中文图书分类号：TP39

密　　级：公开

UDC：004

学　校　代　码：10005

工程硕士学位论文

M.E. DISSERTATION

论　文　题　目：Xen 与 KVM 虚拟化方案的设计与性能评比

论　文　作　者：庄剑锋

领　　域：计算机技术

指　导　教　师：徐旭东　副教授

论文提交日期：2016 年 6 月 17 日

UDC：004　　学校代码：10005

中文图书分类号：TP39　　学　　号：G201207010

密　　级：公开

北京工业大学硕士专业学位论文

（非全日制）

题　　目：Xen 与 KVM 虚拟化方案的设计与性能评比

英文题目：THE DESIGNATION AND PERFORMANCE APPRAISEMENT BETWEEN VIRTUALIZATION SOLUTION OF Xen AND KVM

论 文 作 者：庄剑锋

领　　域：计算机技术

研 究 方 向：计算机软件技术

申 请 学 位：工程硕士专业学位

指 导 教 师：徐旭东 副教授

所 在 单 位：计算机学院

答 辩 日 期：2016 年 05 月

授予学位单位：北京工业大学

独 创 性 声 明

本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京工业大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签　　名：庄剑锋

日　　期：2016 年 6 月 17 日

关于论文使用授权的说明

本人完全了解北京工业大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

（保密的论文在解密后应遵守此规定）

签　　名：庄剑锋　　日　　期：2016 年 6 月 17 日

导师签名：徐旭东　　日　　期：2016 年 6 月 17 日

摘 要

虚拟化是构建云基础架构不可或缺的关键技术之一。所以，在云计算的时代背景之下，支持云计算服务的最底层、最基本的虚拟化技术发展非常迅速，潜力巨大。虚拟化就是指运算单元是在虚拟的基础上而不是在物理的基础上运行，是一个为了简化管理、优化资源的解决方案。通过引入一个新的虚拟化层，对下管理真实的物理资源，对上提供虚拟的系统资源。虚拟化可以带来许多优点，避免物理资源的闲置和浪费，提高物理资源的使用效率，降低 IT 资本投入、节省各种资源，而且可以加强管理者对自身 IT 资源的管理，提高系统的安全性及可靠性，这使得虚拟化正在全面地改变 IT 架构的管理、网络、存储、操作系统、安全以及应用程序等方面的实现方式。

目前已经有了非常多的虚拟化方案可供选择，其中开源虚拟化方案中，最流行的是 Xen 和 KVM，但是这两种虚拟化方案在传输协议、部署方法、用户体验和安全性等方面都有各自的优缺点；在平台管理、资源分配、虚拟机迁移和运维方式上都有所不同，重要的是其相关的测量数据也不多。这使得很多人，很多企业都无法抉择应使用哪一种虚拟化方案、哪一种虚拟化方案更适合当前对生产环境的需求，更无法确定在应用领域的未来，哪一种虚拟化方案能够提供更加稳定、长久的服务性能。

因此，本论文主要是研究基于 Xen 和 KVM 的开源虚拟化平台来设计用于企业实际生产环境的虚拟化方案，并设计了一系列有效的、实际的测试方案。藉由以上数据，深入去对比研究 Xen 与 KVM 的虚拟化性能，以提供实际的参考意义，从而帮助使用者在面对开源虚拟化解决方案时，能够依据不同的需要做出快速准确的抉择。

关键词：Xen；KVM；虚拟化性能测试

Abstract

Virtualization is an indispensable essential technology to establish Cloud Infrastructure. So as the primary element and fundamental supporting of cloud computing, the virtualization technology has been well developed with great rapidity and has been provided with greatly potential under the era of cloud computing. The definition of virtualization is that the computing element be operated on virtual hardware instead of reality hardware. It is highly excellent solution that for simplifying supervision and optimizing the resources. It could be supervise veritable physical resources and provide virtual system resources via implement a virtualization layer. There are various types of merits for implementing virtualization, for instant, refrain from idle and waste of physical resources, improve the service efficiency of physical resources, reduce the capital contribution of information technology, economize lots of variety of resources, and the most one, to help administrators enhance to supervise their own respective information resources, improve the system security and dependability. All above of merits make the virtualization comprehensively alter supervise, network, storage, security, operating system and applications of implementation model for information technology infrastructure.

Nowadays, there are numerous availability virtualization solutions could be implemented. Among various of solutions, the Xen and KVM are the most popular virtualization solutions of open source. But each them has its own respective merit and demerit on aspect of transport protocols, implement method, user experience and security. Also there are some differences of platform management, resource allocation, virtual machine migration and operational way between Xen and KVM. The critical matter is that has not sufficient relative metrical data about the above aspect. So that many people and many companies could not make an affirmative choice about which virtualization solution is more suitable for implementing on the current production environment. Also could not affirm about which virtualization solution is able to provide more stabilization and more permanently service performance.

Therefore, this dissertation mainly studies to design virtualization solution that be able to implement on actually product environment of enterprise, which based on the open source virtualization platform of Xen and KVM, and also mainly study and design a series of effective and practical testing scheme. It is highly and deeply pay attention to do the virtualization performance comparison of Xen and KVM through the above data. Sequentially to provide a practical reference significance, and furthermore to provide a assistance for making a speedy and accurate choice for open source virtualization solution according to the different varieties of demands, and the fact that is the indispensable and essential guarantee procedure of practical

implementing.

Key words: Xen; KVM; Virtualization performance testing

目 录

第 1 章 绪论

1.1 研究背景

在计算设备以及互联网技术极其普及的今天，从技术的热点上看，虚拟化技术无疑是其中最热门的 IT 技术之一，而云计算同样也是最热门的概念之一。所以，由于云计算的强烈需求，支持云计算服务的最底层、最基本的虚拟化技术发展非常迅速，潜力巨大。由于虚拟化近几年的迅猛发展，其在商业领域的应用方面的优势也日益显露，其不仅能够降低 IT 资金投入、减少物理资源的闲置和浪费，增加物理资源的利用率，同时还能强化 IT 资源的管理，从而增强系统的可靠性和安全性。这使得虚拟化正在全面的改变原有 IT 架构的管理、安全、网络、存储、操作系统和应用程序等方面的实现方式，目前已有非常多的虚拟化方案可供选择。面对如此之多的虚拟化方案，究竟应该使用哪一种虚拟化方案、哪一种虚拟化方案更适合当前对生产环境的需求，在应用领域的未来，哪一种虚拟化方案能够提供更加稳定、长久的服务性能？所以如何对虚拟化方案的做出正确且具有实际意义的性能评测便成了最重要的问题，这也是公司项目中我的最核心的任务。

1.2 研究现状

由于虚拟化近年来取得了飞速的发展，通过虚拟化技术，数据中心无需增加服务器的硬件投入，即可大幅提升整体性能。因为虚拟化技术能够实现整合服务器，从而可以在一台服务器上安装多个不同或同一操作系统，并以此为根本，安装使用多个应用程序，即使用最少的服务器得到最大的整体性能，提高服务器的利用率。在虚拟化实施到 X86 服务器上后，加之服务器整合需求的不断增加以及 CPU 性能的不断提升，使得该技术拥有高效、节能等多种优势。事实上，很多公司都因为实施了服务器虚拟化而取得了在硬件成本、管理成本、人力成本、系统性能等方面的巨大收益。

然而，虚拟化的应用与相关性能测试的完备和其具有的实际意义是密不可分的。因为，在挑选方案时，需要斟酌的关键因素是：虚拟化的功能和性能，功能是实现虚拟化的基础，而性能是虚拟化效率的指标，即使某虚拟化技术具有非常强大的功能，但是如果其性能不好，很难想象将其应用到生产环境后效果会不会弊大于利，所以具有实际意义的性能评测可以帮助更好地了解虚拟化，从而更深入的对多种虚拟化方案进行比较，以降低实施应用的风险，事实上，这也会推动

虚拟化市场的发展。

在选择虚拟化方案的时候，虽然已经有了一些性能测试方法，但仍或多或少的存在一些弊端，显得不是很成熟和完善。如（1）大多最大化保留了虚拟化平台上的客户系统实例的纯净，即并没有在客户系统实例上部署目前主要的工作负载，在此基础上进行宏观的性能测试（CPU，内存等），并得出评估报告。（2）即使是在虚拟实例上部署工作负载，但进行的还是宏观的性能测试，即使得出测试结果，但这些测试结果在虚拟化平台的隔离、可扩展性等方面能体现出多大的实际意义就不得而知了。

此外，SPEC（Standard Performance Evaluation Corp，标准性能评估机构，是一个全球性的、权威的第三方非赢利性组织），曾计划针对虚拟化应用出台一套简洁、有效的衡量标准，以便作为虚拟化解决方案选型的依据。但到目前为止，该机构仍然没发布相关的衡量标准。可见在虚拟化性能测试领域，虽然目前有一些针对各个虚拟化软件的性能分析工具，也有一些衡量虚拟化系统中单个方面性能的基准测试工具，但还没有一个比较权威的专门针对虚拟化的性能测试工具能够集成所有性能测试于一体，因为虚拟化性能测试涉及到计算机系统的各个方面，而且没有一个标准化的性能测试工具，因此虚拟化性能测试与性能分析是一个比较具有挑战性的工程研究领域。

1.3 本文主要研究内容和意义

虚拟化技术是当今企业环境中重要的构筑单元，从虚拟化诞生发展至今，已经有很多的技术和产品种类，包括传统的老牌虚拟化软件 VMware、Microsoft 的 Hyper-V、Citrix 的 XenServer/XenClient、Oracle 的 VirtualBox 等，还有在 Linux 平台上两个著名的开源虚拟化技术 Xen、KVM。尽管 Xen 与 KVM 的市场占有率不如老牌的 VMware 虚拟化技术，但开源的好处在于可以有人数庞大的开发社区作为支撑，同时开放源代码也有利于人们研究和学习虚拟机的具体实现，且无需购买许可协议，也无需担心因为“盗版”而遭到诉讼和索赔，有时购买许可是非常昂贵的。所以本文，也同时应公司项目要求，采用的是最流行的两种开源虚拟化解决方案：Xen、KVM。由于这两种虚拟化方案在传输协议、部署方法、用户体验和安全性等方面都有各自的优缺点；在平台管理、资源分配、虚拟机迁移和运维方式上都有所不同，所以我们在相同的底层硬件架构，以及相同的生产环境下，通过实际测试，在稳定性、安全性、可扩充性、响应速度等方面，藉由测试数据，深入去对比研究 Xen 与 KVM 的虚拟化性能，并得出最佳虚拟化方案结论，以提供实际的参考意义。

由于目前还不存在一套基准性能测试方法，来度量哪一种虚拟化的性能更加优越，所以本文的研究内容便在于通过在不同的虚拟化环境中，部署实际的工作负载，并进行详细的测试，以达到得出虚拟化性能优劣的目的。测试方法主要包括：

（1）宏观的整体性能测试。虚拟化性能测试包括的范围比较广泛，可能包含虚拟实例动态迁移时的性能，也可能包括多种物理平台上的性能，还可能需要考虑多个虚拟实例运行在同一个宿主机上时的性能，且在虚拟客户系统实例中应用程序的数量和类型都非常多，如：文件存储，Web 服务，Mail 服务等。尽管这些应用程序对系统的使用特点都不同，但它们几乎都会使用到 CPU、内存、网络、磁盘等基本的子系统，所以分别对这些子系统进行详细测试，就可以得出相应的性能指标。所用的两个虚拟实例工作负载分别为文件服务器和邮件服务器。除了上述两个虚拟实例之外，还有需要在一个非虚拟化环境中的物理主机上，安装原生的操作系统，工作负载与虚拟实例一致，并在此基础之上运行与虚拟实例同样的性能测试，以此作为评测基准。基于实际环境的意义就在于能够给出相同环境下实际的参考意义。

（2）性能隔离度测试。利用虚拟化技术，可以在同一台物理服务器上，部署出多个虚拟实例，每个虚拟实例均能够安装完整的操作系统，减少物理资源的闲置和浪费。虽然虚拟机具备一定的隔离能力，使得虚拟实例之间彼此互不影响。但是每种虚拟机的隔离度性能是不一样的，且目前的相关测试都或多或少的忽略这方面的重要性。即当多个虚拟实例被整合并运行在一个物理硬件之上，其中一台虚拟实例发生错误或使用物理资源较多时，是否会影响到整个系统，导致整个系统瘫痪。所以在企业环境中，虚拟化方案的性能隔离度，同样是考察的关键。因为我们绝对不希望在使用一个应用程序时，这个应用程序占用了全部的 CPU 和内存资源，导致无法并发运行其他的程序。

为了能让性能隔离度测试具有实际的参考意义，在分别运行单个 KVM 与 Xen 的整合堆栈单元时，需要部署两台虚拟实例以生成负载。其中工作负载是文件服务器和邮件服务器，这也是商业领域中虚拟化的最典型的应用。然后对没有压力的第三台虚拟实例和宿主机分别进行性能测试，并以此基准综合考量 Xen 与 KVM 的性能隔离度。其中对无压力的虚拟实例和宿主机都进行测试的意义在于考量虚拟机对各个虚拟实例的隔离性能如何，尤其是在虚拟实例占用物理资源（工作负载）较大的时候，是否会影响到其他的虚拟实例，甚至是整个系统（宿主机）。因为如果能够同时清楚硬件和软件在一定范围的配置下能够做什么，那么终端用户就会得的他们真正需要的，有意义的信息。

（3）可扩充性测试。前面提到虚拟化技术可以整合服务器，来实现用最少的服务器获得最多的整体性能，提高物理资源的利用率，节约硬件投入的成本，但同时会遇到这样的问题：在可预见的未来，在目前服务器硬件的能力范围内，部署多少虚拟实例能发挥其最大的性能？通过可扩充性测试，就可以得出虚拟化方案 Xen 与 KVM 是否具有可扩充性，可扩充性是好是坏。这可以帮助决定需要多大的服务器硬件来运行当前的企业级工作负载。所以在此项测试中，在 Xen 与 KVM 各自的虚拟机中以递增的方式部署客户系统实例，并针对原生主机进行相应的测试，根据测试结果就可以直观的说明可扩充性是优是劣。

1.4 本章小结

本章主要阐述了对虚拟化方案 Xen 与 KVM 性能测试的研究背景，本文的研究内容，以及目前国内外对虚拟化性能测试的研究现状，同时阐述了目前测试方案的优缺点。

第 2 章 虚拟化技术概述

2.1 虚拟化的定义

虚拟化的定义有很多，因为每个行业或个人对其理解并不相同，但是不管怎么定义，其本质都是将物理资源抽象成一个逻辑上的整体，再将其分割成若干个虚拟的物理硬件，所有的运算单元都运行在虚拟的基础上，而非真实的基础上，运算单元与虚拟的物理硬件通过接口进行交互。虚拟出来的物理硬件彼此独立，最终达到简化管理，增加资源利用效率，提高计算资源灵活性的目标。

2.2 虚拟化的特征与优势

虚拟化的主要目的在于充分利用物理资源，简化 IT 基础设施和资源管理方式，降低成本。为了整合资源和提高效率，同时出于对稳定性和兼容性的追求，不愿改变现有架构，则通过引入一个虚拟化层，通常称其为虚拟机监控器（Virtual Machine Monitor VMM），对下将物理资源抽象成多个能被单独客户系统申请使用的离散的虚拟副本，对上提供虚拟的系统资源。虚拟机监控器以宿主机的形式运行在真正的物理硬件上，而虚拟出来的平台称为虚拟实例，在此基础上就可以安装客户系统实例。虚拟实例把虚拟硬件当作真实的硬件来使用，并且彼此隔离，这也是虚拟化最主要的特征。其相关的对应关系，如图 2-1 所示。

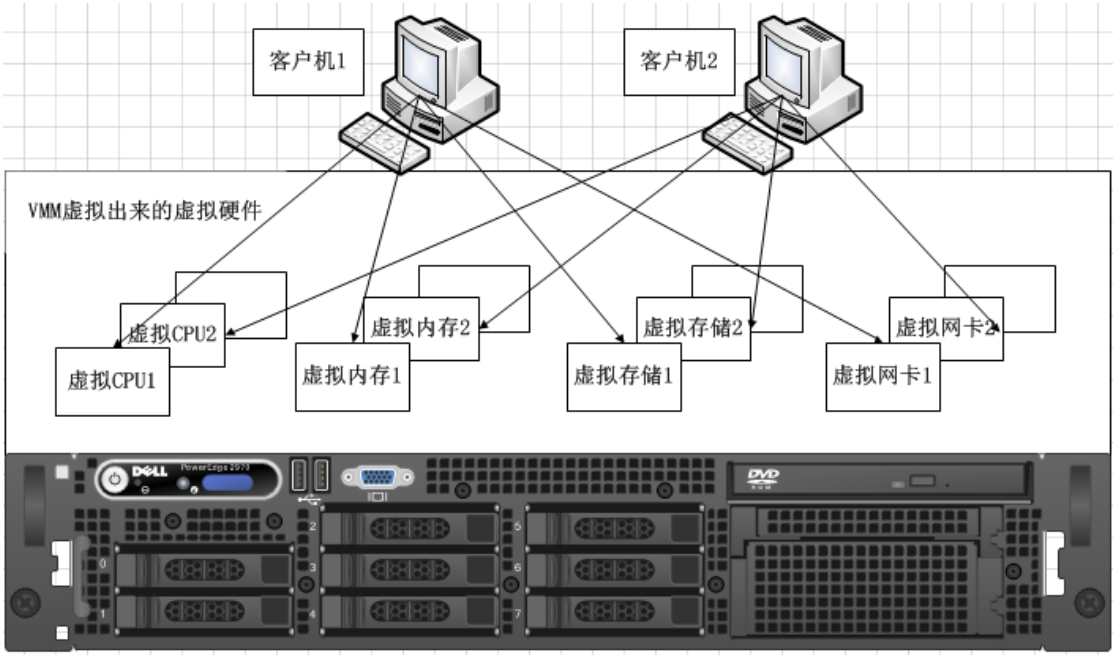


图 2-1 物理硬件、VMM 和虚拟实例的关系

Figure2-1 The relationship of hardware, VMM and virtual guest

自虚拟化问世以来，由于其拥有无可比拟的优势，使其迅速成为企业环境中必不可少应用技术之一。主要因其具有以下几点：

（1）不同的服务器之间由于需要不同的系统库，所以会存在兼容性问题，它们就不能运行在同一系统上。而虚拟化通过虚拟硬件的方式，从而解决了同一系统上的兼容性问题。

（2）操作系统只是简单的运行很多不同的应用程序，为了提高物理硬件的利用效率，使用虚拟机在一台硬件资源上部署多个操作系统，从而运行不同的服务器，而不必为一种操作系统就配备一台物理硬件。此外，由于虚拟出来的资源彼此互相隔离，所以当某一虚拟实例出现问题时，也不会对其他虚拟实例造成影响。甚至安全研究人员可以创造与主机或其他虚拟实例隔离的客户系统实例。研究人员可以在不影响主系统的情况下在这些客户系统实例上研究蠕虫、木马和病毒的影响。

（3）当服务器无法工作时，通过重新部署虚拟实例，虚拟化提供了更高的可用性。服务器环境包括多台物理服务器，它们每台上面都运行了一些客户系统实例。为了最有效的使用总体资源，客户系统实例可以从某一物理硬件无缝地迁移到其他物理硬件上来动态平衡负载。客户系统实例甚至可以在被使用状态下从某一物理硬件上迁移到其他硬件上，这也被称为动态迁移。

（4）在一个多宿主环境中，一个服务提供者可能在同一台物理机器上上运行多个分属与个人或企业的客户系统实例，每个客户系统实例不再需要和其他客户系统实例的所有者协调，它们可以拥有自己的管理员，选择要运行的软件，管理自己的客户系统实例。同样随着更多的应用被开发与多平台，多重引导配置已经不能满足开发人员的需求，他们需要更通用的多重平台功能，这一点上虚拟化的优势无疑更加明显，因为不再需要重启物理机器来切换不同的操作系统。

（5）提到虚拟化的另一个好处就是，虚拟化可以降低总成本（TCO），还可以节约潜在成本。如硬件采购，操作系统许可等。此外由于硬件设备占据了较少的机房空间，就能有更多的空间用于冷却，有时制冷也是一笔很大的开支。

2.3 虚拟化的类型

为了阐述虚拟化这个概念，提到了物理资源、计算资源等。在实际应用中，这些资源可以体现为各种各样的形式。如：将操作系统及其提供的系统调用作为资源，就体现为操作系统级虚拟化，也称为容器虚拟化技术。将 CPU，内存等作为资源，那么对应就表现为平台虚拟化。尽管虚拟化的分类有很多，但是以实现方式，实际应用和部署目的来看，其主要分为以下几大类。

2.3.1 平台虚拟化

平台虚拟化，主要是指针对服务器虚拟化和操作系统虚拟化。实现服务器虚拟化的重要一步在于，虚拟化层必须能够截获虚拟实例对物理资源的直接访问，并将其重定向至虚拟化资源池。

2.3.2 软件虚拟化

软件虚拟化，顾名思义，就是在缺乏硬件虚拟化支持的环境下，完全通过软件实现的 VMM 来模拟处理器的工作，从而实现截获虚拟实例对物理硬件的访问请求和重定向。如：CPU 指令是具有运行等级的，当宿主操作系统是以最高权限等级运行时，为了不出现越权等错误，则虚拟实例就不能也运行在最高权限等级上，但虚拟实例并不知道,仍然要执行最高级别的的指令，此时就需要工作在最高权限等级上虚拟机管理器 VMM 来捕获这种异常（无权限的指令），通过翻译重定向，返回指令给虚拟实例，让虚拟实例认为工作正常。由于这种转换中所有的指令都是通过软件模拟的，所以相应的性能就会比较差并且难于管理[1]。软件虚拟化的架构如图 2-2 所示。

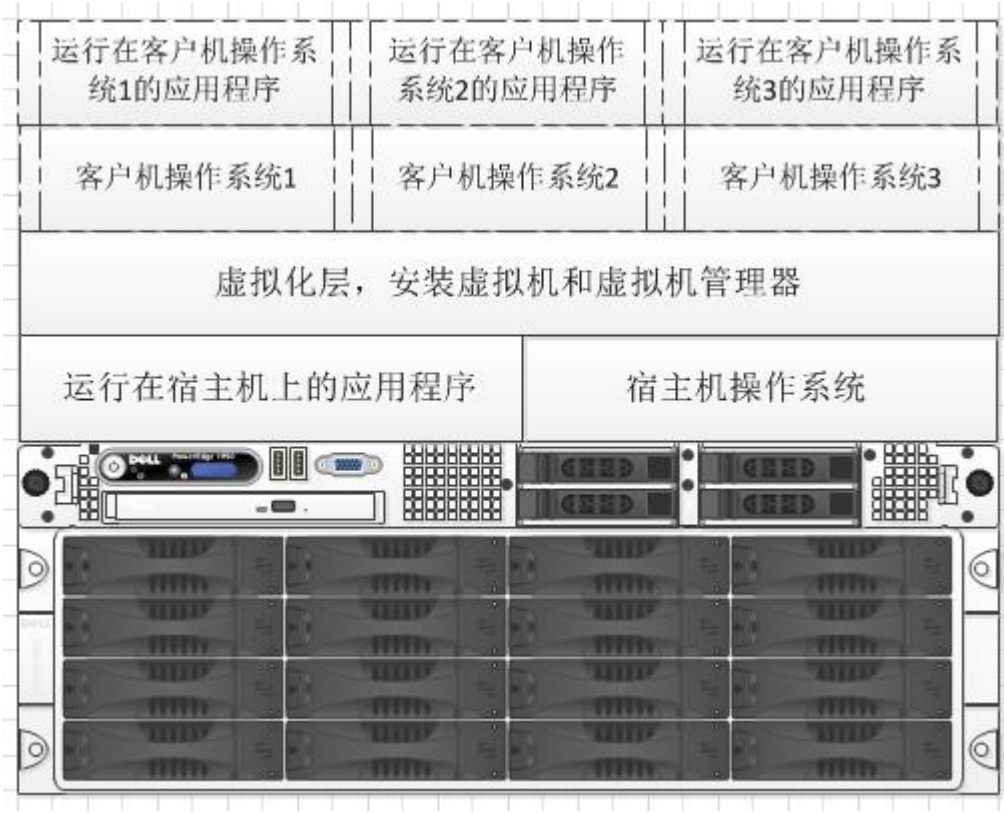


图 2-2 寄居架构的软件虚拟化

Figure2-2 Hosted architecture of software virtualization

2.3.3 硬件虚拟化

硬件虚拟化，其实就是指由硬件来实现捕获虚拟实例对物理硬件的无权限的指令这种异常请求，同时翻译重定向并返回正常指令给虚拟实例，带有特别优化过指令集的 X86CPU 可以直接控制这一过程。如此，虚拟机管理器就可以非常轻松地把虚拟实例放置在一种受控制的模式下运行。当虚拟实例尝试使用硬件资源时，CPU 就会暂时将虚拟实例处于停滞状态，然后将管理权限交还给 VMM 来做相应的安排。甚至，当虚拟实例尝试使用某些特殊硬件时，完全由 CPU 翻译重定向至 VMM 指定的虚拟硬件[2]。在这种硬件虚拟化机制下，整个虚拟实例对物理资源访问的截获和重定向过程不需要 VMM 的参与，从而性能大大提升。硬件虚拟化的架构如图 2-3 所示。

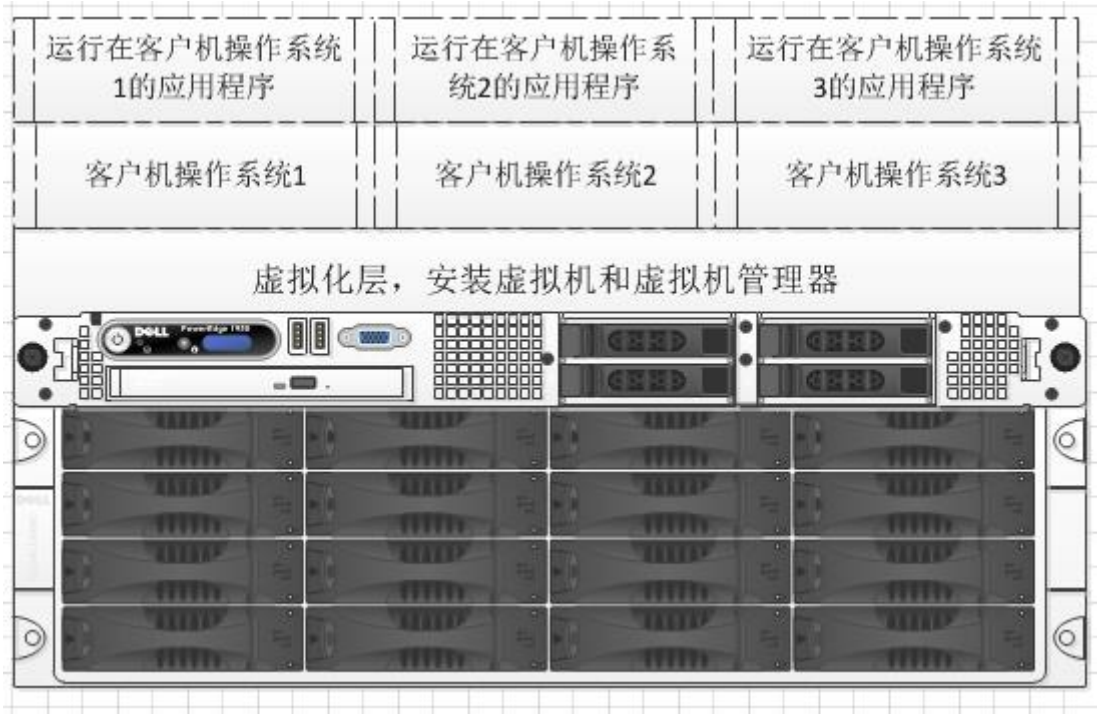


图 2-3 原生架构的硬件虚拟化

Figure2-3 Native architecture of hardware virtualization

2.3.4 准虚拟化

如刚才所述，在缺乏硬件虚拟化支持的环境下，软件虚拟化是完全依赖软件 VMM 来模拟处理器的工作，以监控并保证各个虚拟实例之间的独立和隔离，所以相应的性能较差，为了降低 VMM 的负担从而提高性能，就有了准虚拟化。准虚拟化思想就是弱化 VMM 截获虚拟实例的特殊指令，转为虚拟实例的主动通知。为了实现这一目的，就需要修改虚拟实例的内核，让其能够得知目前并不是运行

在真实的环境下而是虚拟环境下，从而可以和 VMM 互相配合地工作[3]。比如，VMM 提供一套改动过的虚拟硬件，且通过改动虚拟实例的内核，让其清楚目前运行在非 ring0 的级别，则当虚拟实例执行一些最高权限等级的指令时，就会主动通知 VMM。所以在准虚拟化中，极大的减少了 VMM 截获异常，翻译，重定向这一过程，性能大大提升。正是因为速度和不需要硬件扩展这一主要优势，使得以准虚拟化为解决方案的虚拟化产品早期取得成功的关键因素。由于在创建准虚拟化客户系统实例是需要访问操作系统的源代码并加以修改，这使得在部署准虚拟化时难度和时间都大大增加，并且对于那些不开放源代码的操作系统，如 Windows，就显得力不从心。需要注意的是，Windows 操作系统在企业中还是拥有非常高的地位的。准虚拟化架构如图 2-4 所示。

图 2-4 准虚拟化的架构

Figure2-4 Architecture of para-virtualization

2.3.5 全虚拟化

由于准虚拟化技术无法创建那些未经修改内核的客户系统实例，且修改内核的难度和时间都使得准虚拟化达到了发展瓶颈，事实上准虚拟化诞生之后，就几乎没有对自身做出多少优化。所以为了能够提高虚拟化软件的性能和实现各种不同的功能，自 2006 年开始，CPU 厂商 Intel 率先开始支持硬件虚拟化，其开发了一套新的硬件扩展功能，称为 Inter VT-x，是实现处理器虚拟化的硬件扩展，这也是硬件虚拟化的基础，专门用于实现全虚拟化技术并运行那些没有经过修改的

