中图分类号：TP316

论 文 编 号 ：10006ZY1306217

北京航空航天大學

专业硕士学位论文

基于 KVM 虚拟化服务器的性能评估模型研究

作 者 姓 名　　杨 静

学 科 专 业　　计算机技术

指 导 教 师　　吴 超 英　副 教 授

培 养 院 系　　计 算 机 学 院

Research of the Performance Evaluation Model for Virtual Servers in KVM-based Virtualized System

A Dissertation Submitted for the Degree of Master

Candidate：Yang Jing

Supervisor：Associate Prof. Wu Chaoying

School of Computer Science and Engineering
Beihang University, Beijing, China

中图分类号：TP316

论 文 编 号 ：10006ZY1306217

专 业 硕 士 学 位 论 文

基于 KVM 虚拟化服务器的性能评估模型研究

作者姓名	杨静	申请学位级别	工程硕士
指导教师姓名	吴超英	职　　称	副教授
学科专业	计算机技术	研究方向	软件工程
学习时间自	年　月　日	起至	年　月　日止
论文提交日期	年　月　日	论文答辩日期	年　月　日
学位授予单位	北京航空航天大学	学位授予日期	年　月　日

关于学位论文的独创性声明

本人郑重声明：所呈交的论文是本人在指导教师指导下独立进行研究工作所取得的成果，论文中有关资料和数据是实事求是的。尽我所知，除文中已经加以标注和致谢外，本论文不包含其他人已经发表或撰写的研究成果，也不包含本人或他人为获得北京航空航天大学或其它教育机构的学位或学历证书而使用过的材料。与我一同工作的同志对研究所做的任何贡献均已在论文中作出了明确的说明。

若有不实之处，本人愿意承担相关法律责任。

学位论文作者签名：_____ 日期： 年 月 日

学位论文使用授权书

本人完全同意北京航空航天大学有权使用本学位论文（包括但不限于其印刷版和电子版），使用方式包括但不限于：保留学位论文，按规定向国家有关部门（机构）送交学位论文，以学术交流为目的赠送和交换学位论文，允许学位论文被查阅、借阅和复印，将学位论文的全部或部分内容编入有关数据库进行检索，采用影印、缩印或其他复制手段保存学位论文。

保密学位论文在解密后的使用授权同上。

学位论文作者签名：_____ 日期： 年 月 日

指导教师签名：_____ 日期： 年 月 日

摘 要

调查数据显示，传统服务器部署模式的资源利用率低，能源消耗较高。服务器虚拟化可以将多个虚拟服务器整合在一台物理主机上，节约了资源与能源，因此虚拟化服务器部署模式成为越来越多服务提供商的选择，服务器虚拟化技术成为目前研究的热点。虚拟机管理器是服务器虚拟化的核心，对虚拟化系统中的资源进行调度管理，实现虚拟机中虚拟资源到物理资源的映射。服务器虚拟化中虚拟机管理器的引入增加了系统的复杂性与性能消耗，性能问题一直是服务器虚拟化领域的研究重点。

KVM(Kernel-based Virtual Machine)是 Linux 默认虚拟化实现方案，2011 年 IBM 和红帽等硬件供应商成立了开放虚拟化联盟以促进 KVM 发展，KVM 使用量得到迅速增长。目前针对 KVM 虚拟化服务器的性能研究主要关注于虚拟化引起的性能消耗，当虚拟服务器为用户提供访问服务时，其应用性能是服务供应商关注的重点。另一方面，当应用性能不能满足用户要求时，如何提高虚拟化服务器性能是服务提供商面临的问题。对虚拟化服务器应用性能评估的同时，得到其各组件性能，发现性能瓶颈，是改进应用性能的必要条件。本研究针对 KVM 虚拟化服务器建立性能评估模型，借助应用性能来评估虚拟服务器性能，并获得其主要组件性能。本文工作包括以下几点：

1. 分析虚拟化服务器性能度量项，明确 KVM 中应用性能评估指标。
2. 基于虚拟化服务器性能度量指标及其影响因素，详细分析 KVM 中虚拟化实现方式，采用排队网络模型建立性能评估模型。
3. 根据虚拟化服务器性能影响因素，选择实验变量进行实验，通过将性能评估模型获得的评估结果与相关基准测试工具评估结果进行对比分析，验证模型的正确性。
4. 针对应用环境中虚拟化服务器配置问题，基于性能评估模型，提出虚拟化服务器配置决策方法，并结合具体实例进行说明。

关键词：服务器虚拟化，性能评估，排队网络模型，KVM

Abstract

According to the survey, there are low utilization of resources and high energy consumption in the traditional server pattern. Server virtualization can integrate several virtual servers into one physical server and save the resources and energy, so more and more service providers choose virtual servers as default server deployment pattern. Now the server virtualization has been a research focus. The virtual machine monitor is the core of server virtualization that manages the resources and maps virtual resources to physical resources in the virtual servers. It increases the complexity of the system and leads to performance overhead, so the performance problem is the research emphasis of sever virtualization.

KVM (Kernel-based Virtual Machine) is the default virtualization solution for Linux. In 2011, several famous companies founded the open virtualization alliance to promote the KVM development. The number of KVM grows rapidly. Now the performance study of KVM focuses on its performance overhead. When the virtual server provides the service for end-users, the performance of application is concerned by service providers. On the other hand, when the performance of application can't meet the user demands, how to improve the performance is the problem faced by service providers. It is the essential condition to get the performance of main components in virtual servers for improvement the virtual server performance. In this paper, we build the performance evaluation model for virtual servers in KVM-based virtualized system and evaluate the performance of virtual servers by application performance running on them. The works in this paper are as follows:

1. Analyze the performance metrics in virtual servers, and determine the performance evaluation metrics in application layer of KVM.
2. Based on the performance metrics and its influence factors, analyze the implementation of sever virtualization in KVM to build the performance evaluation model by queuing network method.
3. According to the performance-influencing factors, design the experiments to verify the correctness of the performance evaluation model.
4. For the configuration problem of virtual servers in the application scenario, based on the performance evaluation model, give the method to determine the configuration of virtual servers and illustrate it by a specific example.

Keywords: Server Virtualization, Performance Evaluation, Queuing Network Model, KVM

目 录

图 目

表 目

第一章 绪论

1.1 研究背景与意义

随着互联网服务的逐渐普及，服务数据，计算需求等急剧增长，为了保证用户服务质量，服务供应商的服务器集群不断增大，相应的能耗与管理成本也不断增加。统计数据[1]显示 Google、微软、ebay、雅虎与 Facebook 等所拥有的服务器数量均在几十万台以上。但根据著名信息技术研究和顾问咨询公司 Garter 统计报告，大部分企业级应用服务器的平均资源利用率不足 20%，服务器长时间空载或者低载运转浪费了大量的企业运营成本[2]。为了解决服务器数量不断增加与利用率过低的矛盾，减少数据中心的资源消耗，实现绿色 IT，各大服务提供商开始将虚拟化技术应用到服务器资源的整合中，减少服务器数量，提高资源利用率。

虚拟机的概念早在 40 年前就已经被提出。上世纪 60 年代，在个人计算机出现之前，IBM 为其大型机量身打造了操作系统 CP 系列，并提出了管理程序的概念，该分时操作系统提供了硬件的完全模拟，使多个用户同时使用同一台物理主机。后来伴随着硬件的发展，集成电路的出现，计算机体积逐渐减小，个人计算机出现，虚拟化技术逐渐沉寂。八十年代到九十年代早期，由于分布式计算的引入，构建分布式体系的巨大成本使得人们又将目光转向虚拟化技术[3]。近年来，由于云计算概念以及服务器整合的提出使得虚拟化技术成为目前最流行的技术之一。

服务器虚拟化技术将一个物理服务器划分为几个独立和隔离的空间，使得多台服务器可以运行在一个物理服务器之上，提高了资源使用率，节约了成本。现在常用的虚拟化平台有 Citrix XenServer，VMware ESXi 与 Virtual Box 等。在 2014 年，将虚拟服务器作为默认的服务器部署模式的服务器数量占新增服务器数量的 80%，预计在 2017 年将达到 88%[4]。最近几年，Linux 服务器增长迅速，预计在 2017 年，新使用的服务器数量将达 72,000,000。目前 KVM(Kernel-based Virtual Machine)是 Linux 内核的一部分，并且与 OpenStack 云计算平台联系紧密，这为 KVM 的使用提供了巨大的机会，国际数据公司 IDC 的服务器虚拟化追踪报告显示，自从 2011 年起，新使用的 KVM 已经超过 27 万，年复合增长率达 42%[4]。

虽然服务器虚拟化技术可以使多个虚拟化服务器运行在一个物理服务器之上提高了硬件利用率，减少了能源消耗以及空间占用，但是服务器虚拟化的使用引入了逻辑资源到物理资源的转换，应用之间共享资源的分配等问题，也增加了系统的动态性和复杂

性。一般而言，虚拟化服务器的性能要低于物理服务器性能，因此如何评价虚拟化服务器性能，判断其是否能够满足用户需求成为研究的重点。虚拟机管理器是整个虚拟化系统核心，对物理硬件进行了抽象，为虚拟机访问实际硬件资源提供了接口，不同类型的虚拟机管理器对虚拟化服务器有不同影响。由于 Linux 服务器应用的广泛性，KVM 正逐渐替代 Xen 成为流行的虚拟化解决方案，因此对 KVM 平台中的虚拟化服务器性能进行研究是十分必要的。目前对 KVM 的服务器虚拟化性能研究，主要关注于 KVM 与其他虚拟化解决方案性能比较以及 KVM 虚拟化性能消耗，在服务提供者使用 KVM 为用户提供服务时，其更关注虚拟服务器之上的应用性能。本文着眼于 KVM 虚拟服务器中的应用，分析评价虚拟化服务器性能，并通过对虚拟化服务器性能评估，比较特定应用环境中虚拟服务器配置方案的优劣。

1.2 国内外研究现状

目前虚拟化服务器性能评估方法主要分为两类，分别为分析建模的方法和模拟的方法[5]。模拟的方法主要是通过基准测试工具来模拟访问虚拟化服务器的实际请求，建立真实的虚拟化服务器使用环境，通过基准测试工具输出的结果来评估虚拟化服务器性能。分析建模的方法主要是借助数学模型，如排队论，神经网络等，对虚拟化系统进行分析建模，并结合服务器负载的特征得到服务器性能计算方法。

1.2.1 基于模拟的性能评估方法

虚拟化服务器性能评估的模拟化方法是采用基准测试工具模拟某些应用场景，通过获得的定量性能度量值来评估虚拟化服务器性能的方法。基准测试将一个公司的业务过程和性能度量值与业界最好过程或者与业界最佳实践进行比较。基准测试以一系列标准作为参考来评估服务器性能。目前主要有两种类型基准测试工具，分别为度量每个组件性能的基准测试工具与度量端到端性能，即整个系统性能的基准测试工具。

虚拟化系统中单个组件性能主要是指其 CPU 性能，内存性能，网络性能和磁盘 I/O 性能，不同组件对应不同基准测试工具。

CPU 性能测试工具[6]包括 SPEC CPU2006，SPECjbb2013 与 SysBench 等，SPEC（Standard Performance Evaluation Corporation）是非盈利标准化组织，专注于建立，维护和支持一系列标准的基准测试工具，SPEC CPU2006 基准测试工具是 SPEC 新一代工业化标准 CPU 密集型基准测试套件，提供了分别针对整型计算和浮点型计算的数十个基准测试程序，包括人工智能领域的象棋程序，基于隐马尔科夫模型的蛋白质序列分析

等。SPEC CPU2006 基准测试提供了几种不同的方法来度量系统性能，一种是度量系统完成一个单独任务的速度，另一种方式是度量系统在特定时间内完成任务的数量。SPECjbb2013 是用于评估 java 服务器性能的基准测试工具，可以模拟用户对电子商务网站的各种请求与数据挖掘等操作，该基准测试主要测试 Java 虚拟机，操作系统，以及 CPU 与内存等硬件性能。SysBench 是一个模块化的，跨平台的多线程基准测试工具，可以评估操作系统在密集负载下的各项参数。SysBench 并非一个完全 CPU 密集型的基准测试，它主要评估 CPU 调度性能，内存分配和传递速度等性能。

内存性能测试工具有 LMbench，Memtest86+等[5]。LMbench 是一个简单轻便的，针对 UNIX/POSIX 的微基准测试，主要关注系统的延时性与带宽，其中包括很多简单的基准测试，涉及内存读写复制以及上下文切换等操作。Memtest86+主要用于检测内存故障，是基于 Chris Brady 开发的针对内存缺陷检测工具 Memtest86 改进而来。

网络性能测试工具主要为 Netperf 与 Iperf3，测量网络性能的指标主要有可用性，响应时间与网络吞吐量。Netperf 主要针对基于 TPC 或 UDP 协议的传输，根据应用的不同，可以进行不同模式的网络性能测试，主要包括批量数据传输模式和请求/应答模式，测试项目包括使用 BSD Sockets 的 TPC 和 UDP 连接，使用 DLPI 接口的链路级别的数据传输，Unix Domain Socket 和 SCTP 协议的连接[7]。Netperf 工具以 client/server 方式工作，client 端是 Netperf, 用来向 server 发起网络测试，在 client 与 server 之间，首先建立一个控制连接，传递有关测试配置的信息，以及测试的结果。在控制连接建立并传递了测试配置信息以后，client 与 server 之间会再建立一个测试连接，使用特定的流量模式，以测试网络的性能。

Iperf3 基准测试工具具有跨平台，多线程的特点，主要针对于 TPC，SCTP 与 UDP 传输，可以主动度量网络最大可获得带宽。它支持调整与计时，以及缓冲协议有关的各种各样的参数。Iperf 最初由 NLANR/DAST 开发实现，Iperf3 是由 ESnet/Lawrence Berkeley 国家实验室重新开发而来，Iperf3 没有继承 Iperf 的代码，并且与 Iperf3 不兼容。

在 Linux 操作系统中磁盘 I/O 性能可以通过 iotop 与 DD 命令分析判断，IOzone 是文件系统基准测试工具，该工具可以触发和度量多种文件操作，包括读写，随机读写与文件定向写入等操作，并且其可以被移植到多种机器，在多种操作系统下运行。

端到端的系统性能测试工具主要有 SPECvirt_sc2013，VMmark 2.5 与 TPC-VMS。SPECvirt_sc2013 是 SPEC 联合 AMD，惠普，IBM 等知名软硬件公司发布的工业标准化服务器整合测试工具，其通过收集虚拟化平台在最大负载时的相关数据来度量服务器整

合性能[8]。SPECvirt_sc2013 模拟了常用的虚拟化服务器负载并通过主要控制端来管理各个负载运行以及相关数据收集，其组成服务器包括邮件服务器，数据库服务器，应用服务器，网页服务器和文件服务器，在 SPECvirt_sc2013 中将这个五个虚拟服务器成为一个负载片，分别借助 SPECweb2005 产生对网页服务器和文件服务器的负载请求，SPECjAppServer2004 产生对应用服务器与数据库服务器的工作负载，SPECmail2008 产生对邮件服务器的工作负载。SPECvirt_sc2013 的总分值是基于以下三种组件工作负载的度量值而得到的：

网页服务器：指定数量的会话并发时的每秒的请求数；

邮件服务器：指定数量的用户每秒的操作数之和；

应用程序服务器：在指定的注入率，负载系数和突发曲线水平下的每秒操作数。

计算总分值的方式为取得每一“片”中每种组件的工作负载，将它和该负载级别的理论最大值进行归一化处理。将每一“片”的三种归一化后的吞吐量分数求算术平均值，得到每一“片”的子度量值，再将所有“片”的子度量值全部相加，就得到了总体的性能度量值。

VMmark2.5 基准测试工具与 SPECvirt_sc2013 类似，用来度量服务器整合场景中虚拟化平台的应用性，扩展性以及能源消耗，且只可以在 VMware 产品上运行。VMmark 将 8 台虚拟机作为一个片，通过各个片的应用程序负载分数以及架构操作分数来得到整个虚拟化系统的性能得分，这 8 台虚拟机分别为：邮件服务器，空闲服务器，两个 Olio 虚拟机分别为数据库服务器与 web 服务器以及 4 个 DVD 电子商务服务器[9]。VMmark 中应用程序负载得分与多个应用在测试期间完成的满足服务质量的操作或交易数量有关，架构操作分数是根据在测试期间虚拟机克隆和部署，动态迁移，动态存储以及自动负载平衡四类操作得分标准化后的均值获得，最后将每片的得分相加得到虚拟化系统总得分。

TPC-VMS 是事务处理性能委员会 TPC (Transaction Processing Performance Council）开发的专门针对虚拟化数据库进行性能评估的工具[10]。TPC 是成立于 1988 年的非盈利组织，主要工作是定义交易过程与数据库基准测试工具，并且将客观的且经过检验的 TPC 性能数据传递到工业界。TPC-VMS 主要目标是帮助客户回答一个基本的问题“是否能将已有的数据库系统整合在一个新的服务器中？”。TPC 通过在已有的 TPC-C，TPC-E，TPC-H 和 TPC-DS 基准测试中增加相关方法，实现运行和报告虚拟化数据库性能度量的相关需求，提出了 TPC-VMS，TPC-VMS 展示了三个数据库服务器整合在一个

物理服务器之上的性能计算方法，并且这三个数据库服务器有相同的属性。TPC-VMS 以运行在虚拟化环境中的三个 TPC 基准测试获得的关键度量的最小值作为性能度量值。

1.2.2 基于分析模型的性能评估方法

目前对于虚拟化服务器性能的理论分析主要涉及服务器虚拟化与非虚拟化服务器性能比较分析，服务器整合后性能消耗分析以及服务器虚拟化后其上应用性能分析三类。数学建模方法借助于排队论，人工神经网络以及分层排队模型等数学模型，对虚拟化服务器系统进行抽象分析，并结合服务器负载特征，得到虚拟化服务器系统性能分析预测模型。

服务器虚拟化将运行在传统物理服务器上的应用迁移到逻辑服务器之上，在虚拟化环境中，通过为虚拟机中逻辑物理资源创建实际物理资源的应用程序接口，尽可能使运行其上的应用性能与物理服务器之上的应用性能相同，但是由于虚拟机中需要实现逻辑资源到物理资源转换，并且虚拟机管理器的运行也需要消耗如 CPU，内存等物理资源，因此比较分析虚拟化服务器与非虚拟化服务器的性能成为目前研究的热点。

Ahmadi 等研究者评估了数据中心应用中虚拟化服务器性能与非虚拟化服务器性能。研究者通过在不同的负载下，以“片”为单位比较了系统性能关键参数，证明了虚拟化技术的有效性[11] 。

Huber 等研究者提出了影响虚拟化平台性能的特征模型，并基于提出的虚拟化平台性能特征模型，与获得的 Citrix XenServer5.5 与 VMware ESX 虚拟化平台中具体实验数据，提出了一个简单数学方法的性能预测模型，该模型针对 CPU 与内存的虚拟化消耗，虚拟化操作的可扩展性以及 CPU 资源过量使用时虚拟机行为，这三方面分析了虚拟环境与非虚拟化环境之间的性能关系，以及在不同场景下虚拟化系统本身的操作行为[12]。

Benevenuto 等研究者借助排队网络模型提出了虚拟化系统中应用性能预测模型，研究者将虚拟化环境中与非虚拟化环境中资源繁忙时间的比值作为虚拟化服务器性能减速因子，然后使用减速因子确定虚拟化性能消耗以及应用服务需求等性能度量。在文章中研究者展现了应用从 Linux 系统迁移到 Xen 虚拟化环境中的完整案例[13]。

服务器整合是服务器虚拟化的重要应用场景。虽然服务器整合增加了资源的利用率，提高了应用性能，但是服务器整合也导致了管理的复杂性，共享资源的竞争使用与虚拟机管理器资源调度策略等对服务器整合后虚拟化系统性能有重要影响，因此很多研究者从多个方面对服务器整合环境中系统性能消耗进行了研究。

Kraft 等研究者提出了一个轨迹驱动的方法来预测在服务器整合环境中由于存储设

备的竞争导致的磁盘响应时间的降低，研究者将系统中存储资源请求访问过程抽象为排队网络模型，然后记录测试中应用的轨迹，最后使用这些轨迹作为参数来模拟整合环境中性能模型，模型中的参数仅依赖于在虚拟机中得到的度量以及在虚拟机管理器操作时收集的信息，如公平调度信息，请求的拆分与合并信息[14] 等。

Apparao 等研究者提出了一个针对多核心(Chip multiprocessors, CMP)架构的服务器虚拟化性能模型，虚拟化 CMP 服务器是指可以同时运行多种负载的服务器，该研究将虚拟化 CMP 服务器性能影响因素分为三方面, 分别为整合状态中内核干扰，整合状态中内存干扰，与虚拟化本身消耗，并建立系统整体性能模型，然后通过对每一方面具体影响因子分析，建立每个影响因素的子性能模型，最后通过 Xen 虚拟化平台，借助 vConsolidate 负载触发工具搭建实验环境，分析该性能模型的准确性[15]。

Appel 等研究者分析运行相同负载的虚拟化服务器整合环境中虚拟机资源的行为和性能，即 CPU，内存以及磁盘 I/O 输出的行为与性能。该研究通过设置同时运行的虚拟机的数量来评估资源共享的公平性，虚拟化的性能消耗以及虚拟机之间的相互影响[16]。

Brosig 等研究者描述了 Xen 虚拟化环境中性能影响参数，并采用排队网络模型来展现服务器整合环境中资源的使用，基于此提出虚拟化性能消耗计算方法，建立性能预测模型，并在不同实验场景对模型进行评价。该研究只关注于 CPU 资源使用，因为研究者认为应用在虚拟化环境中运行时，与传统服务器环境相比，不会产生更多的网络包传输或触发更多的磁盘请求，因此其它资源，如磁盘与网络 I/O 比率，不会对虚拟化层产生直接影响，虚拟化环境中 CPU 调度分配是重要影响因素[17]。

Hui 等研究者分析了服务器整合环境中面临的挑战，借助 SPECvirt_sc2010 基准测试在 Xen 虚拟化平台触发负载并收集相关数据，然后基于收集的数据详细分析了服务器整合面临的挑战，如可扩展性，虚拟机管理器性能消耗等，最后根据面临的问题，提出了性能优化方法[18]。

IT 系统的最终目的是为用户提供高质量的服务，因此在应用层对虚拟化服务器性能进行评估是重要的。在 IT 环境中，应用性能常常与服务等级协议(Service-Level Agreement, SLA)相联系，在为用户提供服务期间，如果 SLA 被违反，会造成用户满意度下降，因此虚拟化系统中应用的性能也是目前研究的重点。

MENASCE 等研究者分析了具有两台虚拟机的虚拟化环境中资源虚拟化实现方式，以及不同类型指令在系统中的处理方式，最后基于排队网络模型提出了虚拟化系统的服务质量(Quality of Service, QoS)相关的性能度量项计算方法，并通过实验说明了计算方法

的实际应用[19]。

Sajib 等研究者识别了典型应用在虚拟化环境中常用的性能度量指标及其影响因素，并研究了几个常规建模方法对虚拟化环境中应用建模的有效性，最后研究者基于人工神经网络提出了迭代的建模方法，在 Xen 虚拟化环境进行了原型实现，同时评估了模型的精确性[20]。

Keyvan 等研究者介绍了系统分析方法来评估虚拟化系统中的 QoS 参数，研究者通过分析虚拟化系统中虚拟机，虚拟机管理器以及物理资源层与虚拟机的交互来模型化虚拟化系统中的应用，提出 QoS 参数的计算方法，并通过实验数据验证模型的正确性[21]。同时 Keyvan 等人深度分析了 Xen 虚拟化环境下 web 服务器性能度量影响因素，然后借助排队网络模型分别建立了 CPU 密集型负载与 CPU 和 I/O 密集型负载时的性能评估模型，并且研究者通过实验数据对模型进行了分析，说明了各影响因素对性能的定量影响，最后研究者使用 Wikibench 基准测试模拟 wikipedia 负载，使用提出的性能评估模型对应用性能进行了评估[22]。

1.2.3 对当前研究现状的评价

当虚拟服务器作为服务端为终端用户提供服务时，对其上应用性能进行评估，判断其是否能够满足用户要求是重要的。当应用性能不能满足用户需求时，如何根据获得的性能数据发现性能瓶颈，也是应用性能评估时必须解决的问题。由 1.2.1 节中对基于模拟的性能评估方法的分析可以得到，基于组件的性能基准测试用于评估虚拟服务器主要组件性能，端到端性能基准测试用于评估虚拟服务器端到端的应用性能。如何在获得应用性能的同时得到虚拟服务器组件性能，帮助服务提供商进行性能分析，是目前面临的问题。

