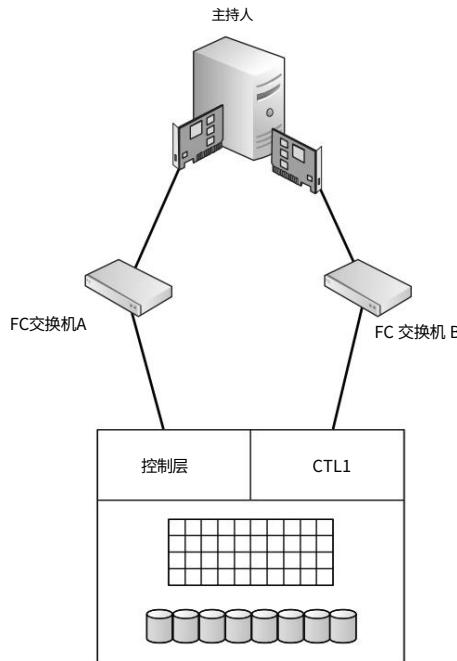
多路径 I/O

所有良好存储设计的核心都是冗余。您需要在每个级别都实现冗余,包括从主机到存储阵列的网络路径。连接到存储的每台服务器必须至少有两个端口来连接到存储网络,这样如果一个端口发生故障或断开连接,另一个端口可用于访问存储。

理想情况下,这些端口将位于单独的 PCIe 卡上。

然而,在微型服务器和刀片服务器环境中,部署的是具有两个端口的单个 HBA。具有两个端口的单个 PCIe HBA 并不像具有两个单独的 PCIe HBA 卡那样冗余,因为双端口 PCIe HBA 是单点故障。这些端口中的每一个都应该连接到独立的交换机,并且每个交换机都应该连接到存储阵列中不同控制器上的不同端口。此设计如图 3.5 所示。

图 3.5 多路径 I/O



基于主机的软件,称为多路径 I/O (MPIO) 软件,控制如何在这些多个链路之间路由或平衡数据负载,以及负责无缝处理故障或应用链路。

常见的MPIO负载均衡算法有：

仅故障转移 当 LUN 的一条路径为主动路径而另一条路径为被动路径时,不会在多条路径之间执行负载平衡。

循环 I/O 在所有路径上交替进行。

最小队列深度 具有最少未完成 I/O 数量的路径将用于下一个 I/O。

所有现代操作系统都带有本机 MPIO 功能,可提供良好的开箱即用路径故障管理和负载平衡。操作系统和虚拟机管理程序 MPIO 架构往往是阵列供应商可以为其编写设备专用模块的框架。这些设备专用模块为主机的 MPIO 框架添加了功能 - 包括专门针对供应商阵列进行调整的附加负载平衡算法。

练习 3.1 引导您配置 MPIO 负载平衡。

练习 3.1

在 Microsoft Windows 中配置 MPIO 负载平衡策略

以下步骤概述了如何使用 Windows MPIO GUI 在 Microsoft Windows 服务器上配置负载平衡策略：

1.在命令行中输入 diskmgmt.msc,打开 Microsoft 磁盘管理插件。

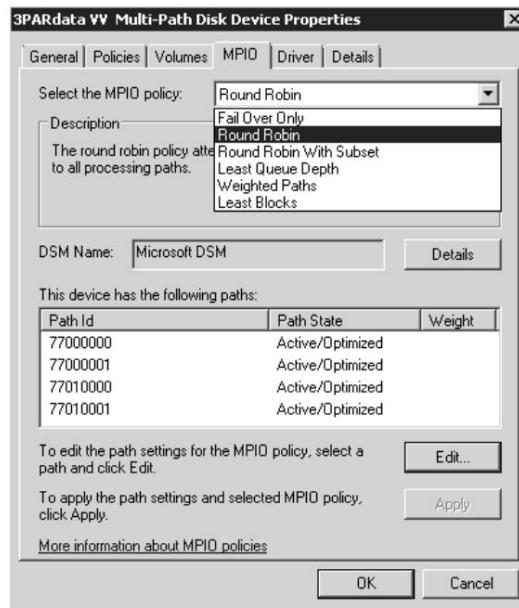
命令行或者运行提示符下。

2.在磁盘管理 UI 中,选择要设置负载平衡策略的磁盘,右键单击它,然后选择属性。

3.单击 MPIO 选项卡。

4.从选择 MPIO 策略下拉列表中,选择要使用的负载平衡策略

您希望使用。



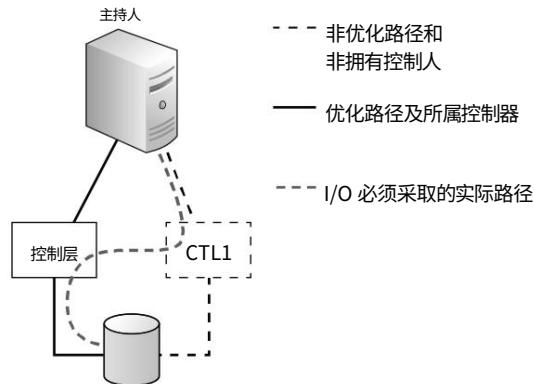
此图显示,通往该 LUN 的所有四条路径均为“活动/优化”,这告诉我们该 LUN 位于基于网格架构的阵列上。如果该 LUN 位于仅支持 ALUA 的双控制器阵列上,则只有一条路径为“活动/优化”,所有其他路径均为仅故障转移。

Microsoft 设备专用模块 (DSM) 提供了一些基本的负载平衡策略,
ies,但特定于供应商的 DSM 可能会提供额外的 cies。

一些供应商还提供自己的多路径软件。最好的例子是 EMC PowerPath。EMC PowerPath 是一款需要付费的授权主机软件。但是,它是专门为 EMC 阵列提供优化的多路径而编写的。它还为比 MPIO 更先进的技术提供了基础。

对于仅支持 ALUA 的双控制器阵列,使用 MPIO 要求将多条 LUN 路径配置为仅故障转移,以确保仅使用到活动控制器(拥有 LUN 的控制器)的路径。通过非优化路径(通过不拥有 LUN 的控制器的路径)访问 LUN 可能会导致该路径的性能不佳。这是因为通过非优化路径访问 LUN 会导致 I/O 必须从不拥有 LUN 的控制器通过控制器互连传输到拥有 LUN 的控制器。这会导致额外的延迟。如图 3.6 所示。

图 3.6 通过非优化路径访问 LUN



前端端口速度

前端端口通常可以配置为以几种不同的速度运行,具体取决于前端端口的类型。例如,16 Gbps FC 前端端口通常可以配置为以 16 Gbps、8 Gbps 或 4 Gbps 运行。在光纤通道 SAN 中,最好对端口速度进行硬编码,而不是依赖自动协商协议。执行此操作时,请确保将电缆两端的端口硬设置为相同的速度。这将确保最可靠和最稳定的配置。但是,基于以太网的存储网络应使用自动协商(AN) 设置,而不应硬编码端口速度。

CPU

CPU 也是前端的关键。CPU(通常是 Intel)用于为前端供电。它们执行固件(存储人员通常称之为微代码),这是存储阵列的大脑。前端 CPU 及其运行的微代码通常负责以下所有操作:

- 处理输入/输出
- 缓存
- 数据完整性

- 复制服务
- 本地复印服务
- 精简配置
- 压缩
- 重复数据删除
- 虚拟机管理程序和操作系统卸载

大型企业级存储阵列通常拥有大约 100 个前端 CPU,以增加前端处理能力并提供冗余。小型低端存储阵列通常拥有很少的 CPU,并使用相同的 CPU 来控制阵列的前端和后端。

供应商热衷于将他们的存储阵列称为智能阵列。虽然这有点夸大其词,而且对任何真正智能的东西都是一种侮辱,但如果存储阵列中有任何智能,那么这种智能就存在于运行该阵列的固件中。



As mentioned earlier, the term microcode is frequently used in the storage industry to refer to the software/firmware that is the brains of a storage array.

您可能还会看到该术语缩写为 ucode。从技术上讲,这实际上应该写为 μcode (符号 μ 是微的 SI 符号),但由于 μ 符号不出现在任何标准 QWERTY 键盘上,因此用字母 u 替代它会更容易。长话短说,术语 ucode

指运行该阵列的微代码。它的发音为 you-code。

因为 rmware 是阵列的大脑,所以你认为 rmware 管理对于任何存储阵列的稳定性和性能都至关重要。如果您不费心保持其最新状态,您将失去供应商的支持,并在出现问题时将自己置于危险之中。您真的不能因为不保持固件最新而将存储置于危险之中。

另一方面,你可能不想处于 rm- 的出血边缘
软件技术。在生产环境中运行最新版的固件 (或预发布版)会大大增加您遇到错误的风险,这在最好的情况下会浪费时间和金钱,在最坏的情况下会暂时使您的组织陷入瘫痪。一个好的经验法则是,在正式发布 (GA) 后至少等待三个月,然后再尝试使用供应商提供的最新最好的固件版本。

有条不紊地将新固件部署到您的资产中也是一个好主意。常见的方法如下:

1. 实验室。首先在实验室中部署并运行至少一个月,测试您的所有

配置,包括测试故障场景和重新启动连接的服务器。

2. DEV。部署到您的开发环境并运行至少一个月。
3. DR。部署到您的灾难恢复 (DR) 阵列 (如果您运行实时/DR 环境)
并运行一周。
4. PROD。部署到您的实时生产阵列。

显然,这个列表必须根据您的环境进行调整,因为不是每个人都拥有功能齐全的实验室、开发环境等等。

不过,有几点值得注意:

■让别人来发现错误。这是允许代码

在你使用它之前,让它在野外成熟三个月。通常在部署后的三个月内,人们已经发现了最严重的错误。这也让供应商有时间修补任何重大问题。

■不要直接部署到实时生产环境。你需要给自己一个机会,清除那些不太重要的环境中的任何潜在问题。

■如果在存储阵列之间进行复制,让它们长时间运行不同版本的固件通常不是一个好主意。通常,一个周末升级 DR 阵列,然后在下一个周末升级实时生产阵列是一个好主意。但是,如果您没有实验室或开发环境来先进行测试,那么在升级实时生产阵列之前,在 DR 中运行两到四周可能是一个更好的主意。



请务必与供应商核实您是否通过受支持的途径进行升级!有些供应商不允许您在相互复制的阵列上运行不同版本的代码!上述规则仅供参考,您需要与供应商或渠道合作伙伴联系,以确保您按规矩办事。*Note to each other. The preceding rules are guidelines only, and you need to engage with your vendor or channel partner to ensure you are doing things by the book.*

规避上述规则的so个明显原因是,如果新版本的代码包含您正在等待的主要错误。*code includes a major bug fix that you are waiting for.*

另外一件事情是许多组织都制定了标准化的年度固件升级计划。每年升级可以让您保持合理的最新状态,并确保您不会落伍。*ows you to keep reasonably up-to-date and ensures you don't fall too far behind.*

关于固件管理,最后需要注意的一点是确保您了解固件升级期间阵列的行为。虽然如今大多数固件升级都倾向于 *GRUB*,但您需要进行检查。您最不想做的事情就是在固件升级过程中发现所有前端端口将在升级过程中脱机,或者整个控制器将脱机一段时间,从而降低升级期间阵列的整体性能!*rough that all front-end ports will go offline part way through the upgrade, or an entire controller will go offline for a while, lowering the overall performance capability of the array during the upgrade!*

LUNs, Volumes, and Shares

Disk drives installed on the backplane and their capacity is combined into volumes by the array. If the array is a standard disk array, these volumes are presented to hosts as network shares, usually NFS or SMB/CIFS. If the array is a SAN array, these volumes are presented to hosts as LUNs. For a host that sees a volume, it looks and behaves exactly like a locally installed disk drive. If the array is a NAS array, these volumes are presented to hosts as network shares, usually NFS or SMB/CIFS.

存储阵列上的 LUN 通常是可扩展的,这意味着您可以相对轻松地增加它们的大小。有时会出现一些复杂情况,例如复制 LUN,但这在任何像样的存储阵列上都不应该成为问题。如果是,那么您购买的就是过时的东西。

另一方面,减小 LUN 的大小则危险得多。因此,在块存储环境中几乎从不执行此操作,因为存储阵列对 LUN 上的文件系统一无所知。但是,NAS 阵列在这方面占了上风,因为它们同时拥有文件系统和卷。知识就是力量,而且我们经常看到好的 NAS 阵列提供缩小卷的功能。

LUN 屏蔽

作为一项安全预防措施,SAN 存储阵列中显示的所有 LUN 都应在阵列上进行屏蔽。LUN 屏蔽是控制哪些服务器可以看到哪些 LUN 的过程。实际上,它是控制哪些 HBA 可以看到哪些 LUN 的过程。它基本上是访问控制。如果没有 LUN 屏蔽,存储阵列前端显示的所有 LUN 对所有连接的服务器都是可见的。可以想象,这将是一场安全和数据损坏的噩梦。

如今,LUN 掩蔽几乎总是在存储阵列上执行 在 FC 环境中使用主机 HBA 的全球端口名称 (WWPN),在 iSCSI 环境中使用 IP 地址或 iSCSI 限定名称 (IQN)。例如,在存储阵列上,您在前端端口上显示 LUN。该端口上有一个访问控制列表,用于确定允许哪些主机 HBA WWPN 访问哪些 LUN。启用 LUN 掩蔽后,如果您的主机 HBA WWPN 不在前端端口的访问控制列表中,您将无法在该端口上看到任何 LUN。它易于实施,并且是所有存储阵列的标准。实施它!



Real World Scenario

正确退役服务器的重要性

存储环境中常见的疏忽是在退役服务器时不清理旧配置。大多数公司在某个时候都会退役服务器,然后将其重建用于其他用途,但不会让存储团队参与退役过程。重建并启动服务器后,它仍然可以看到以前使用的所有旧存储。这是因为服务器仍然具有相同的 HBA 卡和相同的 WWPN,并且这些 WWPN 从未从 SAN 分区和基于阵列的 LUN 掩码中清除。在大多数情况下,这只是一个麻烦,但如果服务器曾经位于可以访问共享存储的集群中,那么这可能是一个大问题!

过去,人们也习惯于在主机 HBA 上应用屏蔽规则。在现代存储环境中,这种情况很少再发生。主要原因之一是它从来都不是一个可扩展的解决方案。此外,除了在存储阵列上执行 LUN 屏蔽和在结构中执行 SAN 分区之外,这几乎没有什么好处。练习 3.2 讨论了如何配置 LUN 屏蔽。

练习 3.2**配置 LUN 屏蔽**

以下分步示例说明了在存储阵列上配置 LUN 屏蔽的常用方法。具体方法可能因您使用的阵列而异。

在此示例中,您将为名为 legend-host.nigelpoulton.com 的 ESX 主机配置新的 ESX 主机定义。

1. 在阵列上创建新的主机定义。
2. 为主机定义指定一个 legend-host 名称。这个名称可以是任意的,但通常会与其所代表的主机的主机名匹配。
3. 设置 Persona 11 - VMware 的主机模式或主机角色。这是角色类型用于 HP 3PAR 阵列上的 ESX 主机,并且应该用于所有 ESX 主机定义。根据不同的阵列技术,该值会有所不同。
4. 将 legend-host 中 HBA 的 WWPN 添加到主机定义中。legendary -host 已在您的阵列上配置,您现在可以将卷映射到它。

接下来的步骤将引导您完成将单个 2 TB 卷映射到传奇主机的过程:

5. 选择现有的 2 TB 卷并选择将该卷导出到主机。
6. 选择 legend-host 作为您希望将卷导出到的主机。

完成这些步骤后,将导出一个 2 TB 卷到 legend-host (实际上是步骤 4 中作为 legend-host 主机定义的一部分分配的 HBA WWPN)。其他主机将无法访问此卷,因为阵列上的 ACL 只允许具有步骤 4 中定义的 WWPN 的启动器访问 LUN。

在 NAS 世界中,LUN 屏蔽的等价物是限制卷/共享导出到的 IP 地址或主机名。除此之外,您还可以实现文件和文件夹权限。

LUN 共享

虽然多台主机可以访问和共享同一个块 LUN,但应格外小心!一般来说,只有在集群配置中,才应允许多台服务器访问和写入同一个共享 LUN,其中集群运行适当的集群软件,以确保 LUN 上数据的完整性。通常通过确保每次只有一台服务器可以写入 LUN 来实现。

某些备份软件和设计可能要求备份媒体服务器将共享 LUN 挂载为只读,以便它可以对 LUN 上的数据进行备份。但同样,备份软件将被设计为执行此操作。

如果两台服务器在没有适当的集群软件的情况下写入同一个 LUN，则 LUN 上的数据几乎肯定会损坏。

练习 3.3 检查如何为集群配置 LUN。

练习 3.3

群集配置中的 LUN 呈现

本练习将引导您向包含两个 ESX 主机（分别名为 ESX-1 和 ESX-2）的 ESX 群集配置单个 2 TB LUN。这两个主机已在阵列上定义。

1. 在阵列上创建新的主机集定义。主机集是可以配置主机集的主机组。

保留多个主机定义。

2. 将主机集命名为 legend-ESX-cluster。

3. 将 ESX-1 和 ESX-2 主机添加到主机集。主机集 legend-ESX-cluster

现已在您的阵列上配置并包含两个 ESX 主机及其各自的 WWPN。

下一步是将您的 2 TB 卷映射到主机集。

4. 选择一个现有的 2 TB 卷，并选择将此卷导出到主机集。

5. 选择 legend-ESX-cluster 作为您希望将卷导出到的主机集。

现在，2 TB 卷已呈现给 ESX 主机 ESX-1 和 ESX-2。两个 ESX 主机现在都可以访问此卷。

密集 LUN 和精简 LUN

存储阵列上的 LUN 可以是厚 LUN 或薄 LUN。传统上，它们总是厚 LUN，但现在，它们往往是薄 LUN。我们将在本章后面的“精简配置”部分讨论厚 LUN 和薄 LUN。

缓存

可以说，缓存是存储阵列的心脏。它无疑是阵列的中央车站。在大多数存储阵列上，一切都必须经过缓存。

对阵列的所有写入操作都会先进入缓存，然后再转入后端磁盘。阵列中的所有读取操作都会先进入缓存，然后再发送到主机。即使在单个阵列内克隆卷，也需要从后端读取数据块，将其放入缓存中，然后将其复制到同一后端的新位置。这一切都意味着缓存极其重要。

缓存对于基于闪存的存储阵列也至关重要,但它们通常缓存较少,因为后端基于固态介质,因此并不急需通过缓存来提高性能。基于闪存的存储阵列往往更多地使用 DRAM 缓存进行元数据缓存,而较少使用 DRAM 缓存进行用户数据缓存。

缓存的性能优势

基于旋转磁盘的存储阵列中缓存的存在理由是为了提高性能。

如果 I/O 可以从缓存中得到满足,而无需转到后端的磁盘,那么该 I/O 的服务速度将提高数百倍!基本上,实施良好的缓存会隐藏机械磁盘的较低性能。

与全闪存阵列相比,DRAM 缓存仍然比闪存更快,并且可用于类似的,只是直接的性能优势并不那么明显。

写入 I/O 必须进入存储阵列,并受到两个独立保护

缓存区域,然后才能向主机发出确认(ACK)。该写入 I/O 随后将在稍后的某个时间点转至后端。此行为显著提高了 ACK 的速度。此操作方式称为写回缓存。如果缓存出现故障,以至于传入的写入无法镜像到缓存(写入两个单独的缓存区域),则阵列将不会发出 ACK,直到数据安全到达后端。此操作模式称为写通模式,对阵列的写入性能产生巨大的负面影响,从而降低磁盘的性能。

谈到性能和直写模式,存储阵列的缓存可能会超载,特别是当后端没有足够的性能来允许在写入活动高峰期间以足够快的速度将缓存中的数据转出时。在这些情况下,缓存可能会被填满到通常称为高写入待处理水位线的点。一旦达到此水位线,阵列就会进入强制刷新模式或紧急缓存转出模式,在这种模式下,它们实际上以缓存直写模式运行并向主机发出命令,迫使它们降低发送 I/O 的速率。这不是一个好的情况!

读缓存和写缓存

用户数据的缓存内存也分为读取和写入区域。阵列是否允许/要求您手动将缓存资源分为读取缓存和写入缓存,各有不同。人们往往会对哪种方法最好有自己的看法。让用户决定还是让阵列决定。我更喜欢让阵列决定的方法,因为阵列可以比我更快地做出反应,并根据当前的 I/O 工作负载做出主动决策。但是,我确实看到了能够在具有特定、众所周知的工作负载的较小阵列上划分缓存资源的优势。但是,在具有随机且频繁变化的工作负载的较大阵列上,我宁愿将此留给阵列。



关于镜像缓存:通常的做法是只镜像写入 it is common practice to mirror only writes
倒缓存(读缓存)不需要镜像,因为数据已经存在于后端受保护的磁盘上。如果缓存丢失,可以再次从后端获取读取数据。如果对读缓存进行镜像,则会不必要地浪费宝贵的缓存资源。

数据缓存和控制缓存

高端存储阵列还往往有用于控制数据（元数据）的专用缓存区域和用于用户数据的专用区域。这会增加成本，但可以提高性能和更可预测的缓存性能。基于磁盘的存储阵列依赖缓存来提高用户数据性能以及缓存必须快速访问的元数据。

全闪阵列较少依赖缓存来提高用户数据性能，但仍然广泛利用缓存进行元数据缓存。

缓存命中和缓存未命中

当请求的数据已在缓存中可用时，读取请求会发生缓存命中。有时这称为读取命中。读取命中是最快的读取类型。如果数据不存在于缓存中并且必须从后端磁盘获取，则表示发生了缓存未命中或读取未命中。读取未命中的速度可能比读取命中慢得多。您是否获得大量读取命中在很大程度上取决于 I/O 工作负载。如果您的工作负载具有较高的引用局部性，您应该会获得良好的读取命中结果。



引用局部性是指数据在地址空间中的分布范围。具有良好引用局部性的工作负载将频繁访问引用距离较近或仅覆盖地址空间一小部分的数据。例如，大型数据库主要访问过去 24 小时内的数据。

对于写入操作，如果缓存运行正常，并且当数据在缓存中受到保护时可以向主机发出 ACK，则这被称为缓存命中。在功能齐全的阵列上，您应该看到 100% 的缓存写入命中统计数据。

保护缓存

由于缓存对于阵列的平稳运行至关重要，因此它始终受到保护

通常通过镜像和电池。事实上，我不记得见过一个阵列如此糟糕以至于没有受保护/镜像的缓存。

保护缓存的方法有很多种，镜像是最常见的。镜像缓存的方法不止一种。有些方法比其他方法更好。在高层次上，高端阵列将接受到达前端端口的写入，并通过快速内部总线在单个操作中将其双工到两个单独的缓存区域。这很快。低端系统通常会接受写入并将其提交到一个控制器中的缓存，然后通过第二个操作，通过外部总线（可能是以太网）将写入数据复制到另一个控制器中的缓存。这更慢，因为它涉及更多操作和更慢的外部总线。

除了镜像之外，所有良好的存储阵列都会配备电池，以便在主电源断电时为缓存供电。请记住，DRAM 缓存是易失性的，当您切断电源时，其内容就会丢失。这些电池通常用于提供足够的电力，使阵列能够在正常断电之前将缓存内容转移到后端，或者为缓存双列直插式内存模块 (DIMM) 提供足够的电量，以保证其内容安全，直到电源恢复。

正如我提到的缓存镜像,值得注意的是,只有写入才应该是镜像的。写入缓存。这是因为读取的数据已经以受保护的非易失性形式存在于后端,因此将读取镜像到缓存中会浪费缓存资源。



如果使用本地保护缓存向量使缓存保持非易失性series is said to make the cache *nonvolatile*.
非易失性缓存通常被称作非易失性随机存取存储器(NDIMM)as nonvolatile random access mem-

缓存持久性

所有优质的高端存储阵列都经过精心设计,缓存 DIMM 故障甚至控制器故障都不需要阵列进入缓存直写模式。这通常是通过实施具有两个以上控制器的网格架构来实现的。例如,一个四节点阵列最多可以丢失两个控制器节点,然后就无法再镜像/保护缓存中的数据。我们假设这个四控制器节点阵列每个节点有 24 GB 的缓存,总共 96 GB 的缓存。该缓存是镜像的,这意味着只有 48 GB 可用于写入数据。现在假设一个控制器节点死机。这会使可用缓存从 96 GB 减少到 72 GB。由于有三个幸存的控制器,每个控制器都有缓存,因此进入阵列的写入仍然可以在多个节点的缓存中受到保护。

显然,由于控制器故障导致 24 GB 的缓存丢失,现在总体可用缓存减少了,但仍然有 36 GB 的镜像缓存可用于写入缓存。

这种行为非常重要,因为如果你从存储阵列中取出缓存,
特别是基于旋转磁盘的存储阵列,你会发现自己陷入痛苦的世界!

常见的缓存算法和技术

虽然您不需要了解缓存算法的来龙去脉才能成为一名优秀的存储专家,但一些基础知识将在您的职业生涯的某个阶段为您提供良好的帮助。

预取是阵列在检测连续工作负载时使用的常见技术。例如,如果阵列已收到从后端读取 2 MB 连续数据的读取请求,则该阵列通常会将下一组连续数据预取到缓存中,因为主机有很大的概率会提出此要求。

这对于基于磁盘的阵列尤其重要,因为 R/W 磁头已经处于正确位置,无需执行昂贵的后端寻道操作即可获取此数据。预取对于全闪存阵列来说没那么有用,因为全闪存阵列不会像旋转磁盘驱动器那样受到机械和位置延迟的影响。

大多数阵列还对缓存中的数据采用某种形式的最近最少使用 (LRU) 队列算法。LRU 基于将最近访问的数据保留在缓存中的原则,而最近最少访问的数据将从缓存中删除。它通常比这更复杂一些,但 LRU 队列是大多数缓存算法的基础。

闪存缓存

在阵列中,将闪存用作二级 (L2) 缓存的一种形式正变得越来越流行。这些配置仍然具有 L1 DRAM 缓存,但增加了一层闪存缓存。

假设您有一个带有 DRAM 缓存的阵列,该缓存位于旋转磁盘的前端。所有读取请求都会将读取的数据拉入缓存,并一直保留在缓存中,直到它从 LRU 队列中掉出来。

通常,一旦这些数据从 LRU 队列中消失,它就不再位于缓存中,如果您想再次读取它,您必须去后端的磁盘中获取它。这太慢了!如果您的缓存增加了 L2 ash 缓存,当您的数据从 L1 DRAM 缓存中消失时,它会进入 L2 ash 缓存。而且由于 ash 通常比 DRAM 多得多,数据可以在那里停留很长时间,直到被遗忘并最终从所有级别的阵列缓存中被逐出。虽然 L2 ash 缓存不如 L1 DRAM 缓存快,但它们仍然比旋转磁盘快得多,而且相对便宜和大!强烈推荐它们。

闪存缓存可以通过多种方式实现。最简单、最粗糙的实现是在后端使用闪存固态硬盘 (SSD),并将其用作 L2 缓存。这样做的性能劣势在于它们与旋转磁盘位于同一个共享后端。这使得闪存内存与控制器相距甚远,并且访问它们会受到任何后端争用和必须遍历后端而产生的延迟的影响。更好的实现是将 L2 闪存缓存放置在更靠近控制器的位置,例如基于 PCIe 的闪存,它位于与控制器和 L1 DRAM 缓存位于同一主板上的 PCIe 通道上。

外部闪存缓存

一些阵列现在开始与主机服务器中安装的闪存设备集成。这些系统往往允许服务器中基于 PCIe 的闪存资源用作阵列缓存 (通常是读取缓存)的扩展。

他们通过各种方式实现这一点,但这里有一种方法:当服务器向阵列发出读取请求时,相关的读取数据将存储在阵列的缓存中。此数据还存储在服务器的 PCIe 闪存中,以便更接近主机,而不必遍历存储网络并受此类操作导致的延迟影响。一旦进入服务器的 PCIe 闪存,它就可以从阵列的缓存中过期。

对于以读取为主的工作负载的服务器来说,这种方法非常有效。

许多人都想尽可能多地了解阵列缓存。很好。我曾经是其中一员。我曾经关心插槽大小、缓存管理算法、缓存的层次结构等等。这会让你听起来好像知道自己在说什么,但在现实世界中,这对你几乎没有什么好处!此外,大多数存储阵列只允许你进行很少的缓存调整,这可能是一件好事。

后端

存储阵列的后端是指驱动器、驱动器架、后端端口以及控制它们所有的 CPU。

CPU

高端企业存储阵列通常具有专用 CPU,用于控制阵列的后端功能。有时这些 CPU 负责 RAID 奇偶校验

计算 (XOR 计算)、I/O 优化和驱动器清理等日常任务。中低端阵列通常具有较少的 CPU，因此前端和后端操作由相同的 CPU 控制。

后端端口和连接

如今，后端端口和磁盘驱动器连接主要采用 SAS。后端的大多数驱动器都具有 SAS 接口并使用 SAS 协议。串行高级技术附件 (SATA) 和 FC 驱动器曾经很流行，但很快就被 SAS 取代。

99% 的时间里，您不需要关心协议和后端的连接性。如果数据中心中没有空间在阵列控制器附近添加更多磁盘架，FC 就具有优势。通过 FC 连接的驱动器允许您将驱动器放置在距离控制器几米远的地方，位于数据中心的不同过道中。然而，这很少实现。



Real World Scenario

后端冗余的重要性

一位客户需要为其存储阵列之一增加容量，但数据中心内没有空闲的地面空间，这些空间与放置阵列控制器的机架相邻。由于他们的后端是基于 FC 的，他们决定在距离控制器几米远的另一排安装一个装满磁盘驱动器的扩展柜。这种方法在一段时间内运行良好，直到控制器突然与距离控制器柜几米远的扩展柜中的磁盘失去连接。

原来，一名在架空地板下工作的数据中心工程师将一块地砖倾斜掉落，损坏了连接控制器和磁盘驱动器架的 FC 电缆。这花了很长时间进行故障排除和修复，在修复期间，阵列处于停机状态。本可以通过在地板下通过不同的路线铺设电缆来避免这种情况，但在本例中并没有这样做。由于电缆连接不良和数据中心设施发生轻微事故，阵列停运了几个小时。

驱动器

如今，大多数驱动器（无论是机械的还是固态的）都是基于 SATA 或 SAS 的。FC 仍在使用，但这些驱动器现在已经过时了。SAS 在存储阵列中越来越受欢迎，尤其是高端存储阵列，因为它们是双端口的，而 SATA 在台式机和笔记本电脑中仍然很受欢迎。

驱动器往往采用行业标准的 2.5 英寸或 3.5 英寸外形尺寸，其中 2.5 英寸越来越流行。而且后端驱动器始终是可热插拔的！

后端，或者至少是后端的数据，应该始终受到保护，而保护后端数据的最常见方式是 RAID。是的，RAID 技术是

虽然非常古老,但它确实有效、广为人知且实施可靠。大多数现代阵列都采用现代形式的 RAID,例如基于网络的 RAID 和并行化 RAID。当组件发生故障时,这些往往可以提供更好的性能和更快的恢复速度。

大多数现代阵列还在后端执行驱动器池,通常称为宽条纹。我们将在本章后面更详细地介绍这一点。

力量

市场上几乎每个存储阵列都配有双热插拔电源。

然而,高端阵列则更进一步。一般来说,高端存储阵列将采用来自不同来源的多个电源,并且倾向于使用 UPS 供电的三相电源。它们还配备大型电池,当两个电源都断电时,可用于为阵列供电。这些电池可为阵列供电足够长的时间,以将缓存内容转储到磁盘或灰烬中,或者将缓存内容保留很长时间(通常为一两天)。

存储阵列智能

是的,智能这个术语有点夸大,但其中一些东西实际上开始变得聪明起来,特别是那些与堆栈中较高层(如操作系统、虚拟机管理程序和应用程序)集成的东西。

复制

在本书中,复制特指远程复制。复制存在的理由是创建生产数据的远程副本(副本)。假设您在纽约数据中心有一个任务关键型生产数据库,您希望将其复制到波士顿数据中心,以便在纽约的生产系统发生故障时,您在波士顿拥有最新的数据副本。复制就是您实现这一目标所需要的。

这些副本有时被称为复制品,可用于各种用途,但更常见的是它们通常用于以下用途:

- 在发生局部中断时提供业务连续性,正如刚才所述 ■ 测试和开发目的,例如在隔离环境中针对真实数据测试新模型和查询,在这些环境中,它们不会影响生产系统或更改生产数据

业务连续性

数据复制只是稳健的业务连续性(BC)计划的一个组成部分。定期测试业务连续性计划也绝对至关重要。您不会希望第一次使用业务连续性计划时正值灾难发生之时 希望当晚一切顺利。很可能不会!

演练对于管理有效的业务连续性计划至关重要。测试并再次测试它们！



在研究复制类型之前,值得指出的是,大多数复制技术无法帮助从逻辑损坏中(例如病毒或在删除的数据)中恢复。这是因为损坏本身(a(病毒或删除))将被复制到目标阵列。备份和快照是从逻辑损坏中恢复所需的!

基于阵列的复制

在基于阵列的复制中,存储阵列控制数据的复制。一个优点是,存储阵列负责繁重的复制管理工作,因此不会给您的应用程序服务器带来任何开销。缺点是存储阵列不了解应用程序。因此基于存储阵列的复制并不智能。这意味着远程站点的复制数据可能不是干净可靠的应用程序恢复的理想状态,并且与使用基于应用程序的复制相比,通过复制链路发送的数据可能更多。显然,这一点非常重要。因此,基于阵列的复制正在逐渐被越来越流行的基于应用程序的复制所取代。

基于阵列的复制几乎总是要求链路两端使用相同类型的阵列。例如,您无法在波士顿的 EMC VMAX 阵列和纽约的 NetApp FAS 之间进行复制。

基于应用程序的复制

基于应用程序的复制不使用基于存储阵列的复制技术。

首先,这意味着复制数据的开销将转嫁到应用服务器上。

这可能不是理想的,但现在大多数系统都具有足够的 CPU 能力和其他资源来应对这种情况。

基于应用程序的复制的第二个也是更重要的事实是,它具有应用程序感知能力,或者说是智能的。这意味着它能够理解应用程序,以应用程序喜欢的方式复制数据,并确保复制的数据副本处于理想状态,以便应用程序能够快速、顺利地进行恢复。

常见示例包括 Oracle Data Guard 等技术以及 Microsoft SQL Server 和 Microsoft Exchange Server 等 Microsoft 技术附带的本机复制技术。

基于主机和基于虚拟机管理程序的复制

一些逻辑卷管理器和虚拟机管理程序开始提供更好的复制技术,它们有各种形式。一些基于主机/虚拟机管理程序的复制技术既不了解应用程序,也不了解存储阵列。它们只是提供相同的

存储阵列提供非智能复制,但无需将复制负担转移到存储阵列。这不是最好的方案,但通常比较便宜。

还有基于虚拟机管理程序的技术,它们可以执行复制或插入存储阵列复制技术。基于虚拟机管理程序的复制技术的一个例子是 VMware vSphere Replication,它由 VMware Site Recovery Manager 管理。

SRM 还支持基于存储阵列的复制,并可与基于阵列的复制技术集成。它允许阵列执行繁重的复制工作,但将其与 SRM 规划和管理站点故障转移的能力集成在一起。

在 VMware SRM 的两个选项中,内置 vSphere 复制是一项新技术
nology 可能更适合较小的环境,而将 SRM 与基于阵列的复制技术相集成的选项被认为更加强大且具有可扩展性。

这些类型的复制技术正变得越来越流行。但要小心。它们有时可能需要存储阵列复制许可证和虚拟机管理程序软件许可证!