虚拟实例。次年 Inter 开发出与芯片组相关的 VT-d 技术，是从芯片组的层面为虚拟化提供必要的支持，通过它能够实现不经过中间介质而分配物理硬件给虚拟实例的功能，也能够处理内存访问重映射和 I/O 旁路转换缓冲。其中内存访问重映射能够防止一个直接内存访问越出虚拟机控制器的内存界限[4]。而 I/O 旁路转换缓冲是一个能够改善系统性能的缓存。

传统的非虚拟化中，操作系统内核运行于 CPU 所提供的多个权限等级中的 ring0 最高权限等级，而用户进程则运行于 ring3 最低权限等级。而在虚拟化系统中，由于 VMM 负责各个客户系统实例的运行状态控制和协调，因此 VMM 本身必须拥有最高的特权级别的访问控制权限，以保证虚拟实例之间的隔离和资源共享。所有的客户系统实例的硬件设备访问请求都必须通过 VMM 的安全检查，然后由 VMM 通过异步事件的方式转化为实际的设备访问操作。而所有的资源也是由 VMM 授权给客户系统实例使用的。尽管此时客户系统实例不再运行在最高权级下，但是为了方便控制用户进程的进行，还是需要保证客户系统实例内核的特权级别能够高于用户态程序的级别。为了解决虚拟化带来的特权级别分配的矛盾，在引入 Inter-VT 技术的 CPU 有 2 种模式，虚拟机扩展根操作模式（VMX root operation）和虚拟机扩展非根操作模式（VMX non-root operation）。这 2 个操作模式均具有 4 个权限运行等级，此时 VMM 运行在虚拟机扩展根操作模式下，而虚拟实例就运行在虚拟机扩展非根操作模式下。由于 VMM 负责实际访问物理硬件，给客户系统实例一种自己拥有硬件的假象，所以与准虚拟化不同，全虚拟化为虚拟实例供应了完备的 X86 平台。不需要对虚拟实例进行任何改动，便可以直接运行在全虚拟化环境中。全虚拟化架构如图 2-5 所示。

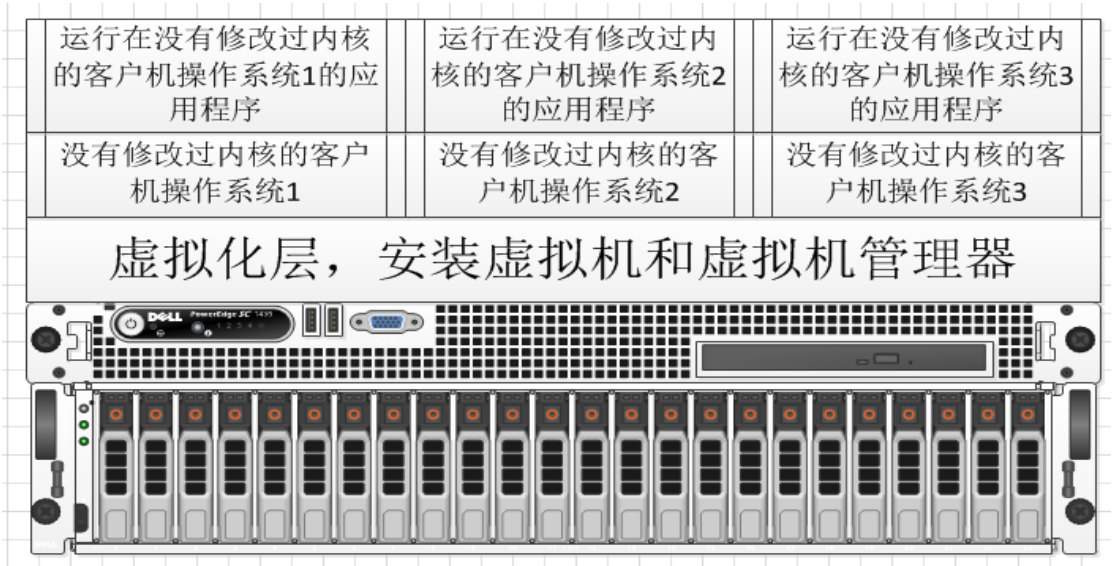


图 2-5 全虚拟化的架构

Figure2-5 Architecture of full-virtualization

2.3.6 操作系统级虚拟化

操作系统级虚拟化技术是一种轻量级的虚拟化技术，又称之为容器虚拟化技术。其没有虚拟机监视器，相反，虚拟化完全在一个传统的单个操作系统映像中完成。支持这种虚拟化的操作系统是一种提通用的分时操作系统，隔离命名空间和资源的能力更强。这种操作系统上创建的虚拟实例看起来仍然是拥有它们自己的文件系统，软件配置的单独机器。

操作系统级虚拟化的优点在于其需要较少的资源，这里的资源主要是指占用宿主机较少的物理内存。虚拟实例通常共享一些程序的用户空间，库甚至软件栈的副本。至少，这些同质的虚拟实例并不需要自己的专有内核，因为它们都是完全一致的二进制。在使用操作系统级的虚拟化时，每个虚拟实例需要的内存非常少，因为大部分这种虚拟实例都能适应给定的物理内存，所以操作系统级的虚拟化在那些对同时运行的虚拟实例可扩展性要求极高的场合非常有优势[5]。但是这里所指的虚拟实例与之前所指的虚拟实例本质上完全不同。单一的宿主操作系统，通过划分其特定部分，生成很多虚拟操作系统环境，，从而允许多个不同应用程序在一份操作系统内核实例的控制下隔离运行，无需为每一种互不兼容的应用程序创建一个操作系统实例，这也是操作系统级虚拟化的核心，即在应用程序与操作系统之间的加入一个虚拟层，将操作系统资源访问虚拟化。但需要注意的是，这里的虚拟实例只是一个紧耦合的用户空间进程的容器，而不是一个成熟的操作系统。

操作系统级虚拟化的缺点在于对虚拟实例的隔离不强，当一个虚拟实例使用较多资源时，其他虚拟实例的性能就会相应降低。由于虚拟实例不是一个成熟完整的操作系统，所以操作系统级虚拟化无法支持一些需要直接访问虚拟硬件或物理硬件的应用程序。且在每个虚拟实例都想运行相同的操作系统的情况下，选择操作系统级的虚拟化非常合适，但是对大多数用户来说，在同一台机器上运行不同操作系统是选择虚拟化的主要原因，显然操作系统级虚拟化的定义并不满足这一要求。操作系统级虚拟化架构如图 2-6 所示。

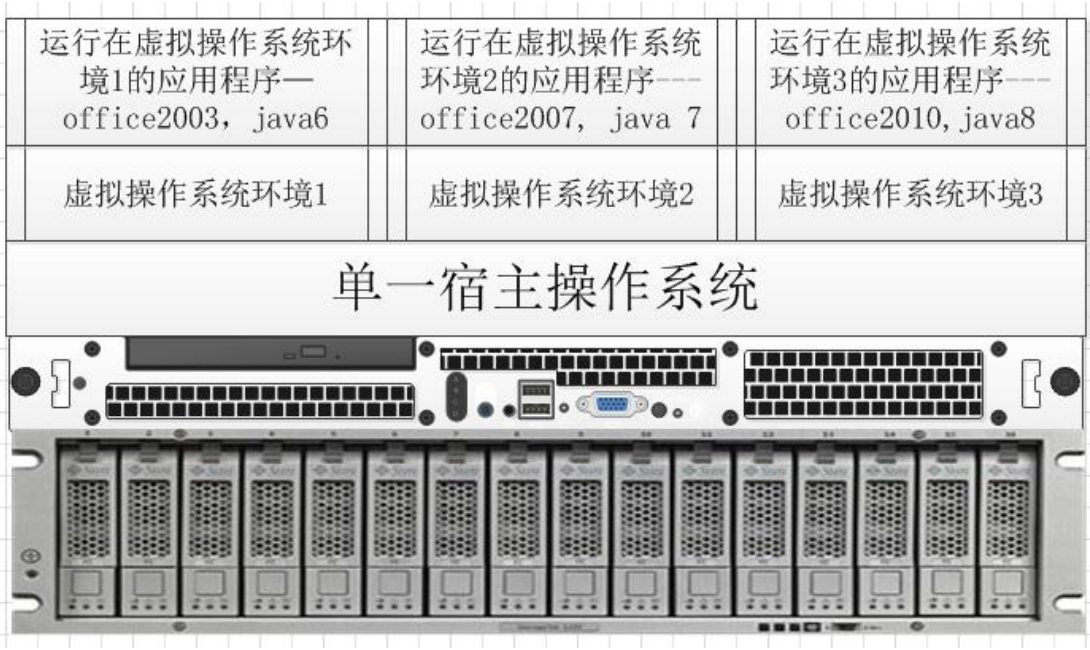


图 2-6 操作系统级虚拟化的架构

Figure2-6 Architecture of paene-virtualization

2.4 本章小结

本章说明了虚拟化技术的基本概念及其原理和本质，详细阐述了当前虚拟化技术的类型，虚拟化技术的优势，以及虚拟化和云计算的应用关系。

第 3 章 测试平台的设计与实现

3.1 数据安全

无论是单独的服务器亦或是集群，都是建立在硬件架构的基础之上。从数据安全的角度来看，没有人能够保证磁盘永远不损坏、数据永远安全。如果不给数据提供安全保障，就有可能遭到毁灭性的打击。2000 年底，中国最早的互联网个人主页 ChinaRen 就遭受了毁灭性的灾难，硬盘损坏，30 余万的个人主页一夜之间丢失殆尽，如果当时 ChinaRen 能够重视为数据提供安全的保障，ChinaRen 绝对不是今天这种局面[6]。所以在以数据为中心的信息时代，妥善有效地保护数据是核心问题之一。

3.1.1 磁盘阵列

人们可以忍受计算机宕机，操作系统或应用程序的重新启动，甚至硬件损坏，但是无法接受数据丢失。RAID 技术的出现，将数据安全推向了一个空前的高度，其中提高传输速率和提供容错功能是最大的优点。因为目前 CPU 的处理速度越来越快，而磁盘 I/O 速度又无法大幅度提升，此外，由于单个磁盘没有容错功能，无法保证数据的安全性。所以采引用 RAID 技术，在多个设备上同时读写数据，不但使得磁盘 I/O 的吞吐量得到显著的提升，而且具有非常完善的校验和容错能力，提升了系统的稳定冗余性。

RAID 是 Redundant Array of Independent，即独立磁盘冗余阵列。通过将多个独立的磁盘整合变为单一的虚拟设备，并根据区块大小，将数据进行等价分割，然后在写入各个磁盘上。所组成的阵列，从逻辑上来看只是一个单独的磁盘驱动器，在应用程序看起来与一块普通的磁盘一样，但却提供可以媲美大型的昂贵磁盘的存储性能和冗余性，即数据保护功能。甚至配合热插拔技术，能够在线更换故障设备并恢复数据，对实现不间断服务具有重要的意义。RAID 的实现方式有三种：

（1）外接式磁盘阵列柜，具有热交换功能，但是价格很贵，一般公司无法承受。

（2）内置磁盘阵列卡（硬件磁盘阵列），是目前主要实现方式之一，通过磁盘阵列卡来完成数组的目的，具有热插拔、在线扩容、动态更改阵列等级、主动执行数据重组、高速缓存等功能。因其拥有一个专门的处理器来控制各个磁盘，尤其是在计算同位检查码时，不会过度损耗 CPU 和内存的资源，所以磁盘子系

统的性能较佳。但同样价格不菲，此外操作系统必须要具有磁盘阵列卡的驱动程序，才能够正确识别磁盘阵列所产生的磁盘驱动器。

（3）软件磁盘阵列，即利用软件仿真磁盘阵列的功能，同样具有提高传输速度和冗余性，但因为需要依靠软件来仿真数组的任务，所以往往会损耗 CPU、I/O 总线等的资源。但是由于目前 CPU 的处理速度已经非常快了，所以速度限制已经不再是主要问题。

由于 RAID 等级选择的不同，而使得整合后的磁盘具有不同的功能，目前 RAID 等级及其功能主要有以下几种：

（1）RAID0，即磁盘分条技术。在这种 RAID 模式下，磁盘会被分割成等量的小区块，在文件要被写入 RAID 时，该文件也会被分割成符合小区块相等的大小，之后再依次写入到各个磁盘中，此外由于每个磁盘的容量会被加总到 RAID0 的总容量，所以磁盘的总容量也会变大。RAID0 的工作模式如图 3-1 所示。

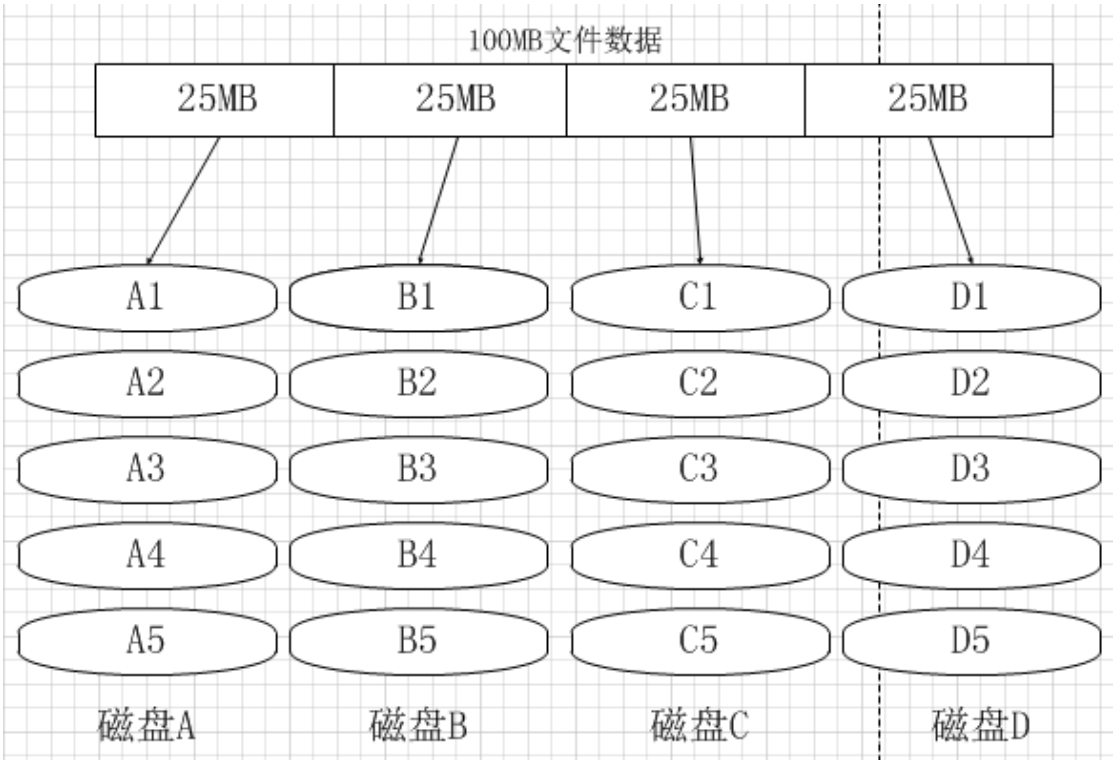


图 3-1 RAID0 的工作模式

Figure3-1 Operating pattern of RAID0

假设当有 100MB 的数据要被写入时，则每个磁盘会被分配到 25MB 的存储量，即原来需要写入的数据被分散到 4 块磁盘中同时进行读写，这种数据会被等量放置在各个磁盘上面的并行操作模式，使得在同一时间内每个磁盘所负责的数

据量就降低了，则整个磁盘的读写性能显著提升。因此最好用等量的磁盘才能让 RAID0 的效果最佳。如果使用容量不同的磁盘来组成 RAID0，当小容量磁盘的区块被用完了，那么所有的数据都将被写入到最大的那块磁盘中，此时性能就会变差，因为只剩下一个硬盘可以存放数据。RAID0 的磁盘 I/O 如图 3-2 所示。

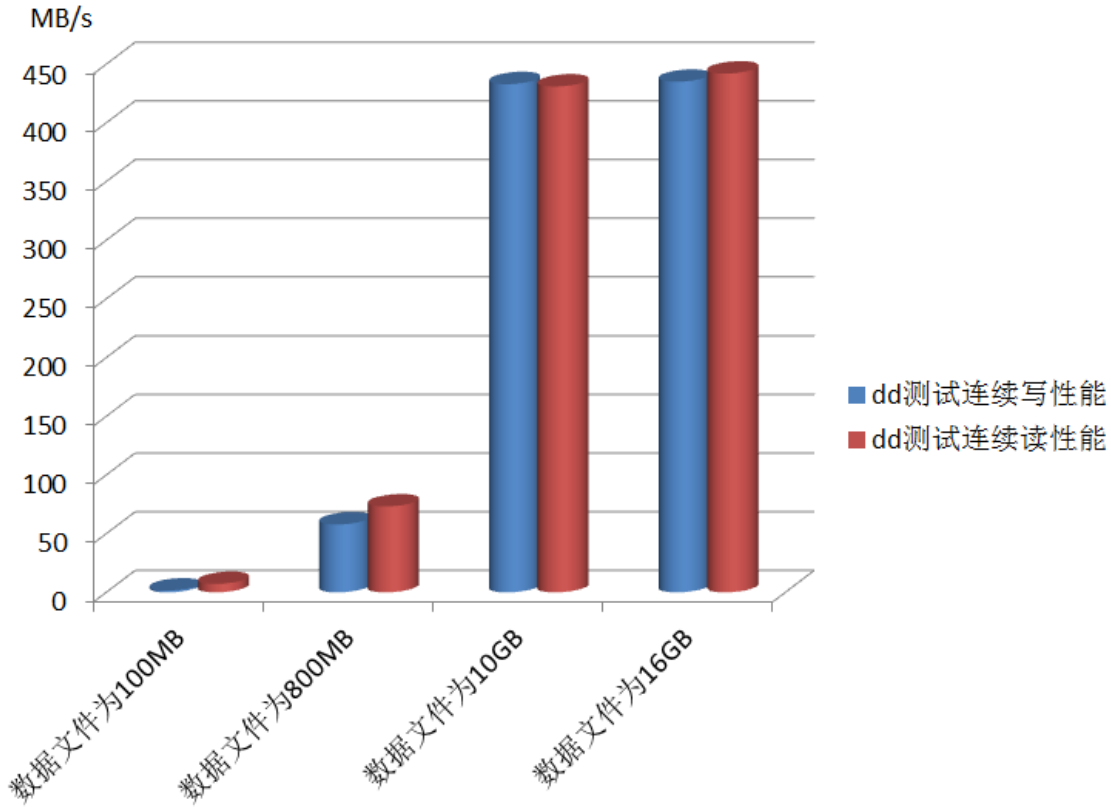


图 3-2 RAID0 的磁盘 I/O

Figure3-2 The hard disk I/O of RAID0

虽然 RAID0 可以提高磁盘的读写性能，存储空间利用率和整个服务器的磁盘容量，降低成本，但 RAID0 没有提供数据冗余功能或错误修复能力。因为每个文件都是被切割成适合磁盘区块的大小，然后每个磁盘会交错存放数据，此时构成阵列的任何一个磁盘出现故障，就会导致所有文件的数据缺少一块，则在 RAID 上面的所有数据都会丢失而无法读取，可靠性极差。在企业环境中使用 RAID0 来作为数据存储方案是非常不明智的选择，所以 RAID0 只适合用于不要求数据安全性，只要求读写性能的应用环境中。

（2）RAID1，是将某一磁盘内的全部数据，以镜像文件的方式备份至其他磁盘上，即让同一份数据完整的保存在多个磁盘上面，是对数据进行 100%的备份，故称之为磁盘镜像。如当有一个大小为 100MB 的文件需要写入，且有 4 个磁盘组成的 RAID1 时，那么这四个磁盘将会同步写入 100MB 到各自的存储空间去。当阵列中某三个磁盘发生故障时，系统会自动忽视这些故障磁盘，并在正常

的磁盘上读取和存储数据，因此只要阵列中至少有一个磁盘正常工作，系统就不会出现灾难性的故障，具备极强的冗余能力。此外这种 RAID 模式同样最好使用相同容量的磁盘，如果是不同容量的磁盘，则总容量以最小的那一个磁盘的容量为准。RAID1 的工作模式如图 3-3 所示。

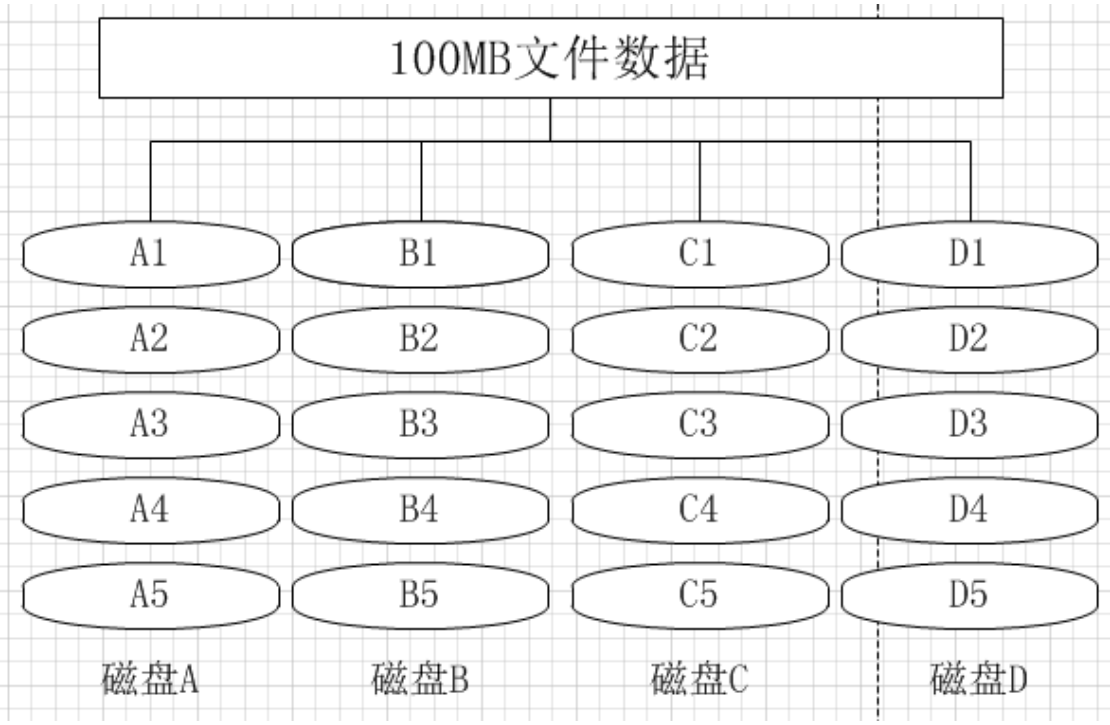


图 3-3 RAID1 的工作模式

Figure3-3 Operating pattern of RAID1

虽然 RAID1 对数据来讲是安全性是最高的阵列级别，不过由于同一份数据会被分别写入到其他不同的磁盘，所以当数据传送到 I/O 总线后会被复制多份到各个磁盘中，感觉就好像数据量成倍一样，因此在读写数据比较频繁和更换新磁盘后的镜像重组环境中，由于磁盘控制器的负载较高，所以其写入性能就比较差。此外由于整个磁盘设备有一半的容量要用于备份，因此磁盘利用率仅为 50%，这会导致成本明显增加。RAID1 的磁盘 I/O 如图 3-4 所示。

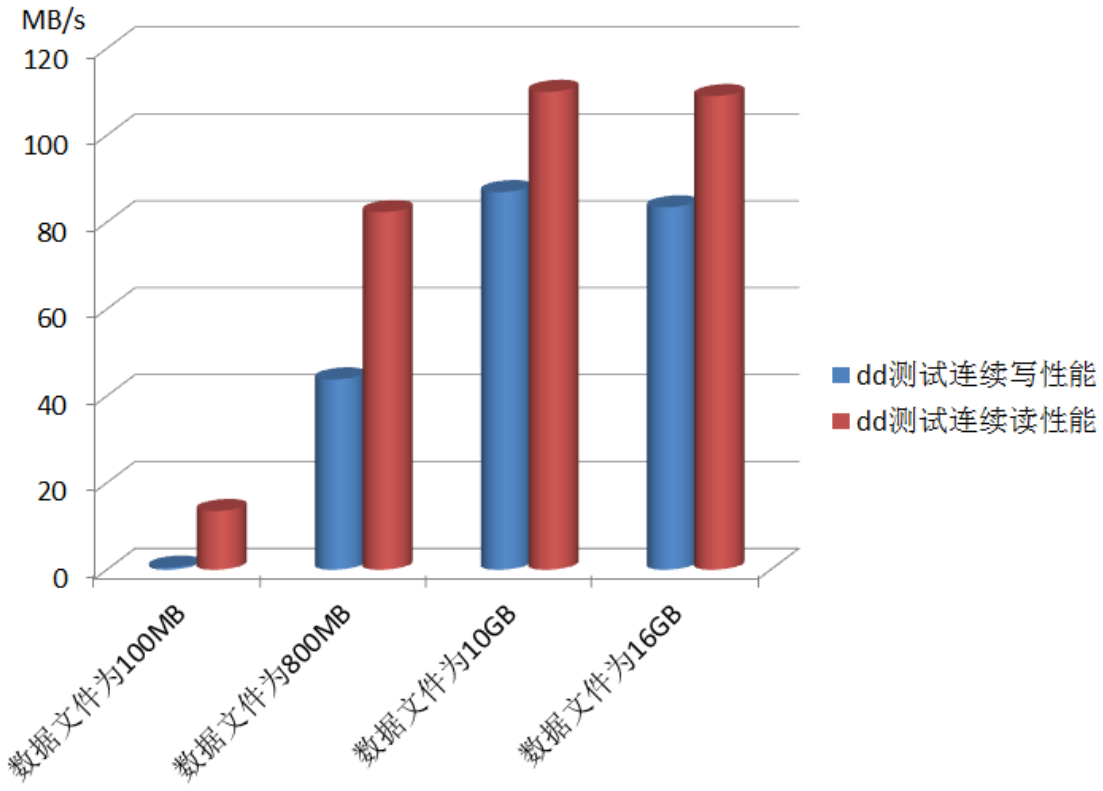


图 3-4 RAID1 的磁盘 I/O

Figure3-4 The hard disk I/O of RAID1

由于 RAID1 是数据的完全磁盘镜像，所以极大的保护了数据的安全性，且技术原理简单。但是相应的磁盘空间浪费较大、存储成本较高，所以多用于对数据安全性要求极高的应用环境中。

（3）RAID5，分布式同位校验的独立磁盘阵列。其数据写入的方式与 RAID0 类似，只是在每次循环写入的过程中，在每个磁盘中还加入了一个同位校验码，这个同位校验码会记录其他磁盘的备份数据，并且记录的同位校验码每次都记录在不同的磁盘上，因此该阵列中任何一个磁盘发生故障时，都可以通过其他磁盘上的校验信息和数据去推算出损坏的数据。但是由于同位校验码的存在，因此只会少 1 个磁盘的容量，磁盘利用率还是比较高的。RAID5 的工作模式如图 3-5 所示。

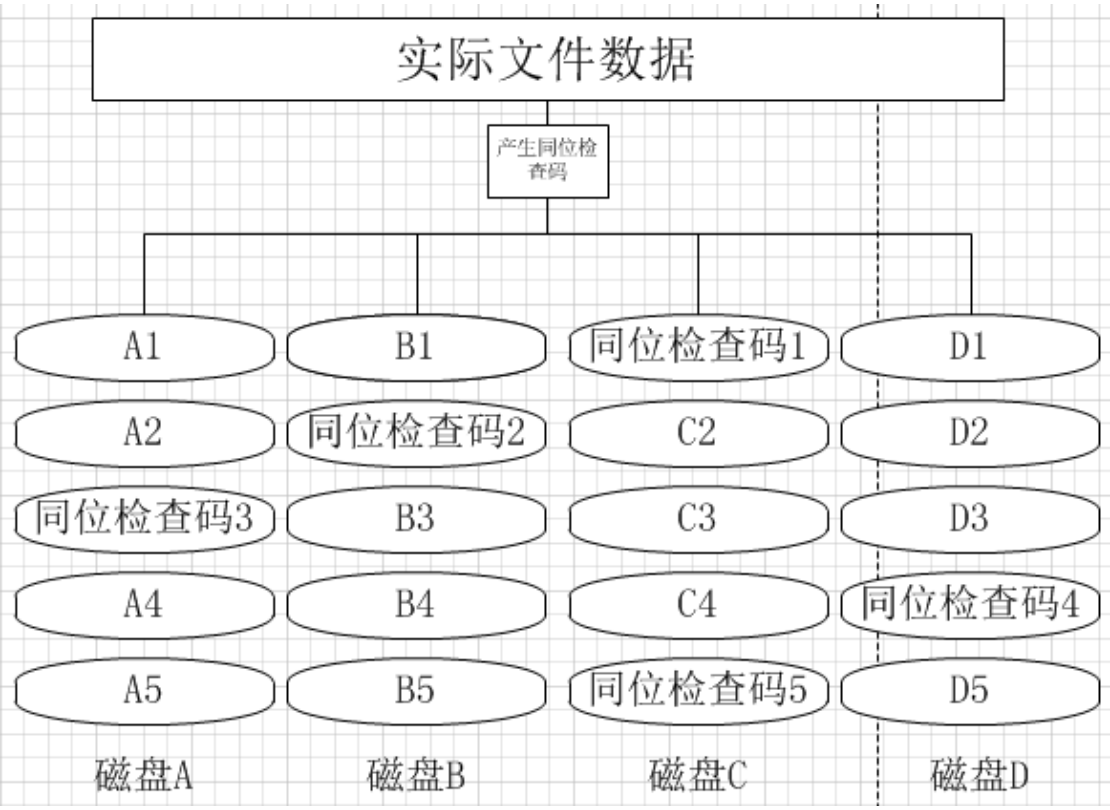


图 3-5 RAID5 的工作模式

Figure3-5 Operating pattern of RAID5

RAID5 是在每个磁盘上交叉地存取同位校验信息和数据，所以提高了数据的安全性。其中 Parity0 代表第 0 带区的同位校验码，Parity1 代表第 1 带去的同位校验码。RAID5 的读取性能很高，但是其写入性能表现一般，这是因为每次写入 RAID5 的数据需要经过计算同位检查码，由于加上这一层计算的操作，所以写入的性能与系统的硬件关系较大，尤其是利用软件磁盘阵列时，需要 CPU 来计算同位检查码，而不是靠磁盘阵列卡进行计算。RAID5 的磁盘 I/O 如图 3-6 所示。

