基于 Xen 的私有云①

张 博[1,2], 李喜旺[2], 李广野[3]

[1](中国科学院大学, 北京 100049)
[2](中国科学院 沈阳计算技术研究所, 沈阳 110168)
[3](东北大学 信息科学与工程学院 通信与信息系统研究所, 沈阳 110819)

摘 要: 为了充分利用企业分布在不同系统的基础设施, 保护企业的核心数据, 提供一个统一的安全的运行接口和运行环境, 避免为每个系统重新部署、设计的重复繁杂工作, 一个较好的解决方案是构建企业私有云. 本文分析了私有云的架构及其优势, 例如拓扑结构、缓存机制、负载均衡策略和存储池设计, 通过物理机集群技术和 Xen 虚拟化技术, 设计基于 Xen 的私有云基础设施, 为用户提供统一的、简单的结构界面.

关键词: 云计算; 云架构; 私有云; Xen; 虚拟化; 实时性

Private Cloud Based on Xen Technology

ZHANG Bo[1,2], LI Xi-Wang[2], LI Guang-Ye[3]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China)
[3](Institute of Communication and Information System, College of Information Science and Engineering, Northeastern University, Shenyang 110819, China)

Abstract: In order to take full advantage of enterprises' infrastructure located in different systems, protect the company's core data and provide a unified secure operating interface and operating environment and avoid redeployment for each system, duplicate complex work of design, a better solution is building an enterprise private cloud. This paper analyzes the private cloud architecture and its advantages, such as topology, caching mechanism, load balancing strategy and storage pool design, through physical machine cluster technology and Xen virtualization technology designs Xen-based private cloud infrastructure, and provide users with a unified the simple structure interface.

Key words: cloud computing; cloud architecture; private cloud; Xen; virtualization; real-time

1 引言

云计算是并行计算技术, 软件技术和网络技术发展的必然结果[1]. 追求高速的计算能力使得并行计算出现, 但是, 并行计算没能充分利用现有的基础设施资源解决性能要求, CPU 的处理速度依然无法满足人们的需求. 因此, 提出了云计算, 它使得人们摆脱了不断通过硬件升级和底层基础设施重构来解决系统对性能日益增加的需求.

云计算通过物理机集群技术, 将性能较低的多台物理机整合, 提供一个稳定性高、性能好的设备, 而且可以按需服务和按需付费, 充分利用现有的资源. 云计算将底层设计全部隐藏在云中, 普通用户不用关注底层如何运行, 所有这一切都是由云计算的中心完成, 可以提供几乎无限的计算能力, 并且容易扩展.

云计算主要分为三种: 公有云、私有云和混合云, 而私有云备受企业和政府的青睐. 数据作为企业的生命线, 私有云在保护数据安全方面具有天然的优势[2]: 1)将分散的数据统一保存, 保证数据的一致性; 2)用户不用管理自己的数据存储, 减少重复的工作, 避免浪费额外的存储空间和人力资源; 3)可以高效的进行信息控制和信息披露, 保证数据安全.

除此之外, 私有云容易实现集中备份和灾难恢复, 易于扩展和升级. 私有云用户只知道存储接口, 不知道的实现的存储空间, 这相当于在私有存储云和连接

① 收稿时间:2015-10-21;收到修改稿时间:2015-11-19 [doi:10.15888/j.cnki.csa.005198]

的用户之间添加中间层, 提高了数据安全性. 私有存储云的空间扩展, 维护, 升级具有很大的灵活性, 后端的变化对系统的影响甚微[3].

2 Xen虚拟化技术简介

Xen 是基于 GPL 许可的开源虚拟化软件, 起源于剑桥大学的研究项目, 后来从该研究项目独立, 成为一个社区驱动的开源软件项目. Xen 支持全虚拟化(HVM)与半虚拟化(PV). 最初使用 Xen 虚拟化技术是通过修改 Linux 内核, 以实现在 CPU 和存储器的虚拟化, 硬件虚拟化(HVM)也开始加入了 Xen 的发展, Xen 具有英特尔 VT 和 AMD-V 硬件技术的全力支持[4].