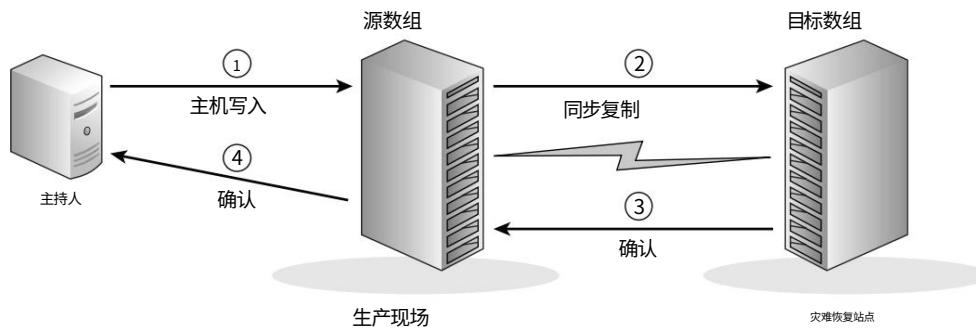
同步复制

同步复制可保证零数据丢失。太棒了!我相信我们都想要这样。
然而,总会有小字注明,这次小字注明同步复制会带来性能成本。

同步复制技术确保写入操作是
在向应用程序发出 ACK 并且应用程序认为写入已提交之前,源阵列和目标阵列上的数据必须得到保护。这很简单。
遗憾的是,等待目标阵列上的写入完成通常会导致显著的延迟。确切的延迟量取决于几个因素,但通常归结为源阵列和目标阵列之间的网络往返时间 (RTT)。网络 RTT 通常也与源阵列和目标阵列之间的距离成正比。

图 3.7 显示了基于阵列的同步复制的故事板。

图 3.7 同步复制情节提要





Because synchronous replication offers zero data loss, this gives a recovery point objective (RPO) of zero. At which level, RPO is the amount of data lost if a failure occurs.

因此,RPO 决定了用于保护业务数据的业务连续性技术。例如,如果应用程序的 RPO 为 1 小时,则每天备份该应用程序是没有用的。

复制距离和延迟

大致来说,源阵列和目标阵列之间的距离为 75 英里时,可能会产生 1-2 毫秒 (ms) 的 RTT。如果您没有同步复制数据,则不会产生 1-2 毫秒的延迟。

说到距离,同步复制配置中可能要覆盖的最大距离大概约为 100 英里。但是,请务必就此类问题咨询阵列供应商,因为他们可能会施加其他限制并根据其特定技术提供建议。您不想自己设计并出错。

处理站点到站点的复制链接时,请确保包括阵列供应商以及网络团队。



Always make sure when referring to the distance between data centers, you are referring to the physical distance, not the network distance. The network distance may be longer than the physical distance.



It is also important to be able to test and guarantee the latency of the link. You may want to get latency guarantees from your Wide Area Network.

复制链接注意事项

部署同步复制时,您还应该了解连接源阵列和目标阵列的网络链路的重要性。如果链路经常处于连接和断开状态,则意味着您的解决方案无法保证零数据丢失,除非在网络链路断开时停止对应用程序的写入。大多数组织将允许在复制中断时继续对应用程序进行写入;但是,在复制中断期间,远程副本将不同步。如果您在复制中断时丢失了主副本,则远程站点上的副本将不是最新的,您的服务级别协议 (SLA) 将被破坏,并且如果您需要在 DR 中启动应用程序,您将面临数据丢失的情况。

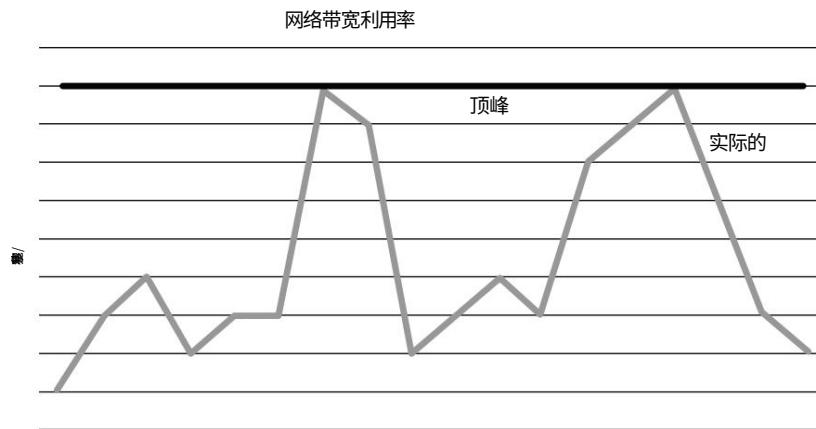


虽然很少见,但有些组织确实有某些应用程序,如果远程副本无法保持同步,它们会将其隔离并停止写入。这些往往是特殊情况的应用程序,ca cannot be kept in sync. These tend to

鉴于以上所有情况,站点之间拥有多个可靠、多样化的复制链接非常重要。所有一级数据中心都将采用双尾设计,站点之间有两条链接,每条链接采用不同的路由。

您还需要考虑复制链接的大小。如果您没有调整复制链接的大小,链路能够应对最大的流量突发,当您遇到高写入数据突发时,链路将饱和,随后应用程序的性能将下降。一些组织很乐意接受这一点,因为这意味着他们不必扩大复制链路来处理峰值流量。重要的是,在确定复制链路的大小和规格时,您要知道自己的需求。图 3.8 显示了每小时的网络带宽利用率。为了使此解决方案确保远程副本不落后于生产源卷,需要调整网络链路的大小以应对峰值流量。

图 3.8 同步远程复制带宽要求



异步复制

同步和异步复制之间的主要区别在于,对于异步复制,当本地阵列确认时,每次写入都被视为完成。

发出 ACK 之前无需等待写入提交到副本卷。

这意味着几件事。

首先,异步复制并不能保证零数据丢失。事实上,异步复制肯定会丢失数据!丢失多少数据完全取决于您的配置。例如,如果您的复制解决方案每 5 分钟复制一次,您可能会丢失略多于 5 分钟的数据。

其次,更积极的一点是,异步复制不会像同步复制那样导致性能损失。这是因为对远程阵列的写入是在稍后以惰性方式完成的。

这也意味着源阵列和目标阵列之间的距离可以比同步配置中的距离大得多。这是因为源阵列和目标阵列之间的 RTT 不再重要。理论上,您可以将源阵列和目标阵列放置在地球的两端。

也无需指定站点之间的网络连接以满足峰值需求。但是,您仍应指定网络链接,以免超出约定的 RPO。

这一切都很好,前提是你能安然入睡,因为你知道如果
您是否需要调用 DR。



调用 DRG(灾难恢复)是指使用其复制的存储启动应用程序。这通常涉及使应用程序发
生故障 its replicated storage. This usually involves failing the application—
包括服务器、网络和存储 转移到远程灾难恢复数据中心的备用服务器、网络和存储。

存储阵列倾向于采用以下两种方式之一实现异步复制,如图 3.9 所示:

- 基于快照
- 基于期刊

图 3.9 异步远程复制

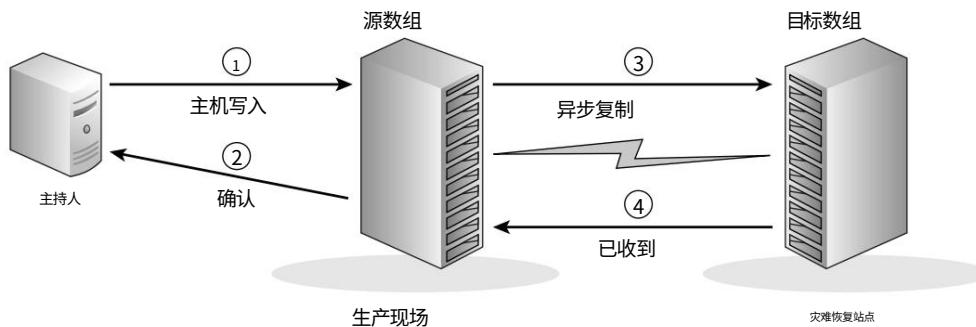


图 3.9 和图 3.7 之间要注意的区别是,主机接收到
图 3.9 中的步骤 2 之后,ACK 因此可以继续发出更多 I/O。

基于快照的异步复制

在基于快照的异步复制中,源阵列定期对源卷进行时间点快照,然后通过网络将快照数据复制到目标阵列,并在目标阵列中将其应用于副本卷。

这种类型的复制是基于计划的,这意味着您可以安排快照和复制

根据您所需的恢复点目标 (RPO) 确定复制间隔。假设您已与企业达成 SLA,以便能够在事件发生后 10 分钟内恢复数据。配置 5 分钟的复制间隔将满足该 RPO,并确保副本卷不会滞后于源卷超过 5 分钟。

好的存储阵列有一个允许您指定 RPO 的界面,它会为您配置复制规范,以确保满足您的 RPO SLA。更好的存储阵列的 RPO 驱动程度甚至更高。例如,如果您将一些卷配置为 15 分钟 RPO,将其他卷配置为 20 分钟 RPO,如果阵列遇到复制拥塞,阵列将根据哪些 RPO 最接近中断来确定复制的优先级。这接近阵列智能!

如果你的存储阵列还停留在 90 年代,不了解 RPO 和 SLA,那么你在配置复制时必须手动考虑这些事情。



请注意 10 分钟的 RPO 并不意味着您可以配置 10 分钟的复制更新间隔。这主要是因为您的快照数据不会在发送后立即到达目标阵列。数据通过网络传输需要时间,并且使用异步复制的线路通常相对便宜的低带宽线路。假设您的所有数据需要 2 分钟才能到达目标阵列,那么这可能是 12 分钟的 RPO,而不是 10 分钟的 RPO。

通常,如果您的阵列支持基于快照的异步复制,则该阵列将使用与拍摄本地快照相同的快照引擎。根据您的阵列,这可能意味着基于复制的快照会占用阵列上支持的最大快照数量。因此,如果您的阵列最多支持 1,024 个快照,并且您正在使用基于快照的异步复制复制 256 个卷,那么您可能已经占用了阵列最大 1,024 个快照中的 256 个。



当基于快照的异步复制时请注意快照扩展大小 - 即快照增长的粒度。如果快照扩展大小为 64 KB,但您在复制间隔之间更新的只是 1 MB 块,则实际上通过网络复制的是 $1,000 \times 1 \text{ MB}$ 快照扩展。没什么大不了的,对吧?现在假设几个大型阵列共享一个远程复制链接,每个阵列都使用基于快照的复制。您更新 1,000 个唯一的 4 KB 块,并假设这将通过网络复制不到 4 MB 的数据。但是,每个 4 KB 更新都在每个卷上的唯一扩展中,这意味着您最终将复制 $1,000 \times 64 \text{ KB}$,而不是 $1,000 \times 4 \text{ KB}$ 。差异非常大 - 62.5 MB,而不是 3.9 MB!

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更新同一数据块 1,000 次,则只有该数据块的最新更新将与下一组增量一起通过网络发送,而不是所有 1,000 次更新。

基于日志的异步复制

基于日志的异步复制技术将写入数据缓冲到专用日志卷,有时称为写入意图日志。它们必须大小合适(通常超大),以应对大流量突发或复制链接中断的情况。如果这些日志卷的大小太小,复制将在大突发或复制链接长时间停机期间中断。另一方面,如果它们的大小太大,99% 的时间都是在浪费空间。它可以被视为一种平衡行为或一种暗黑艺术。无论哪种方式,从设计角度来看它都相当复杂。

当写入 I/O 进入源阵列时,它们会像往常一样命中缓存,并且阵列会发出 ACK 发送至主机。写入 I/O 也带有元数据标记,以表明其目的地是复制卷,以确保数据也异步复制到本地日志卷。然后,根据标准缓存转储将数据转储到磁盘。

此时或左右,数据也会写入日志卷。然后,数据会按照阵列复制技术规格确定的时间复制到目标阵列。但一般来说,基于日志的复制不会落后于源卷太多。

好的阵列会将写入顺序元数据应用于缓存到日志卷的数据,这样当数据写入目标阵列上的目标卷时,就能保持正确的写入顺序。一旦数据在目标阵列上提交,就可以从源阵列上的日志卷中释放出来。

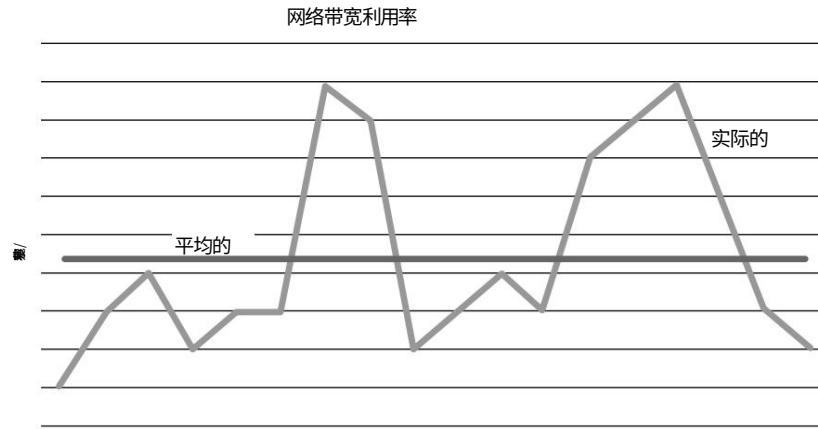
实现方式因源阵列是否将更新推送到目标阵列或目标阵列是否提取更新而异。对于您的日常工作,您不必担心这些事情,当然不必像确保您满足 SLA 和 RPO 那样担心。



尽管异步复制不需要源阵列和目标阵列之间一致的网络带宽,但请确保您的网络连接不是网络忙碌中的佼佼者,不会沦为在其他人享用完备份后剩下的残羹剩饭。如果您没有为这些链接提供足够的带宽,您将经常超出 SLA 范围,这不仅会危及您的工作,还会危及您的公司的数据。

图 3.10 显示了每小时捕获的实际带宽利用率,其中包括一条平均线。异步复制链路不需要根据峰值需求进行调整。

图 3.10 异步复制的带宽要求



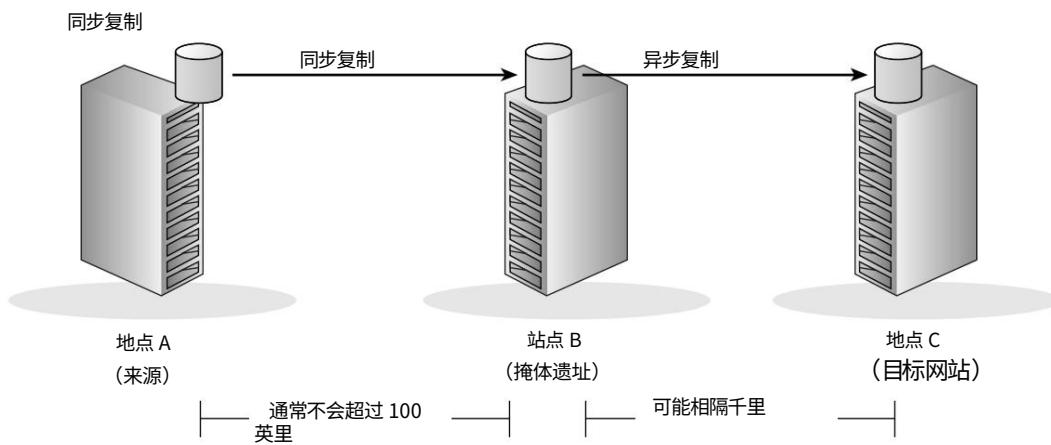
复制拓扑根据您的阵列技术,有

各种复制拓扑可供选择。这些拓扑可以包括两个以上的站点，并同时使用同步和异步复制技术。支持大多数复制拓扑的往往是高端企业级阵列。中端阵列往往只支持更基本的拓扑。

三站点级联

三站点级联复制拓扑利用中间站点（有时称为掩体站点）连接源阵列和目标阵列。图 3.11 显示了此拓扑的一个示例。

图 3.11 三站点级联拓扑



如图 3.11 所示,三站点级联同时采用了同步和异步复制。从源站点到掩体站点进行同步复制,然后从掩体站点到目标站点进行异步复制。

三站点级联拓扑的一个主要用例是实现应用程序恢复

如果您的主数据中心发生局部问题,需要您调用灾难恢复计划,则不会丢失任何数据。在这种情况下,您可以在堡垒站点启动应用程序,并且可能不会丢失任何数据。但是,如果发生重大灾难导致主数据中心和堡垒站点都无法运行,您可以在远程站点获得第三份副本。由于堡垒站点和目标站点之间的复制是异步的,因此距离可能足够大,即使影响源站点和堡垒站点的重大本地灾难也不会影响目标站点。如果真的发生了足以影响目标站点的灾难,那么很有可能您最不想考虑的事情就是恢复公司的应用程序!

三站点级联还可让您的应用程序在源站点关闭时仍受到保护。这是因为仍然有两个站点之间存在复制链接,并且这两个站点之间的复制仍可运行。

三地点级联模型的主要缺点是掩体地点的故障

也会影响目标站点。基本上,如果你失去了掩体站点,你的目标站点就会开始越来越落后于生产,你的 RPO 会变得非常危险。

该模型还适用于主数据中心的辅助阵列,而不是远程掩体站点,以防止主阵列故障,而不是整个站点故障。

三点多目标

三点多目标拓扑结构使源阵列同时复制到两个目标阵列。一个目标阵列位于相对较近的掩体站点,足够近以进行同步复制。另一个目标距离较远,并且异步复制。此配置如图 3.12 所示。

三点多目标相对于三站点级联的主要优势在于,掩体站点的故障不会影响到向目标站点的复制。从这一点来看,它更加稳健。

三点多目标拓扑的主要缺点是,如果源站点丢失,则掩体站点和目标站点之间不存在复制。它还会对主阵列施加更重的负载。

三地点三角

三地点三角拓扑类似于三点多目标,并提供所有相同的选项,另外还在堡垒和目标站点之间增加了备用复制链接。

在正常运行情况下,此附加链路不会主动发送复制流量,但如果源站点不可用,则可以启用。三地点三角拓扑如图 3.13 所示。

图 3.12 三点多目标

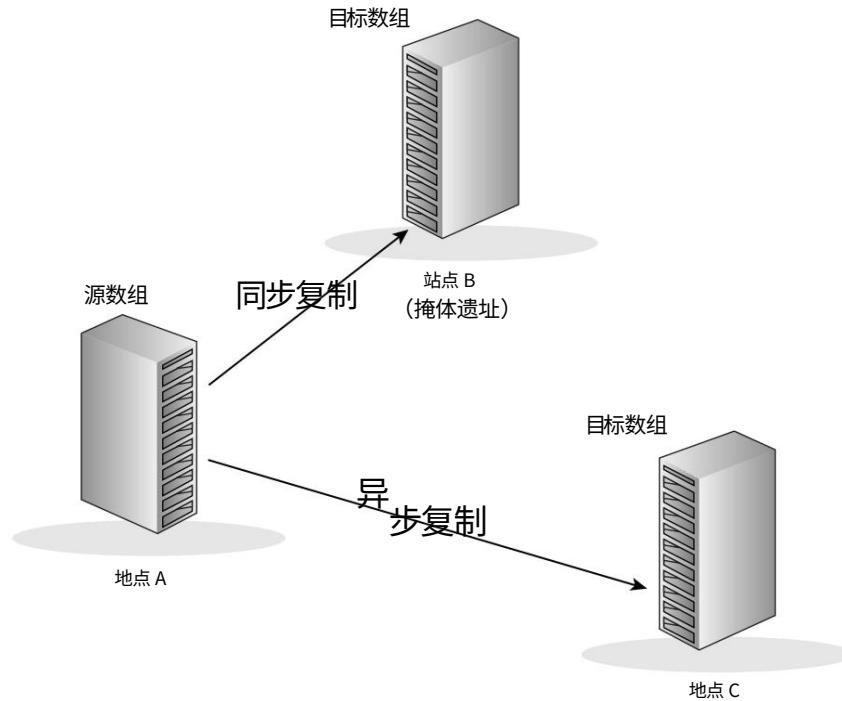
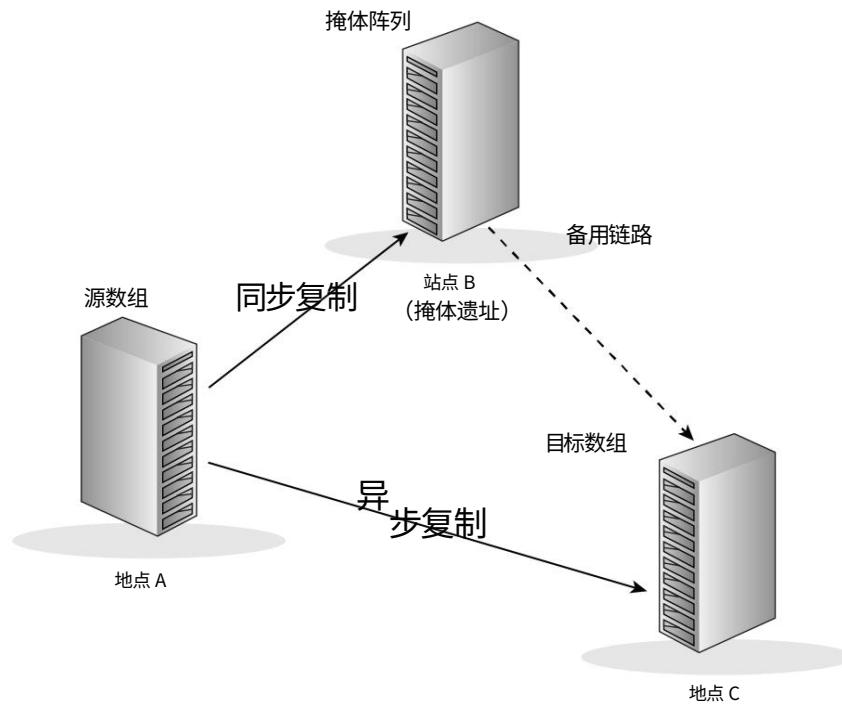


图 3.13 三站点三角复制拓扑



如果源站点丢失，则在堡垒站点和目标站点之间同步复制不需要完全同步。堡垒和目标站点中幸存的阵列能够通信并确定目标站点缺少哪些更新，从而允许从堡垒阵列到目标阵列进行增量更新。这使您能够在短时间内重新启动并运行应用程序并通过远程复制对其进行保护。



并非所有存储阵列都支持所有复制拓扑。在假设您的阵列支持特定的多站点复制拓扑之前，务必咨询阵列供应商或查看阵列文档。 array documentation before assuming that

本地快照

快照并不是什么新鲜事物，在某些情况下，其背后的技术也不是什么新鲜事物。一些快照实现方式已经过时了。我们将揭秘其中一些。

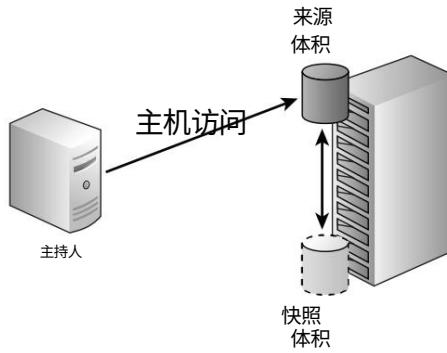
首先，让我们就术语达成一致：当提到快照时，我指的是数据的本地副本。本地是指快照与源卷位于同一阵列上。一般来说，快照是时间点（PIT）副本。

这两个事实——本地和时间点——是本地快照和远程副本之间的两个主要区别。本地快照是在本地阵列上创建和维护的生产卷的副本，而远程副本则保存在远程阵列上。

此外，本地快照是时间点的副本，而远程副本通常与主生产卷保持同步或半同步。

最后，我使用术语源卷和主卷来指代实时生产卷。我使用术语快照来指代实时生产卷的时间点副本。如图 3.14 所示。

图 3.14 简单快照示例



快照和克隆是即时创建的，并且根据您的阵列，快照和克隆可以标记为只读或读/写。所有好的阵列都支持只读和读写快照。



快照不是备份!记住这句话,将其刻进你的大脑,RAID至少使用双重奇偶校验来保护它,并将其固定到你大脑的离性能缓存中,以便随时可以快速调用。快照不是备份!为什么?如果你丢失了 RAID 集、阵列或托管主卷的站点,你也会丢失快照。

你肯定不想陷入这种困境。

基于阵列的快照

目前通常使用两种基于阵列的快照技术：

- 节省空间的快照
- 完整克隆

虽然它们可能有各种名称,但本书坚持使用这两个名称。本节涵盖快照扩展大小、空间预留、快照数量、如何将分层与快照结合使用、快照一致性以及写时复制和写时重定向快照之间的区别等主题。

节省空间的快照

节省空间的快照通常基于指针。这告诉您两件重要的事情：

空间高效意味着快照仅包含自拍摄快照以来主卷中发生变化的部分。

基于指针意味着自快照以来没有发生改变的数据

创建的内容被复制到快照中。相反,对于自快照创建以来未发生改变的任何数据,指针将指向原始卷中的数据。

让我们看一个带有一两个图的简单快捷的例子。

假设您有一个名为LegendaryVol 的主卷。LegendaryVol 有 100 个块,编号为 0-99。星期一下午 1 点,您制作了 LegendaryVol 的空间高效快照,此时, LegendaryVol 中的所有块都设置为二进制零 (0)。您的快照名为LegendaryVol.snap。创建快照时, LegendaryVol.snap 不占用任何空间。如果主机要安装此快照并读取其内容,则所有读取 I/O 都将重定向回 LegendaryVol 中包含的原始数据。LegendaryVol.snap 所消耗的只是内存中的指针,这些指针将读取 I/O 重定向回数据所在的主卷。

现在我们假设到下午 2 点, LegendaryVol 已经将 0-9 区块更新为二进制
— (1)。任何时候主机想要读取 LegendaryVol.snap 中的数据,除了读取块 0-9 之外,它都会被重定向回源卷 (LegendaryVol)。这是因为块 0-9 在主卷中已经发生变化,并且为了确保快照看起来与拍摄快照时 (下午 1 点) 的主卷完全一样,原始内容也被复制到快照中。

因此,在下午 2 点,快照仍然是主卷在下午 1 点的精确映像。所有块均为零。任务完成。此外,快照节省空间,因为所有

自创建快照以来未更改的数据仍然只是指向原始数据的指针，并且快照仅消耗保存自那时以来已更改的数据所需的空间。

完整克隆快照

完整克隆不节省空间，因此不基于指针。但它们是主卷的时间点副本。

在创建完整克隆时，将获取主卷的精确完整副本。

这意味着，如果主卷在克隆时占用了 2 TB 的存储空间，则该卷的每一个完整克隆也将占用 2 TB。这显然意味着完整克隆可能是磁盘空间的主要消耗者。

那么，为什么要使用完整克隆而不是节省空间的快照呢？有几个常见原因：

■ 读取和写入完整克隆不会对主卷产生直接影响。这是因为完整克隆完全独立于主卷，不包含任何指针。因此，您可以使用读写请求敲打克隆，这不会对主卷的性能产生影响。另一方面，快照与其主卷共享大量数据，这意味着如果您用密集的读写 I/O 敲打快照，也会影响主卷的性能。

■ 完整克隆完全独立于创建它们的主卷。

这意味着，如果主卷所在的物理驱动器发生故障，克隆将不会受到影响（假设您没有犯将主卷和完整克隆放在同一驱动器上的错误）。相比之下，对于快照，如果主卷后面的磁盘发生故障，快照也会受到影响，因为快照的大部分内容都指向主卷。

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来介绍节省空间的快照。

快照范围大小

对于节省空间的快照，大小绝对很重要。而且越大并不一定越好！

如果您的阵列的快照扩展大小（有时称为粒度）为 4 KB，则其空间效率将远高于快照粒度为 128 KB 的阵列。下面是一个简单的示例。假设自创建快照以来，您对主卷进行了 100 次更改。如果您的快照扩展大小为 4 KB，则快照最多占用 400 KB。另一方面，如果您的快照扩展大小为 128 KB，则快照占用的最大空间将跃升至 12,800 KB！

再举一个例子，如果您对具有快照的主卷进行 4 KB 更新，并且快照扩展区大小为 128 KB，则可能必须为快照分配额外的 128 KB。这是因为扩展区是快照增长的最小单位。

虽然这听起来可能不是世界末日,但请考虑一下当你有大量快照并长期保存时这会产生影响。这种空间效率低下

效率很快就会增加。这也会影响基于快照的异步复制配置中通过网络发送的数据量。因此,当谈到快照粒度或范围大小时,越小越好。

保留快照空间

在基于传统架构的阵列上,您需要猜测快照可能占用多少空间,然后提前预留这些空间!没错,第一天您就必须掏钱购买并预留空间以备快照使用,而这些空间可能永远不会被使用。这种架构更糟糕的是,在许多情况下,您必须为最坏的情况做好准备,而且往往会被预留太多空间。在现代 IT 世界中,这样的传统架构是不可原谅的。不幸的是,基于这种方法构建的传统阵列架构仍然存在,并且仍在销售!要当心。

一种更现代的方法是让快照动态地从共享池(通常与主卷不是同一个池)中为自己分配空间。此共享池将由其他卷(主卷和快照卷)使用,并且您可以设置策略来处理池中空间不足的情况。例如,您可以设置一个策略,当池达到 90% 满时删除最旧的快照,以确保精简配置的主卷不会因快照而受到影响。

快照太多

在大多数阵列中,快照被视为卷。这意味着,如果您的阵列仅支持 8,192 个卷,则主卷和快照卷的总和可以是 8,192。它通常不是指 8,192 个卷加上快照。请务必查看您的阵列文档!



Real World Scenario

如果不了解数组限制会发什么

一家公司正在使用与 Microsoft 卷影复制服务 (VSS) 集成的基于阵列的快照来改进其 Exchange 备份和恢复解决方案。他们的宏伟计划是每天为每个 Exchange 邮箱服务器拍摄一个快照,并将这些快照保留 30 天。他们有一个专用于 Exchange 的块存储阵列,支持 1,024 个卷(包括快照)。他们一开始没有考虑到的是,虽然他们没有很多 Exchange 服务器,但每个 Exchange 服务器都有多个卷。总共有 41 个卷需要快照。他们显然在第 23 或 24 天左右遇到了瓶颈,当时阵列上的主卷和快照数量达到 1,024。突然间,阵列无法再创建任何快照或卷。

为了解决这个问题,必须重新设计解决方案,以提供 21 天的快照,而不是 30 天。

我们还应该指出,虽然该公司使用快照进行 Exchange 备份,但快照被用来增强现有的备份流程。基于阵列的快照技术被用来加速和简化获取 Exchange 环境应用程序一致性副本的过程。然后,这些快照被复制到磁带上,磁带被带到异地。快照技术被用来提高备份的可靠性和性能,并提供可能更快的恢复方法。如果他们丢失了快照,他们仍然可以使用磁带进行恢复。

分层快照

大多数现代阵列还允许快照卷由其自动分层算法控制,从而使较旧的快照能够逐层向下流动,并在消耗缓慢、廉价的存储的情况下结束其使用寿命。例如,假设您的阵列已设置并运行,以便超过 10 天未访问的数据块会自动分层到较低级别的磁盘。任何 10 天或更长时间未访问的快照都将停止使用高性能磁盘。这允许您的快照在其生命周期的早期(此时更有可能使用高性能磁盘)使用,然后在准备退役时将其放在较低级别的磁盘上。

崩溃一致性和应用程序感知快照

与基于阵列的复制类似,基于阵列的快照在其原始形式下没有智能。它们不了解应用程序!不了解应用程序的快照被视为崩溃一致性。这个矛盾的术语基本上意味着快照作为应用程序恢复的工具是毫无价值的;在应用程序看来,快照中包含的数据只与服务器和应用程序实际崩溃一样好,并且可能被视为损坏。这是因为大多数应用程序和数据库将数据缓存在本地缓冲区中,并延迟将这些写入提交到磁盘。这往往是出于性能原因。这意味着在任何时间点,磁盘上的数据都不能准确地表示应用程序的状态。

幸运的是,基于阵列的快照可以实现应用程序感知!一个常见的例子是 VSS。大多数好的阵列都可以与 Microsoft VSS 集成,以便阵列和应用程序协调快照过程,确保生成的快照是应用程序一致的快照 - 应用程序可以有效且高效地从这些快照中恢复。在 Microsoft 世界中对应用程序进行快照或克隆时,这绝对是您想要的!