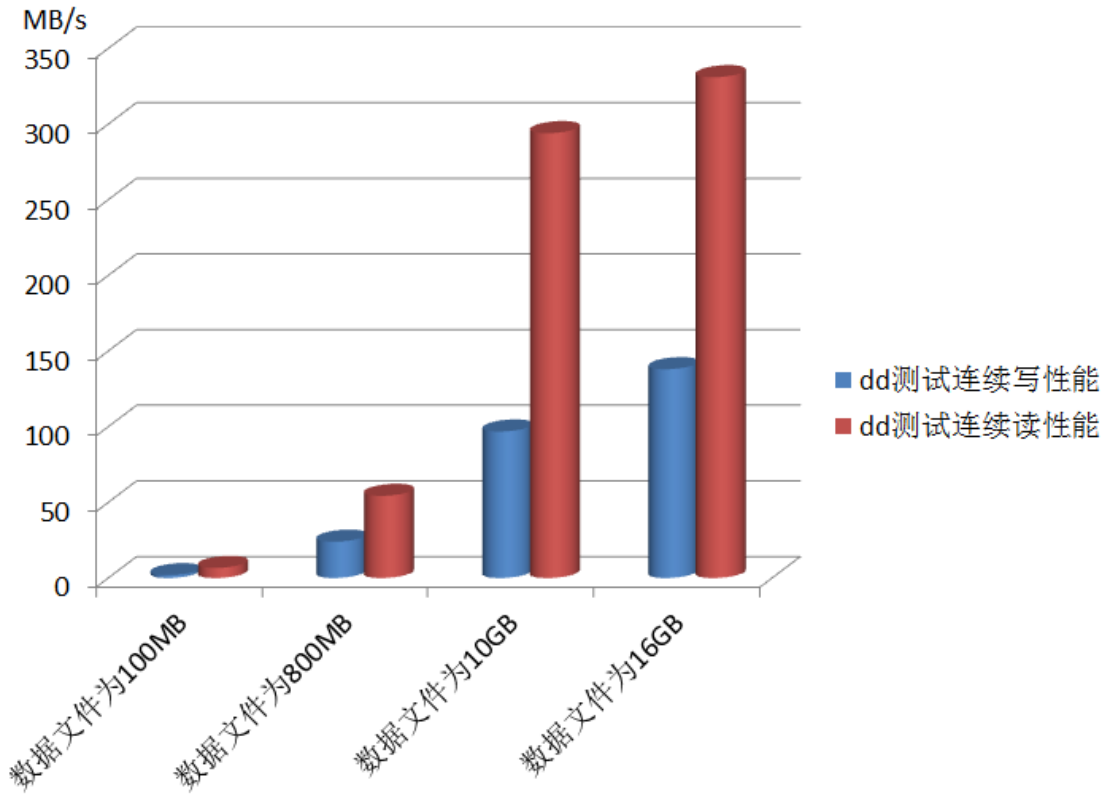


图 3-6 RAID5 的磁盘 I/O

Figure3-6 The hard disk I/O of RAID5

RAID5 可以看作是 RAID0 和 RAID1 的折中方案，性能和数据备份的均衡考虑，在 RAID0 的基础上加了校验，但又不像 RAID1 那样对数据做 100%的备份。在一定程序上，既保证了数据的安全性，又提高了磁盘利用率，但缺点是有较大的写损失，尤其是文件大小比较大时，超过缓存大小，RAID5 无法在缓存中完成校验信息的计算，此时需要将数据和同位校验码信息同时写入阵列中，此时的写性能相比 RAID10 会急剧下降，且仅能支持一个磁盘损毁的情况，如果损毁的磁盘数量大于等于 2，整个阵列的数据就全部损毁了。此外在某一磁盘发生故障后，该阵列的读写性能就会大幅度降低，因为需要从同位校验信息和余下的数据来恢复损坏的数据。

（4）RAID6，拥有两种分布存储的同位校验码的独立磁盘阵列。由于 RAID5 仅能支持一个磁盘的损毁，所以发展出了这种新的阵列，它其实是对 RAID5 的扩展，与其相比，这种新的阵列占用更多的容量作为同位校验码的存储，并且两种独立的同位校验系统使用不同的算法，大大的提高了数据的安全性，即使两个磁盘同时失效也不会造成数据的损毁。但同时由于引入了第二个独立的同位校验信息，因此会减少 2 个磁盘的容量。RAID6 的工作模式如图 3-7 所示。

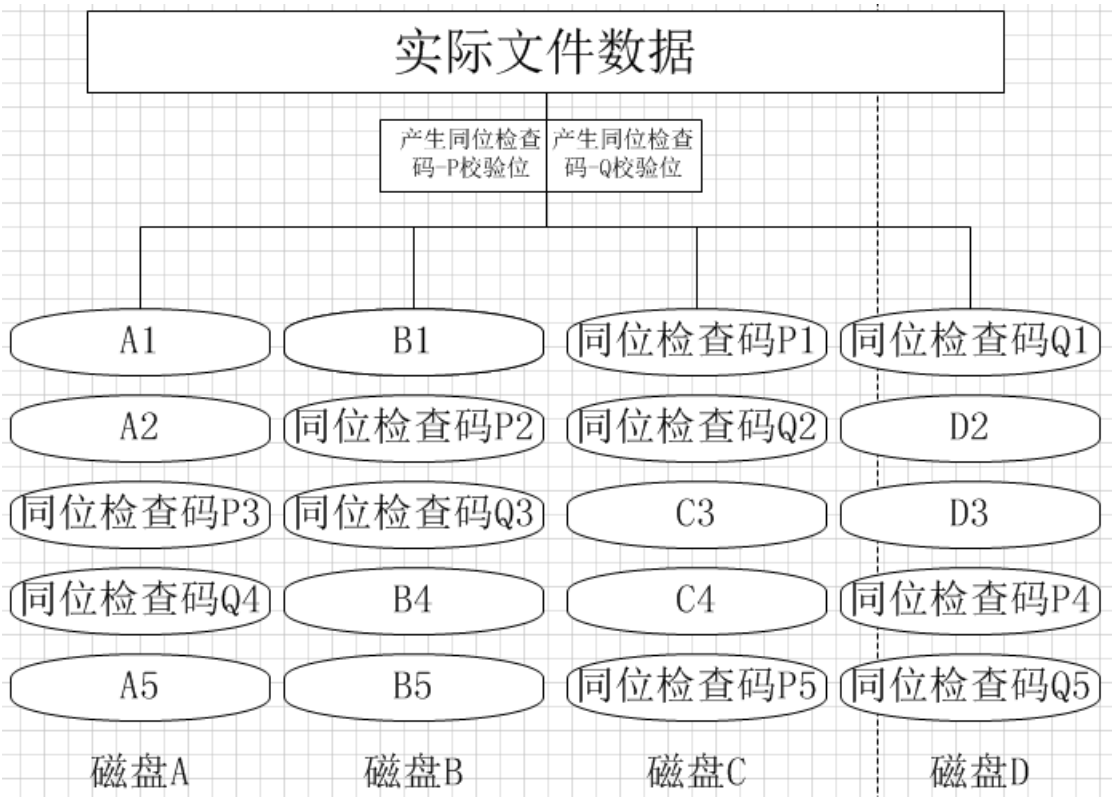


图 3-7 RAID6 的工作模式

Figure3-7 Operating pattern of RAID6

RAID6 同样不单独指定同位校验磁盘，而是在所有磁盘上交叉地存取同位校验信息的数据，其中 Parity0 和 Qarity0 代表第 0 带区的同位校验值，Parity1 和 Qarity1 代表第 1 带区的同位校验值。RAID6 有更大的写损失，因为每次写入 RAID6 的数据需要经过计算两种不同的同位检查码，造成验证数据正确性所花费的时间比较多，负载比较大。RAID6 的磁盘 I/O 如图 3-8 所示。

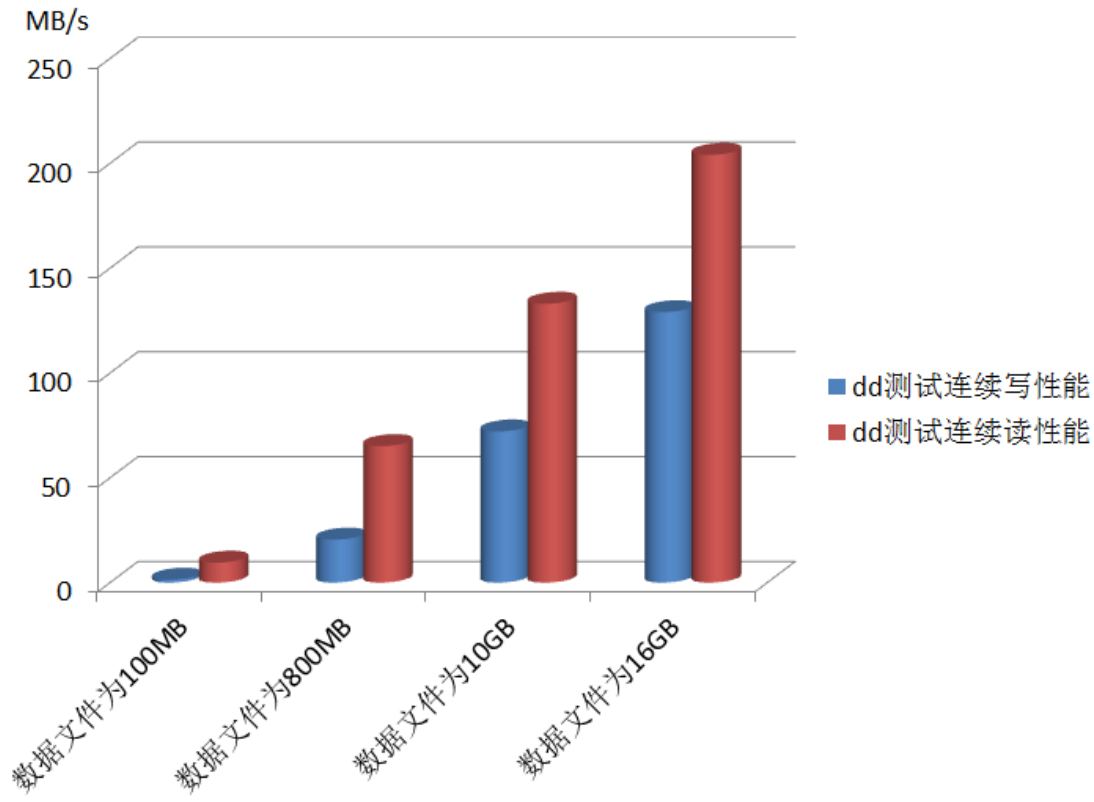


图 3-8 RAID6 的磁盘 I/O

Figure3-8 The hard disk I/O of RAID6

由于使用两种不同的同位校验系统，且彼此算法也不一样，RAID6 相对于 RAID5 有更大的写损失。

（5）RAID10，是一个既具有良好的存储性能，又确保了数据安全的方案。实际上其是将两种级别进行组合而得到的混合阵列，使该阵列不仅拥有 RAID0 出众的存储性能，还具备了 RAID1 杰出的数据安全性。其原理是先独立磁盘互作镜像，然后再将两套完整的镜像做磁盘分条。RAID10 的工作模式如图 3-9 所示。

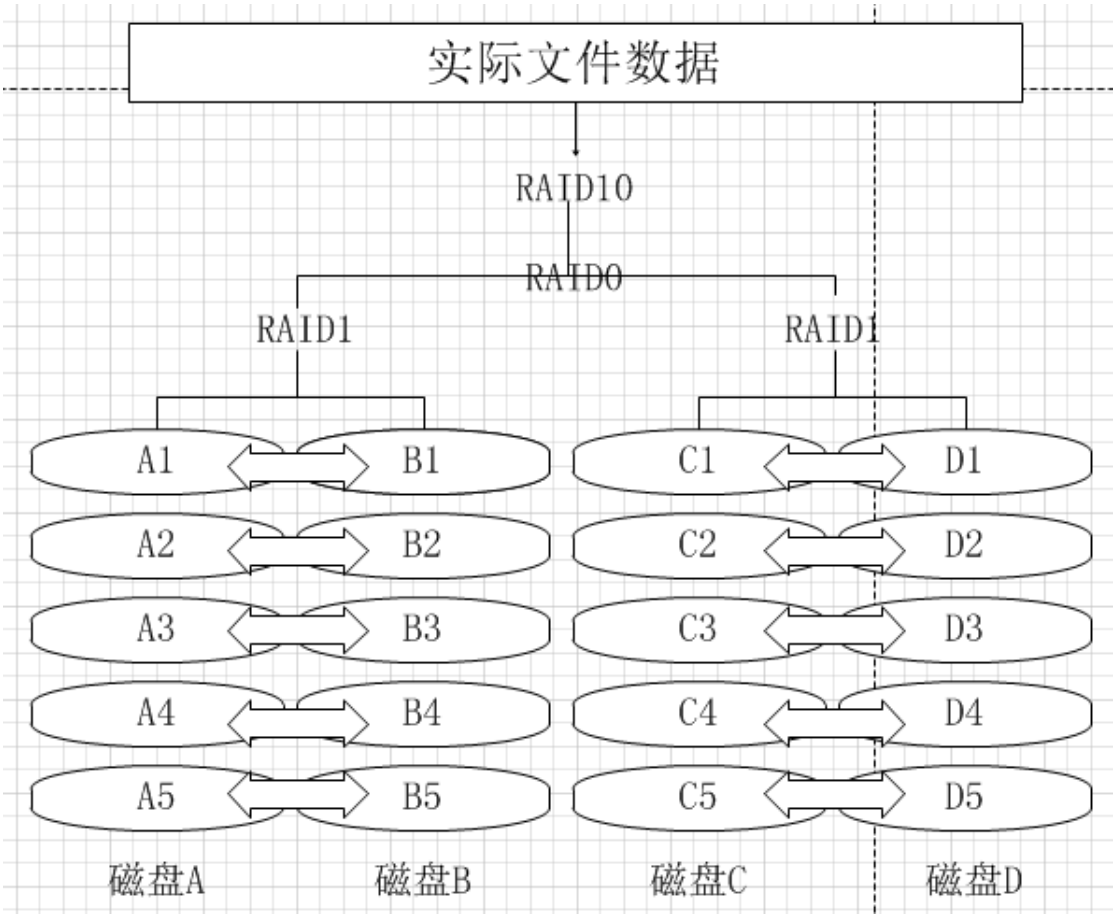


图 3-9 RAID10 的工作模式

Figure3-9 Operating pattern of RAID6

由于 RAID0 的优点，所以读写性能出色，由于具有 RAID1 的优点，当某一个磁盘损坏时，只要不是同组 RAID1 中的硬件损坏，整个阵列中的数据就不会受到影响，所以阵列失效率为 1/3，最大程度上保证了数据的安全性。但也由于 RAID1 的缺点，所以组建该阵列的成本比较高。但实际上，多投入一些磁盘成本既能提高系统性能，又能提高数据的安全性是非常值得的，因为数据是无价的。RAID10 的磁盘 I/O 如图 3-10 所示。

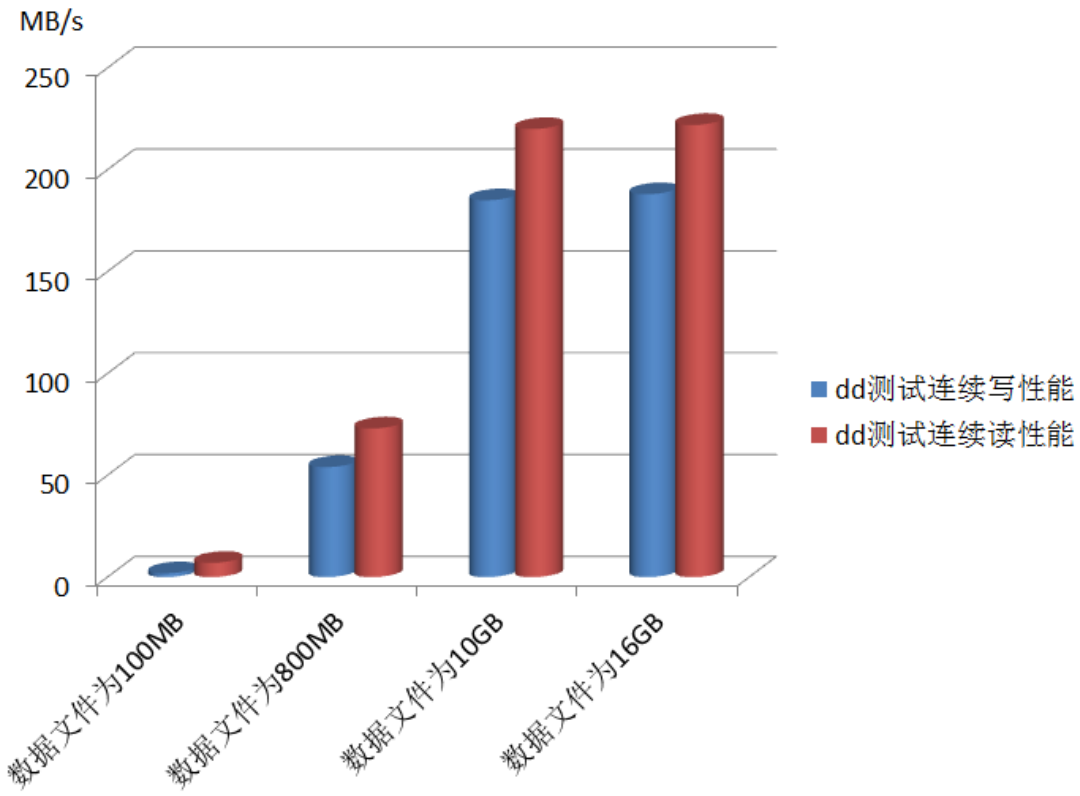


图 3-10 RAID10 的磁盘 I/O

Figure3-10 The hard disk I/O of RAID10

为了能够选择出在传输速率，容错性能和磁盘利用率这三个方面表现最佳的 RAID 级别，上面利用 dd 和 hdparm 对 RAID 的五种级别分别做出了磁盘 I/O 测试，而容错性能和磁盘利用率如图 3-11 所示。

阵列等级	阵列名称	支持Hot Spare	容错性能	阵列发生故障概率（n代表组成阵列的磁盘个数）	磁盘利用率	最小磁盘数
RAID0	磁盘分条	不支持	无	2n	100%	n≥1
RAID1	磁盘镜像	支持	极高	1/2n，只要阵列中存在一个磁盘可用，阵列就不会受损	1/n	2n，且n≥1
RAID5	分布式同位校验的独立磁盘阵列	支持	一般	n-1，最多支持一个磁盘损坏	(n-1)/n	n≥3
RAID6	带有两种分布存储的同位校验码的独立磁盘阵列	支持	高	n-2，最多支持两个磁盘损坏	(n-2)/n	n≥4
RAID10	磁盘分条再镜像	支持	高	1/(2n-1)，只要非同组镜像盘损坏，阵列不会受损	50%	2n≥4，且n≥2

图 3-11 各级别 RAID 的容错性和磁盘利用率

Figure3-11 The fault tolerance and disk utilization of the respective RAID

在进行阵列的读写性能测试时，由于目前的磁盘都带有缓存机制，所以在dd 的读取测试中，需要加入 iflag=direct 参数以绕过页面缓存，从而得到最真实的读取磁盘性能。此外在 dd 的连续写测试中，加入 conv=fsync 参数表示每次写入都要同步到阵列设备后才返回，而 oflag=direct 参数是为了使用直写的方式绕过页面缓存，以求反应最真实的阵列读写性能。

根据阵列的读写性能测试以及数据损坏的可能性，虽然 RAID10 的磁盘利用率不如 RAID5，但是当需要修改数据文件内容时，RAID5 必须先读出文件数据和校验信息，然后计算新的同位校验码，再将修改后的数据和新的校验信息写入，共发生了 4 次 I/O，但相比较之下，RAID10 在做同样操作的时候，只会发生读取文件数据和写入数据或镜像 2 次 I/O，所以无论是阵列读性能还是写性能，RAID10 都远比 RAID5 的速度快，且毋庸置疑，RAID10 的安全性高于 RAID5。所以 RAID10 既能提高系统的性能，又能提供数据的安全性，尤其是当系统既有大量数据需要存取，又对数据安全性有较高的要求时，RAID10 毫无疑问的是最佳方案。

3.1.2 LVM 与镜像文件格式

在商业领域的实际应用中，经常会遇到当初规划的主机某分区容量较小，导致该文件系统容量已满，无法继续使用。如根分区随着/var 和/home 目录的不断增大，很可能需要扩容，但是目前 mdadm 命令不支持 RAID10 阵列的扩容，即无法直接为 RAID10 扩展存储空间，除非备份数据，重建阵列，恢复数据。但由于主机已经处在运作中，不可能重建阵列，此时就需要采用 RAID+LVM 的方案，在 RAID 上面建立 LVM，以实现随时扩展存储空间而不需要重建整个存储架构。

LVM 的全称是 Logical Volume Manager 逻辑卷组管理器，它将多个物理分区或磁盘通过软件组合，使这些分区或磁盘表现成一个大的磁盘，再将这个大磁盘分区使之成为可使用的分区，最终挂载使用。同时还可以将其他物理分区或磁盘加入这个由 LVM 管理的大磁盘中。其重点在于弹性调整文件系统的容量，而不在磁盘读写性能和数据安全性上面。所以在 RAID 基础上建立逻辑卷组管理器，不失为一种最佳的数据安全和管理方案。

LVM 之所以能够对文件系统的容量进行扩充或缩小，最主要的原因是因为它的选项，尤其与 PE 选项有关。LVM 中的选项如下：

（1）Physical Volume，物理卷 PV。其处于 LVM 系统的最底层，其可以是磁盘分区，也可以是 RAID 设备。先将这些设备的 system ID 转换成 8e，然后再利用 pvcreate 将它们转换成 LVM 最底层的物理卷。

（2）Volume Group，卷组 VG。LVM 大磁盘其实就是将许多物理卷整合成一个卷组，这个卷组就像非 LVM 系统中的物理磁盘。可以利用至少一个或多个物理卷来创建卷组，并且在其创建之后，可以随意增加更多的物理卷或减少在其内的物理卷，且 LVM 允许存在多个卷组。

（3）Physical Extend，物理扩展块 PE。具有唯一编号的物理扩展块是 LVM 系统中可被寻址的最小存储单元，物理扩展块的大小在创建卷组时指定，且一旦指定就无法再次改变其大小。由于卷组最多仅能包含 65534 个物理扩展块，所以由卷组整合成的大磁盘所能达到的最大容量就基于物理扩展块[7]。物理扩展块默认为 32MB，那么此时卷组最大容量为 65534×32M=2TB。物理扩展块的作用就像 ext4 中的 block。

（4）Logical Volume，逻辑卷 LV。逻辑卷是在卷组的基础上分割转换出来的。逻辑卷与普通的磁盘分区类似，能够被格式化成相应的文件系统，卷组中剩余未分配的空间可用于创建额外的逻辑卷，创建逻辑卷后，可以随时对存储空间进行扩容或缩小。但是逻辑卷不可以随意指定大小，因为物理扩展块是最小存储单元，所以逻辑卷的大小就与物理扩展块的总数相关。LVM 的实现过程如图 3-12 所示。

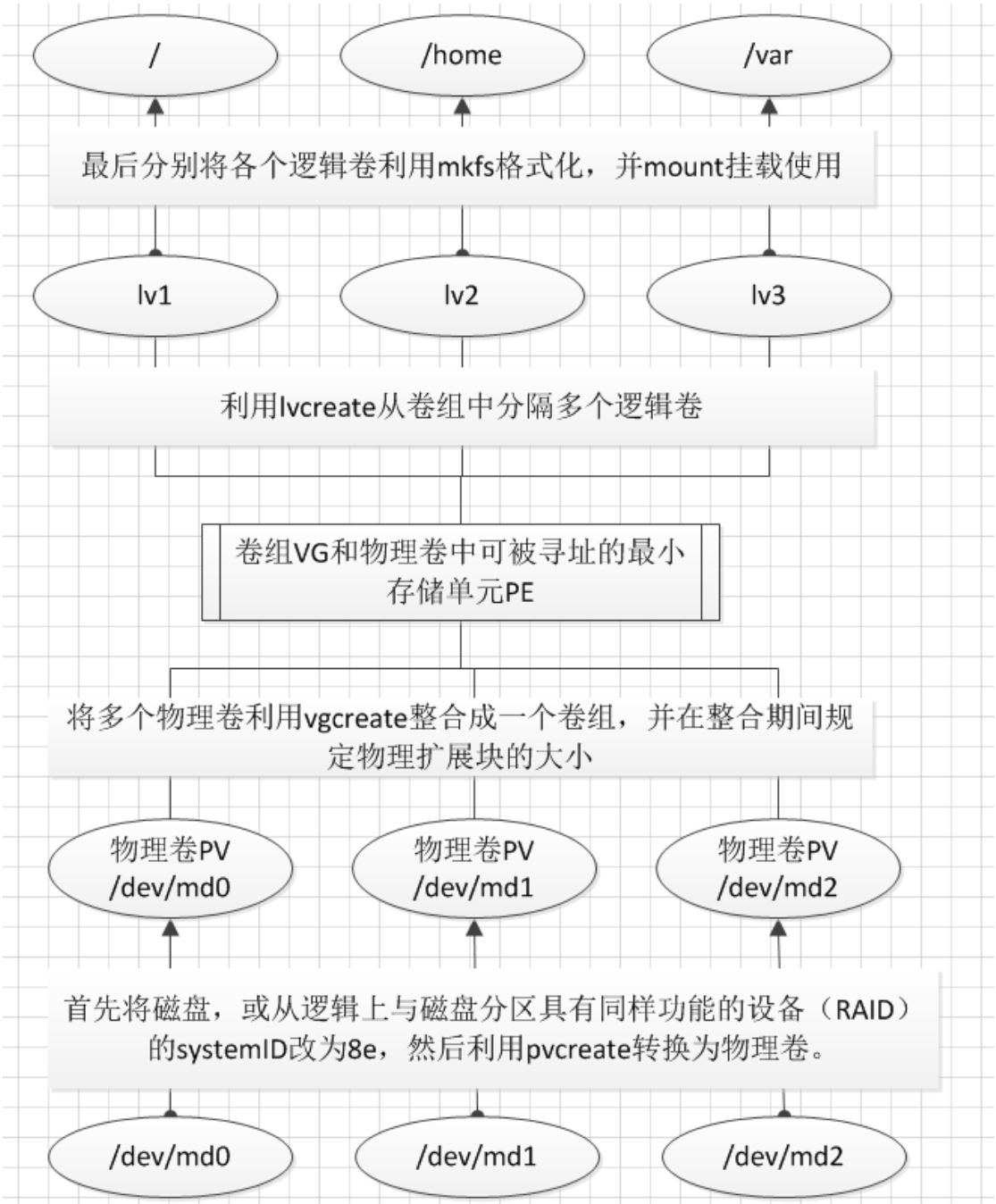


图 3-12 LVM 实现过程

Figure3-12 The process of LVM deployment

LVM 之所以能够弹性更改文件系统的容量，其原理是通过交换物理扩展块来实现的。将需要更改文件系统容量的逻辑卷中的物理扩展块转移到其他卷组中以降低该逻辑卷的容量，或将其他卷组中物理扩展块加入到此逻辑卷以增加空间。

在 RAID10 和 LVM 方案中，当存储空间不足时，就需要添加新的物理设备，但为了保证数据安全，新加入的磁盘也需要先组成 RAID 阵列，再加入 LVM，否则由于 LVM 的交错模式，一旦发生故障，会让数据遭遇灾难性的损毁。利用