Xen 管理程序是虚拟机管理程序, 称为 VMM, 它位于硬件和操作系统之间, 为操作系统提供虚拟环境. 在 Xen 中有两种虚拟域, 分别 Domain-0 和 Domain-U, Domain-0 运行在特权模式, 可以调用硬件驱动完成实际操作, Domain-U 是我们常说的虚拟机.

虚拟 MMU 被用来虚拟内存, 帮助客户机操作系统完成地址转换.

事件信道主要用于域域之间、域和虚拟机管理程序之间异步事件通知机制.

物理设备驱动程序用于通用操作系统和 Xen 的运行在特权模式域, 可以调用本地设备驱动程序来访问硬件.

前部和后部驱动(FE/ BE)常常应用在 I/O 设备仿真, 通过请求来实现域之间的事务数据通信. 前端设备用来发送接入请求, 通过事件信道异步发送到后端, 并后端设备最终回将调用回物理驱动程序操作硬件, 并且在前端反馈.

3 基于Xen的私有云基础设施

对于外部用户, 私有云仅仅提供一个调用接口, 而在内部的实现则依赖于分布式计算机节点的动态启动. 设计可以动态扩展云集群的大小, 云可以有效地利用计算机资源和虚拟机技术的优势. 在私有云中, 每个节点需要一个完善的管理控制机制, 特别是在云拉伸和节点启动, 关闭, 数据转移和资源分配.

3.1 拓扑结构

在组网的过程中, 常用的逻辑拓扑结构有星型、总线型、环型、树型和网状拓扑结构五种(图 1).

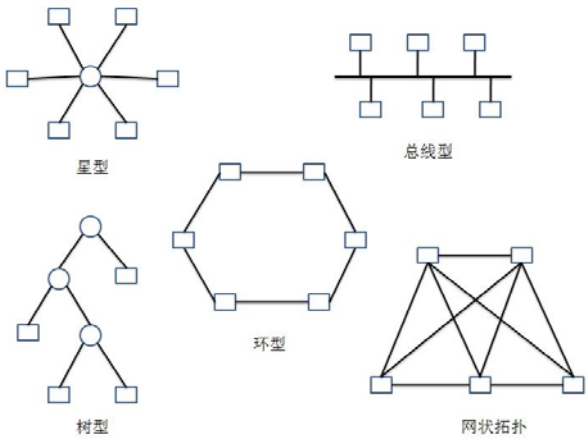

图 1 常用的逻辑拓扑结构

在星型拓扑结构中, 其他节点通过物理链路与中心节点直接相连, 具有结构简单、便于集中管理的优点, 但是, 所有的数据都必须通过中心节点的处理, 中心节点负载大, 可靠性低.

总线形拓扑结构通过一条总线将所有的设备节点相连, 信道利用率高, 且结构简单、价格低廉, 但在同一时间只有两个节点可以进行通信, 系统的扩展是受到节点的数目的限制(节点的数目是有限的), 不能满足私有云对灵活性和扩展性的要求.

环型拓扑是将各个节点连接成一个闭合的环, 随着节点数目的增多, 配置的难度增加, 单个节点故障会导致系统瘫痪, 存在单点故障问题, 不能达到私有云对安全性和可靠性的要求.

网状结构分为全连接网状和不完全连接网状两种形式. 全连接网状中, 每一个节点和网中其它所有节点均通过链路直接连接; 不完全连接网中, 两节点之间不一定有直接连接的链路, 它们之间的通信依靠与其它节点相连的链路. 网状结构使得两个节点间的通信路径增多, 碰撞和阻塞可大大减少, 局部的故障不会影响整个网络的正常工作, 可靠性高, 扩展比较灵活、简单. 但节点间关系复杂, 组建代价高, 控制管理机制复杂.