由 1.2.2 节可以得出，目前对于虚拟化服务器性能评估模型的研究多集中于对虚拟化服务器不同使用场景造成的性能消耗研究，并且对于虚拟化系统中应用性能的研究多基于开源的虚拟化平台 Xen。KVM 在 2007 年被合并入 Linux 内核，2011 年 Red Hat 公司在其企业版的 Linux 系统中已经移除了对开源平台 Xen 的支持，并联合 IBM 等大型硬件供应商成立开源联盟以支持 KVM 发展。由于 KVM 也是 OpenStack 云部署中流行的管理程序，这也为 KVM 的快速发展提供了机会。目前 KVM 使用量增长迅速，因此对基于 KVM 的虚拟化服务器进行性能评估是十分必要的。

1.3 研究内容和目标

本文的研究目的是对 KVM 虚拟服务器中应用性能进行评估，由虚拟服务器中应用性能评估虚拟服务器性能优劣，简化虚拟化服务器评估步骤与成本。同时对如何基于虚拟服务器性能评估模型，确定不同应用场景中虚拟化服务器的配置进行了实例说明。

基于上述研究目标，本文要解决四个问题：如何选择虚拟化服务器性能度量指标；选择何种方法对基于 KVM 的虚拟化服务器建模；如何设计实验，验证提出模型的正确性；如何说明性能评估模型的可用性。针对需要解决的问题，本文主要包括以下研究内容：

首先，本文研究了虚拟化服务器性能评估维度，确定根据虚拟服务器中应用的性能对服务器性能进行评估。基于对应用性能评估指标的研究，选择 SLA 中针对服务提供商的约束指标作为虚拟化服务器性能评估指标。

其次，本文分析了 KVM 中虚拟化服务器性能影响因素，根据性能影响因素，结合 KVM 中虚拟化实现原理，建立了基于排队网络的性能评估模型。借助 Linux 操作系统开源特点，说明了在 Linux 环境中采用性能评估模型计算 KVM 虚拟化服务器性能度量指标的具体方法。

再次，本文根据虚拟化服务器性能影响因素，选择 CPU 调度与虚拟机数量作为实验变量，进行对比实验，验证性能评估模型的正确性。

最后，本文调研了目前 KVM 虚拟化服务器配置方法以及用户层面应用性能约束条件，结合多准则决策中的层次分析法与权重加和法，研究提出了基于性能评估模型的虚拟化服务器配置决策方法，证明了性能评估模型在应用领域中的可用性。

1.4 论文组织

全文由六部分组成：

第一章绪论：阐述了研究工作的背景意义，从基于模拟的性能评估方法与基于分析模型的性能评估方法两个方面，分析了虚拟化服务器性能评估的国内外研究现状，介绍了论文的研究内容与研究目标以及论文的组织结构。

第二章虚拟化服务器性能评估相关技术：首先论述了服务器虚拟化实现类型与原理，以及虚拟化服务器性能的不同分类。基于不同类型的性能评估维度，概述了虚拟化服务器性能评估度量指标的选择，然后从基准测试工具与理论研究两方面分析了虚拟化服务器性能评估方法的实际应用以及研究发展。最后论述了 KVM 虚拟化技术的实现原理，

相关技术与工具发展，以及 KVM 虚拟化服务器中性能评估的相关研究。

第三章建立基于 KVM 虚拟化服务器的性能评估模型：首先分析了基于 KVM 的虚拟化服务器性能度量指标，然后详细研究了 KVM 虚拟化环境中性能度量指标的影响因素，根据 KVM 中资源虚拟实现方式，建立性能评估模型建模，并详细说明了性能评估模型中性能度量指标的计算方法，最后说明了本文提出的性能评估模型适用的范围及其局限性。

第四章性能评估模型验证：首先根据 KVM 虚拟服务器中性能影响因素，确定模型正确性验证的实验设计方案，以及实验配置，然后对实验实施过程进行了详细说明，最后对获得的实验数据进行了分析。通过对实验结果不同维度的对比分析，验证了性能影响因素选择的合适性与性能度量模型的正确性，同时通过与基准测试工具得到的性能数据的对比，进一步验证了模型的精确性。

第五章性能评估模型的应用：首先介绍了目前常用的虚拟化服务器配置原则，从用户角度，分析了虚拟服务器中应用性能的约束条件，然后基于层次分析法与权重求和法，结合性能评估模型，提出了虚拟化服务器配置决策方法，并对其在实际环境中的使用方法进行了说明，最后通过在 I/O 密集型操作环境中的实验实例说明了决策分析方法的具体使用步骤。

总结与展望：总结了本文的工作，并基于本文的研究现状说明了下一步工作方向。

第二章 虚拟化服务器性能评估相关技术

2.1 服务器虚拟化技术概述

虚拟化技术可以使多个虚拟机运行在一个物理服务器之上，并且各个虚拟机之间相互独立。Goldberg 把虚拟机描述为现存真实机器的副本或者硬件到软件的复制品[3]。在 20 世纪 60 年代早期虚拟机概念被提出，IBM 工程师与 MIT 编程人员一起开发了分时操作系统，可以使多个用户同时进行操作，实现了用户对大型机的共享，随后虚拟化技术在 IBM OS/360 model 67, VM/370 以及 OS/390 系统中进一步发展。20 世纪 80 年代与 90 年代早期分布式计算进入数据中心领域，具有专用功能的独立服务器渐渐取代集中式计算与虚拟化技术成为研究热点，目前由于专用服务器集群资源利用率过低，能源消耗严重等问题，服务器整合成为研究重点，因此虚拟化技术再次成为人们关注的重点。

在虚拟化环境中，虚拟机管理器（Virtual Machine Monitor, VMM）是系统资源的主要管理者，是虚拟化平台的核心。VMM 对硬件资源进行抽象，以系统调用的方式呈现给运行之上的客户操作系统。根据 VMM 在虚拟化系统中的不同位置，可以将虚拟化实现方式分为两类，如图 1 所示，Type-I 是 VMM 直接运行在物理硬件之上， 如图 2 所示，Type-II 虚拟化是 VMM 运行在一个主机操作系统之上，被称为宿主模式。

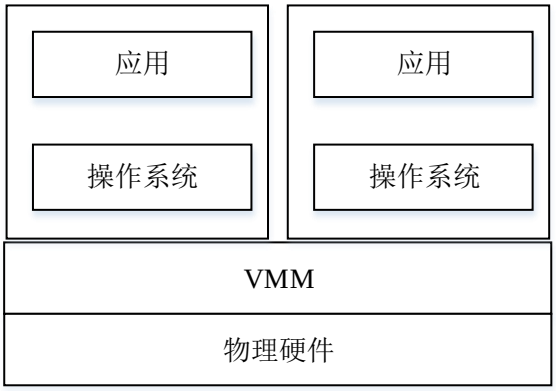

图 1　虚拟化类型 I

在宿主模式中 VMM 作为应用程序，运行在宿主操作系统之上，这种方法对物理服务器配置没有特殊要求。在类型 I 中，VMM 直接安装在物理硬件之上，因此 VMM 可以直接获取硬件资源，而不必借助主机操作系统进行物理资源调度，因此一般类型 I 的虚拟化环境比宿主模式虚拟化环境有更好的可扩展性，健壮性以及更好的应用性能。因此在实际企业应用中常采用类型 I 建立虚拟化环境。

目前常用的 X86 体系架构为操作系统与应用程序提供了 4 种优先级，即 Ring 0,1,2,3，来进行系统资源的访问控制，并且优先级从 0 到 3 逐级递减。应用程序一般运行在特权

等级 3，因为操作系统需要直接获取内存等其它硬件资源，因此运行在特权等级 0。X86 体系虚拟化需要在操作系统之外增加一个虚拟层来创建和管理虚拟机，但是由于 X86 体系中的操作系统被设计为直接运行在物理硬件之上，其认为操作系统拥有完整的系统硬件控制权，因此 X86 体系中一些敏感指令不在 Ring0 层执行时，有不同的语义。所以要实现 X86 体系的虚拟化，必须解决这一问题。根据解决这一问题采用的方法不同，虚拟化类型 I 又可以分为全虚拟化与半虚拟化，其中全虚拟化可以通过二进制翻译与硬件支持两种方式实现[23]。

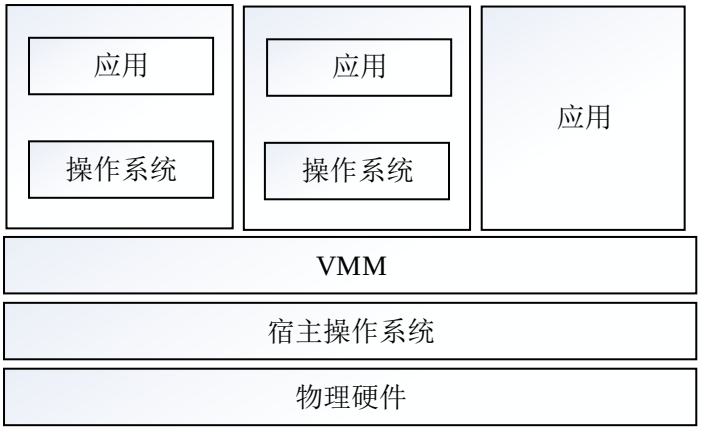

图 2　虚拟化类型 II

全虚拟化是指客户机操作系统无需修改直接可以运行在虚拟机之中。借助二进制翻译实现的全虚拟化，即基于软件的全虚拟化，其借助翻译内核代码将虚拟机中操作系统的指令翻译成 X86 的指令子集，并将敏感指令替换为 VMM 陷入指令，VMM 为每个虚拟机提供了物理系统的所有服务，如虚拟化的 BIOS 与虚拟化内存管理等。因此在该实现方式下，客户机操作系统不会“意识到”其正运行在虚拟平台之上。为提高虚拟化效率，解决 X86 体系存在的缺陷，Intel 与 AMD 分别推出了支持虚拟化的 CPU，VMM 可以利用硬件虚拟化技术，实现基于硬件的全虚拟化。Intel 的 VT-X 与 AMD 的虚拟化技术为 CPU 增加了一种新的执行模式，使得 VMM 可以在 Ring0 的新模式下运行。在该模式下，客户系统调用的特权与敏感系统指令可以自动陷入管理程序，而不需额外处理，目前常见的全虚拟化解决方案有 VMware ESXi 与 Citrix XenServer 等。

半虚拟化需要修改操作系统的内核将无法虚拟化的指令替换为超级调用，使其可以直接与 VMM 进行交互，VMM 也为其它重要的内核操作，如内存管理，终端处理等，提供了超级调用接口。半虚拟化与全虚拟化的不同之处在于需要修改运行在虚拟机之上的操作系统，半虚拟化不支持不可修改的操作系统，如微软 Windows 系列操作系统。因此其兼容性较差，并且半虚拟化系统性能严重依赖于系统负载。Xen 是目前常见的半

虚拟化解决方案。

在众多虚拟化解决方案中，研究者对 KVM 虚拟化技术的分类存在争议。KVM 是 Linux 内核的一部分，用户可以从命令行启动 KVM，这使得 KVM 类似于运行在操作系统之上的宿主 VMM，并且 Linux 操作系统中其他应用程序会与虚拟机进行资源竞争，因此从这方面来看，KVM 可以视为宿主型虚拟化解决方案，即属于虚拟化类型 II。但是 KVM 依赖于硬件虚拟化技术，借助 Linux 操作系统的进程调度策略与内存管理方法等实现虚拟机管理，将 Linux 操作系统转化为 VMM，因此可以认为在 KVM 中，VMM 直接运行在物理硬件之上，即属于虚拟化类型 I。由于 KVM 中不需要对虚拟机操作系统进行修改，因此 KVM 虚拟化可视为硬件支持的全虚拟化。

2.2 虚拟化服务器性能评估指标体系

在对虚拟化服务器性能进行评估之前必须要明确虚拟化服务器性能评估指标。在虚拟化系统中，对虚拟化服务器性能关注点不同，对应的性能评估指标也不同。

2.2.1 性能评估度量指标

虚拟化系统性能的优劣是是否采用虚拟化的重要约束条件，对于已经建立的虚拟化系统其性能评估主要包含两个不同的维度：运行在虚拟化服务器之上的应用必须与运行在本机的相同应用有几乎相同的性能；运行在同一个物理主机上的多个虚拟机必须有很好的扩展性，并且可以有效的共享资源，即保证虚拟化系统有良好的可扩展性与隔离性[24]。可扩展性是指虚拟化系统能够支持的虚拟机的数量以及可运行应用的数量，迁移性能是指将虚拟机从一个物理机迁移到另一台物理机时，不同迁移策略对应的性能，隔离性是指位于同一台物理机的虚拟机之间的相互影响程度[25]，而应用性能是服务供应商关注重点，其直接影响用户满意度。

基于以上服务器虚拟化性能评估的不同维度，可以分析得到虚拟化性能度量指标。在应用性能评估维度，SLA 是服务提供商与用户之间的一个正式协定，最初用于对网络供应商进行约束，现在被广泛用于各种 IT 服务管理。在 SLA 协议中，QoS 参数明确了服务器供应商应该提供的服务质量。通常 QoS 有三个属性来度量输出服务性能，分别为：时效性，精确性与准确性。时效性度量提供服务消耗的时间，精确性度量产生的输出的数量，准确性度量输出的正确性[26]。基于 QoS 的具体度量项依赖于服务的应用场景以及用户关注的重点确定。

吞吐量是运行在每个客户虚拟机之上的单个负载以及整个虚拟化系统性能的关键

度量。虚拟化的目标是使单个或多个并发的负载具有足够的吞吐量来满足服务水平需求。延迟是也虚拟化系统中应用的重要度量指标，运行在每个虚拟机上的应用必须满足 QoS 对运行在物理机上的相同应用的相同约束。资源的使用率是虚拟化系统可扩展性能的关键度量指标，虚拟化系统中资源使用率主要包括 CPU 使用率与内存使用率，并且吞吐量与延迟也是系统可扩展性的重要指标，其中延迟是指系统对请求的响应时间。根据对具体度量项分析可以得出，吞吐量与延时是虚拟服务器性能评估的常用度量指标。

2.2.2 性能影响因素

虚拟化实现方式对虚拟化服务器性能有重要影响，一般而言全虚拟化性能要好于半虚拟化[27]。全虚拟化提供了好的虚拟机隔离性与安全性，由于半虚拟化需要修改客户机操作系统导致其可移植性较差，但是其通常有较低的资源使用率，半虚拟化性能依赖于运行之上的负载类型。另一方面，因为硬件的执行速度要快于软件，所以基于硬件的全虚拟化性能要好于基于软件的全虚拟化性能。Huber 等研究者调研了目前常见的虚拟化实现方案，包括基于宿主模型的 VirtualBox 与 VMware Server，基于类型 I 的全虚拟化或半虚拟化技术 Xen，KVM，VMware ESXi，以及基于容器的虚拟机技术 OpenVZ 与 Linux-VServer [28]。通过总结分析以上虚拟化平台性能影响因素，建立了层次化的特征模型，该特征模型将虚拟化平台性能影响因素划分为虚拟化类型，资源配置管理与工作负载配置三大类，如图 3 所示。

为提高虚拟化系统的性能，研究者进行了大量分析，深入研究了具体虚拟化平台的性能影响因素，并提出了优化方案。目前多数研究集中于开源虚拟化平台中共享资源的竞争使用以及虚拟机之间的资源调度，即虚拟化平台资源配置管理方面的研究。

Ding 等研究者研究了 KVM 中宿主操作系统，即 VMM 的调度策略，VMM 的调度策略决定了整个系统的性能。在 KVM 中，Linux 将每个虚拟机视为普通的用户进程，采用抢占优先级调度策略，继承了 Linux 调度特征的所有优势，但是虚拟机进程需要更高的优先级来运行其上的应用，否则当虚拟机正在执行系统特权指令时，如果优先级高于虚拟机进程，但是低于虚拟机中正在运行的应用的进程到来，Linux 将会让虚拟机放弃资源，执行新进程，因此将会造成更多的虚拟机上下文切换，导致性能消耗。研究者建议在 Linux 中增加一个新的高优先级队列，使 VM 进程高于一般进程，减少虚拟机上下文切换[29]。

Hui 等研究者在 Xen 虚拟化平台创建了典型的服务器整合场景，并借助 SPECvirt_sc2013 进行了性能测试。通过对实验数据分析发现，随着虚拟机数量的增加，

虚拟机和虚拟机管理程序占用的内存逐渐增多，导致硬件资源如分层高速缓存，不能有效地处理请求，降低指令了执行速度，因此频繁上下文切换造成的内存占用量过大是虚拟化系统的主要瓶颈。研究者提出从动态分配机制与上下文切换比例控制两个方面研究了 Xen 虚拟化平台性能提升[17]。

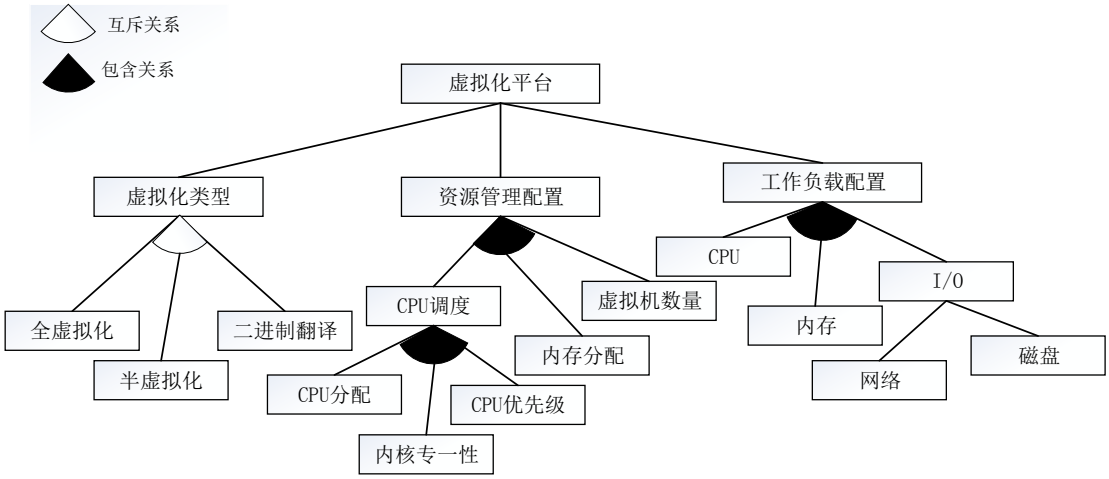

图 3　虚拟化服务器性能影响因素特征模型

Ravi 等研究者借助 vConsolidate 在多芯片服务器上模拟了服务器整合场景，分析研究了共享资源竞争，如 CPU 内核，共享的内存空间与带宽等，对虚拟机性能的影响，并将共享资源对性能的影响模型化。该性能模型包括虚拟化消耗，内核竞争消耗以及共享内存竞争消耗三部分，用户借助该模型可以估计在服务器整合环境中虚拟机潜在的性能损失。另一方面，基于以上的研究分析，研究者提出了一种虚拟化平台体系架构来帮助进行共享资源管理，提高虚拟化服务器的性能隔离[30]。

2.3 虚拟化服务器性能评估方法

目前虚拟化服务器性能评估方法可以分为两大类，一类是以著名标准化组织 SPEC 开发的 SPECvirt_sc2013 为代表的端到端虚拟化服务器性能评估模拟化方法，一类是以分析模型为基础的虚拟化服务器性能评估方法，其中又可以分为理论分析建模方法和实验评估建模方法[23]。理论分析建模方法借助传统数学模型，如排队网络模型，分层排队网络模型等，对虚拟化系统进行抽象建模，然后结合虚拟化系统负载特性对虚拟化服务器进行性能评估。实验评估建模方法借助神经网络，线性回归等建模方法，通过实验数据分析，确定虚拟化性能关键影响参数以及参数分析方法，然后结合虚拟化系统具体特征建立性能评估模型。

2.3.1 端到端的性能评估方法

基于工业调查报告，SPEC 选择了服务器整合场景中常见的应用，提出了评估虚拟化平台管理服务器整合场景能力的基准测试工具 SPECvirt_sc2013。SPECvirt_sc2013 基准套件通过对 SPEC 已有基准负载进行修改，使其符合虚拟化环境，建立了以“片”为单位的基准测试工具。每个“片”包括一个网页服务器，一个批处理服务器，一个基础结构服务器，一个应用程序服务器，一个邮件服务器以及一个数据库服务器，其中基础结构服务器为网页服务器提供后端支持，数据库服务器是应用服务器的后端支持。在 SPECvrit_sc2013 是自动化测试工具，通过在主控制端与被测虚拟化服务器中启动相关命令，客户端发起到虚拟化服务器的请求，虚拟化服务器记录相关处理数据。实验结束后主控制端收集各虚拟化服务器中的数据得到性能结果，图 4 显示了 SPECvirt_sc2013 体系结构。SPECvirt_sc2013 基准测试工具是从应用层面对虚拟化服务器进行性能测试，展现了当虚拟化服务器满足 QoS 时，虚拟化系统的最大吞吐量。

SPECvirt_sc2013 中 web 服务器 QoS 参数为 Time_Good 与 Time_Tolerable，即 95%的页面请求必须在 Time_Good 时间内完成，99%的页面请求必须在 Time_Tolerable 时间内完成。对于较大的静态文件，SPECvirt_sc2013 明确了字节率来检验文件下载性能。SPECvirt_sc2013 中邮件服务器 QoS 要求为：95%的交易必须在 5 秒内完成，对于每一种 IMAP 操作不能有多于 1.5%的失败率，所有操作类型的失败数量必须少于所有操作类型的 1%。SPECvirt_sc2013 中批处理服务器 QoS 约束为 99.5%的轮询请求必须在 1 秒内响应。SPECvirt_sc2013 中应用服务器 QoS 要求至少每种业务类型操作量的 90%在 2 秒内完成，并且每种业务交易类型的平均响应时间必须不比 90%响应时间多于 0.1 秒[7]。因此我们可以看出，SPECvirt_sc2013 关注服务器整合环境中应用的响应时间与吞吐量，即 SPECvirt_sc2013 选择虚拟化服务器中应用的响应时间与吞吐量作为性能度量度量指标，并对其度量值范围进行了详细划分。当系统中虚拟服务器不能满足 QoS 中时间参数时，认为其已经达到最大负载，将此时系统中虚拟机数量作为评估虚拟化系统度量指标之一。

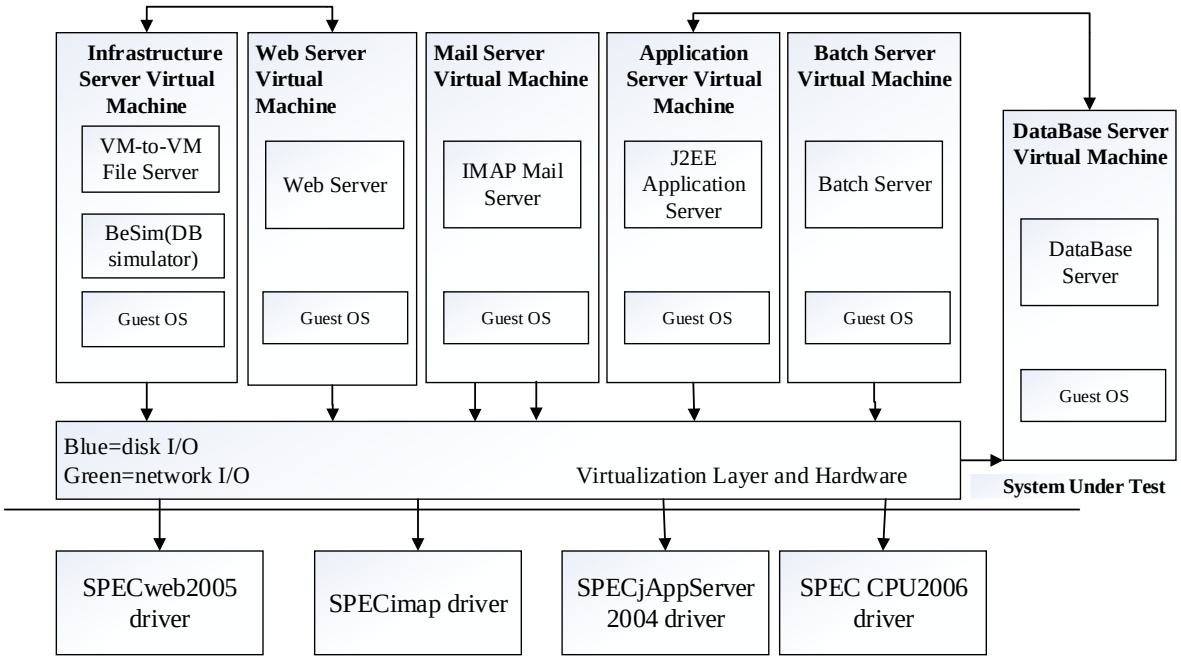

图 4　SPECvirt_sc2013 架构

SPECvirt_sc2013 由两部分组成，分别为被测系统与客户端，其中客户端又分为主要控制器与客户端管理者。在进行测试时，首先通过指令为四个主要客户端，即被测系统开启对应的客户端管理进程，然后客户端管理者为四种负载开启对应的客户请求进程。当所有的客户端管理进程开始监听端口，使用命令启动主要控制器开启基准测试。主要控制器将会通知客户端管理者开启虚拟化服务器对应的客户端，开始产生用户请求，等待一定时间后，主要控制器通知主要客户端的管理程序开启虚拟化服务器，此时实验正式开始。当用户请求结束时，主要客户端向主要控制器报告测试运行结束，主要控制器结束虚拟化服务器以及客户端运行，最后在客户端启动命令收集实验结果。在 SPECvirt_sc2013 客户端中，通过度量请求发出时间与请求响应时间差值，以及测试过程中实际获得请求结果来度量虚拟化服务器性能。另一方面，SPECvirt_sc2013 也可以度量虚拟服务器的能源消耗，度量项分为 SPECvirt_sc2013_PPW(performance with SUT power)与 SPECvirt_sc2013_ServerPPW(performance with Server-only power)两种，其中 SUT(System Under Test)是指被测试系统。当虚拟系统负载达到最大时，该工具会得出每瓦特性能度量值。SPECvrit_sc2013 结构如图 5 所示，其中存储组件用于备份虚拟服务器数据，防止实验故障导致数据丢失。