LVM 实现磁盘扩容的实现过程如图 3-13 所示。

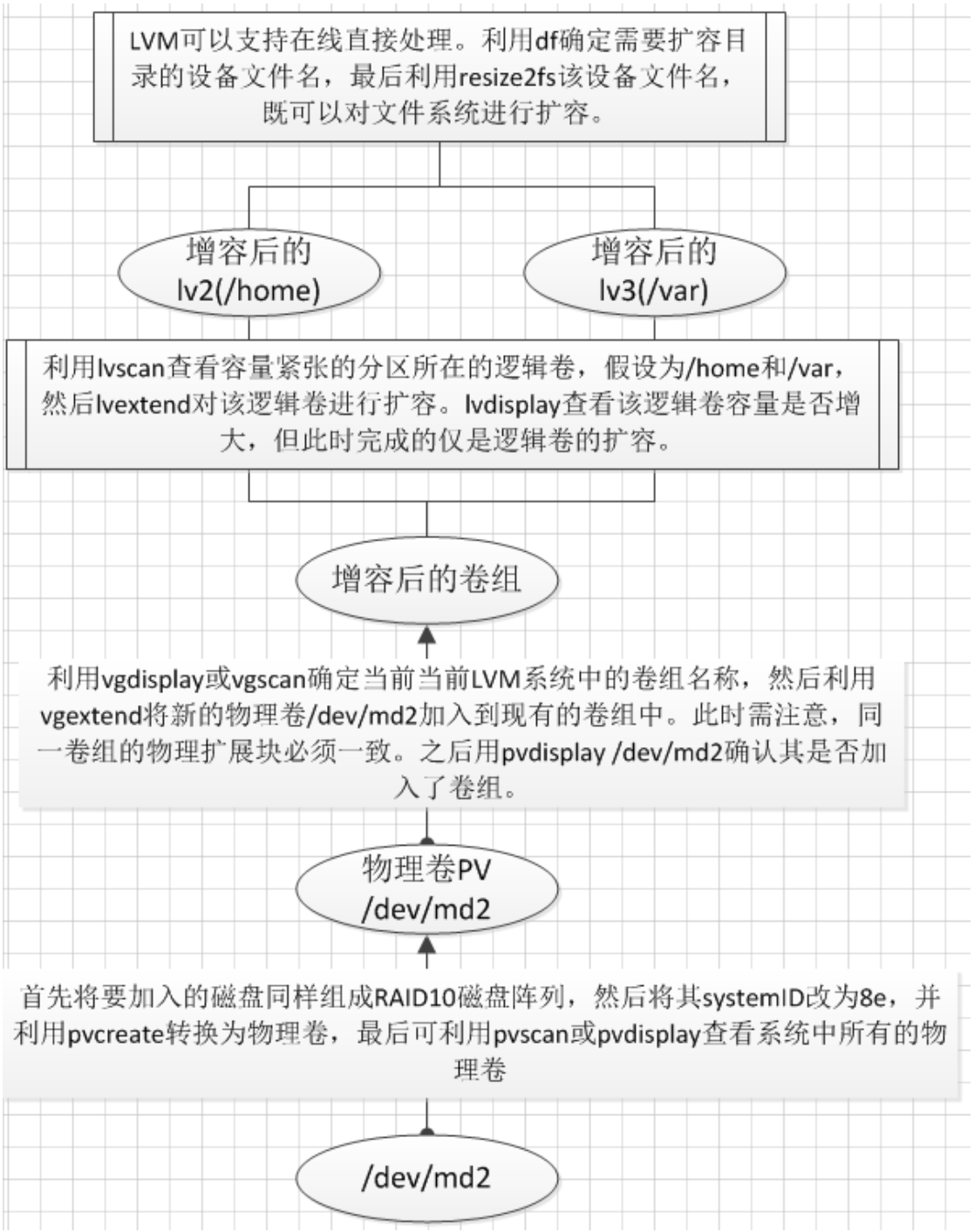


图 3-13 利用 LVM 实现扩容

Figure3-13 The capacity dilatation by deploying LVM

但正如刚才所提到 LVM 最主要的功能是实现弹性调整文件系统的容量，而不是磁盘的读取性能和备份，所以我们将 RAID 和 LVM 结合，这样一来就可以利用 RAID 来弥补 LVM 的磁盘读取性能和安全性的缺陷，因为 LVM 是建立在 RAID 之上的，所以如果某个磁盘出现故障，是 RAID 需要处理的问题，完全与

LVM 无关。与此同时，利用 LVM 来弥补 RAID 阵列建立完成后除非删除重建，否则无法扩充所带来的困扰，这无疑是实际生产环境中最佳的数据安全和性能实施方案。

3.2 测试用例的设计与构建

在 CPU 刚刚对虚拟化支持的时代，由于操作系统和 VMM 的紧密配合，准虚拟化实现了针对全虚拟化软件方案的性能优势，但是随着 CPU 对虚拟化支持的不断完善，全虚拟化的性能早已反超了准虚拟化技术，其优势在 X86_64 的操作系统上表现的更加明显。由于硬件的支持，捕获异常、翻译和模拟就不需要虚拟机的来实现了。再加上，无需改变客户系统实例的内核这一优势，使得基于硬件的全虚拟化已经成为虚拟化的核心技术。所以本论文统一选取全虚拟化技术，来构建测试环境。

通过第二章对虚拟化类型的详细阐述，得知从虚拟机的基本架构上来区分，虚拟机一般分为两种。

（1）第一种以 Xen 为代表，由于操作系统是安装在 Xen 建立的虚拟机中，而虚拟机监控器是专门为虚拟机研发的微内核，并运行在最底层来负责完成资源管理和调度，以及系统的创建和初始化工作等，所以当硬件服务器开机后，虚拟机监控器是被最先加载完成的。此类虚拟机大多不会精确的模拟物理设备的实际硬件特性，而是存在一个特权虚拟机，由它来管理操作系统环境，并提供给虚拟实例来进行常规操作和使用。

（2）第二种以 KVM 为代表，不同于第一种虚拟机监控程序，此类型虚拟机首先启动的是宿主机操作系统，而虚拟机监控器是作为特殊的应用程序来启动的。由于虚拟机监控器无需去管理硬件资源及调度，使得可以充分利用现有的操作系统。但是由于依赖于操作系统实现管理和调度，所以无法通过修改操作系统来达到优化虚拟机的目的。

3.2.1 虚拟化中网络模式

网络是现代计算机中不可或缺的一部分，KVM 和 Xen 都对虚拟机提供了丰富的网络支持。主要包括以下 3 种不同模式的网络。

（1）桥接模式。桥接模式可以让虚拟实例和宿主机使用同一真实的网络硬件来接入网络，虚拟实例拥有独立的 IP 地址，其所处的网络环境与宿主机并无任何差异，，虚拟实例能够访问外部网络，外部网络同时能够直接访问虚拟实例。尽管宿主机只拥有一个网络硬件，但是利用桥接模式，同样能够允许众多虚拟实

例与宿主机使用同一网络硬件，且桥接模式配置方法比较简单，在虚拟化环境中被普通的采用。

（2）NAT 模式。NAT 即 Network Address Translation 网络地址转换，属于广域网接入技术的一种，其将内网地址转化为外网的合法 IP 地址，被广泛的应用于各种类型的 Internet 接入方式和各种类型的网络中。处于 NAT 环境中的内部主机是通过 NAT 去访问外部网络的，所以其内部 IP 对外是不可见的，即隐藏了 NAT 内部网络拓扑结构和 IP 信息，避免了内部主机受到外部网络的攻击。这一点同时也变成了 NAT 的缺点，在某些特点的网络环境中，以 Web 或数据库服务器等需要接受来自外部网络的主动连接的主机，由于 NAT 隐藏了内部主机细节以提高安全性，从而造成这些需要被外部网络访问的主机无法被访问。不过可以通过在外网 IP 的主机上使用 iptables 等工具实现端口映射（Port Address Translation，PAT），从而让外网对这个 IP 端口的访问被从新关联到 NAT 内网的对应端口来解决此问题。虽然将 PAT 与 NAT 结合可以达到既节省 IP 资源，又能保护内网安全，但是相应的网络复杂度就会增加，并且网络性能较桥接模式差。

（3）主机模式。主机模式是将虚拟实例所处的虚拟环境与宿主机所在的真实环境隔离开。在这种与外部网络完全隔离的环境中，所有的虚拟实例之间能够自由通信，但是虚拟实例仅能访问宿主机，无法连接外部网络，同时外部网络也无法连接虚拟实例。其优点在于具有极高的安全性，因为外部无法访问，但同时这种隔离机制也成了其最大缺点，无法提供任何的服务。

本文中，在部署 KVM 和 Xen 虚拟化环境时，统一采用桥接模式来实现虚拟化网络环境，且虚拟实例使用的是与宿主机同一网段的固定 IP 地址。此外，由于虚拟实例需要访问外部网络，而访问外部网络基本都是通过域名来访问，所以需要为虚拟机配置 DNS 服务器。在本文中使用的是 Google 提供的免费的 DNS 服务器：8.8.8.8 和 8.8.4.4。

3.2.2 KVM 的架构和功能

KVM 的全称是 Kernel Virtual Machine。Qumranet 公司在开发 KVM 之初，为了简化开发过程并未从底层创建一个虚拟机管理器，而是基于 Linux 的内核，通过加载新的 Module 从而使 Linux 内核本身变成一个虚拟机管理器[8]。2006 年 10 月，虽然当时的 KVM 的性能和成熟度都略逊于 Xen 的项目，但由于内核社区对 Xen 取代内核由自身管理系统资源的架构的不满和抵制，且又急于将虚拟化的支持包含在内，所以在 KVM 问世的短时间之内就被内核社区接纳，使得 KVM 模块的源代码成为了内核源代码的一部分。

KVM 作为一个可被加载的内核 Module，使得其他能够支持广泛的各类操作系统。在 KVM 的架构中，虚拟机由常规的 Linux 调度程序进行调度，所以可以利用内核几乎所有的功能。KVM 充分利用了 Linux 内核的既有实现，尽可能的重用代码，所以 KVM 是通过内核模块的形式直接体现出来的基于宿主操作系统的虚拟机，且其虚拟机监控程序十分简洁和容易实现。

在 KVM 的架构中，有两个组成部分，分别是 KVM 模块和 QEMU。KVM 模块被加载到内核之后，通过使用一个新的字符设备驱动来模拟输入输出设备给客户系统实例使用，该驱动由 QEMU 通过系统管理程序的文件系统上的/dev/kvm 入口点以设置地址空间来完成。

（1）KVM 模块，是 KVM 虚拟化技术最重要的组件。首要功能在于负责开启并初始化系统硬件，从而支持虚拟机的正常工作，之后将客户系统实例运行在虚拟机模式下，并在一定程度上帮助客户系统实例的运作。在处理器虚拟化时，KVM 模块首先会去检测当前系统的 CPU，如果具备硬件虚拟化的功能，则尝试开启 CPU 控制寄存器中的虚拟化模式开关，通过使用 vmx on 命令将宿主操作系统运行在虚拟化的根模式，最后会在系统管理程序的文件系统上创建特殊的设备文件/dev/kvm，并等待来自用户空间的命令[9]。

KVM 模块与用户空间的通信接口其实是对特殊设备文件的 IOCTL 调用，以获得设备信息并发送设备控制参数等。其中针对虚拟处理器的调用就是执行虚拟处理器（VCPU），当 KVM 创建多个可供运行的 VCPU 时，会针对这些 VCPU 生成相应的文件句柄，并针对这些文件句柄来执行有针对性的 IOCTL 调用，使用户空间准备好的虚拟机在 KVM 模块的支持下，被放置在虚拟化模式中的非根模式，开始执行二进制指令[10]。在非根模式下，所有敏感的指令都会被 CPU 截获，CPU 在保存这些敏感指令之后自动切换回根模式，然后直接交由 KVM 模块去处理，从而可以对 VCPU 进行相应地管理。

此外，在提供可供运行的 VCPU 后，KVM 模块还需要实现内存的虚拟化。处理中的内存管理单元是通过页表的形式将程序运行的虚拟地址转换成为物理地址[11]，但在虚拟化环境中，因为 KVM 模块建立了虚拟实例程序的虚拟地址和虚拟实例物理地址与实际内存物理地址的映射关系，使得的内存管理单元的页表在一次查询时必须完成两次地址转换，即将虚拟实例程序的虚拟地址转换成为虚拟实例物理地址，在将虚拟实例物理地址转换成真实的物理地址，虽然 KVM 模块使用的影子页表技术解决了这一问题，但影子页表实现复杂，而且开销比较大。为了解决性能开销问题，通过引入 Inter 的 EPT 技术[12]，实现二维分页机制，即通过引入第二级页表来描述虚拟实例虚拟地址和真实物理地址的转换，从而硬件

就可以自动进行两级转换生成正确的内存访问地址，使得性能开销大大降低。

（2）QEMU。KVM 并不会去完成任何的仿真，代替的是通过 QEMU 来完成。其实 QEMU 本身就是一个开源虚拟化软件，是纯软件的实现方法，所以性能比较差。但是其优点在于通过 QEMU 本身编译运行的平台上就可以实现虚拟机的功能，且 QEMU 的代码中有成熟的整套的虚拟机实现[13]。同样基于代码重用的原则，KVM 在 QEMU 的基础上进行了修改，直接使用 QEMU 来模拟网卡，磁盘等虚拟设备。当虚拟机有 I/O 行为时，KVM 模块将虚拟机的 I/O 行为请求告知给 QEMU，之后 QEMU 会负责模拟这些设备并解析相应的请求。在 KVM 模块通过和 QEMU 协同处理下，QEMU 使用了 KVM 模块的虚拟化功能，而虚拟机的配置和创建，以及虚拟机运行时依赖的虚拟设备、用户操作环境和交互都由 QEMU 来实现，从而极大的提到了虚拟机的性能。KVM 的架构如图 3-14 所示。

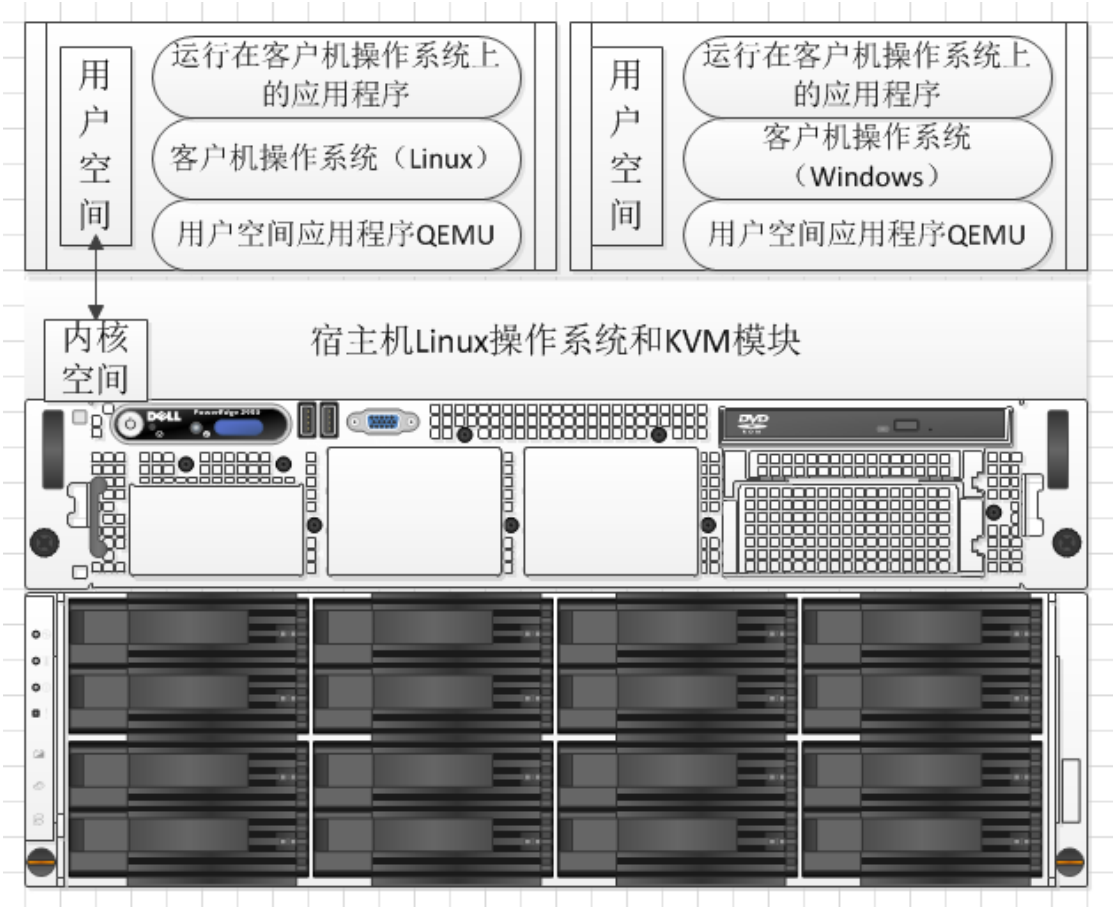


图 3-14 KVM 的架构

Figure3-14 Architecture of KVM

3.2.3 部署 KVM 虚拟化

KVM 以模块形式被集成在 Linux 内核的源代码中，所以只需要下载 2.6.20

以后的版本的Linux内核即可编译和使用KVM。但是在部署至实际生产环境中，最好是最新的稳定版本的内核，再进行编译和安装。具体配置和编译 KVM 的过程如图 3-15 和图 3-16 所示。

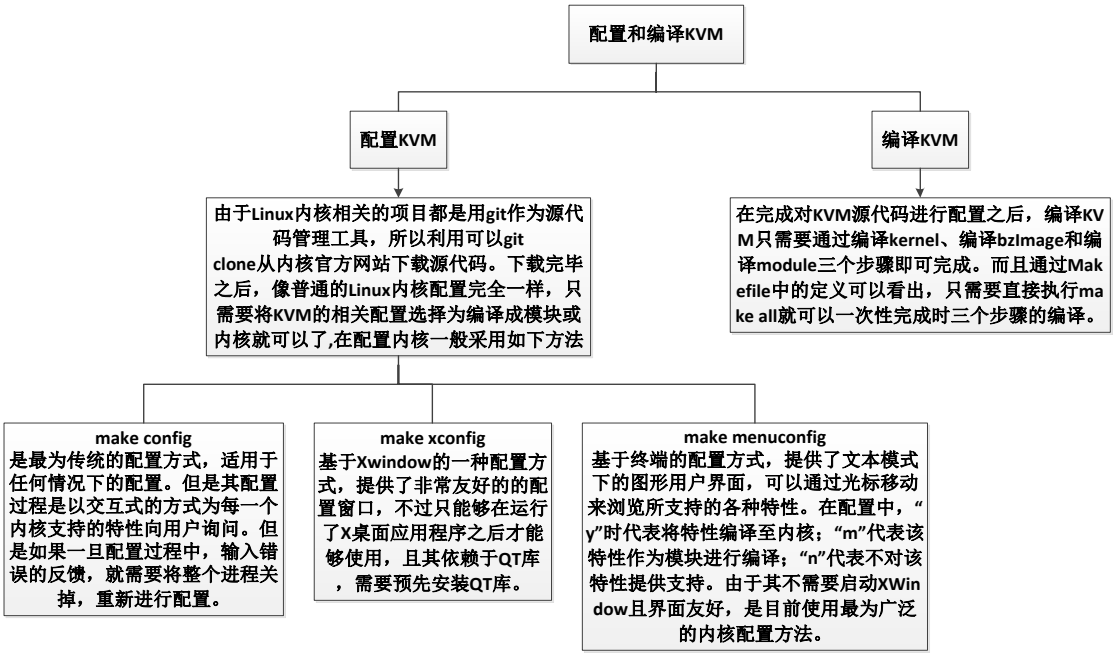


图 3-15 配置和编译 KVM

Figure3-15 Configuration and compiling of KVM

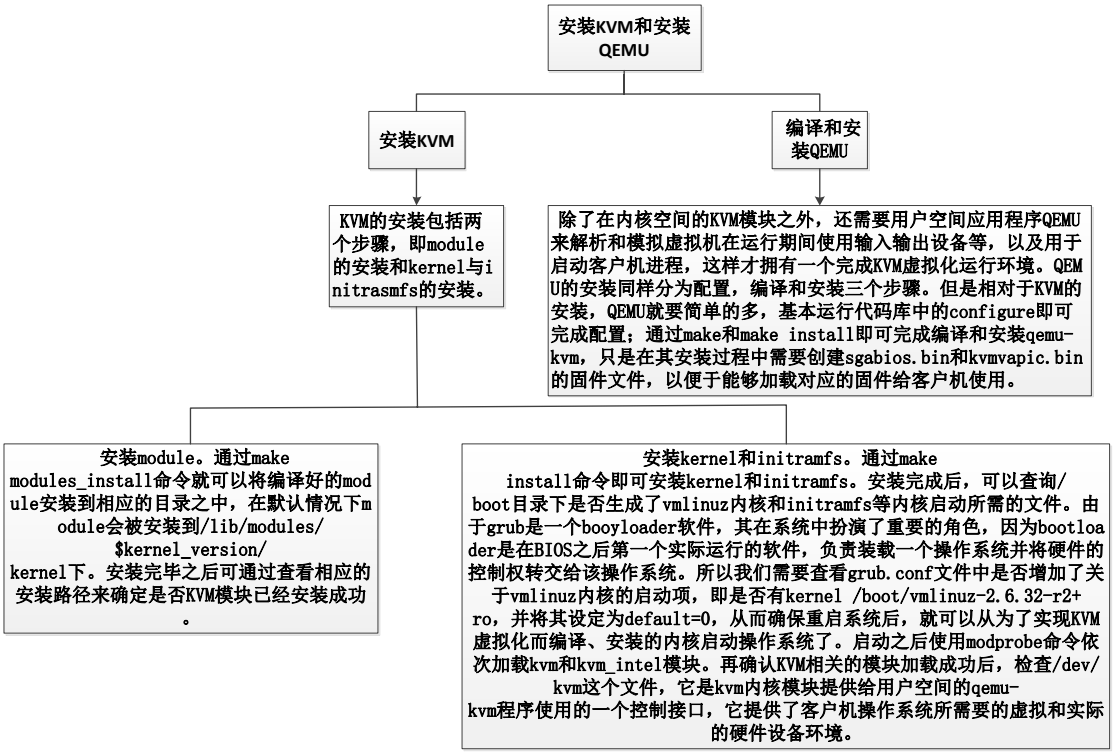


图 3-16 安装 KVM

Figure3-16 The deployment of KVM

在完成上述步骤之后，就能够开始利用磁盘镜像模式安装客户系统实例了。为了避免内存溢出以及性能测试的有效性，固化所有客户系统实例的内存为 4Gb，分配给虚拟实例 4 个 vCPU，每个虚拟实例拥有 100Gb 的存储空间。

3.2.4 Xen 的架构和功能

Xen 虚拟机最早是由 Ian Pratt 和 Keir Fraser 研发出来的，Xen 虚拟机负责完成硬件资源的分配和管理，以及各 domain 之间的隔离和保护。Xen 第一次公开亮相是在 2003 年 ACM 的操作系统原理会议收录的一篇论文上。商用 X86 机器会支持虚拟化的观点很快引起了学术界的广泛兴趣。随后，越来越多的组织开始对虚拟化感兴趣。2005 年，Xen2.0 发布，这个新版本允许用户自由配置防火墙规则，同时也支持 LVM 卷写时复制以及使用 loopback 文件存储虚拟实例的磁盘映像。2006 年，随着 Xen 已经在虚拟化市场占据了很大份额的同时，Xen3.0 问世，在此次更新版本中增加了一个用于 Intel VT 技术的硬件虚拟化技术的抽象层，这样除了传统的准虚拟化系统，Xen 还可以支持未经修改的虚拟实例。虽然 Xen 的发展速度相当快，但是 Xen 由于内核开发人员对 Xen 的架构和实现产生强烈的抵触，导致 Xen 的相关改动一直不能顺利的进入内核源代码[14]。

Xen 虚拟机监控器是 Xen 系统的核心组件，位于虚拟实例 DomainU 和底层硬件之间，负责完成硬件资源分配和管理，以及虚拟实例之间的隔离和保护。当多个客户系统实例运行于同一个物理平台上时，Xen 的虚拟机监控器负责为各个虚拟实例分配物理资源，并完成设备的复用。其可以在多个虚拟实例之间平均的分配资源，也可以采用更灵活的分配策略，每个虚拟实例可用的存储空间和处理器资源都会受到限制，而虚拟实例可访问的设备也仅限于虚拟机监控器提供的访问接口，甚至虚拟机监控器还可以给虚拟实例提供实际并不存在的物理设备的虚拟机访问接口。此外 Xen 的虚拟机监控器提供的是通用的块设备模型，而客户系统实例可见的只是通用的块设备访问接口，标准的访问接口的好处是简化了客户系统实例在不同物理主机之间进行迁移时，只需要为客户系统实例提供标准的虚拟设备驱动即可，从而大大的降低了移植操作系统的复杂性。

在 Xen 环境中，除了 VMM 之外，还存在一特权虚拟机，通常将其叫做 Domain0。在 Xen 虚拟机系统启动时，Domain0 是最早运行起来的，随后它在利用相关的创建与管理工具，完成对其他虚拟实例的配置以及初始化工作[15]。由于虚拟机监控器并没有精确的模拟物理设备的实际硬件特性，而是提供了统一的、理想化的设备访问模型，所以虚拟实例并不会了解硬件平台具体使用的物理设备类型，实际的设备驱动是由 Domain0 提供的，所以 Domain0 本身可以直接操作

物理硬件，并通过异步的共享内存机制与其他虚拟实例进行交互。

运行于 Domain0 之上的设备驱动程序称为设备后端，则在其他虚拟实例上运行了设备前端，设备后端不仅管理着实际的物理设备，还给设备前端提供了统一的访问接口，它接收来自不同的虚拟实例的设备访问请求，并将其转化成具体的设备访问操作，除此之外，设备后端还实现了设备复用，使得每个虚拟实例都认为自己独占的使用某个真实硬件设备[16]。通过这种对 VMM 和 Domain0 功能上的划分，可以更加有效对虚拟机进行管理和调度。Xen 的架构如图 3-17 所示。

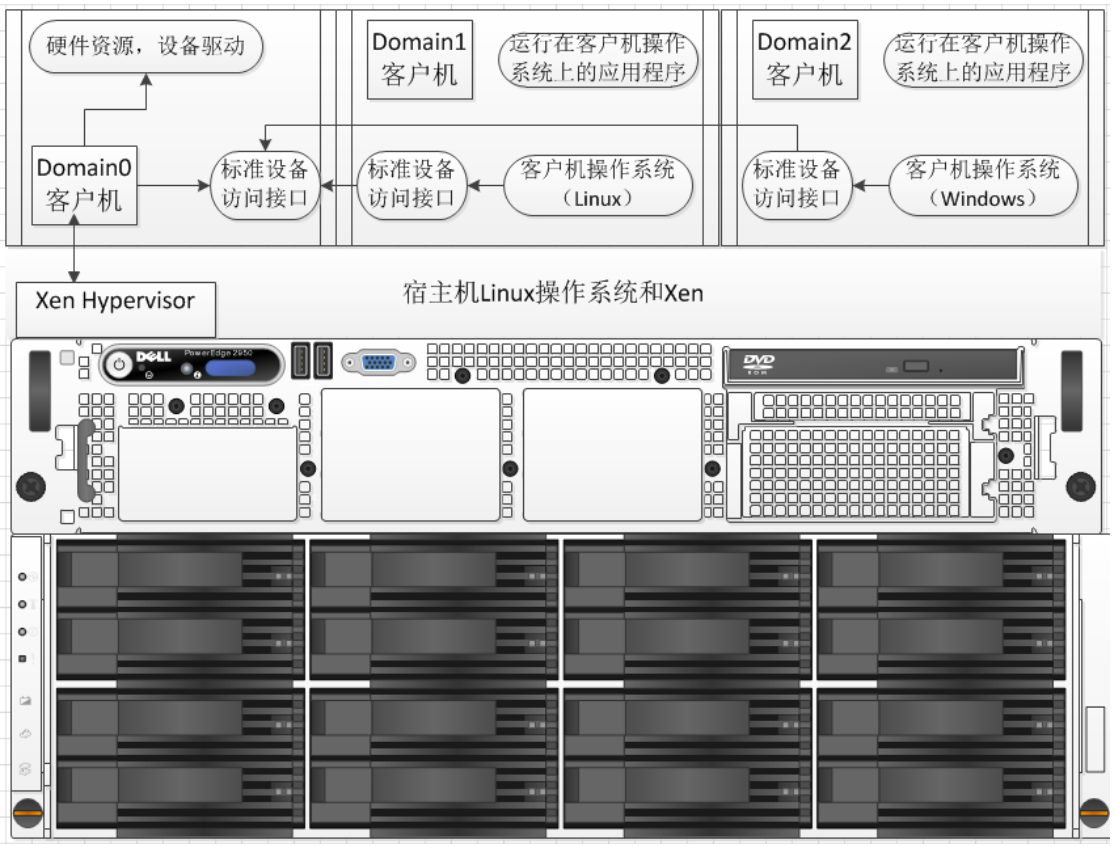


图 3-17 Xen 的架构

Figure3-14 Architecture of Xen

3.2.5 部署 Xen 虚拟化

Xen 的 VMM 并不能依靠其自身来提供一个完整地功能化的操作系统环境。它需要一个用户空间的配合来为构建和管理客户系统实例提供便利。用户空间环境对于 Xen 的 domain0 来说是非常重要的，因为它是所有虚拟实例依赖的基础。其中全球最大的开源技术厂家 Redhat 的 CentOS 操作系统可以说是最重要的扮演者。但由于在其 CentOS5.4Beta 版本开始，就用 KVM 虚拟化技术加入其内核，与 Xen 虚拟化技术并存，并在后续的 CentOS6.0 版本以后，在其内核中完全取消弃用了 Xen 虚拟化技术。所以需要对 Xen 进行全新安装和部署。具体部署步骤

如下：

（1）安装 xen、kernel-xen 和 centos-release-xen。通过 yum 的 install 功能可以一次性安装所有的 xen 组件。其中 xen 软件包提供了 VMM 和相应的依赖包，而 kernel-xen 软件包提供了 Domain0 和 DomainU 内核以及相应的依赖包。

（2）配置 grub 文件。由于目前 redhat 已经从 CentOS 的内核中完全弃用了 Xen，所以再安装了 Xen 的必要组件之后，我们需要修改现在有操作系统的引导文件/boot/grup/grup. conf，确保让其以 Xen 的内核启动，并加载相应的 VMM 软件。在重新启动系统之后，手动开启 xend 服务，通过 xm list 可以查看 domain0 是否启动成功。