但是,如果您的应用程序不支持 VSS 或者您在 Linux 上运行您的应用程序,您可能能够运行脚本将您的应用程序置于热备份模式,同时可以在阵列上获取一致的快照。



Microsoft SQL Server, Microsoft SharePoint 和 Microsoft Exchange 等应用程序创建应用程序导致的基于阵列的快照 applications such as Microsoft SQL Server and Microsoft

写时复制快照

写时复制 (CoW) 快照是一种过时的技术,而且正在衰落。其缺点是,它们具有首次写入性能的开销,但优点是它们保持了主卷的连续布局。让我们快速看一下这两种方法:

- 首次写入时复制的代价导致对主卷的每次写入都变成三个 I/O:读取原始数据、将原始数据写入快照以及将新数据写入主卷。这还会增加基于闪存的卷的写入放大,从而缩短闪存介质的使用寿命。
- CoW 快照的优点在于它们保留了主卷的原始连续布局。接下来讨论重定向写入快照时将介绍这一点。

写入时重定向快照

写入时重定向 (RoW) 快照(有时称为写入时分配)的工作方式与写入时复制不同。当写入 I/O 进入受快照保护的块时,块中的原始数据不会更改。相反,它会被冻结并成为快照的一部分,更新的数据只会写入新位置,并且主卷的元数据表会更新。

举个例子:一个卷包含 100 个连续的块,0-99。您快照了该卷,然后想更新块 50。使用重定向写入,块 50 的内容不会改变,但会冻结在其原始状态并成为快照卷地址空间的一部分。原本要写入块 50 的数据被写入到其他地方(比如说块 365)。然后更新主卷的元数据,以便逻辑块 50 现在映射到磁盘上的物理块 365(元数据写入速度非常快,不会影响卷性能)。

此行为开始使主卷的布局变得碎片化,最终导致其变得非常碎片化或不连续。这种不连续的布局会严重影响基于旋转磁盘的卷的性能。但对于基于闪存驱动器的卷来说,这不是问题。

重定向快照越来越受欢迎,特别是在全灰阵列中,由 RoW 快照导致的主卷布局不连续不会引起性能问题。

基于应用程序和主机的快照

一般而言,基于应用程序和主机的快照(例如逻辑卷管理器(LVM)技术提供的快照)不会将任何工作转移至阵列。

它们还可能遭受这样的困扰:每次写入时,必须向磁盘发出两次写入

应用程序，并且通常仅限于将生成的快照呈现给运行 LVM 的同一主机。然而，它们在小型商店和开发环境中被广泛使用。

精简配置

在存储领域，精简配置 (TP) 是一项相对较新的技术，但它已迅速受到热烈欢迎和广泛采用。其迅速采用的主要原因是它有助于避免浪费空间，从而节省资金。

熟悉存储环境的人都知道，存储中存在大量的容量浪费。

在讨论细节之前，我们先就术语达成一致。本书使用了厚卷和薄卷这两个术语，但请注意，还存在其他术语，例如虚拟卷和传统卷。

厚卷

让我们先回顾一下厚卷。厚卷是最容易理解的，自零年以来就一直存在。厚卷背后的理论是，它们在第一天就保留了 100% 的配置容量。

让我们看一个简单的例子。假设您有一个全新的存储阵列，其中有 100 TB 的可用存储空间。您在该阵列上创建了一个 4 TB 的厚卷，但什么也没做。您甚至没有将其呈现给主机。您立即将阵列上的可用容量从 100 TB 减少到 96 TB。如果您打算很快使用这 4 TB，那就没问题。但通常情况并非如此。通常，这 4 TB 会长期闲置。这就是厚卷的真正弱点。大多数人会为自己获取尽可能多的存储空间，然后从不使用它。

让我们更仔细地看一下这个问题。

大多数请求存储的应用程序所有者都大大高估了他们的需求。您肯定不希望应用程序的容量太小，对吧？人们常常会盲目猜测其容量需求，然后将猜测值翻倍，以防估算错误，然后为了更加安全，再将猜测值翻倍。最初估计 250 GB，然后将其翻倍到 500 GB，以防估算错误，然后为了更加安全，再将猜测值翻倍到 1 TB。第一天，厚卷会占用阵列上 1 TB 的物理空间，尽管应用程序在第一天只需要 50 GB。使用三年后，尽管该卷占用了 1 TB 的物理容量，但应用程序仍然只占用了区区 240 GB——浪费了超过 750 GB。

在经济繁荣时期，或者如果您的公司拥有大片土地专门种植摇钱树，那么这可能不是什么问题。但是，如果您生活在现实世界中，这相当于严重浪费昂贵的公司资产，并且在许多组织中理所当然地成为死罪。

精简卷

这时精简卷就派上用场了。1 TB 厚 LUN 在创建时会为自己预留 1 TB 的物理容量,而精简卷则不会预先预留任何空间。

其工作方式与文件类似,文件的大小只有在写入时才会增大。



实际上,精简卷在创建后就会占用空间。sume space as soon as they are created.
但是,这个空间通常很小,可以忽略不计。大多数阵列会保留几兆字节的空间供数据存放,其余空间则作为缓存中的元数据指针。

让我们再次查看示例 1 TB 卷,只是这次是精简卷。这次我们的精简 1 TB 卷没有预先保留任何物理容量,第一天只消耗 50 GB 的物理容量 因为应用程序只写入了 50 GB 的数据。三年后,它只消耗了 240 GB。剩下的 760 GB 可供阵列在其他地方分配。这很简单,对吧?

精简配置范围大小

精简配置世界与节省空间的快照世界类似:大小很重要。

与节省空间的快照世界一样,越小越好。

TP 区大小是应用于精简卷的增长单位。假设您的 1 TB TP 卷的初始大小为 0 MB。如果 TP 区大小为 128 MB,则只要主机向该卷写入 1 KB,它就会占用存储阵列上的单个 128 MB TP 区。

同样,如果您的 TP 区大小为 42 MB,则 1 KB 写入操作仅消耗 42 MB 的后端空间。或者,如果您的 TP 区大小为 1 GB,则 1 KB 写入操作将消耗相对巨大的 1 GB。从容量角度来看,这应该很容易看出,区大小越小越好。

TP 范围有时被称为页面。

精简配置和性能

非常小的区段大小可能没有好处的一个地方是性能方面。您会看到,每次必须将新的区段分配给精简卷时,在阵列分配新的区段并将其添加到卷的元数据映射时,性能开销很小(非常小)。但是,这很少被视为真正的问题,较大的 TP 区段大小几乎总是表明阵列功能不够强大,无法处理和映射大量区段。

但一般来说,由于精简配置是建立在宽条带化之上的/
池化、精简配置的卷分布在后端的大多数(如果不是全部)驱动器上,使它们能够访问整个后端的所有 IOPS 和 MB/秒。这往往会有更好的性能。

精简配置卷可能导致性能降低的一个用例是旋转磁盘阵列中大量顺序读取。顺序读取要求将数据按顺序排列在旋转磁盘上,以尽量减少磁头移动。由于TP卷在写入数据时只占用后端的驱动器空间,因此这些数据往往不是来自连续区域,导致卷的地址空间在阵列的后端出现碎片。(基于主机的碎片整理工具对此无能为力。)这可能导致TP卷的性能明显低于这种特定工作负载类型的厚卷。

过度配置

精简配置自然会导致过度配置阵列。让我们看一个例子。

假设一个阵列的物理安装可用容量为100 TB。此阵列有100连接的服务器(我们使用整数来简化计算)。每台服务器分配了2 TB 的存储空间,总共分配了200 TB 的存储空间。由于阵列只有100 TB 的物理存储空间,因此该阵列的超额配置率是100%。

使用厚卷时,这种容量过度配置是不可能的,因为每个厚卷在创建时都需要保留其全部容量。一旦创建了第50个2 TB 卷,所有阵列空间都将用完,并且无法再创建其他卷。

过度提供财务利益

让我们从财务角度来看一下过度配置。如果我们假设每TB的成本为4,000美元,那么我们的100 TB 存储阵列将花费400,000美元。但是,如果我们没有过度配置此阵列,而必须购买200 TB 以满足我们在上一个示例中配置的200 TB 存储,则该阵列将花费800,000美元。因此,不难看出过度配置的经济效益。

过度配置的风险

尽管如今过度配置越来越普遍,但只有当您了解容量使用情况和趋势以及公司的采购周期时才应该这样做!

它的各个方面都应该经过彻底的测试 包括报告和趋势
在生产环境中部署之前。

主要风险是存储挤兑。如果你过度配置存储阵列,并且
如果容量耗尽,所有连接的主机都可能无法写入阵列,并且您可能会丢掉工作。因此,对超额配置
的阵列进行良好的规划和管理至关重要!

需要注意的是,过度配置是一种一次性技巧。您只能使用它一次。此后,需要进行大量的维护才能确保您不会耗尽存储空间。再次假设您有一个100 TB 的阵列,并且已配置了100 TB,但实际只使用了50 TB。您可以过度配置,比如说180 TB 的配置存储,为您提供总共180 TB 的导出存储。太好了,您刚刚变魔术般地增加了80 TB,总资本支出成本为0.00美元。管理层会喜欢你的。

但是,明年您将无法再使用同样的伎俩。事实上,明年您将面临额外的风险,如果您管理不当,您可能会耗尽存储空间并导致所有应用程序瘫痪。这种增加的风险是真实存在的,绝对不能忽视或忽略。

由于过度拨备本身存在风险,因此高级管理层必须理解这些风险,知道自己将面临什么,并正式批准使用过度拨备。

过度配置趋势

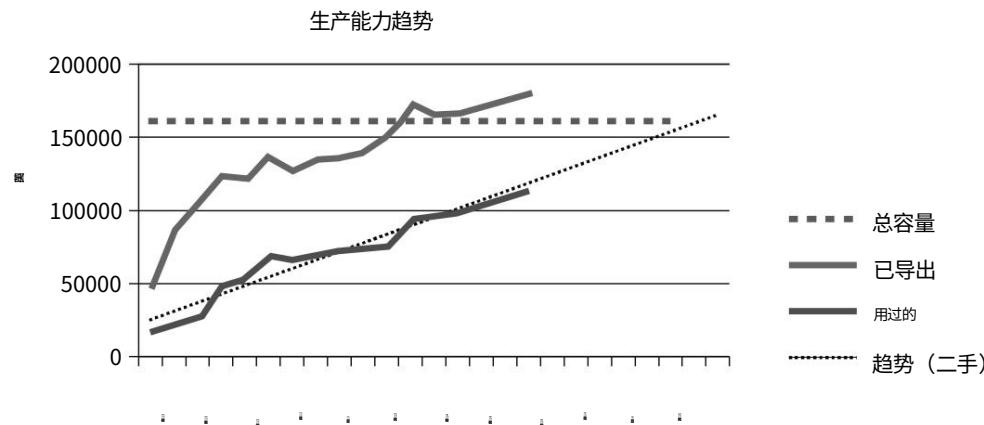
开始过度配置后,您需要密切监控和了解容量利用率趋势。以下是在过度配置环境中需要监控的关键指标:

- 装机容量
- 已配置/已分配容量
- 已使用容量

其中每项都需要至少每月进行记录,并且需要进行趋势分析。

图 3.15 显示了迄今为止 16 个月内这三个指标的值。它还提供了一条趋势线,预测了使用容量的未来增长。趋势线显示,根据目前的增长率,该阵列将在 2015 年 1 月或 2 月左右耗尽空间。

图 3.15 过度配置图表



何时购买过度配置阵列的容量

您需要与高层管理人员商定触发新容量采购的门槛,并且必须在了解公司采购流程的基础上决定这些门槛。例如,如果您的公司需要很长时间才能批准采购,而您的供应商需要很长时间才能报价和运送存储,那么您就不想把事情搞得太过分。

许多公司至少使用两个触发点进行容量购买。一旦达到第一个触发点,就会提出更多容量的采购订单。一些常见的触发点包括:

- 实际可用容量 (安装容量减去已用容量)
- 可用容量百分比 (未使用的安装容量百分比)
- 过度配置百分比

根据这些触发器,您可以实施以下触发规则:

- 一旦可用容量 (安装容量减去已用容量) 达到 10 TB 以下,您就开始购买周期。
- 一旦可用容量 (安装容量减去使用容量,以百分比表示) 的百分比达到 10% 以下,您就可以启动购买周期。
- 一旦阵列的过度配置超过 60%,您就可能启动容量购买周期。

这些规则中的每一条都在很大程度上取决于您的环境,您必须了解环境的增长特征。例如,如果阵列的增长速度为每月 15 TB,并且平均需要八周时间才能在您的环境中获得额外容量,那么等到阵列只有 10 TB 的可用容量对您来说毫无用处。在这种情况下,50 TB 或更多的可用容量值可能是更好的值。

空间回收

空间回收就是保持精简卷的精简。精简配置卷允许您避免预先分配容量,而仅按需分配。但是,如果您的精简配置阵列无法识别主机何时删除数据,您的精简卷将很快变得臃肿。这就是空间回收发挥作用的地方。

零空间回收

从高层次上讲,零空间回收是识别精简卷中已删除的空间并将该空间释放回空闲池以供其他卷使用的行为。

然而,秘诀在于阵列能够知道主机何时不再使用空间。

老式太空回收方法

传统上,当操作系统/虚拟机管理程序/文件系统删除数据时,它实际上并没有删除数据。相反,它只是将删除的数据标记为不再需要。这对阵列来说没什么用,因为它们不知道这些数据已被删除,所以它们无法从 TP 卷中释放空间。在这些情况下,需要脚本或工具将二进制零物理写入已删除的数据区域,以便阵列知道有可以释放的容量。



零空间回收基于写进制零的原理。要回收数组中未使用的空间，必须将该空间清零。For a zeroed-out delete region, unused space, that space must be zeroed out—the act of writing

主机集成空间回收

Red Hat Enterprise Linux 6 (RHEL)、Windows Server 2012 和 vSphere 5 等现代操作系统都了解精简配置以及保持精简卷精简的重要性，因此它们被设计为与基于阵列的零空间回收技术顺利协作。

为了保持精简卷的精简，这些操作系统实现了基于 T10 标准的UNMAP命令，该命令专门用于通知存储阵列可以回收的区域。 UNMAP命令通知阵列某些逻辑块地址 (LBA) 范围不再使用，然后阵列可以从 TP 卷中释放它们并将其分配回空闲池。不再需要脚本和工具来清零未使用的空间；操作系统和文件系统现在会为您处理此问题。

内联或后处理空间回收

有些阵列足够聪明，能够识别 y 上的传入零，并在零流进入阵列时动态释放 TP 范围。其他阵列没有那么聪明，需要管理员手动运行后处理空间回收作业，以释放主机上已清零的空间。

两种方法都可以，但是内联方法显然更简单，总体来说更好，只要它能够以不会影响数组性能的方式实现。

过度配置需要空间回收

如果您的阵列支持精简配置但不支持空间回收，那么您将走上一条死胡同。是的，您将避免厚卷的前期容量浪费。

但是，由于数据被删除但阵列不会回收，因此您的精简卷很快就会出现膨胀。如果您的阵列支持精简配置，并且您计划过度配置，请确保您的阵列也支持空间回收！

需要注意的空间回收问题

请注意两个方面的潜在性能影响：

- 具有高度顺序读取模式的应用程序
- 每次分配新页面或范围时的性能开销

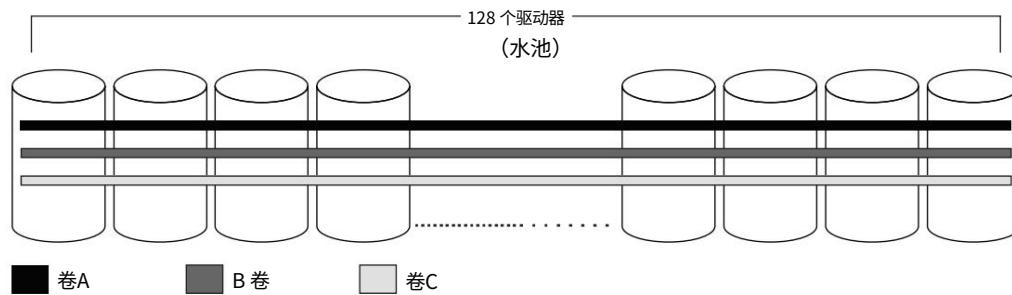
如果配置过多，则需要密切关注阵列的容量使用情况和趋势。如果疏忽大意，可能会耗尽空间并陷入困境。

睁大眼睛考虑过度配置。过度配置不会简化容量管理或存储管理。任何持相反观点的人可能在现实世界中没有太多实际经验。

池化和宽条带化

池化和宽条带化经常被同时使用。这两种技术都是指将大量驱动器池化在一起，并将 LUN 和卷分在池中的所有驱动器上。图 3.16 显示了池中所有 128 个驱动器上的三个宽条带化卷。

图 3.16 宽条带卷布局



应该说,没有实现宽条带化的阵列应该大多数部分被视为恐龙。所有好的阵列都支持池化和宽条带化。

池化和性能

池化能够提高存储阵列上卷的性能。这是因为所有卷都可能访问后端所有驱动器的所有 IOPS 和 MB/秒。

即使池化不能提高特定卷的性能,它也会减少后端的热点和冷点,从而确保您从阵列的后端获得最大的性能。使用池化,您不应该看到一些驱动器处于空闲状态,而其他驱动器则已达到极限。

池化和简化管理

除了最大化和平衡性能之外,宽条带化的出现还大大简化了存储设计和管理。在我们有池化和宽条带化之前,存储管理员和架构师会花费数小时来确保两个繁忙的卷不会使用阵列上的相同四个或八个磁盘,因为这会产生热点和性能瓶颈。Excel 电子表格被推到了极限,以细致地跟踪和映射卷在后端的布局方式。值得庆幸的是,这现在已成为过去!

如今,您可以部署一个阵列,创建几个池,然后开始创建卷。您可能仍希望将数据和日志卷放在不同的池中,但除此之外,您只需在池中创建卷,然后让阵列平衡它们。后端热点和基于主轴的瓶颈现在极为罕见。



确保阵列支持在后端添加额外驱动器时重新平衡。您最不希望看到的是后端平衡得很好!从容量和性能的角度来看,所有驱动器的使用率都为 80%。但安装 128 个新驱动器后,发现这两个方面的使用率都为 0%。这将导致后端严重不平衡,并且需要大量手动工作来重新平衡。所有好的阵列都支持在配置中允许新驱动器时重新平衡。

未来在泳池

所有供应商也都倾向于在池和宽条带化方面进行创新。例如,精简配置、重复数据删除和空间回收等功能都是在池之上构建的,无法与传统的 RAID 组很好地配合使用。

泳池和吵闹的邻居

某些组织中的某些业务关键型应用程序需要隔离,而根据设计,池化与隔离完全相反。

第 1 层业务线应用程序所有者希望感到安全,并且他们是世界的中心 在许多情况下,就业务而言,这些应用程序就是世界的中心。因此,将他们心爱的应用程序与其他所有不值得的 3 层和 4 层应用程序放在同一套主轴上,不会给他们带来温暖。在这些情况下,许多组织选择使用专用驱动器和其他资源创建隔离池,甚至将这些应用程序部署在专用阵列和服务器上。

虽然池和宽条带化可能并不适合每种应用程序,但它们确实适用于绝大多数应用程序。根据我在各个行业领域的多年经验,粗略估计池至少在 80% 到 90% 的时间里是合适的。如今,您很少甚至根本不需要针对传统 RAID 组部署卷。但无论您做什么,都要了解您的需求!

压缩

存储行业中的压缩是指通过查找可以缩小的重复模式来减小数据集大小的过程。可以使用各种复杂的算法来压缩数据,但具体操作往往侧重于删除重复字符集和其他空白。

主存储压缩从未真正流行起来。是的,我们时常看到这种做法,但主要参与者实施起来却进展缓慢。



主存储是用于存储正在使用的数据的存储,即使是不经常访问且位于较低层的数据仍被视为主存储。非主存储的示例包括存档和备份存储。Storing on lower tiers is still considered

主存储压缩尚未普及,主要是因为它会影响性能。没有人愿意等待主存储解压数据。如果从存档中调出一封三年前的电子邮件需要几秒钟以上,人们就会抱怨,更不用说等待对卷的长时间查询了,而存储阵列解压查询的数据时还需要额外的等待时间。

话虽如此,压缩在存储领域仍有一席之地,尤其是在备份和存档用例中,以及在基于 SSD 的存储阵列中,这有点有趣。压缩有两种常见方法:

■内联

■后期处理

在继续考虑压缩性能之前,让我们先来看一下两者影响和压缩的未来。

在线压缩

内联压缩在数据到达后端驱动器之前,先在缓存内存中对其进行压缩。这显然会占用大量缓存,而且如果阵列正在经历大量 I/O,则压缩数据的过程很容易导致响应时间变慢。

不过,从积极的一面来看,内联压缩可以大幅减少后端的写入 I/O 量。这显然具有额外的好处,即后端磁盘空间使用量减少,但这也意味着内部带宽消耗量减少。后者并不总是被考虑在内,但它可能很重要。

后处理压缩

后处理压缩将首先将未压缩的数据放在后端驱动器上,然后在稍后运行后台任务来压缩数据。

后处理数据可以消除潜在的性能影响,但无助于增加阵列的可用内部带宽。顺便说一句,这可能是压缩技术的一大优势。毕竟,内部带宽通常比磁盘容量更稀缺,而且总是更昂贵。

无论压缩是在线完成还是在后期处理中完成,解压缩显然也必须在线并按需完成。

性能影响

数据压缩导致的一个主要性能问题可能是解压数据所需的时间。每当主机向阵列发出读取请求并且必须从后端获取数据时,这是一个缓慢的操作。如果再加上解压数据的要求,您可能会看到一个明显更长的操作。

这绝不是理想的情况。为了缓解这种情况,许多存储阵列使用 Lempel-Ziv-Oberhumer (LZO) 压缩算法,该算法虽然不能产生世界上最好的压缩率,但却简单且快速。

压缩在主存储中的未来

由于现代 CPU 具有运行周期,并且经常需要压缩负载,因此主存储的压缩变得越来越可行。此外,使用 ash 作为存储介质也很有帮助,因为由于 ash 的读取速度更快,解压缩数据时产生的延迟更低,并且可用容量的增加可以使 ash 成为更具成本效益的选择。

一些存储系统正在实施内联和后处理技术的组合,当阵列可以承受时执行内联压缩,但是当 I/O 级别增加并且情况变得艰难时,它们会回退到后处理数据。

在 NAS 设备中,您应确保压缩是在块级别而不是文件级别进行的。如果您只需要读取一小部分,您最不希望发生的事情就是必须解压整个文件。事实上,在某些情况下,例如带有旋转磁盘的 NAS 阵列,如果使用良好的压缩技术,您可能会获得更高的性能。

最后两条建议是,一如既往地,先试后买,先测试后部署。在某些用例中,压缩会起作用,而在其他用例中则不会。

当心。

重复数据删除

重复数据删除与压缩的不同之处在于,重复数据删除算法会在数据流中寻找之前见过的模式 也许是很久以前见过的模式。例如,在本书的手稿中,单词“存储”已被使用了数百次。一个简单的重复数据删除算法可以取单词“存储”的单个实例并删除所有其他实例,用指向原始实例的指针替换所有其他实例。如果单词“存储”占用 4 KB,而指针仅占用 1 KB,我们可以看到节省的资金来自何处。显然,这是一个过于简单的例子。

而压缩会消除彼此靠近的小的、重复的模式 (例如重复数据删除 (Deduplication) 可以消除更大范围的数据中的较大重复数据 (例如,相同的 32-128 KB 块),而重复数据删除可以消除更大范围的数据中的较大重复数据。重复数据删除的历史与压缩相似,它被广泛用于存档和备份目的,但在主存储中的采用率并不高。同样,这主要是由于潜在的性能损失。

与压缩一样,重复数据删除应始终在块级别进行,而绝不应在文件级别进行。事实上,文件级别重复数据删除可能更准确地称为单实例。显然,块阵列只能进行基于块的重复数据删除,但您的 NAS 供应商可能会试图通过将文件级别单实例作为重复数据删除出售给您来蒙骗您。不要让他们这样对您!重复数据删除应始终在块级别进行,并且最好使用较小的块大小。

与压缩类似,它通常是一项内联或后处理工作。

内联重复数据删除

在内联重复数据删除中,数据在前端进行重复数据删除,然后才在后端提交到磁盘。

内联重复数据删除需要先进行哈希运算,然后进行查找。强哈希运算消耗 CPU,但命中/未命中精度较高;而弱哈希运算消耗的 CPU 较少,但需要进行更多逐位查找(这在旋转磁盘上很慢,但在 SSD 上可行)。

内联重复数据删除不需要扩大着陆区域,但可能会影响前端性能。

后处理重复数据删除

数据以膨胀形式写入磁盘,然后检查是否已存在于其他地方。如果确定数据模式已存在于阵列的其他地方,则可以丢弃该数据模式的实例并用指针替换。

后处理方法的一个缺点是,您需要在后端有足够的容量来将数据以未去重的形式存储。不过,这个空间只是暂时需要的,直到数据被去重。这会导致后端过大。

后处理的好处是不会影响前端的性能。

需要注意什么

在 NAS 阵列上,您需要注意基于文件的单实例化被误认为是重复数据删除。如前所述,这两者并不相同。使用基于文件的单实例化,您需要两个文件完全匹配才能进行重复数据删除。两个 1 MB 大小的 Word 文档如果整个文件只有一个字符不同,则无法进行重复数据删除。相比之下,块级别将对整个文件进行重复数据删除,但包含单个字符差异的几千字节除外。您总是需要块重复数据删除,块大小越小越好!

重复数据删除的范围也很重要。理想情况下,您需要一个全局重复数据删除的阵列。这意味着在检查重复数据模式时,阵列将检查整个阵列中的重复模式。与将搜索限制在单个卷相比,检查整个阵列可以更好地找到重复项。

自动分层

子 LUN 自动分层现在是所有优质存储阵列的支柱。自动分层背后的理论是根据数据的使用情况将数据放置在适当的存储层上。

这通常意味着经常访问的数据(热数据)驻留在诸如 ASH 之类的快速介质上,而很少访问的数据(冷数据)驻留在较慢的介质上,例如大型 7.2K 近线串行连接 SCSI (NL-SAS) 驱动器上。

子LUN

说到自动分层,子 LUN 是关键所在。早期的自动分层实现适用于整个卷。如果某个卷被频繁访问,则整个卷将提升到更高的存储层。如果某个卷不经常被访问,则

整个卷被降级到较低的存储层。问题是卷的某些部分可能非常热,而其余部分可能很冷。这可能导致这样的情况:当只有 10 MB 的卷被频繁访问,而其余 499.99 GB 是冷的时,大卷 (例如 500 GB) 被移入灰烬层。这对您昂贵的灰烬层来说不是很好的用途。

进入子 LUN 技术。子 LUN 技术将卷划分为更小的范围,并监控和移动范围而不是整个卷。假设我们之前的 500 GB 示例卷,如果范围大小足够小,则只有卷中的热 10 MB 会移动到灰层,其余冷数据甚至可以过滤到较低的层。这是一个更好的解决方案!

目标是,例如,如果你的数据中只有 20% 是活跃的,剩余的 80% 相对不活跃,您可以将这 20% 放在高性能驱动器上,并将不活动的数据集转储到更便宜、更慢的驱动器上。

子 LUN 扩展大小

就范围大小而言 (无论是快照范围、精简配置范围还是现在的子 LUN 范围),似乎总是越小越好。

子 LUN 扩展大小是为了监控活动和在层之间迁移而将卷拆分成的粒度。7.6 MB 的子 LUN 扩展大小意味着卷内可在层之间移动的最小数据量为 7.6 MB。

一般来说,高端企业级阵列的区大小小于中端阵列,但这并不总是正确的。但是,您应该在购买阵列之前找出阵列使用的区大小。通常可以在产品文档中或通过快速搜索互联网轻松找到此信息。一些中端阵列的子 LUN 区大小为 1 GB,这可能会大量浪费优质的基于灰分的资源;如果只有 50 MB 的区实际上处于繁忙状态,那么没有一个头脑正常的人愿意将 1 GB 的数据移动到灰分层中。

评估你的层级

大多数阵列和大多数分层存储设计都基于三层。这些层通常如下:

第 1 层:闪存/SSD

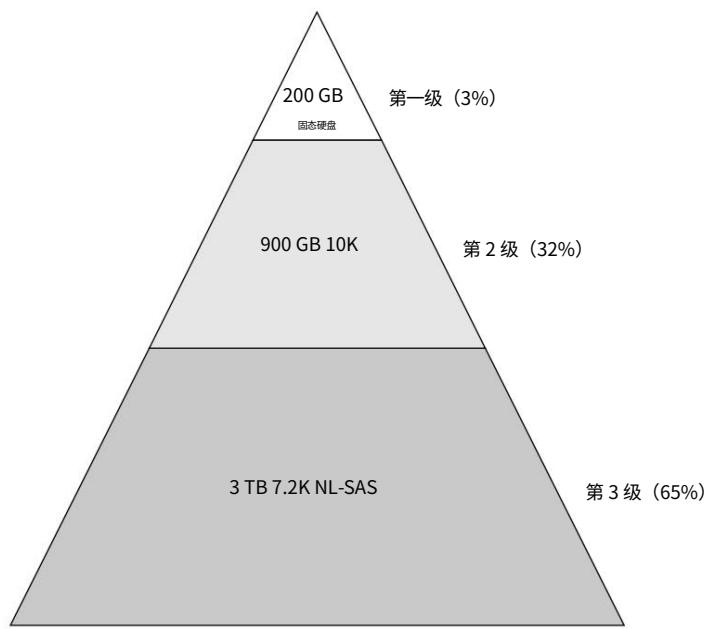
第 2 层:SAS 10K 或 15K

第 3 层:NL-SAS 7.2K 或 5.4K

每个层级的购买量很大程度上取决于你的工作量,以及你的供应商或渠道合作伙伴应该有可用的工具来帮助您做出决定。

一般来说,三层的比例可以用金字塔图来表示,如下如图3.17所示。

图 3.17 层级数量金字塔图



在图 3.17 中,百分比值表示容量百分比,而不是驱动器百分比。虽然不建议在您的环境中使用图表中的百分比,但它们是针对混合工作负载阵列量身定制的,在设计时,工作负载参数并不为人所知,因此这可能是调整您自己的配置的一个很好的粗略起点。

监测期

为了决定将哪些数据区在可用层上上下移动,阵列需要监控后端的 I/O 活动。在正确的时间监控后端性能非常重要。例如,如果您的组织在夜间运行备份,您可能希望将备份窗口从监控计划中排除,以便备份工作负载不会扭曲统计数据并影响决策过程。100 次中有 99 次,您不会希望备份卷占用更高级别的存储空间。

一种常见的方法是在工作日的核心工作时间进行监控,但这可能不适合您的环境。

调度分层操作

除了决定何时监控系统之外,您还需要确定允许阵列在层之间移动范围的数据移动窗口。

再次强调,这将取决于您的业务需求,并且可能会根据您的阵列技术允许您移动数据的频率而受到限制。但要记住的一点是,在后端移动数据会使用后端资源,而且在许多情况下还会使用缓存。

大多数公司决定在工作时间以外移动数据。通常是在凌晨 1 点到 4 点之间,但在某些系统上,连续移动数据是一个更好的主意。

除外责任及政策

良好的存储阵列为您提供从自动分层操作中包括和排除卷的方法。例如,您可能确定某些卷需要始终 100% 驻留在中间层。

您可能还希望通过自动分层算法控制卷

但您想要限制它们消耗的一级空间量,或者您可能想要确保它们永远不会被降级到最低级。

您的阵列应该允许您创建分层策略,以实现对
如果需要,阵列的自动分层行为。

使用小型阵列进行自动分层

自动分层解决方案往往更适合较大的配置,而不太适合较小的阵列。主要原因之一是每个层都需要足够的驱动器来提供足够的性能。如果小型阵列还具有较大的子 LUN 扩展大小,这将增加效率低下。此外,许可成本往往使自动分层解决方案在较小的配置中不具有商业可行性。从成本、平衡和性能的角度来看,使用自动分层解决方案的情况往往在较大的阵列中更有利。

自动分层和远程复制

使用某些自动分层解决方案很容易导致卷在源阵列中以最佳方式分布在层上,但在目标阵列中却没有。这种情况可能由于各种原因而发生,但主要原因是复制技术仅将写入数据从源复制到目标阵列。对源阵列的读取请求严重影响源阵列上卷的布局。但是,这些读取的影响不会反映在目标阵列上,因此目标阵列上的副本卷很可能占据目标阵列上较低层的最大百分比。

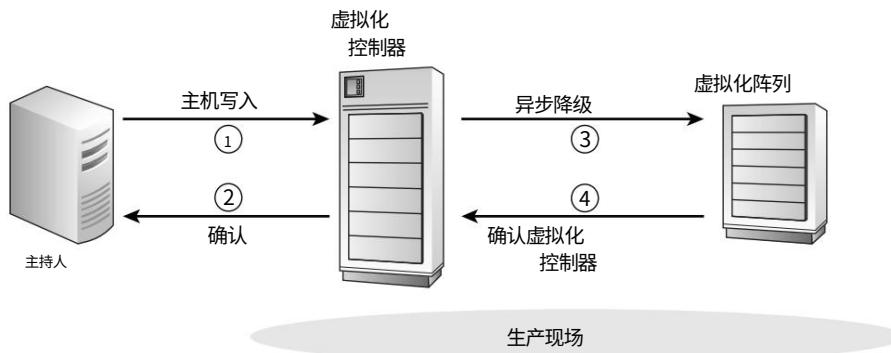
显然,如果您需要调用 DR,您可能会看到性能因此下降。

存储虚拟化

本章中的存储虚拟化是指基于控制器的虚拟化。

基于控制器的虚拟化是将一个存储阵列虚拟化在另一个存储阵列后面,如图 3.18 所示。在本节后面的图 3.19 中,您可以看到写入 I/O 在到达虚拟化控制器中的缓存时可以得到确认。一旦数据进入虚拟化控制器的缓存中,就可以以惰性方式将其转储到虚拟化阵列。

图 3.18 基于控制器的虚拟化



在存储虚拟化设置中,一个阵列充当主阵列,另一个阵列充当从阵列。我们将主设备称为虚拟化控制器,将从设备称为虚拟化阵列。

虚拟化控制器是所有智能和功能所在。虚拟化阵列仅提供受 RAID 保护的容量。虚拟化阵列中本身可用的其他特性和功能均未使用。

典型的存储虚拟化配置

配置典型的存储虚拟化需要几个步骤：

1. 配置正在虚拟化的阵列
2. 配置虚拟化控制器
3. 连接虚拟化控制器和虚拟化阵列

让我们更仔细地看一下每个步骤,然后您将看到它们如何组合在一起的示例。

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,您可以创建受 RAID 保护的常规卷,并将它们作为 SCSI LUN 呈现给虚拟化控制器。应使用 LUN 掩码,以便只有虚拟化控制器的 WWPN 才能访问 LUN。这些 LUN 不需要任何特殊配置参数并使用常规缓存设置。重要的是,您要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。例如,如果虚拟化控制器模仿 Windows 主机,则需要确保连接到虚拟化控制器的 LUN 和端口已进行相应配置。



至关重要的是,没有其他主机访问呈现给虚拟化控制器的 LUN。如果发生这种情况,这些 LUN 上的数据将被破坏。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,就不需要任何主机连接到虚拟化阵列。

配置虚拟化控制器

配置阵列后,您需要配置虚拟化控制器。在这里,您需要将两个或多个前端端口(是的,前端端口)配置为虚拟化模式。这是一种启动器模式,允许端口连接到正在虚拟化的阵列的前端端口并发现和使用其LUN。为简单起见,连接到正在虚拟化的阵列上的端口将模拟标准的Windows或Linux主机,这样正在虚拟化的阵列就不需要配置任何特殊的主机模式。

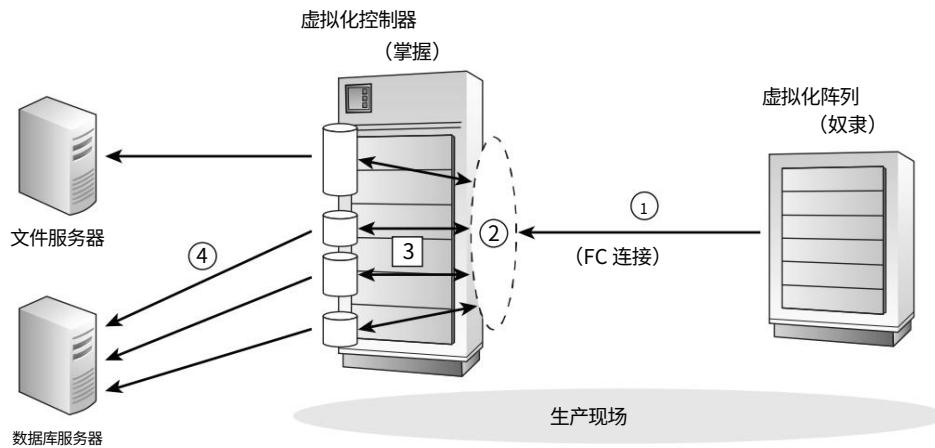
连接虚拟化控制器和虚拟化阵列

连接通常为FC,可直接连接或通过SAN连接。由于这些连接的重要性,有些人选择直接连接,以便尽可能保持两个阵列之间的路径简单干净。

一旦虚拟化控制器发现并认领了虚拟化阵列提供的LUN,它就可以像使用本地安装的驱动器的容量一样使用发现的容量。一个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。