通过对 SPECvirt_sc2013 中使用的度量指标与其结构分析可以得到，其以 QoS 参数中常用应用性能度量指标为约束，来评估虚拟系统能够负载的最大虚拟机数量。为了模拟了真实的服务器整合环境，SPECvirt_sc2013 整合了多个虚拟机，并为每个虚拟机加

载了真实应用环境中使用的应用数据。测试过程中客户端模拟用户访问模式发出访问请求，其测试过程一般周期较长。测试过程结束后，主控制器在客户端收集度量项，判断其 QoS 参数是否超出某个范围，因此其度量值精确度较高，近似等同于真实环境中实际的用户访问体验。该特点是基于模拟的性能评估方法与分析建模性能评估方法的显著区别。

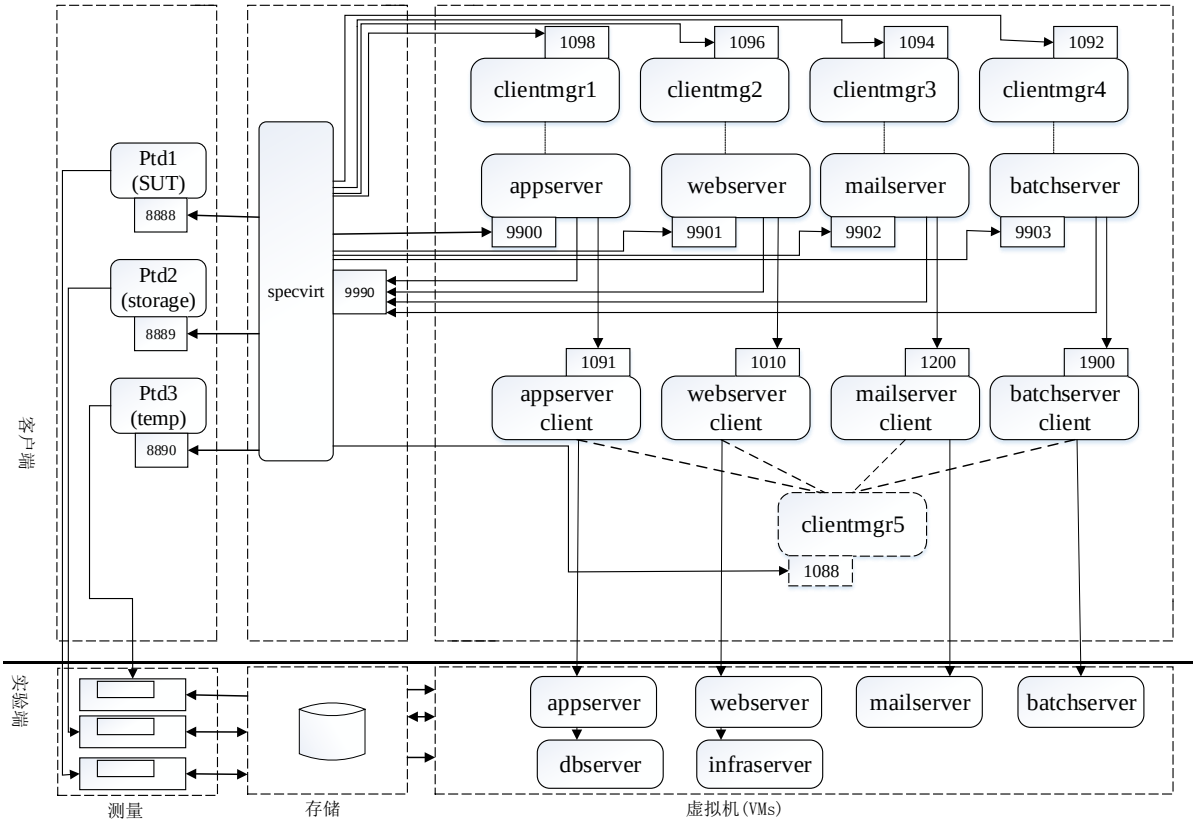

图 5　SPECvrit_sc2013 实现原理

2.3.2 基于排队网络模型的性能评估方法

通过对不同虚拟化平台的性能相关研究进行分析总结可以得出，虚拟化平台性能主要与资源虚拟化实现方式，以及 VMM 进程调度有关，即虚拟化平台性能主要与虚拟化系统中资源使用有关。由于排队网络模型主要用于展现和分析系统中资源的使用，因此目前研究者在对虚拟服务器性能进行理论分析时通常采用排队网络模型进行建模。

排队网络模型分为服务中心，客户和网络拓扑三部分，排队网络即为互联的服务中心。在排队网络模型中硬件与软件资源被视为服务中心，流过排队网络的工作流即为客户。一个客户中可能有多个工作负载类，一个负载类中包含相同特征的请求，根据类中请求的数量是固定的还是可变的，可以将模型分为封闭型排队网络模型和开放型排队网络模型。

在开放型排队网络模型中，一个负载类可以通过负载密度和负载服务需求刻画，负

载服务需求是指一个负载类在给定的系统资源中花费的总时间，负载密度即为负载类中请求的到达率。根据模型中负载的种类，模型又分为多负载类模型与单负载类模型。

单负载类开放型排队网络模型如图 6 所示，在单负载类模型中，假设有 K 个服务中心，负载到达率为 λ，在服务中心 m 中的服务需求为D_m，V_m为单个负载经过服务中心 m 的次数，则该排队网络模型的输出如公式(2.1)至(2.4)所示。

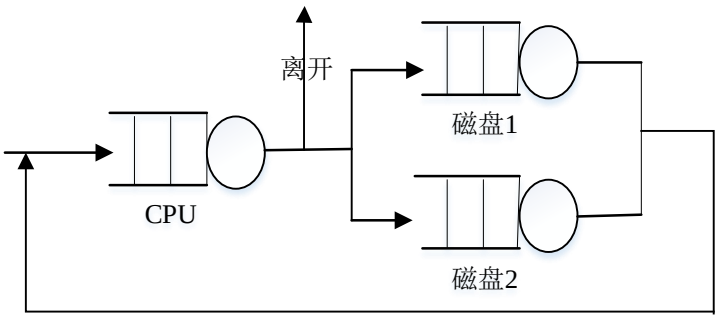

图 6　单负载类开放型排队网络模型

1) 工作负载在服务中心 m 的吞吐量，见公式(2.1)。

$$X_m = \lambda V_m \tag{2.1}$$

2) 工作负载在服务中心 m 的服务需求，见公式(2.2)。

$$D_m = \frac{U_m}{X_m} \tag{2.2}$$

其中U_m是服务中心 m 的资源使用率。

3) 工作负载在服务中心 m 中的响应时间, 见公式(2.3)。

$$R_m(\lambda) = \begin{cases} D_m & IS \\ D_m[1 + A_m(\lambda)] & FCFS, PS, LCFSPR \end{cases} \tag{2.3}$$

其中A_m为负载类中请求达到时，服务中心 m 中排队的请求数，所以A_m与表示服务中心 m 中队列长度Q_m相等，因为$Q_m(\lambda) = \lambda R_m(\lambda)$，因此可以得到在 FCFS，PS，LCFSPR 调度情况下，服务中心 m 的响应时间可以转换为使用率与服务需求的比值，见公式(2.4)。

$$R_m(\lambda) = \frac{D_m}{1-\lambda D_m} = \frac{D_m}{1-U_m} \tag{2.4}$$

多负载类开放型排队网络模型如图 7 所示。在多负载类开放型排队网络模型中，存在多种负载类型，每种负载类型有各自的请求到达率与服务需求，假设模型中有 K 个服务器中心与 C 种负载类型，负载类型 c 的请求到达率为λ_c，$\lambda=(\lambda_1, \lambda_2, \cdots, \lambda_c)$，$D_{c,m}$为负载类 c 在服务中心 m 的服务需求，$V_{c,m}$为负载类 c 对服务器中心 m 的访问次数，则该排队网络模型输出如公式(2.5)至(2.10)所示。

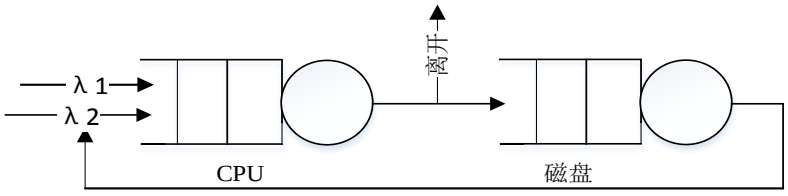

图 7　多负载类开放型排队网络

1) 负载类 c 在服务中心 m 的吞吐量，见公式(2.5)。

$$X_{c,m}(\lambda) = \lambda_c V_{c,m} \tag{2.5}$$

2) 负载类 c 在服务中心 m 的资源使用率，见公式(2.6)。

$$\rho_{c,m}(\lambda) = \lambda_c D_{c,m} \tag{2.6}$$

3) 服务中心 m 的资源总使用率，见公式(2.7)。

$$\rho_m = \sum_{c=1}^{C} \lambda_c D_{c,m} \tag{2.7}$$

4) 负载类 c 在服务中心 m 的响应时间，见公式(2.9)。

$$R_{c,m}(\lambda) = \begin{cases} D_{c,m} & IS \\ D_{c,m}\left[1 + A_{c,m}(\lambda)\right] & FCFS, PS, LCFSPR \end{cases} \tag{2.8}$$

与单负载类排队网络模型类似，$A_{c,m}(\lambda)$表示当负载类 c 的请求到达服务中心 m 时，服务中心中已有的队列长度，假设$Q_m(\lambda)$表示服务器中心 m 中已有的队列长度，则如公式(2.9)所示。

$$\mathrm{A_{c,m}}(\lambda) = \mathrm{Q_m}(\lambda) = \sum_{\mathrm{c=1}}^{\mathrm{C}} \mathrm{Q_{c,m}}(\lambda) \tag{2.9}$$

因此可以得到，当服务中心中调用规则为处理器共享(PS)，先到先服务(FCFS)与后到先服务-抢占后重新开始，以及无限服务时，响应时间可以表示为公式(2.10)

$$R_{c,m}(\lambda) = \frac{D_{c,m}}{1-\rho_m(\lambda)} \tag{2.10}$$

在封闭型排队网络模型中请求数量是固定的，即客户不断向服务器中心提供负载请求，以保证模型中请求数量不变，则封闭型排队网络模型中负载通过请求数量与服务器需求刻画。与开放型排队网络模型类似，封闭型排队网络模型也可以分为单负载类模型与多负载类模型，封闭型单负载类排队网络模型如图 8 所示。

在单负载类模型中，假设有 M 个服务器中心，系统中请求数量为 N，在服务器中心 m 中的服务需求为D_m，该模型的输出如公式(2.11)至(2.14)所示。

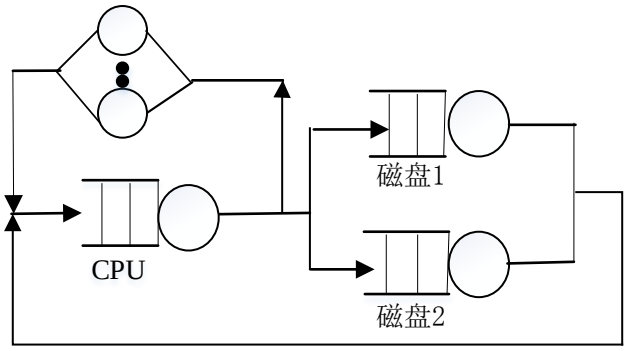

图 8　单负载类封闭型排队网络模型

1) 模型中虚拟服务器吞吐量，见公式(2.11)。

$$X(N)=\frac{N}{\sum_{m=1}^{M}R_m(N)} \tag{2.11}$$

其中$R_m(N)$为 N 个请求在服务中心 m 的响应时间；

2) 模型中服务器中心 m 的队列长度，见公式(2.12)。

$$Q_m(N)=X(N)R_m(N) \tag{2.12}$$

3) 模型中服务器中心 m 的响应时间，见公式(2.13)。

$$R_m(N)=\begin{cases} D_m & IS \\ D_m[1+A_m(N)] & FCFS,PS,LCFSPR \end{cases} \tag{2.13}$$

其中A_m(N)为请求到达服务器中心 m 时，请求队列中存在的请求数目，在封闭型模型中，当 N>0 时，$A_m(N)=Q_m(N-1)$，当 N=0 时，$Q_m(N)=0$，因此当 N>0 时，响应时间如公式(2.14)所示。

$$R_m(N)=\begin{cases} D_m & IS \\ D_m[1+Q_m(N-1)] & FCFS,PS,LCFSPR \end{cases} \tag{2.14}$$

通过迭代算法，MVA (Mean Value Analysis)，可以计算封闭型模型输出，算法[31]如下所示：

输入：

每个服务中心的服务需求 D，D={D_1，D_2,⋯, D_m}

模型中总请求数 N

每个服务中心队列长度 Q，Q={Q_1，Q_2,⋯, Q_m}

输出：该排队网络模型的性能度量

for m=0, m<M, m++; do Q_m=0

for j=1, j<N, j++; do

for m=1, m<M, m++; do $R_m=\begin{cases} D_m & IS \\ D_m[1+Q_m] & FCFS,PS,LCFSPR \end{cases}$

$$X = \frac{N}{\sum_{m=1}^{M} R_m}$$

for m=1, m<M, m++; do $Q_m = XR_m$

end for

在多负载类封闭型排队网络模型中，假设有 M 个服务中心与 C 种负载请求，N_c表示负载类 c 的请求数，$\underline{n}=(n_1, n_2, \cdots, n_c)$，与单负载类封闭型模型输出类似，可以使用多负载平均值算法求解模型，算法如下：

输入：

每种负载的请求数n_c， $\underline{n}=(n_1, n_2, \cdots, n_c)$

每种负载在每个服务中心的服务器需求$D_{c,m}$

每个服务中心每种负载的队列长度$A_{c,m}$

输出：

for m=1, m<M, m++; do $Q_m(\underline{0})=0$

for j=1, j<$\sum_{c=1}^{C} n_c$, j++; do

for j∈ $\underline{n}=(n_1, n_2, \cdots, n_c)$ do

for c=1, c<C, c++; do

for m=1, m<M, m++; do

$$R_{c,m}(\underline{n})=\begin{cases} D_{c,m} & IS \\ D_{c,m}\left[1 + A_{c,m}(\underline{n})\right] & FCFS, PS, LCFSPR \end{cases}$$

for c=1, c<C, c++; do $X_c=\frac{n_c}{\sum_{m=1}^{M} R_{c,m}(n)}$

for m=1, m<M, m++; do

$$Q_m(\underline{n}) = \sum_{c=1}^{C} X_c R_{c,m}$$

end for

end for

实验评估建模方法通过在虚拟化平台上进行多种类型不同基准测试实验，收集实验数据，然后对实验数据进行分析，识别确定影响因素与性能度量关系，建立训练集。借助训练集对模型进行训练提高输出精确度，最后建立性能模型，人工神经网络是常见的建模方法。因此训练样本的数量与质量对模型好坏有重要影响，不同类型的数据集有不同的训练过程。

2.4 KVM 虚拟化概述

最初 KVM 由位于以色列的 Qumranet 公司开发提出，2008 年 Red Hat 收购 Qumranet，KVM 开始在 Linux 中广泛应用，自从 RHEL5.4 版本起，KVM 成为了 Linux 默认的 VMM[32]。KVM 可以看作是借助硬件支持的全虚拟化解决方案。

2.4.1 KVM 实现机制

KVM 是内核的一个模块，当需要创建虚拟化服务器时，KVM 被加载进入内核，将 Linux 转变为 VMM，Linux 内核的进程调度策略与内存管理方法均可以应用于虚拟机管理 ，在 KVM 中每个虚拟机对应 Linux 中一个进程，因此 KVM 是轻量级的虚拟化解决方案。

在标准的 Linux 环境中每个进程可以运行在用户模式和内核模式两种模式，KVM 在 Linux 中引入第三种模式，客户模式。在客户模式中运行的进程有自己的内核模式和用户模式，因此在客户模式下可以运行操作系统。在 KVM 虚拟化系统中，客户模式主要运行除 I/O 操作外的客户操作系统相关的指令，在用户模型下主要执行客户模式下不能完成的 I/O 操作或特殊指令，内核模式主要负责捕获 I/O 操作或特殊指令的信息，完成客户模式到用户模式的转换，KVM 体系结构如图 9 所示[31]。

在 KVM 中 CPU 虚拟化借助 Intel VT-x (Intel Virtualization Technology for x86)或 AMD-V 技术实现，在 Intel VT-x 技术中为处理器增加了 VMX (Virtual Machine Extension) 根操作模式与 VMX 非根操作模式，CPU 的 VMX 非根模式为客户操作系统的执行模式，如果在该模式下企图执行敏感指令，CPU 将会检测到相关信息，然后将执行模式转化为根模式。

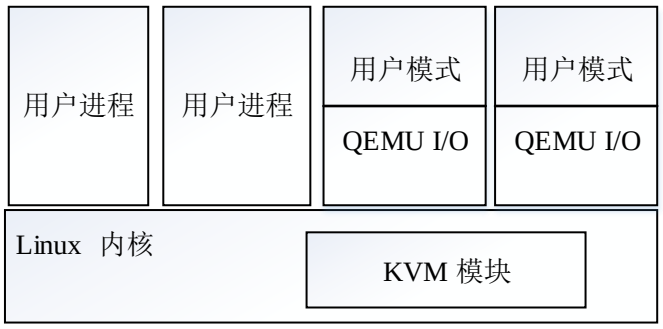

图 9　KVM 虚拟化架构

KVM 修改了 Linux 内存管理代码，为虚拟机中的每个进程页表增加一个影子页表，来实现虚拟机地址空间到物理地址的转换。为提高内存虚拟化效率，Intel 与 AMD 提出了硬件支持的内存虚拟化实现方式。KVM 本身不能创建虚拟机，需要借助 QEMU 来实

现硬件模拟，QEMU 是一个硬件模拟器，运行在 Linux 的用户模式下，它能够模拟标准 X86 体系架构中的硬件。当虚拟机需要进行 I/O 操作时，QEMU 可以解释和执行相关 CPU 指令，实现实际物理设备的访问。由于 QEMU 是通过纯软件的方式实现 I/O 模拟，需要多次上下文切换，效率较低，目前在 KVM 中常采用 virtio，一个管理器之上的 API 接口，以半虚拟化方式实现 I/O 操作。从内核版本 2.6.25 开始，virtio 成为 Linux 内核的一个模块。

2.4.2 KVM 管理工具

为了简化 KVM 中虚拟机的配置与管理，研究者对 KVM 相关命令进行了封装，开发了多种虚拟机管理工具，提供了友好的用户交互接口，简化了操作。

libvirt 是使用最广泛的 KVM 虚拟机管理工具，支持多种虚拟化方案。 libvirt 通过基于驱动程序的架构来实现对多种 VMM 的支持，libvirt 作为中间适配层，隐藏了 hypervisor 实现细节，使其对上层用户完全透明。libvirt 主要由三部分组成，分别为：应用程序编程接口库，一个守护进程和一个默认命令行管理工具。应用程序接口库为其他虚拟机管理工具提供程序库支持，如 virt-manager，守护进程负责执行虚拟机的管理工作，在使用各种工具对虚拟机管理时，守护进程一直处于运行状态，virsh 是 libvirt 中默认的对虚拟机管理的命令行工具[5]。

virt-manager 是管理虚拟机的图形化用户接口，是 KVM 虚拟化管理工具中最易用的工具之一，仅支持在 Linux 或其他类 Unix 系统中运行。virt-manager 为用户提供了丰富易用的虚拟机管理功能，包括：对虚拟机生命周期的管理，如创建，编辑虚拟机与虚拟机动态迁移等，客户机实时性能与资源利用率的图形化展示，以及虚拟机的远程管理[5]。

virt-viewer 是一个用于虚拟机图形显示的轻量级交互接口工具。由于传统的 VNC 客户端查看器不支持 SSL/TLS 加密，virt-manager 也常被用于查询虚拟机的 VNC 服务器端的信息。virt-install 命令行工具为安装虚拟机提供了一个简便易用的方式[5]。

2.4.3 KVM 虚拟化性能消耗

虚拟化技术提高了物理资源利用率，理想状态下，虚拟化服务器性能应该不低于物理服务器性能，但是由于虚拟化服务器中资源虚拟化的特性，使其存在性能降低风险，即虚拟化技术可能导致额外的性能消耗。虚拟化系统中的 VMM 主要负责虚拟化系统中 CPU，内存与 I/O 操作的虚拟化资源调度，其直接影响虚拟化服务器性能。

目前很多研究者通过具体实验，从不同角度对 KVM 虚拟化系统中性能消耗进行了

定量或定性研究。P.Vijaya 等研究者在 CloudStack 上创建了以 KVM 为虚拟化实现方式的虚拟化系统，并且为每个虚拟化系统配置了相同的物理操作系统，然后借助基准测试工具分别产生 CPU 密集型，内存密集型与 I/O 密集型应用请求，通过对不同系统中 CPU 使用率，可获得内存量以及 I/O 读写速度与网络 I/O 接收和发送数据的速度来比较分析 KVM 性能优劣 [33]。该研究实验结果显示，在 CPU 密集型操作与内存密集型操作中，KVM 性能消耗最大，低于本机系统性能，在磁盘 I/O 密集型操作中，KVM 与本机系统相比有多于 30%左右的性能消耗，在网络 I/O 实验中，KVM 性能消耗高于本机系统 22%左右。Jianhua 等研究者通过三个实验，使用多种基准测试工具定量与定性比较分析了 KVM 和 Xen 虚拟化系统性能消耗，其中 KVM 中 I/O 虚拟化借助 QEMU 实现。该研究实验结果显示，Xen 虚拟化系统中 CPU 虚拟化消耗要低于 KVM，因为 KVM 中需要对每个执行命令进行检查判读其是否为敏感指令，在内存虚拟化中，一般情况下，KVM 通过影子页表实现客户虚拟内存到物理内存转换，性能消耗高于 Xen，并且通常 Xen 具有更高的上下文切换速度，在文件 I/O 操作中，KVM 中基于软件模拟的 I/O 虚拟化方式性能消耗较大[34]。

基于已有的研究成果可以得出，KVM 中 CPU 上下文切换与 I/O 虚拟化实现方式对其性能消耗有重要影响。KVM 作为一个新兴企业级虚拟化解决方案，在虚拟存储支持，安全性，容错性等方面还需要进一步改进。由于 KVM 与 Linux 内核进行了集成，KVM 可以直接获益于 Linux 内核发展成果，因此 KVM 性能将会得到快速改进完善。在 SPEC 公布的 2015 年 11 月份的 SPECvirt_sc2013 基准测试结果中，HP 与 IBM 以 KVM 为虚拟化解决方案对其虚拟化系统进行了性能测试，实验数据显示，与其他虚拟化技术相比，KVM 可以使单个物理机上运行的虚拟机数达到较高水平[7]。

2.5 本章小结

本章首先介绍了服务器虚拟化实现方式及其特点，然后阐述了虚拟化服务器性能度量的不同维度，以及每个维度涉及的度量指标。随后根据目前已有的研究成果，总结分析了虚拟化服务器中不同性能度量指标对应的影响因素，然后从应用模拟与理论分析的角度，选取了具有代表性的性能评估方法对其度量项选择与实现原理进行了详细分析说明，最后介绍了 KVM 虚拟机技术的架构与虚拟化系统中主要组件的虚拟化原理，结合目前应用实践，选取了常用的 KVM 虚拟机辅助管理工具进行了简单说明，并且基于已有的研究，分析了 KVM 中虚拟化性能的消耗。