基于安全性、可靠性、可扩展性等方面的考虑, 我们改进了星型拓扑结构(图 2), 提出了更适合私有云建设的框架. 改进星型结构的设计有利于充分发挥的云的可扩展性. 在私有云中, 整个系统包含中心控制节点、副中心控制节点和普通的子节点. 中心控制节点连接到每个子节点, 通过中心控制节点来控制该星型

结构中的所有节点. 当中心节点出现故障时, 副中心节点是替代中心节点工作, 可确保拓扑继续连接.

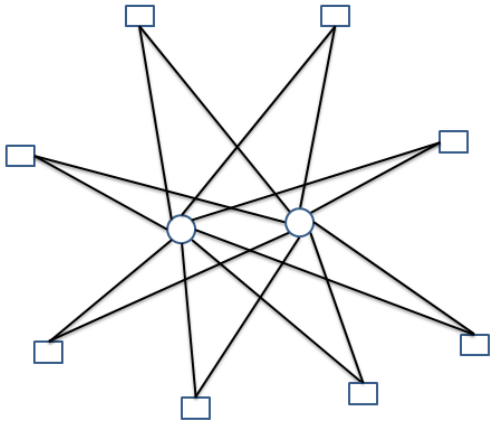

图 2 改进星型拓扑结构

3.2 缓存机制

在私有云的存储服务中, 数据首先存储在子节点上. 但是, 由于云中节点的数量和每个节点的资源是有限的, 所以存储在节点中的数据是有限的, 必须将其从节点转储到数据池中才能进行长期存储. 只有当节点存储的数据容量达到存储上限或者中心控制节点向该节点发送明确的数据上传请求时, 节点才将存储在其上的数据转移到数据池. 根据这个特点, 可以将节点设计为高速缓冲存储器, 即节点的数据上传到私有云的数据池前, 数据仍保留在未传输到数据池的节点, 如果此时接收到服务请求, 该节点可以直接为用户服务[5]. 缓存的设计原理是: (1)每个节点都可以在云中作为高速缓存; (2)在中心节点建立高速缓存表, 存放当前的缓存节点. 当节点的存储数据未转移到数据池之前, 服务器把该节点作为缓存节点处理, 并将缓存节点信息存储在高速缓存表; (3)当客户端请求下载数据时, 中心节点检查高速缓存表, 确定请求下载的数据是否存在于节点中, 如果发现存在, 即视为缓存命中, 中心控制节点允许客户直接与缓存节点进行交互, 完成数据访问; 如果没有找到, 则允许客户和数据池建立连接, 进行数据访问.

中心节点完成了从客户端接收数据传给相应的处理节点和从数据池或处理节点下载请求数据返回给客户的任务. 为了维护高速缓存表, 每个节点上的数据更新后, 节点通过与控制中心节点进行交互, 实时更新中心节点的高速缓存表信息. 当一个节点被销毁, 该节点在高速缓存表中对应的信息也被毁坏.

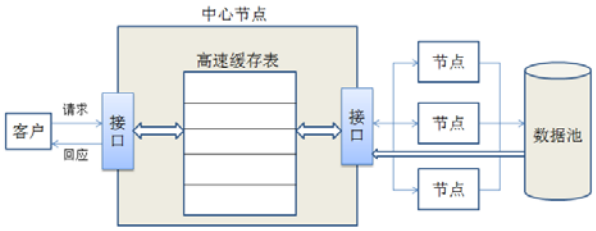

图 3 中心节点的缓存机制

3.3 负载均衡策略

在私有云中, 每个节点都有不同的性能, 而且每个节点的实际操作也不完全相同, 因此, 节点存在着处理能力不能满足到达该节点的所有请求的情况, 需要中心控制节点设计负载均衡策略, 为每个节点的请求进行均衡. 控制中心持续监控所有正在运行的节点, 并通过每个节点硬件的性能为其定义一个性能参数, 代表该节点的剩余计算能力. 当控制中心节点分配任务时, 按照确定的算法, 对任务运行的需要进行量化, 得出一个期望的计算值, 并与每个运行节点的剩余计算能力进行比较, 确定最适当运行此任务的节点, 并分发任务到该节点. 当任务需求的计算能力大于任何运行中节点的容量, 控制中心将新建一个节点, 将任务分发给新的节点.