图3.19显示了一个虚拟化阵列,它向虚拟化控制器呈现了五个LUN。虚拟化控制器登录到虚拟化阵列,发现并声明这些LUN,并将它们组成一个池。然后,该池用于为四个新创建的卷提供容量,这些卷作为SCSI LUN从虚拟化控制器的前端呈现给两个主机。

图3.19 存储虚拟化步骤



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。
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。

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称为外部存储或虚拟化存储。

综合起来

要理解配置存储虚拟化的所有阶段,最好按顺序查看步骤的示例。练习 3.4 对此进行了探讨。

练习 3.4

配置存储虚拟化

以下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会有所不同,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。

在正在虚拟化的阵列上,执行以下任务:

1. 将后端存储划分为受 RAID 保护的卷。通常,大卷
例如 2 TB。
2. 在前端端口上显示这些 LUN,并将它们 LUN 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设置。

在虚拟化控制器上,执行以下任务:

5. 将多个前端端口配置为外部端口(虚拟化模式)。为了实现冗余,至少应配置两个。

现在,虚拟化控制器上已执行了基本任务,并且阵列已虚拟化,需要连接阵列。这些连接可以是直接连接,也可以是 SAN 连接。阵列连接并可以相互看到后,请在虚拟化控制器上执行以下步骤:

6. 使用虚拟化控制器的 GUI 或命令行界面(CLI),发现
虚拟化阵列上显示的 LUN。

7. 将新发现的 LUN 配置到池中。

在虚拟化控制器上发现并导入 LUN 可能需要一段时间,创建池也需要一段时间。创建池后,通常可以将其用作使用内部磁盘的非虚拟化池。从此时起,可以创建卷并将其绑定到新池。这些卷可以映射到前端的主机,也可以用于创建池。

延长外部阵列的使用寿命

虚拟化阵列的潜在原因之一是延长其使用寿命。我们不必因为旧阵列过时且不支持我们所需的最新高级功能而将其关闭，而是可以将其插入我们崭新的阵列并对其进行大脑移植。

这个想法在原则上是可行的，但现实往往要复杂得多。

出于这个原因，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，那么让它在下一个工作日 (NBD) 响应并不是最好的主意。

■如果您从第三方供应商虚拟化阵列，则多供应商支持可能会很复杂

第三方供应商。当您虚拟化的阵列较旧时，这变得更具挑战性，因为供应商通常不希望您长期维护旧套件，并且供应商通常会停止发布固件更新。

经验表明，虽然虚拟化阵列可以延长其寿命，但实际情况往往有些混乱。

向虚拟化阵列添加功能

客户通常会购买新的 1 级企业级阵列和新的 2/3 级中端阵列，并对企业级阵列后面的中端阵列进行虚拟化。

这种配置允许将第 1 层阵列的高级功能和智能扩展到虚拟化第 2 层阵列提供的容量。由于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，因此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、虚拟机管理程序卸载……等等。

存储虚拟化和自动分层

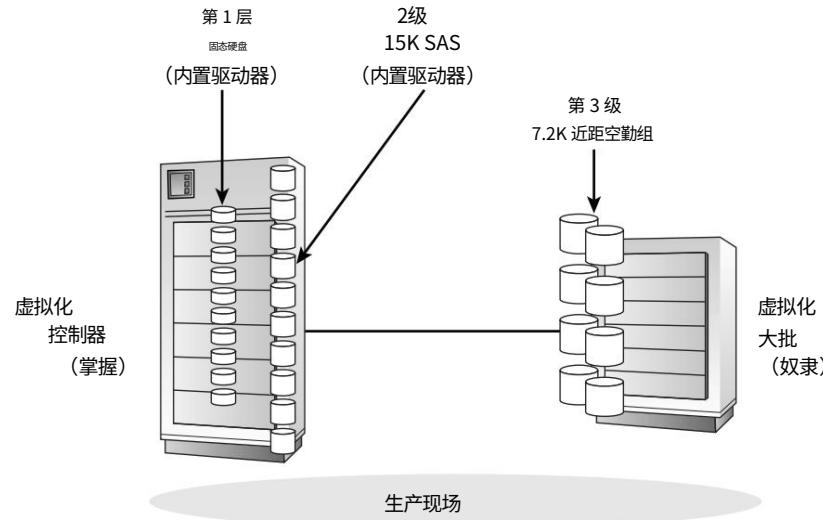
所有支持基于控制器的虚拟化的优质存储阵列都将允许虚拟化阵列中的容量成为可由其自动分层算法使用的存储层。

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最低层的存储具有财务和技术上的合理性。

因为将磁盘放入第 1 层阵列的成本令人难以置信 即使是 4 TB NL-SAS 等低层磁盘 将第 1 层阵列的内部驱动器插槽仅用于高性能驱动器具有可靠的技术优势。除了加载内部驱动器插槽的成本外，它们还比连接到虚拟化阵列中的容量提供更低的延迟。因此，从财务和技术角度来看，保留用于高性能驱动器的内部插槽都是有利的。

这很好地导致了感觉自然的多层配置，如图 3.20 所示。

图 3.2.0 存储虚拟化和自动分层



在内部灰烬存储中拥有卷的热范围没有问题,温暖的
内部高性能 SAS 上的最大扩展区,而外部 7.2 K 或 5.4 K NL-SAS 上的最大扩展区和最
不常访问的扩展区。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价格不会太高
从而破坏你的商业案例。

虚拟化陷阱

无法避免:存储虚拟化会为您的配置增加一层复杂性。这不是您应该逃避的事情,但绝对是您
在深入研究之前应该意识到的事情。



Real World Scenario

存储虚拟化可能带来的并发症

一家公司一直乐于使用存储虚拟化,直到一位经验丰富的存储管理员犯了一个简单的错误,导致整个系统离线。存储管理员删除了旧的、未使用的外部卷 - 这些卷是通过虚拟化控制器从虚拟化阵列映射的。但是,虚拟化控制器为卷提供了十六进制数,而虚拟化阵列为卷提供了十进制数。长话短说,管理员混淆了十六进制数
和十进制数,删除了错误的卷。这导致许多系统丢失了卷,许多人为此辛苦工作了一整夜。如果没有使用存储
虚拟化,这种十六进制和十进制编号方案的复杂性就不会存在于环境中。

根据阵列实现存储虚拟化的方式,有时还取决于您如何配置它,您可能会将自己的资产投入一条难以退出的单行道。如果您日后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解开的设计中。

如果您的阵列设计和大小不正确,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后端的容量(包括任何虚拟化阵列中的容量),您可能会面临一系列性能问题,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性能(通常是驱动器不够)

为了能够足够快地从缓存中退出,您可以使虚拟化头的缓存成为一个阻塞点。

说到性能,目前最常见的存储虚拟化实现方式是使用虚拟化阵列实现廉价且深度的3级存储。很少看到客户使用高性能驱动器加载虚拟化阵列。

最后,如果你不清楚自己要做什么,那么成本可能是一个问题。
虚拟化许可证很少是免费的。

硬件卸载

硬件卸载,有时也称为硬件加速,是开放系统存储领域中相对较新的趋势。硬件卸载是在专用硬件而非软件中执行功能的行为。就存储而言,这通常转化为操作系统和虚拟机管理程序卸载与存储相关的功能,例如大型复制作业和归零到存储阵列。

在对存储阵列实施硬件卸载时,需要考虑几个目标。这些目标通常如下:

- 提高性能
- 减少主机CPU的负载
- 减少网络负载

虽然现在大多数服务器都有大量的CPU周期需要消耗,但通过将这些任务转移到存储阵列上,许多与存储相关的工作仍然可以更快、更有效地执行。

让我们看一下现实世界中几个比较流行的例子。

VMware VAAI

VMware可能是开放系统领域中第一个推动硬件卸载概念的供应商。VMware的vStorage API for Array Integration(VAAI)是一套技术,旨在将与存储相关的操作卸载到VAAI感知的存储阵列。块存储和NAS存储都有VAAI卸载,称为块原语

和NAS原语。

块存储原语基于T10 SCSI标准,并且原生支持
任何支持这些T10标准的存储阵列都可以,但一定要检查

在做出任何假设之前,请先查看 VMware 硬件兼容性列表 (HCL)。另一方面,NAS 的 VAAI 需要 NAS 供应商提供的插件。



Despite the name, vStorage API for Array Integration—VAAI primitives are not real APIs. Instead, the Sphere hypervisor is simply written to

让我们看一下每个可用的 VAAI 负载。

自动测试系统

如果您对虚拟机文件系统 (VMFS) 有所了解,您就会知道某些元数据更新需要锁来确保更新的元数据和整个 VMFS 卷的完整性。

在过去,唯一可用的锁定机制是 SCSI 预留。

SCSI 预留的问题在于它会锁定整个 LUN,这意味着每次更新元数据时都必须锁定整个 LUN。当 LUN 被锁定时,除了发出预留命令的主机之外,其他任何主机都无法更新它。如果我们谈论的是小型、使用频率较低的 VMFS 卷,那还不算什么。但是,如果是多个主机访问的大型、使用频率较高的 VMFS 卷,这可能会成为问题。这时,原子测试和设置 (ATS) 就可以派上用场了!

ATS 也称为硬件辅助锁定,它具有两大特点:

基于范围的锁定基于范围的锁定意味着在更新 VMFS 卷中的元数据时,您不再需要使用 SCSI 保留锁定整个 LUN。您只需锁定包含正在更新的元数据的范围。

更高效的锁定程序基于范围的锁的接合和释放动作被转移到存储阵列上,并且锁定机制需要更少的步骤,从而更快、更高效。

这两者都显著增加了 VMFS 的可扩展性,并且是主要因素
部署更大 VMFS 数据存储的能力。

自 VAAI 首次发布以来,ATS 现已正式成为 T10 SCSI 标准
使用 SCSI 操作码 0x89 和COMPARE_AND_WRITE命令。这可确保所有支持存储阵列均能实现基于标准的实施。

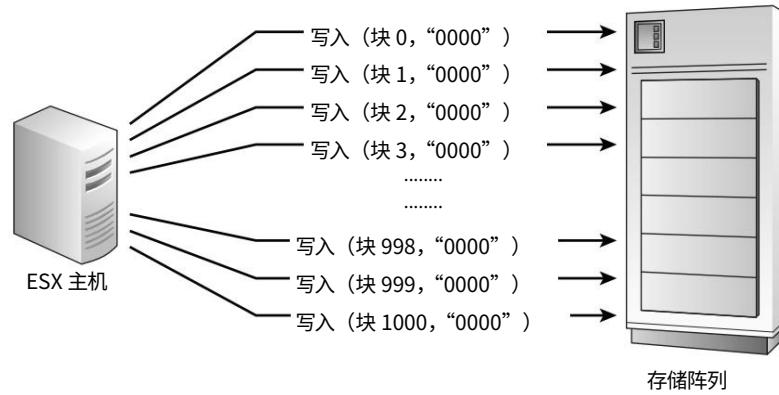
硬件加速归零

VMware 环境中的某些情况需要将整个卷归零。
例如,EZT 卷是通过将零写入卷中的每个扇区来清除卷。这具有积极的安全意义,因为它确保卷上一次使用的数据不会被无意中看到。它还具有潜在的积极性能意义,因为它不需要 ESX 必须将

第一次写入时会阻塞 - 尽管不要指望性能改进会让你惊叹。

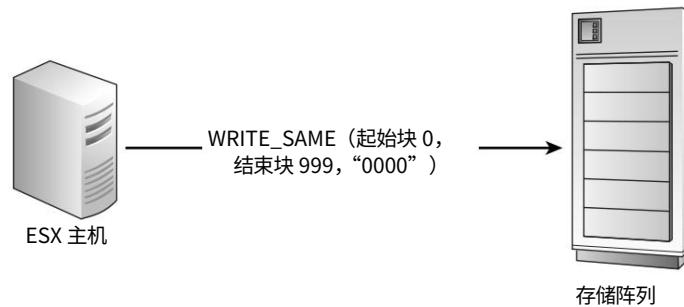
让我们看一个硬件加速归零的简单示例。假设一个卷有 1,000 个块，您需要将整个卷归零。如果没有硬件加速归零，ESX 将不得不发出图 3.21 中列出的一组命令。

图 3.21 不使用 VAAI 将卷清零



随着硬件加速归零的引入，可以实现完全相同的结果
通过单个命令实现，如图3.22所示。

图 3.22 使用 VAAI 将卷清零



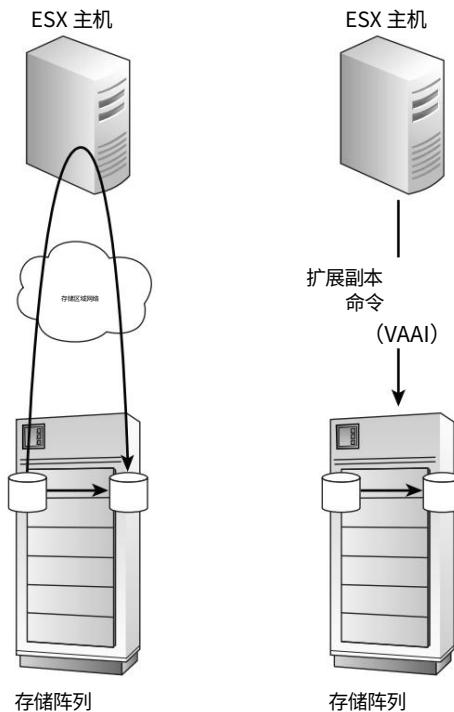
这不仅减轻了 ESX 主机的操作负担，释放了 CPU 周期，还大大减少了网络干扰，因为 ESX 主机可以发出单个命令，阵列可以给出单个响应。此外，存储可能能够更快地完成工作。最终结果是操作完成得更快，使用的 ESX CPU 和网络资源更少。

硬件加速归零通过 T10 SCSI 标准 WRITE_SAME 命令（操作码 0x93）实现。

硬件加速复制

流行的 VMware 相关操作（例如 Storage vMotion 和从模板创建新虚拟机）利用了大数据复制操作。硬件加速复制使用 SCSI EXTENDED_COPY 命令（操作码 0x83）来减轻将数据复制到存储阵列的繁重工作。图 3.23 显示了不使用和使用 EXTENDED_COPY 的大型复制操作。

图 3.23 不使用和使用硬件辅助卸载的大型复制命令



在图 3.23 中,左侧不使用EXTENDED_COPY加载的操作要求所有从源卷复制到目标卷的数据都从阵列向上传递,通过 SAN,通过 ESX 主机中的 HBA,通过内核数据移动引擎,通过 HBA 反向传递,通过 SAN 反向传递,然后返回到存储阵列。相比之下,右侧的操作显示从 ESX 主机向阵列发出的单个EXTENDED_COPY命令。然后,阵列在内部复制数据,而不必将数据一直传递到堆栈的顶部,通过主机,然后再反向传递。

通过使用EXTENDED_COPY硬件辅助,网络带宽利用率可以以及基于主机的资源利用率都会大幅下降。

精简配置

精简配置关闭 (TP Stun) 是一种尝试,旨在妥善处理过度配置的存储阵列上的空间不足情况。TP Stun 是一种机制,通过该机制,可以暂停从空间已用尽的存储阵列请求额外空间的虚拟机。

而不是崩溃。这是可以实现的，因为存储阵列现在可以通知主机其磁盘已满，而不仅仅是向主机发出写入失败，从而使主机能够更优雅地处理这种情况。一旦阵列有更多空间，情况就会清除，虚拟机就可以恢复。

虽然这听起来可能并不完美，但这是试图最大限度地利用噩梦场景的一种尝试。当然，这适用于过度配置的精简配置存储阵列和驻留在存储阵列精简配置的数据存储上的虚拟机。

VAAI 和零空间回收

最新版本的 VMware 现已完全了解基于阵列的精简配置，并针对持续保持精简卷精简提供了优化协助。

VAAI 现在使用 T10 SCSI 标准 UNMAP 命令来通知存储阵列某些块不再使用。示例包括已删除的虚拟机或已迁移到其他数据存储区的虚拟机。

NAS 完整文件克隆

NAS 的完整文件克隆将虚拟磁盘克隆的负载转移到 VMware NAS 一方。此功能类似于块世界中的 EXTENDED_COPY 功能，因为它将 VM 复制操作转移到 NAS 阵列。但是，它的功能不如其块对应项那么强大，因为它只能用于克隆操作，不能用于 Storage vMotion（这意味着必须关闭 VM）。Storage vMotion 操作仍然恢复使用 VMkernel 的软件数据移动器。

NAS 快速文件克隆

NAS 的快速文件克隆为 NAS 阵列的原生快照功能提供了对虚拟机快照创建和管理的支持。这样可以更快地创建快照，并且减少与网络相关的流量。

NAS 无需 ATS

有趣的是，NAS 世界中不需要 ATS 原语的等价物，因为 NAS 世界中从未受到诸如 SCSI RESERVE 机制之类的基本锁定系统的困扰。这是因为 NAS 阵列理解并拥有底层导出文件系统，因此具有本机高级文件级锁定功能。

微软 ODX

在 Windows 8 和 Server 2012 中，微软引入了 Offloaded 数据传输 (ODX) 面向世界。ODX 是一种数据复制技术，旨在将数据复制功能应用于智能存储阵列，其方式与 VMware 的 VAAI 非常相似。可以说，ODX 之于 Windows 就如同 VAAI 之于 VMware。不同之处在于，VMware 从事这项工作的时间更长，因此具有更多功能和更成熟的技术，尽管随着时间的推移，这种技术无疑会趋于稳定。

由于 ODX 与某些 VAAI 原语类似,它的存在是为了加速某些基于存储的操作,例如大型复制操作和批量归零,同时节省主机 CPU、NIC 利用率和网络带宽。

用于大文件复制的 ODX

图 3.24 显示了 ODX 如何执行大型复制操作。

图 3.24 高级 ODX 复制操作



在图 3.24 所示的大型复制操作中,运行 Windows Server 2012 或更高版本的主机向支持 ODX 的阵列发出 ofoad 读取命令。阵列以 512 字节令牌响应,该令牌是要复制的日期的特定于阵列的逻辑表示。此令牌称为数据表示 (ROD) 令牌。然后,主机向阵列发出 ofoad 写入请求以及相关的 ROD 令牌。然后,阵列执行复制,而数据永远不会离开阵列。

ODX 还可以在两个都支持此特定 ODX 配置的阵列之间工作。

ODX 采用以下技术:

- Hyper-V 虚拟硬盘 (VHD)
- SMB 共享 (有时称为 CIFS)
- 物理磁盘
- 光纤通道、iSCSI、FCoE、SAS

只要服务器运行的是 Windows Server 2012 或更高版本,并且底层存储阵列支持 ODX,则每次从 CLI、PowerShell 提示符、Windows 资源管理器或 Hyper-V 迁移作业发出复制命令时,都会调用 ODX 来执行数据复制。但是,重复数据删除和 Windows BitLocker 加密驱动器等功能不适用于 ODX 的初始版本。这些过滤器驱动程序的未来版本可能会支持 ODX。

ODX 利用 T10 XCOPY LITE 原语,使其基于标准。

和所有正在发展的技术一样,它正在展翅翱翔。请咨询供应商和 Microsoft 以了解支持的配置。需要注意的事项包括弹性文件系统 (ReFS)、动态磁盘和类似的发展。

ODX 和批量归零

ODX 还可以通过使用众所周知的零令牌 ROD 令牌来执行批量归零。



A ROD 令牌可以是特定于供应商的 972 字节字符串,例如,表示要复制的数据范围或者,ROD 令牌可以是众所周知的令牌(例如用于执行批量零操作的零令牌)。ROD token can be a

章节概要

双控制器和网格架构 双控制器架构有时也称为中端架构,它提供企业级网格架构中的许多高级功能,但价格更便宜。双控制器架构的可扩展性也有限,并且不能像网格架构那样处理硬件故障。网格架构提供横向扩展功能,可以更好地处理硬件故障,但成本较高。

冗余 冗余是存储设计的一大特点。大多数存储阵列在每一个可能的层面都采用冗余,以确保发生故障的组件不会中断阵列的运行。即使在主机层面,主机和存储之间通常也会在多路径 I/O (MPIO) 配置中配置多条路径,以确保主机和存储阵列之间路径或网络链路的丢失不会导致系统瘫痪。

复制基于阵列的复制 可以对生产卷进行远程复制,这在灾难恢复 (DR) 和业务连续性 (BC) 规划中发挥着至关重要的作用。

根据您的应用程序和业务需求,远程副本可以是零损失的、最新的同步副本,也可以使用异步复制技术稍微滞后。异步复制技术可以在源卷和目标卷之间相距数千英里,但同步复制要求源卷和目标卷之间的距离不超过约 100 英里。

精简配置 精简配置技术可用于更有效地利用存储阵列中的容量。但是,如果过度配置阵列,则必须非常小心,以确保阵列不会耗尽可用空间。

子 LUN 分层 子 LUN 分层技术允许您将数据放在最合适的存储层上。将经常访问的数据放在快速介质上,将不活跃的数据放在慢速介质上。这可以提高阵列的性能,并且由于大多数数据访问频率较低,因此无需在阵列中填充快速磁盘,从而降低成本。

概括

在本章中,我们介绍了与存储阵列相关的所有内容。我们讨论了架构原则,包括前端、后端、缓存的重要性、不同类型的持久性介质以及 LUN 和卷。我们甚至谈到了快照和克隆以及复制技术,尽管这些技术将在本书的专门章节中更详细地介绍。我们讨论了基于闪存的固态介质的优缺点、用例以及影响和破坏性。

我们还讨论了存储的一些高级功能和所谓的智能。

时代阵列,包括分层、重复数据删除和硬件负载,其中许多与基于服务器和基于虚拟机管理程序的技术集成。

章节

4



RAID:保持你的 数据安全

本章涵盖的主题：

- ✓ 什么是 RAID 以及我们为什么需要它
- ✓ 硬件与软件 RAID
- ✓ 条纹
- ✓ 平价
- ✓ 镜像
- ✓ 标准 RAID 级别
- ✓ 性能
- ✓ 可靠性
- ✓ RAID 的未来



本章涵盖了 CompTIA Storage+ 考试所需的所有 RAID 材料,以及更多内容。您将从基础知识开始,包括 RAID 技术的历史以及为什么它一直是并将继续成为全球企业技术的基本要求。

本章比较了 RAID 的不同实现,包括对硬件 RAID 和软件 RAID,各自的优缺点,以及两者的一些潜在用例。它还涵盖了现实世界中常见的大多数 RAID 实现的基本原理和概念,包括 RAID 组、条带化、奇偶校验、镜像和常用的 RAID 级别。

本章将介绍一些最新、更具体的实现,以及这项长期技术的未来前景如何,它是否可能开始难以满足现代世界的需求。

RAID 的历史和原因

过去,昂贵的大型计算机会安装单个大型昂贵驱动器 (SLED)。这些磁盘驱动器

- 价格非常昂贵
- 造成单点故障 (SPOF)
- IOPS 有限

RAID 的发明就是为了克服所有这些因素。



In 1981, Garth Gaudin, David Patterson, and Randy Katz published a paper titled "廉价磁盘冗余阵列 (RAID)" (the term RAID was first used in this paper). This paper described the most common RAID levels used today and has been updated every 20 years. It introduced the concept of using multiple disks to protect data against disk failures. Due to its publication as a paper and not a patent, it has seen almost universal uptake in the industry and remains fundamental to data protection in data centers across the globe.

快进 25 年多,磁盘驱动器仍然经常出现故障,而且往往没有警告。大多数大型数据中心每天都会遇到多个磁盘驱动器故障。尽管近年来硬盘驱动器容量呈爆炸式增长,但性能却没有增长,导致磁盘驱动器膨胀问题,容量和性能严重不匹配,即固态介质 bloat 问题,where capacity and performance are severely mismatched—although solid-state media is

稍微改变一下游戏规则。这些事实结合在一起，确保 RAID 技术仍然是全球数据中心平稳运行的基础。

因此，尽管 RAID 技术已有 25 年历史，但对其工作原理的深入理解正常工作并保护您的数据将使您在整个 IT 职业生涯中受益匪浅。



RAID是您的数据的一种保险单。您不希望在危机时刻发现自己没有保险。是的，RAID 是高成本的，但它停机或应用程序重建的成本相比，这笔成本简直是九牛二虎之力。此外，您的数据保护保险组合应该不仅仅包括 RAID！您的业务关键型应用程序应该具有多层次的保护，至少包括以下几项：RAID、MPIO 和备份。Portfolio for data protection should include far more than just RAID! Your business-critical applications should have multiple levels of protection, including at least the following: RAID, MPIO, replication, and backups.

什么是 RAID?

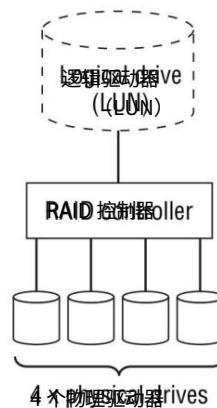
RAID 是首字母缩写词。过去它是指廉价磁盘冗余阵列 to redundant array of inexpensive disks。如今，大多数人都将其称为独立磁盘冗余阵列。而随着固态硬盘 (并非真正的磁盘) 的出现，也许它即将迎来另一种变化，成为独立驱动器冗余阵列。really disks, maybe it's about to see another change and become redundant array of independent drives.

无论如何，RAID 技术在现代 IT 世界中存在的理由有两个。the modern IT world is twofold:

- 保护您的数据免受驱动器故障的影响。
- 通过跨多个驱动器并行执行 I/O 来提高 I/O 性能。

为了提供更高的性能和保护，RAID 技术结合了 and protection, RAID technology combines 多个物理驱动器，并创建一个或多个跨越该多个物理驱动器的逻辑驱动器，如图 4.1 所示。multiple physical drives, as shown in Figure 4.1.

图 4.1 从四个物理驱动器创建的逻辑驱动器
Created from four physical drives



逻辑驱动器通常有两种类型。它可以是单个物理驱动器的子集，用于创建多个逻辑驱动器（有时称为分区）

来自单个物理驱动器。或者，如图 4.1 所示，逻辑驱动器可以是从多个物理驱动器聚合的容量创建的设备。图 4.1 显示了从四个物理驱动器的容量创建的单个逻辑驱动器。有时，逻辑驱动器可以称为逻辑磁盘、逻辑卷、虚拟磁盘或虚拟卷。

目前有几种广泛使用的 RAID 级别，例如 RAID 1、RAID 10、RAID 5 和 RAID 6，每种级别都有自己的优点和缺点。您将在本章中仔细研究更流行的 RAID 级别，并考虑它们的优点和缺点以及潜在的用例。然而，在仔细研究 RAID 级别之前，让我们先了解一些基本概念和组件。



实际上，在我们开始之前，首先要强调的是 RAID 不是备份解决方案，也不是备份的替代品。RAID 只能保护数据免受硬盘驱动器故障的影响，而不能保护数据免受其他许多可能破坏数据的因素的影响。only against disk drive failures and not the many other things that can destroy your data.

硬件或软件 RAID Software RAID

RAID 可以通过软件或硬件实现。*in software or hardware.*

软件 RAID 通常作为操作系统的 *一部分实现*，由逻辑卷管理器（如 Linux LVM）在操作系统 *上实现*，或作为文件系统的 *一部分*（如 NFS）。无论哪种方式，这都会导致软件 RAID 使用主机，资源（如 CPU 和 RAM）。由于如今我们在服务器中经常看到的超大 CPU 和大量 RAM，这个问题不再像以前那么严重。如果实现 RAID 在镜像，则基于主机的资源消耗不是什么大问题，但在执行奇偶校验 RAID（如 RAID 5 或 RAID 6）时，这个问题可能会变得更严重；尤其是在 RAID 组中有大量驱动器的情况下。此外，由于需要在 RAID，软件初始化这前启动操作系统，因此无法使用软件 RAID 对操作系统启动卷进行 RAID 保护。再加上缺少硬件 RAID 通常具有的一些附加优势，比如更快的写入性能和更快的重建速度，软件 RAID 很快就开始看起来像穷人的 RAID。*ware can initialize, OS boot volumes cannot be RAID protected with software RAID. Throw into the mix the lack of some of the fringe benefits that typically come with hardware RAID, such as faster write performance and faster rebuilds, and software RAID quickly starts to look like a poor-man's RAID.*

Microsoft 存储空间在 Windows Server 2012 操作系统中，Microsoft 实现了称为存储空间的 LVM，known as Storage Spaces。

存储空间允许您将串行驱动器汇集在一起，并使用这些池中的容量创建虚拟卷（称为存储空间）。这些存储空间可以选择受软件 RAID 保护，镜像保护（RAID 1）或奇偶校验保护（RAID 5）。*storage spaces can optionally be software RAID protected—either mirror protected (RAID 1) or parity protected (RAID 5).*

虽然存储空间是最近对尝试在relatively recent attempt at providing advanced storage features in 操作系统软件，其软件 RAID 的实现仍然受到相同性能的影响。still suffers from the same performance

在使用奇偶校验进行保护时,其他软件 RAID 解决方案会遭受损失。此外,存储空间不提供 ZFS 所具有的高级文件系统和软件 RAID 集成。

虽然存储空间的池化方面可能很好,但是软件 RAID 实现还很基础,还未能达到逃逸速度,仍然停留在许多其他软件 RAID 实现中发现的相同性能和使用限制上。



Mirroring parity and RAID levels are explained in more detail in the “RAID Concepts” section.