第三章 基于 KVM 虚拟化服务器的性能评估模型

服务器虚拟化已经成为服务器部署的默认模式，目前有多种虚拟化解决方案，如 VMware ESxi，Citrix XenServer 等。由于宿主模式虚拟化技术的局限性，本研究提到的虚拟化是指类型 I 的虚拟化，KVM 在本研究中被归为虚拟化类型 I。2006 年 KVM 被正式发布，是一个相对较新的虚拟机管理程序。由于 KVM 是 Linux 系统的默认虚拟化解决方案，因此随着 Linux 服务器使用数量的不断增加，KVM 也逐渐成为主要的开源虚拟化解决方案，因此基于 KVM 的虚拟化服务器性能评估成为研究热点。本章通过对 KVM 虚拟化实现方式的深入分析，建立了虚拟化服务器性能评估模型，以方便对虚拟化服务器进行评价。

3.1 KVM 虚拟化服务器性能评估指标体系

3.1.1 性能评估度量项分析

虚拟化技术可以将多个应用整合在一台物理机上，但是将应用迁移到虚拟机后，必须保证应用性能可以被用户接受，即应用性能满足 SLA 协议。根据 SLA 协议中 QoS 参数属性，可以得到响应时间与吞吐量是各种网络应用性能评估的主要指标。从实际应用角度，目前常见的基准测试工具，如 SPECvrit_sc2013 等，也采用响应时间与吞吐量作为虚拟化服务器性能评估指标，因此选择的性能评估度量项为：

1) 响应时间：服务器处理请求消耗的时间，即用户请求到达虚拟服务器的时间与虚拟服务器处理完成请求的时间差值。
2) 吞吐量：虚拟服务器每秒处理的请求数量。

3.1.2 性能影响因素分析

虚拟机中应用的性能与虚拟化系统中虚拟机，VMM 以及物理资源之间的交互有关，虚拟机中请求的类型，VMM 对资源的调度管理是影响虚拟化系统中应用性能的主要因素。VMM 为其上层虚拟机操作系统提供了透明的服务，使运行之上的虚拟机操作系统“完全控制”物理硬件资源，VMM 通过干预关键事件的执行来为虚拟机提供完整的执行环境，VMM 完全控制虚拟化环境中资源的分配。如图 10 所示，VMM 中要实现服务器中 CPU，内存与磁盘这三类资源的虚拟化处理与调度。

1) 负载类型对虚拟化服务器性能影响

在 KVM 中，Linux 即为 VMM，是虚拟机与物理资源的中间层，实现客户虚拟机行为与物理资源操作之间的映射，当虚拟机中应用请求改变物理资源状态时，KVM 需要

借助硬件虚拟化技术将 CPU 从非根模式转换为根模式，当虚拟机中应用需要对文件进行读写操作时，KVM 要通过系统中用户模式下的 I/O 线程，对物理磁盘进行相关操作，得到处理结果。

由于在初始的 X86 体系中 VMM 需要捕获敏感的指令进行处理，因此需要多次执行客户机退出操作，性能消耗较大，为了解决 X86 体系的设计缺陷，主要处理器供应商 Intel 与 AMD 提出了硬件支持的 CPU 虚拟化技术。 KVM 借助硬件支持实现 CPU 虚拟化，减少了虚拟机退出次数，并且虚拟机的中断执行都是由硬件完成，提高了虚拟化性能。在内存虚拟化方面，KVM 通过影子页表实现虚拟服务器内存虚拟地址到主机物理地址的转换，即基于软件的内存虚拟化方式，KVM 需要为虚拟服务器中的每个进程页表都维护一套相应的影子页表，资源消耗较大，为了简化内存虚拟化实现方式，Intel 引入了扩展页表（Extended Page Table，EPT）技术，AMD 引入了嵌套页表（Nested Page Table, NPT）技术，即借助硬件支持实现内存虚拟化。在硬件支持的内存虚拟化技术中，不需要通过 VMM 辅助，客户机操作系统可以直接对页表进行操作。

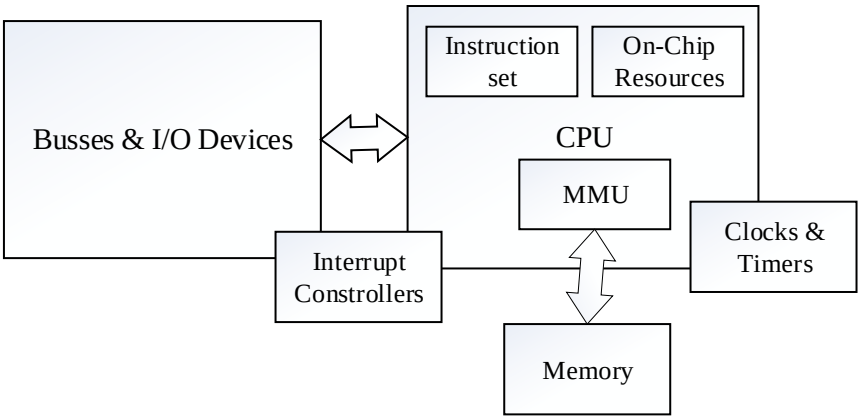

图 10　服务器组件

I/O 是系统中一个重要组件，KVM 中 I/O 虚拟化需要借助 QEMU 模拟器实现，在处理 I/O 请求时需要多次上下文切换，效率较低。KVM 中由内核模式进入客户模式的上下文切换过程可以归纳为三个步骤[35]：1) KVM 保存自己的上下文；2) KVM 将 VCPU（Virtual CPU）对应的数据结构中的上下文加载到物理 CPU 中；3) KVM 执行 kvm_x86_ops 中的 VCPU 运行函数，调用具体的平台指令，进入虚拟机环境。目前 virtio，I/O 半虚拟化实现方式，被广泛应用与 KVM 虚拟化环境中，实现了 I/O 请求的批处理，减少了上下文切换次数。由于在 KVM 中，CPU 相关操作可以由硬件辅助执行，而 I/O 操作需要借助软件或半虚拟化方式实现，并且虚拟机中 I/O 操作要由对应物理 I/O 操作

完成。因此在 KVM 中，对于高 I/O 负载的操作虚拟化性能与 CPU 密集型操作性能存在一定差别，即 KVM 中负载类型会影响虚拟机性能。

2) 资源调度对虚拟化服务器性能影响

在 KVM 中，虚拟机作为一个普通的 Linux 进程运行，由 Linux 操作系统调度程序进行统一调度。Linux 中调度程序实现了一系列调度策略，这些调度策略决定了一个任务何时可以得到处理，影响了 KVM 中虚拟机性能。

Linux 将进程分为实时进程与普通进程两种，基于此调度策略可以分为实时调度策略与普通调度策略。实时进程的调度优先级高于普通进程，实时调度策略针对应实时进程，用于时间关键性任务，分为先入先出调度策略(First Input First Output, FIFO)和时间片轮转调度策略(Round-Robin, RR) [36]。FIFO 也被称为静态的优先级调度，其为每个进程定义了 1 到 99 之间的固定优先级，调度程序会扫描以优先级排列的 FIFO 进程列表，调度高优先级进程运行，当进程执行结束，或者受阻，或者被就绪的更高优先级进程抢占时，进程结束执行。在 RR 调度策略也给定先出一个固定的优先级，但是相同优先级的进程以时间片轮转形式调度，适用于多个进程处于相同优先级的情况。

当系统中不存在实时进程或实时进程都已运行结束后，普通进程开始被运行。普通进程调度策略包括完全公平性调度策略(Completely Fair Scheduler, CFS)，批处理调度策略以及空闲调度策略，其中批处理调度策略以及空闲调度策略针对于优先级较低任务。CFS 是普通进程默认的调度策略，CFS 基于进程的优先级修正参数，建立了进程动态优先级列表，通过引入进程的等待时间来确定进程执行顺序，确保系统中进程尽可能公平的共享 CPU 时间。在 Linux 中虚拟机作为普通进程，因此虚拟机进程必须在实时进程之后运行，会对虚拟机中应用性能产生影响。

另一方面，当系统中存在多个 CPU 时，调度方法可以分为非对称多处理器调度以及对称多处理器调度。在非对称多处理器策略中一个处理器是主控制器，控制其它内核的运行活动，对称处理器策略中从共享就绪队列，或者从分开的各自的就绪队列中调度自己的任务。在多处理环境中会涉及处理器关联问题，处理器中包含缓存，可以加快数据访问，如果处理器从一个处理器转换到另一个，则会使缓存中内容失效，新处理器重新加载。有研究者设计了相关实验研究处理器的关联性对虚拟机性能的影响，实验结果显示，在具有 24 个虚拟机的 Xen 虚拟化环境中，使用 CPU 基准测试工具进行测试时，将虚拟机进程保持在相同的处理器的性能比虚拟机进程在处理器之间相互轮转的性能高 9.65%，在内存基准测试中，两者相比性能提高 18.82%[27]。

由 KVM 中 I/O 虚拟化实现方式可知，由于 virtio 实现了 I/O 请求的批处理，因此纯软件实现的 I/O 虚拟化性能低于借助 virtio 实现的 I/O 半虚拟化性能。在 KVM 中，QEMU 进程实现 I/O 处理。在 Linux 中，I/O 调度策略分为完全公平排队(Completely Fair Queuing, CFQ)，Deadline I/O 调度以及 Noop I/O 调度。CFQ 是 Linux 系统的默认调度策略，该调度策略的目的是力图为触发 I/O 的进程提供具有一定公平性的 I/O 调度策略，CFQ 为每个 I/O 进程分配时间片。Deadline I/O 调度策略力图为请求提供延迟性保障，即确保请求在一定时间内完成 I/O 操作，默认情况下读操作给定的优先级要高于写。Noop I/O 调度实现了简单的先进先出调度算法。Steven L.P 等研究者[37]在多种负载场景中，如 web 与邮件服务应用场景，研究了四种 I/O 调度策略，实现了 CFQ，deadline 与 Noop 以及预期 I/O 调度策略的定量性能分析，实验结果显示，在 RAID-5 环境中 CFQ 是效率最高的方案，因此 Linux 中 I/O 调度策略会影响应用性能。

在 KVM 中，客户虚拟机进程从主 qemu-kvm 进程中继承内存，当虚拟机运行相同的操作系统或应用时，可以共享虚拟机操作系统镜像的内容。合并内存页(Kernel Samepage Merging, KSM)技术可以将 KVM 中相同内存页面合并为一个内存页面，优化了 KVM 中内存使用，提高了内存访问速度与使用率。在 KSM 中共享进程数据被存储在缓存或主内存中，减少了 KVM 客户机的高速缓存缺失，从而提高应用和系统的性能。另一方面，Linux 内核使用页式内存管理方法，将内存区域划分为 4KB 为一页的内存页表，当程序申请分配内存时，系统以内存页为单位进行分配，程序得到的内存地址是虚拟地址，当需要对内存操作时，需要通过页表进行变化，得到真正的物理地址，进行相关操作。在 X86 体系 CPU 可以使用 2MB 的内存页面替代 4KB 内存页面，即使用大内存页面。索引缓存（Translation Look-aside Buffer, TLB）是页表的高速缓存，可以加快虚拟地址到物理地址的转换速度，大内存页可以减少页数，使更多的内存可以通过 TLB 转换，因此大内存页可以减少 TLB 的失败次数提高检索性能。KVM 客户机可以部署在大内存页面中来增加 CPU 缓存命中率，对于内存密集型操作，大页面能够有效改进性能。因此以上分析可以看出，KVM 虚拟化系统中对内存分配采用不同方法，对虚拟化系统有不同影响，并且在 KVM 中内存虚拟化实现形式不同，性能也不同，所以在 KVM 中内存分配可以影响虚拟机应用中性能。

在服务器整合环境中，虚拟数量对各虚拟机访问共享资源的效率有直接影响，因此虚拟机数量也是影响虚拟机中应用性能的重要因素。基于以上分析可以得出 KVM 虚拟化环境中应用性能度量指标及其影响因素，图 11 展现了虚拟化服务器性能度量指标以

及影响因素之间的层次关系。

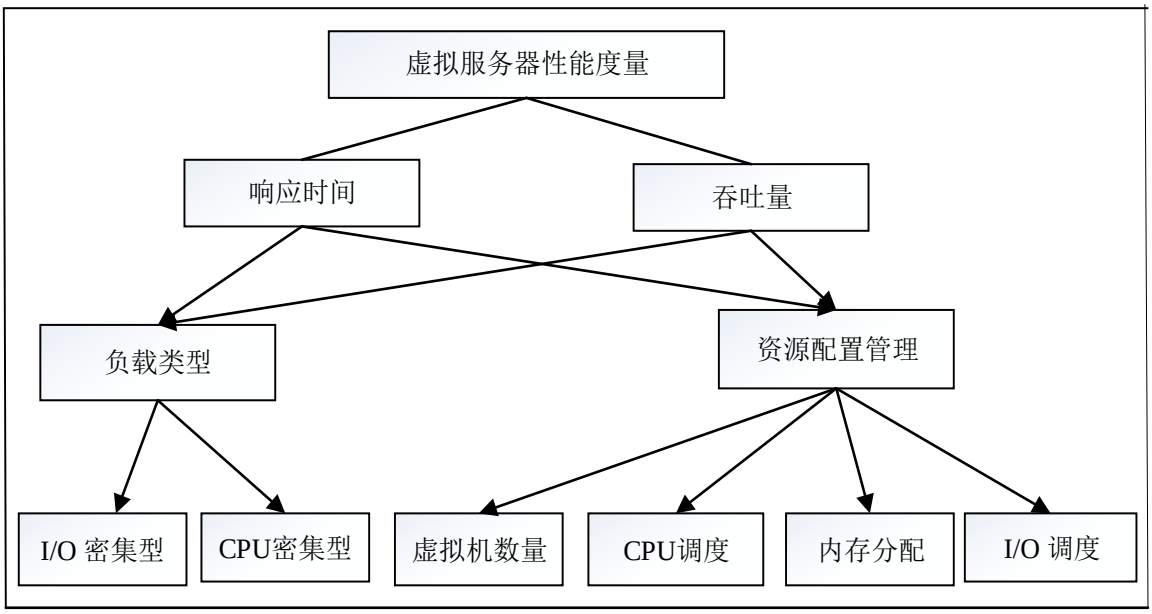

图 11　KVM 虚拟化服务器性能评估指标体系

响应时间与吞吐量是虚拟化服务器性能评估的度量指标，负载类型与资源配置方式是影响度量指标的主要因素，其中负载类型主要分为 I/O 密集型与 CPU 密集型两类，资源配置方式主要涉及 CPU 调度，内存分配与 I/O 调度，以及虚拟机数量。

3.2 基于排队网络的 KVM 虚拟化服务器性能评估模型

排队网络模型被广泛用于展现和分析系统中资源的使用，在 KVM 中虚拟化服务器性能影响因素主要涉及资源使用，并且目前对虚拟化系统的分析建模也多基于数学排队论，因此本研究借助排队网络模型抽象虚拟化系统，分析虚拟化系统资源使用特征，建立性能评估模型。

3.2.1 KVM 资源虚拟化实现方式

KVM 借助硬件支持实现 CPU 虚拟化，以 Intel VT-x 技术为例进行分析。Intel VT-x 技术主要目的是，当检测到客户模式中的 CPU 执行敏感指令时将进程转换到 VMM 处理。为了解决实现虚拟化技术时 X86 体系存在的缺陷，Intel VT-x 技术为 CPU 引入了根操作模式与非根操作模式，被统称为 VMX 操作模式[38]，如图 12 所示。这两种操作模式中的特权级与 X86 体系相同，都有 0 到 3，4 个特权级别。AMD 的 SVM（Secure Virtual Machine）提供的 CPU 虚拟化技术与 Intel VT-x 原理类似，通过引入新的模式来解决虚拟机中敏感指令的异常陷入问题。

客户操作系统运行在 CPU 的 VMX 非根模式下，如果客户操作系统企图执行敏感指

令，则 CPU 将会执行 VM Exit 命令，切换到 VMX 根模式下运行，当指令执行结束后，CPU 执行 VM Entry 命令进入 VMX 非根模式，客户操作系统继续运行。此处敏感指令是指企图改变系统资源状态的控制敏感指令与操作系统资源状态的操作敏感指令。KVM 控制 CPU 中 VM Exit 与 VM Entry 命令的执行，为保证 CPU 状态转换的正确性，虚拟化技术创建相应数据结构来保存 CPU 不同模式下的相关状态。

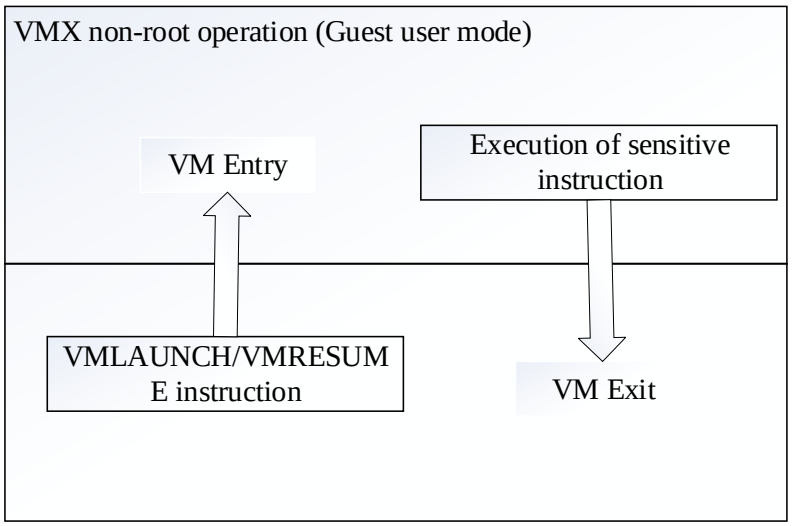

图 12　Intel VT-x 架构

VCPU 工作模式如图 13 所示。当要虚拟机启动时，QEMU 调用驱动函数 ioctl 通知 KVM 内核启动客户虚拟机，然后 KVM 模块执行 VM 进入命令，开始运行客户虚拟机中应用。当虚拟机执行敏感指令时，CPU 执行 VM 退出命令，KVM 识别退出原因，如果需要 QEMU 进行干预，则 QEMU 获得控制权，执行任务。当任务执行结束后，调用函数 iotcl 进入 KVM 模块，KVM 加载虚拟机状态，运行客户虚拟机。

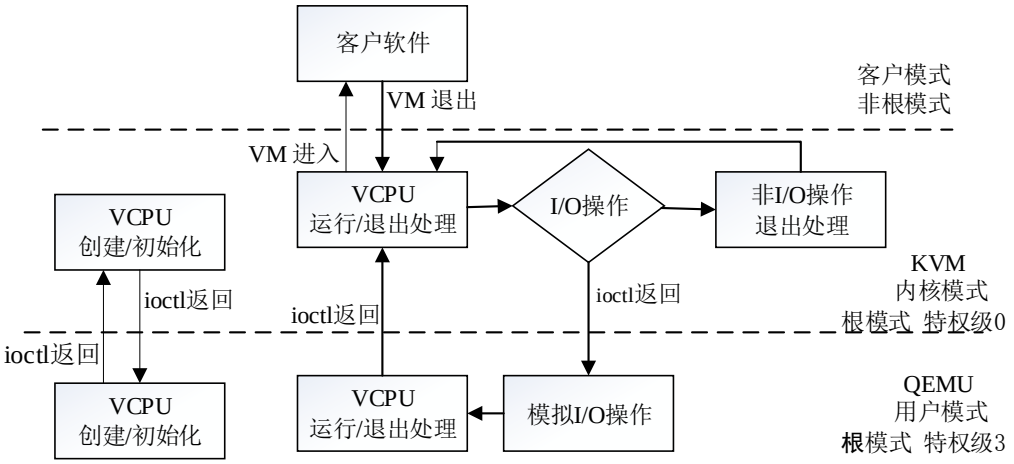

图 13　vcpu 工作模式

Virtio 是 VMM 上的抽象 API 接口，针对不同的设备模拟不同驱动，现在也是 Linux 中标准化的 I/O 半虚拟化设备，由前端驱动与后端驱动两部分组成，其中前端驱动是客

户机中存在的驱动程序模块，在磁盘操作中前端驱动即为 virtio 中的块驱动，后端驱动程序是在 Linux 用户模式中的 QEMU 中实现的，如图 14 所示。Virtio 在前后端之间，增加了两层结构来处理前端与后端通信，一层是虚拟机队列接口，在概念上将前端驱动程序附加到后端驱动程序。virtio 在用户模式组件中实现了虚拟队列(Virtual Queue，vqueue)，能够访问虚拟机的地址空间，被客户机操作系统与虚拟机管理程序共享，vqueue 分为出站队列与入站队列两种。在出站队列中，客户机操作系统添加数据，VMM 获取数据，在入站队列中 VMM 提供数据，客户机操作系统使用数据。另一层是环形缓冲区，用于保存前端驱动与后端驱动的执行信息，是 vqueue 的具体实现方式[39] 。因为 virtio 可以一次性将多次 I/O 请求，传递给主机中物理设备进行处理，因此提高了客户虚拟机与虚拟机管理程序信息交换效率。

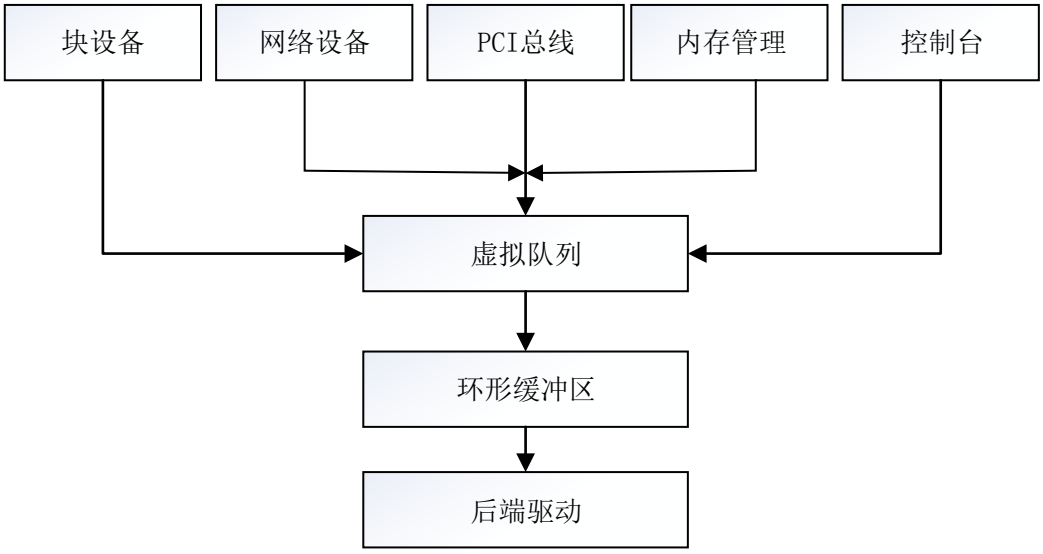

图 14　virtio 架构

当虚拟机中应用程序需要进行 I/O 操作时，首先通过相应的系统调用来初始化读写操作，客户操作系统将信息“写入”驱动，或从驱动“读入”信息，前端驱动计算操作所需的缓冲区数量。在写操作时，从出站队列取出描述符放入缓冲区，然后通过将相关总线信息写入对应的寄存器来通知 virtio 设备。KVM 内核解析 VM 退出命令产生原因，确定是由于 I/O 操作引起后，退出到用户模式。在用户模式中，根据 PCI 中地址检测访问 QEMU 中的 virtio 设备，调用相应功能，获取共享内存结构。在写操作时，从出站队列获取相关信息，在读操作时，在入站队列获取相关信息。最后在用户模式中的 virtio 后端驱动通过相应的系统调用初始化读写操作，宿主操作系统获取物理设备驱动，执行相关 I/O 操作，返回数据或确认信息。virtio 后端设备将结果信息写入共享内存结构，用户模式组件调用 KVM 内核模块，执行 VM 进入命令，恢复虚拟机的状态信息。客户模