3.4 存储池

在私有云中, 节点主要用于提供海量数据的计算和处理能力, 将海量数据传输并存储在私有云的存储池中. 节点具有少量的存储能力, 仅仅用来提供高速缓存, 而不提供数据的存储能力. 当子节点面临破坏(包括节点数据达到容量上限)或中心控制节点明确请求数据上传时, 将该节点上的数据转移到数据池. 当用户需要存储大量的数据时, 使用多个子节点进行分别处理和上传, 数据传输效率比较低. 此时, 中心控制节点调整数据上传方式, 让用户与数据池直接进行通信, 提高数据上传速度[6].

4 基于Xen的私有云架构

4.1 结构描述

经过上述分析, 我们构建私有云服务, 如图 4 所示. 在私有云, 控制中心节点使用存储性能好的物理机, 与其他节点建立星型分布式结构. 每个子节点是安装了 XEN 和 Linux 组件的物理集群中的物理机, 数

据存储池提供数据的永久存储. 为每个物理机打开 SSH 服务[7], 控制中心通过 SSH 连接到物理机, 位于控制中心的 SSH 控制器利用 jash 库建立 SSH 连接到物理集群机器. 当连接成功后, 调用他们执行已经编写好的 shell 脚本, 并启动虚拟机, 然后通知控制中心.

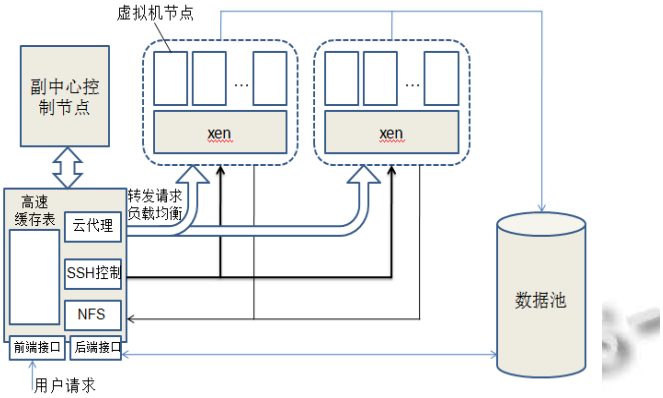

图 4　基于 Xen 的私有云架构

控制中心通过运行 Xen 的 XM 命令管理虚拟机. 通常采用加载配置文件[8]的方式启动虚拟机, 配置文件中主要包含虚拟主机名、内存大小、VCPU 数目、磁盘容量等信息. 配置文件是由 shell 程序产生, 磁盘容量对应于虚拟磁盘文件, 它位于控制中心 NFS 共享目录, 在物理机 SSH 连接运行 shell 脚本在首次将文件复制到本地副本, 然后生成的配置文件, 而 UUID 是随机的产生的, 能够唯一识别一台虚拟机.

为了实现私有云的容错, 控制中心采用 NFS 文件系统来实现同步更新, 消除文件在节点间的拷贝产生的影响. 在私有云基础架构, 控制中心的主要任务是接收和处理大量并发用户请求, 用户与控制中心的互动通过私有云平台提供的 Web 服务接口进行交互. 内部节点的交互通过云进行代理, 当客户端发出请求时, 控制中心在接收和分析后将其发送到云代理, 云代理实现了映射<Sting, CloudNode>, 关键字 key 为节点 ID, 映射值 value 为 CloudNode 对象的值, 当检查所有节点找到最大节点, 如果该节点能够容纳一个新作业, 然后调用该节点 addJob 方法来添加新作业到节点的作业列表中, 或者直接调用新节点进行作业处理. 云监控代理运行在任何时刻的线程, 不断检查运行在每个节点上工作, 如果发现客户端操作请求超时, 则删除操作并释放节点资源. 如果中心控制节点发生故障时, 副中心控制节点接管完成任务.

4.2 节点的管理流程

节点的管理主要包括节点的启动、初始化、分配资源和释放资源等一系列的操作. 控制中心通过在每个节点上运行动态调整算法, 监控云中资源的运行状态. 节点的管理流程图如下.