（3）安装 libvirt 和 virt-manager，同样可通过 yum 的 install 功能一次性安装相应的软件包。virt-manager 提供了用于创建和管理虚拟实例的强大功能。成功安装两个软件包之后，手动开始 libvirtd 服务。

（4）安装 apache。因为在安装虚拟实例的时候，需要寻找安装源，这里我们使用 http 的路径方式来安装。在安装完 apache 之后，需要手动开启 httpd 服务。在 apache 根目录下创建 xen. tree 目录，然后将安装源拷贝至该目录下，并将安装 image 挂载至该目录，最后可以利用 http://本地的 IP 地址/xen.tree 来验证镜像文件是否挂载成功并加以使用。

（5）由于安装源的镜像文件是以 http 路径来指定的，所以在安装客户系统实例之前，需要关闭防火墙和 selinux 服务，否则在安装过程中会出现错误。

（6）安装客户系统实例。在完成上述步骤之后，就能够利用 virt-install 命令来部署虚拟实例了。其中为了后续的性能测试，每个客户系统实例的配置都应与以 KVM 部署的虚拟实例一样。指定每个虚拟实例的内存为 4GB，每个虚拟实例拥有 100Gb 的存储空间。

3. 2. 6 Postfix 邮件服务器

邮件系统是企业正常运行最基础的设施之一。虽然 exchange 邮件系统非常好用且性能卓越，但是其版权费和服务器也都相当昂贵。目前有很多的开源邮件系统，其功能也很丰富，如 sendmail、postfix 以及 iRedMail 等。尽管如此，如果没有非常清晰的弄明白邮件系统和 DNS 的关系，就贸然部署该系统是非常不明智的。

大多数情况下，收发邮件的时候都是使用帐号@主机名称的方法，很少使用 IP 去发送邮件，这就和平时上网都是输入网址而很少输入 IP 地址一样。但由于

最初对于 IP 地址的监管不力，以至于后来再以无法无法统一管理，所以导致目前不允许直接利用主机的 IP 来寄信。所以想要部署邮件系统，就一定要拥有一个对应的合法主机名称。此外，加之如今垃圾邮件和病毒邮件等占用了太多的带宽，导致企业要花费很多的成本来处理这些垃圾信息，为了杜绝或减少垃圾邮件的影响，目前的大型邮件主机商都会对不明来源的邮件加以限制，因此当架设的邮件服务器发送出一封邮件后，如果没有 DNS 的反向解析，基本都会被当成是垃圾邮件而无法正常收发。整个收发邮件的过程如图 3-18 所示。

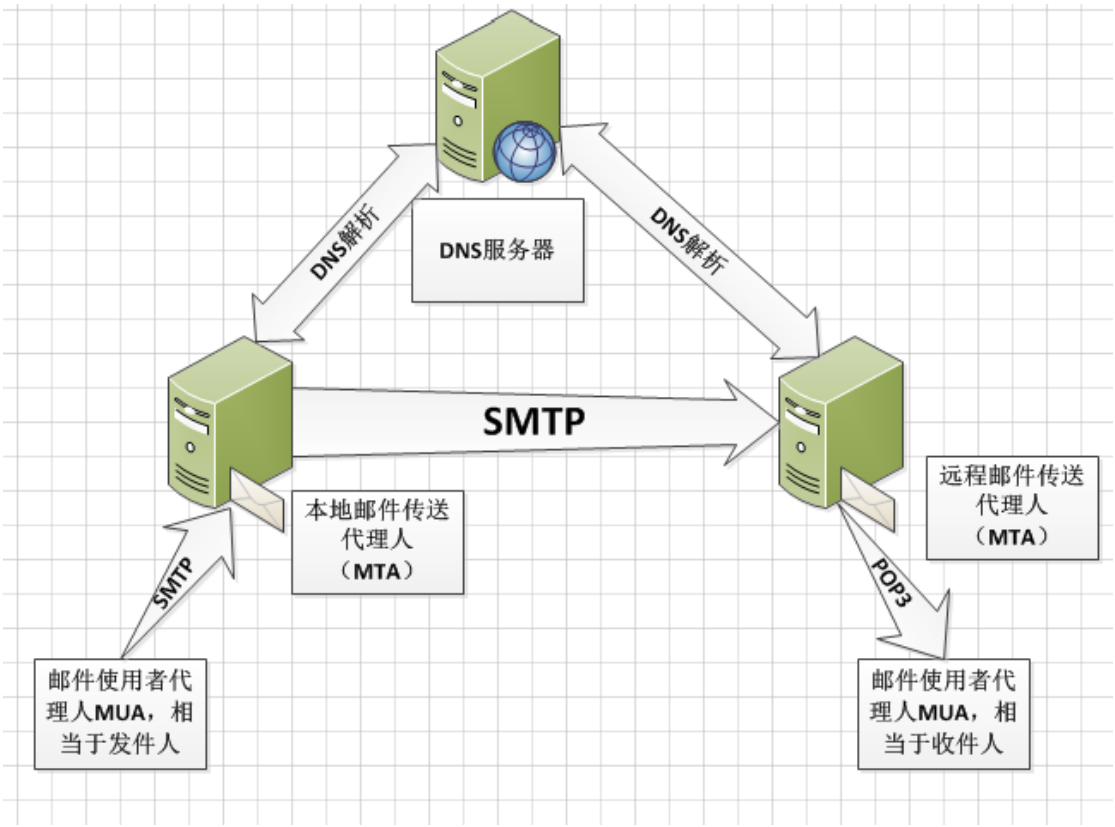


图 3-18 邮件传输过程

Figure3-18 The process of mail transmission

Postfix 与其它开源的邮件系统软件相比。具有无可比拟的优势。（1）部署方便，在 DNS 的 MX 和 SPF 生效的情况下，就可以在很短的时间内部署一个强大安全方便的邮件系统。（2）安全性高，Postfix 具有多层防御结构，可以有效的抵御恶意入侵者，且其提供安全针对 web 的 SSL/TLS 加密支持。（3）更好的兼容性和更高的灵活度。由于 Posfix 是由很多小程序组成的，每个程序完成特定的功能，可以通过配置文件来设置每个程序的运行参数。（4）更加稳定。Postfix 能在重负载的情况下仍然可以正常工作，当系统运行超出可用的内存或磁盘空间时，Postfix 会自动减少运行进程的数量，且当处理的邮件数量增加时，其本身运行的进程并不会跟着增加，降低主机的压力。安装和配置 DNS 与 Postfix 的过程如图

3-19 所示。

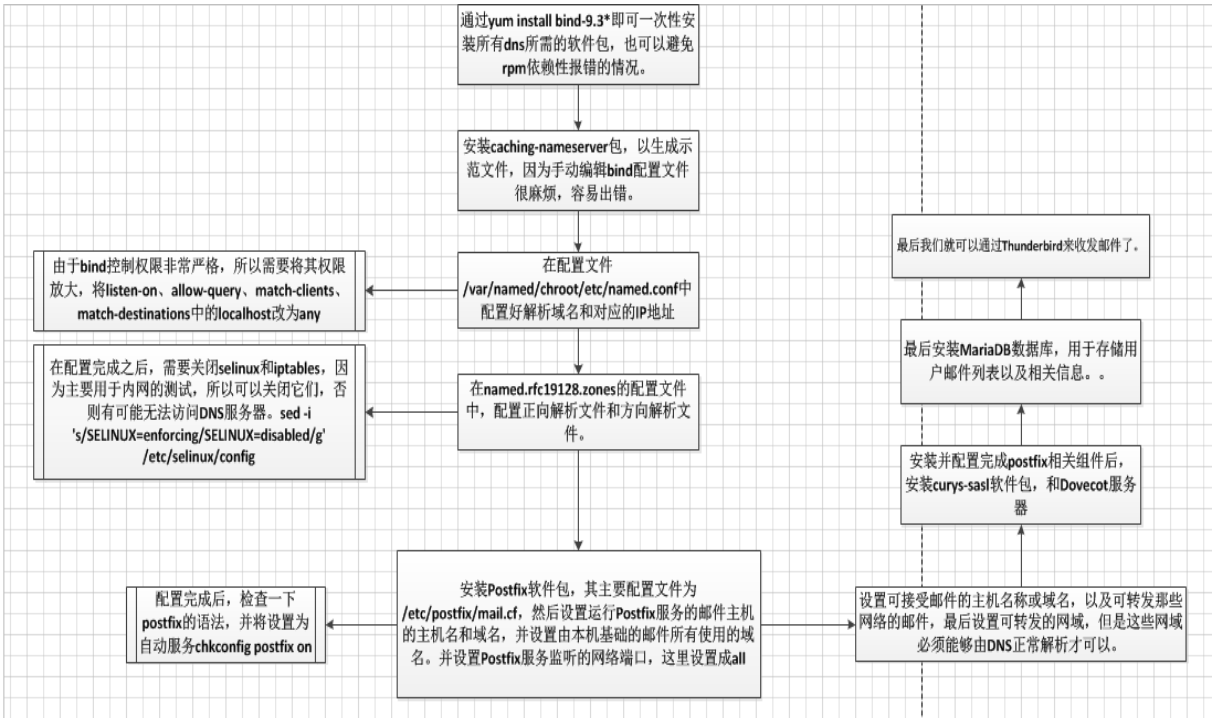


图 3-19 安装和配置 DNS 与 Postfix

Diagram3-19 The installation and configuration of DNS and Postfix

3.2.7 SAMBA 文件服务器

在 B/S 的结构下，文件服务器是承担对中央存储和数据文件管理的计算机，它可以作为其他计算机的远程磁盘驱动器来运行，允许处于同一网络内的其他计算机访问在服务器中的存储文件，用户不必使用外部存储设备来以物理的方式转移文件，可以直接在网络上共享信息。所以在商业领域的生产环境下，文件存储是运用最广的基础服务器之一[17]。

在早期的网络环境中，大部分是使用 FTP 作为文件服务器，以达到文件数据在不同主机或虚拟实例之间传输的目的。但是 FTP 服务器有个致命的缺点，就是无法直接修改服务器上的文件，即每次需要修改文件数据时，都需要先从服务器下载到本地才能进行修改，如果忘记将修改改过的文件上传回服务器，就会造成无法判断那个文件的数据是最新的，从而造成文件数据的混乱。

为了能够让客户端直接修改服务器端的文件，发展出了 NFS（Network File System）和 CIFS（Common Internet File System）。但 NFS 是用来让 Unix Like 操作系统之间进行共享文件资料，而 CIFS 是用来让 Windows 操作系统之间进行共享文件资料。但目前企业级的生产环境中，还没有完全使用 Linux 而不用 Windows。为了让 Unix Like 和 Windows 这两种不同的平台之间可以共享文件数据，在 1991

年由 Andrew Tridgell 对 SMB（Server Message Block）进行了反向开发，并在此基础上创建了 SAMBA[18]。通过 SAMBA 程序部署出的文件服务器就能够使 Unix 和 DOS 之间互相地访问彼此的数据了。

SAMBA 是通过 smbd 和 nmbd 两个服务守护进程实现了 Windows 的域控和工作组功能。其中守护进程 smbd 是最重要的，其主要功能是用来触发进程、身份识别、实现 SMB 协议的资源共享功能，管理 SAMBA 主机共享的目录，文件和打印机等。其监听 445 端口及 139TCP 端口。而 nmb 守护进程主要是承担管理工作组和 NetBIOS 的解析，它能够把某一操作系统共享的工作组名称与其 IP 地址进行对应，如果 nmb 守护进程没有启用，则在访问服务器时无法使用服务器的 NetBIOS Name 进行访问，只能使用服务器的 IP 地址。其监听 137 和 138UDP 端口。

在企业级生产环境中，大部分都采用域控连接模式，而非点对点的对等模式，因为这种模式不便于管理，而且安全性差。所以本文中部署的 SAMBA 文件服务器，采用的是域控连接模式，以实现不同的用户访问同一个共享目录具有不同的权限，便于管理和维护。这也是企业级生产环境中应用最广的方式，从而进一步保证了后续测试数据具有的说明意义，具体部署流程与实现功能如图 3-20 所示。

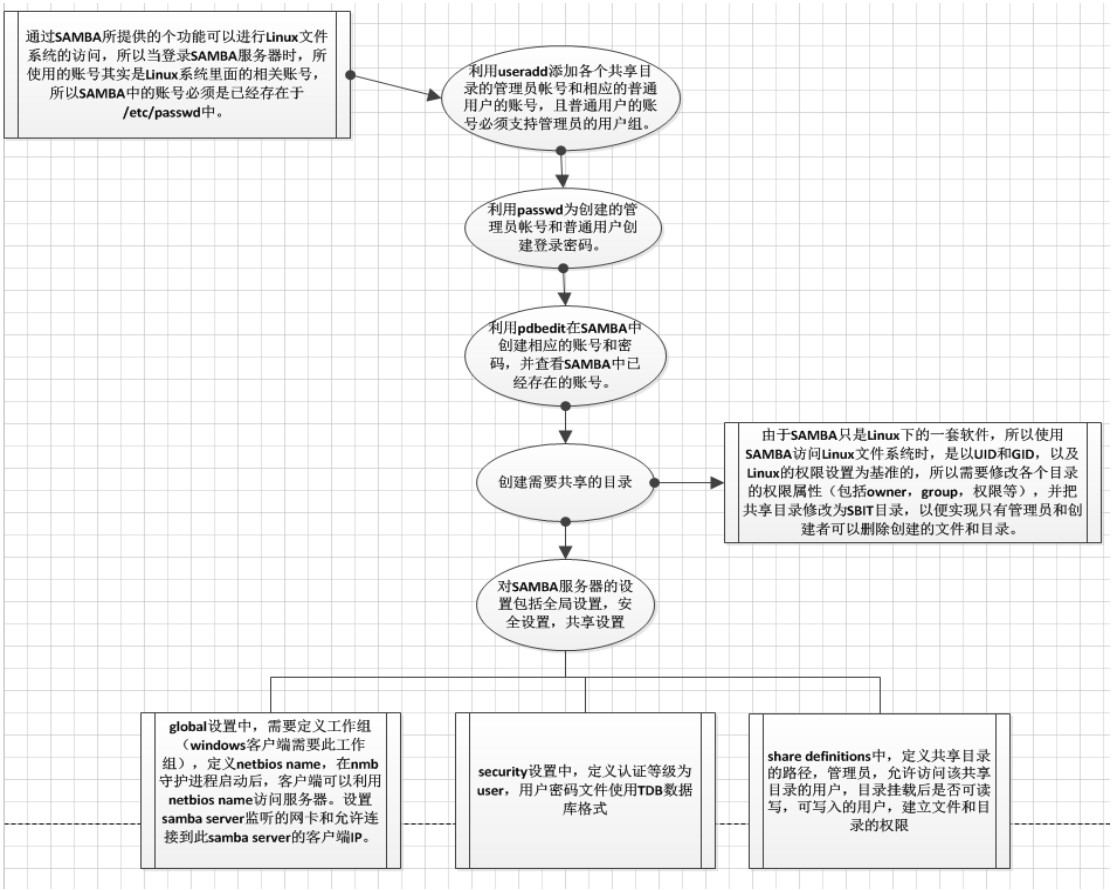


图 3-20 Samba 文件服务器部署流程

Diagram3-20 The deployment process of Samba

3.3 本章小结

本章主要阐述了 RAID 和 LVM 的原理，以及其在数据安全方面的应用。同时说明了的KVM和Xen虚拟化的结构和功能，以及如何部署相应的虚拟化环境。最后介绍了 Postfix 邮件服务器和 SAMBA 文件服务器，并说明了如何部署这两种工作负载。

第 4 章 虚拟化性能测试与结论

4.1 虚拟化性能测试方法与准则

纵然虚拟化有很多好处，如提高物理资源利用率，更方便监控和管理系统资源，节约硬件投入的成本等等。但是在实施生产环境的虚拟化，选择适合且稳定的虚拟化方案时，就需要重点考虑虚拟化方案的性能和功能。功能是实现虚拟化的基础，性能是衡量其效率的关键指标。即便是虚拟化技术具有非常强大的功能，但是其性能不好，那么将它应用到生产环境中就会带来意想不到的后果。此时一个具有实际参考意义的虚拟化性能测试就具有很大的意义。

虚拟化性能测试包含的领域非常宽，其中主要有最基本的子系统的性能，如：CPU、网络、磁盘和内存的性能；以及会对在同一台机器上部署不同的虚拟实例时的性能，因为同一台机器上运行不同操作系统是选择虚拟化的主要原因；同时物理机器硬件配置的差异也会导致性能的差异，所以当将虚拟实例部署在不一样的机器上时进行相应的性能测试是很有必要的；有时甚至还会对虚拟实例进行动态迁移，以测量在迁移过程中的性能。虚拟化性能测试似乎很复杂，难以操作，但实际上虚拟化性能并没有那么高深，因为对于终端用户最关心的是实际使用的应用软件和互联网站点的性能。因为应用程序和网络站点才是真正关系到用户体验。从这个层面来说，只要确保一般应用程序能够在虚拟化环境中表现的性能良好，自然就能带来良好的性能体验。

然而应用程序的数量和类型如此繁多，如文件存储系统、邮件系统服务、Web 服务、科学计算服务、数据库服务、缓存服务、办公软件等等，且它们对系统的运用特性都不尽相同。如：邮件系统服务属于网络 I/O 聚集型；文件存储服务属于磁盘 I/O 聚集型；而缓存和科学计算服务则分别属于 CPU 聚集型和内存聚集型。所以，衡量虚拟机的性能最有效且最具有说明意义的方式，就是将准备实施虚拟化的系统中运行的应用程序迁移到虚拟实例中试运行，如果性能良好且稳定，则说明该虚拟化方案适合此类应用程序，便可考虑真正实施该虚拟化方案。

虽然程序的数量和种类如此之多，且对系统的运用特性都不尽相同，但它们都会运用计算中的最基本的子系统，也就是说 CPU、内存、网络、磁盘等。

4.1.1 CPU 性能测试方法与准则

CPU 是机器系统中必不可少同时也是最核心的组件，其负责程序指令的运行，所以其性能直接决定了机器的计算能力，因为任何程序的运行都会消耗 CPU

的资源。CPU 的测试工具有很多，采用 SuperPI 和内核编译两种方法来进行测试。

（1）SuperPI，其是一个典型的 CPU 密集型基准测试工具，由日本数学家金田康正在 1995 年发明用于计算圆周率，他计算出了 π 小数点后的 2^{32} 个数据位[19]。SuperPI 基准测试程序的原理非常简单，根据用户的设置计算圆周率小数点后的 N 个位数，且支持单线程程序，可以执行多个实例从而实现多个计算程序同时执行。本文中选择执行计算圆周率小数点后 3355443，即 2^{35} 个数据位，之后统计计算所花费的时间，根据时间长度的比较就能衡量 CPU 计算能力的优劣。即花费的时间越少，说明性能越高。

（2）内核编译，其是以固定的配置文件对 Linux 内核代码进行编译，也是开发者社区中最常用的系统性能测试方法，可以算是一个极其典型的基准测试程序。内核编译时 CPU 聚集型程序。本文中通过对正式发布版本中的 2.6.32 内核进行编译，并用 time 命令对编译过程计时。time 命令会输出三种结果，分别是 Real、User 和 Sys。其中 real 是时钟时间减去程序从开始至结束的总时间，还包括程序运行期间其他进程所占用的时间片和进程被阻塞的时间，例如 IO 等待的时间。User 是被测试程序在用户模式下所花的 CPU 时间，是该程序进程执行的真正的 CPU 时间，其他进程调度的时间片以及诸如 IO 等阻塞的时间不包含在内。Sys 是测试程序进程在内核中所花费的 CPU 时间，表示该程序进程在内核调用中所花的 CPU 时间，而程序的库调用仍然运行在用户空间下。所以为了保证让 CPU 性能测试数据具有最大的实际意义，以在内核编译时利用 time 所计算出来的 User+Sys 时间为准，因为 User+Sys 表示程序所执行的 CPU 总时间，且不包括 IO 阻塞以及其他进程的 CPU 时间。

由于编译过程中极其的消耗系统资源，所以只要使用相同的内核代码，使用相同的内核配置（内核编译统一采用 make menuconfig 中的默认值），使用相同的命令进行编译，最后对比编译时间的长短即可直观的评价系统之间的性能差异。

4.1.2 磁盘 I/O 性能测试方法与准则

在计算机系统中，CPU 获取自身缓存数据的速度非常快，内部局域网的速度也比较快，到那时磁盘 I/O 的速度就相对比较慢，但是磁盘存储着系统内的数据、程序等内容，是系统不可或缺的一部分，尤其是以 TB 甚至 PB 为计量的海量数据时代的今天，对磁盘的存储容量和存取速率的要求越来越高。所以在部署虚拟化方案时，不得不对磁盘 I/O 的性能进行详细的测试，以满足对存储的容量和速度的巨大需求。采用 dd 和 IOzone 两种方法进行测试。

（1）dd 命令是 Linux 上一款自带的且非常流行的文件复制工具，在复制文

件的同时可以根据其具体选项进行转换和格式化等操作。通过 dd 复制相同数据量的文件，且无论读取还是写入，都绕过页面缓存，在写入时配置成每次写入都要同步到物理磁盘设备后才返回测试数据。此外，当文件的数量越多且单个文件的大小越小，则写入的速度越慢，所以通过跳过页面缓存以及实际写入并同步后再返回数值的方式，写入和读取 100000 个单个大小为 8KB 的数据，以及写入和读取 10000 个单个大小为 1MB 的数据，来测试磁盘的 I/O 性能。通过查看其所需时间长短，即可真实有效地来评估磁盘 I/O 的性能。时间越短，速度越快，则说明磁盘 I/O 的性能越好。

（2）IOzone 是一个非常强大的磁盘读写性能基准测试工具，通过对数据进行多种操作，如读写、重读、重写、随机读写等，来衡量一个磁盘的读写性能[20]。本文中通过规定待测数据的大小是 40960MB，一次对数据的读写操作单位为 16K，一次性运行写/重写、读/重读、随机读写三种测试方案，并且使用直接 I/O 方式读写而绕过页面缓存，以求得倒最真实的读取和写入磁盘的性能。通过对比测试报告中的分数，即可得到对 KVM 和 Xen 虚拟化环境中磁盘虚拟化的性能对比。其中 IOzone 的输出数据包括 Write，Re-Write，Read，Re-Read，Random Read 和 Random Write。各项类别说明如图 4-1 所示。

类别	意义
Wrtie	测试写入一个新文件时的磁盘性能。其中当写入一个新文件时，不仅仅需要存储文件中的数据，还包含元数据，即用于定位数据存储在存储介质的具体位置的额外信息，如inode，目录信息，所分配的空间和与该文件有关但又不是该文件所含的数据等。
Re-Write	测试向一个已存在的文件写入的性能。由于这个文件已经存在，所以更多的是利用缓存写入，所以速度较快。
Read	测试读取一个已存在的文件的性能。
Re-Read	测试读取一个最近读取过的文件的性能。同样由于最近读取过，所以更多的是从缓存读取，所以性能更佳。
Random Read	测试随机读取一个文件中的随机偏移量的性能
Random Write	测试随机写入一个文件中的随机偏移量的性能。在此两项测试中，测试结果受到众多因素的影响，包括缓存的大小，磁盘数量，寻址延迟等。此两项测试结果是最能说明磁盘性能的数据。

图 4-1 IOzone 的测试类别

Figure4-1 Measurement category of IOzone

4.1.3 网络性能测试方法与准则

网络功能是一个非常重要的功能，尤其是在互联网、云计算盛行的时代。一个计算机如果没有网络连接，它就与外界隔离起来，失去了一半的价值。在企业环境中，无论是员工的办公 PC 还是存储数据、处理工作流的服务器，都存在于计算机网络之中，网络瘫痪会给企业造成巨大的经济损失。如果 Google 和 Amazon 这些大型互联网公司拥有的成千上网的服务器，都是基于网络为客户提供服务的。因此在选择具体的虚拟化方案和部署前，网络性能也是需要着重考虑的环节。采用专门用于测试网络带宽的 Netperf 和 Linux 上著名的文件传输工具 SCP。

（1）Netperf，是由惠普公司开发的网络性能测试工具，是目前非常流行的网络性能测试工具[21]。Netperf 能够衡量多个方面的网络效能，主要包含 TCP 和 UDP 等协议的单向传输数据类型的性能以及请求-响应类型的数据传输性能。具体采用的方法是：Server 端通过 netserver，以监听来自 Client 端的请求。Client 端通过 netperf，以向 Server 端发起请求。首先在 Client 端和 Server 端之间建立一个控制连接，用于传递有关测试的配置信息和测试完成后的结果。在控制连接建立并传递测试配置信息以后，客户端与服务端之间会另外建立一个测试数据连接，用来传递指定测试模式的所有数据。当测试完成后数据连接就断开，控制连接会收集好客户端和服务端的测试结果。为了尽可能体现真实的网络传输场景，执行测试 TCP 和 UPD 这两个非常典型的数据封包协议，并要求客户端发送缓冲字节为 1500 字节，测试持续的时间长度为 60 秒。

（2）SCP。一般而言，本地网络和远程网络进行数据迁移时，常用三种方法：ftp，wget 和 rsync。但是在迁移的数据量比较大的时候，通常利用 ssh 的 scp 方法，因为 scp 无论是速度还是效果都比前三种方法表现的性能要好很多。scp 是 Secure Copy 的缩写，是 Linux 系统中最常用的远程文件复制程序，且 scp 的传输是加密的。它可以作为实际的应用来测试网络传输的效率，即用 SCP 远程传输一个大小同样均为 20G 的文件，根据网络传输的速度，就可以评估出网络性能的好坏。传递速率越快，则表明网络性能越好

4.1.4 内存性能测试方法与准则

内存是一个非常重要的组件，虽然内存的容量不及磁盘，但是内存是与 CPU 沟通的一个桥梁，其作用是暂时存放 CPU 中将要执行的指令和数据，所有的应用程序都必须触发被加载到内存中才能被 CPU 所读取使用。内存的大小及其访问速度也直接影响整个系统性能，所以在虚拟化方案中，对内存性能的考量也是

比较关键的。在 CPU 测试环节中，提到内核编译其实也是内存密集型，但是由于 CPU 性能测试已经涉及，所以仅采用 LMBench 工具进行内存性能测试。

（1）LMBench。LMBench 用于评价系统综合性能的可移植性良好的基准测试工具套件，主要关注两个方面：带宽和延迟。LMBench 中包含很多基准测试，囊括了内存操作，管道，系统调用，上下文切换，进程创建和销毁，文档读写和网络等多个方面的性能测试[22]。具体采用的方法是，创建 SCCS 目录和 a. ChangeSet 文件，否则在测试过程中会报错。由于测试过程中，会有以交互式的方式询问测试细节，虽然默认值设定的很好，但有些需要根据实际的测试环境进行相应的更改。主要有：测试内存的大小为 1024MB；根据 CPU 的相关信息，修改频率为 2500。以求最大限度的体现出在实际测试环境下的有效的测试结果。

在内存性能测试所得到的数据中，主要包括:本地通信带宽测试和内存操作延迟测试。在本地通信带宽测试中，所得的数值越高，则代表性能越好。而在内存操作延迟测试中，所得数值越低，则代表性能越好。具体内容如下表：

表 4-1 内存性能测试的各项数据说明

Table4-1 The data specification of memory performance test

测试分类	测试内容	测试项描述
本地通信带宽测试	Mmap(re-read)	代表将文件映射至内存中，然后再从内存中读取文件，并计入映射时间和读取时间，最终得出读取内存映射速度
	Bcopy（libc）	代表使用 libc 从指定的内存区域拷贝指定大小的字节内容至另一个指定的内存区域的速度
	Bcopy（hand）	代表把在磁盘上某一位置的指定大小的数据拷贝到某一内存区域，并计入所用时间，从而得到其拷贝速度
	Mem（read）	代表内存读取速度
	Mem（write）	代表内存写入速度
内存操作延迟测试	Main Mem	代表系统内存连续操作延迟
	Rand Mem	代表系统内存随机访问操作延迟

（2）Memtest86+。除了 LMBench，Memtest86+也是非常著名的内存检测工具，有 Chris Brady 编写，该软件的目标就是提供一个可靠的内存测试工具，从而进行内存故障检测，且其不依赖于操作系统[23]。Memtest86+之所以是一个非常可靠的内存检测工具，主要是因为其提供可启动文件镜像，所以可以直接刻录至 U 盘或光盘，然后不以磁盘的方式启动系统，此时除了 BIOS 占用小部分内存外，内存基本上是未使用状态。但正因为这个特点，所以不选用 Memtest86+，因为其主要目的是测试未使用状态下的内存，而在实际生产环境中内存不可能处于未使用状态，所以采用 Memtest86+z 在实际生产环境中的进行性能测试并不具

有实际意义。

4.2 虚拟化性能测试方案

根据上述测试方法，采用三种性能测试方案，在宏观的整体性能，性能隔离度和可扩充性方面做出充分的对比，以求最大限度的保证本文的实际研究意义，这也是本文的创新之处。

4.2.1 宏观性能测试

宏观性能测试。正如上面提到，虚拟化性能测试包括的范围比较广泛，且在虚拟客户系统实例中应用程序的数量和类型都非常多，虽然应用程序对系统的使用特点都不同，但它们几乎都会使用 CPU、内存、网络、磁盘等基本的子系统。所以对两个运行在自己单独的虚拟机和操作系统下的虚拟实例，工作负载分别为文件服务器和邮件服务器，进行 CPU、内存、网络、磁盘性能测试。除了虚拟实例之外，对工作负载与虚拟实例一致的非虚拟化环境中的物理主机上，进行与虚拟实例同样的性能测试，以此作为评测基准。