式中的前端驱动获取数据，提交给客户操作系统，再通过系统调用返回给虚拟机中应用[40]，如图 15 所示。

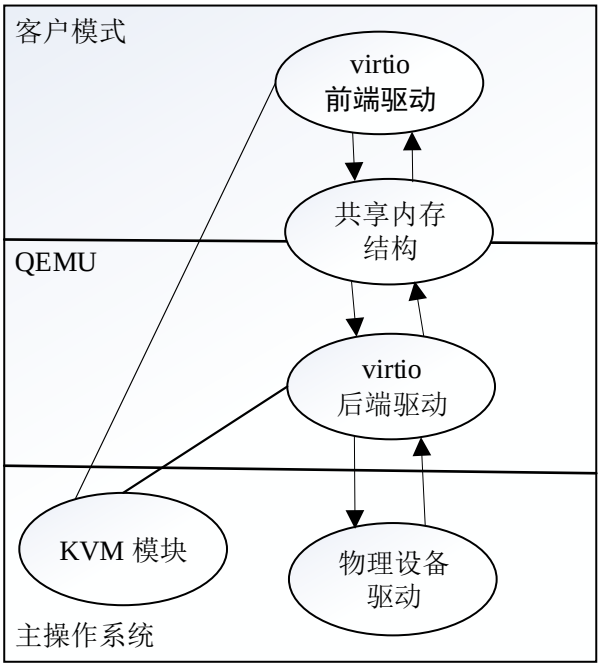

图 15　virtio 中 I/O 操作实现原理

在内存虚拟化中，为保证客户机具有从零开始的连续的内存空间，KVM 引入了客户机物理地址空间，该地址空间是 VMM 抽象的，并不是真正的物理地址空间，虚拟化系统实现内存虚拟化即实现客户机虚拟地址到实际物理地址转化。KVM 一般采用影子页表方法实现内存虚拟化，影子页表可以实现客户机虚拟地址到宿主机物理地址的直接映射。在客户机访问内存时，真正被载入宿主机存储单元的是当前页表对应的影子页表，从而实现内存地址的直接转换，达到的效果如图 16 所示。

由于使用影子页表时，KVM 需要为客户机中的所有进程维护一套影子页表，开销较大，并且每次内存访问都需要 KVM 进行转换，CPU 中上下文切换频繁，因此效率较低。目前基于硬件支持的内存虚拟化技术被广泛应用，该技术将客户机虚拟地址到客户机物理地址，以及客户机物理地址到宿主机物理地址的两次地址转换由 CPU 自动完成[41]。以 Intel VT-x 技术的 EPT 为例，KVM 预先将实现了客户机物理地址到宿主机物理地址的 EPT 页表载入到 CPU 中，并且客户机可以修改设置 CPU 中客户机虚拟地址到客户机物理地址的页表映射。当进行地址转换时，CPU 自动查找两张页表完成地址转换，其结构如图 17 所示。

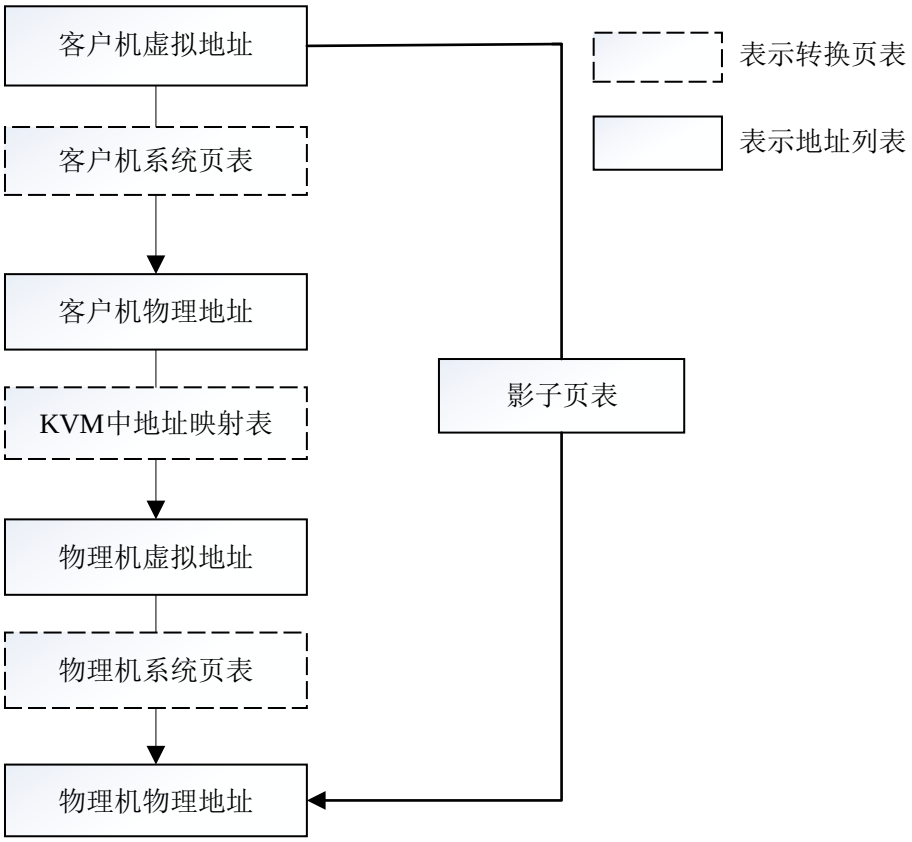

图 16　影子页表

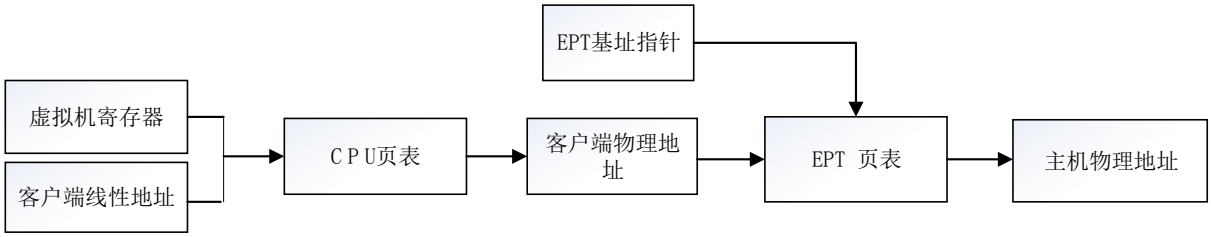

图 17　EPT 实现原理

3.2.2 虚拟化服务器负载特征

虚拟化服务器是传统物理服务器的替代模式，因此虚拟化服务器处理的负载类型与物理服务器相似。典型的服务器负载类型包括数据服务器，web 服务器与文件服务器等，每类服务器中用户访问模式存在一定差别[42]。本研究选择对数据库服务器负载与 web 服务器负载进行分析。

由于无法直接获得实际应用领域用户的访问模型，SPEC 组织为模拟真实的用户使用环境，在进行基准测试工具设计时，调研分析了实际使用环境中用户访问模式的特征，因此本研究借助 SPEC 组织各负载工具的负载特征，分析虚拟化 web 服务器与数据库服务器中实际常见负载类型。

在设计 SPECweb 基准测试工具时，SPEC 组织分析了真实的电子商务网站的日志文

件，并对其相关 web 网站访问数据进行了分析。借助马尔科夫链模拟了用户访问时页面的请求频率，因此 SPECweb 产生的负载可以近似于实际应用环境中的用户访问负载。在 web 服务器中主要涉及两类负载，页面查询浏览与文件下载，并且负载中两次请求之间存在一定时间差，即用户在发出一个请求后会随机等待一段时间发出另一个请求，因此 web 服务器中用户请求到达率近似服从指数分布。

在实际应用中数据库服务器常作为应用服务器的支持服务器。SPECjbb 基准测试工具模拟了一个全球性的购物平台，数据库作为其后端支持系统，负载分为联机事务处理型负载与联机分析处理型负载。在 SPECjbb 中每类的负载注入率是变化的，注入率由一个注入曲线决定，曲线上有 39 个不同注入率，每个注入率执行 40 秒，平均注入率为每秒钟 100 个请求。联机事务处理型负载（On-Line Transaction Processing, OLTP）特点是短时间内大量数据交易，主要涉及插入，更新与删除操作。OLTP 类型负载主要强调在多用户环境中快速处理查询进程，并维护数据一致性[28]。联机分析处理型负载交易量相对较低，专门设计用于支持复杂的分析操作，其查询操作常常比较复杂并且涉及聚合，侧重对决策人员的决策支持。对于普通用户而言，在获取应用服务时主要涉及联机事务处理型负载，该负载类型可以通过每秒中处理事务数刻画。

通过对 web 服务器与数据库服务器的负载分析可以得出，实际应用 web 服务器与数据库服务器一般为多类负载，并且每类负载在一个时间段内可近似视为以固定速率到达。通常使用 CPU 使用率来刻画 web 服务器中负载，虚拟化 web 服务器应该被分配充足的 CPU 与内存，即可以使 web 负载为 CPU 密集型负载。数据库负载一般视为 I/O 密集型负载，需要较大的内存以及高性能 I/O 块设备。

3.2.3 KVM 虚拟化服务器性能评估模型

通过 3.2.2 节中对虚拟化服务器负载特征的分析可以得出虚拟化服务系统性能模型应该属于多负载类开放型排队网络模型。

在 KVM 中，当用户请求到达虚拟服务器时，首先由虚拟化服务器中虚拟 CPU，即处于 VMX 非根模式下的物理 CPU，对用户请求进行处理。如果涉及敏感指令执行，则 KVM 控制 CPU 由非根模式退出，并将客户机操作系统信息保存到虚拟机控制结构（Virtual Machine Control Structure，VMCS），然后宿主操作系统状态被载入处理器，CPU 进入 VMX 根模式，CPU 进入 VMX 根模式后，KVM 读取 VMCS 中相关信息，判读执行 VM Exit 命令的原因。如果敏感指令为非 I/O 操作，则调用 KVM 中相关函数进行处理。如果敏感指令为 I/O 操作，则宿主系统进入用户模式，运行在用户模式下的 QEMU

中的 virtio 后端程序获取共享内存中相关数据，然后调用宿主系统中相关系统调用命令，获取物理设备驱动，在实际物理设备中执行 I/O 操作，其执行原理如图 18 所示。

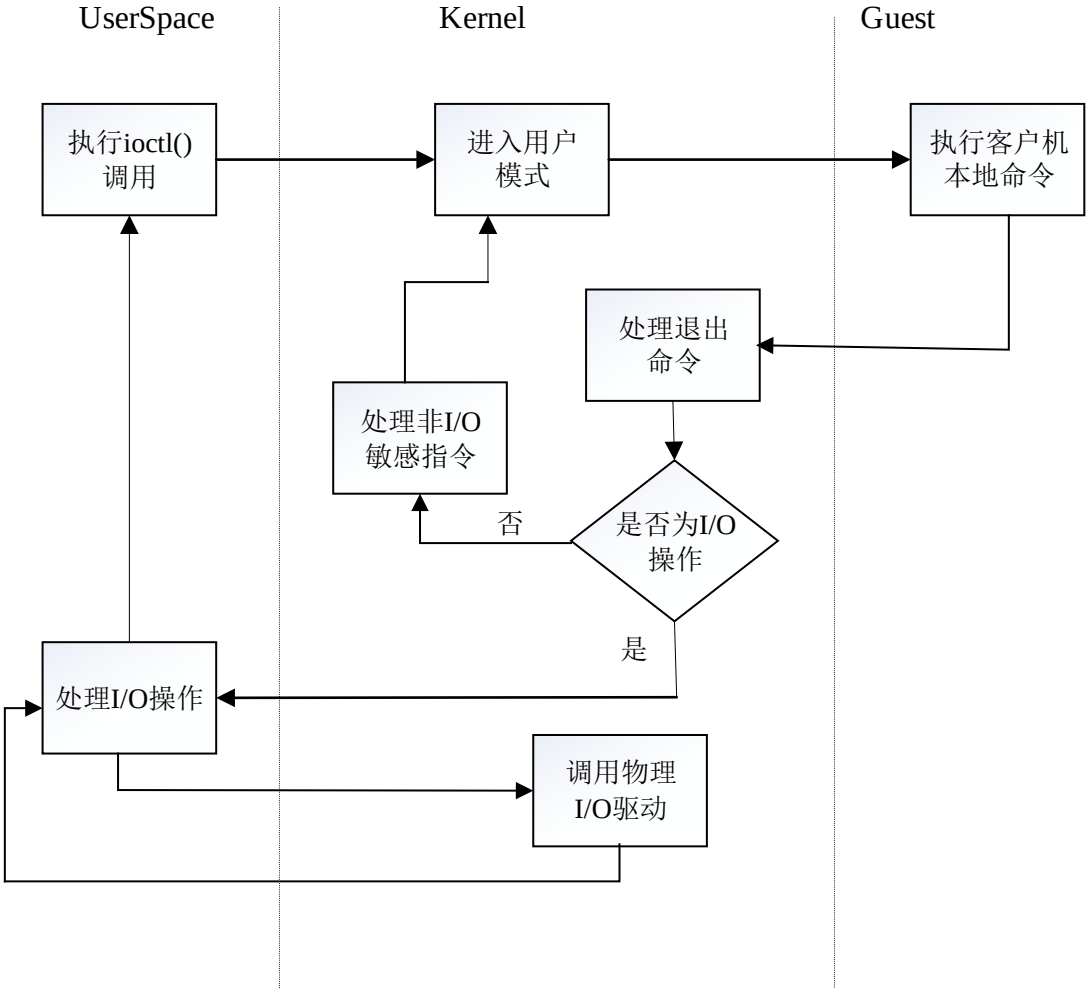

图 18　KVM 执行原理

多负载类开放型排队网络模型的求解过程与单负载类排队网络模型相同，可以由单负载类开放型排队网络模型直接推广至多负载类开放型排队网络模型。本研究的目的是评估虚拟化服务器性能，由单负载类性能评估模型获得的性能度量值，也可以反映系统性能优劣，因此本研究选择建立 KVM 虚拟化系统的单负载类排队网络模型。由 KVM 中请求处理流程可以得到 KVM 虚拟化系统的性能评估模型。

图 19 展示了当虚拟环境中有 4 个虚拟机时，虚拟系统的排队网络模型，其中 VM1、VM2、VM3 与 VM4 分别代表虚拟机 1、虚拟机 2、虚拟机 3 与虚拟机 4，Disk 表示物理磁盘，KVM 表示加载的内核中的 KVM 模块，QEMU 表示 Linux 用户模式中的 QEMU 进程。在 KVM 中，每个虚拟机在用户模式有其对应的 QEMU 进程来处理 I/O 操作。

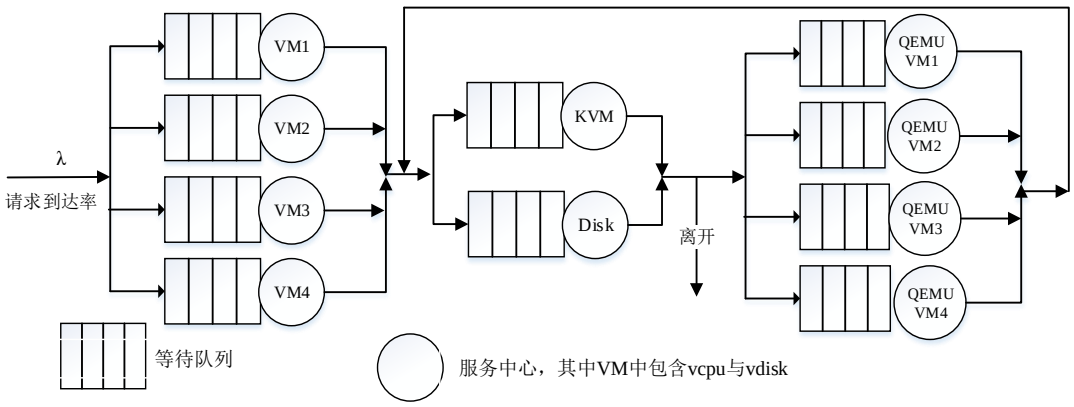

图 19　虚拟化服务器性能评估模型

假设用户请求到达率为 λ，请求在 VM 的消耗时间记为R_{VM}。在 VM 中请求首先由其中虚拟 CPU 进行处理，消耗时间记为R_{vcpu}。如果请求涉及 I/O 操作，则虚拟机中前端驱动将请求信息放入共享内存，消耗时间记为R_{vdisk}。虚拟机调用敏感指令以完成请求处理时，虚拟机退出，内核中 KVM 模块检测其原因。如果为非 I/O 操作，则 KVM 调用相应处理函数执行命令，消耗时间记为R_{kvm}，则请求处理结束，如果为 I/O 操作，用户模式下的 QEMU 中 virtio 后端程序根据共享内存中信息，通过宿主操作系统的系统调用对物理磁盘 Disk 进行操作，消耗时间记为R_{disk}，因此则该排队网络模型中一般请求处理总时间可以表示为：R_{req}=R_{vcpu} + R_{kvm}+R_{vdisk}+R_{disk}。因为虚拟 CPU 是物理 CPU 处于 VMX 非根模式下的运行状态，KVM 是 CPU 中一个模块，因此 R_{cpu} = R_{vcpu} + R_{kvm}，可以得到R_{req}=R_{cpu} + R_{disk}+R_{vdisk}，非 I/O 操作请求处理总时间为：R_{req}=R_{vcpu} + R_{kvm}=R_{cpu}。模型的吞吐量为单位时间内系统处理的请求数。

3.3 性能评估指标计算方法

由于直接获取 KVM 虚拟化系统中各部分处理请求的时间比较困难，当虚拟机都采用 Linux 操作系统时，由于 Linux 与 KVM 为开源软件，可以比较容易获得系统中处理请求的相关进程信息。因此在系统进程级统计相关数据，使用评估模型进行性能度量指标计算。另一方面，目前常用的基准测试工具，多采用求均值的方法计算性能度量值。虚拟化系统性能评估研究人员也使用性能度量值的均值来评估虚拟服务器性能。

1) 请求在 CPU 中消耗的时间

在客户操作系统中借助 top 命令查看系统的进程信息，统计与应用相关的进程在 CPU，即虚拟 CPU 上消耗的时间，通过 iotop 命令获取磁盘操作信息。在采用 virtio 技术的 KVM 虚拟机中，对磁盘的操作即为对共享内存的操作。另一方面，借助 Linux 系

统 sys/block/目录下的磁盘状态文件可以得到磁盘活跃时间，请求队列长度等信息。对所有磁盘操作数据进行分析计算，得到应用请求磁盘操作消耗的时间，最后将请求在各部分驻留的时间相加获得系统对请求的平均处理时间。基于对相关日志文件分析，得到虚拟化服务器处理请求数，请求总数与服务总时间之比得到虚拟化服务器吞吐量，具体计算方法如公式(3.1)与(3.2)所示。

$$T_{cpu,req} = T_{cpu,pid_i} + T_{cpu,pid_j} + \cdots + T_{cpu,pid_k} \tag{3.1}$$

$$R_{cpu,req} = \frac{T_{cpu,req}}{\lambda_{total}} \tag{3.2}$$

其中T_{cpu,pid_i}表示在服务器处理请求期间，进程 i 使用的 CPU 时间，进程$pid_i, pid_j \cdots pid_k$表示处理请求相关的应用进程，$\lambda_{total}$表示服务器处理的请求总数，$R_{cpu,req}$表示 CPU 处理请求的平均时间。

2) 请求在磁盘中消耗的时间

由于无法直接获取请求在磁盘驻留时间，本研究通过服务器处理请求期间，磁盘总的操作数与请求涉及的磁盘操作的比值来计算磁盘使用率，然后通过排队网络模型中磁盘使用率与服务需求的关系，计算得到请求在资源中心消耗的时间，公式如(3.3)至(3.6)所示。

$$U_{disk,total} = \frac{T_{disk}}{T} \tag{3.3}$$

$$IO_{disk,total} = Read_{disk,total} + Write_{disk,total} \tag{3.4}$$

$$IO_{disk,req} = Read_{disk,pid_i} + \cdots + Read_{disk,pid_k} + Write_{disk,pid_i} + \cdots + Write_{disk,pid_k} \tag{3.5}$$

$$U_{disk,req} = \frac{IO_{disk,req}}{IO_{disk,total}} * U_{disk,total} \tag{3.6}$$

其中 T 指服务器处理用户请求的总时间，T_{disk}表示服务器处理用户请求期间磁盘忙碌的时间，$Read_{disk,total}$与$Write_{disk,total}$分别表示服务器处理用户请求期间总的磁盘读写数，$Read_{disk,pid_i}$与$Write_{disk,pid_i}$分别表示请求涉及的磁盘读写操作数，其中pid_i，$pid_j \cdots pid_k$表示与请求相关的磁盘访问进程，$U_{disk,total}$表示服务器在处理请求期间磁盘总的使用率，$U_{disk,req}$表示处理请求造成的磁盘使用率，请求进行 I/O 操作花费的时间如公式(3.7)所示。

$$D_{disk,req} = \frac{U_{disk,req}}{X_{disk,req}} \tag{3.7}$$

其中$X_{disk,req}$为磁盘吞吐量，即磁盘在单位时间内处理的读写操作数，则磁盘处理 I/O 请求的响应时间如公式(3.8)所示。

$$R_{disk,req} = \frac{D_{disk,req}}{1-U_{disk,req}} \tag{3.8}$$

在 KVM 虚拟化服务器中，通过计算请求在排队网络模型中各服务中心消耗的时间总和，可以得到虚拟化服务器中请求响应时间。

3) 吞吐量

各服务器中心的吞吐量，可以通过计算其单位时间处理进程数获得。虚拟服务器吞吐量计算处理的总请求数与总请求处理时间的比值获得。

3.4 KVM 虚拟化服务器性能评估模型适用范围

本研究提出的性能评估模型从服务使用者角度出发，选择了响应时间与吞吐量作为评估虚拟化服务器性能度量指标。通过计算请求在虚拟化系统中各部分的处理时间与吞吐量获得性能度量的定量度量。基于虚拟化系统的定量性能度量值，可以比较分析不同虚拟化系统性能优劣。

使用本研究提出的性能评估模型，对虚拟化服务器性能进行评估时，可以获得虚拟化系统中各主要组件的性能度量值，通过分析比较不同资源中心性能度量值，可以帮助服务提供商发现虚拟化系统性能瓶颈，从而进一步提升虚拟化系统性能。本研究关注于用户请求到达后虚拟化服务器的处理过程，通过排队网络模型抽象系统处理请求流程，建立了性能评估模型，因此在本性能评估模型中没有考虑 KVM 中网络 I/O 性能，并且不涉及虚拟化服务器响应请求到达用户的网络传输时间，所以本模型对虚拟化系统进行评估，只是针对虚拟化系统对请求的处理性能。由于网络可能出现拥塞情况，导致丢包，因此使用本模型计算获得的评估结果越好，不一定说明用户体验越好，这是本模型与端到端基准测试工具的一个区别。

针对服务器供应商而言，虚拟化系统的可扩展性以及资源利用率也是其性能评估的重要方面。目前研究者从 CPU 调度策略，内存超量使用等多个角度，提出了性能评估方法。本研究是在虚拟化环境中虚拟机配置完成后进行的应用性能评估，不涉及虚拟化系统资源配置改进等方面的定量性能评价。

3.5 本章小结

本章重点介绍了基于 KVM 的虚拟化服务器性能评估模型的建立过程。首先明确了本研究将要采用的虚拟化服务器性能评估指标，并基于性能评估指标分析了 KVM 虚拟化环境中性能度量指标影响因素，根据性能评估指标及其影响因素，建立了 KVM 虚拟化服务器性能度量指标体系。由于虚拟化服务器性能主要于其资源使用方式有关，然后

详细分析了 KVM 虚拟化环境中 CPU，内存以及磁盘 I/O 实现方式以及目前常见的应用类型，进一步说明了 web 服务器与数据库服务器的负载特征。基于虚拟化服务器负载特性与虚拟化资源具体实现，结合多负载类排队网络模型与单负载类排队网络模型特点，选择单负载类排队网络模型对 KVM 虚拟化环境中请求执行过程进行抽象，建立 KVM 虚拟化服务器性能评估模型。随后说明了在 Linux 操作系统中，基于性能评估模型的性能度量指标计算方法，最后论述了该性能评估模型的特点以及适用场景。

第四章 性能评估模型实验验证

本研究的目的是提出基于 KVM 虚拟化服务器的性能评估模型，降低性能评估成本，并且提高性能评估效率。第三章详细说明了 KVM 虚拟化环境中，性能评估模型的建立过程，以及基于性能评估模型性能度量项在虚拟化服务器中具体计算方法。本章通过实验对提出的模型进行验证，确保模型的正确性。

4.1 实验基本原则

由于本研究提出的性能评估模型不适用于端到端的性能评估，而是通过计算虚拟化系统对请求的处理速度以及效率，来对虚拟化服务器性能进行评估，因此无法通过判断该性能评估结果与端到端的基准测试工具结果是否相等来评估该方法是否正确。并且如果使用针对组件的基准测试工具，则对虚拟化系统中不同组件需要选用不同的基准测试工具进行评估。各基准测试工具选择的度量项，触发的请求类型以及性能度量项计算方法等存在一定差异，因此无法对不同组件基准测试工具得到的组件性能度量进行整合获得虚拟化服务器整体请求处理性能，所以本研究采用对比实验方法来对提出的性能评估模型进行验证。