另一方面,硬件 RAID 是指在服务器硬件 (主板上或扩展卡) 中实现的物理 RAID 控制器,该控制器拥有自己的专用 CPU 以及由电池或闪存支持的 RAM 缓存。由于有此专用硬件,因此主机 CPU 上不会产生任何 RAID 开销。此外,由于硬件 RAID 控制器在操作系统之前初始化,这意味着启动卷可以受到 RAID 保护。

RAID 控制器上的电池备份或热备份缓存还通过写回缓存提供了改进的写入性能。当 I/O 到达缓存时向主机确认,而不必等到它写入 RAID 组中的驱动器。服务器内部的硬件 RAID 控制器的一个潜在缺点是它们提供 SPOF。丢失 RAID 控制器,您就会丢失 RAID 组。尽管如此,如果您负担得起,大多数人会发现他们大多数时候都会使用硬件 RAID。

表 4.1 重点介绍了硬件和软件 RAID 的优缺点。

表 4.1 硬件 RAID 与软件 RAID

特征	硬件 RAID	软件 RAID
主机 CPU 卸载	是的	不
启动卷的 RAID	是的	不
回写缓存	是的	不
热插拔驱动器	是的	不总是
复杂	降低	更高
成本	高的	低的
灵活性	降低	更高

外部 RAID 控制器（例如存储阵列）也是硬件 RAID 的形式。与通常安装在服务器硬件中的独立硬件 RAID 控制器相比，存储阵列通常具有多项优势。这些优势包括：

- 冗余。安装在服务器主板上或作为 PCIe 卡的 RAID 控制器是单点故障。虽然存储阵列本身可能成为单点故障，但它们往往具有极高的可用性，很少发生灾难性故障。
- 存储阵列提供更大的缓存，因此通常可以获得更高的性能。
- 存储阵列支持多种驱动器类型。内部 RAID 控制器通常不允许您混合搭配驱动器。
- 存储阵列支持更多驱动器，以增加容量并通过并行化提高性能。
- 存储阵列通常提供高级功能，例如快照、复制、精简配置等。

Microsoft 存储空间 软件 RAID

在 Windows Server 2012 操作系统中，Microsoft 实现了称为存储空间的 LVM。

存储空间允许您将异构驱动器汇集在一起，并使用这些池中的容量创建虚拟卷（称为存储空间）。这些存储空间可以选择受软件 RAID 保护 - 镜像保护 (RAID 1) 或奇偶校验保护 (RAID 5)。

虽然存储空间是最近在操作系统软件中提供高级存储功能的尝试，但其软件 RAID 实现在使用奇偶校验进行保护时仍然会遭受其他软件 RAID 解决方案所遭受的性能损失。此外，存储空间不提供 ZFS 所具有的高级文件系统和软件 RAID 集成。

虽然存储空间池化方面可能很好，但软件 RAID 实现思维是基础的，还没有能够达到逃逸速度，仍然停留在许多其他软件 RAID 实现中发现的相同性能和使用限制上。

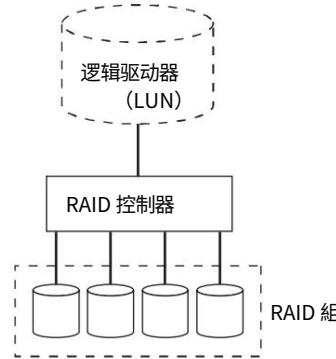
RAID 概念

让我们来了解一下构成大多数 RAID 技术的主要概念。充分理解这些概念对于全面理解 RAID 技术及其对数据可用性和性能的影响至关重要。

RAID 组

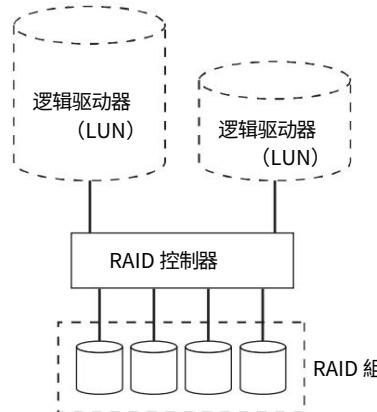
RAID 组,也称为 RAID 集或 RAID 阵列,是指一组组合并配置为协同工作以提供更高容量、更高性能和更高可靠性的驱动器。RAID 组中的所有驱动器都连接到同一个 RAID 控制器,并归该 RAID 控制器所有。图 4.2 显示了一个包含四个驱动器的 RAID 集,该集已配置为单个逻辑驱动器。

图 4.2 配置为单个逻辑驱动器的 RAID 组



每个 RAID 组也可以配置多个逻辑卷。图 4.3 显示相同的四磁盘 RAID 组,但这次配置为两个不同大小的逻辑驱动器。

图 4.3 配置为两个逻辑驱动器的 RAID 组



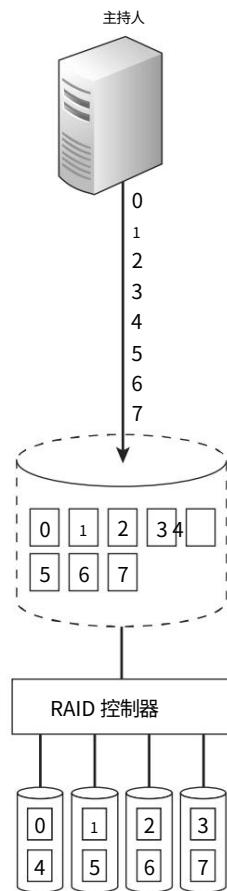
RAID 组中的驱动器有时被称为成员。

条纹

创建分布在 RAID 集所有驱动器上的逻辑驱动器/逻辑卷的过程称为条带化。条带化背后的理念是将 I/O 并行到尽可能多的驱动器上，以获得每个驱动器的性能。对分布在 RAID 集所有驱动器上的逻辑驱动器的每次写入都会分布在 RAID 集的所有驱动器上，从而有可能使其能够访问每个驱动器的所有 IOPS 和 MB/秒。

图 4.4 显示了主机的 I/O 被写入到条带化的逻辑卷四个物理驱动器。如果每个物理驱动器能够达到 200 IOPS，则逻辑卷的潜在性能为 800 IOPs。如果卷不是条带化的，而是连续的，则逻辑卷的潜在性能为 800 IOPs。由于它完全依赖于单个驱动器，因此其性能只能达到 200 IOPS。

图 4.4 I/O 分布在四个物理驱动器上



条带化是许多 RAID 算法的基础技术，适用于和可靠性原因。

将逻辑驱动器条带化到多个物理驱动器的过程还允许逻辑驱动器访问每个物理驱动器上的所有可用容量。例如，在包含四个 900 GB 驱动器的简单 RAID 集上创建的逻辑卷可能大到 3,600 GB。因此，条带化还可以提高容量和性能。

平价

奇偶校验是 RAID 算法中使用的一种技术，用于提高 RAID 集的弹性，有时也称为容错能力。

基于奇偶校验的 RAID 算法会保留 RAID 集的一定比例的容量（通常情况下，一个或两个驱动器的容量）用于存储奇偶校验数据，使 RAID 组能够从某些故障情况（例如驱动器故障）中恢复。

让我们看一个简单而快速的例子，了解奇偶校验如何保护和重建二进制数据。奇偶校验可以是偶校验或奇校验，使用哪一种都无所谓，因为它们都实现了相同的最终目标：为二进制数据集提供容错能力。让我们在示例中使用奇校验。奇校验的工作原理是确保二进制序列中始终有奇数个 1。让我们假设所有二进制序列都包含四位数据，例如 0000、0001、1111 等等。现在让我们在第四位后添加一个额外的数据位，并将这个额外的位用作我们的奇偶校验位。由于我们使用奇校验，我们将使用此位来确保我们的五位数据中始终有奇数个 1。让我们看几个例子：

00001 数据的第 f 位（我们的奇偶校验位）为 1，因此我们的数据集中有奇数个（一个）1。

01110 这次我们将数据的第 f 位设为 0，以确保数据集中 1 的个数为奇数（三个）。

10000 这次我们将奇偶校验位设为 0，这样数据集中的 1 就有奇数个（一个）。如果我们将其设为 1，数据集中就会有两个 1。

一切都很好，但这如何使我们的数据集具有容错能力？首先，由于我们只添加了一个奇偶校验位，我们的数据集只能容忍丢失一个数据位，但丢失的位可以是数据集中的任何位。无论如何，要了解它的工作原理，让我们再次查看我们的三个示例数据集，只是这次我们删除了一个数据位，并将使用奇校验原理来确定和重建丢失的数据位。

0_001 由于我们使用奇校验，所以我们缺失的位一定是 0。否则，数据集中就会有偶数个 1。

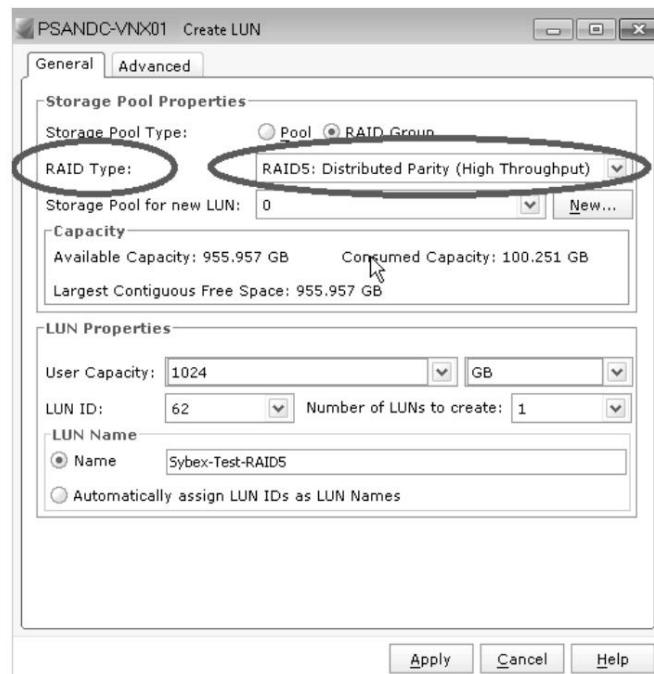
011_0 这次我们知道缺失的位是 1。如果是 0，我们的数据集中就会有偶数个 1。

1000_ 这次我们丢失了奇偶校验位。但这没什么区别。我们仍然可以应用相同的逻辑并确定我们缺少了一个 0。

由于我们的示例数据集有 5 位数据,但仅包含 4 位实际用户数据 (一位是奇偶校验开销),因此我们损失了一位,即 20%。换句话说,我们的数据集的效率为 80%,因为 80% 的位用于实际数据,20% 用于提供容错能力。了解在底层,奇偶校验是使用异或 (XOR) 运算进行计算的,这可能很有用,尽管了解这一点不会对您的日常工作有所帮助。

图 4.5 显示了在 EMC VNX 阵列上配置 RAID 5 卷的选项。
您可以看到,用于创建卷的 GUI 不会询问有关 XOR 计算或其他 RAID 内部的任何信息。

图 4.5 在 EMC 存储阵列上创建 RAID 5 卷



为数据保护预留多少奇偶校验以及 RAID 组的整体容错能力取决于所采用的 RAID 级别。例如,RAID 5 可以容忍单个驱动器故障,而 RAID 6 可以容忍两个驱动器故障。我们将在稍后详细讨论这些内容。

图 4.6 显示了五驱动器 RAID 组中的单个驱动器被保留用于奇偶校验数据。在此示例中,RAID 组的可用容量减少了 20%。此 RAID 组还可以容忍单个驱动器故障。

图 4.6 带有单个奇偶校验驱动器的五驱动器 RAID 组

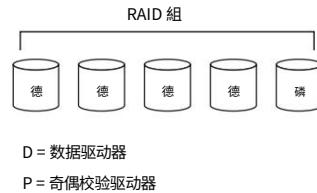
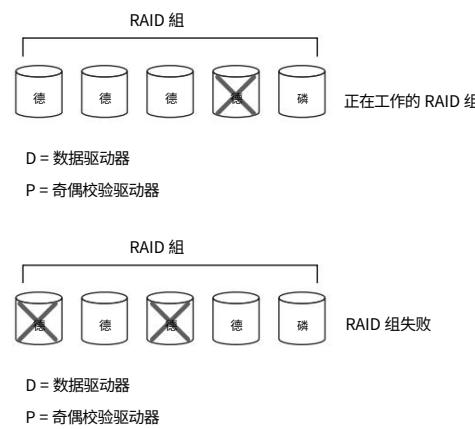


图 4.7 显示了同一五驱动器 RAID 组在经历单驱动器和双驱动器故障后的情况。

图 4.7 五驱动器 RAID 组,其中单驱动器和双驱动器发生故障



如果图 4.7 中的 RAID 组包含两个奇偶校验驱动器 (容错性更高),那么它就可以在双驱动器故障的情况下幸存下来。但是,第二个奇偶校验驱动器会使 RAID 组的可用容量从 80% 减少到 60%。

奇偶校验通常通过排他或 (XOR) 计算来计算,而 RAID 控制器使用奇校验还是偶校验并不重要 只要它能正确计算即可。

基于奇偶校验的 RAID 方案在承受高写入工作负载时可能会受到性能影响,尤其是由小块随机写入组成的高写入工作负载。这被称为写入惩罚,我们将在后面详细讨论。当遇到高读取工作负载时,基于奇偶校验的 RAID 方案表现良好。

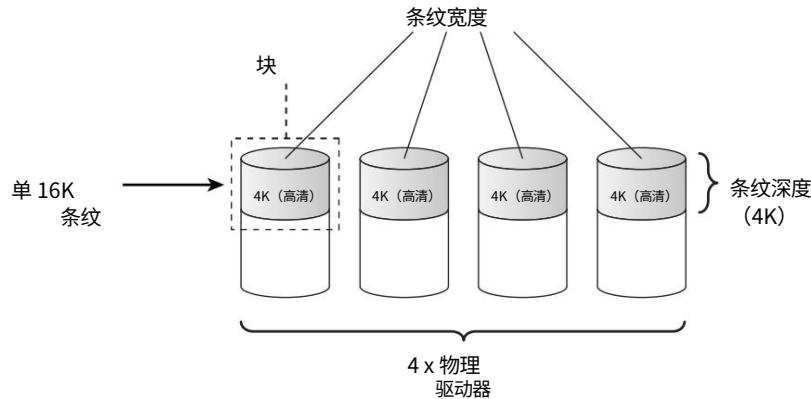
RAID 条带和条带大小

所有采用条带化和奇偶校验的传统 RAID 形式均在条带基础上工作。

条带由一排块组成。这些块的大小决定了条带深度和条带大小,而 RAID 集中的成员数量决定了条带宽度。我觉得我们需要一张图!

图 4.8 显示了配置为 RAID 5 (3+1) 的四驱动器 RAID 组，并有说明组成条纹的每个元素。

图 4.8 条纹元素



现在介绍适用于大多数传统 RAID 集的几个简单原则和一些术语。

条带由 RAID 集中每个驱动器的块组成。每个块由单个物理驱动器中的连续块组成。RAID 配置中的所有块大小相等（通常是驱动器扇区大小的倍数），此大小称为条带深度。条带宽度等于 RAID 组中的驱动器数量。图 4.8 中的 RAID 集具有以下特征：

- 条带深度为 4 KB。这是因为每个块为 4 KB，并且块大小等于条纹深度。
- 条带大小为 16 KB。四个 4 KB 块。
- 条带宽度为 4。RAID 集中有四个成员。

当主机向 RAID 控制器上的卷写入数据时，RAID 控制器会将这些数据以条带形式写入，这样这些数据就会触及 RAID 集中的所有驱动器。例如，如果主机向图 4.8 中的 RAID 集写入 24 KB，RAID 控制器会将其以两个条带形式写入。

这是因为每个条带具有 12 KB 的可用数据和 4 KB 的奇偶校验。当它将这些数据写入条带时，它会将 4 KB 写入驱动器 0，然后切换到驱动器 1，在那里写入下一个 4 KB，依此类推。将 24 KB 写入此 RAID 集将需要两行或条带。



存储阵列（本身就是 RAID 控制器的条带）的大小通常比单机的 RAID 控制器大得多。在增加阵列的块大小通常为 64KB、128KB 或 256KB。如果存储阵列的块大小为 128KB，则 RAID 5 (3+1) 的条带大小将为 1,024 KB。存储阵列中较大的条带大小往往与阵列缓存的插槽大小一致。

块大小和性能

以下内容可作为在大多数情况下适用的指导原则：如果您的 I/O 大小很大，则需要使用较小的块大小。如果您的 I/O 大小很小，则需要较大的块。

那么，什么构成了小块或大块大小？通常，小块可以从 512 字节到 4 KB 或 8 KB 不等。任何大于该大小的块都开始进入大块的范畴。

小 I/O 和大块

小型 I/O（例如数据库环境中常见的 4 KB I/O）往往在较大的区块大小下工作得更好。这里的目标是让 RAID 集中的每个驱动器忙于工作并独立地搜索单独的 I/O。这种技术称为重叠 I/O。

如果您的块大小较大，而 I/O 较小，则很容易导致定位延迟增加。也就是说，虽然一个驱动器上的磁头可能已经到达位置并准备好为其部分 I/O 提供服务，但 RAID 组中其他驱动器上的磁头可能尚未到达位置，这意味着您必须等到 RAID 组中所有驱动器上的磁头都到达位置后，才能返回 I/O。如果您让 RAID 控制器承受大量具有大块大小的小块随机 I/O，这可能会造成损害并降低性能。

大 I/O 和小块

处理大型文件的低 I/O 环境通常受益于较小的条带大小，因为这允许每个大型文件跨越 RAID 集中的所有物理驱动器。这种方法使每个文件的访问速度更快，涉及 RAID 集中的每个驱动器来读取或写入文件。这种配置不利于重叠 I/O，因为阵列中的所有驱动器都将用于读取或写入单个大型文件。

小块大小的常见用例包括访问大型音频的应用程序/
视频文件或经常处理大文件的科学成像应用程序。

混合工作负载

幸运的是，大多数 RAID 控制器上的默认设置通常足够好，并且专为混合工作负载而设计。建议仅在您知道自己在做什么并且知道您的工作负载特征是什么的情况下更改默认值。如果您也不知道，请不要弄乱块和条带大小。此外，大多数存储阵列不允许您对此进行调整，因为它与缓存插槽大小和其他阵列内部紧密相关。

热备件和重建

某些 RAID 阵列包含备用驱动器，称为热备用或在线备用。此热备用在正常运行情况下处于待机模式（通常处于通电状态但不在使用中），但如果驱动器发生故障，RAID 控制器会自动启用它。例如，具有六个驱动器的系统可以配置为 RAID 5 (4+1)，其余驱动器用作热备用。

热备件的主要原理是使 RAID 组能够开始重建尽快恢复。例如，假设您的 RAID 组在凌晨 2 点遇到故障驱动器，而现场没有人更换故障驱动器。如果它有热备用，它可以自动启动重建过程，并在您早上上班时重新开始恢复。显然，较大的磁盘驱动器需要更长的时间来重建，而在当今世界，驱动器超过 4 TB，这些驱动器可能需要几天才能重建。

重建热备用驱动器的过程通常称为“备用”。

热备用在基于主机的 RAID 控制器中并不常见，但在存储阵列中实际上是强制性的，大型存储阵列通常包含多个热备用驱动器。

分布式备用和备用空间

一些现代存储阵列实际上并不包含物理热备用驱动器。相反，它们在阵列中的每个驱动器上保留少量空间，并将此空间留出以备驱动器发生故障时使用。这有时被称为分布式

备用。例如，具有 100 个 1 TB 驱动器的存储阵列可能会在每个驱动器上保留约 2 GB 的空间作为备用空间，以备驱动器发生故障时使用。此空间将从阵列的总可用容量中扣除。

此保留备用空间量相当于两个 1 TB 驱动器，这意味着阵列具有相当于两个备用驱动器的空间。如果发生驱动器故障，阵列将把故障驱动器的内容重建到阵列中每个幸存驱动器上的备用空间。这种方法的优点是可以知道备用空间所在的驱动器处于工作状态。如果只在驱动器发生故障时使用专用备用驱动器，则热备用驱动器本身很容易在最需要时发生故障。同样重要的是，系统中的所有驱动器始终可用，从而提高了系统性能，并允许 RAID 重建分布在所有驱动器上，从而使重建操作更快并降低性能影响。

当然，这个例子过于简单，但可以解释其原理。

重建通常通过以下两种方法之一完成：

驱动器副本，用于在 RAID 1 等镜像集中重建

校正副本/奇偶校验重建，用于基于奇偶校验保护重建集合

重建的性能影响

与奇偶校验重建相比，驱动器副本重建在计算上很简单，这意味着它们比从奇偶校验重建要快得多。

RAID 1 镜像集始终使用驱动器副本进行恢复，但基于奇偶校验的 RAID 方案例如 RAID 5 和 RAID 6 通常必须通过奇偶校验重建来恢复数据。

让我们看看驱动器副本重建和奇偶校验重建：

■ 驱动器复制操作是首选的重建方法,但只有当 RAID 控制器主动检测到驱动器故障时才能执行。这意味着驱动器实际上尚未发生故障,但 RAID 控制器监控的驱动器阈值表明即将发生故障。这称为预测性故障。驱动器复制重建只会对故障驱动器和热备用驱动器施加额外的压力。

■ 一旦奇偶校验集中的驱动器实际发生故障,奇偶校验重建绝对是最后一道防线。这些重建的计算量比驱动器复制更大,并且会给 RAID 集中的所有驱动器 (包括热备用驱动器) 带来额外负载。所有幸存的驱动器都从条带 0 读取到最后一个条带,并且对于每个条带,通过异或 (XOR) 计算由奇偶校验重建丢失的数据。当奇偶校验重建期间第二个驱动器发生故障时,有时会将 RAID 集中幸存驱动器上的额外负载归咎于此。

关于将额外负载归咎于第二个驱动器故障这一点可能很重要。在重建操作期间,RAID 集中的所有驱动器都会承受额外负载。

这个额外的负载很可能会导致 RAID 组中的其他驱动器发生故障。

一些较现代的 RAID 实施正在解决这个风险。

RAID 重建优先级

此外,许多 RAID 控制器 (包括存储阵列) 都允许您对重建过程进行优先级排序。为重建赋予较低的优先级会导致重建时间更长,而为重建赋予较高的优先级可能会在重建操作期间对用户 I/O 产生影响。

如果您拥有大型池或单奇偶校验 RAID,您可能希望提高重建的优先级,以便减少第二个驱动器故障的可能性。相反,如果您使用双奇偶校验 RAID,您可能更愿意降低重建的优先级,因为您知道您可以安全地承受第二个驱动器故障而不会丢失数据。

图 4.9 显示了 EMC CLARiiON 阵列的屏幕截图,其中显示了可用的选项
重建优先。

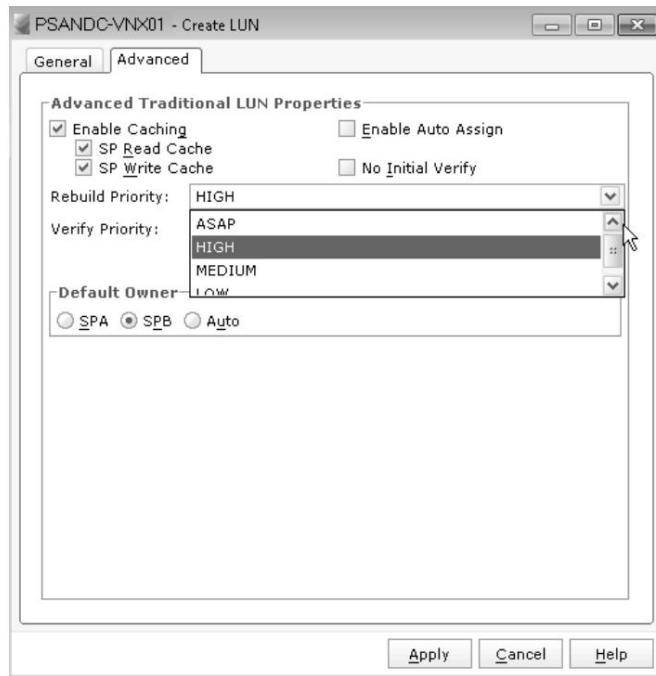
重建域和爆炸区域

重建 RAID 组中的故障驱动器时,RAID 组通常是重建域。

这意味着所有重建活动和开销都放在 RAID 集内的驱动器上。同样,爆炸区是指因 RAID 集故障 (例如 RAID 5 集中的双驱动器故障) 而丢失数据时的影响区域。使用传统 RAID 集时,爆炸区只是故障的 RAID 集和使用该 RAID 集的任何卷。

然而,在实施由多个 RAID 集组成的存储池的更现代的阵列中,故障的 RAID 集可能导致与 RAID 集关联的池中每个卷的数据丢失,从而大大增加爆炸区域的大小。

图 4.9 设置 EMC 阵列的重建优先级



RAID 控制器

RAID 控制器可以是软件也可以是硬件。软件 RAID 控制器就是软件，通常通过逻辑卷管理器实现。另一方面，硬件 RAID 控制器是专门为执行 RAID 和卷管理功能而设计的专用硬件。

硬件 RAID 控制器有两种主要类型：

- 内部
- 外部

如前所述，硬件 RAID 控制器是专用硬件，位于主板上或作为 PCIe 卡，可将所有 RAID 功能从主机 CPU 和 RAM 中卸载。这些 RAID 功能包括创建逻辑磁盘（在 SCSI 术语中称为逻辑单元，通常缩写为 LUN）、条带化、镜像、奇偶校验计算、重建、热备用和缓存。

内部 RAID 控制器上的写回缓存

大多数内部 RAID 控制器都具有板载 DRAM/NVRAM 缓存，用于加快读写速度。但是，为了安全地加快写入速度，RAID 控制器必须

能够在电量不足时保护其缓存不丢失数据。现代 RAID 控制器有两种常见方法可以实现此目的：

电池供电缓存 在电池供电缓存中,RAID 控制器卡上放置一个小电池或电容器 (SuperCap),用于为 DRAM 缓存供电,以便在断电时不会丢失其内容。但是,必须在电池耗尽之前恢复供电。否则,数据仍然会丢失。

闪存支持缓存 在闪存支持缓存中,电池或超级电容放置在 RAID 控制卡上,用于为 DRAM 缓存供电足够长的时间以将其内容复制到闪存中,从而保留其内容。

受数据中心崩溃影响,电池供电的写回缓存 (BBWC) 正在逐渐失去吸引力。
备份缓存。但是,它们仍然存在。

缓存技术对于存储行业来说就像氧气一样。如果存储解决方案缺少缓存,其性能就会急剧下降。本书的多个章节都介绍了将缓存置于旋转磁盘前面的好处,但缓存在 RAID 系统中也有积极的好处。任何缓存所有入站写入的 RAID 控制器或存储阵列都应该能够在正常操作情况下避免大多数写入的奇偶校验开销。这是通过将小写入保存在缓存中并将它们合并为更大的全条带写入 (FSW) 来实现的,其中奇偶校验计算的开销可以忽略不计。

RAID 级别

本节介绍现实世界中最常见的 RAID 级别:RAID 0、RAID 1、RAID 10、RAID 5 和 RAID 6。我们还将探讨不太常用的 RAID 级别。

RAID 0

首先,RAID 0 没有任何冗余!这意味着它无法保护您的数据。没有奇偶校验。没有镜像。什么都没有。什么都没有!

如果要我推荐 RAID 0 的一件事,那就是永远不要使用它。虽然有些人可能会想出一些适合它的配置,但我宁愿谨慎行事,像躲避瘟疫一样避免使用它。

然而,如果它出现在考试或电视智力竞赛节目中,我们会给它几行。

RAID 0 是无奇偶校验的条带化。条带化意味着数据分布在 RAID 组的所有驱动器上,从而实现并行性。磁盘越多 = 性能越高。此外,缺少奇偶校验意味着没有任何容量会因奇偶校验而丢失。但这不是一件好事。缺少奇偶校验使 RAID 0 成为一种有毒技术,如果您使用它,肯定会让您大吃一惊。

RAID 0 成本和开销

RAID 0 没有 RAID 开销,因为没有使用任何容量进行镜像或奇偶校验。而且,由于无需执行奇偶校验计算,因此也不会产生性能开销。因此,RAID 0 在成本和性能方面无疑是好消息。然而,在弹性方面,它却是一颗定时炸弹。

RAID 0 的使用案例

RAID 0 的潜在用例包括:

- 1.如果你想丢失数据
- 2.如果你想丢掉工作
- 3.如果你的数据 100% 是临时的,那么丢失数据就不是什么问题,而且你需要尽可能多地利用硬盘的容量
- 4.只要有其他形式的数据保护措施 (如网络 RAID 或副本),就可以在 RAID 0 组中丢失数据时使用
- 5.如果要在已经是 RAID 的卷上创建条带逻辑卷
受保护

不推荐选项 1 和 2。

选项 3 可能是一个毒药。太多系统最初只是为了测试目的而存在,然后突然发现它们对于任务至关重要。而且您不希望任何任务关键型的东西都建立在 RAID 0 的沙质基础上。

选项 4 更合理,但也可能最终成为一杯毒酒。当数据丢失时,人们总是会寻找其他人来承担责任。你可能会发现,责任的矛头指向了你。

选项 5 在 Linux 世界中相当常见,其中 LVM 卷管理器是众所周知,非常流行。但是,支持 RAID 0 条带卷的底层卷必须受到 RAID 保护,这一点至关重要。练习 4.1 引导您完成使用 Linux LVM 创建条带卷 (RAID 0) 的示例。

练习 4.1

使用 Linux LVM 创建条带卷

假设您有一台 Linux 主机,其中有四个 100 GB 的卷,它们是 RAID 6 卷: sda、sdb、sdc 和 sdd。您将在这四个 RAID 6 卷之上创建一个条带卷 (RAID 0)。请按照以下步骤操作:

- 1.在所有四个 RAID 6 卷上运行 pvcreate 命令。请小心,因为这样做将会破坏这些卷上已有的所有数据。

```
# pvcreate /dev/sda
# pvcreate /dev/sdb
```

```
# pvcreate /dev/sdc
# pvcreate /dev/sdd
```

2.现在,您将使用四个卷创建一个名为vg_sybex_stripped的卷组

您刚刚运行了pvcreate命令。

```
# vgcreate vg_sybex_stripped /dev/sda /dev/sdb /dev/sdc /dev/sdd
```

3.创建卷组后,运行vgdisplay命令以确保

卷组配置已应用并且看起来正确。

```
# vgdisplay
--- 卷组 ---
容量名称          删除 vg_sybex_stripped
VG 访问          读/写
VG 状态          可用/可调整大小
非常量#          2
<输出被截断>
```

4.确认卷组配置符合预期后,运行

按照lvcreate命令创建单个 200 GB 的逻辑卷。

```
# lvcreate -i4 -I4 -L 200G -n lv_sybex_test_vvol vg_sybex_stripped
```

-i选项确保逻辑卷将使用您使用pvcreate命令创建的所有四个物理卷。 -I选项指定 4 KB 条带大小。 -L命令指定大小为 200 GB。

您现在有一个 200 GB 的 RAID 0 (条带化)逻辑卷,它位于四个 100 GB 的 RAID 6 卷之上。这些 RAID 6 卷是 SAN 呈现的卷,这意味着存储阵列正在提供 RAID 6 保护。此外,卷组中还有大约 200 GB 的可用空间,您可以使用这些空间来创建更多逻辑卷或在未来扩展现有的逻辑卷。

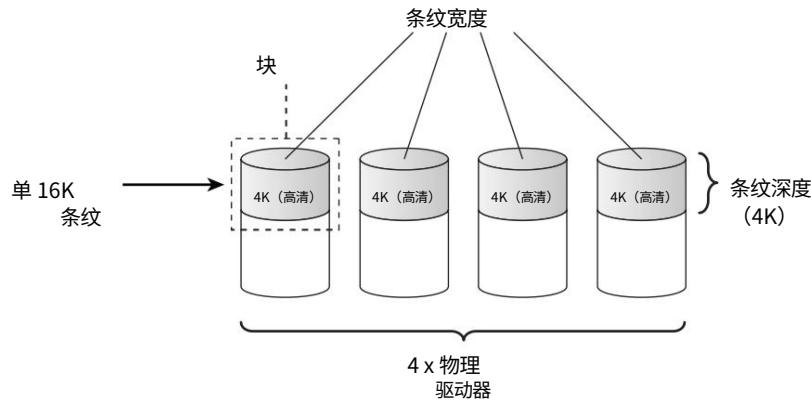
当 RAID 0 组中的某个驱动器发生故障时会发生什么情况

您会丢失数据。

故事结束了。您可能会面临当前职位的终止。您无需担心性能下降或重建时间。两者都不会发生。如果幸运的话,您将拥有一个远程副本或备份,以实现某种形式的恢复。

图 4.10 显示了职业限制的 RAID 0。

图 4.10 RAID 0 的弊端



Real World Scenario

RAID 0 无法保护您的数据！

一家公司因 RAID 0 卷故障而丢失了数据。必须根据近 24 小时前的备份从备份磁带中恢复这些数据。负责存储阵列的管理员坚持认为他没有创建任何 RAID 0 卷，阵列肯定有问题。供应商分析了阵列的日志，证明管理员确实创建了多个 RAID 0 卷。日志甚至显示了创建卷的日期，并记录了管理员的

RAID 0 卷会在任意驱动器发生故障时丢失数据的警告。不用说，这位管理员在公司待不了多久。

RAID 1

RAID 1 是镜像。典型的 RAID 1 组包含两个磁盘 其中一个是另一个的副本。这两个磁盘通常被称为镜像对。当写入进入 RAID 控制器时，它会写入镜像对中的两个磁盘。在任何时候，镜像中的两个磁盘都是最新的并且彼此同步。对 RAID 1 卷的所有写入都会写入 RAID 组中的两个磁盘。如果这不可能，则 RAID 组被标记为降级。



降级模式是指 RAID 组的一个或多个成员（驱动器）发生故障时的状态。在这种情况下，RAID 级别将根据剩余驱动器的数量降低性能。在降级模式下，RAID 组将继续提供 I/O 服务，但如果剩余的驱动器发生故障，则数据丢失的风险会更大，并且性能可能会因必须从奇偶校验重建数据而受到阻碍。对于不使用奇偶校验的 RAID 1 来说，后者并非如此。但是，即使在 RAID 1 组中，读取性能也可能降低，因为无法再从镜像对中的任何一个磁盘提供读取服务。

RAID 1 组至少包含两个驱动器。正如镜像一词所暗示的那样，一个驱动器是另一个驱动器的镜像副本。这种配置通常写为 RAID 1 (1+1)。

1+1 指的是一个驱动器用于主副本，另一个驱动器用于镜像副本。如果 RAID 组中的任一驱动器发生故障，则另一个驱动器可用于读取和写入。如果两个驱动器都发生故障，则 RAID 组中的数据将丢失。

RAID 1 被认为是一种高性能且安全的 RAID 级别。它之所以被认为是高性能，是因为与基于奇偶校验的 RAID 相比，镜像写入操作的计算量较小。它之所以被认为是安全的，是因为重建时间相对较快。这是因为幸存的成员上仍存在数据的完整副本，并且可以通过快速驱动器副本而不是较慢的奇偶校验重建来重建 RAID 集。

都好。

练习 4.2 展示如何使用 Linux LVM 创建镜像逻辑卷。

练习 4.2

使用 Linux LVM 创建镜像逻辑卷

假设您有一台 Linux 主机，该主机已配备两个未受 RAID 保护的 250 GB 卷。这些卷可能是 SAN 卷或本地连接的 DAS 驱动器：sda 和 sdb。现在让我们将这两个卷组成一个镜像逻辑卷（RAID 1）。

1. 在两个 250 GB 卷（sda 和 sdb）上运行 pvcreate 命令。这是一个破坏性过程，将破坏设备上已有的所有数据。

```
# pvcreate /dev/sda
# pvcreate /dev/sdb
```

2. 现在使用两个卷创建一个名为 vg_sybex_mirrored 的卷组
您刚刚运行了 pvcreate 命令。

```
# vgcreate vg_sybex_mirrored /dev/sda /dev/sdb
```

3. 创建卷组后，运行 vgdisplay 命令以确保
卷组配置已应用并且看起来正确。

```
# vgdisplay
--- 卷组 ---
容量名称          镜像
VG 访问          读/写
VG 状态          可用/可调整大小
非常量#          1
<输出被截断>
```

4. 确认卷组配置符合预期后，运行以下 lvcreate 命令来创建单个 200 GB 的镜像逻辑卷。

```
# lvcreate -L 200G -m1 -n lv_sybex_mirror_vol vg_sybex_mirrored
```

-L 命令指定大小为 200 GB。值得注意的是，200 GB 的镜像逻辑卷不会占用您创建的卷组的所有容量。这样就可以了。

您现在有一个 200 GB 的 RAID 1（镜像）逻辑卷，位于两个 250 GB 设备的顶部。

RAID 1 成本和开销

RAID 1 的缺点（而且这只是存储管理员和拥有存储预算的成本中心所有者的缺点）是 RAID 1 牺牲了大量容量来提供保护。所有 RAID 1 配置都使用 50% 的总物理容量来提供保护镜像。这种牺牲的容量称为 RAID 开销。

另外，假设我们假设资本支出（cap-ex）成本为每 TB 5,000 美元，您需要购买 32 TB。从表面上看，这听起来可能是 160,000 美元。但是，如果需要 RAID 1 保护，您需要购买双倍的原始容量，这显然会使资本支出成本从 160,000 美元翻倍至 320,000 美元。这很伤人！

出于这个原因，RAID 1 往往被谨慎使用。这不是双关语。

图 4.11 显示了 RAID 1 组中的两个 600 GB 驱动器。尽管生成的逻辑驱动器只有 600 GB，但它实际上占用了 RAID 组中所有可用空间，因为这是 RAID 1 组，因此会因 RAID 开销而损失其物理原始容量的一半。

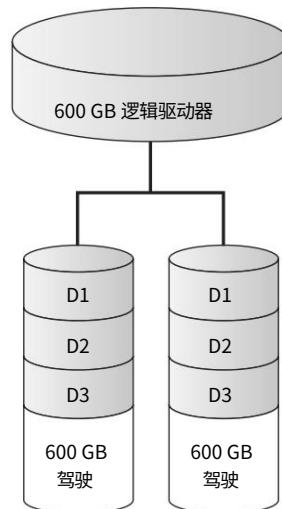
RAID 1 的使用案例

如果不严格控制，每个人都会要求使用 RAID 1。数据库管理员（DBA）尤其容易犯这种错误，因为他们都接受了反对 RAID 5 的宣传。如果他们坚持使用 RAID 1，请让他们言出必行，并向他们收取额外费用。

RAID 1 的各方面性能均十分出色，并且在以下方面表现优异：

- 随机写入性能。这是因为对于小规模随机写入，不会产生写入损失，而 RAID 5 和 RAID 6 则不会产生写入损失（稍后会详细介绍）。
- 读取性能。根据 RAID 控制器如何实现 RAID 1，镜像对中的任一驱动器都可以满足读取请求。这意味着，对于读取性能，您可以获得两个驱动器的所有 IOPS。并非所有 RAID 控制器都实现这一点。

图 4.11 RAID 1 镜像



驱动器发生故障时会发生什么情况

当 RAID 1 组中的某个驱动器发生故障时，该组将被标记为降级，但所有读/写操作仍可继续。读取性能可能会受到影响。

一旦故障驱动器被新驱动器替换或热备用驱动器启动，RAID 组将通过驱动器复制操作重建，其中幸存驱动器的所有内容将复制到新驱动器。一旦此驱动器复制操作完成，RAID 组将恢复到完全冗余。

RAID 10

RAID 1+0，通常称为 RAID 10，读作 RAID ten，是 RAID 1（镜像集）和 RAID 0（条带集）的混合体。其目的是实现两全其美。一般来说，它的工作原理正是如此。

让我们看看如何创建 RAID 10 组。假设我们有四个 600 GB 的驱动器，并且我们想要创建一个 RAID 10 组。我们取前两个驱动器并从它们创建一个镜像对。然后我们取最后两个驱动器并从它们创建一个单独的镜像对。