4.2.2 性能隔离度测试

性能隔离度测试。利用虚拟化技术，可以在同一个机器上部署出多个虚拟实例，每个虚拟实例都能够拥有各自独立的完整的操作系统，但当多个虚拟实例被整合并运行在一个物理硬件之上，其中一台虚拟实例发生错误或使用物理资源较多时，是否会影响到整个主机的运行，以致于整个主机瘫痪。所以在生产环境部署虚拟化方案时，性能隔离度测试就占有非常高的比重，因为我们绝对不希望在使用一个应用程序时，这个应用程序占用了全部的 CPU 和内存资源，导致无法并发运行其他的程序。

为了能让性能隔离度测试具有实际的参考意义，在分别在 KVM 与 Xen 的基础下，同样部署文件服务器和邮件服务器并同时生成较大负载，对没有压力的第三台虚拟实例和宿主机以基准机器的身份分别进行 CPU、内存、网络、磁盘性能测试，并以此基准综合考量 Xen 与 KVM 的性能隔离度，这样可以清晰明了的看出，虚拟机对各个虚拟实例的隔离性能如何。之所以同时对宿主机也进行性能测试的原因是在虚拟实例占用物理资源（工作负载）较大的时候，是否会影响到其他的虚拟实例，甚至是整个系统（宿主机）。

4.2.3 可扩充性测试

可扩充性测试。通过可扩充性测试，得知在一台机器上，虚拟化方案 Xen 与 KVM 所能支持的最大工作负载数，从而得知以目前企业的工作负载，到底需要多高的硬件配置才能承载。以达到在部署 Xen 或 KVM 方案时，使用最少的硬件设备的同时，最大限度提升物理资源的利用率，节约成本。所以在此项测试中，通过在同样的硬件条件内，在 Xen 与 KVM 各自的环境中以递增的方式部署虚拟实例。每次部署完毕之后，就对基本子系统进行性能测试。当这些子系统的性能出现巨大的差别时，观察在 Xen 与 KVM 的虚拟化方案中各部署了多少个虚拟实例，即部署的时间越少，支持虚拟实例的数目越多，说明可扩充性越好。此外在部署工作负载时，在同时部署文件和邮件两个服务器之后，再根据性能测试方法和方案进行一次完整的测试并进行记录。之所以要同时部署两种服务器，是因为部署更多的同等服务器，可以保证冗余，即资源虚拟化，也就是通常所讲的虚拟化资源池。如此就不必要总担心硬件损坏，任一虚拟实例发生宕机的情况时，也不会影响至让整个服务的终止，同时也可以让服务的扩展或收缩变得更加的方便和平稳。

4.3 虚拟化性能测试结果及结论

4.3.1 宏观性能测试结果与结论

在宏观的整体性能测试中，尽管虚拟化性能测试包括的范围比较广泛，但 CPU、内存、网络、磁盘等都是基本的子系统。所以对这些子系统做一项整合性能指标评测，就可构成了一个整合测试堆栈单元。

（1）CPU 性能测试结果与结论。根据前两节说明的虚拟化性能测试方法和方案，在两种不同的工作负载的前提下，宏观性能测试方案中 CPU 的性能测试结果如图 4-2 和 4-3 所示。无论是利用 superpi 去运算圆周率 π 的小数点后 225 个数据位，还是利用 time 命令来计算内核编译所用的时间，所得的数据单位均为分钟，时间越短则说明 CPU 的性能越好。可以看出以 KVM 部署两种工作负载的 CPU 的性能分别是原生主机的 83%，82%，78%和 77%，而以 Xen 部署两种工作负载的 CPU 性能分别是原生主机的 60%，56%，64%和 59%。在同样的硬件条件下，即处理器均支持虚拟化技术，说明 KVM 的 CPU 性能要强于 Xen 的 CPU 性能。这主要是因为在 Xen 的虚拟化环境中，分配给虚拟实例的虚拟 CPU 并没有与任何特定的 CPU 绑定，虚拟 CPU 在哪个物理 CPU 上运行是自由的，所以在 Xen 中的 CPU 可以被多个虚拟实例共享，并且可以将上至 32 个虚拟 CPU

分配给任何单个虚拟实例，单个虚拟实例中允许拥有的虚拟机 CPU 数量不受物理 CPU 的数量限制，只是上限为 32 个。虽然不绑定 CPU 可以让虚拟 CPU 支持热交换，但是运行多个进程时，就需要在多个 CPU 之间交换运行，必然会造成性能损耗。而在 KVM 虚拟化环境中，宿主机运行在硬件之上，而 KVM 的内核部分是作为可动态加载内核模块运行在宿主机中的，其中包括实现虚拟化核心基础架构的 kvm 模块和 kvm_intel 模块。而 KVM 中的所有虚拟实例是作为多个用户空间进程运行的，它和其他普通的用户空间进程一样，都是由内核来调度使其运行在物理 CPU 上，只不过需要受到 KVM 模块的控制。而一个虚拟实例的多个虚拟机 CPU 就是一个用户空间进程中的多个线程，且在虚拟实例中，同样分别运行着虚拟实例的内核和虚拟实例的用户空间应用程序。此外，KVM 默认采用锁定处理器的设置，即锁定进程由某一个或某几个处理器来运行，而不允许该进程被协调至由其他处理器来运行。这样的设置能够降低进程在多个处理器之间频繁的调换执行，而所引起的缓存命中失效，虽然使得虚拟实例不在支持虚拟处理器的热交换，但是使性能能够得到提升，所以在性能上要比 Xen 高。

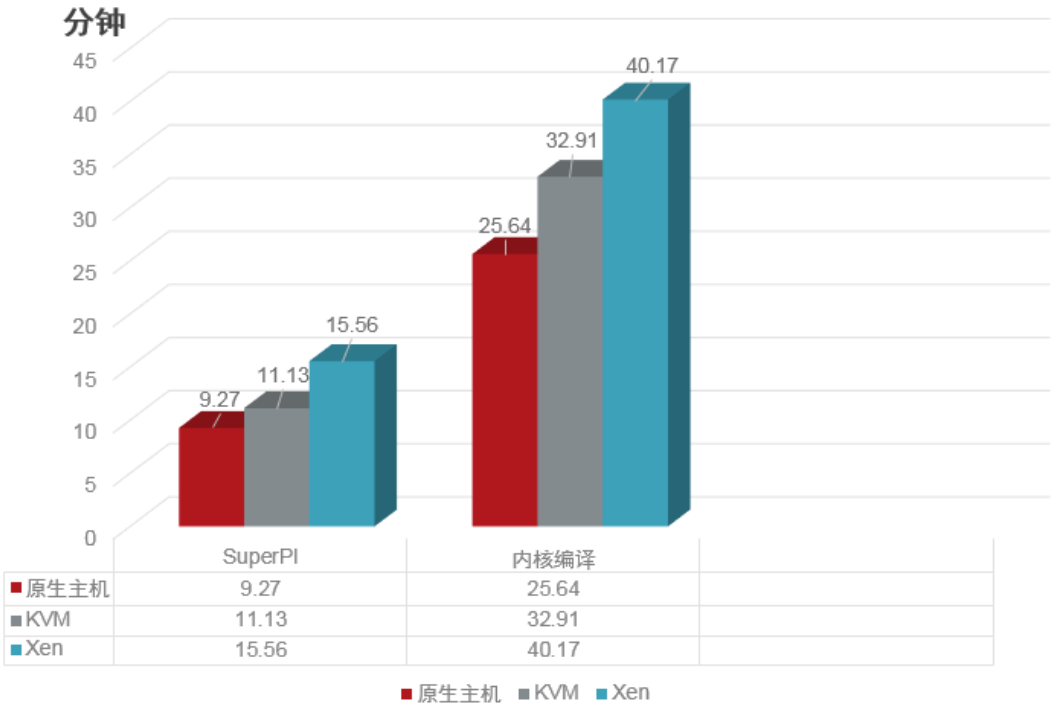


图 4-2 以文件服务器为工作负载，KVM、Xen 与原生主机的 CPU 测试数据

Figure4-2 The CPU test data of KVM, Xen and Host based on file server

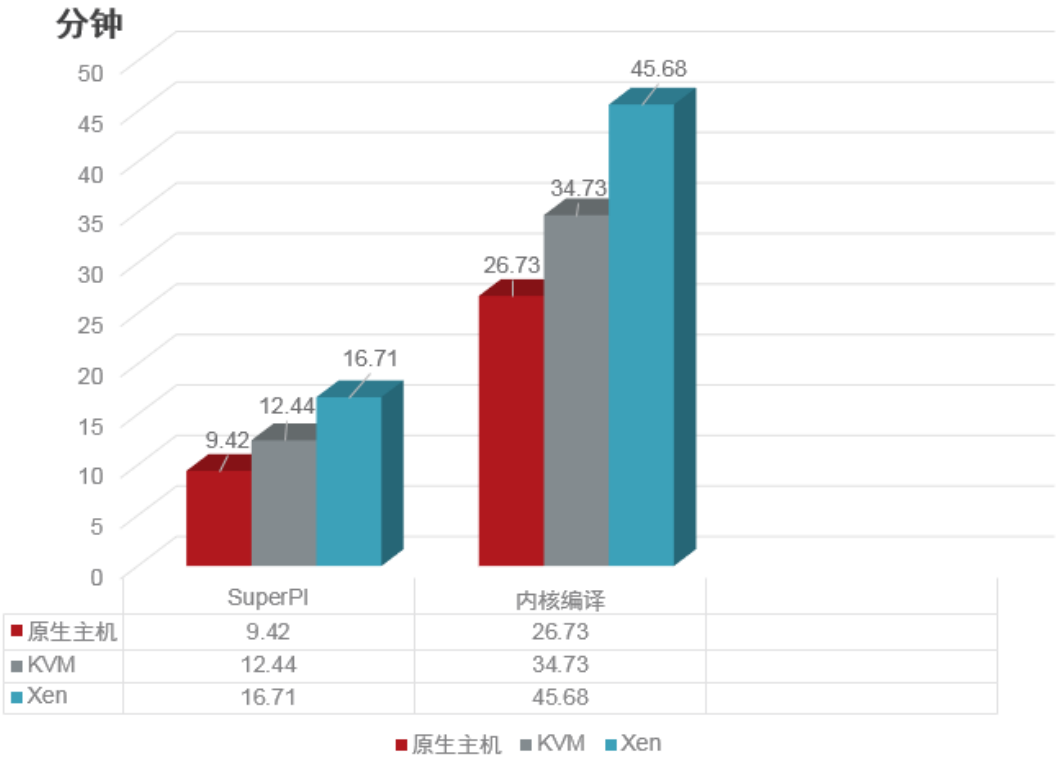
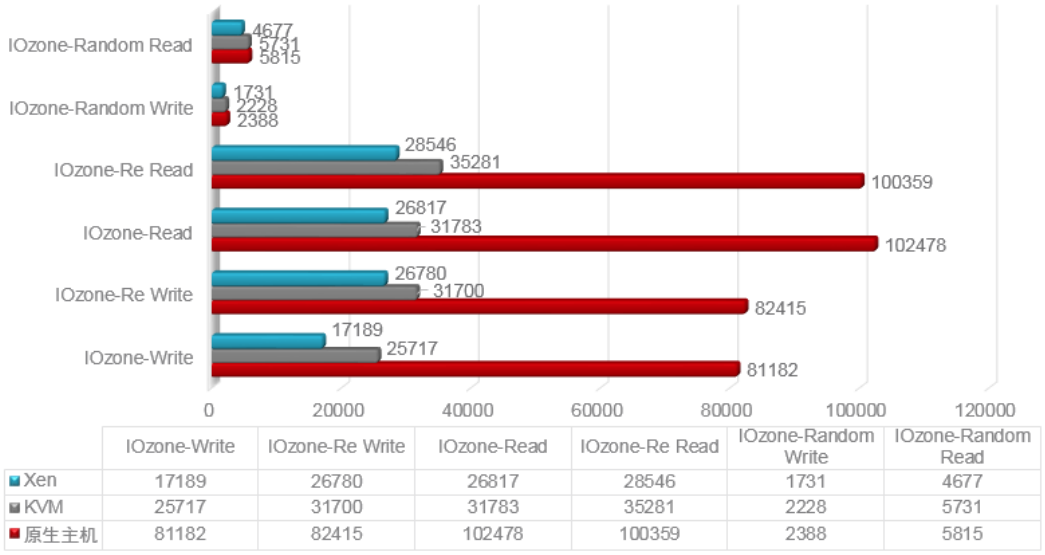


图 4-3 以邮件服务器为工作负载，KVM、Xen 与原生主机的 CPU 测试数据

Figure4-3 The CPU test data of KVM, Xen and Host based on mail server

（2）磁盘 I/O 性能测试结果与结论。在宏观性能测试方案中的磁盘 I/O 的性能测试中，通过对两种不同工作负载进行测试，测试结果如图 4-4 至 4-7 所示。由于文件系统的类型对磁盘 I/O 的性能是有影响的，因此无论是 KVM 虚拟实例还是 Xen 虚拟实例，采用的都是 ext4 文件系统的。但不一样的虚拟机采用的映像文件格式也具有差异，而虚拟机映像文件存储了虚拟机硬盘的所有数据，根据存储方式的不同，一般有两种类型：全映像型和稀疏型。全映像型存储了虚拟硬盘中的所有字节信息，稀疏型只存储对文件系统有意义的信息，只占用一定的磁盘空间。但在整个虚拟化环境中，当虚拟实例尝试对磁盘进行存取操作时，虚拟机监控器就会捕获其读写指令并调用宿主机内核的系统调用，然后根据虚拟机实现策略的不同进行必要的改变，最后重定向至对应的映像文件中的文件块。因此，为了能够准确的定位到虚拟实例的映像文件在磁盘上的扇区位置，其块地址须通过很多次的更替。在找到磁盘上正确扇区位置之后，就可以进行虚拟实例所请求的各项操作，最终完成虚拟实例的写入/读取请求。对于全映像型，通常只须一个线性转换即可完成，所以转换的过程就比较容易。但是对于稀疏型，通常须至少 2 次以上的地址转换，才可以实现虚拟块地址到物理块地址的更替过程。在 KVM 虚拟化环境中，是利用 qemu 用户空间程序来实现上述完成地址转换过程的。而在 Xen 虚拟化环境中，是由 Domain0 所触发的 TapDisk 进程，来承担其

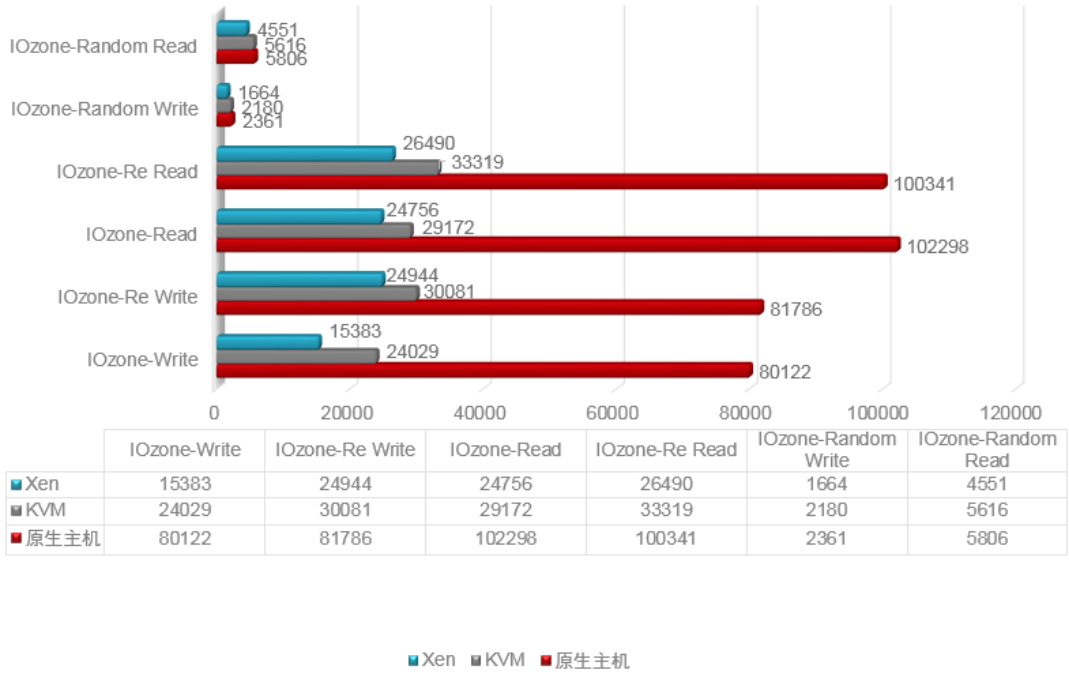
他 DomainU 的读写请求并完成地址转换过程的工作。其中 Xen 虚拟机默认采用的镜像文件格式是 qcow2，属于稀疏模式的文件格式，其在被创建之初并不占用很多的磁盘空间，而是随着实际写入数据才占用物理磁盘，比较灵活且节省磁盘空间，重要的是还支持快照功能，这一点使其更易于移植和备份，但是在写入数据时由于需要额外的在宿主机中分配空间，且地址转换过程长，所以其磁盘读写的效能较低。而 KVM 默认使用的映像文件格式是 raw，raw 是一种传统的映像文件格式，raw 文件不跟着映像的使用而增长，在创建之初即完全占用磁盘空间，而不是采用稀疏文件的方式来节省磁盘空间。尽管 raw 映像文件格式一开始就实际占用磁盘空间的方式没有节省磁盘的效果，但这种模式在存储新的文件时宿主机无须从现有的磁盘空间中分配且地址转换过程短，因此在写入/读取数据时，采用全映像型 raw 文件格式的 KVM 磁盘写入/读取的效能要比采用稀疏型 qcow2 文件格式的 Xen 磁盘写入/读取性能强很多。



	IOzone-Write	IOzone-Re Write	IOzone-Read	IOzone-Re Read	IOzone-Random Write	IOzone-Random Read
Xen	17189	26780	26817	28546	1731	4677
KVM	25717	31700	31783	35281	2228	5731
原生主机	81182	82415	102478	100359	2388	5815

图 4-4 以文件服务器为工作负载，KVM、Xen 与原生主机的磁盘 I/O 测试数据（IOzone）

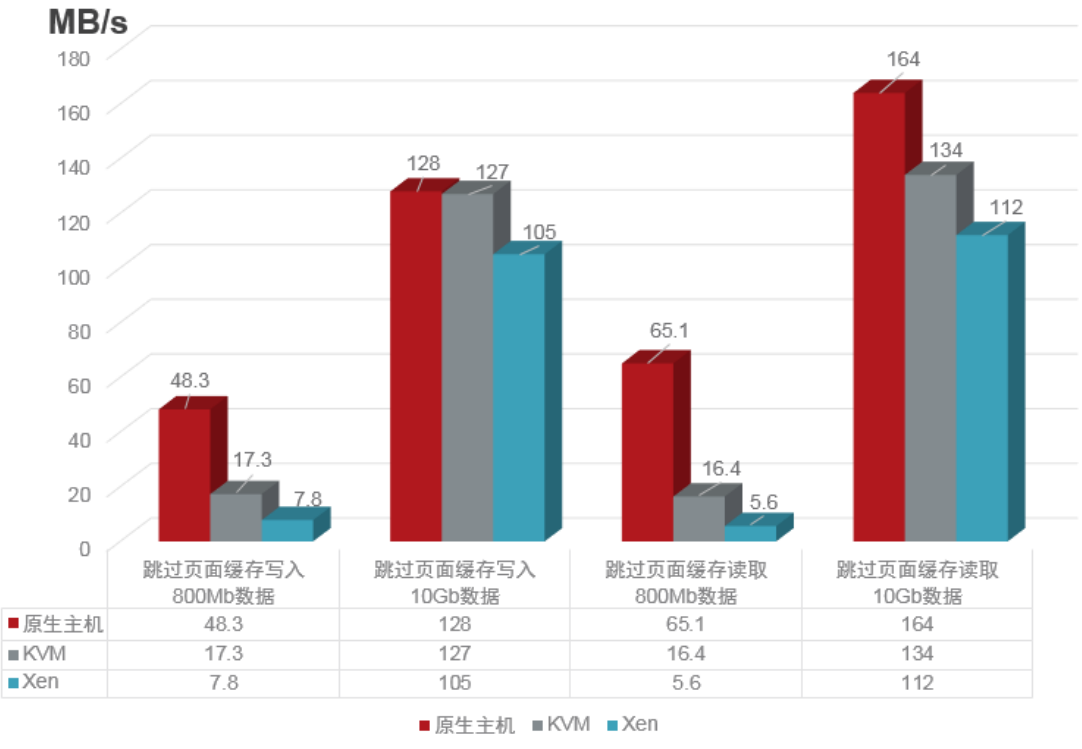
Figure4-4 The disk test data of KVM, Xen and Host based on file server (IOzone)



	IOzone-Write	IOzone-Re Write	IOzone-Read	IOzone-Re Read	IOzone-Random Write	IOzone-Random Read
Xen	15383	24944	24756	26490	1664	4551
KVM	24029	30081	29172	33319	2180	5616
原生主机	80122	81786	102298	100341	2361	5806

图 4-5 以邮件服务器为工作负载，KVM、Xen 与原生主机的磁盘 I/O 测试数据（IOzone）

Figure4-5 The disk test data of KVM, Xen and Host based on mail server (IOzone)



	跳过页面缓存写入 800Mb数据	跳过页面缓存写入 10Gb数据	跳过页面缓存读取 800Mb数据	跳过页面缓存读取 10Gb数据
原生主机	48.3	128	65.1	164
KVM	17.3	127	16.4	134
Xen	7.8	105	5.6	112

图 4-6 以文件服务器为工作负载，KVM、Xen 与原生主机的磁盘 I/O 测试数据（dd）

Figure4-6 The disk test data of KVM, Xen and Host based on file server (dd)

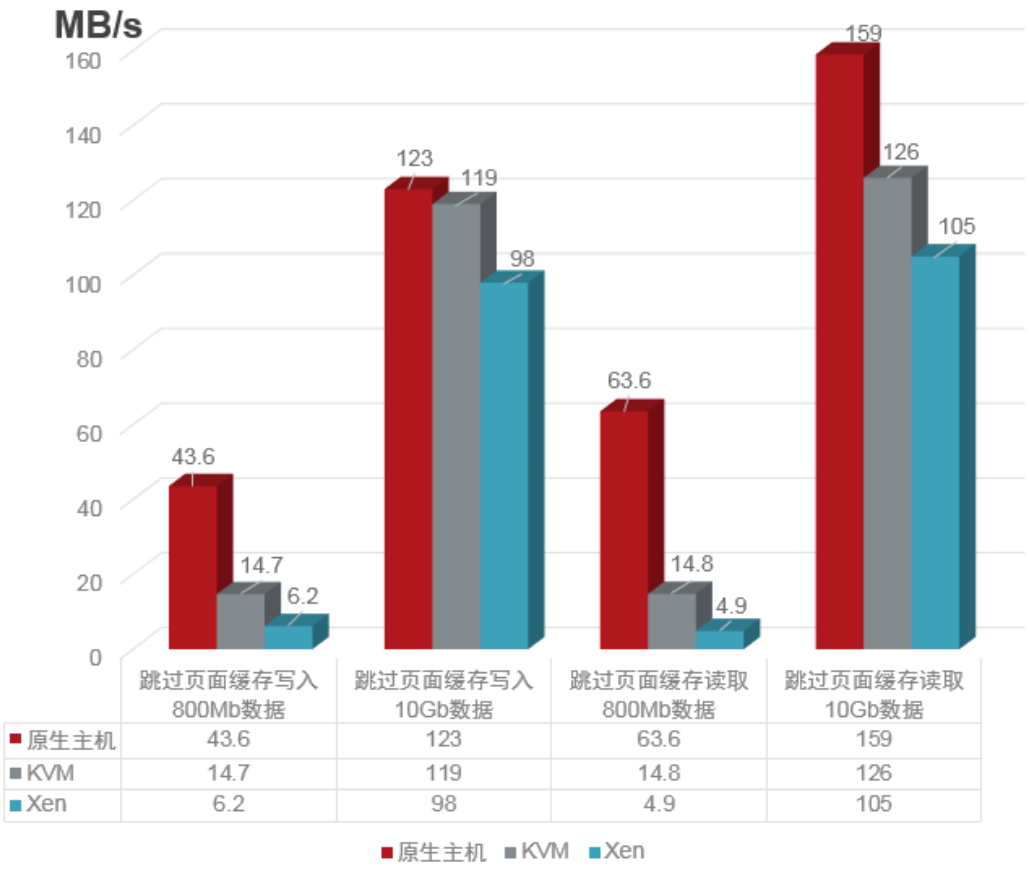


图 4-7 以邮件服务器为工作负载，KVM、Xen 与原生主机的磁盘 I/O 测试数据（dd）

Figure4-7 The disk test data of KVM, Xen and Host based on mail server (dd)

（3）网络性能测试结果与结论。在宏观性能测试方案中对网络虚拟化的性能测试中，通过对两种不同工作负载进行测试，测试结果如图 4-8 和 4-9 所示。采用 netperf 对 KVM 和 Xen 的网络虚拟化性能进行测量，通过比较所得数据。KVM 的网络虚拟化性能达到了原生主机的 96%和 97%，Xen 的网络虚拟化性能达到了 93%和 91%。造成这些差距的原因，还是在于 KVM 虚拟化采用 qemu 用户空间程序去虚拟网卡，与其模块虚拟 CPU 等功能进行有效的分离，使得性能要比 Xen 虚拟化中单一的采用 Domain0 去虚拟化所有硬件设备要更加高效。而使用 scp 的测试环节中，无论 KVM 还是 Xen 虚拟化，都比原生主机的网络性能差很多，这主要是因为 scp 复制受到本地磁盘 I/O 的限制。所以，在网络虚拟化性能测试中，虽然都有磁盘写入/读取性能的影响，但综合磁盘写入/读取性能测试的结论，Xen 在网络虚拟化性能方面还是差于 KVM。

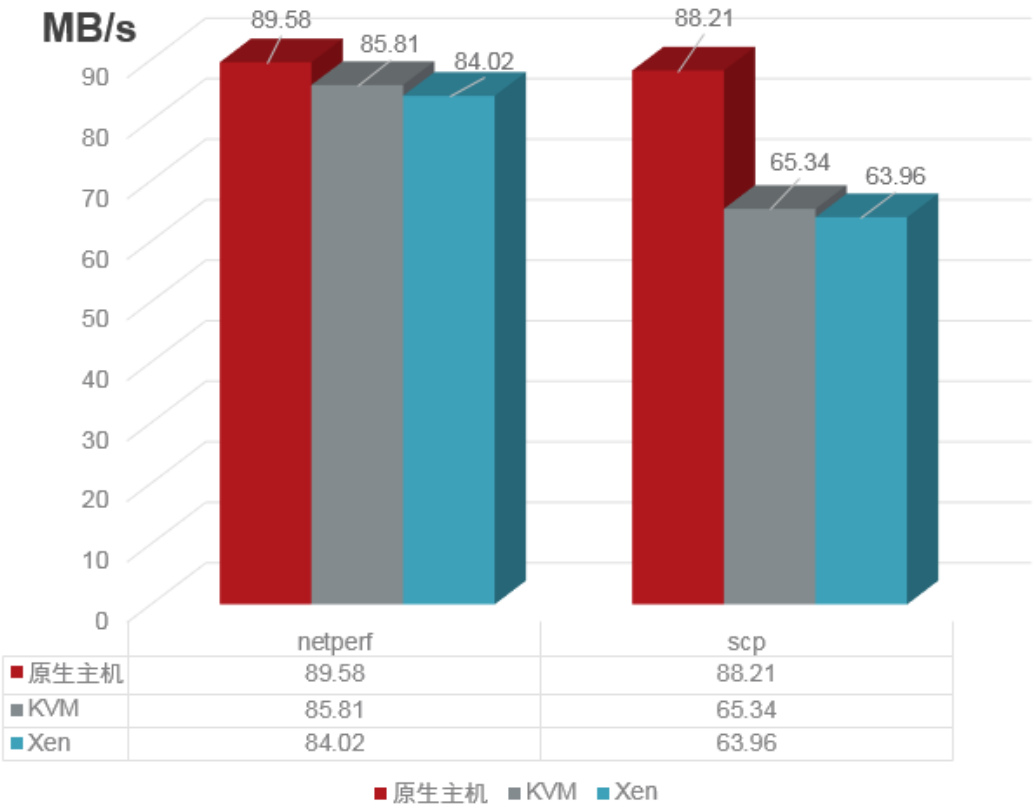


图 4-8 以文件服务器为工作负载，KVM、Xen 与原生主机的网络测试数据

Figure4-8 The network test data of KVM, Xen and Host based on file server

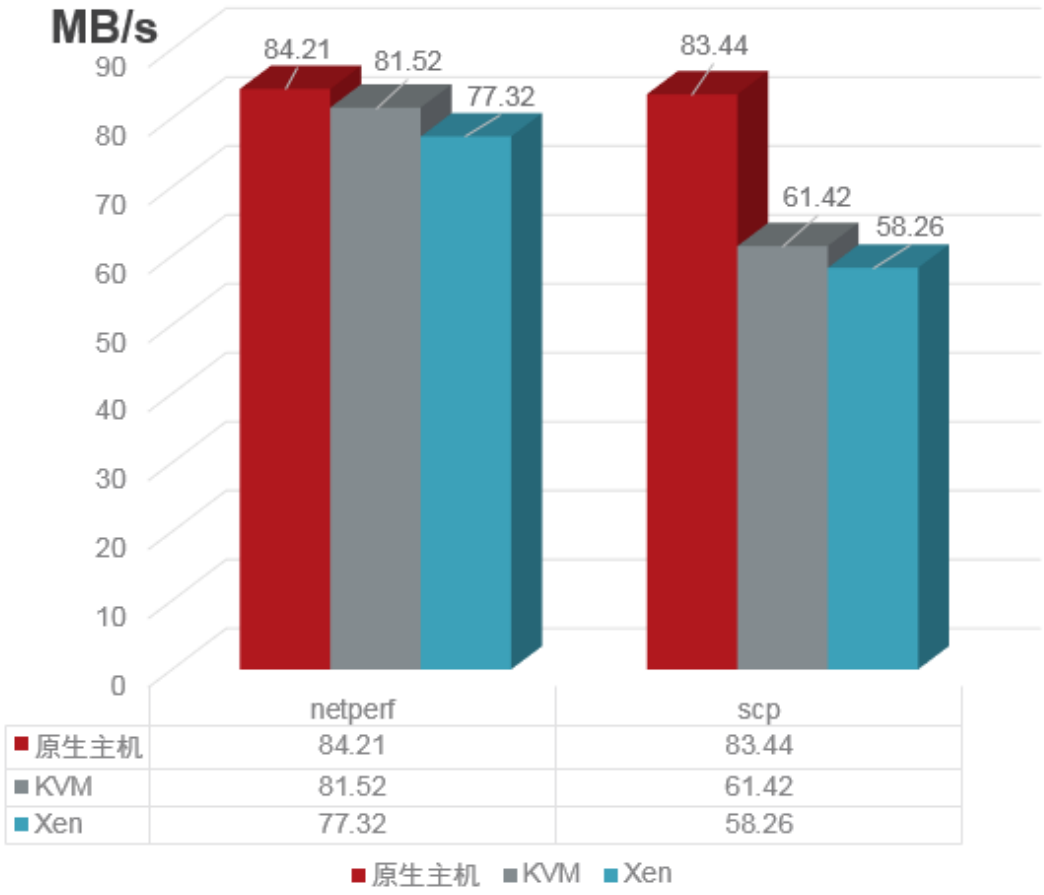


图 4-9 以邮件服务器为工作负载，KVM、Xen 与原生主机的网络测试数据

Figure4-9 The network test data of KVM, Xen and Host based on mail server

（4）内存性能测试结果与结论. 通过在两种不同工作负载的前提下，在宏观性能测试方案中内存虚拟化的性能测试结果如图 4-10 至 4-13 所示。和运行在真实硬件上的操作系统一样，在虚拟实例看来，其可用的内存空间也是一个连续的主内存区中的数据块。为了达到这一目的，Xen 和 KVM 采用的是不同的机制。其中 Xen 是在虚拟内存地址空间的开始部分为 VMM 保留了一部分内存，虚拟内存的剩余部分分配给了 Domain0 和 DomainU，这部分描述了可用于共享内存的一些限制和控制措施。而 KVM 是引入了一层与真实硬件上主内存区中的数据块存在着一层映射关系的虚拟实例内存区。但是在 Xen 和 KVM 的环境下，内存使用都需要两层的内存区更换，即由虚拟实例应用程序可见的虚拟实例虚拟内存区到虚拟实例的虚拟主内存区的更换，再从虚拟实例虚拟主内存区到宿主机主内存区的更换。