根据 3.1 节对性能影响因素的分析，可以发现负载类型与资源配置都会对虚拟服务器性能产生影响。本研究选取资源配置中 CPU 调度与虚拟机数量这两个因素，设计了两个实验场景来验证模型的正确性，具体分析如下：

1) CPU 调度

由于在 KVM 中，虚拟机相当于 Linux 中的普通进程，Linux 中 CPU 的调度策略会影响虚拟机进程处理时间。在多核 CPU 情况下，线程在内核之间切换越频繁，其缓存缺失率越高，性能消耗越大。

在相同虚拟 CPU 数量的情况下，如果物理 CPU 核数较少，会导致虚拟 CPU 线程争用物理 CPU 内核，进程在 CPU 内核之间切换频繁，并且宿主系统必须分配额外的 CPU 时间处理资源的竞争，从而导致虚拟机性能降低。如果物理 CPU 核数大于或等于虚拟机 CPU 数，则虚拟 CPU 线程切换频率降低，CPU 虚拟化性能提高。在只有物理 CPU 配置不同的两个虚拟化系统中，如果物理 CPU 核数大于或等于虚拟 CPU 数，则虚拟化系统性能较好。如果物理 CPU 核数少于虚拟 CPU 数，则虚拟化系统性能消耗较大，性能较差。

2) 虚拟机数量

当配置环境相同，虚拟机数量越多，虚拟化系统中虚拟机进程 qemu-kvm 越多，资源竞争加剧，Linux 进程调度时间增加，从而增加虚拟化性能消耗，因此性能降低。

4.2 实验方案

根据以上分析，实验方案分为以下两类：

方案一：在两台 CPU 配置不同的物理服务器上，创建两个虚拟机，分别作为 web 服务器与数据库服务器，并且使得一台物理机的 CPU 核数等于虚拟 CPU 数，另一台物理机的 CPU 核数小于虚拟 CPU 数。通过前面的理论分析以及 Huber 等研究者的相关实验结果可以确定以上两台物理机中的虚拟化服务器性能优劣[27]，因此在已知性能优劣的情况下，本研究使用 3.2.3 节建立的性能评估模型对实验环境中的虚拟化服务器性能进行评估，将得到的实验结果与已知结论进行对比，判断是否一致，从而对模型的正确性进行验证。

方案二：在两台配置相同的物理服务器上，分别创建不同数量的虚拟机，每个虚拟机创建的虚拟 CPU 数相同，通过理论分析可以得到，虚拟机数量较少的虚拟化系统的性能消耗要少于虚拟机数量较多的虚拟化系统性能消耗，因此在虚拟机数量较少的虚拟化环境中的虚拟机性能高于在虚拟机数量较多的虚拟化环境中的虚拟机性能，与方案一相同，在已知虚拟机性能优劣的情况下，使用本研究提出的性能评估模型对虚拟机性能进行计算，通过对得到的结果进行比较，验证性能评估模型的正确性。

4.3 实验说明

4.3.1 实验环境配置

本研究在两个不同的实验场景中进行了实验，在每个实验环境中，除非特别说明，否则在宿主机与客户机中默认采用 CentOS7 作为 Linux 操作系统，virt-manager 作为虚拟机图形化管理器，具体物理机与虚拟机配置如下所示：

实验环境一：物理服务器 1 为 Dell optiplex 780，配置有主频为 2.66GHZ 的具有 4 个逻辑核的 Intel 酷睿 CPU，4GB 内存以及 500GB 硬盘，物理服务器 2 配置有 2 个逻辑核的 Intel CPU，2GB 内存以及 500GB 硬盘。在两台物理服务器上分别创建两个虚拟机，配置为 2 个虚拟 CPU，512MB 虚拟内存以及 44GB 虚拟磁盘，并且在两个虚拟机服务器上分别安装 Apache 与 MySql，使其作为 web 服务器与数据库服务器。

实验环境二：两个配置相同的物理机，都拥有 4 个逻辑核 Intel CPU，4GB 内存与 500G 容量磁盘。在本实验中，在物理服务器 3 中配置两台虚拟机，在物理服务器 4 中

配置一台虚拟机。在有两台虚拟机的虚拟化环境中，虚拟机被作为 web 服务器与数据库服务器，在一台虚拟机的虚拟化环境中，虚拟机被作为数据库服务器，每个虚拟化服务器配置分别为 2 个虚拟 CPU，内存 512MB 与 44MB 磁盘。

在两组实验中，选择单独的物理服务器作为客户端，发出对虚拟服务器的访问请求，实验结构如图 20 所示。

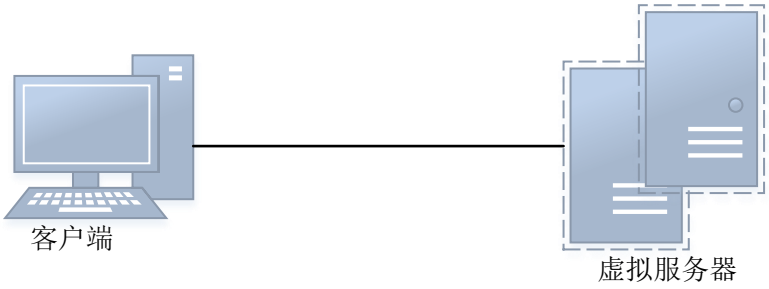

图 20　实验环境设置

本实验借助基准测试工具产生虚拟化服务器访问请求。Httperf 是一个 web 服务器性能测试工具，可以产生多种负载，支持 HTTP1.1 与 SSL 协议，其触发的负载特性与 SPECweb 客户端触发的负载特性相同[43]。在本实验中借助 Httperf 产生到 web 虚拟服务器的页面访问请求。SysBench 是常用的数据库性能基准测试工具，主要通过产生 OLTP 类型负载来评估系统性能，有简单，复杂以及非事务三种测试模式。在简单模式中，SysBench 产生简单的查询请求，在复杂模式中，SysBench 产生数据库事务操作，包括多种数据库查询，更新插入与删除 SQL 语句。非事务模式与简单模式类似，但是除查询类操作外，还包括其他数据库操作。在本研究中采用 SysBench 的复杂测试模式产生到虚拟化数据库的访问请求。

4.3.2 实验实施过程

在两组实验中，实验执行过程相同，由于在实验 2 中物理服务器 4 之上只有一台虚拟化数据库服务器，因此在客户端只由 SysBench 工具触发数据库操作请求，其它步骤不变。

在实验开始前，通过 Linux 中 top 与 iotop 命令收集客户虚拟机与宿主机初始负载信息，并通过 sys/block/目录下的文件获取磁盘初始信息，然后在客户端启动 Httperf 与 SysBench 工具，分别产生对于虚拟化 web 服务器与数据库服务器的访问请求，同时在客户虚拟机与宿主机中启动 top 与 iotop 命令，收集虚拟服务器处理请求期间产生的相关数据。

在实验中，web 虚拟化服务器中 Apache 采用 prefork 工作模式，初始时有 5 个 httpd

进程等待处理用户请求，每个进程只有一个线程。当线程池中的进程数少于MinSpareServer 时，Apache 会产生新的进程进入线程池。Httperf 是 web 服务器性能测试工具，可以产生多种 HTTP 负载并且能够获得端到端服务器性能数据。在本实验中，借助 Httperf 工具每秒产生 5 个对 web 服务器的连接，每个连接中包含一个请求，即 web 服务器中请求到达率为 5req/s。

由于 SysBench 需要对数据库的存储表进行操作，所以在实验正式开始前，首先使用 SysbBench 在虚拟化数据库中创建实验表并在其中生成 1000000 条记录。在实验中，mysql 数据库服务器开启 general query log 功能记录请求对数据表的所有操作，SysBench 在客户端创建 16 个线程对数据库进行测试，产生多个事务请求，每个事务请求包括多个 SQL 语句。

当客户端中 Httperf 与 SysBench 触发的请求全部结束后，停止客户虚拟机与宿主系统中的 top 与 iotop 命令，分类计算过程数据。

4.3.3 实验结果分析

基于 3.3 节给出的性能度量计算方法，对实验过程中获得的数据进行分析计算，得到虚拟化服务器性能度量值，对两种类型实验结果详细分析如下：

实验方案一

表 1　物理服务器 1 中虚拟化服务器性能度量值

虚拟服务器类别	R_{vcpu}	R_{vdisk}	R_{disk}	X
web 服务器	0.75ms	0.33ms	1.6ms	5req/s
数据库服务器	3.8ms	10ms	2.5ms	77.5req/s

表 2　物理服务器 2 中虚拟化服务器性能度量值

虚拟服务器类别	R_{vcpu}	R_{vdisk}	R_{disk}	X
web 服务器	3.65ms	12.3ms	3.7ms	5req/s
数据库服务器	4.97ms	26.3ms	32ms	54.7req/s

由图 21 与图 22 中两个物理服务器中虚拟化 web 服务器与虚拟化数据库服务器性能比较可以看出，物理服务器 1 中的虚拟化服务器性能要好于物理服务器 2 中的虚拟化服务器性能，与实际理论分析结论相符。并且基于基准测试工具 Httperf 和 SysBench 在客户端显示的结果，也可以得到相同的结论，因此该实验结果验证了本研究提出的性能评估模型的正确性。

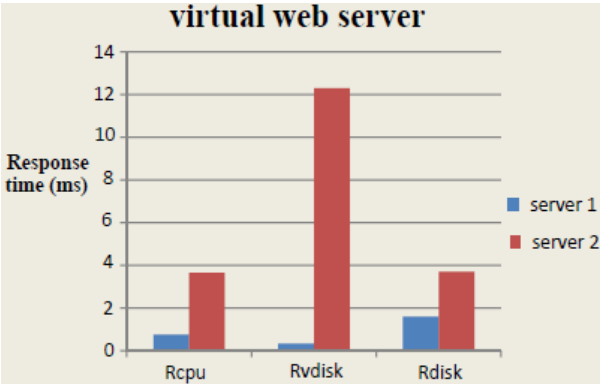

图 21　虚拟化 web 服务器性能比较

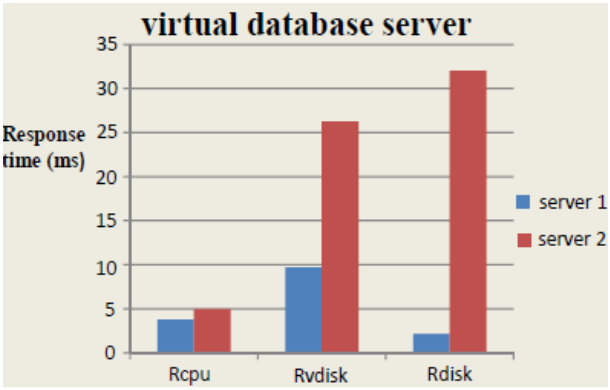

图 22　虚拟化数据库服务器性能比较

另一方面，通过对表 1 与表 2 中虚拟 web 服务器与虚拟数据库服务器中性能度量值进行比较可以得出，在两个虚拟化环境中，web 服务器性能都要好于数据库服务器性能，因此说明，在 KVM 虚拟化环境中 I/O 密集型操作性能消耗要大于 CPU 密集型操作性能消耗，证明虚拟化服务器中负载类型，以及 CPU 与 I/O 虚拟化实现方式对性能有重要影响。

实验方案二

在实验 2 中，通过性能评估模型得到的度量结果如表 3 所示。

表 3　虚拟化服务器性能度量值

物理服务器	R_{vcpu}	R_{vdisk}	R_{disk}	R	X
物理服务器 3	3.8ms	1ms	3.4ms	8.2ms	89req/s
物理服务器 4	4.04ms	1ms	4.8ms	9.84ms	97req/s

由提出的性能评估模型计算得到的性能数据可以得出，物理服务器 3 中虚拟化服务器响应时间略好于物理服务器 4 中虚拟化服务器响应时间，一般而言，系统响应时间越短，吞吐量越大，在物理服务器 3 中的虚拟化服务器性能度量值存在矛盾。对物理服务

器 3 中性能数据进一步分析，由 SysBench 基准测试工具在客户端得到的结果如图 23 与图 24 所示。

```
OLTP test statistics:
    queries performed:
        read:                            140014
        write:                           50002
        other:                           20001
        total:                           210017
    transactions:                        10000  (128.05 per sec.)
    deadlocks:                           1      (0.01 per sec.)
    read/write requests:                 190016 (2433.22 per sec.)
    other operations:                    20001  (256.12 per sec.)

Test execution summary:
    total time:                          78.0926s
    total number of events:              10000
    total time taken by event execution: 1248.8549
    per-request statistics:
         min:                                  32.14ms
         avg:                                 124.89ms
         max:                                 877.25ms
         approx.  95 percentile:              266.55ms
```

图 23　物理服务器 3 中虚拟化服务器工具测试结果

```
OLTP test statistics:
    queries performed:
        read:                            140084
        write:                           50030
        other:                           20012
        total:                           210126
    transactions:                        10006  (101.75 per sec.)
    deadlocks:                           0      (0.00 per sec.)
    read/write requests:                 190114 (1933.19 per sec.)
    other operations:                    20012  (203.49 per sec.)

Test execution summary:
    total time:                          98.3423s
    total number of events:              10006
    total time taken by event execution: 1572.0032
    per-request statistics:
         min:                                  40.60ms
         avg:                                 157.11ms
         max:                                 825.11ms
         approx.  95 percentile:              274.98ms
```

图 24　物理服务器 4 中虚拟化服务器工具测试结果

基准测试工具得到的虚拟化服务器每秒事务处理数，即图 23 与图 24 中 transactions

这一项显示的结果显示物理服务器 3 中虚拟化服务器性能要好于物理服务器 4 中虚拟化服务器性能。由于 SysBench 基准测试工具使用 16 个线程随机产生对虚拟化服务器的访问请求，因此两个虚拟化场景中，虚拟化服务器接收的请求数存在少量差别。由 SysBench 测试结果可以得到物理服务器 3 与物理服务器 4 中虚拟化数据库服务器每秒处理事务数分别为 128 与 102，即一个事务请求平均响应时间分别为 7.8ms 与 9.83ms，小于使用性能评估模型得到的计算结果。由于基准测试工具测试的为端到端性能，因此在理论层面，基准测试工具得到的请求响应时间应大于性能评估模型得到的性能结果。

两种性能评估方法得到的性能项度量偏差分别为：

响应时间：服务器 3：|8.2-7.8|/7.8*100%=5%，服务器 4：|9.84-9.83|/9.83*100%=1%。

吞吐量：服务器 3：|89-128|/128*100%=30%，服务器 4：|97-101|/101*100%=4%。

从以下几方面对两种方法的度量比较结果进行分析：

1) 实验结果中除物理服务器 3 的吞吐量与基准测试工具结果相差较大外，性能评估模型得到的其他性能度量值与端到端基准测试工具性能度量值相似且偏低，对除异常点外的性能数据进行分析。基于性能评估模型得到的性能度量结果与端到端基准测试工具结果相似的原因主要分为以下两类：

在本实验中，根据客户端中基准测试工具是否显示返回结果，来判断服务器端是否已处理完成请求，如果服务器处理请求结束，则停止服务器中相关 Linux 命令，在对数据进行处理时，根据获得的进程数据的变化情况，与相关访问日志文件中的时间信息，确定服务器中请求处理的结束时间，由于在实验中时间度量单位较小，因此很容易导致实验中收集的服务器的请求处理时间增加，导致响应时间偏大，吞吐量偏小。

在实验中，客户端与服务器位于同一局域网，并且不存在网络拥塞现象，因此请求在网络中传播时延与传输时延较小。虽然性能评估模型只针对虚拟化服务器对请求的处理性能，但因为网络时延较小，导致性能评估模型得到的结果与基准测试工具相似。

2) 针对两种方法得到的物理服务器 3 中虚拟化服务器吞吐量偏差较大这一问题，进行原因分析，主要有以下两个原因：

在物理服务器 3 中存在 web 与数据库两个虚拟化服务器，实验时有两个客户端分别产生到 web 服务器与数据库服务的请求，在实验中，当两个客户端都显示请求结果时，认为实验结束，停止收集数据，实验过程不规范，导致出现异常数据。

本研究提出的性能评估模型存在缺陷，导致得到的性能度量结果偏离正常范围。

为验证结果数据偏差过大的真正原因，针对原因 a 中说明的问题重新进行实验，在

实验过程中使用两台独立的客户端向物理服务器 3 中的两台虚拟服务器产生访问请求，根据性能评估模型对得到的实验数据进行计算，得到实验结果如表 4 所示。

表 4　服务器 3 中虚拟服务器性能度量结果(2)

物理服务器	R_{vcpu}	R_{vdisk}	R_{disk}	R	X
物理服务器 3	4.4ms	0.7ms	4.5ms	9.6ms	106req/s

由基准测试工具得到的结果如图 25 所示，基准测试工具显示虚拟化服务器响应时间为 9.1ms，吞吐量为 110req/s，两种方法的性能度量值偏差分别为：

响应时间偏差：|9.1-9.6|/9.1*100%=5.4%。

吞吐量偏差：|106-110|/110*100%=3.6%。

与第一次实验相比，吞吐量偏差率减小，响应时间偏差近似不变，因此可以确定实验方法导致第一次实验吞吐量偏差过大，而不是性能评估方法存在严重缺陷造成的。

```
OLTP test statistics:
    queries performed:
        read:                   140070
        write:                  50025
        other:                  20010
        total:                  210105
    transactions:               10005  (110.47 per sec.)
    deadlocks:                  0      (0.00 per sec.)
    read/write requests:        190095 (2098.85 per sec.)
    other operations:           20010  (220.93 per sec.)

Test execution summary:
    total time:                 90.5711s
    total number of events:     10005
    total time taken by event execution: 1448.0306
    per-request statistics:
        min:                    33.66ms
        avg:                    144.73ms
        max:                    1846.75ms
        approx.  95 percentile: 366.41ms
```

图 25　物理服务器 3 中虚拟化服务器工具测试结果(2)

通过以上两类实验分析，可以得出本研究提出的性能评估模型得到的性能度量结果基本正确，精确性在可接收范围内。

在两个虚拟化环境中，由性能评估模型与基准测试工具得到的性能评估结果显示，虽然物理服务器 3 中有两个虚拟机同时运行，物理服务器 4 中只有一个虚拟机运行，但

是位于物理服务器 4 中的虚拟化服务器性能没有明显优于位于物理服务器 3 中的虚拟化服务器，分析原因主要有以下两点：

1) 由于在两虚拟化环境中不存在 CPU 与内存资源过载，因此没有出现线程在内核频繁切换以及内存不足客户机被强制停止情况，同时 KVM 不同虚拟机之间具有较好的隔离性，所以虽然物理服务器 3 中虚拟化服务器数量大于物理服务器 4，但物理服务器 4 中虚拟化服务器性能度量值并没有优于物理服务器 3 中虚拟化服务器。

2) 由于在物理服务器 3 中另一台虚拟化服务器为 web 服务器，在本实验中处理请求为非 I/O 密集型，因此不会产生较多磁盘 I/O 操作，因此对虚拟化数据库服务器性能影响较小。

4.4 本章小结

本章主要验证了第三章提出的虚拟化服务器性能评估模型的正确性。由于无法将性能评估模型得到的性能评估结果与基准测试工具得到的结果直接比较，本章通过设置对比实验，将性能评估模型得到的结论与已有结论进行比较，来验证模型的正确性。本章选择了虚拟化服务器性能影响因素中的 CPU 调度与虚拟机数量这两个因素来设计实验，并详细说明了实验设计原理；然后介绍了两类实验的环境配置及实验实施过程；最后详细分析了两类实验的实验结果。实验类型 1 说明了性能评估模型的正确性，实验类型 2 说明该模型得到的度量结果精确性可以进一步提高，但偏差在可接收范围之内，性能评估模型满足可用性。

第五章 KVM 虚拟化服务器性能评估模型应用

在应用层面对虚拟化服务器进行性能评估的目的是，评价运行在虚拟服务器上的应用是否可以提供满足 SLA 要求的服务，从而保证用户良好的应用体验。另一方面当现有虚拟化服务器性能不能满足用户要求时，应该如何对相关配置进行改进，以及如何综合判断对某一方面的改进要优于对另一方面的修改，或者当虚拟服务器有多种配置方案时，如何根据具体应用场景的要求选择合适的配置方案，也是服务提供商必须解决的问题。因此本章通过使用多准则决策中的多属性决策法，基于第四章提出的性能评估模型，提出了一种虚拟机配置决策方法，并通过实例详细说明了在 I/O 密集型应用环境中如何借助性能评估模型进行虚拟机配置方案决策。

5.1 KVM 虚拟化服务器配置原则

在 KVM 虚拟化系统中，一台 Linux 物理服务器上可以配置的虚拟化服务器数量以及各虚拟化服务器中配置的虚拟 CPU 数量与虚拟内存容量没有明确的规则。KVM 允许客户虚拟机过载使用物理资源[5]。CPU 的过载使用是指多个客户机使用的虚拟 CPU 总数超过实际拥有的物理 CPU 数。但是一般情况下，不允许某一个客户虚拟机的虚拟 CPU 数量超过物理机上存在的 CPU 数量，否则可能导致系统性能下降。KVM 中内存过载使用是指分配给客户机的内存总数大于实际可用的物理内存总数。虽然 KVM 中允许客户虚拟机过载使用内存，但是过多的内存过载使用可能导致系统稳定性降低，因此虚拟机的不同配置对性能有不同影响。

Linux 操作系统供应商 Red Hat 给出了基于 virt-manager 的虚拟机配置指导[44]。virt-manager 针对选择的客户操作系统的版本与类型提供了不同的配置方案。在 virt-manager 中可以设置虚拟 CPU 数量，当 CPU 过载使用时会产生警告，并且可以在配置虚拟 CPU 时选择需要的 CPU 模型，选定 CPU 模型后，就会展现该 CPU 模型的特征以及指令集，在虚拟机配置时推荐选择主机 CPU 模型。在非一致性内存访问（Non-Uniform Memory Access, NUMA）的 CPU 硬件体系架构中，还可以设置 CPU 拓扑，明确 Socket, core 与线程数。当一个核对应一个线程时得到的性能结果较好，频繁的线程调度可能导致资源竞争，影响性能。当客户虚拟机拥有的虚拟 CPU 数少于 NUMA 单个节点时，可以设置 CPU pinning 使虚拟机只能使用所在节点的处理器，提高性能，如图 26 所示。

在 virt-manager 中可以通过配置缓存模式，IO 模式与 IO 优化来配置虚拟机虚拟磁

盘。不同的应用场景下，有不同的最优配置方案，如在大量 I/O 需求环境中，cache 应配置为 none，此时客户虚拟机的 I/O 不会被缓存在宿主机中，而是可能被保存在一个回写缓存中。当 cache 设置为 writeback 时，客户机 I/O 将会被缓存在主机中。在虚拟化系统中，当多个虚拟机同时运行时，过度的使用磁盘 I/O 会降低系统性能，在 KVM 中可以使用磁盘 I/O 节流来设置客户虚拟机磁盘 I/O 请求限制，阻止一个虚拟机过度使用共享资源影响其他虚拟机性能。

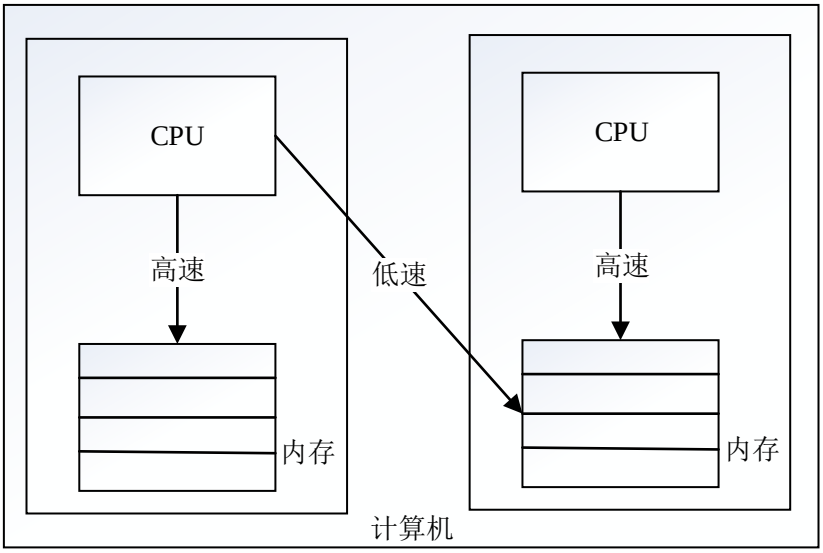

图 26　NUMA 架构中 CPU 调度

KVM 虚拟环境中内存，分配应该遵循以下三点：不要为虚拟机分配超出其需要的过多的内存；在 NUMA 处理器架构中，与虚拟 CPU 配置类似，一个虚拟机分配一个单一的 NUMA 节点；开启透明大页（Transparent Huge Pages, THP），使空闲内存被用作缓存，可以提高系统性能。