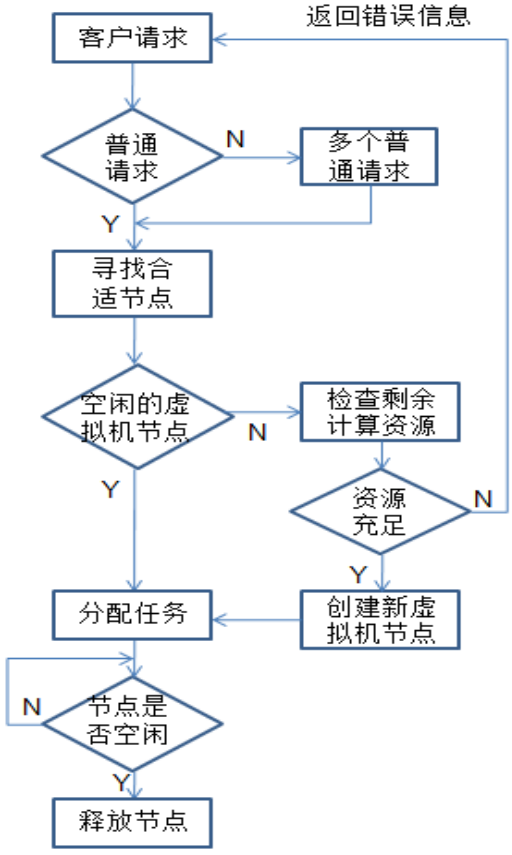

图 5　私有云中节点的管理流程

根据需求将用户的请求分为两类: 普通请求和高可靠性请求. 普通请求是用于处理一般用户的作业, 而高可靠性请求是在用户的作业需要保证高可靠性时产生的. 控制中心接受到用户请求后, 判断请求的类型, 如果为普通请求则将请求转发给云中的运行节点, 为其分配资源并完成请求; 如果为高可靠性请求, 则将该请求转化为多个普通请求, 分发到不同的运行节点.

当一个节点出现计算能力不够时, 不适合以运行新的用户请求时, 控制中心将启动新虚拟机节点进行初始化, 扩大云的负载能力; 当云达到一个节点极限空间, 控制中心妥善处理节点的数据, 并释放节点.

当云中的剩余资源不足以满足并发用户的请求,

且没有可用的资源来运行新的节点, 将会向用户返回一个错误信息.

4.3 实时性处理

Xen 是一个使用广泛的虚拟化平台, 但是它不支持在客户域上运行实时操作系统. 实时 OS 中的实时任务需要较低的调度延迟, 而 Xen 默认的 Credit 调度器公平共享 CPU. 它为每个客户域分配一个权重参数, 每 30 毫秒更新一次, 根据客户域的权重值, 将 CPU 资源分配到不同域的 VCPU. 每个物理 CPU 有一个 VCPU 的运行队列, 队列的前部的 VCPU 具有 boost 优先级, 下一部分是 under 优先级, 之后是 over 优先级, 最后是空闲. 当 VCPU 进入运行队列, 根据优先级次序, 它们将被插入到队列的相应区域的最后, 然后调度器调出首部的 VCPU 进行执行. 该 VCPU 将进入运行状态、等待状态或阻塞状态. 此外, 信用调度程序可以多个物理处理器的内核间进行负载均衡. 当一个物理核心是闲置, 或者没有 boost 和 under 优先级的 VCPU, 调度器会检查所有的队列中, 如果其他队列存在较高优先级的 VCPU, 则分配为该物理核心处理执行.

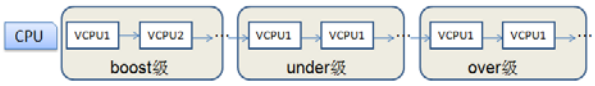

图 6 credit 调度器的 VCPU 队列

通过改变客户系统的权重值对实时性能的影响不大. 包含实时任务的客户域在运动队列中的位置由优先级决定, 当实时客户需要从设备或 Domain0 配置紧急事件时, 它将需要等待运行队列排在前面的客户被调度, 并且用完给定的时间片或陷入阻塞状态, 才会调度当前的实时客户. 这样将对实时客户产生不可避免地额外的响应延迟.