这样我们就得到了两个逻辑上的 600 GB 驱动器，每个驱动器都受 RAID 1 保护。下一步是利用这两个受 RAID 1 保护的 600 GB 驱动器，并在它们之间创建一个条带集。条带集不会添加任何镜像或奇偶校验，因此不会丢失任何额外容量。

它所做的只是通过将数据条带化到更多驱动器来提高性能。现在的最终结果是 1,200 GB 的容量既是镜像的又是条带化的。

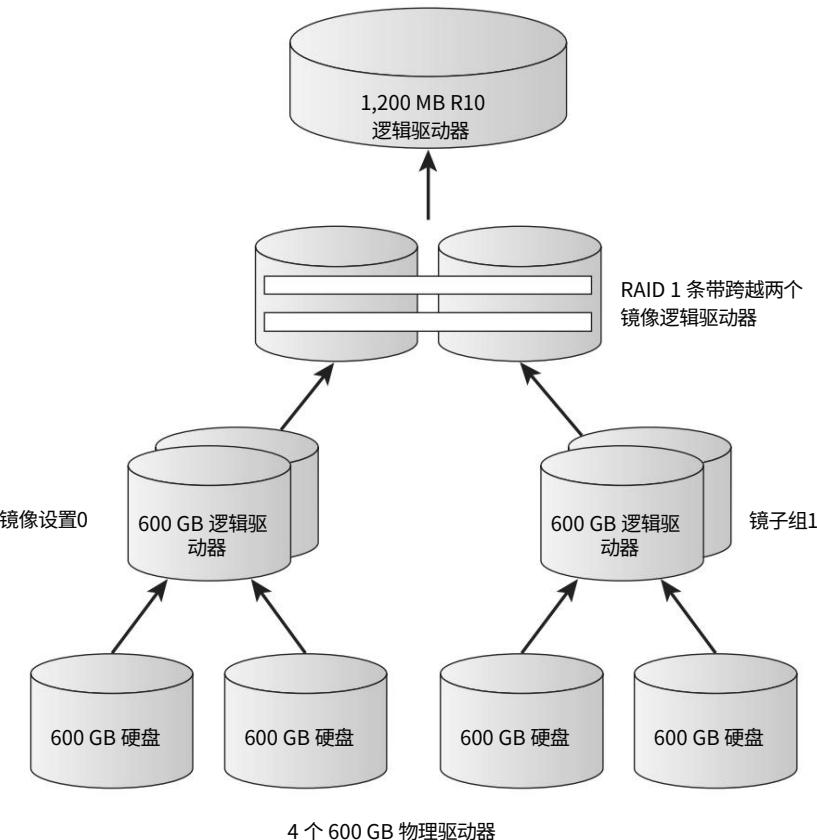
RAID 10 的意义

RAID 10 背后的理念非常简单：获得 RAID 1 的保护和性能，并通过额外的条带化层来加速其性能。条带化会向 RAID 集添加更多驱动器，驱动器 = IOPS 和 MB/秒！

因此，只要您能够承担 RAID 1 的任何衍生产品带来的容量开销，RAID 10 就有可能提供传统 RAID 级别中最佳的性能和保护组合。

图 4.12 以图形方式显示了如何创建 RAID 10 组，练习 4.3 引导我们了解创建 RAID 10 组的高级过程。

图 4.12 创建 RAID 10 组



练习 4.3

创建 RAID 10 RAID 集

您的 RAID 控制器将为您完成此操作,因此这纯粹是概念上的。假设您的服务器/存储阵列中有四个 1 TB 的物理驱动器。要创建 RAID 10 组,请按照以下步骤操作:

1. 使用两个驱动器创建 1 TB 镜像卷。
2. 对其余两个驱动器执行完全相同的操作。在完成这两个步骤后,
您将拥有两个 1 TB 的镜像卷。
3. 使用这两个镜像卷创建一个 2 TB RAID 0 条带卷
跨越它们。

最终结果将是一个受 RAID 1 保护的 2 TB 卷,并且具有所有读写在驱动器上进行条带化的额外性能优势。

警惕 RAID 01

RAID 10 和 RAID 01 并不相同,人们经常会混淆这两者。您总是想要 RAID 10 而不是 RAID 01。技术差异在于 RAID 01 首先创建两个条带集,然后在它们之间创建镜像。RAID 01 的主要问题是它比 RAID 10 更容易丢失数据。一些 RAID 控制器会自动构建 RAID 10 集,即使它们将其称为 RAID 1。

RAID 5

尽管受到批评,RAID 5 可能仍然是最常用的 RAID 级别 -
尽管大多数人认为对所有 1 TB 或更大的驱动器使用 RAID 6 是最佳做法。

从技术上讲,RAID 5 被称为具有分布式奇偶校验的块级条带化。这告诉我们两件事:

块级别告诉我们它不是位或字节级别。块大小是任意的,并映射到我们在讨论条带化时解释的块大小。

分布式奇偶校验告诉我们,RAID 组中没有指定单个驱动器作为奇偶校验驱动器。相反,奇偶校验分布在 RAID 组中所有驱动器上。

RAID 5 成本和开销

即使 RAID 5 组中没有专用的奇偶校验驱动器,RAID 5 也始终会在 RAID 组的所有驱动器上保留相当于一个磁盘的块,以用于奇偶校验。

这样,无论 RAID 5 组中有多少成员,只要有一块驱动器丢失,数据都不会丢失。但是,如果您的 RAID 5 组在重建第一个驱动器之前丢失了第二个驱动器,则 RAID 组将发生故障,您将丢失数据。因此,您需要 RAID 5 组的重建时间要快。

常见的 RAID 5 配置包括以下几种:

- RAID 5 (3+1)
- RAID 5 (7+1)

RAID 符号

让我们快速了解一下常见的 RAID 符号。写出 RAID 集配置时,通常使用以下符号:

突袭 X (D+P)

X 表示 RAID 级别,例如 1.5 或 6。

D 表示每个条带的数据块数量。

P 表示每个条带的奇偶校验块的数量。

因此,如果我们的 RAID 组相当于三个数据驱动器和一个奇偶校验驱动器,这将使其成为 RAID 5 组,我们将其写如下:RAID 5 (3 + 1)。

由于 RAID 5 的规则仅规定它是分布式块级奇偶校验,因此 RAID 5 组可以包含比前面示例多得多的驱动器 例如 RAID 5 (14+1)。但是,尽管 RAID 5 组中的驱动器数量很多,但一如既往,只能容忍单个驱动器发生故障而不会丢失数据。

RAID 5 组中奇偶校验丢失的容量取决于有多少个数据块
每个奇偶校验条带的奇偶校验数。计算方式为奇偶校验/数据 $\times 100$,以百分比表示。

因此,在我们的 RAID 5 (3+1) 组中,我们因奇偶校验而损失了 25% 的容量,因为 $1/4 \times 100 = 25$ 。
RAID 5 (7+1) 组的空间效率更高,因为它仅牺牲总空间的 12.5% 用于奇偶校验: $1/8 \times 100 = 12.5$ 。而在极端情况下,RAID 5 (15+1) 仅牺牲 6.25% 的空间用于奇偶校验。

不过要小心。虽然 RAID 5 (15+1) 比 RAID 5 (3+1) 提供了更多的可用空间,但也存在两个缺点:

- 重建时间更长。15+1 RAID 组的重建时间将比 3+1 组更长,因为 XOR 计算需要更长的时间,而且重建数据时需要读取更多的驱动器。
- 性能较低,因为创建 FSW 和避免 RAID 5 写入损失更加困难。我们将很快讨论这个问题。

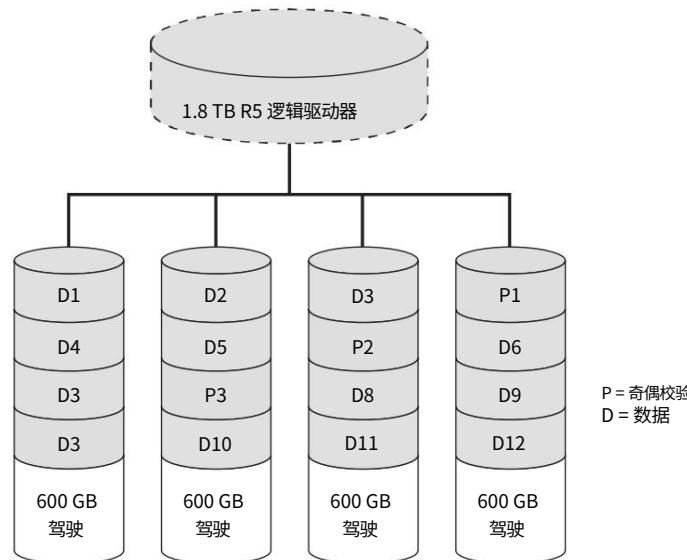
当 RAID 5 组中的某个驱动器发生故障时会发生什么情况

当 RAID 5 组中的某个驱动器发生故障时,您不会丢失任何数据。但是,您需要尽快重建 RAID 组,因为当 RAID 组中有一个驱动器发生故障时,它会以降级模式运行。RAID 组的性能及其承受驱动器故障的能力会降低。因此,尽快退出降级模式是个好主意。

假设有可用的热备用驱动器,RAID 控制器可以立即开始重建该组。这是通过对组中的幸存数据执行异或 (XOR) 运算来实现的。XOR 运算有效地告诉 RAID 控制器故障驱动器上的内容,从而使热备用驱动器能够用故障驱动器上的内容进行填充。重建完成后,RAID 组不再处于降级模式,您可以放心。

图 4.13 显示的是 RAID 5 (3+1) 阵列。

图 4.13 具有旋转奇偶校验的 RAID 5 (3+1) 阵列



RAID 5 写入惩罚

基于奇偶校验的 RAID 方案 (例如 RAID 5 和 RAID 6) 存在固有的性能缺陷,称为写入惩罚。这在写入活动较多 (尤其是随机写入活动较多) 的情况下尤其成问题。发生这种情况的原因是,计算小块写入的奇偶校验会导致后端产生以下额外 I/O。对 RAID 5 组的小块随机写入每次写入都会产生四个 I/O:

1. 读取旧数据。
2. 计算新数据 (在内存中)。

3.写入新数据。

4.计算并写入新的奇偶校验。

让我们看一个简单的例子。图 4.13 显示了一个简单的 RAID 5 (3+1) RAID 集。

假设我们只想写入第 0 行上的块 D1。要计算第 0 行的新奇偶校验,RAID 控制器必须读取现有数据,读取现有奇偶校验,将新数据写入 D1,并将新奇偶校验写入 P1。这种开销非常麻烦,如果对 RAID 集进行大量小块写入,可能会降低其性能。出于这个原因,许多应用程序所有者,尤其是 DBA,都不喜欢 RAID 5。但是,许多 RAID 5 实现在尝试缓解此问题方面大有作为。

全条带写入不会受到写入惩罚

由于写入惩罚仅适用于小块写入,其中写入大小小于 RAID 集的条带大小,因此执行较大的写入可避免此问题。这可以通过缓存实现。

通过在 RAID 控制器前面放置一个足够大的缓存,RAID 控制器能够存储小写入并将它们捆绑在一起形成更大的写入。这是一种称为写入合并的技术。目的是始终写入完整的条带 - 因为写入完整的条带不需要额外的操作来读取您未写入的块,并且您可以在单个操作中写入数据和奇偶校验。

然而,写入合并是高端外部 RAID 控制器 (存储阵列) 的一项功能,是一项先进的技术。它通常不是服务器中硬件 RAID 控制器的功能,当然也不是软件 RAID 控制器使用的技术。如果 RAID 控制器承受特别高的写入工作负载,写入合并的有效性也会降低。

分布式 RAID 方案可能会限制写入惩罚的影响

分布式 RAID 是指将受 RAID 保护的卷分布在数百个驱动器上的能力,其中每个连续的条带存在于一组单独的驱动器上。

图 4.14 显示了传统的四驱动器 RAID 组。在该图中,四个连续的显示逻辑卷中的条带。

图 4.14 具有四个连续条带的传统 RAID 组

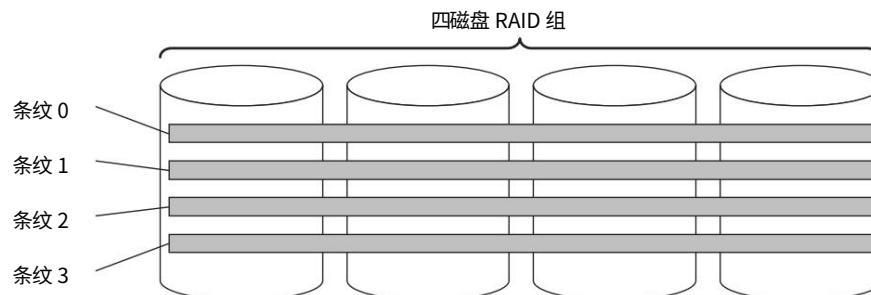


图 4.15 显示了同一逻辑卷的相同四个条带,但这次分布涵盖 16 驱动器池中的所有 16 个驱动器。

图 4.15 具有四个连续条带的并行 RAID 配置



虽然图 4.15 中的配置不能避免写入损失,但它可以通过并行使用所有 16 个驱动器并可能并行写入多个条带来抵消部分影响。



The write parity rate differs only in parity-based RAID systems and does not affect RAID 1 or RAID 5. RAID 1写入工作负载时,这种损失最为明显。

RAID 6

RAID 6 与 RAID 5 类似。两者都使用分布式奇偶校验进行块级条带化。但 RAID 6 不是使用单奇偶校验,而是使用双奇偶校验。使用双奇偶校验有明显的优缺点。



RAID 6 基于多项式奇偶校验码。您将很快理解其重要性,尤其是在基于云的章节中,因为它们可能会在未来对 RAID 和数据保护产生重大影响。

RAID 6 成本和开销

与 RAID 5 一样,RAID 6 组中没有专用的奇偶校验驱动器。奇偶校验分布在 RAID 组的所有成员中。但是,RAID 5 始终保留相当于一个驱动器的块用于奇偶校验,而 RAID 6 保留两个驱动器的块用于两个独立的奇偶校验组。这使得 RAID 6 组无论

RAID 6 是一组非常安全的 RAID 级别！

常见的 RAID 6 配置包括以下几种：

- RAID 6 (6+2)
- RAID 6 (14+2)

与 RAID 5 一样，RAID 6 组可以有任意数量的成员，但它始终具有两组离散奇偶校验！因此，只要您的 RAID 控制器支持它并且您在该组中有足够的驱动器，RAID 6 (30+2) 就是有效的 RAID 6 配置。

RAID 6 组中 RAID 开销造成的容量损失遵循相同规则。为 RAID 5。将 2 除以组中的驱动器总数，然后将该数字乘以 100，如下所示：

■ RAID 6 (14+2): $2/16 \times 100 = 12.5\%$ 的容量开销。有时也称为 87.5% 的效率。

■ RAID 6 (30+2): $2/32 \times 100 = 6.25\%$ 的容量开销，或 93.75% 的效率。

与 RAID 5 一样，RAID 组中的驱动器越多，重建速度越慢，写入性能越低。请记住，基于奇偶校验的 RAID 方案在执行小块写入时会遭受写入损失。这些可以通过使用缓存合并写入并执行全条带写入来抵消。但是，RAID 组越大（实际上条带大小越大），合并足够的写入以形成全条带写入就越困难。

何时使用 RAID 6

从积极的一面来看，RAID 6 被认为是极其安全的。它可以承受两个驱动器故障而不会丢失数据。这使得它成为大型驱动器或大量驱动器池的理想选择。

大型驱动器往往是速度较慢的驱动器（以每分钟转数 (RPM) 计算），因此重建所需的时间比小型、较快的驱动器要长得多。因此，在执行重建时，发生第二个驱动器故障的可能性大大增加。RAID 6 为您提供了额外的安全网，允许您承受第二个驱动器故障而不会丢失数据。此外，如果您部署了包含数百个驱动器的池，如果您丢失了组成该池的 RAID 集，则可能会在使用该池的每个卷中造成漏洞。这意味着使用该池的每个卷上的数据都会丢失！因此，对于这两种用例，RAID 6 是事实上的选择。

RAID 6 在高读取情况下表现良好。

何时不使用 RAID 6

RAID 6 的主要缺点是性能，尤其是在写入次数较多的环境中小块写入的性能。由于 RAID 6 将每个 RAID 条带的奇偶校验量增加了一倍，因此 RAID 6 受到的写入损失比 RAID 5 更严重。RAID 5 组的每个小块写入需要 4 个 I/O，而 RAID 6 组的每个写入需要 6 个 I/O。因此，不建议将 RAID 6 用于小块密集型 I/O 要求，并且在绝大多数情况下，它不适用于任何基于写入的繁重工作负载。

RAID 6 的另一个缺点是可用容量。这是因为 RAID 6 每个条带有两个奇偶校验块。但是,可以通过在大型 RAID 集中部署 RAID 6 (例如 RAID 6 (6+2) 或非常常见的 RAID 6 (14+2)) 来弥补这一缺点。这两个组分别损失 25% 和 12.5%。不过,不要忘记,条带大小越大,RAID 控制器就越难在缓存中合并全条带写入的写入。

总而言之,RAID 6 被视为一种缓慢但安全的 RAID 选择。对于包含 1 TB 以上驱动器的 RAID 组,它几乎总是首选的 RAID 级别。



Real World Scenario

RAID 6:一双安全的双手

一家公司的单个存储阵列在不到一周的时间内就出现了 30 多个驱动器故障,原因是某家供应商的某批磁盘驱动器存在错误。但是,该公司正在使用大型驱动器池,因此部署了 RAID 6。尽管在如此短的时间内丢失了这么多驱动器,但组成该池的单个 RAID 组中没有发生三驱动器故障,因此没有数据丢失!

不太常用的 RAID 级别

有些 RAID 级别并不像前面讨论的 RAID 级别那样常用。即便如此,了解这些 RAID 级别对您来说也很重要。这里简要介绍了 RAID 2、RAID 3 和 RAID 4。

RAID 2

RAID 2 是位级条带化,是唯一能够从单比特错误中恢复的 RAID。由于它在现实世界中很少使用,所以您只需要了解这些内容。

RAID 3

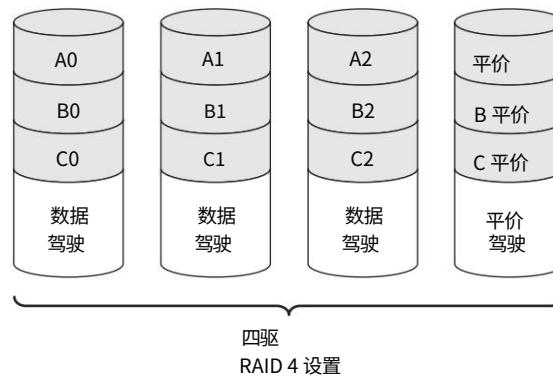
RAID 3 在字节级别执行奇偶校验,并使用专用奇偶校验驱动器。RAID 3 在现实世界中极为罕见。

RAID 4

RAID 4 与 RAID 5 类似。它执行块级条带化,但具有专用的奇偶校验驱动器。

使用专用奇偶校验驱动器的一个常见问题是奇偶校验驱动器可能成为写入性能的瓶颈。图 4.16 显示了带有专用奇偶校验驱动器的四磁盘 RAID 4 组。在此 RAID 配置中,每次更新每行上的单个块时,都必须更新专用奇偶校验驱动器上的奇偶校验。这可能会导致奇偶校验驱动器成为瓶颈。

图 4.16 RAID 4 奇偶校验驱动器成为瓶颈



但是,它的优点是可以通过向 RAID 组添加更多磁盘并仅需重建奇偶校验驱动器来轻松扩展 RAID 组。其他具有分布式奇偶校验的 RAID 方案在添加驱动器时需要重建整个 RAID 组。

RAID 4 比 RAID 2 和 RAID 3 更常见,但远不如 RAID 5 和 RAID 6。

NetApp 的 Data ONTAP 支持 RAID 4,并将其与同样属于 Data ONTAP 的随处写入文件布局 (WAFL) 文件系统紧密集成。通过集成文件系统和 RAID,只要系统容量接近上限并达到极限,Data ONTAP 就能够合并写入并避免专用奇偶校验驱动器的性能影响 (奇偶校验驱动器受到的冲击比 RAID 组中的其他驱动器更大)。

集成文件系统和 RAID 方案可以带来有趣的好处,这些好处将在本章末尾讨论 ZFS 时介绍。

RAID 的未来

RAID 技术已经存在了几十年。可以说,虽然技术领域的大多数事物在过去二十年中取得了巨大进步,但 RAID 的大多数实现却没有取得任何进展。此外,磁盘驱动器膨胀问题 (驱动器越来越大但速度却没有加快)导致重建时间越来越长。而重建时间越长,意味着您将在更长的时间内面临更多驱动器故障的风险。这导致了我们现在所处的境地,即传统的 RAID 方法无法应对当今业务和 IT 需求的挑战。

幸运的是,一些有趣且重要的新 RAID 技巧和技术开始出现。其中一些最受欢迎的包括那些实现并行或分布式 RAID 形式的技术。我们将介绍其中一些新方法和技术,重点介绍其中的一些进展。



在回顾这些现代 RAID 方法时,我们将参考一些特定于供应商的技术。这样做的目的不是推广或谴责任何特定技术。我们的兴趣纯粹是从学术角度研究它们与传统方法的不同之处以及它们如何尝试满足当今的需求。

IBM XIV 和 RAID-X

IBM XIV 存储阵列实现了一种称为 RAID-X 的 RAID 1 技术。

我们对 RAID-X 感兴趣的原因在于它不是你祖母认识的任何类型的 RAID 1。是的,它会因保护而损失 50% 的容量,是的,它容易受到双驱动器故障的影响。但这是 21 世纪的涡轮增压、基于对象的分布式 RAID。因此,我们有很多很酷的东西可以研究。

基于对象的 RAID

基于对象意味着 RAID 保护基于对象而不是整个驱动器。

XIV 的基于对象的 RAID 实现基于 1 MB 区段,即分区。XIV 上的每个卷由多个 1 MB 区段组成,这些 1 MB 区段受到 RAID-X 保护。

但是,由于 XIV 上的所有卷都是原生精简配置的,因此在将数据写入卷时,会根据需要将扩展区分配给卷。有趣的是,当驱动器发生故障时,这会改善重新保护和重建操作,我们将很快介绍这一点。

分布式 RAID

RAID-X 的基于对象的特性使组成 RAID-X 卷的 1 MB 区段能够分布在 XIV 阵列的整个后端——数百个驱动器上。这种跨整个后端的宽条带化卷可实现大规模并行读取、写入和重新保护操作。这与您之前看到的双驱动器 RAID 1 配置完全不同!

RAID-X 和重新保护

RAID-X 容易受到双驱动器故障的影响。如果 XIV 中的一个驱动器发生故障,然后在第一个驱动器的重新保护操作完成之前第二个驱动器发生故障,则系统将丢失数据。而且由于每个 XIV 卷几乎分布在后端的每个驱动器上,双驱动器故障的影响范围非常大。它可能会给系统的每个卷都打出一个小洞——哦,天哪!然而,这很重要,RAID-X 经过高度优化,可以快速重新保护自身!它通过大规模并行重新保护操作来实现这一点。让我们来看看。



对于传统的基于驱动器的 RAID 方法，当驱动器发生故障并且 RAID 组进入降级模式时，恢复和修理恢复到完全受保护模式的唯一方法是将故障驱动器的内容重建到另一个驱动器（通常是热备用驱动器）。在多 TB 驱动器领域，驱动器重建操作可能需要大量时间。在某些阵列上，重建大型驱动器需要数天时间。因此，建议对大型、慢速驱动器使用 RAID 6。传统的基于驱动器的 RAID 也会重建整个驱动器，即使驱动器上只有少量数据。RAID-X 总体上更加智能。



建议对大型驱动器进行双重保护的另一个原因是，在执行重建时遇到不可恢复的读取错误的可能性大大增加。如果您对大型驱动器只有单层保护，并且在重建期间遇到不可恢复的读取错误，这实际上与第二个驱动器故障相同，您将丢失数据。因此，随着驱动器越来越大，强烈建议至少进行双重保护。

RAID-X 大规模并行重建

当 XIV 中的某个驱动器发生故障时，RAID-X 不会尝试重建发生故障的驱动器，至少目前不会。相反，它会花费精力重新保护受影响的数据。这是很重要的。

假设我们有一个 2 TB 的驱动器发生故障，目前使用率为 50%。当此驱动器发生故障时，它将留下 1 TB 的数据，这些数据分布在现在不受保护的后端的其余部分。这个不受保护的 1 TB 仍然完好无损，但每个 1,024 个 1 MB 扩展区在阵列的其他地方都不再有镜像副本。而且由于这 1 TB 的未受保护数据分布在后端的所有驱动器上，任何其他驱动器发生故障都会导致这些未受保护的 1 MB 扩展区中的一些丢失。因此，为了尽快重新保护这些 1 MB 扩展区，XIV 会从后端的所有驱动器读取每个扩展区，并将它们的新镜像副本写入分布在整个后端的其他驱动器 - 所有这些都是并行的。当然，RAID-X 确保任何 1 MB 扩展区的主副本和镜像副本永远不会位于同一驱动器上，并且它进一步确保它们也位于不同的节点/机架上。

这是大规模并行多对多操作的绝佳示例，只需几分钟即可完成。XIV 上的驱动器重建操作很少持续超过 20-30 分钟，而且通常完成速度要快得多。重新保护操作的速度对于如此大规模的 RAID-X 的可行性至关重要，因为它大大减少了第二个驱动器同时发生故障的脆弱性窗口。

此外，与传统 RAID 不同，RAID-X 重建的并行特性意味着它避免了在正常 RAID 重建操作期间通常对 RAID 集中的所有磁盘施加的增加的负载压力。在大多数 XIV 配置中，每个驱动器只承担约 1% 的重建重担。

RAID-X 和基于对象的重建

此外,由于 RAID-X 是基于对象的,因此它仅重新保护实际数据。在我们的示例中,一个 2 TB 驱动器已满 50%,只有 1 TB 的数据需要重新保护和重建。

非基于对象的 RAID 方案通常会浪费时间重建没有数据的磁道。通过仅重新保护数据,RAID-X 可加快重新保护操作。

一旦数据得到重新保护,RAID-X 将继续重建故障驱动器。这重建是多对一的操作,其中新驱动器是瓶颈。

IBM XIV 上的 RAID-X 并不是现代基于对象的并行的唯一实现 RAID,但它是一个众所周知且备受争议的例子。

关于基于对象的重建,其他存储阵列(如 HP 3PAR、Dell Compellent 和 XIO)也采用了这种方法。但是,这些技术使用的是具有奇偶校验和双奇偶校验的基于对象的 RAID,而不是 XIV 的基于对象的镜像形式。由于采用基于对象的 RAID 方法的技术都是现代阵列,因此这显然是 RAID 技术的下一步。



RAID-X 以及几乎所有基于池的后端设计都会增加发生双驱动器故障时的潜在爆炸区。由于所有卷都分布在后端的所有 n (实际上是大多数驱动器上),因此双驱动器故障可能会在大多数 (如果不是所有) 卷中造成一个小漏洞。

这是需要注意的,也是大多数传统阵列在部署大型池时推荐使用 RAID 6 的主要原因。这样做的目的是降低风险,而 RAID-X 通过其超快的大规模并行重新保护操作来实现这一点。

ZFS 和 RAID-Z

RAID-Z 是一种基于奇偶校验的 RAID 方案,与 ZFS 文件系统紧密集成。

它提供单、双和三重奇偶校验选项,并使用动态条带宽度。这种动态、可变大小的条带宽度非常强大,可以有效地使每个 RAID-Z 写入成为全条带写入。除通常镜像的小写入外,从性能角度来看,这非常棒,这意味着 RAID-Z 很少(如果有的话)遭受传统基于奇偶校验的 RAID 方案在执行小块写入时所遭受的读取-修改-写入损失。

RAID-Z 卷能够通过将可变条带宽度与 ZFS 文件系统的重定向写入特性相结合,对所有写入执行全条带写入。重定向写入

意味着数据永远不会就地更新,即使您正在更新现有数据。相反,数据总是被写入一个新的位置(作为可变大小的全条带写入),并且指针表会更新以反映更新数据的新位置,旧数据将被标记为不再使用。这种操作模式可带来出色的性能,但当可用空间开始变低时,性能会开始显著下降。

使用 RAID-Z 恢复

RAID-Z 重建比典型的基于奇偶校验的重建要复杂得多，后者对每个 RAID 条带执行简单的 XOR 计算。由于 RAID-Z 条带的大小可变，RAID-Z 需要查询 ZFS 文件系统以获取有关 RAID-Z 布局的信息。如果池接近容量上限或繁忙，这可能会导致更长的重建时间。从积极的一面来看，由于 RAID-Z 和 ZFS 文件系统相互通信，因此重建操作只会重建实际数据，而不会浪费时间重建空块。

此外，由于 RAID-Z 和 ZFS 的集成，数据可以免受静默损坏。每次读取数据时，都会通过校验和（上次更新数据时创建的校验和），如果验证失败，则重建校验和。这是没有文件系统知识的基于块的 RAID 方案无法做到的。

磁盘阵列

RAID-TM 是基于三重镜像的 RAID。RAID-TM 不会像 RAID 1 镜像集那样保留两份数据，而是保留三份数据。因此，它会损失 $2/3$ 的容量来提供保护，但如果您能克服最初只能使用所购容量的 $1/3$ 的震惊，它将提供良好的性能和出色的保护。

RAID 7

RAID 7 背后的想法是将 RAID 6 更进一步，添加第三组奇偶校验。乍一看，这似乎是一件令人厌恶的事情，但它的出现是出于保护越来越大的驱动器上数据的真正需要。例如，不久之后，驱动器就会变得如此之大、如此之慢，以至于从头到尾读取它们需要一整天的时间，更不用说从奇偶校验重建它们了！而且正如我们一直强调的那样，更长的重建时间会导致更大的暴露窗口，以及在重建期间遇到不可恢复的读取错误的可能性。

雨

RAIN 是冗余/可靠廉价节点阵列的缩写。它是一种基于网络的容错形式，其中节点是保护的基本单位，而不是驱动器或盘区。基于 RAIN 的容错方法在横向扩展文件系统、基于对象的存储和基于云的技术中越来越受欢迎。基于 RAIN 的容错方法在其他章节中介绍，我们讨论横向扩展架构、对象存储和云存储。

纠删码

纠删码是另一项开始强势进入存储领域的技术。更恰当的说法是前向纠错 (FEC)，有时也称为信息

分散算法 (IDA),这项技术已经存在很长时间,并广泛应用于电信行业。此外,擦除码已用于 RAID 6 技术以及一些消费电子技术,如 CD 和 DVD。让我们面对现实吧,任何在我的孩子接触过 CD 后仍能播放出声音的数据保护和恢复技术肯定可以为企业存储领域提供一些帮助!

虽然奇偶校验方案往往在磁盘驱动器级别运行良好,但擦除码最初在节点级别似乎是一个更好的选择,可以跨节点而不是驱动器保护数据。它们可以 (并且已经) 用于保护地理上分散的多个节点上的数据。这使它们成为现代云规模技术的理想选择。

纠删码的工作原理也与奇偶校验略有不同。虽然奇偶校验将奇偶校验与纠删码区分开来,但纠删码扩展了数据块,使其既包含实际数据,又包含纠删码。与 RAID 类似,纠删码提供不同级别的保护,每种保护级别在保护、性能和可用容量之间都有类似的权衡。例如,使用纠删码技术可以提供以下保护级别:

■ 9/11

■ 6/12

在第一个例子中,只要 11 个数据块中剩余 9 个,就可以重建数据,而 6/12 则只要 12 个数据块中剩余 6 个,就可以重建数据。6/12 在可用容量方面显然与 RAID 1 相似。

虽然擦除码是 RAID 6 的基础,并且正在进入大容量存储的备份和存档使用案例,但它们很可能会在企业存储领域,特别是基于云的解决方案中越来越成为主存储保护不可或缺的一部分。

最后,目前的纠删码实现在以下方面表现不佳:
小规模写入,这意味着如果擦除码确实成为未来数据保护的基础,那么我们很可能会看到它们用于云设计中,也可能用于大型横向扩展文件系统,而更传统的 RAID 方法将继续保护小规模系统和产生小规模写入的系统,例如数据库。

章节概要

RAID 的重要性 RAID 技术对于全球数据中心的平稳运行仍然至关重要。RAID 可以保护您的数据免受驱动器故障的影响并提高性能。

硬件或软件 RAID 如果经济条件允许,大多数人会选择硬件 RAID。它可以提供更高的性能、更快的重建和热备份,并可以保护操作系统启动卷。然而,软件 RAID 往往更灵活,更便宜。

条带大小和性能 大多数情况下,RAID 控制器上的默认条带大小就足够了 而且存储阵列通常不允许您更改它。但是,如果您知道您的工作负载主要是小块,则应选择较大的条带大小。如果您知道您的 I/O 大小会很大,则应选择较小的条带大小。

RAID 1 镜像 RAID 1 性能良好,但 RAID 容量开销为 50%。对于小块随机工作负载,它是一个很好的选择,并且不会受到写入损失的影响。

RAID 5 RAID 5 是块级交错奇偶校验,其中每个条带的单个奇偶校验块在 RAID 集的所有驱动器之间轮换。RAID 5 可以容忍单个驱动器故障,并且在执行小块写入时会遭受写入损失。

RAID 6 RAID 6 是块级交错奇偶校验,其中每个条带的两个离散奇偶校验块在 RAID 集的所有驱动器之间轮换。RAID 6 可以容忍两个驱动器故障,并且在执行小块写入时会遭受写入损失。

概括

在本章中,我们介绍了目前使用的所有常见 RAID 级别,包括 CompTIA Storage+ 考试所需的级别。作为我们介绍的一部分,我们研究了每个 RAID 级别提供的各种保护和性能特征,以及每个级别的成本和开销。我们还解决了传统 RAID 级别在现代 IT 世界中面临的一些挑战,并提到了 RAID 的一些潜在未来。

章节

5



光纤通道 SAN

本章涵盖的主题：

- 1 什么是光纤通道 SAN
- 1 HBA 和 CNA
- 1 交换机和控制器
- 1 布线
- 1 结构服务
- 1 分区
- 1 SAN 拓扑
- 1 FC 路由和虚拟结构
- 1 流量控制
- 1 故障排除



本章深入介绍构成光纤通道 SAN 的重要技术。它涵盖理论和实际应用，并分享在现实世界中对您有帮助的技巧。您将学习有关冗余、布线、交换机、端口和 HBA 配置的基础知识，以及交换机间链路、虚拟结构和

VSAN 间路由等更高级的主题。

在本章结束时，你将了解部署高可用性系统所需的所有理论。功能强大且性能卓越的 FC SAN，以及一些需要避免的常见陷阱。您还将掌握通过 FC SAN 考试部分所需的知识。

什么是 FC SAN？

首先，让我们先了解一下术语。FC SAN 是光纤通道存储区域网络的缩写。大多数人将其缩写为 SAN。

从最高层次来看，FC SAN 是一种存储网络技术，它允许通过专用高速光纤通道 (FC) 网络共享块存储资源。更详细地说，光纤通道协议 (FCP) 是光纤通道网络上的 SCSI 协议的映射。因此，实际上，SCSI 命令和数据块被包装在 FC 帧中并通过 FC 网络传送。本章将对此进行更多介绍。



在光纤通道的上下文中，“通道”这个词很重要，这是因为 SCSI 是一种通道技术！基本通道的技术与普通网络技术略有不同。通道的目标是高性能和低开销。通道技术的可扩展性也往往远低于网络技术。FC 是将信道技术联网的第一次尝试，试图在兼顾两方面的优势。FC 是一个试图联网通道技术——旨在兼顾两者优势的可扩展架构。FC 是一个试图联网通道技术——旨在兼顾两者优势的可扩展架构。



FC 协议可以在光纤或铜缆上运行。约 99% 的时间都是在光纤电缆上运行。FC 网络中的铜缆通常仅限于黑盒配置或存储阵列后端的电缆。Copper cabling in FC networks is usually restricted to black-box configurations or the cabling on the back-end of a storage array.