由以上分析可以看出，在 KVM 中，需要结合实际应用环境特点对虚拟化服务器配置进行分析实验，确定在现有物理资源情况下虚拟机最优配置方案。

5.2 应用性能约束分析

在不同应用环境中，服务提供商可能对性能度量指标侧重点不同。在 I/O 操作密集的数据库应用中，相对于单个请求响应时间的优化，服务提供商更关注于单位时间内，应用整体的吞吐量优化，因此仅仅使用性能评估模型计算得到虚拟化服务器单个请求的平均响应时间以及单位时间的吞吐量，无法直接比较两个虚拟化环境优劣，需要对两个性能度量指标结合用户偏好进行综合分析评估。并且在 web 应用环境中，页面负载类型分为静态页面和动态页面两种，静态页面负载只涉及简单的服务器与客户端的交互，用于显示网页，动态页面负载涉及客户端与服务器的交互，需要对应用程序进行解释和执

行，因此动态网页负载将会使用更多的服务器资源，所以在动态网页负载中对响应时间的关注要高于吞吐量[45]，与数据库应用类似，无法直接使用响应时间与吞吐量两个指标独立比较虚拟化服务器性能优劣。

有研究者调研了大量用户对于其常用 web 网站的访问经历，提出了响应时间的 3 个维度 [46]，具体如下所示：

1) 当请求的响应时间在 0.1 秒之内时，可以认为是瞬时响应，此时用户感觉其直接操作应用；

2) 当请求的响应时间大于 0.1 秒小于 1 秒时，可以保证用户访问的流畅性，此时用户可以感觉到延时，但是用户仍然可以自由访问；

3) 当请求时间为 1 秒到 10 秒时，用户可以明确感觉到延迟，可能会离开该网站，会给用户造成产生不愉快的访问经历。

由以上 3 维度限制可以看出，当 web 服务器对请求处理时间较短时，如在 0.1 秒之内，响应时间的细微差异不会对用户体验产生影响，并且不会侵犯 SLA 协议中的 QoS 要求，因此在虚拟化环境中虚拟化服务器配置时，可以在保证 QoS 要求的下，根据虚拟化服务器集群的要求，合理分配物理资源，而并不是响应时间越高越好。

通过对 3.5 节中实验数据分析可以发现，I/O 密集型操作性能要低于 CPU 密集型性能，因此在现有物理设备情况下，如何配置虚拟化服务器，使其既能保证资源充分利用，也可以保证操作环境中用户访问体验，是服务提供商面临的重要问题。本章将以 I/O 密集型操作环境中，虚拟化服务器配置选择为例，说明借助本研究提出的虚拟化服务器性能评估模型如何评估不同配置方案的优劣。

5.3 决策分析模型研究

多准则决策问题是指涉及多个指标的决策问题，其中准则是决策现象有效性的基础，有目标和属性两种基本的表现形式。目标是决策者对决策事务的期望，属性则是决策事物的某种特性。决策者需要根据实际情况选择定量或定性的多种准则，因此该决策方法，依赖于决策者的偏好。多准则决策模型又进一步分为多目标决策与多属性决策两种，在多目标决策中，没有预先可供选择的方案，而是对一组目标函数基于约束集合进行优化从而确定方案。在多属性决策中，已知可供选择的方案，通过一组属性集合对方案进行评估，选择最优的方案，并且一般该方法中用到的属性很难定量化。

在已知虚拟机不同配置方案的情况下，根据不同评价标准确定最优选择方案，应该采用多属性决策方法。目前针对多属性决策问题，研究者提出了多种方案评估方法，如

层次分析法，权重加和法以及产品加权方法等，其中层次分析法是目前广泛应用的将定性与定量度量相结合的多属性决策方法。

5.3.1 层次分析法简介

层次分析方法是美国运筹学家 ThomasL. Satty 在 70 年代提出的决策分析方法，层次分析法（Analytic Hierarchy Process，AHP）将复杂的多目标决策问题中的各种因素按支配关系划分为层次结构，然后通过两两比较的方式确定各层因素的相对重要性，最后利用相关数学方法，综合决策者的判断，确定不同方案的优先级权重[47]。将定性指标与定量指标相结合是层次分析法的一个大优点，使用层次分析法进行决策分析的步骤如下所示：

1) 分析系统中目标与各因素之间的影响关系，建立指标层次结构；
2) 对同一层次的各因素对上一层次因素的影响程度进行两两比较，构造判断矩阵；
3) 根据判断矩阵计算各层因素的相对权重，并进行一致性检验；
4) 计算各层因素对系统目标的合成权重，进行一致性检验，得到方案优先级。

当使用层次分析法进行决策分析时，首先需要将问题逐步分解，建立层次结构模型。在该模型中，最高层一般为问题的目标或理想结果，被称为目标层，模型的中间层一般为影响目标达成的因素，即为实现目标涉及的中间环节，它可以包含若干层，也被称为准则层，模型的底层一般表示为实现目标可供选择的决策方案，因此该层一般被称为方案层。当层次模型建立后，需要根据每层因素相对其上一层元素的相对重要程度，建立判断矩阵，基于判断矩阵使用特征根计算方法，得到各层的层次单排序权重向量，并进行一致性检验，判断每层中确定的各因素相对重要性是否存在矛盾，当得到各层的层次单排序权重后，基于单排序权重采用自上而下的方法计算层次总排序权重，最后得到各方案对目标的相对重要性，从而帮助用户制定决策。

5.3.2 层次分析法求解实例

图 27 展现了层次分析法建立的指标层次结构，自上往下分别为目标层，准则层与方案层，其中目标层为分析问题的预定目标，中间多个准则层为决策过程中必须考虑的多种因素，方案层表示为实现目标可以选择的解决方案。在建立指标层次结构后，决策涉及的各因素之间层级关系被确定，下面以图 27 的中间准则层为例说明层次分析法计算方法。

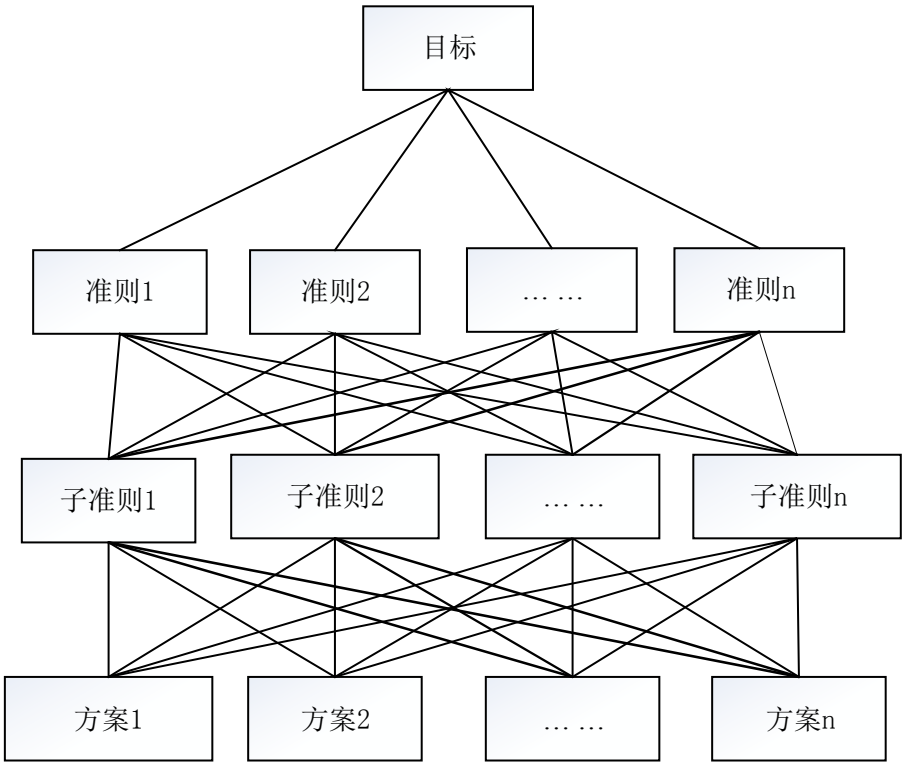

图 27　层次分析结构图

1)　构造两两比较判断矩阵

根据下一层元素对上一层元素的影响程度，建立判断矩阵。在图 27 中对准则 1 有影响的下层因素分别为子准则 1，子准则 2，一直到子准则 n。根据下层各子准则对准则 1 的相对重要性，赋值相应的权重，建立子准则层判断矩阵，当各子准则可以直接定量表示时，相应权重可以直接使用定量值确定，对于定性指标，可以参照 AHP 算法的 1-9 标度含义表确定，如表 5 所示。

表 5　AHP 算法的 1-9 标度含义

标度	含义
1	表示两个元素相比，具有同样重要性
3	表示两个元素相比，前者比后者稍微重要
5	表示两个元素相比，前者比后者明显重要
7	表示两个元素相比，前者比后者强烈重要
9	表示两个元素相比，前者比后者极端重要
2,4,6,8	介于相邻判断的两个标度之间，取中值
倒数	若元素 i 与元素 j 的重要性之比为a_{ij}，那么元素 j 与元素 i 重要性之比为$a_{ji}=\frac{1}{a_{ij}}$

由此对于准则 1，可以得到 n 个被比较元素构成的一个 n 阶方阵，如(5.1)所示。

$$A = (a_{ij})_{nxn} \tag{5.1}$$

其中a_{ij}即为子准则 i 与子准则 j 相对于准则 1 的重要性标度，根据判断矩阵构造方法可以得出其具有以下性质：a_{ij}>0，a_{ji}=$\frac{1}{a_{ij}}$，a_{ii}=1，即判断矩阵为互反矩阵。

2) 层次单排序及一致性检验

判断矩阵建立后，需要根据判断矩阵计算子准则层对于准则 1 的相对权重，特征根法是目前应用比较广泛的一种权重计算方法，使用特征根法进行权重计算，就是求解判断矩阵的最大特征值对应的归一化特征向量的过程。对上述矩阵 A 使用特征根方法进行计算，首先求解 A 的最大特征值λ_{max}，然后对得到的对应特征向量进行归一化，即向量 W=$(w_{1,}w_{2,\cdots,}\ w_n)^T$满足公式(5.2)。

$$\sum_{k=1}^{n} w_k = 1 \tag{5.2}$$

为防止构造判断矩阵时出现重要性前后不一致情况，需要对得到的权重向量进行一致性检验，首先定义一致性指标 CI，计算公式如(5.3)所示。

$$CI= (\lambda_{max} - n)/(n-1) \tag{5.3}$$

通过查表可以得到与矩阵对应的平均一致性指标 RI，计算一致性比例 CR，见公式(5.4)。

$$CR=CI/RI \tag{5.4}$$

一般认为，当 CR<0.1 时，权重向量通过一致性检验，当 CR≥0.1 时应该对判断矩阵进行修正，并且对于一阶与二阶矩阵而言，各因素的相对重要性总是一致的，此时 CR=0。

3) 层次总排序及其一致性检验

通过上面的计算可以得到下层元素对其上层中某个元素的权重向量，层次分析法的目的是得到底层因素对总目标的排序权重，因此需要根据得到的权重向量自上而下进行层次总排序，并进行一致性检验[42]。

假定准则层各因素对总目标的层次单排序为$w^{(2)}$=$(w_1^{(2)}, w_2^{(2)}, \cdots, w_n^{(2)})^T$，子准则层对上层元素准则 1 的层次单排序为$p_1^{(3)}$=$(p_{11}^{(3)}, p_{21}^{(3)}, \cdots, p_{n1}^{(3)})^T$，则子准则层对上层的各元素权重可以表示为：$p^{(3)}$=$(p_1^{(3)}, p_2^{(3)}, \cdots, p_n^{(3)})^T$，由此可以得到子准则层对总目标的层次总排序为$w^{(3)}$=$p^{(3)}w^{(2)}$，以此类推，最后可以得到方案层对于总目标的层次排序。

同样地需要对层次总排序自上而下进行一致性检验，假设子准则层对准则层元素 1

的层次单排序一致性指标为$CI_1^{(3)}$，随机一致性指标为$RI_1^{(3)}$，一致性比例为$CR_1^{(3)}$，则综合指标$CI^{(3)}$，$RI^{(3)}$与$CR^{(3)}$计算公式如公式(5.5)至(5.7)所示。

$$CI^{(3)}=(CI_1^{(3)},CI_2^{(3)},\cdots,CI_n^{(3)})w^{(2)} \tag{5.5}$$

$$RI^{(3)}=(RI_1^{(3)},RI_2^{(3)},\cdots,RI_n^{(3)})w^{(2)} \tag{5.6}$$

$$CR^{(3)}=CI^{(3)}/RI^{(3)} \tag{5.7}$$

由此可以得到第 3 层一致性比例，如果$CR^{(3)}$ ≤0.1 则认为第 3 层水平以上的所有判断矩阵具有整体满意一致性，根据上计算方法以此类推可以获得方案层一致性检验值，如果层次总排序通过一致性检验，则根据方案层的层次总排序向量做出最后决策。

5.4 建立虚拟化服务器配置决策模型

根据实际应用场景，进行虚拟化服务器配置方案决策时，虚拟化服务器性能是首要考虑的因素，采用决策分析法对多种虚拟机配置方案性能进行比较评估。由 3.1 与 3.2 节可以得到响应时间与吞吐量是虚拟化服务器性能主要度量指标，并且在虚拟化系统中响应时间与吞吐量主要由 CPU，Disk I/O 以及 vDisk I/O 三部分性能度量组成，假设存在三种虚拟机配置方案，可以建立图 28 所示的层次分析结构图。

响应时间与吞吐量为层次结构目标级度量指标，CPU，Disk I/O 与 vDisk I/O 性能是影响目标度量的主要因素，解决方案层中虚拟机配置（Virtual Machine Configuration，VMC）是现有虚拟化系统配置方案，本决策模型的目的是使用准则层的各属性对方案层各方案进行评估，最后得到各方案的优先级。由于借助本研究提出的性能评估模型，在不同的配置环境中可以得到各方案对应准则层的各因素的实际度量值，因此本章结合多准则决策中的层次分析法与权重加和法建立虚拟机配置方案决策模型。

权重加和法是最常用的多准则决策方法，主要针对于单维度问题。如果有 M 种选择方案，N 种选择准则，则根据多准则决策方法，最好的选择方案应该满足公式(5.8)[48]。

$$A_{wsm}=\text{Max}\sum_i^j a_{ij}\,w_j \qquad \text{i=1,2,3,}\cdots\text{M} \tag{5.8}$$

其中A_{wsm}是最优方案对应的优先级得分，a_{ij}是第 i 个可选方案在准则第 j 个准则中的实际值，w_j是第 j 个准则对应的权重。基于以上分析，本章建立的决策模型具体计算步骤如下：

首先，根据实际应用环境，确定 SLA 协议关注的重点，判断响应时间与吞吐量的相对重要性，得到判断矩阵 A。

随后，判断准则层各因素相对于目标层各因素的重要性，分别得到针对响应时间与

吞吐量的判断矩阵B_r与B_t，通过性能评估模型可以得到各方案对应的准则层各因素的定量判断矩阵$\mathrm{C_r}$与C_t。

然后，使用矩阵特征值方法计算判断矩阵 A、B_r与B_t的最大特征值以及最大特征向量，即可以得到每层各元素相对于上层各元素的相对权重，并进行一致性检验。

其次，由于定量判断矩阵C_r与C_t量纲不同，因此需要对矩阵进行无量纲化，目前常用的无量纲化方法包括：线性比例法，归一化处理法，向量规范化等。归一化处理法是常用的简单的无量纲转化方法，因此本研究采用归一化对判断矩阵C_r与C_t进行无量纲化处理，并且由于在虚拟化系统中请求响应时间越短，表示虚拟化服务器性能可能越好，所以在对均值C_r进行归一化处理之前，先对其属性值取倒数，假设C_r记为$(x_{ij})_{3x3}$，C_t记为$(y_{ij})_{3x3}$，归一化方法见公式(5.9)至(5.11)。

$$x_{ij} = \frac{1}{x_{ij}} \tag{5.9}$$

$$\mathrm{z_{ij}} = \frac{x_{ij}}{\sum_{i=1}^{3} x_{ij}} \tag{5.10}$$

$$\mathrm{r_{ij}} = \frac{y_{ij}}{\sum_{i=1}^{3} y_{ij}} \tag{5.11}$$

则矩阵$C_r{}'$=$(z_{ij})_{3x3}$，$C_t{}'$=$(r_{ij})_{3x3}$即为归一化后矩阵。

最后，使用权重加和法，计算归一化后的矩阵$C_r{}'$和$C_t{}'$与上层权重向量的乘积，最终得到各配置方案相对于总目标的优先级。

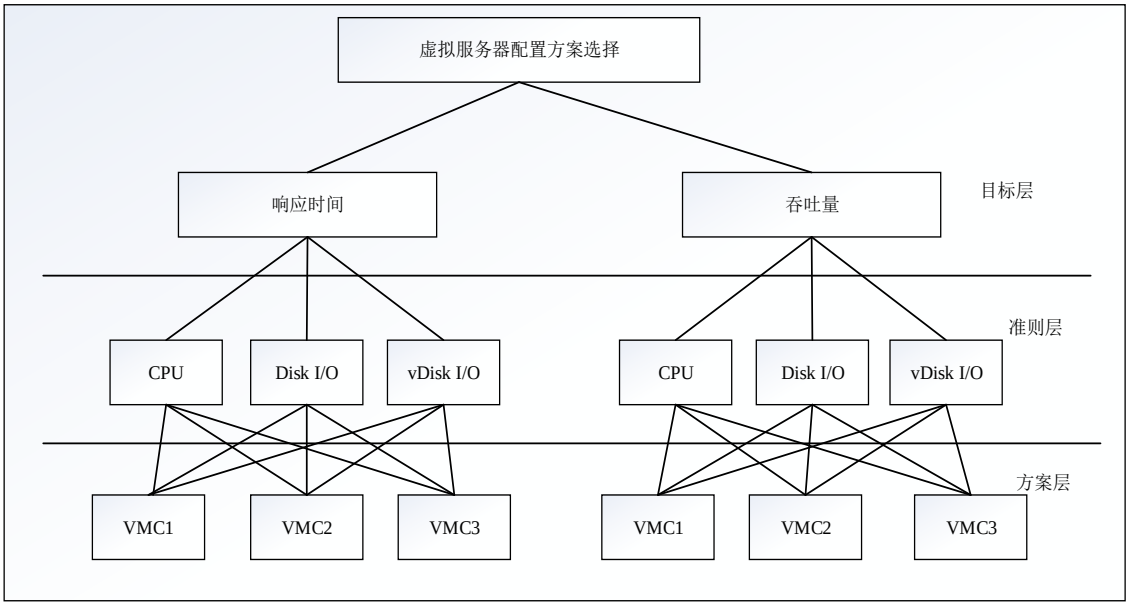

图 28　配置方案决策层次结构

5.5 实例说明

本节以 I/O 密集型应用环境中，负载类型为 OLTP 的虚拟数据库服务器的配置方案选择为例，详细说明了借助本研究提出的性能评估模型，使用决策方法对三种不同虚拟化配置方案进行优先级排序的过程。

5.5.1 实验基本原则

服务器主要由中央处理器，内存与 I/O 设备构成。CPU 的架构与核数，以及其高速缓存的大小，对 CPU 处理速度有重要影响。内存的性能主要与内存容量，内存带宽以及内存读写速度有关，磁盘 I/O 设备性能主要涉及磁盘转速，接口技术等。当服务器中其它设备配置相同的情况下，某一设备配置越好则服务器性能越好。同样，因为在虚拟化服务器中所有操作，最终映射为对物理资源的操作，因此当物理服务器中虚拟化服务器配置相同时，物理服务器配置越好，则虚拟化服务器性能越好。

虚拟服务器与物理服务器类似，在 KVM 虚拟化中，虚拟 CPU 作为标准的线程，被 Linux 调度。当虚拟机分配的 CPU 总数小于物理 CPU 数时，不存在资源的过度竞争，某个虚拟机分配的虚拟 CPU 数量越多，则虚拟化系统中对应线程越多，虚拟机中请求处理速度越快。

因为在 KVM 中，借助 I/O 半虚拟化工具 virtio，虚拟机可以将 I/O 请求存入共享内存，virtio 后端驱动读取共享内存中相关信息，调用物理驱动设备对实际物理磁盘进行操作。因此在 KVM 中，虚拟机对虚拟磁盘的操作，即为对虚拟内存的操作，因此在内存容量以及读写速度，会对虚拟 I/O 性能产生影响。在 KVM 中虚拟机是普通的 Linux 进程，遵循常驻内存程序的通用法则。一般在 KVM 中虚拟机启动时仅设置指向虚拟内存的指针。当虚拟机处于运行状态时，VMM 会为其分配常驻内存，常驻内存等于虚拟机正在使用的内存大小。并且在 KVM 中虚拟机内存是动态变化的，KVM 引入了 virtio_ballon 在客户机运行时动态调整虚拟机占用的宿主内存资源。Balloning 技术在客户机内存中引入了气球的概念，气球中的内存可以供宿主机使用。当宿主机内存使用紧张时，可以请求客户机回收部分已分配给客户机的内存，客户机会释放空闲的内存，并且如果客户机空闲内存不足，可能还会将使用中的内存将其交换到客户机交换分区中，增加空闲内存，从而使可释放内存空间增加，增大宿主机可用内存。反之，当客户机中内存不足时，也可以压缩其中的内存气球，释放其中内存，从而让客户机使用更多的内存，这也是 KVM 中内存过载使用的原理[5]。由此可以得出在 KVM 中创建虚拟机时分配的内存大小与虚拟机在实际运行时可用的内存大小可能不一致，因此无法直观判断虚

拟机中内存分配大小对虚拟化服务器性能影响。

由以上分析可以得出以下结论：

1) 在某些单一因素中，如当虚拟机配置相同时，可以由理论分析得到物理服务器配置越好，其上虚拟化服务器性能应该越好。在虚拟机中某些虚拟资源配置不同时，如虚虚拟 CPU 配置不同，并且虚拟环境中不存在 CPU 过载情况，当其他配置相同时虚拟机中虚拟 CPU 数量越多，则虚拟服务器性能越好。当虚拟资源为虚拟内存时，则无法根据虚拟内存大小来直接判断虚拟机性能。

2) 当虚拟化环境中多种资源配置不同时，如配置相同的物理服务器上创建内存与 CPU 都不同虚拟机时，如何判断虚拟机性能，以及当物理服务器配置与虚拟机中虚拟资源配置都不相同时，如何结合实际应用环境中性能侧重点来选择虚拟机配置方案是用户关注重点。

在 I/O 密集型操作环境中，如何针对上述问题进行分析决策，使组织中资源既可以得到充分利用，也可以满足用户需求，这是本实例设计的主要目的。本实验首先在两台配置不同的物理服务器上，创建虚拟 CPU 数以及虚拟内存数小于物理资源的配置相同的虚拟机，并且同时选择与上述两台服务器中一个配置相同的物理服务器在其上创建配置不同的虚拟机，然后，使用性能评估模型，计算以上虚拟机的性能度量指标，再使用决策模型确定各种配置方案的优先级，最后，结合应用环境中实际需求，给出虚拟机配置方案决策建议。

5.5.2 实验环境配置

实验中使用的三台物理机配置及其上虚拟机配置分别为：

物理服务器 1：一个 2 核 CPU，共 4 个线程，CPU 主频为 2.67Hz，4GB 内存，物理磁盘 500GB，其上虚拟机配置为：2 个虚拟 CPU，512MB 虚拟内存。

物理服务器 2：一个 2 核 CPU，共 4 个线程，CPU 主频为 3.3GHZ，8GB 内存，物理磁盘 1000GB，其上虚拟机配置为：2 个虚拟 CPU，512MB 虚拟内存。

物理服务器 3：一个 2 核 CPU，共 4 个线程，CPU 主频 3.2GHZ，8GB 内存，物理磁盘 1000GB，其上虚拟机配置为：1 个虚拟 CPU，1024MB 内存。

CPU 的主频为 CPU 内核工作时钟频率，主频越高，CPU 的速度越快，但 CPU 主频并不是影响系统性能的唯一因素，CPU 每个时钟频率能够处理的指令数是 CPU 性能的重要因素，因此只由物理服务器 2 与物理服务器 3 的 CPU 主频不能直接判断系统性能。同时物理服务器 2 与物理服务器 3 的 CPU 二级缓存，内存读写速度以及物理磁盘转速