R_Credit 调度器优先调度实时客户, 在 VCPU 队列中出现实时任务时可以优先调度, 减少调度响应延迟, 但当有两个实时客户会出现并发抢占, 会导致实时性能的下降. 两个实时客户会在等待状态耗费较多的时间, 增加延迟.

因此, 我们引入多实时客户均衡的方法解决并发抢占问题. 当有两个或两个以上的实时客户位于同一个运行队列, 调度器会将实时的客户分发到其他没有实时客户的运行队列中. 为每个处理器内核分配一个实时客户, 以确保抢占能力和低延时服务. 非实时客户使用 Credit 调度器使用处理器内核. 这就是 R_B_Credit 调度器, 它包含了 R_Credit 调度器的功能, 并且支持多实时客户的均衡.

在参考文献[9]中, 详细的对不同调度器的调度延迟进行对比试验, 得出 R_B_Credit 调度器可以显著降低实时任务的平均调度延迟. 因此, 使用 Xen 的 R_B_Credit 调度器, 可以在私有云中很好支持实时系统和实时任务.

4.4 数据共享

Xen 可以让不同的客户系统同步运行, 通过在不同的存储空间运行客户系统实现了物理隔离, 通过将不同的客户系统绑定不同的网络实现网络的隔离. 通过隔离, 因此, 系统的安全得到很大保证, 但客户系统间的数据共享受到了限制, 客户系统在网络下不可互达.

常用的数据共享方式有三种: 网络传输、可移动存储设备和共享内存. 利用网络进行数据共享, 由于数据将在网络上进行分发、重装、传输, 因此容易遭受破坏. 使用可移动存储设备进行数据共享, 操作比较复杂, 对于应用程序既不方便也不便于控制. 而传统的共享内存只适合于运行在同一 OS 的不同应用程序间的数据共享, 它不能在不同 OS 中使用.

Xen 提出了一个新的共享内存方法, 通过将 Xen 驱动程序进行分离, 分为前端驱动程序和后端驱动程序, 前端驱动程序运行在不同的客户 OS, 后端驱动程序运行在管理域 Domain0. 如果客户 OS 需要进行数据共享, 需要通过前端驱动设备将请求发送到后端驱动设备, 由后端设备调用物理设备访问内存数据. 该技术可以使位于不同域间的用户方便的进行数据共享. 但这种方法存在两个不足之处: 首先, 未考虑安全性和可控性, 共享数据不会被做任何具有特定策略的检查, 无法确保数据的完整性; 其次, 这种方法是基于 Xen 的半虚拟化, 即在该方法下仅仅支持客户系统是 Linux 或 UNIX, 而不支持 Windows 作为客户系统.

针对上述问题, 提出了虚拟磁盘共享[10]的解决方案. 在全虚拟化的模式下, Xen 使用 Qemu 设备虚拟化模型, 每个客户系统对应一个 Qemu-DM 进程, 该进程提供系统需要的所有虚拟设备. 除了为客户系统提供的一个统一服务的虚拟磁盘外, 其余创建的所有虚拟磁盘用于数据共享. 多个客户系统可以共享虚拟磁盘,

而实际上是物理存储资源相同的部件.

我们对虚拟磁盘共享方法在性能上和有效性进行了实验验证.

1) 实验环境配置

主机名称	硬件配置		系统内核版本
	内存	磁盘	
Host OS	2G	100G	Linux 2.6.18
GuestOS 1	1G	10G + 5G 共享磁盘	Rhel 4.5
GuestOS 2	1G	10G + 5G 共享磁盘	Rhel 4.5

2) 实验设计

GuestOS 1 为虚拟磁盘的所有者, 拥有对虚拟磁盘的读写权限. GuestOS 1 设置 GuestOS 2 对虚拟磁盘具有只读权限时, GuestOS 2 如果尝试对虚拟磁盘写操作, 将在2- 4 秒内产生写入错误的警报; 如果GuestOS 1 禁止对虚拟磁盘的所有访问, GuestOS 2 在对虚拟磁盘进行读取或写入操作时会得到错误警报. 实验结论证明了虚拟磁盘共享对操作的有效性.