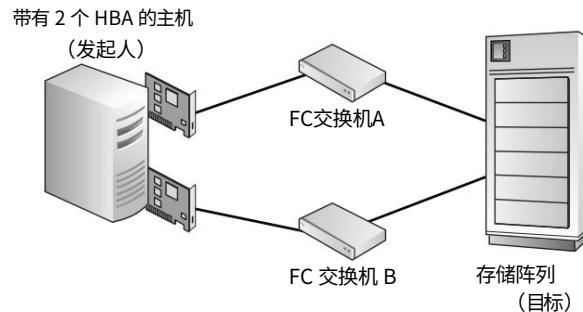
最后再提醒一下,以免让您感到困惑,当提到光纤通道协议或光纤通道 SAN 时,fibre 通常拼写末尾会带有ore。当提到光纤 SAN 时,通常拼写末尾会带有here at the end. When

由于 FCP 封装了 SCSI,因此从技术上来说,可以通过 FC SAN。然而,FC SAN 上共享的设备 99.9% 都是磁盘存储设备,或者磁带驱动器和磁带库。

FC SAN 上共享的设备与所有 SCSI 设备一样,都是块设备。块设备是实际上,这些设备实际上是原始设备,在主机操作系统和虚拟机管理程序看来,它们实际上就像是本地连接的设备。此外,它们没有应用任何更高级别的抽象,例如文件系统。这意味着在 FC SAN 环境中,文件系统的添加是访问块存储设备的主机的责任。

图 5.1 显示了一个简单的 FC SAN。

图 5.1 简单 FC SAN 高级视图



如图 5.1 所示,以下是 FC SAN 中的一些主要组件:

- **发起者** (通常简称为**主机**或**服务器**)
- **目标** (**磁盘阵列**或**磁带库**)
- **光纤通道交换机**

图 5.1 中的启动器和目标是 SCSI 概念,可以视为客户端和服务器,就像客户端-服务器网络模型中一样。发起者与客户端同义,而目标与服务器同义。发起者通常通过 FC SAN 向目标发出读写命令。

启动器通常是主机,或者更准确地说,它们是主机中的 HBA 或 CNA 端口。目标通常是存储阵列或磁带库中的 HBA 或 CNA 端口。而 SAN 是一个或多个 FC 交换机,用于在启动器和目标之间提供连接。
简单的。



HBA 是主机总线适配器的缩写，CNA 是融合网络适配器的缩写。两者都是 SAN 设备，可以作为扩展中安装在服务器中，也可以直接安装在服务器 PCIe 总线上。这些 HBA 和 CNA 的驱动程序使它们在虚拟机管理程序的主机操作系统中显示为本地 SCSI 卡。

这意味着通过 HBA 或 CNA 通过 SAN 呈现给操作系统或虚拟机管理程序的任何设备在操作系统或虚拟机管理程序看来都像是本地连接的设备。SAN 上的存储阵列和磁带设备也安装了 HBA 或 CNA。

为什么选择 FC SAN？

在服务器内部部署直接连接存储 (DAS) 的传统方法有优点也有缺点。优点之一是 DAS 存储可通过非常短、专用、低延迟、无争用互连 (SCSI 总线) 进行访问。这可确保快速可靠地访问设备。然而，DAS 的一些缺点包括管理增加、利用率低的容量孤岛以及磁盘驱动器等设备数量有限。

FC SAN 技术背后的理念是保留 DAS 存储的优点，同时克服其缺点。因此，SAN 技术提供了一种高速、低延迟的存储网络（通道），该网络相对无争用，尤其是与传统以太网相比。SAN 还允许将大量驱动器汇集并在多个主机之间共享，从而摆脱容量孤岛。它们还通过减少管理点数量来简化管理。虽然 FC SAN 需要花费一些成本 (\$)，但它们已经为数据中心提供了多年的良好服务。然而，它们正面临威胁。

一些重要的新技术的出现正在威胁霸权
SAN 在企业 IT 中的应用。这些技术包括：

■ 数据中心桥接 (DCB) 以太网 有时称为融合增强
以太网 (CEE)

云技术

■ 本地连接闪存

云存储和云一切正在挤进许多 IT 环境，通常从底层开始，然后逐渐向上层。常见的例子包括将测试和开发环境迁移到云中，这通常会导致暂存和一些生产要求最终进入云中。在顶部，业务关键型业务线应用程序（由于其高性能、低延迟要求，它们对云的免疫力更强）受益于本地连接的闪存存储（将闪存放置在服务器的 PCIe 总线上，使其靠近主机 CPU 和内存）。这样就只剩下其余的生产环境，它们太重要了，不能信任云，但不需要超低延迟。但即使是这种用例也处于

FCoE 的压力,它将 FCP 映射到 10 GB DCB 以太网上。总而言之,FC SAN 承受着极大的压力,这是理所应当的!但它不会在一夜之间或毫无征兆地消失。一套良好的 FC 技能将在未来在 FCoE 环境中为您提供良好的服务。

接下来我们来看一下构成FC SAN的一些主要技术组件。

FC SAN 组件

FC SAN 由几个关键组件组成,一些是物理组件,一些是逻辑组件。在本节中,您将更详细地了解以下内容:

- 主机总线适配器和融合网络适配器
- FC 交换机和导向器
- FC 存储阵列
- FC 布线
- FC 面料
- FC 简单名称服务器
- 分区
- FC 寻址
- FC 服务等级
- VSAN 技术

物理组件

从高层次上看,SAN 由终端设备和交换机组成。终端设备通过交换机进行通信。

我们通常认为终端设备 (有时称为节点端口或 N_Port) 是服务器和存储阵列。但从技术上讲,终端设备是安装在服务器、存储阵列和磁带设备上的 HBA 和 CNA 上的端口。终端设备通过电缆连接到交换机。遗憾的是,数据中心网络技术尚未实现无线化。

FC SAN 使用光纤电缆。多个交换机可以连接在一起以形成更大的网络结构。



本节通常将 FC SAN 称为 SAN。这并不意味着 SAN 只是指一个 FC SAN,而是指所有 SAN,包括 FC SAN。

无论如何,这些都是基础知识。让我们仔细看看 FC SAN 的每个组件。

HBA 和 CNA

主机和服务器通过安装在主机 PCIe 总线上的一个或多个光纤通道主机总线适配器 (HBA) 或融合网络适配器 (CNA) 连接到 SAN。按照 SCSI 的说法,这些是 SAN 上的启动器。

如前所述,HBA 和 CNA 可以作为 PCIe 扩展实现
卡上或直接在服务器主板上 - 主板上的局域网 (LOM)。
每个 HBA 和 CNA 都有自己的专用硬件资源,用于减轻主机 CPU 和内存的所有 FCP 相关开销。这种减轻是 FC SAN
成为快速、低开销但相对昂贵的存储网络技术的一个关键原因。

此外,大多数 HBA 和 CNA 都带有 BIOS,这意味着安装它们的服务器可以从 SAN 启动。



从技术上讲,CNA 是多协议适配器,可为多种协议 (包括 FCoE 和 iSCSI) 提供硬件卸载。这意味着
它们可用于高性能 SAN,如 iSCSI SAN 和以太网网络。这种灵活性为您
的环境提供了面向未来的元素;今天您可能希望您的 CNA 作为 iSCSI 启动器运行,但六个月后,如果服务器重建,
您可能需要将 CNA 配置为 FC 启动器。但是,正如您所料,这种灵活性和面向未来的特性需要付出
代价才能体现其便利性!然而,在现实世界中,CNA 几乎只用作 FCoE 适配器。iSCSI 卸载很少
(如果有的话)实施和使用。

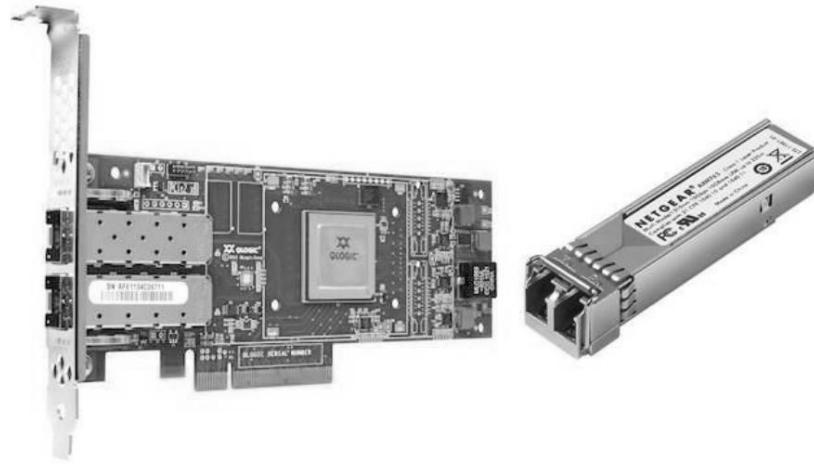
对于虚拟机管理程序和操作系统而言,HBA 和 CNA 在 PCI 设备树上显示为 SCSI 适配器,提供对标准原始 SCSI
设备 (称为 LUN) 的访问。操作系统和虚拟机管理程序不知道它们正在访问的存储设备是通过共享网络访问的。



术语 “LUN” 是从 SCSI 的固有指代的是逻辑单元号 LUN, referring to logical unit number.
先熟悉 *scanning* 为使 SCSI 总线上 (或 SAN) 上的设备能够被寻址, 它们需要 SCSI LUNs
术语 LUN 和卷
经常互换使用。因此,我们有时可能将 LUN 称为卷,反之亦然。

HBA 和 CNA 通常支持铜质直连电缆 (DAC) 或光纤收发器。购买 HBA 或 CNA 时,通常必须指定需要哪种。
一般而言,光纤成本更高,但支持更长的距离、需要的功率更少,而且更受欢迎。这些收发器通常是 SFP 或 SFP+ 模块。
SFP 代表小型可插拔模块,而 SFP+ 是 SFP 的增强版本,支持更高的数据速率,包括 10 GB 以太网。这两种类型的收
发器都是热插拔的。图 5.2 显示了带有光纤收发器和 SFP 光纤的 HBA。

图 5.2 HBA 和 SFP 光学器件



FC 交换机和导向器

FC 交换机和 FC 导向器本质上是同一件事物 - 包含多个物理端口并支持 FC 协议的物理网络交换机。

交换机和导向器为主机和存储等终端设备提供连接。

FC 交换机是下一代网络的组成部分。它们在 FC-0、FC-1 和 FC-2 层上运行，并在通信终端设备之间提供全带宽。此外，它们还提供各种结构服务，以简化管理并实现可扩展性。此外，如果多个 FC 交换机正确联网在一起，它们将合并并形成一个通用结构。

控制器和交换机之间的区别只是惯例，所以不要纠结

赶快行动起来！通常，大型交换机（通常有 128 个或更多端口）被称为导向器，而端口数较少的交换机则被称为交换机、部门交换机或工作组交换机。老实说，现在大多数人只使用交换机这个术语。

尽管如此，导向器具有更多的高可用性 (HA) 功能和更多的内置冗余比小型工作组型交换机更灵活。例如，导向器交换机有两个以主动/被动模式运行的控制处理器卡。如果主动控制处理器发生故障，备用控制处理器将接管控制并维持服务。这种冗余控制处理器模型还允许无中断固件更新。小型工作组交换机没有这种级别的冗余。

导向器往往是基于刀片的架构，允许您选择要用哪种刀片类型填充导向器底盘。这些刀片具有不同的端口数（例如 48 端口或 64 端口），并提供不同的服务（例如 16 Gbps FC 交换、支持 10 GB FCoE 的 DCB 以太网网络、加密功能等）。

在开始对刀片进行任何更改之前，您需要确定导向器交换机中的哪些插槽已安装，以及每个插槽安装哪种刀片。练习 5.1 显示了如何轻松完成此操作。

练习 5.1

在 Brocade 交换机中使用 hashow 和 slotshow

本练习将指导您检查导向器交换机中的插槽数量和刀片类型。本练习专门检查 Brocade DCX 导向器交换机中的插槽。您的系统和结果可能有所不同，但此处介绍的一般原则可能仍然适用。

1. 在开始之前，最好先检查一下刀片服务器的 HA 状态，看看一切是否正常。您可以通过运行 hashow 命令来执行此操作：

```
LegendarySw01:admin>hashow
```

本地 CP (插槽 6,CP0) :活动、热恢复

远程 CP (插槽 7,CP1) :待机、健康

HA 已启用，心跳已启动，HA 状态已同步

这些结果显示了 Brocade DCX 导向器交换机中安装的两个控制处理器 (CP) 刀片的 HA 状态。如您所见，插槽 6 中的 CP 当前处于活动状态，而插槽 7 中的 CP 处于待机模式。两个 CP 当前处于同步状态，这意味着可以进行故障转移。

2. 现在运行 slotshow 命令，该命令显示 Brocade DCX 导向器交换机中安装了哪些插槽，以及每个插槽安装了哪种刀片：

```
LegendarySw01:admin>slotshow
```

槽刀类型	ID	型号	状态
1 SW 刀片	77	FC8-64	已启用
2 SW 刀片	97	FC16-48	已启用
3 SW 刀片	96	FC16-48	已启用
4 SW 刀片	96	FC16-48	已启用
5 核心刀片	98	CR16-8	已启用
6 CP 刀片	50	CP8	已启用
7 CP 刀片	50	CP8	已启用
8 核心刀片	98	CR16-8	已启用
9 未知			空的
10 未知			空的
11 未知			空的
12 未知			空的

如您所见，该导向器交换机的 12 个插槽中有 8 个已占用。它具有三个 48 端口 16 Gbps 交换刀片、一个 64 端口 8 Gbps 交换刀片、两个核心刀片和两个控制处理器刀片。

交换机端口

FC 交换机和导向器包含多个不同类型的端口。在物理层面上,这些端口类型将是光纤或铜缆 SFP/SFP+ 端口,支持 FCP 或 DCB 以太网/FCoE。光纤比铜缆更受欢迎,但我们很快会更详细地讨论这一点。正如我们在讨论 HBA 和 CNA 时提到的,SFP 是一种行业标准的可插拔连接器。

Brocade 交换机上的以下 sfpshow 命令显示有关交换机中安装的SFP模块:

```
LegendarySw01:admin> sfpshow
插槽 1/端口 0:id (sw) 供应商 :HP-A BROCADE
序列号 :UAA110130000672U 速度 :2,4,8 MB/s
```

从中,您可以看到此 SFP 位于插槽 1 中的端口 0。您还可以看到它具有短波 (sw) 激光器,可以看到它的序列号,并且可以看到它支持 2.4 和 8 MB/s 的传输速度。

交换机端口也有逻辑参数。在 FC SAN 中,交换机端口可以处于多种模式中的任意一种。以下解释了其中最重要和最常见的模式:

U_Port 这是 FC 端口在未配置和未初始化时通常处于的状态。

N_Port 这是节点端口。结构中的终端设备 (例如服务器或存储阵列 HBA 中的端口) 以 N_Port 的形式登录到结构中。主机 N_Port 连接到交换机 F_Port。

接受来自 N_Port 的连接的 F_Port 交换机端口作为结构端口运行。

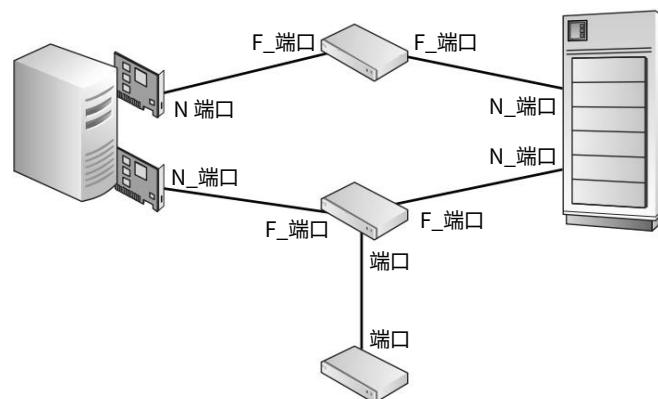
E_Port 扩展端口用于将一个 FC 交换机连接到结构中的另一个 FC 交换机。两个交换机之间的链路称为 ISL。

EX_Port 这是用于 FC 路由的 E_Port 的一个特殊版本。E_Port 和 EX_Port 之间的主要区别在于 E_Port 允许两个交换机合并结构,而 EX_Port 阻止结构合并。

还有其他端口模式,但列出的是最常见的。

图 5.3 显示了 N_Port、F_Port 和 E_Port 的连接性。

图 5.3 N_Port、F_Port 和 E_Port 连接



端口速度

正如您之前在 `sfpshow` 命令的输出中看到的,交换机端口可以以不同的速度运行。常见的 FC 速度包括以下几种:

- 2 Gbps
- 4 Gbps
- 8 Gbps
- 16 Gbps

FCoE/DCB 以太网端口通常在以下情况下运行:

- 10 GB
- 40 GB

FC 端口 (HBA 端口、交换机端口和存储阵列端口) 可以配置为自动协商其速度。自动协商是一种小型协议,允许两个设备就链路的共同速度达成一致。通常情况下,这种方法是有效的,可以协商出最高速度。但是,这种方法并不总是有效,在现实世界中,在链路的两端硬编码交换机端口速度是一种很好的做法。例如,如果支持最大端口速度为 8 Gbps 的主机 HBA 连接到支持最高 16 Gbps 的交换机端口,则您需要手动将两个端口硬编码为以 8 Gbps 运行。

对于运行速度为 10 GB 或更高的 DCB 端口的 FCoE 交换机和刀片,此做法可能有所不同。对于 FCoE 使用案例,请咨询交换机供应商的最佳做法。

域 ID

每个 FC 交换机都需要一个域 ID。此域 ID 是一个数字字符串,用于唯一标识结构中的交换机。域 ID 可以由管理员设置,也可以由结构中的主交换机在重新配置结构事件期间动态分配。

重要的是域 ID 在结构内必须是唯一的。如果在同一结构中为两台交换机配置相同的域 ID,则可能会发生糟糕的事情。



如果您使用端口分区 (我们稍后会介绍),更改交换机的域 ID 将更改该交换机的所有端口 in ID,这需要您更新分区配置。这简直是噩梦。 *switch, requiring you to update your zoning configuration. That is a nightmare.*

Principal Switches

每个结构都有一个且只有一个主交换机,主交换机是结构中等级最高的交换机。主交换机说了什么就执行什么。comes to the pecking order within a fabric. What the principal switch says, goes!

主交换机负责结构中的几件事,但与您的日常工作最相关的事如下:fabric, but the things that will be most pertinent to your day job are the following:

- 它们管理结构内域 ID 的分发。distribution of domain IDs within the fabric.
- 它们是织物中权威的时间来源。A source of time in a fabric.

除了这些因素之外,您不必担心主要开关在您的日常工作中的作用。

本机模式和互操作模式

FC 结构可以在纯模式或互操作模式下运行。您的交换机将在 99.9% 的时间里以纯模式运行,这绝对是您希望它们以这种模式运行。

互操作模式旨在支持异构结构 包含来自多个供应商的交换机的结构。为了支持这些异构配置,互操作模式会禁用某些高级和供应商特定的功能。互操作模式的使用案例往往仅限于收购等情况,即两家公司的 IT 环境合并在一起,并且结构运行来自不同供应商的交换机。但即使在这些情况下,合并此类结构并以互操作模式运行也很少是一种好的体验。关于互操作模式的最佳建议是无论如何都避免使用它们!

但是,如果您确实需要使用来自不同供应商的交换机来操作结构,那么现在有一种比使用互操作模式更好的方法。一种称为 NPV 的新技术已经解决了这个问题,我们将在本章后面讨论它。

因此,根据您目前所学的知识,可以使用以下switchshow命令
LegendarySw01告诉我们很多有用的信息:

```
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

FC 集线器

与网络世界的其他部分一样,FC 集线器已被交换机取代,这是件好事。交换机比集线器更具可扩展性和性能。

从技术上讲,FC 集线器在 FC-0 层运行,用于连接 FC-AL (仲裁环路)设备。由于 FC-AL 寻址限制,集线器只能寻址 127 个设备,但实际上甚至更少。FC-AL 集线器还要求所有设备共享带宽并仲裁线路控制,确保在任何一个时间点只有一个设备可以在集线器上通信。

另一方面,交换机可运行至 FC-2 层,提供大规模可扩展性,允许多个同时通信,并为所有连接设备提供完全无阻塞带宽。

值得庆幸的是,FC 集线器基本上已经成为过去的东西,除了计算机历史博物馆之外,你不太可能在其他地方看到它们。

FC 存储阵列

由于本书有一整章专门讨论存储阵列,因此本节仅对其稍加涉及。

在 FC SAN 中,存储阵列是具有一个或多个节点端口的端点/终端设备 (N_Ports)。这些节点端口配置为目标模式,因此充当 SCSI 目标,接受来自 SCSI 发起方 (例如基于服务器的 HBA 和 CNA) 的 SCSI 命令。由于它们是节点端口,因此它们完全参与 FC SAN 服务,例如注册和查询名称服务器以及遵守分区规则。

存储阵列往往会容纳大量 (数百甚至数千个) 磁盘和闪存驱动器,这些磁盘和闪存驱动器作为块设备 (称为 LUN 或卷) 在 FC SAN 上共享。



要深入了解存储阵列,请参阅第 3 章“存储阵列”Chapter 3, “Storage Arrays.”

电缆

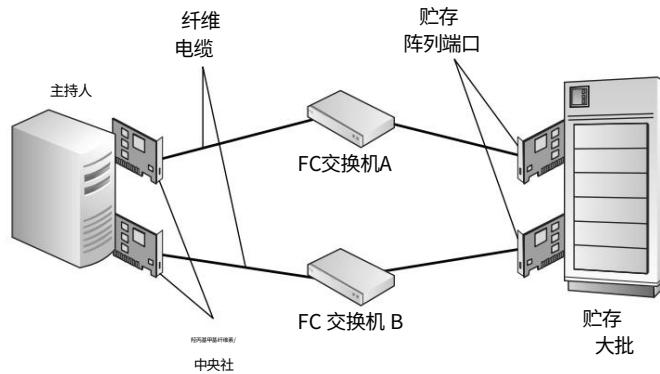
不要忽视电缆,因为它是任何 SAN 的关键组件。遗憾的是,FC 尚未实现无线化。事实上,其他主要数据中心网络技术也没有。因此,电缆和电缆在 SAN 环境中非常重要。

让我们先了解一些基础知识,然后再进行深入了解。

几乎所有 FC SAN 布线都是光纤。光纤以激光或 LED 产生的光的形式传输数据。可以使用铜缆,但与光纤相比很少见。光纤可用于机架顶部 (ToR) 或行末 (EoR) 情况。一些存储供应商在转向 SAS 后端之前在其旧 FC 后端上使用铜缆。所有光纤布线都由两对光纤组成 - 发送 (TX) 和接收 (RX) - 允许全双工模式。

图 5.4 显示了具有两个终端设备和一个 FC 交换机的简单 FC SAN,并指出了一些物理组件的位置。

图 5.4 简单的 FC SAN 物理组件



那么,让我们详细了解一下。光纤电缆有两种主要类型:

- 多模光纤 (MMF)
- 单模光纤 (SMF)

多模光纤传输短波激光器的光,而单模光纤传输长波激光器的光。

说到 SAN 布线,多模光纤是数据中心无可争议的王者。这归功于两点:

- 多模光纤比单模光纤便宜。
- 多模光纤可轻松传输数据,距离可达最大的数据中心常见的距离。

单模光纤及其相关的长波激光器 (光学器件)价格更昂贵,往往被需要更长电缆 (例如数百英里)的电信公司所使用。

所有光纤电缆 (多模和单模)均由以下主要组件组成:

- 核心
- 包层
- 夹克

纤芯是电缆中心用于传输光线的超小玻璃或塑料圆筒。如果您看到剥开的光纤电缆,并认为您用肉眼看到了电缆的纤芯,那么您要么是超人,要么您看到的不是纤芯。光纤电缆的纤芯太小,肉眼无法看到!

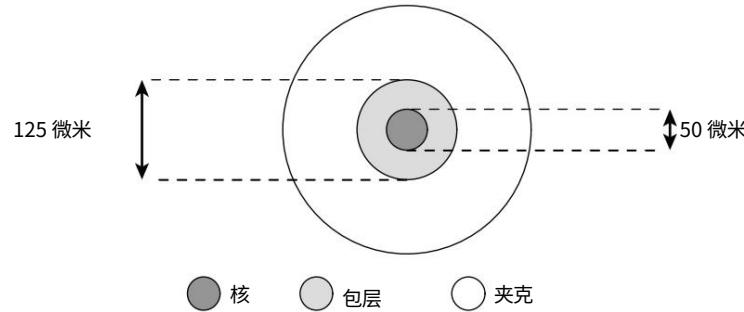
纤芯周围是包层。包层保护纤芯,并确保光不会从核心逸出。

所有光纤电缆的外护套都称为护套。护套通常根据所承载的光纤类型进行颜色编码。

是的,包层和护套之间还有其他层和物质。但是,了解它们对你的日常工作毫无用处。

图 5.5 显示了光纤电缆的纤芯、包层和护套。

图 5.5 芯线、包层和护套

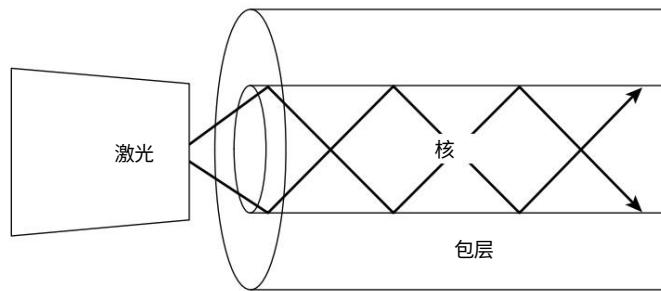


多模光纤

相对于其所承载的光的波长,多模光纤具有较大的纤芯直径,这使得产生光的电子设备和激光器比单模光纤使用的电子设备和激光器精度更低,因此也更便宜。

当光被注入多模光纤时,它可以以略微不同的角度进入电缆,导致光波在沿着电缆长度传播时从纤芯/包层周边反弹。如图 5.6 所示。

图 5.6 承载两束光的多模光纤



然而,随着距离的增加,这些离散光束会交叉,导致一些信号损耗称为衰减。电缆越长,光束交叉越多,导致衰减增加。多模光纤不能传输很长的距离。

多模光纤具有以下两种纤芯/包层选项:

- 62.5/125 微米
- 50/125 微米

通常,它们分别被称为 62.5 微米和 50 微米。62.5 和 50 指的是纤芯的横截面积大小 (以微米为单位)。 $50 \mu\text{m}$ 纤芯比 62.5 纤芯更有效地传输光线,因此现在它比 62.5 更受欢迎。

多模光缆的芯线尺寸相对较大 (显然非常小,但与单模相比还是很大),这意味着它们可以制造成比单模光缆质量稍差、设计精度稍低的电缆。这使得它们比单模更便宜。

多模光纤也按照光学多模 (OM) 标识符进行分类:

- OM1 (62.5/125)
- OM2 (50/125)
- OM3 (50/125) 有时被称为激光优化
- OM4 (50/125)

按照惯例,OM1 和 OM2 护套颜色应为橙色,而 OM3 和 OM4 护套颜色应为浅绿色 (浅蓝色)。但是,大多数光纤电缆都可以以您想要的任何颜色购买,有些人会选择更易于管理数据中心的配色方案。例如,一种结构使用一种电缆颜色,另一种结构使用另一种电缆颜色。这将防止您在不同结构之间混淆电缆。但是,更好的选择可能是对电缆连接器进行颜色编码,而不是对实际的电缆护套进行颜色编码。

通常,每次 FCP 的速度增加时,例如从 4 Gbps 增加到 8 Gbps,或从 8 Gbps 增加到 16 Gbps,给定电缆类型的最大传输距离就会减小。表 5.1 显示了不同多模电缆类型在各种常见 FCP 速度下支持的最大距离。

表 5.1 MMF 距离

光纤通道速度 (Gbps) OM1	OM2	OM3	OM4
4 Gbps	70 米	150 米	380米
8 Gbps	21 米	50公尺	150 米
16 Gbps	15 米	35 米	100 米

表 5.1 应该可以帮助您了解需要购买哪种类型的电缆。虽然一些公司现在选择到处使用 OM3,但如果要在机架内或相邻机架之间铺设电缆,这样做会让您不必要地支付更多费用。另一方面,在结构化电缆铺设中到处使用 OM3 可能是一个好主意,因为这可以为您的基础设施提供未来保障,例如 40 GB 和 100 GB 以太网。但是,机柜内和相邻机柜的铺设目前还不需要 OM3,如果将来需要,它们很容易被撕掉和更换。

单模光纤

单模光纤电缆仅传输一束光，通常由长波激光产生。单模光纤电缆相对于激光的波长较窄，这意味着仅捕获一种模式，因此当光沿电缆长度传播时，不会发生干扰或从芯/包层边界过度反射。这使得单模光纤能够将光传输到以英里而不是英尺为单位的距离。单模光纤电缆也往往比多模光纤更昂贵。

单模光纤纤芯尺寸通常为 $9 \mu\text{m}$ ，但仍带有 $125 \mu\text{m}$ 包层。此 $9 \mu\text{m}$ 纤芯需要极其精确的工程设计，这也是单模光纤比多模光纤更昂贵的主要原因。

单模光纤在数据中心中不如多模光纤那么流行，而且通常带有黄色外壳。

弯曲半径和灰尘

由于光纤电缆具有玻璃芯，因此弯曲时需要非常小心。
弯曲过度会损坏它们。而且它们并不便宜，所以请注意！

当谈论弯曲光纤电缆时，我们使用弯曲半径这个术语。最小弯曲半径是电缆可以弯曲到不会严重损坏的程度。

如果将电缆弯曲得太紧，通常会在电缆上留下明显的扭结，很容易看到电缆断裂。但是，即使您没有扭结电缆或使玻璃芯破裂，也会对包层施加过大的压力，使其失去折射能力，即保持光线在芯内的能力。无论哪种方式，电缆都会损坏，如果您继续使用它，只会让您头疼。

光纤电缆的建议最小弯曲半径根据光纤电缆的类型而有所不同 - OM1 与 OM3 不同，单模与多模不同，等等。

一般来说，弯曲半径越大越好，一般来说，不小于 3 英寸是一个不错的起点。这样，您就可以降低弯曲造成损坏的风险。

确保您始终遵守制造商的建议并且不要冒险！

我们还应该指出，电缆连接器末端或光交换机端口上的灰尘可能会导致诸如连接不畅或记录高错误计数的连接等问题。这就是为什么新的光纤电缆配有橡胶或塑料盖，以防止灰尘进入。在使用光纤电缆之前，没有足够多的人遵循正确清洁光纤电缆末端的良好做法。保持电缆末端清洁！图 5.7 显示了带有橡胶防尘盖的 SFP 光纤。



Cleaning fiber cable connectors has to be done properly with the appropriate equipment. Do not spit on the end of a fiber-optic cable and rub it

图 5.7 SFP+ 模块（带橡胶防尘盖）



电缆连接器

大多数光纤电缆都配有公头连接器,可轻松插入和拔出交换机、HBA 和存储阵列端口 - 这一过程有时称为配对。虽然有许多光纤连接器类型可供选择,但作为存储管理员,您需要熟悉其中几种,因此我们将介绍这些类型。

SC 和 LC 可能是最常见的连接器。SC 代表标准连接器,LC 代表透明连接器。两者都是公连接器。LC 连接器只有 SC 连接器的一半大小,因此越来越受欢迎。一半的大小可以等于两倍的密度。



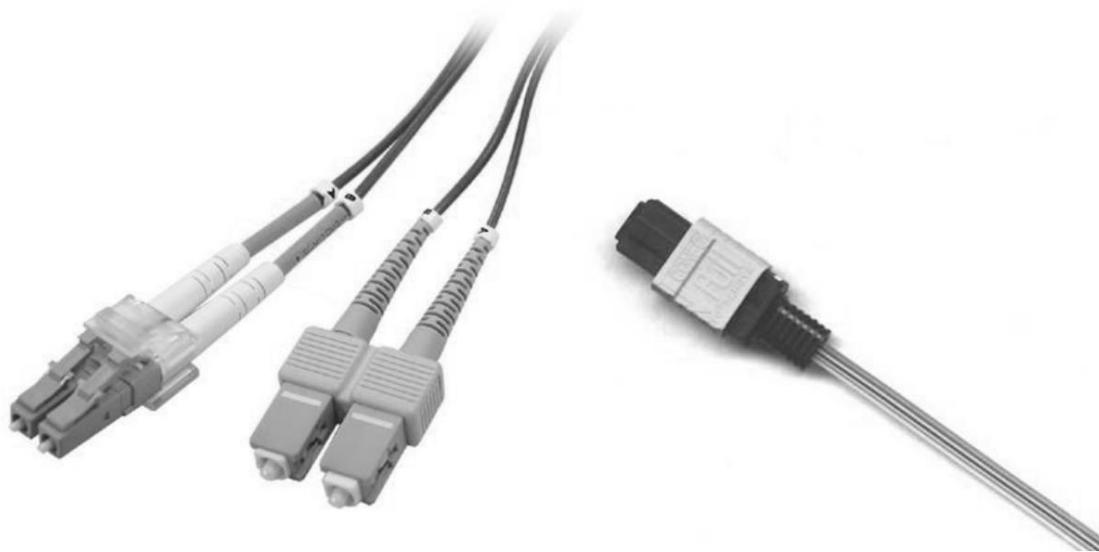
LC 连接器的尺寸是 SC 连接器的一半,并且两者都可用于多模和单模电缆。They are available in multi-mode and single-mode cables.

另一种常用的电缆是 MPO 或 MTP 电缆。MTP 代表多光纤端接推入式,是一种带状电缆,可承载多根光纤(通常为 12 根或 24 根),并将它们端接在单个连接器上。MTP 是 MPO 连接器的一种变体。

确实存在其他连接器类型,但 SC、LC 和 MTP 是目前最流行的大数据中心。

图 5.8 显示了 SC、LC 和 MTP 电缆连接器。

图 5.8 常见的光缆连接器



如果光纤电缆没有连接器,则将两根电缆连接在一起的唯一方法是将它们拼接起来。拼接是一种需要专业设备的专业程序,通常由专业人员执行。您很可能在整个存储生涯中都不需要拼接电缆。

现在是时候讨论 FC SAN 的逻辑组件了。

逻辑 SAN 组件

现在我们来讨论一下 SAN 的一些理论和逻辑组成部分。

FCP 堆栈

与所有优秀的网络技术一样,FCP 是一种分层协议,只是它不像以太网和 TCP/IP 等协议那样丰富。最终结果是更简单的堆栈。图 5.9 显示了光纤通道协议的层次。

图 5.9 FCP 堆栈



由于 FCP 是一种更简单的协议,因此你会发现自己需要重新参考 FCP 层与 TCP/IP 网络相比,频率要低一些。

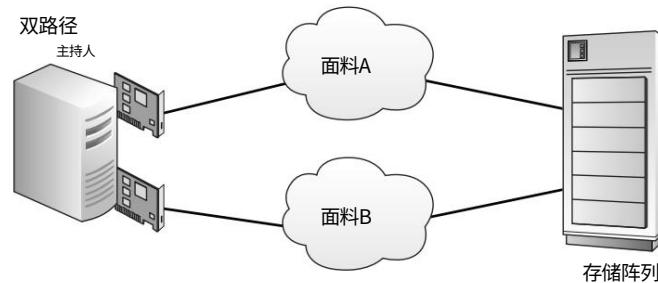
假如 IP 网络团队试图告诉你 FCP 的简单性或更少的层数是 FCP 的某种缺点,事实并非如此。这实际上反映了 FCP 更多的是一种通道技术而不是网络技术。请记住,在本章开头我们解释过,通道是低开销、低延迟、高速互连,试图重建通道而不是大规模复杂的网络。

FC 面料

光纤通道交换机是一组相互连接的光纤通道交换机的集合,这些交换机具有一组通用的服务。例如,它们共享一个通用的名称服务器、通用的分区数据库、通用的 FSPS 路由表等等。

大多数情况下,您需要部署双冗余结构以实现弹性。每个这两种结构是相互独立的,如图5.10所示。

图 5.10 双冗余结构





将织物画成云是一种常见的做法。draw a fabric as a cloud.

每个结构都被视为和管理为单个逻辑实体。如果您想要更新结构中的分区配置，则该配置在整个结构中是通用的，并且可从结构中的任何交换机进行更新。

如果您有两台未相互连接的交换机，则每台交换机将运行其自己的独立结构。如果您通过在每个交换机的 E_Port 之间拉一根电缆来连接这两台交换机，则结构将合并并形成单个结构。这种合并如何发生，以及现有分区配置和主交换机会发生什么，可能很复杂，并且取决于您的结构的配置。在执行此类操作之前，请进行研究并遵循供应商的最佳实践。

两种织物可能不会合并。当这种情况发生时，织物被称为分段。如果每个交换机的配置文件中的某些重要配置参数不匹配，则可能会发生交换机与架构的分段。在合并架构之前，请确保架构中的所有交换机均已正确配置。

一旦结构合并，结构中的任何设备都可以与其他任何设备进行通信只要分区配置允许，就可以将一个设备添加到结构中。



当通过两个 E_Port (每个交换机上只有一个连接由交换机会合并) 这两个 E_Port 之间形成的链路称为交换机间链路 (ISL) 但是如果许多交换机端口检测到它们另一端有另一个交换机，它们会自动地将自己配置为 E_Port。您需要注意这一点，因为如果您的布线错误，您可能会无意中合并两个结构。我们在讨论结构拓扑时详细讨论了 ISL。

结构服务

结构中的所有交换机都支持并参与提供一组通用的结构服务。根据 FC 标准的定义，这些结构服务可以通过以下众所周知的 FC 地址访问：

0xFF FF F5:多播服务器

0xFF FF F6:时钟同步服务器

0xFF FF FA:管理服务器

0xFF FF FB:时间服务器

0xFF FF FC:目录服务器/名称服务器

0xFF FF FD:结构控制器服务器

0xFF FF FE:Fabric 登录服务器

0xFF FF FF:广播地址

并非所有这些结构服务都已实现,从日常角度来看,有些比其他更重要。让我们仔细看看。

时间服务器负责结构中的时间同步。管理服务器允许从结构中的任何交换机管理结构。分区服务也是管理服务器的一部分。结构控制器服务器负责主交换机选择、发出注册状态更改通知 (RSCN) 以及维护结构最短路径优先 (FSPF) 路由表。结构登录服务器负责发出 N_Port ID 并维护结构上注册设备的列表。

FC 名称服务器

FC 简单名称服务器 (SNS),如今更广为人知的名字是名称服务器,是结构中注册的所有设备的分布式数据库。这是一项至关重要的结构服务,您将在分区和故障排除过程中定期与之交互。

所有加入 Fabric 的设备都需要向名称服务器注册,使其成为 Fabric 中动态的、集中的信息点。由于它是分布式的,Fabric 中的每台交换机都包含一个本地副本。

目标和发起者在知名的名称服务器上注册并查询地址0xFFFFFC。

分区

SAN 结构中的设备可见性通过分区控制。如果您希望两个设备能够相互通信,请确保它们被划分到同一个区域!

回到 SCSI 的最初时代,当时所有设备都直接连接到单个服务器内的单根电缆,这是一种分区形式 只有单个服务器才能访问这些设备。快进到共享 SAN 环境的时代,数百个目标和数百个发起者可以相互通信,很容易看出这与 SCSI 的初衷有很大不同。必须采取一些措施!

结构和端口登录

在深入研究区域划分之前,我们将介绍一些有关设备发现以及结构登录和端口登录的概念的重要背景工作。

当启动器首次在 SCSI 总线 (包括 FC SAN) 上初始化时,它会执行重置,然后发现总线上的所有设备。对于 SAN,总线就是 SAN。在没有分区的 SAN 结构上,启动器将探测并发现 SAN 结构上的所有设备。作为此发现的一部分,还将查询每个设备以发现其属性和功能。在大型 SAN 上,这可能需要很长时间,并且会浪费大量资源。但更重要的是,访问其他主机正在使用的设备可能会很危险。



Real World Scenario

分区的重要性

在采用 FC SAN 的早期,许多公司部署了开放式 SAN (无分区)或具有多个启动器和目标的大型区域。虽然这很简单,但却导致了很多问题。我工作过的一家公司就遇到过这样的问题,即神秘地重置磁带驱动器。磁带驱动器是连接到 FC SAN 的小型 SCSI 磁带库的一部分。但是,没有分区,这意味着小型 SAN 上的所有设备都可以互相看到并相互影响。经过大量的思考和故障排除,问题的根本原因被确定为每次服务器重新启动时都会发生的 SCSI 重置。由于 SAN 是完全开放的 (无分区),因此每次发出 SCSI 重置时,SAN 上的所有设备都会收到它。事实证明,磁带驱动器的设计目的是在收到 SCSI 重置时执行重置和完全介质倒带。最终结果是,每次服务器重新启动时,随后的 SCSI 重置都会导致 SAN 上的所有磁带驱动器重置和倒带,这显然会中断任何正在进行的备份工作。通过实施分区,这个问题得到了解决。

为了加快、平滑并普遍改善发现过程,名称

服务器的创建。自从名称服务器发明以来,每次任何设备加入结构时,它都会执行一个称为结构登录 (FLOGI) 的过程。此 FLOGI 过程执行一系列重要功能:

- 为设备分配最重要的 24 位 N_Port ID
- 指定要使用的服务类别

■ 建立设备的初始信用库存

FLOGI 过程成功后,设备将执行端口登录 (PLOGI)

名称服务器来注册其功能。加入结构的所有设备都必须对名称服务器执行 PLOGI。作为 PLOGI 的一部分,设备将向名称服务器请求结构上的设备列表,这就是分区发挥作用的地方。名称服务器不返回结构上的所有设备的列表,而是仅返回那些已分区以便从执行 PLOGI 的设备访问的设备列表。执行登录的设备获得此列表后,它会对每个设备执行 PLOGI 以查询每个设备的功能。此过程比探测整个 SAN 中的所有设备更快、更安全,并且还允许更大的控制和更灵活的 SAN 管理。

即使采用上述流程,如果结构上没有分区,那么实际上就是一场混战。每台设备都可以看到其他所有设备,您将发现自己处于非常糟糕的境地。

现在让我们仔细看看分区。

区域

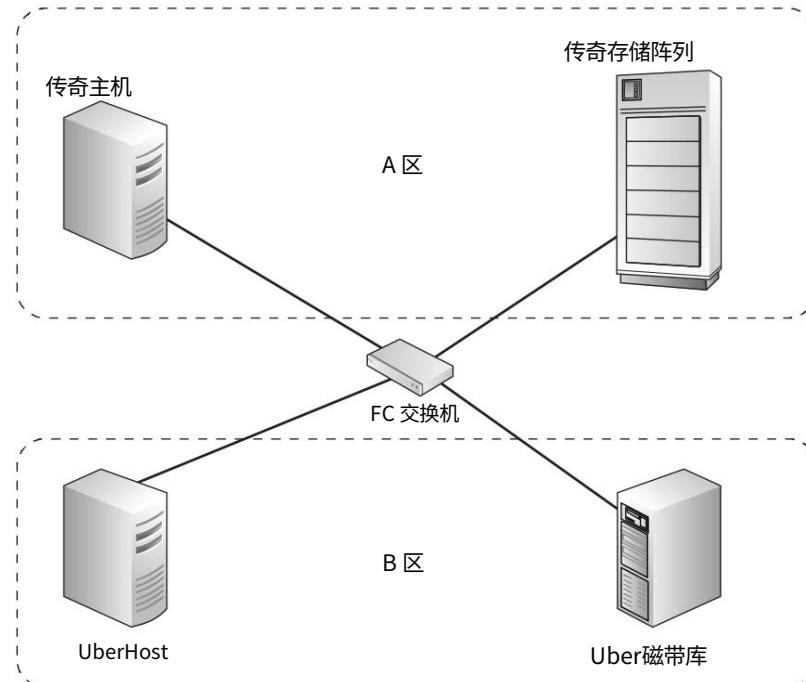
分区的基本元素是区域,所有 SAN 结构都包含多个区域。

多个区域被分组到称为区域集的容器中。

正如我们已经暗示的那样,分区是一种划分结构的方法,每个单独的区域就像一个微型结构,是更广泛结构的一个子集。如果您希望两台设备相互通信,请将它们放在同一个区域中;所有其他设备都应排除在该特定区域之外。

图 5.11 显示了具有两个区域 (区域 A 和区域 B) 的简单 SAN。图中, LegendaryHost 只能与 LegendaryStorageArray 通信,同时, UberHost 只能与 UberTapeLibrary 通信。Fabric 上有四个设备,两个区域在运行 有点像两个微型 Fabric。

图 5.11 简单分区示例



尽管图 5.11 中的示例很小,但分区可以扩展到非常大的结构,可能包含数千个设备和数千个区域。如本例所示,区域由发起方和目标组成。标准做法是为这些发起方和目标赋予友好名称(称为别名),这在解决连接问题时非常有用。

练习 5.2 展示了如何配置区域划分。

练习 5.2

配置区域

为简单起见,您将在单个结构中创建单个区域。通常,您将部署双冗余结构并在每个结构中创建一个区域,以便每个服务器都可以通过两个独立的结构访问其存储。

1. 为已添加到 SAN 的新主机创建别名。新主机名为 LegendaryHost, 其 HBA 的 WWPN 为 50:01:43:80:06:45:d3:2f。在此示例中, 您将别名命名为 LegendaryHost_HBA1, 因为它是服务器中名为 LegendaryHost 的第一个 HBA 的 WWPN 的别名。