等近似相等，因此可以视物理服务器 2 与物理服务 3 配置相同，物理服务器 1 内存容量小于物理服务器 2，同时其内存读写速度也低于服务器 2。虚拟化系统其它配置与 3.5 节相同，使用 CentOS 7 作为宿主机与虚拟机机操作系统，安装 virt-manager 作为虚拟机管理工具，使用 SysBench 作为请求触发工具。

5.5.3 实验实施过程

本节实验过程与 3.5 节相同，首先在虚拟化服务器中借助 Linux 命令与 Linux 相关系统文件收集三个虚拟化系统初始负载信息，然后在独立客户端触发负载，使用 SysBench 产生到虚拟化服务器请求，同时启动 Linux 中 top 与 iotop 等命令收集过程数据，最后，当请求结束时，停止收集数据命令，并使用性能评估模型计算各虚拟化系统性能数据。

根据 4.3 节建立的决策层次模型，首先使用性能评估模型由实验数据计算得到的虚拟化服务器各主要组件的定量性能数据，计算各方案的优先级，由于性能度量指标响应时间与吞吐量度量单位不同，因此需要在计算时对获得的数据进行无量纲化处理。

5.5.4 实验数据分析

根据层次分析法，结合建立的决策层次结构，使用获得的实验数据计算三种配置方案决策优先级，具体步骤如下：

首先，建立判断矩阵。不同应用的性能侧重点不同，因此响应时间与吞吐量对于虚拟化服务器配置选择影响不同，在以 OLTP 负载为主的 I/O 密集型操作环境中，一般主要使用吞吐量，即系统在单位时间内处理事务的个数，来反映 OLTP 应用系统性能，响应时间通常用来度量系统完成特定事务所需的时间，一般用来反映 OLAP 应用系统性能，因此，结合 AHP 算法的标度表得到目标层中响应时间与吞吐量两两比较的判断矩阵，见 (5.12)。

$$OS=\begin{bmatrix} 1 & 3 \\ 1/3 & 1 \end{bmatrix} \tag{5.12}$$

同样地可以得到准则层各影响因素对于响应时间与吞吐量的判断矩阵，如矩阵(5.13)与(5.14)所示，CR 表示标准层相对于响应时间的判断矩阵，CT 表示标准层相对于吞吐量的判断矩阵。

$$CR=\begin{bmatrix} 1 & 1/6 & 1/3 \\ 6 & 1 & 4 \\ 3 & 1/4 & 1 \end{bmatrix} \tag{5.13}$$

$$CT=\begin{bmatrix}1 & 1/5 & 1/2\\ 5 & 1 & 4\\ 2 & 1/4 & 1\end{bmatrix} \tag{5.14}$$

借助 KVM 虚拟化服务器性能评估模型可以得到三种虚拟机配置方案下，虚拟服务器各部分性能数据，因此可以得到方案层对标准层的定量判断矩阵，见(5.15)与(5.16)。

$$AR=\begin{bmatrix}4.04 & 1.01 & 4.89\\ 3.82 & 1.2 & 4.32\\ 3.83 & 4.9 & 6.4\end{bmatrix} \tag{5.15}$$

$$AT=\begin{bmatrix}168 & 144 & 169\\ 263 & 149 & 199\\ 182 & 115 & 125\end{bmatrix} \tag{5.16}$$

其次，层次单排序及一致性检验。根据得到的判断矩阵，使用特征根方法，计算下层各因素相对于上层因素的权重向量，结果见(5.17)至(5.19)。

$$w_{OS}=[0.75 \quad 0.25]^{T} \tag{5.17}$$

$$w_{CR}=[0.09 \quad 0.69 \quad 0.22]^{T} \tag{5.18}$$

$$w_{CT}=[0.12 \quad 0.68 \quad 0.20]^{T} \tag{5.19}$$

对层次单排序向量进行一致性检验，查表可以得到矩阵阶数对应的 RI 值，表 6 给出了各阶矩阵计算 1000 次得到的平均随机一致性指标：

表 6　RI 指标值

矩阵阶数	1	2	3	4	5	6	7
RI	0	0	0.52	0.89	1.12	1.26	1.36

计算得到CR_{OS}=0，CR_{CR}=0.05，CR_{CT}=0.0246，因为 CR<0.1 所有，所以权重向量通过一致性检验。

为保证数据可比性，对判断矩阵 AR 与 AT 进行无量纲化处理，并且由于组件处理请求响应时间越长，性能越低，因此在对 AR 数据进行无量纲处理之前，对其中各值取倒数，保证比较的一致性，结果如(5.21)与(5.22)所示。

$$AR^{“}=\begin{bmatrix}0.248 & 0.99 & 0.205\\ 0.262 & 0.833 & 0.231\\ 0.261 & 0.204 & 0.156\end{bmatrix} \tag{5.20}$$

$$AR^{‘}=\begin{bmatrix}0.32 & 0.49 & 0.35\\ 0.34 & 0.41 & 0.39\\ 0.34 & 0.1 & 0.26\end{bmatrix} \tag{5.21}$$

$$AT^{‘}=\begin{bmatrix}0.27 & 0.35 & 0.34\\ 0.43 & 0.37 & 0.40\\ 0.30 & 0.28 & 0.25\end{bmatrix} \tag{5.22}$$

其中$AR^{“}$为一致性处理后得到的性能矩阵，$AR^{‘}$与$AT^{‘}$为归一化的标准矩阵。

根据权重加和法计算目标层各指标对最终决策的影响权重，结果见(5.25)。

$$w_R=AR^{'} * w_{CR}=\begin{bmatrix}0.32 & 0.49 & 0.35\\ 0.34 & 0.41 & 0.39\\ 0.34 & 0.1 & 0.26\end{bmatrix}\begin{bmatrix}0.09\\ 0.69\\ 0.22\end{bmatrix}=\begin{bmatrix}0.4439\\ 0.3993\\ 0.1568\end{bmatrix} \tag{5.23}$$

$$w_T=AT^{'} * w_{CT}=\begin{bmatrix}0.27 & 0.35 & 0.34\\ 0.43 & 0.37 & 0.40\\ 0.30 & 0.28 & 0.25\end{bmatrix}\begin{bmatrix}0.12\\ 0.68\\ 0.20\end{bmatrix}=\begin{bmatrix}0.3384\\ 0.3832\\ 0.2764\end{bmatrix} \tag{5.24}$$

$$\begin{aligned} w&=(w_R, w_T)* w_{OS}\\ &=\begin{bmatrix}0.4439 & 0.3384\\ 0.3993 & 0.3832\\ 0.1568 & 0.2764\end{bmatrix}\begin{bmatrix}0.75\\ 0.25\end{bmatrix}\\ &=[0.4175 \quad 0.3953 \quad 0.1867]^T \end{aligned} \tag{5.25}$$

由计算结果可以得出 VMC3< VMC2< VMC1，在实验设计的应用场景中虚拟化服务器配置方案 1 的优先级最高，并且方案 1 与方案 2 优先级接近，因此可以得出，I/O 密集型操作中，在侧重虚拟化系统吞吐量情况下，当不存在资源过载使用时，物理内存大小对虚拟化服务器配置方案选择没有显著影响，并且在该实验环境中，虚拟 CPU 数量对虚拟机性能有重要影响，在对虚拟机进行配置方案选择时，需要重点考虑该因素。

同样，由基准测试工具可以得到三种虚拟机配置方案中，虚拟化服务器性能测试结果，见图 29，图 30 与图 31。

由基准测试工具得到的数据，可以得到三种配置方案的吞吐量与 95%请求的被处理时所用的时间，如表 7 所示。虽然配置方案 2 的吞吐量最高，事务的平均响应时间最高，但其大多数请求的完成时间要小于配置方案 1，因此无法直接基于平均性能度量值判断配置方案的优劣，进一步验证了基于性能评估模型的配置决策方法的意义。

表 7　不同配置方案的虚拟化服务器性能度量数据

配置方案 / 度量项	VMC1	VMC2	VMC3
每秒钟事务数	101.75	104.12	70.92
95%请求完成时间	274.98ms	308.47ms	498.73ms

由于本实验中所产生的负载请求总量较少，并且单位时间内请求到达率较低，所以虚拟化服务器处于低负载状态下，虚拟 CPU 以及虚拟内存数量充足，不存在虚拟机内存不足情况时内存合并等内存优化处理，所以虚拟机中虚拟内存大小变化没有对虚拟机性能产生重要影响。当虚拟机负载逐渐增大，直至其最大负载时，由于虚拟化系统中资源竞争激烈，虚拟机性能消耗增长迅速，因此性能降低较大，因此本实验得到的虚拟化

服务器配置方案优先级可能会改变，所以在实际应用中需要结合实际需求对虚拟化服务器配置需求进行评估，此处虚拟机最大负载状况是指，当请求到达后，响应时间达到用户可以忍受的边界。

```
OLTP test statistics:
    queries performed:
        read:                            140084
        write:                           50030
        other:                           20012
        total:                           210126
    transactions:                        10006  (101.75 per sec.)
    deadlocks:                           0      (0.00 per sec.)
    read/write requests:                 190114 (1933.19 per sec.)
    other operations:                    20012  (203.49 per sec.)

Test execution summary:
    total time:                          98.3423s
    total number of events:              10006
    total time taken by event execution: 1572.0032
    per-request statistics:
         min:                                  40.60ms
         avg:                                 157.11ms
         max:                                 825.11ms
         approx.  95 percentile:              274.98ms
```

图 29　物理服务器 1 中虚拟化服务器工具测试结果

```
OLTP test statistics:
    queries performed:
        read:                            140112
        write:                           50040
        other:                           20016
        total:                           210168
    transactions:                        10008  (104.12 per sec.)
    deadlocks:                           0      (0.00 per sec.)
    read/write requests:                 190152 (1978.28 per sec.)
    other operations:                    20016  (208.24 per sec.)

Test execution summary:
    total time:                          96.1201s
    total number of events:              10008
    total time taken by event execution: 1537.0293
    per-request statistics:
         min:                                  27.37ms
         avg:                                 153.58ms
         max:                                 1707.06ms
         approx.  95 percentile:              308.47ms
```

图 30　物理服务器 2 中虚拟化服务器工具测试结果

```
OLTP test statistics:
    queries performed:
        read:                   140168
        write:                  50060
        other:                  20024
        total:                  210252
    transactions:               10012  (70.92 per sec.)
    deadlocks:                  0      (0.00 per sec.)
    read/write requests:        190228 (1347.42 per sec.)
    other operations:           20024  (141.83 per sec.)

Test execution summary:
    total time:                 141.1792s
    total number of events:     10012
    total time taken by event execution: 2256.9915
    per-request statistics:
        min:                    32.75ms
        avg:                    225.43ms
        max:                    2106.82ms
        approx.  95 percentile:         498.73ms
```

图 31　物理服务器 3 中虚拟化服务器工具测试结果

5.6 本章小结

本章主要介绍了在评估虚拟化服务器配置方案时，KVM 虚拟化服务器性能评估模型的应用。首先阐述了目前 KVM 中虚拟化服务器配置原则以及应用性能约束准则，并说明了本应用研究的目的。其次介绍了多准则决策问题的特点以及常用方法，重点说明了层次分析法的主要思想以及具体计算步骤。然后基于 3.1 节对虚拟化服务器性能相关性能指标的分析，建立了层次化的决策分析模型。最后通过实际实验，说明如何在已提出的性能评估模型的支持下，基于层次化的决策分析模型，对虚拟机配置方案进行优先级排序，并对得到的实验结果进行了分析。

总结与展望

工作总结

KVM 正逐渐成为最流行的虚拟化解决方案，因此对 KVM 中虚拟化服务器的性能进行评估是十分重要的。目前 KVM 虚拟化服务器性能研究主要关注于虚拟化系统中某一资源的性能消耗或性能比较，当虚拟服务器为终端用户提供服务器时，服务器提供商需要对其上应用性能进行评估，判断其性能是否满足 SLA 协议。在工业实践领域，目前常用的性能评估工具可以获得虚拟服务器端到端性能，当服务器提供商要对虚拟服务器进行改进时，需要在得到应用性能的基础上，获得主要组件性能，发现性能瓶颈，因此对 KVM 虚拟化服务器性能评估是十分必要的。本文的研究目的是基于目前常用的虚拟化服务器性能评估指标，通过对 KVM 中资源虚拟化实现方式以及运行原理的分析，借助排队网络模型，在应用层面建立 KVM 虚拟化服务器性能评估模型。

本文的研究工作主要分为以下四方面：

1) 通过分析虚拟化服务器性能评估的不同维度，确定采用虚拟化服务器中应用的性能来判断虚拟化服务器性能优劣。根据 SLA 协议中的 QoS 参数，结合已有应用性能度量项研究确定 KVM 中虚拟化服务器性能度量项，并通过调研目前常用基准测试工具使用的性能度量项，对选择度量项的适用性进行了分析判断。

2) 根据 KVM 中虚拟化服务器性能影响因素，对 KVM 中资源虚拟化实现方式以及请求在虚拟化服务器中处理流程进行分析，提出了基于开放型排队网络模型的虚拟化服务器性能评估模型。

3) 基于 KVM 中虚拟化服务器性能影响因素，设计对比实验，通过将获得的性能度量结果与端到端基准测试工具 SysBench 的结果进行对比分析，验证了提出的性能评估模型的正确性与可用性。

4) 借助层次分析法，结合 KVM 中虚拟化服务器性能评估实际特征，提出虚拟化服务器配置决策模型。通过实例说明了性能评估模型在虚拟化服务器配置决策领域的具体应用方法，并对得到的结果进行了详细分析，说明了虚拟化服务器配置决策模型提出的必要性。

进一步工作

由于时间与工作量的限制，本文的研究还存在不足之处，进一步的工作将着重从以下几方面展开：

1) 基于 KVM 中虚拟化服务器性能影响因素，通过改变物理服务器中虚拟化服务器的配置与数量，搭建不同的虚拟化环境，比较性能评估模型得到的性能度量值与基准测试工具获得的结果，进一步验证并提高性能评估模型的精确性。

2) 基于 KVM 虚拟化服务器性能评估指标体系，研究 CPU 与内存分配等性能影响因素对虚拟化服务器性能的定量影响，辅助虚拟化服务器配置决策。

另一方面，随着硬件以及 KVM 虚拟化技术的进一步发展，KVM 虚拟化中资源虚拟化实现方式以及效率可能会发生变化，因此该性能评估模型也需要不断进行修正。如果条件允许，希望将本研究提出的性能评估模型投入企业实际应用，在实践中验证并改进该性能评估模型。

参考文献

[1] 张伟，宋莹，阮利等. 面向 Internet 数据中心的资源管理[J]. 软件学报，2012，23(2)：179-199

[2] Cappuccio D.J.. Energy Saving via virtualization: Green IT on a Budget[R]. Stamford: Gartner Incorporations, 2008

[3] Daniels Jeff. SERVER VIRTUALIZATION ARCHITECTURE AND IMPLEMENTATION[R]. Washington: Avanade inc., 2009

[4] Chen Gart. KVM-Open Source Virtualization for the Enterprise and OpenStack Clouds[R]. Framingham: International Data Corporation, 2014

[5] Eeckhout L. Computer architecture performance evaluation methods[J]. Synthesis Lectures on Computer Architecture, 2010, 5(1): 1-145

[6] 任永杰，单海涛. KVM 虚拟化技术实战原理解析[M]. 北京：机械工业出版社，2013：347-369

[7] 汤凯. Netperf 与网络性能策略[EB/OL]. http://www.ibm.com/developerworks/cn/linux/l-netperf/, 2004-07-01

[8] Standard Performance Evaluation Corporation. SPECvirt_sc@2013 [EB/OL]. https://www.spec.org/virt_sc2013/, 2013-05-22/2015-01-05

[9] VMware. VMmark2.5 Virtualization Platform Benchmark[EB/OL] http://www.vmware.com/products/vmmark/, 2014-02-12

[10] TPC. TPC-VMS[EB/OL]. http://www.tpc.org/tpcvms/default.asp, 2013-11-07/2015-11-19

[11] Ahmadi M.R, Maleki D. Performance evaluation of server virtualization in data center applications[A]. Telecommunications (IST), 2010 5th International Symposium on[C]. Telecom Research Center, 2010:638-644

[12] Huber N, von Quast M, Brosig F, et al. A method for experimental analysis and modeling of virtualization performance overhead[M]. Springer New York: Cloud Computing and Services Science, 2012: 353-370

[13] Benevenuto F, Fernandes C, Santos M, et al. Performance models for virtualized applications[A]. Frontiers of High Performance Computing and Networking–ISPA 2006 Workshops[C]. Springer Berlin Heidelberg, 2006: 427-439

[14] Kraft S, Casale G, Krishnamurthy D, et al. IO performance prediction in consolidated virtualized environments[J]. ACM SIGSOFT Software Engineering Notes. 2011, 36(5): 295-306

[15] Apparao P, Iyer R, Newell D. Towards modeling & analysis of consolidated CMP servers[J]. ACM SIGARCH Computer Architecture News, 2008, 36(2): 38-45

[16] Appel S, Petrov I, Buchmann A. Performance Evaluation of Multi Machine Virtual Environments[A]. 2010 SPEC Benchmark Workshop[C]. Germany Paderbor, 2010: 1-13

[17] Brosig F, Gorsler F, Huber N, et al. Evaluating approaches for performance prediction in virtualized environments[A]. Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)[C]. 2013 IEEE 21st International Symposium on. IEEE, 2013: 404-408

[18] Lv H, Dong Y, Duan J, et al. Virtualization challenges: a view from server consolidation perspective[J]. ACM SIGPLAN Notices, 2012, 47(7): 15-26

[19] Menasc éD A. Virtualization: Concepts, applications, and performance modeling[A]. Int. CMG Conference[C]. 2005: 407-414

[20] Kundu S, Rangaswami R, Dutta K, et al. Application performance modeling in a virtualized environment[A].High Performance Computer Architecture (HPCA)[C], 2010 IEEE 16th International Symposium on, 2010: 1-10

[21] Zadeh K R, Analoui M, Kabiri P. A Performance Model for Evaluating of a Virtualized Computer System[J], 2013

[22] RahimiZadeh K, Nasiri Gerde R, AnaLoui M, et al. Performance evaluation of Web server workloads in Xen-based virtualized computer system: analytical modeling and experimental validation[J]. Concurrency and Computation: Practice and Experience, 2015

[23] Marshall D. Understanding Full Virtualization Paravirtualization, and Hardware Assist[R].Palo Alto: VMware Corp, 2014

[24] McDougall R, Anderson J. Virtualization performance: perspectives and challenges ahead[J]. ACM SIGOPS Operating Systems Review, 2010, 44(4): 40-56

[25] 黄伟达，叶可江，陈建海等，虚拟计算系统性能评测[J]. 中国计算机学会通讯，2011，7（10）：9-15

[26] Chen Y, Farley T, Ye N. QoS requirements of network applications on the Internet[J]. Information, Knowledge, Systems Management, 2004, 4(1): 55-76

[27] Sanjay P. A, Full and Para Virtualization[R]. Florida: University of north florida, 2010

[28] Huber N, Von Quast M, Brosig F, et al. Analysis of the performance-influencing factors of virtualization platforms[A]. On the Move to Meaningful Internet Systems, OTM 2010[C]. Springer Berlin Heidelberg, 2010: 811-828

[29] Tao D, Qin-fen H, Bing Z, et al. Scheduling policy optimization in kernel-based virtual machine[A]. Computational Intelligence and Software Engineering (CiSE)[C], 2010 International Conference on. IEEE, 2010: 1-4

[30] Iyer R, Illikkal R, Tickoo O, et al. VM 3: Measuring, modeling and managing VM shared resources[J]. Computer Networks, 2009, 53(17): 2873-2887

[31] 赵建光. 云计算环境下数据库系统的分层排队网络模型[D]. 太原：太原理工大学，2012

[32] Hirt T. KVM- The Kernel-based virtual machine.[R]. Raleigh: Red Hat Inc, 2010

[33] Reddy P V V, Rajamani L. Evaluation of different hypervisors performance in the private cloud with SIGAR framework[J]. International Journal of Advanced Computer Science and Applications, 2014, 5(2):60-66

[34] Che J, He Q, Gao Q, et al. Performance measuring and comparing of virtual machine monitors[A]. Embedded and Ubiquitous Computing[C]. EUC'08. IEEE/IFIP International Conference on. IEEE, 2008, 2: 381-386

[35] 广小明，胡杰，陈龙等. 虚拟化技术原理与实现[M]. 北京：电子工业出版社，2012：180-181

[36] 丁涛. 内核虚拟机调度策略的研究与优化[D]. 北京：北京航空航天大学，2010

[37] Steven L. p, Dominique A. H, Workload Dependent Performance Evaluation of the Linux 2.6 I/O Schedulers[J]. Linux Symposium, 2004, 7(2):425-448

[38] Goto Y. Kernel-based virtual machine technology[J]. Fujitsu Scientific and Technical Journal, 2011, 47: 362-368

[39] 曹丙部. Virtio 基本概念和设备操作[EB/OL]，
http://www.ibm.com/developerworks/cn/linux/1402_caobb_virtio/，2014-02-10

[40] Grinberg S, Weiss S. Architectural virtualization extensions: A systems perspective[J]. Computer Science Review, 2012, 6(5): 209-224

[41] 华为技术有限公司. 华为 FusionSphere5.0 虚拟化技术白皮书[R]. 深圳：华为技术有限公司，2014

[42] AMD. Virtualizing Server Workloads[R]. Sunnyvale: Advanced Micro Devices. 2008

[43] Hashemian R, Krishnamurthy D, Arlitt M, et al. Improving the scalability of a multi-core web server[A] Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering[C]. ACM, 2013: 161-172

[44] Dayle Parker, Scott Radvan. Virtualization Tuning and Optimization Guide[EB/OL]. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization_Tuning_and_Optimization_Guide/index.html, 2015-07-15

[45] 毛兴江. 服务器性能测试与能效研究[D]. 北京：北京邮电大学，2012

[46] Jakob Nielsen. WebSite Response Times[EB/OL]. http://www.nngroup.com/articles/website-response-times/, 2010-06-21

[47] 朴春华. 层次分析法的研究与应用[D]. 北京：华北电力大学, 2008

[48] Pohekar S D, Ramachandran M. Application of multi-criteria decision making to sustainable energy planning—a review[J]. Renewable and sustainable energy reviews, 2004, 8(4): 365-381

攻读硕士期间取得的研究成果

[1] Jing Yang, Yuqing Lan. A Performance Evaluation Model for Virtual Servers in KVM-based Virtualized System[A]. 2015 IEEE International Conference on Smart City[C]. 2015

致谢

时光荏苒，转瞬即逝，在北航的两年半研究生学习生涯即将结束，我十分感谢在这期间老师，同学与家人给予我的帮助与支持，使我能够顺利完成研究生期间的学业，获得理论与技术的提升。

首先，我要衷心感谢我的导师吴超英老师，在研究生阶段的两年多时间里，吴老师在学习与课题研究方面对我悉心指导，在生活上倍加关系，使我能够在科研工作中全力以赴。在学术研究的实际问题解决上吴老师传授给了我很多宝贵的经验与方法，这为我今后的发展奠定了良好的基础。吴老师严谨的治学态度，平等、自由的学术讨论氛围，使我受益匪浅，并一直激励着我在学习和工作中不断进取，不断前进，在此我特别向尊敬的吴老师致以忠心的感谢和崇高的敬意。

同时我要感谢兰雨晴老师，从论文的选题到论文的定稿都得到了兰老师的详细指导，兰老师在学术研究的很多关键问题解决上给予了我很重要的帮助，确保我可以顺利完成研究工作，兰老师渊博的知识与敏锐的学术思想给我留下了深刻的印象，在此我对兰老师表示我最衷心的感谢。

我也要感谢同课题组的刘平师兄，他在学习与科研方面，给予我很多建议与无私的帮助，使我可以避免不必要的问题，同时我要感谢同课题组的运明纯师弟，在完成老师安排的任务时，他总是给予最大程度的配合，保证任务顺利完成。在课题的研究工作中，研究生期间的同学王洋与夏庆新师兄也给予了很大的帮助与支持，在此对他们表示真诚的感谢。

我还要感谢同实验室的吴际老师，刘超老师与徐红老师在学习方面给予的帮助。虽然同实验室的全体同学，有不同的研究课题，但是在平时的科研和工作中，通过相互工作交流与学术探讨，也使我受益匪浅，在此我要向他们表示感谢。

最后，我要感谢我的父母与家人，他们的支持与鼓励是我前进的动力，没有他们的付出就没有我今天的进步，在此我郑重向他们表示感谢。并且我要感谢我研究生期间的朋友，她们的友谊为我的研究生时光增添了很多快乐。