为了降低这种转换性能开销，Xen 采用影子页表技术，影子页表是一种可管理的内存。一个页是从诸如硬件设备等块设备处复制至主内存区的一个数据块，而页表记录了当前哪个页在主内存区中。虚拟实例维护自己的页表，而 VMM 把

这些虚拟实例的影子页表载入到物理上的内存管理单元中进行地址翻译，避免了两次地址转换，进而实现在普通的内存访问时从虚拟实例虚拟内存区直接到宿主机主内存区的转换。尽管影子页表提供了在真实内存管理单元中能应用的页表，但是其缺点是实现非常复杂，开发、调试和维护都比较困难，最重要的是影子页表的内存开销比较大，因为须为所有虚拟实例对应的页表都维护一个影子页表。

相比之下, KVM 则由于内存虚拟化是由其模块完成的，所以支持 IntelCPU 提供的 EPT 技术。EPT 技术是针对内存管理单元的虚拟化扩展，主要是通过控制寄存器将虚拟实例程序所见的虚拟实例虚拟内存区转化为虚拟实例虚拟主内存区，然后在通过 EPT 将虚拟实例虚拟主内存区转化为宿主机主内存区，这两次地址转换都是直接由 CPU 硬件来自动完成的，其转换效率非常高。所以相比 Xen 采用的影子页表，降低了内存虚拟化实现的复杂度，提升了内存虚拟化的性能。

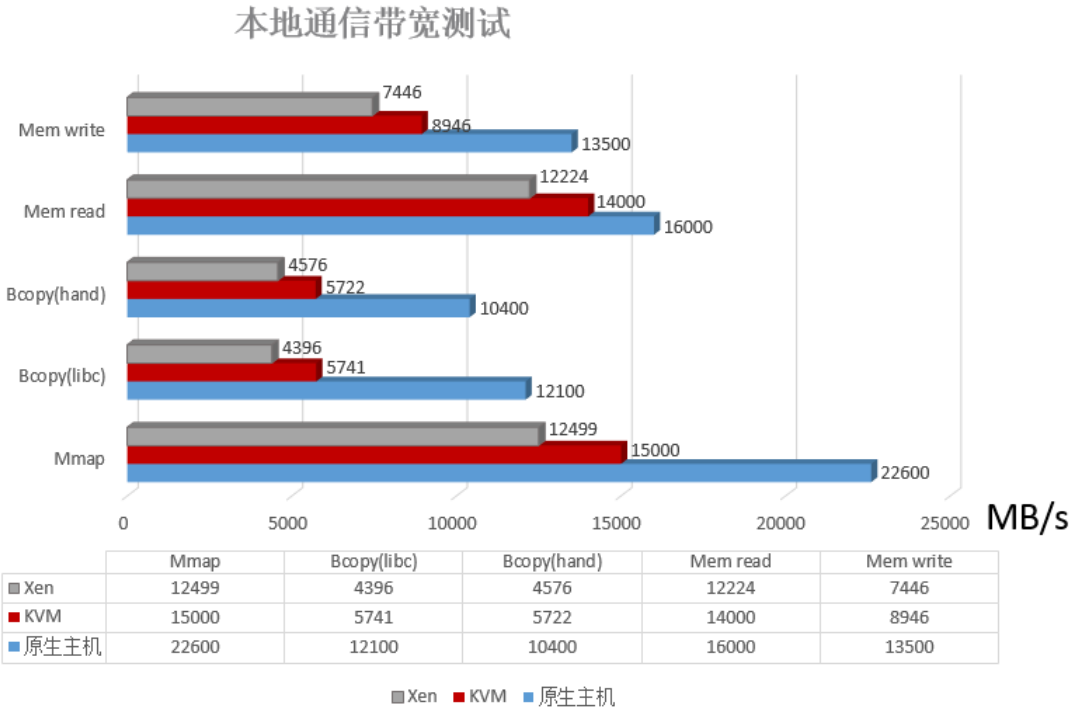


图 4-10 以文件服务器为工作负载，KVM、Xen 与原生主机的内存测试数据（本地通信带宽）

Figure4-10 The memory test data of KVM, Xen and Host based on file server (local communication bandwidth)

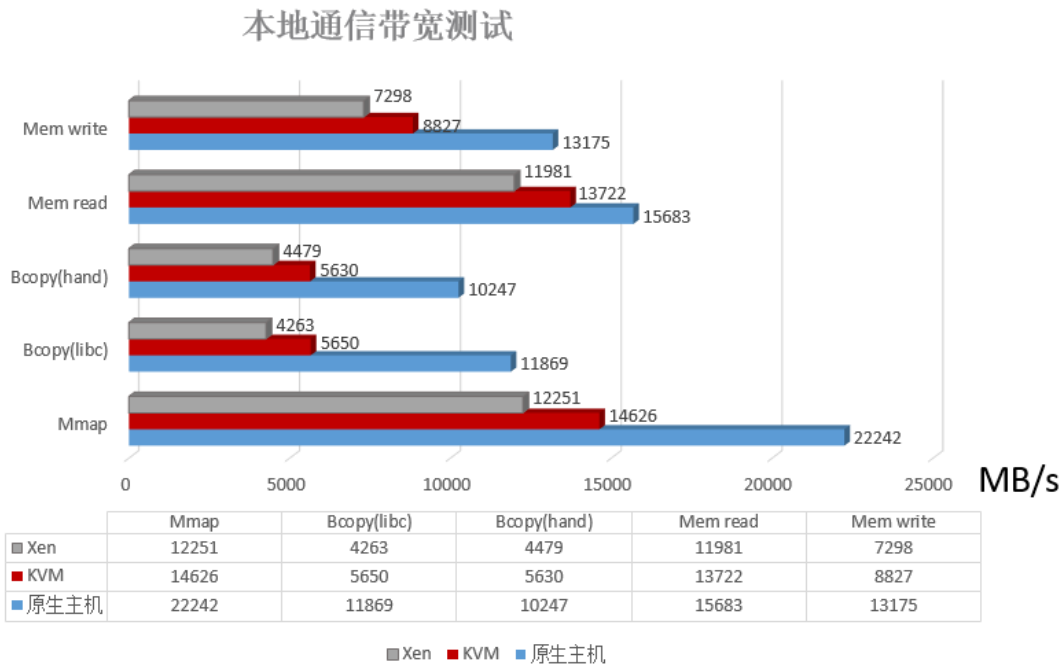


图 4-11 以邮件服务器为工作负载，KVM、Xen 与原生主机的内存测试数据（本地通信带宽）

Figure4-11 The memory test data of KVM, Xen and Host based on mail server (local communication bandwidth)

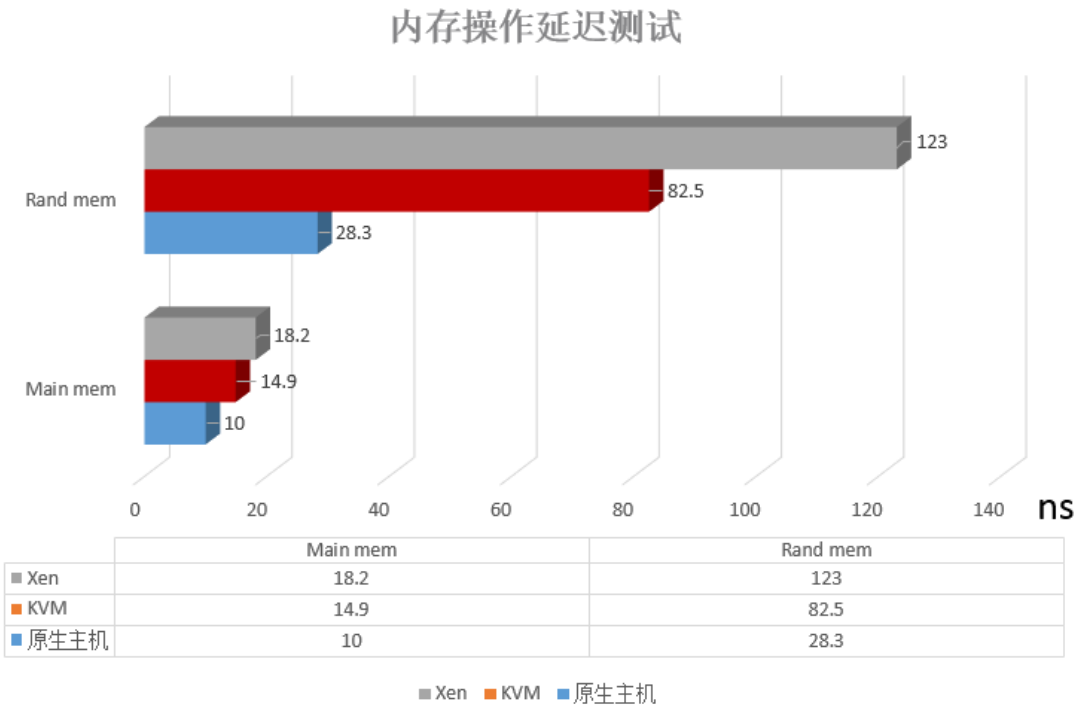


图 4-12 以文件服务器为工作负载，KVM、Xen 与原生主机的内存测试数据（内存操作延迟）

Figure4-12 The memory test data of KVM, Xen and Host based on file server (memory latencies)

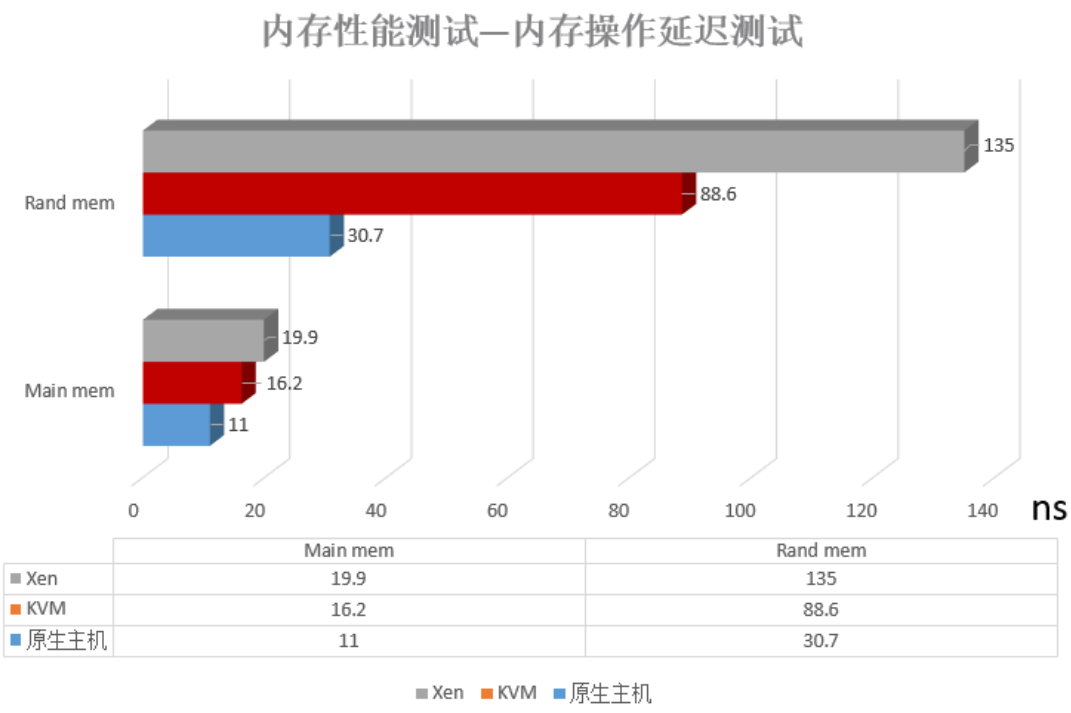


图 4-13 以邮件服务器为工作负载，KVM、Xen 与原生主机的内存测试数据（内存操作延迟）

Figure4-13 The memory test data of KVM, Xen and Host based on mail server (memory latencies)

4.3.2 性能隔离度测试结果与结论

通过采取虚拟化技术，能够减少物理资源的闲置和浪费。但是虚拟实例之间必须具有一定的隔离性能，以让它们彼此互不干扰。否则当多个虚拟实例被整合并运行在一个物理硬件之上，且某一虚拟实例使用过多的 CPU 资源或占用较大磁盘 I/O 时，就也许会影响到整个主机的运行，以致于整个主机瘫痪。所以在生产环境部署虚拟化方案时，性能隔离度测试就占有非常高的比重，因为我们绝对不希望在使用一个应用程序时，这个应用程序占用了全部的 CPU 和内存资源，导致无法并发运行其他的程序。

（1）CPU 性能隔离度测试。在性能隔离度的 CPU 性能测试中，所得数据如图 4-14 所示。当虚拟化环境中某一个虚拟实例异常活跃时（即负载很大时），那么这个虚拟实例将会占用绝大部分 CPU 的处理能力，为了保证性能隔离，Xen 和 KVM 采用了不同的机制。Xen 采用的是可信调度器，主要是针对在所有的逻辑 CPU 之上平衡虚拟 CPU 的工作，并限制一个行为异常或 CPU 密集型的虚拟实例占用过多的 CPU 资源。而在 KVM 中每个虚拟 CPU 都是宿主机中的一个普通的用户空间线程，且绑定至某一个或几个固定的 CPU 上去调度，当某一个虚拟实例使用的虚拟 CPU 的总数目大于实际具有的物理 CPU 的数目，用户空间就会启动更多的线程来为虚拟实例提供服务，这些线程也是被宿主机内核调度运行

在物理 CPU 硬件上。所以通过测试结果来看，显然 KVM 采用的绑定虚拟 CPU 的方法，使其性能隔离度要强 Xen 的共享虚拟 CPU 机制。

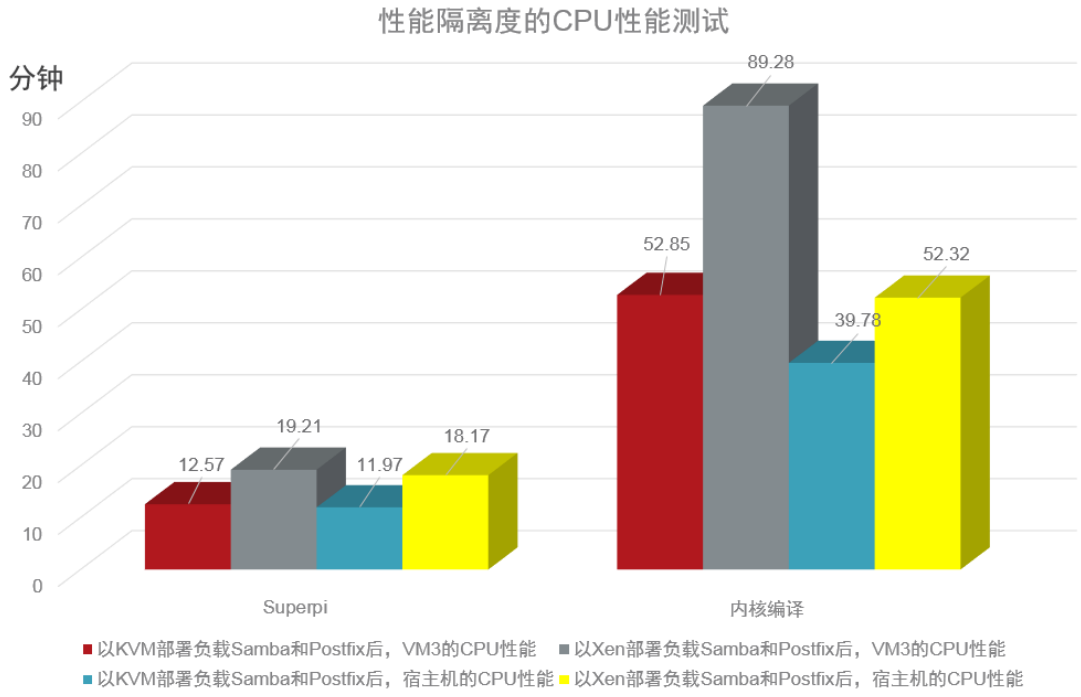
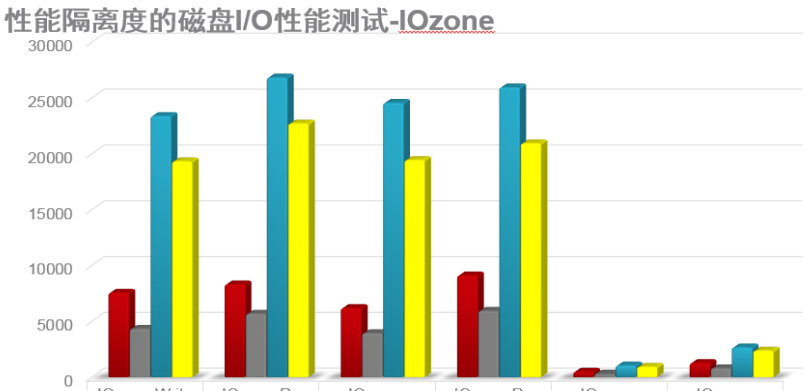


图 4-14 基于 KVM 与 Xen 部署具有负载的邮件服务器和文件服务器环境下的 CPU 隔离性能

Figure4-14 The CPU isolation performance test of both KVM and Xen based on utilized file server and mail server

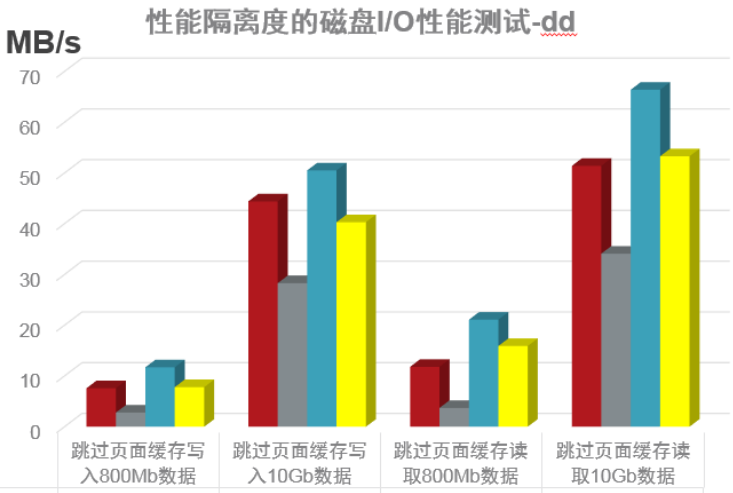
（2）磁盘 I/O 性能隔离度测试。在性能隔离度的磁盘 I/O 性能测试中，所得数据如图 4-15 和图 4-16 所示。由于 KVM 是由模块来虚拟 CPU 和内存，利用 QEMU 来虚拟磁盘和网络，这种分开的方式，使得各个虚拟硬件性能更加高效，在重负载的情况下，体现出来的优势尤为突出。而 Xen 完全是由 Domain0 来控制，管理和调度所有的硬件资源，并提供一个通用接口，这种模式的处理，各个虚拟硬件都受到的 domain0 的制约，性能会比分开管理要差，加之前面所叙述的 KVM 与 Xen 采用的是不同的镜像文件格式来存储虚拟机的信息，所以在重负载的情况下，KVM 关于磁盘写入/读取的性能隔离要强于 Xen。



	IOzone-Write	IOzone-Re Write	IOzone-Read	IOzone-Re Read	IOzone-Random Write	IOzone-Random Read
以KVM部署负载Samba和Postfix后，VM3的磁盘I/O性能	7557	8307	6187	9108	491	1261
以Xen部署负载Samba和Postfix后，VM3的磁盘I/O性能	4325	5678	3932	5942	294	795
以KVM部署负载Samba和Postfix后，宿主机的磁盘I/O性能	23388	26838	24564	25957	1043	2661
以Xen部署负载Samba和Postfix后，宿主机的磁盘I/O性能	19329	22717	19440	20939	924	2370

图 4-15 基于 KVM 与 Xen 部署具有负载的邮件服务器和文件服务器环境下的磁盘 I/O 隔离性能（IOzone）

Figure4-15 The disk I/O isolation performance test of both KVM and Xen based on utilized file server and mail server (IOzone)



	跳过页面缓存写入800Mb数据	跳过页面缓存写入10Gb数据	跳过页面缓存读取800Mb数据	跳过页面缓存读取10Gb数据
以KVM部署负载Samba和Postfix后，VM3的磁盘I/O性能	7.6	44.4	11.8	51.4
以Xen部署负载Samba和Postfix后，VM3的磁盘I/O性能	2.8	28.3	3.7	34.1
以KVM部署负载Samba和Postfix后，宿主机的磁盘I/O性能	11.7	50.5	21.1	66.4
以Xen部署负载Samba和Postfix后，宿主机的磁盘I/O性能	7.8	40.3	15.9	53.3

图 4-16 基于 KVM 与 Xen 部署具有负载的邮件服务器和文件服务器环境下的磁盘 I/O 隔离性能（dd）

Figure4-16 The disk I/O isolation performance test of both KVM and Xen based on utilized file server and mail server (dd)

（3）网络性能隔离度测试。在性能隔离度的网络性能测试中，所得数据如

图 4-17 所示。从测试结果上来看，虽然双方的网络隔离性能差距不大，但是还是可以判断出 KVM 的网络隔离性能比 Xen 的网络隔离性能更佳。这主要是因为 KVM 是利用 QEMU 来完成网络虚拟的，而 Xen 全部依靠 Domain0 来提供统一的访问接口。

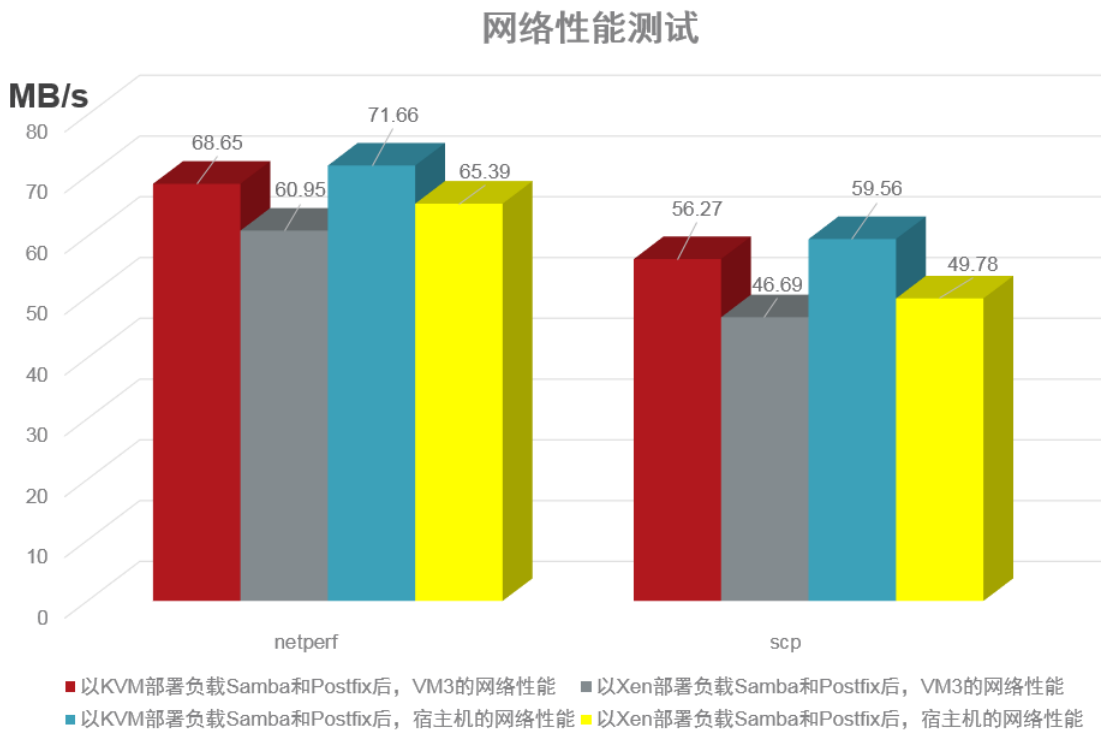
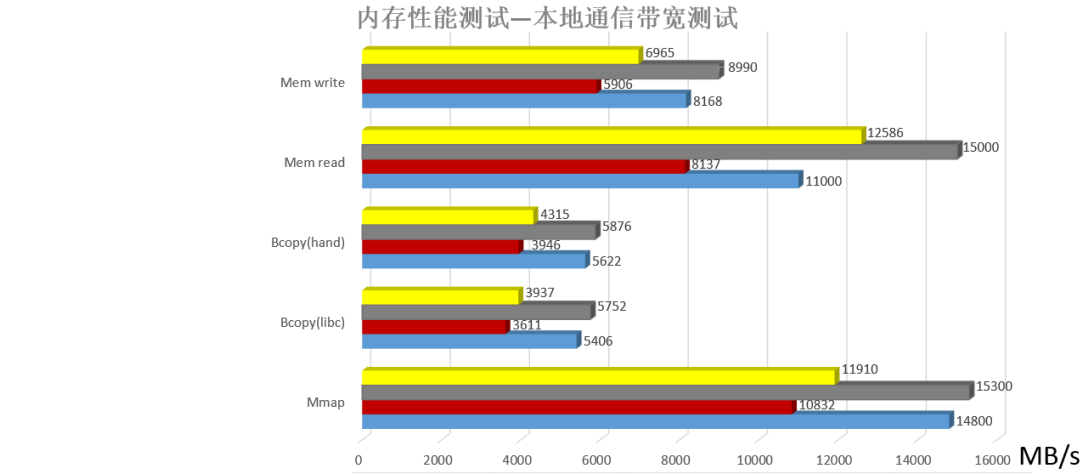


图 4-17 基于 KVM 与 Xen 部署具有负载的邮件服务器和文件服务器环境下的网络隔离性能

Figure4-17 The network isolation performance test of both KVM and Xen based on utilized file server and mail server

（4）内存性能隔离度测试。在性能隔离度的内存性能测试中，所得数据如图 4-18 和图 4-19 所示。在 KVM 和 Xen 的环境中，都准许内存过载的运用。且 KVM 和 Xen 都能够让划分给虚拟实例的内存数量超过真实内存数量。但双方对于实现内存过载的方式不一样。其中 Xen 所采用的气球驱动程序对于内存过载提供了一种特别而简单的替代方法。气球驱动程序不再改变通过虚拟实例内核寻址的主内存区数量，而是把内存页交给虚拟实例。这就意味着气球驱动程序将在虚拟实例消耗内存并允许 VMM 在其他地方分配内存，而虚拟实例内核仍然认为气球驱动程序拥有那部分被分配的内存，却并不知道这些内存可能正在被一个虚拟实例使用。而 KVM 采用的 KSM 内核同页合并技术，KSM 允许内核在两个或多个进程，包括虚拟实例，之间共享完全相同的内存页，且在 KVM 中，一个虚拟实例就是一个用户空间进程，所以使用 KSM 可以实现多个虚拟实例之间的相同内存合并，此外 KSM 只会识别并合并那些不会干扰虚拟实例运行，不会影响