```
alidcreate LegedyHost_HBA1 "50:01:43:80:06:45:d3:2f"
```

假设您的 EMC VMAX 存储阵列端口的别名已在结构中创建, 名为 VMAX_8E0。

2. 现在您将创建一个包含两个别名的新区域。一个别名将是您刚刚创建的新别名, 另一个别名将用于 EMC VMAX 存储阵列上的端口 8E0。新区域将被称为 LegendaryHost_HBA1_VMAX_8E0, 这样当您看到区域的名称时, 您就知道该区域的用途。使用 zonecreate 命令创建区域:

```
zonecreate LegendaryHost_HBA1_VMAX_8E0, LegedyHost_HBA1; VMAX_8E0
```

您现在有一个包含两个别名的新区域, 它将允许 LegendaryHost 中的 HBA1 与 EMC VMAX 阵列上的端口 8E0 进行通信。

3. 使用以下命令将新创建的区域添加到名为 Prod_Config 的现有区域集
cfgadd 命令:

```
cfgadd Prod_Config, LegendaryHost_HBA1_VMAX_8E0
```

4. 保存配置。

保存配置文件

5. 确保 Prod_Config 是活动区域集:

```
cfgactvshow
```

有效配置:

```
cfg: 产品配置
<输出被截断>
```

此过程在LegendaryHost中为HBA1创建了一个新别名，并创建了一个新区域，其中包含LegendaryHost的WWPN和我们的EMC VMAX阵列端口8E0的别名。它还将新创建的区域添加到活动区域集并启用配置。因此，LegendaryHost现在将能够与我们的VMAX阵列进行通信。

分区最佳实践

以下有关分区的经验规则将使您的SAN保持良好状态：

- 1.保持区域较小。这将使故障排除更简单。
- 2.每个区域只有一个发起者。拥有更多发起者被认为是不好的做法
一个区域中可以有多个启动器。
- 3.保持区域中目标的数量较少。每个区域有多个目标比每个区域有多个启动器更容易接受。但是，不要太过分，因为这会使故障排除更加困难。
- 4.为您的区域和别名赋予有意义的名称。
- 5.在您的环境中进行区域更改时要格外小心。

在这些最佳实践中，现实世界中最重要的可能是

第2点，称为单启动器分区。简而言之，单启动器分区规定每个区域应仅包含一个启动器以及启动器需要访问的任何目标。遵循这一原则，您的生活将变得轻松很多。



重要的是要了解启动器和目标可以在多个区域的成员。事实上，如果启动器是区域的成员，这是必要的，但它也允许多个启动器访问单个目标。例如，如果您有两个启动器想要访问同一个目标，您可以创建两个区域：一个区域包含第一个启动器和目标，第二个区域包含第二个启动器和同一个目标。这样，您就有两个单独的区域，每个区域只有一个启动器，但两个区域都有相同的目标。

这种方法的最终结果是两个启动器能够访问单个目标，但它也保持了实施单启动器区域的最佳实践。

区域集

多个区域被组合成区域集，区域集将应用于结构。如果您配置了新区域，则需要将其添加到活动区域集，以便将其应用于结构。



Real World Scenario

不要忘记应用您的分区更改！

当新区域似乎无法正常工作时,SAN 管理员感到困惑。经过大量的反复检查和研究,管理员最终意识到他一直在创建新区域并将其保存到活动区域集,然后就此打住。为了将新创建的区域应用于结构,需要将它们添加到活动区域集,并且更新的活动区域集通常需要重新应用于结构。忘记将活动区域集重新应用于结构是许多组织中常见的错误。

FC 结构可以有多个已定义的区域集。但是,在任何给定结构上,任何给定时间只能有一个区域集处于活动状态。

别名

我们简要提到了别名,但让我们仔细检查一下。

WWPN 非常长,对普通人来说毫无意义。别名允许我们为 SAN 上的 WWPN 赋予友好名称。我们不必记住 WWPN 50:01:43:80:06:45:d3:2f 指的是我们名为 LegendaryHost 的主机中的 HBA1,而是可以为该 WWPN 创建具有有意义名称的别名。

以下命令在 Brocade 结构中创建一个名为 LegendaryHost_HBA1 的别名:

```
LegendarySw01#管理员> alicreate LegedaryHost_HBA1 "50:01:43:80:06:45:d3:2f"
```

以下命令集创建相同的别名,但这次是在 Cisco MDS 结构上:

MDS-sw01# 配置

输入配置命令,每行一个。以 CNTL/Z 结尾。

```
MDS-sw01 (配置)# fc别名 LegendaryHost_HBA1
MDS-sw01 (config-fcalias)# 成员 pwwn 50:01:43:80:06:45:d3:2f
MDS-sw01 (config-fcalias)# 退出
```

大多数人在配置区域时使用别名,这是一个值得坚持的做法。

端口或 WWN 分区

有两种流行的分区形式:

- WWN 分区
- 口径分区

本质上,这只是识别您正在分区的设备的两种方式 要么通过 WWPN 或设备连接到的交换机端口的 ID。

每种方法都有其优缺点。如果您采用端口分区,然后必须更换主机中发生故障的 HBA,则只要新 HBA 仍连接到发生故障的 HBA 所在的同一交换机端口,您就无需更新分区。但是,如果您实施 WWN 分区,则需要更新别名并重新应用区域集以反映新 HBA 的新 WWPN,或者重置新 HBA 上的 WWPN 以匹配旧 HBA 的 WWPN。

在这方面,端口分区似乎更简单。但是,端口分区中使用的端口 ID 可能会发生变化,这一点非常重要!这种情况很少见,但确实会发生。这是因为端口 ID 包含交换机的域 ID,如果结构必须执行破坏性重新配置,交换机的域 ID 可能会发生变化。如果真的发生这种情况,您将面临无效分区的困境,并且需要付出大量努力才能让一切恢复正常。出于这个原因,大多数人选择使用 WWN 分区。此外,从安全角度来看,端口分区被认为比 WWN 分区更安全,因为伪造 WWN 并不太难,而伪造您连接的端口则更困难。



有些人将端口分区称为硬分区,将WWN分区称为软分区,这可能会产生误导。Zoning as soft zoning. This can be misleading.

硬分区或软分区

软分区基本上是名称服务器强制分区。

当设备向名称服务器执行 PLOGI 并请求结构上的设备列表时,名称服务器仅返回与执行 PLOGI 的设备位于同一区域中的设备列表。这样,登录到结构的设备只能看到结构,并且应该只登录到这些设备。但是,HBA 或 CNA 中的坏驱动程序可能会决定对结构上的任何或所有可能的设备地址执行 PLOGI。如果它这样做了,没有什么可以阻止它。因此,软分区只不过是通过隐蔽性实现的安全性,并依赖于连接的设备配合。

WWN 和端口分区都可以作为软分区来实现。

另一方面,在硬分区中,交换机硬件会检查穿过结构的所有流量,并根据分区控制主动过滤和丢弃不允许的帧。

guration。名称服务器分区仍然有效 - 名称服务器仍然为登录的设备提供结构受限视图 - 但硬分区也作为附加安全级别实施。以下命令显示,在UberSw01 上,硬分区已到位,我们正在使用 WWN 分区:

```
UberSw01:管理员> portzoneshow
```

端口: 0 (0) F 端口	执行:HARD WWN defaultHard:0 IFID:0x44120b38
----------------	---

端口:1 (1) 离线	
-------------	--

端口:2 (2) F 端口	执行:HARD WWN defaultHard:0 IFID:0x4412004b
---------------	---

您的结构上是否采用硬分区取决于供应商,并且取决于您的结构和分区配置方式。请咨询供应商的最佳实践以部署硬分区。



Once an active zone is defined, no fabric-aware devices that do not reside in the active zone set are frozen out and cannot access any devices.

SAN 拓扑

现在让我们介绍一下 SAN 拓扑及其所依据的一些原则,例如冗余和高可用性。

冗余

所有良好存储设计的标志都是冗余。

看看任何值得一看的存储设计,你都会看到冗余

无处不在:具有多个 HBA 的服务器,连接到不同的冗余结构,连接到冗余存储处理器;受 RAID 保护的磁盘;复制卷;等等。这很重要,因为虽然存储世界中通常没有那么多糟糕的日子,但当那些糟糕的日子真的出现时,它们往往是地狱般的日子!

几乎世界上所有的 SAN 环境都具有双冗余结构 至少两个完全隔离的结构。它们在物理和逻辑上都是隔离的,中间没有线缆。



在设计冗余结构时,请确保每个子结构都使用单独的电源电路,或者至少结构中的每个子结构都由独立的电源供电。还要确保光纤电缆走不同的路由,这样即使一根干线掉落在某块地板上不会同时与两个结构断开连接。

Figure 12 显示了两个冗余结构,并展示了不同的布线路线、电源以及类似的担忧。

常见的 SAN 拓扑

有几种公认的 SAN 拓扑,这些包括点对点 (point-to-point) SAN 拓扑 (FC-P2P)、仲裁环路 (FC-AL) 和各种交换结构 (switched fabric) 拓扑 (FC-SW)。接下来我们将对每一个进行探讨。

Point-to-Point

点对点,技术上称为 FC-P2P,是从主机 HBA 或 iSCSI 端口到存储阵列端口的直接连接。此连接可以是直接飞线连接或通过配线架连接,但不能通过 FC 交换机连接。This connection can be a direct fly-lead connection or via a patch panel, but it cannot be via an FC switch.

FC-P2P 很简单,不具备任何可扩展性。如果你的存储阵列有 8 个前端端口,最多可以有八台直接连接的服务器与该存储阵列通信。

图 5.13 显示了 FC-P2P 连接。

图 5.12 冗余电缆和电源

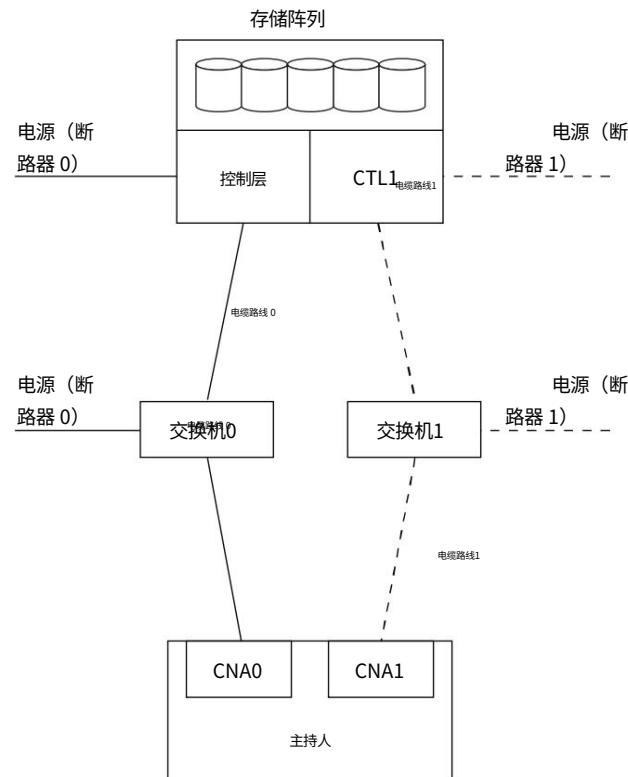
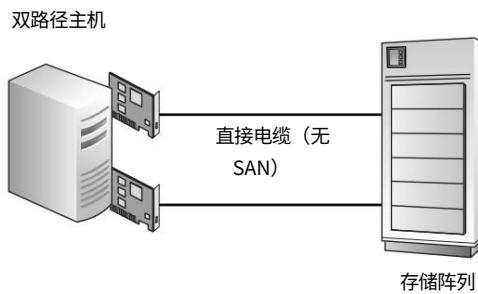


图 5.13 简单的 FC-P2P 配置



点对点配置在小型环境中最为常见,其中 FC 开关并非必需,而且不会不必要地增加成本。

仲裁环

光纤通道仲裁环路 (简称 FC-AL) 允许设备以环路拓扑连接,通常通过 FC 集线器连接。但是,集线器不是必需的,因为您可以在环路配置中以菊花链形式连接服务器,其中一台服务器的传输端口连接到相邻服务器的接收端口。

如今,FC-AL 已经过时了,很少见。这很好,因为它缺乏可扩展性 (最多127个设备) 和性能。

在 FC-AL 配置中,环路上的所有设备都会争用环路,并且任何时间点都只允许一个设备在环路上传输 I/O。这自然会导致在活动高峰期间出现争用并导致性能下降。

FC-AL 配置还存在一种称为 LIP 风暴的现象。基本上,每当将设备添加到环路或从环路中移除时,环路都会通过发送环路初始化原语 (LIP) 重新初始化。在此类重新初始化事件期间,设备无法在环路上传输数据。当 FC-AL 配置更常用时,存储管理员最头疼的就是环路上有一台故障设备,它会不断向环路发送 LIP 帧。

如今,FC-AL 配置主要仅存在于此类书籍或博物馆中。
RIP

交换结构

光纤通道交换结构在技术上称为 FC-SW,但大多数人都称之为结构。它是 FC 网络技术的事实标准。

FC-SW 克服了 FC-AL 和 FC-P2P 的所有限制,并增加了许多改进功能。因此,本节将比 FC-P2P 和 FC-AL 更加详细地介绍它。光纤通道结构由一个或多个 FC 交换机组成。FC 交换机最高可运行至 FC-2 层,每个交换机都支持并协助提供一组丰富的结构服务,例如 FC 名称服务器、分区数据库、时间同步服务等。

终端设备可以无中断地在结构中添加和移除。所有设备都可以无阻塞地访问结构的全部带宽。结构可以扩展到数千台设备。标准规定有数百万台设备,但实际上只有数千台。许多 FC 交换机都采用直通交换模式,这样就无需在交换机中缓冲每个帧,从而确保尽可能低的延迟。

当一个结构包含多个交换机时,这些交换机通过称为交换机间链路的链路连接起来。

交换机间链路

为了连接两台 FC 交换机并形成一个公共结构（具有名称服务器和分区表的单个实例）,您需要将每台交换机上的一个端口配置为 E_Port,并在这两个端口之间拉一根电缆。E_Port 是用于将两台交换机连接成一个公共结构的特殊端口,它们形成的链路称为交换机间链路或 ISL

简称。

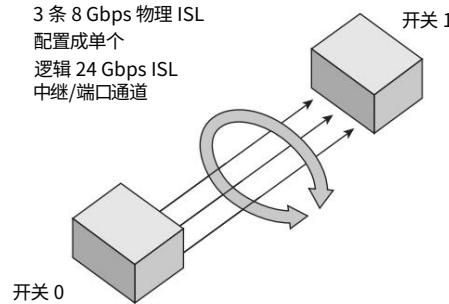
可以在两个交换机之间运行多个 ISL,这是实现冗余的常见做法。ISL 可以作为独立的 ISL 运行,也可以绑定在一起作为单个逻辑 ISL 运行。一般来说,将多个物理 ISL 绑定为单个逻辑 ISL 是首选方法,因为这可以提供卓越的冗余性以及逻辑 ISL 中所有物理链路的卓越负载平衡。



不同的 FC 交换机供应商对于将多个物理 ISL 绑定到单个逻辑 ISL 有不同的称呼。思科称之为 ISLs Trunk, Brocade 和 QLogic 称之为 ISL 中继。将逻辑 ISL 称为中继。如果不小心的话可能会造成混淆,因为网络世界的其余部分已经使用术语 trunk 来指代 VLAN 中继。

图 5.14 显示了三条物理链路绑定成一条逻辑 ISL。

图 5.14 逻辑 ISL (PortChannel/trunk)



创建 ISL PortChannel 时需要考虑的一个重要问题是去偏移。去偏移是指信号在两条或多条电缆上传输所需的时间差异。例如,信号在 100 米电缆上传输所需的时间可能比在 10 米电缆上传输所需的时间长 300 纳秒左右。当逻辑 ISL 覆盖长距离 (例如几英里) 时,去偏移值尤其重要。在这种配置中,形成逻辑 ISL 的所有物理电缆都需要具有符合交换机供应商要求的去偏移值。不遵守此规则会导致问题,例如逻辑 ISL 无法形成,甚至更糟的是,逻辑 ISL 形成但性能不佳。在涉及 ISL 相关去偏移值时,黄金法则是始终遵循交换机供应商的最佳实践!

以下两个命令显示两个物理 ISL 之间形成 ISL 中继

两台 Brocade 交换机。每条物理 ISL 的运行速度为 8 Gbps,从而创建一条带宽为 16 Gbps 的逻辑 ISL。每条物理 ISL 还具有完全相同的去偏斜值 在我们的特定情况下,这是因为它们都运行在 10 米长的短多模电缆上。它还显示主干中的一条物理 ISL 是主 ISL。并非每个供应商的逻辑 ISL 实现都具有主 ISL 和从属 ISL 的概念,如果主 ISL 发生故障,则拥有主 ISL 可能会出现问题。

```
LegendarySw01:admin>islshow
```

```
1:14->14 10:00:00:05:AA:BB:CC:EE 03 LegendarySw03 sp:8G bw:16G TRUNK QOS
```

```
LegendarySw01:admin>trunkshow
```

1:14->14 10:00:00:05:AA:BB:CC:EE	21 去偏斜 15 MASTER
15->15 10:00:00:05:AA:BB:CC:FF	21 校正倾斜 15

以下命令显示由两个物理 ISL 组成的类似逻辑 ISL
Cisco MDS 交换机：

```
MDS-sw01# 显示端口通道数据库
```

端口通道 10

管理通道模式处于活动状态

操作通道模式已激活

上次会员资格更新成功

共 2 个端口,2 个端口处于启用状态

端口:fc2/1 [向上]*

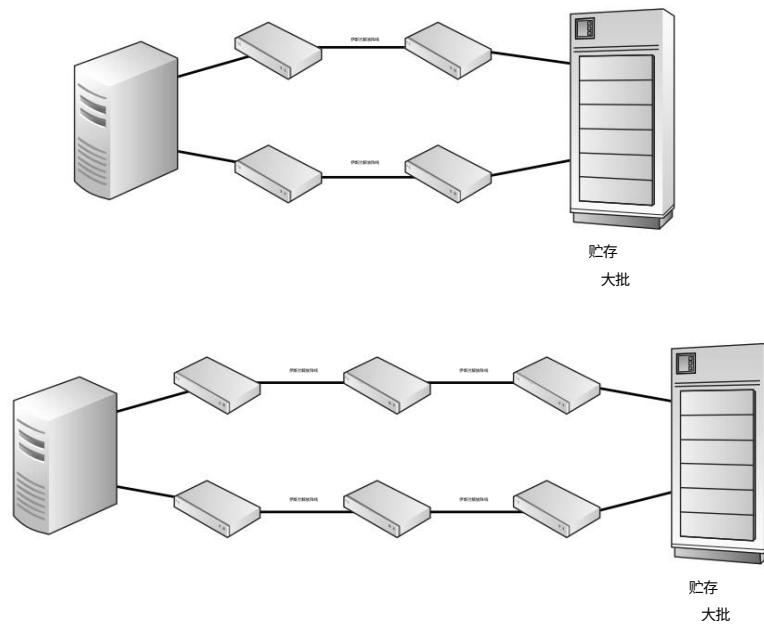
fc2/8 [上]

关于 ISL 的最后一项考虑是跳数。帧从源 N_Port 到目标 N_Port 途中必须经过的每条 ISL 称为一跳。如果您的发起方和目标被连接到通过单个 ISL 直接相互连接的单独交换机,则两者之间的任何通信都必须经过 ISL,从而产生一跳。如果必须经过两条 ISL,则相当于两跳。最好尽可能保持 SAN 的连通性并将跳数保持在最低限度。也就是说,连接到结构的节点不知道结构中有多少个交换机。

图 5.15 显示了两个简单的配置,一个需要单跳,另一个需要两跳。

在 FC 交换结构中,有几种大致可接受的拓扑结构。迄今为止,最流行的是核心边缘拓扑结构。其他设计 (如级联、环形和网状)确实存在,但与几乎无处不在的核心边缘相比,它们很少见。让我们仔细看看每种设计。

图 5.15 跳数



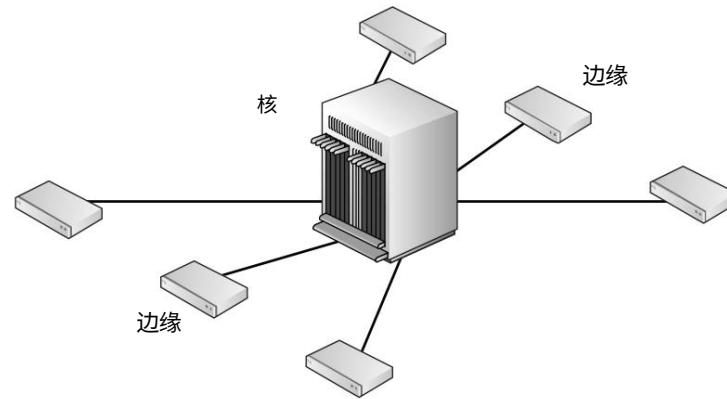
Because of the popularity of the core-edge design, the alternative lack of
要是因为这一经典设计的流行以及在现实世界中相对容易实现，所以有理由认为提出替代结构的
原因。因此，如果希望实施星型或环型网络设计，那么必须花更多时间去仔细考虑而不能草率地决定。

在介绍某些拓扑之前，需要指出的是，在提到结构拓扑时，我们仅指交换机和 ISL。将终端设备（例如主机或存储阵列）从一个交换机端口移动到另一个交换机端口，甚至从一个交换机移动到另一个交换机，通常不会影响结构的拓扑。分层拓扑是一个值得注意的例外。

核心-边缘拓扑

核心边缘拓扑是迄今为止最广泛实施的拓扑，它将满足大多数要求。核心边缘结构看起来很像旧的轮辐式或星型网络拓扑。在结构的物理中心有一个核心交换机，每个边缘交换机通过 ISL 连接到核心交换机。图 5.16 显示了核心边缘设计，其中核心处有一个导向器，六个边缘交换机连接到核心。

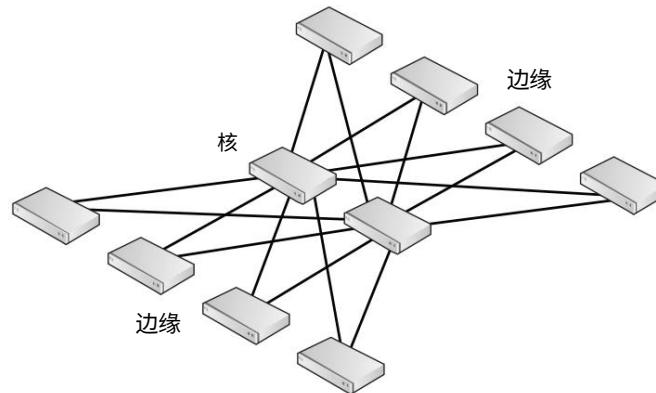
图 5.16 核心边缘拓扑



核心边缘设计自然适合刀片服务器环境,因此小型支持 SAN 或 FCoE 的交换机嵌入在刀片服务器机箱中。然后,ISL 从这些嵌入式边缘交换机运行到核心交换机,存储直接连接到该核心交换机。

对于核心边缘设计来说,每个结构拥有多个核心交换机是完全可以接受的,如图 5.17 所示。

图 5.17 具有两个核心交换机的核心边缘设计



就结构服务而言,核心交换机没有什么特别之处。然而,许多人选择强制将核心交换机作为结构中的主交换机,并选择从核心交换机执行大部分结构管理。

但情况不一定如此。不过,将核心交换机作为主交换机是一个好主意,因为它是最不可能重新启动的交换机。