至于对虚拟磁盘共享的性能, 我们通过与 FTP 传输方式进行对比实验. 结果如图 7.

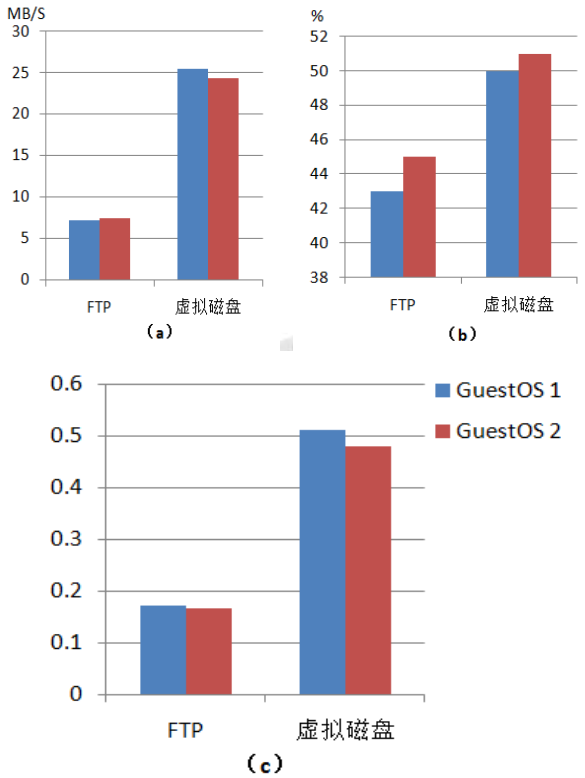

图 7　实验数据

在图 7(a)中, 纵轴表示数据的传输速度, 磁盘共享传输速度是 FTP 的 3.5 倍左右; 在图 7(b)中, 纵轴表示 CPU 的使用率, 使用磁盘共享方式占用更多的 CPU; 在图 7(c)中, 我们用数据传输速度除以 CPU 使用率, 得出消耗相同的 CPU 下的传输速度, 可知采用虚拟磁盘共享的性能是使用 FTP 进行数据传输的 3 倍.

6　结语

本文研究了基于 Xen 的私有云架构设计, 提出了通过底层架构的设计, 保证私有云支持实时系统的实时任务和不同系统间的高效数据访问, 为提供即时服务打下基础. 设计方案只是建设私有云的开始, 在后续的建设中还会重点研究如何提高业务的可靠性, 为用户提高高效、可靠的服务.

参考文献

1 孙香花.云计算研究现状与发展趋势.计算机测量与控制, 2011,19(5):998-1011.

2 李治勃.基于私有云数据处理关键技术的研究与应用[硕士学位论文].北京:北京工业大学,2014.

3 钱进进. 私有云安全存储技术的研究与实现[硕士学位论文].广州:广东工业大学,2013.

4 石磊,邹德清,金海.Xen 虚拟化技术.武汉:华中科技大学出版社, 2009.

5 Zhang J, Li XY, Guan HB. The optimization of Xen network virtualization. Computer Science and Software Engineering, 2008.

6 Wu JY, Ping LD, Ge XP, Wang Y, Fu JQ. Cloud storage as the infrastructure of cloud computing. Intelligent Computing and Cognitive Informatics, 2010.

7 Xenserver. XAPI 用户使用帮助文档. 2013. http://www.xenproject.org/developers/teams/xapi.html.

8 Miao XY, Han J. The Design of a private cloud infrastructure based on Xen. International Symposium on Distributed Computing and Applications to Business, Engineering and Science, 2011.

9 Yu PJ, Xia MY. Real-time enhancement for Xen hypervisor. IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2010.

10 Liu FG, Zhou W, Zhou M. A Xen-based data sharing & access controlling method. International Symposium on Intelligent Information Technology Application, 2009.