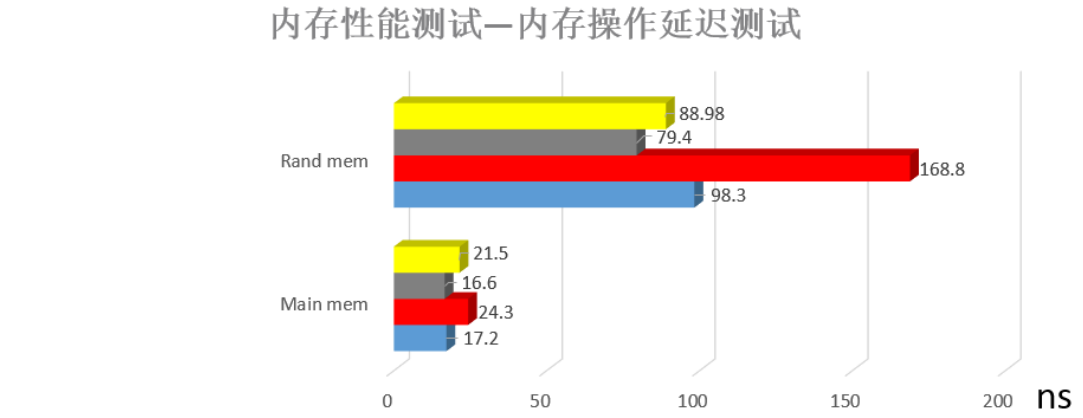
宿主机或虚拟实例的安全内存页，所以在 KVM 虚拟化环境中，KSM 能够提高内存的速度和使用效率。从测试结果来看，由于 Xen 与 KVM 实现内存过载的机制不同，导致 KVM 的内存隔离性能要明显优于 Xen 的内存隔离性能。



	Mmap	Bcopy(libc)	Bcopy(hand)	Mem read	Mem write
以Xen部署负载Samba和Postfix后，宿主机的内存性能	11910	3937	4315	12586	6965
以KVM部署负载Samba和Postfix后，宿主机的内存性能	15300	5752	5876	15000	8990
以Xen部署负载Samba和Postfix后，VM3的内存性能	10832	3611	3946	8137	5906
以KVM部署负载Samba和Postfix后，VM3的内存性能	14800	5406	5622	11000	8168

图 4-18 基于 KVM 与 Xen 部署具有负载的邮件服务器和文件服务器环境下的内存隔离性能（本地通信带宽）

Figure4-18 The memory isolation performance test of both KVM and Xen based on utilized file server and mail server (local communication bandwidth)



	Main mem	Rand mem
以Xen部署负载Samba和Postfix后，宿主机的内存性能	21.5	88.98
以KVM部署负载Samba和Postfix后，宿主机的内存性能	16.6	79.4
以Xen部署负载Samba和Postfix后，VM3的内存性能	24.3	168.8
以KVM部署负载Samba和Postfix后，VM3的内存性能	17.2	98.3

图 4-19 基于 KVM 与 Xen 部署具有负载的邮件服务器和文件服务器环境下的内存隔离性能

（内存操作延迟）

Figure4-19 The memory isolation performance test of both KVM and Xen based on utilized file server and mail server (memory latencies)

4.3.3 可扩充性测试结果与结论

既然采用虚拟化，就是为了最大程度的使用硬件资源，以实现节约成本的目的。那么虚拟化方案可扩充性越佳，则在节约成本和物理资源最大化方面就能体现的越明显。所以，依据可扩充性测试方案，分别对 KVM 和 Xen 两种虚拟化环境进行了详细的扩充性测试。

（1）在可扩充性能测试中对 CPU 进行测试。基于 KVM 和 Xen 各自虚拟化环境下，在部署 6 个虚拟实例以及相应的工作负载之后，双方的 CPU 性能测试结果如图 4-20 所示。可以明显的发现，当双方都部署到 6 个工作负载的时候，KVM 的 CPU 性能甚至超过基于 Xen 部署四个工作负载。

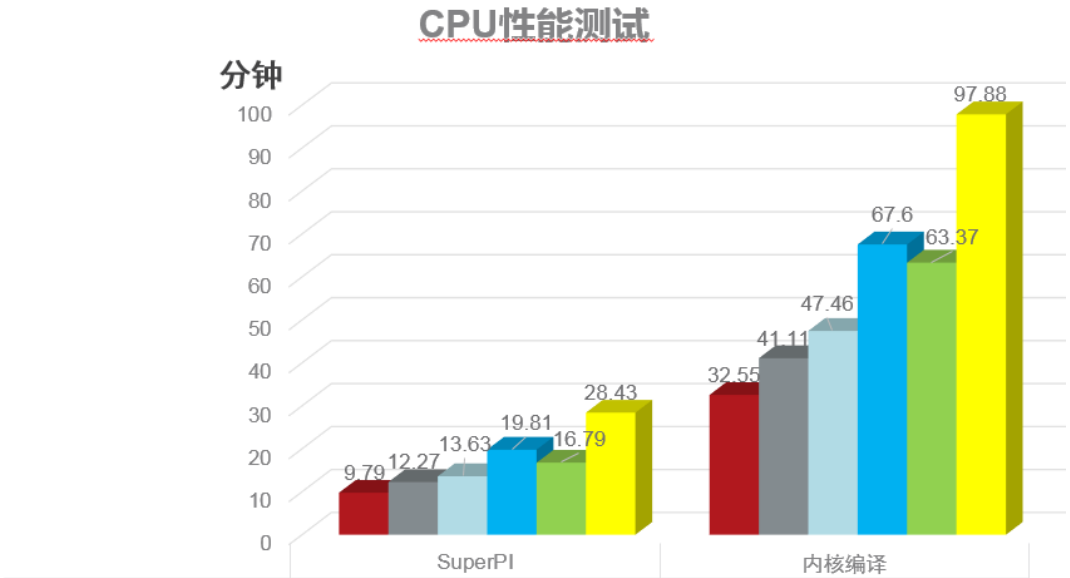
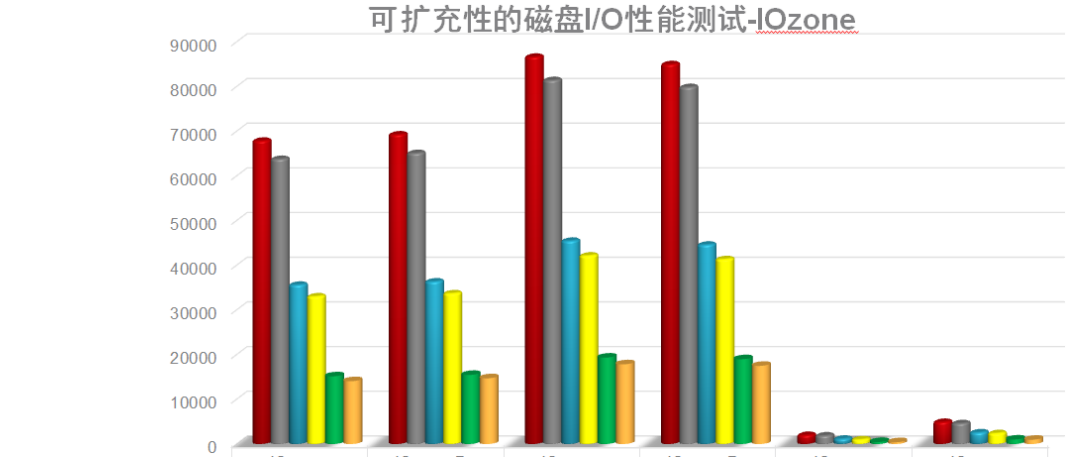


图 4-20 扩充性测试中的 CPU 性能测试

Figure 4-20 The CPU performance test data of expandability

（2）在可扩充性测试中对磁盘 I/O 性能进行测试，测试结果如图 4-21 和图 4-22 所示。在 KVM 和 Xen 各自的环境下，部署 6 个虚拟实例以及相应的工作负

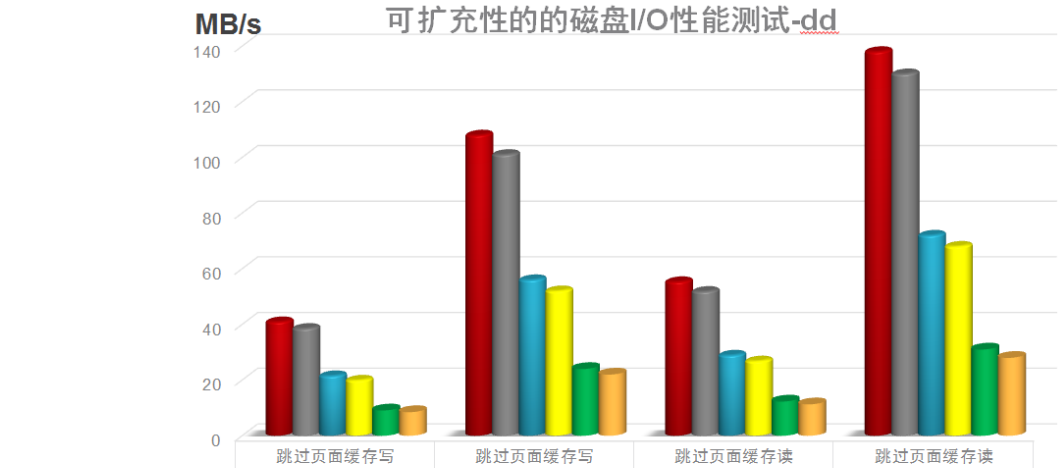
载之后，双方的磁盘写入/读取性能都开始呈指数级下降，尽管如此，以 KVM 部署出的虚拟化环境的磁盘写入/读取效能仍然强于以 Xen 部署出的虚拟化环境的磁盘写入/读取效能。



	IOzone-Write	IOzone-Re Write	IOzone-Read	IOzone-Re Read	IOzone-Random Write	IOzone-Random Read
基于KVM部署两个工作负载	67858	69267	86640	84982	1999	4917
基于Xen部署两个工作负载	63770	65094	81420	79863	1879	4621
基于KVM部署四个工作负载	35609	36349	45465	44596	1156	2580
基于Xen部署四个工作负载	33013	33698	42150	41344	1006	2392
基于KVM部署六个工作负载	15158	15473	19353	18983	446	998
基于Xen部署六个工作负载	14075	14721	17850	17509	411	956

图 4-21 扩充性测试中的磁盘 I/O 性能测试(IOzone)

Figure4-21 The disk I/O performance test data of expandability (IOzone)



	跳过页面缓存写入800Mb数据	跳过页面缓存写入10Gb数据	跳过页面缓存读取800Mb数据	跳过页面缓存读取10Gb数据
基于KVM部署两个工作负载	40.9	108	55.3	138
基于Xen部署两个工作负载	38.5	101	51.8	130
基于KVM部署四个工作负载	21.5	56	28.9	72
基于Xen部署四个工作负载	19.9	52	26.8	68
基于KVM部署六个工作负载	9.1	24	12.3	31
基于Xen部署六个工作负载	8.5	22	11.3	28

图 4-22 扩充性测试中的磁盘 I/O 性能测试(dd)

Figure4-22 The disk I/O performance test data of expandability (dd)

（3）在可扩充性测试中对网络性能进行测试，测试结果如图 4-23 所示。在两种虚拟化环境下，部署至 6 个工作负载之后，网络性能也随之大幅下降。但仍然可以观察到，以 KVM 部署出的虚拟化环境的网络性能仍然要快于以 Xen 部署出的虚拟化环境的网络性能。

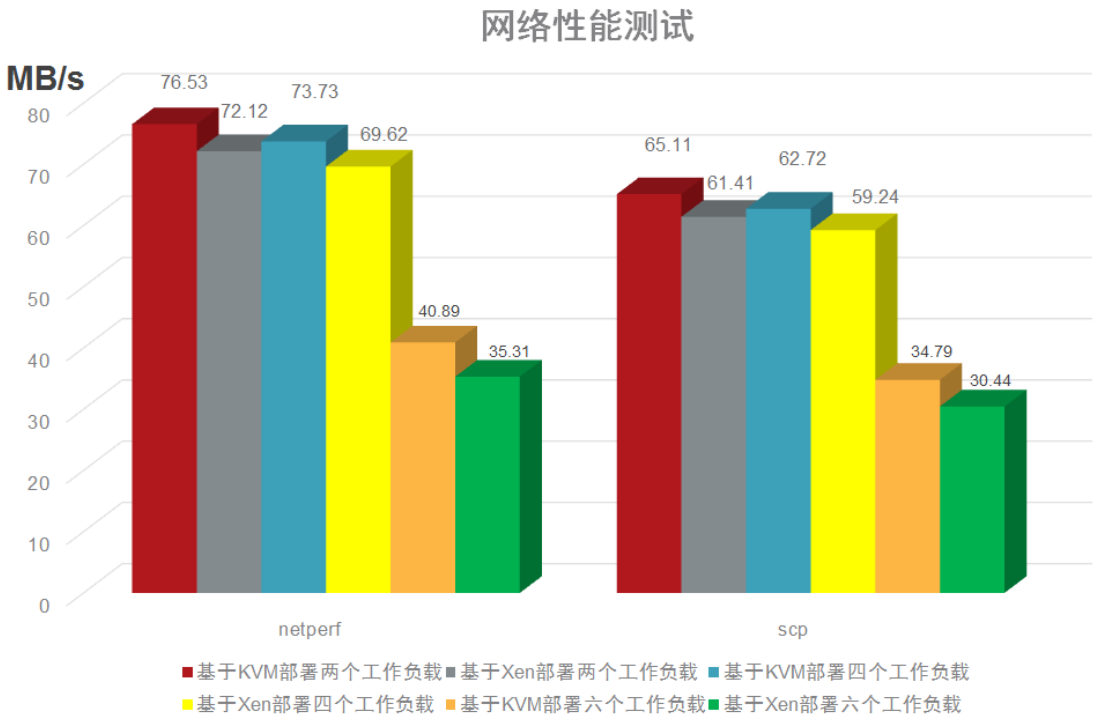
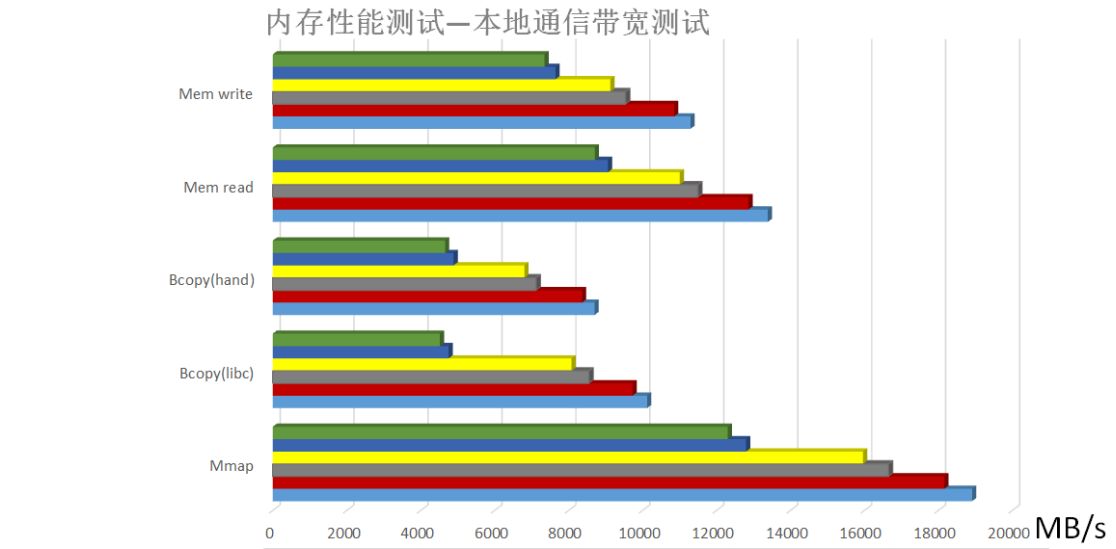


图 4-23 扩充性测试中的网络性能测试

Figure4-23 The network performance test data of expandability

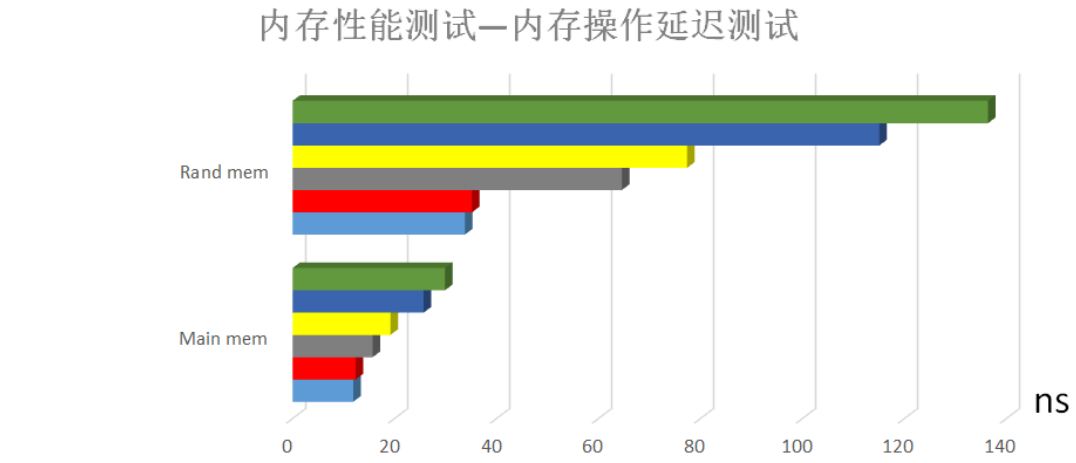
（4）在可扩充性能测试中对内存进行测试。基于 KVM 和 Xen 各自虚拟化环境下，在部署 6 个客户系统实例以及相应的工作负载之后，双方的内存性能测试结果如图 4-24 和图 4-25 所示。虽然在通信带宽和延迟测试中，两种虚拟化的内存性能都随着工作负载的增加而减少，只是减少的部分在直观上影响不大。尽管如此，仍然可以明显看出，以 KVM 部署出的虚拟化环境的内存性能仍然要优于以 Xen 部署出的虚拟化环境的内存性能。



	Mmap	Bcopy(libc)	Bcopy(hand)	Mem read	Mem write
基于Xen部署六个工作负载	12305	4523	4662	8711	7350
基于KVM部署六个工作负载	12803	4755	4891	9064	7647
基于Xen部署四个工作负载	15963	8082	6806	11009	9133
基于KVM部署四个工作负载	16651	8550	7122	11496	9543
基于Xen部署两个工作负载	18163	9731	8364	12868	10857
基于KVM部署两个工作负载	18914	10126	8703	13390	11298

图 4-24 扩充性测试中的内存性能测试（本地通信带宽）

Figure4-24 The memory performance test data of expandability (local communication bandwidth)



	Main mem	Rand mem
基于Xen部署六个工作负载	29.9	136.4
基于KVM部署六个工作负载	25.7	115.1
基于Xen部署四个工作负载	19.2	77.4
基于KVM部署四个工作负载	15.7	64.6
基于Xen部署两个工作负载	12.4	35.2
基于KVM部署两个工作负载	11.9	33.8

图 4-25 扩充性测试中的内存性能测试（内存操作延迟）

Figure4-25 The memory performance test data of expandability (memory latencies)

4.4 本章小结

本章介绍了在KVM和Xen各自的虚拟化环境下部署的工作负载与非虚拟化原生主机中运行相同的测试，并将测试结果进行了对比，以测试Xen和KVM的性能。并详细的介绍了虚拟化性能测试的准则和方案，针对 CPU，磁盘，内存和网络这些子系统进行了深入的测试，且详细的说明了所得测试结果之间差异的本质所在。综合宏观性能，隔离性能和可扩充性能这三种方案的测试数据，对两种虚拟化技术的性能做出相应的结论。

结论

本文主要围绕两个开源虚拟化方案 Xen 与 KVM 产开叙述，根据公司项目要求，以及针对目前还不存在一套基准性能测试方法，来度量哪一种虚拟化的性能更加优越，设计并实施了一整套生产环境方案以及工作负载。从数据安全，磁盘空间弹性增减和存储资源池三方面的角度考虑，RAID 和 LVM 无疑是非常完美的组合，且通过对吞吐量进行详细的测试，从而选择出在安全还是空间利用率方面都很优秀的阵列方案。在此基础之上，设计了详细的性能测试方法和方案，并严格遵照此方法和方案对两种开源虚拟化方案进行了深入的测试，并对测试结果进行了详细的分析。

由于拥有众多的优点，所以虚拟化已经是每个企业必有的项目。在对两种虚拟化技术进行详细的测试和分析后，得出的主要如下 3 点结论：

（1）虽然 Xen 是一个开发较早的虚拟化方案，VMM 是专门为虚拟机研发的微内核，其资源管理和调度策略也完全是针对虚拟机的特性而开发，所以能相对完善的支持各种虚拟化功能。尽管 Xen 的功能比较强大，但相对的难于配置和使用，且部署时会占用较大的磁盘空间，更重要的是因为其虚拟机监控器并没有精确的模拟物理设备的实际硬件特性，而只是提供了统一的、理想化的设备访问模型，所以虚拟实例的配置和初始化工作，以及使用模拟设备时就需要 Domain0 将相应的指令转换为物理硬件指令才能进行操作。种种这些缺点导致直接将 Xen 微内核运行于实际物理硬件之上，研发和调试都比基于操作系统的虚拟化困难，再加之由于内核开发人员对 Xen 的架构和实现产生强烈的抵触，导致 Xen 的相关改动一直不能顺利的进入内核源代码，这也加重 Linux 内核负担。

（2）相比之下，尽管 KVM 是与 Linux 内核集成且需要硬件虚拟化支持的方案。但也正因为 KVM 工作于内核环境中，使其能够充分的利用已有的操作系统，直接和硬件以及集成到 Linux 内核中的虚拟化管理程序进行交互，从而减少虚拟机监控器在实现管理物理资源和调度算法上的压力，更直接的硬件调度代表更高的效率，所以其执行效率要比传统意义上的虚拟机技术高很多，以使得在虚拟机运行方面，KVM 是一个效率更高的方案。虽然 KVM 是相对较新的技术，但在 OpenStack 等云计算平台的推动下，使得 KVM 发展十分迅速，且经过本文中的性能测试，可以确定 KVM 拥有更好的进程调度支持、虚拟网络的支持、虚拟存储支持、虚拟 CPU 支持以及更佳的高可用性。加之全虚拟化的性能已经超过了准虚拟化，这让基于硬件的全虚拟化将是虚拟化技术的核心。相信未来 KVM 的功能会更加的强大，且性能会更加的高效，势必使得 KVM 将会成为开源虚拟

化方案中的主流虚拟化技术。

（3）本文中实现的存储架构和部署的两种工作负载都非常具有代表性，几乎是所有企业中必不可少的服务，这一点足以体现本文的参考价值与实际意义。经过详细性能测试，发现了一个无法回避的问题，即在保证每个虚拟实例都没有故障的前提下，虚拟机的各项性能指数都有正常的损耗，但是磁盘 I/O 的性能损耗比较大，即使为每个虚拟实例都分配了固定的存储空间，但是 I/O 指数仍然较高，这也是当前虚拟化需要研究和突破的重点。尽管如此，只要不在虚拟实例中部署磁盘写入/读取聚集型的工作负载，虚拟化的优势还是无可比拟的。加之现在已经出现闪存阵列技术，相信在未来硬件性能将不再是虚拟化的应用瓶颈。

最后，在企业生产环境下开源虚拟化的实践和经验都是最宝贵的财富，在本文中未涉及到的负载均衡和动态迁移等技术都是以后需要继续研究和解决的问题，在日后的工作中，应尽可能更多的了解虚拟化的相关技术和性能，扬长避短，最大限度的发挥其功能。

参考文献

[1] 马博峰著. VMware、Citrix 和 Microsoft 虚拟化技术详解与应用实践[M]. 机械工业出版社，2012.

[2] Graziano, Charles David. A Performance Analysis of Xen and KVM[M]. Proquest, Umi Dissertation Publishing，2012.

[3] 任永杰，单海涛著. KVM 虚拟化技术:实战与原理解析[M]. 机械工业出版社，2013

[4] 祁伟，刘冰，路士华等著. 云计算:从基础架构到最佳实践[M]. 清华大学出版社，2013.

[5] Jeanna Matthews 著，张炯，吕孟轩，刘铭等译. 运行 Xen:虚拟化艺术指南[M]. 北京航空航天大学出版社，2014.

[6] 高俊峰著. 高性能 Linux 服务器构建实战：系统安全、故障排查、自动化运维与集群架构[M]. 机械工业出版社，2014.

[7] Derek Vadala. Managing RAID on Linux[M]. O'Reilly Media，2002.

[8] 戢友著. OpenStack 开源云王者归来[M]. 清华大学出版社，2014.

[9] Michael Kolfer. KVM---Virtualisierung unter Linux[M]. Ebooks. kofler，2011.

[10] Sander van Vugt. Red Hat Enterprise Linux 6 Administration: Real World Skills For Red Hat Administrators[M]. Wiley，2013.

[11] Dan Kusnetzky. Virtualization: A Manager's Guide[M]. O'Reilly Media，2011.

[12] Robert Love 著，陈莉君，康华译. Linux 内核设计与实现（原书第 3 版）[M]. 机械工业出版，2011.

[13] 刘晓辉，陈洪彬著. Red Hat Linux 服务器管理及配置实战详解[M]. 化学工业出版社，2010.

[14] 石磊，邹德清，金海著. Xen 虚拟化技术[M]. 华中科技大学出版社，2009.

[15] David Chisnall. The Definitive Guide to the Xen Hypervisor[M]. Prentice Hall，2013.

[16] Chris Takemura, Luke S. Crawford. The Book of Xen: A Practical Guide for the System Administrator[M]. NO STARCH PRESS，2009.

[17] 余洪春著. 构建高可用 Linux 服务器（第 3 版）[M]. 机械工业出版，2014.

[18] 高俊峰著. 循序渐进 Linux---基础知识、服务器搭建、系统管理、性能调优、集群应用[M]. 机械工业出版社，2014.

[19] 高俊峰著. 高性能 Linux 服务器构建实战:运维监控、性能调优与集群应用[M]. 机械

工业出版社，2014.

[20] 王柏生著. 深度探索 Linux 操作系统:系统构建和原理解析[M]. 机械工业出版社，2013.

[21] Thomas Nadeau D，Ken Gray. SDN: Software Defined Networks[M]. O'Reilly Media，2014.

[22] 陶利军著. 掌控:构建 Linux 系统 Nagios 监控服务器[M]. 清华大学出版，2013.

[23] Don R Crawley. The Accidental Administrator: Linux Server Step-by-Step Configuration Guide[M]. CreateSpace Independent Publishing Platform，2010.

[24] Ed Sawicki. Advanced Guide to Linux Networking and Security[R]. Cengage Learning，2005.

[25] Carter Gerald, Eckstein Robert. Using Samba: A File and Print Server for Linux, Unix & Mac OS X[M]. O'Reilly Media，2007.

[26] Kyle D. Dent. Postfix: The Definitive Guide[M]. O'Reilly Media，2004.

[27] Eleen Frisch. Essential System Administration: Tools and Techniques for Linux and Unix Administration[M]. O'Reilly Media，2002.

[28] David E. Williams. Virtualization with Xen[M]. Syngress，2007.

[29] Rogier Dittner, David Rule Jr. The Best Damn Server Virtualization Book Period[M]. Syngress，2007.

[30] William von Hagen. Professional Xen Virtualizatio[M]. Wiley，2009.

[31] 姜凯，李帮锐，谢一凡等著. 桌面虚拟化实战宝典[M]. 电子工业出版社，2014.

[32] 李洋等著. Linux 安全策略与实例[M]. 机械工业出版社，2009.

[33] James Kirkland, David Carmichael, Christopher L.Tinker, Gregory L.Tinker 著，周良忠等译. Linux 系统故障诊断与排除[M]. 人民邮电出版社，2007.

[34] Steve Suehring. Linux Firewalls: Enhancing Security with nftables and Beyond[M]. Addison-Wesley Professional，2015.

[35] Amy Newman, Kenneth Hess. Practical Virtualization Solutions: Virtualization from the Trenches[M]. Prentice Hall，2009.

[36] Jeanna Matthews. Running XEN:A Hands-on Guide to The Art of Virtualization[M]. Prentice Hall PTR，2008.

[37] UK Research funding Council, Intel, Network Appliance, and XenSource. The Xen virtual machine monitor[R]. University of CAMBRIDGE，2008.

[38] Gregor N. Purdy. Linux iptables Pocket Reference[M]. O'Reilly Media，2004.

［39］Michael Rash. Linux Firewalls : Attack Detection and Response with iptables, psad, and fwsnort［M］. No Starch Press, 2007.

致谢

转眼间，研究生生活即将结束。回想这段时间的求学历程，对那些引导我、帮助我、鼓励我的人，我心中充满了感激。

首先要感谢我的导师徐旭东教授，他严谨的治学态度、深厚的理论素养、丰富的实践知识以及科学的工作方法都给了我极大的帮助和启发。本论文从定题到写作成稿，徐老师都倾注了大量的心血，尽管身负教学、科研的重任，仍多次悉心指导，为我指点迷津。我的毕业论文能够顺利完成，与徐老师无私帮助和热忱鼓励是分不开的。再次，谨向徐老师表示崇高的敬意和由衷的感谢。

其次感谢一直关心与帮助我的同事和朋友们，在论文中的项目实施时，他们给与了我很多宝贵的建议，再次同事之情，朋友之谊，我终身难忘。

最后感谢我的家人，感谢他们对与我的包容，是他们的深切关怀和大力支持才让我能顺利完成学业。