清 华 大 学

综 合 论 文 训 练

题目：物联网 2.0——基于虚拟社区的物联使能环境

系　　别：自动化系

专　　业：自动化

姓　　名：陈伟

指导教师：曹军威　研究员

2010 年 6 月 19 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：_____导师签名：_____日　期：_____

中文摘要

物联网在最近的十几年间有了飞速的发展，而随着网络环境的进一步普及以及，我们身边的每一件物品都有可能连入网络，成为物联网的一员。例如我们将可以使用任何一种终端来控制电器等，各种传感器设备也可以将信息通过网络传送给任何需要这些信息的人。而我们提出的物联网 2.0，将流行的虚拟社区的概念引入了物联网，使得资源共享更加充分，大大扩展了物联网的应用前景。

我们将虚拟机作为物联网终端的一种特例进行开发，在物联网中进行虚拟机的创建，远程访问，共享等。证书认证等机制可以使得访问更加安全可靠。用户可以使用远程桌面软件登录创建的虚拟机并进行操作，或将程序运行于远程的虚拟机，如同操作本地计算机一样。虚拟机的共享建立在虚拟组织的基础上，通过对用户和资源的统一管理，可以很容易的进行访问控制。

关键词：物联网；虚拟社区；虚拟机；证书认证

ABSTRACT

Internet of things has been rapidly growth in last ten years. As the further popularization of network environment, everything around us will be likely connected into network and act as a member of internet of things. For example, we will be able to use any terminal to control electrical appliances, and all kinds of sensors will send information to anyone who needs the information by this network. While the Internet of Things 2.0 we proposed will include the popular concept of virtual communities, so that the resource sharing can be more fully and the future of Internet of Things will be brighter.

We develop the system by considering the virtual machines as a special case of Things, and make the operations like creating, remotely operating, sharing the virtual machines. Certificate authentication mechanism makes access more secure and reliable. Uses use remote desktop software to login and operate the remote virtual machines, or run programs in remote virtual machines, like what they do on local computer. The sharing of virtual machine is based on virtual organization. Through the unified management of users and resources, control access is easy to do.

Keywords： Internet of Things, Virtual Communities, Virtual machines, Certificate authentication

目 录

第 1 章 引 言 1

1.1 项目背景 1

1.2 技术背景 3

1.2.1 云计算技术介绍 3

1.2.2 网格计算技术介绍 6

1.2.3 虚拟化技术介绍 7

1.2.4 证书认证技术介绍 9

1.3 相关工作 10

1.4 物联网 2.0 设计理念 11

1.5 文章结构安排 12

第 2 章 关键技术分析 13

2.1 ODBC 数据库 13

2.2 虚拟机技术 15

2.2.1 虚拟机选择和简介 15

2.2.2 VirtualBox 的命令行控制方式 16

2.2.3 虚拟机的网络连接 19

2.3 ELOP 框架 21

第 3 章 系统框架 23

3.1 系统框架 23

3.2 各部分实现 24

3.2.1 客户端实现 24

3.2.2 社区中心服务器实现 25

3.2.3 虚拟机提供端实现 27

第 4 章 系统评估和未来展望 29

4.1 系统评估 29

4.1.1 客户端评估 29

4.1.2 社区中心服务器评估 35
4.1.3 虚拟机提供端评估 37
4.2 未来展望 41
插图索引 43
参考文献 45
致 谢 46
声 明 47
附录 A 外文资料的调研阅读报告（或书面翻译） 48
第一章：分布式和云计算系统模型[1] 48
第六章：云系统构建和数据中心设计 50

第1章 引 言

1.1 项目背景

物联网集合了传感技术，通信技术，计算技术，网络技术，控制技术于一体，其应用前景遍及生活，生产，服务等各个领域，将是一种改变人类生活方式的新技术。国际社会公认其为信息产业中继计算机，互联网之后的第三次革命浪潮。加强物联网的研究将有助于我国在新一轮的科技革命中占得先机。

物联网概念的起源可以追溯到 1999 年，由 MIT 的 AUTO-ID CENTER 首先提出。随后的互联网蓬勃发展，无线网络技术越来越成熟，为这两者的结合奠定了良好的基础。通过把现有的互联网向各个载体扩展，尤其是传感设备和控制设备的扩展，使得信息的采集，传输，计算处理，执行一体化完成，从而可以顺利完成智能化识别，定位，跟踪，监控，管理等任务。物联网既是现有产业集成的结果，也对其他产业的发展起到了推动作用。

物联网概念从 1999 年提出后，各国，各领域都对其给予了高度关注和重视，物联网的研究也因此成为国际科技竞争的制高点。目前欧美等国家都在加速部署物联网基础设施，加速物联网的进一步研究，而信息产业的龙头企业如 GE，IBM，西门子等也已全力投入了物联网的研究。在国际大环境下，我国也指定了物联网的发展计划，将物联网同新能源、绿色制造等并列为我国五大新兴战略产业。加快物联网的研究有助于提升科学技术水平，改善人民生活质量，提高社会服务管理能力，实现信息化推动工业化，提高国际竞争力。[1]

第一，夯实物联网基础理论是提高我国科技创新水平的内在需求。我国科技发展的战略核心是提高自主创新水平。物联网是一个新兴的产业，目前世界各国都处在起步阶段，而我国在这方面的研究与发达国家几乎同时起步，甚至在某些研究方向上还处于领先地位，继续坚持开拓创新将是我们与发达国家持续同步或领先的必要条件。本项目提出的物联网 2.0 就是在物联网的基础上进行的一次有意义的尝试和创新。

第二，促进物联网产业腾飞是推动国民经济跨越式发展的迫切需求。物联网被公认为是继计算机和互联网之后的第三次信息技术革命浪潮。物联网作为数字系统向物理系统的延伸和拓展，其产业链相比于计算机和互联网更长，产业规模

也要更大。市场前景将不可估量。传统的计算机，网络，信息服务等行业在物联网的带动下将继续快速发展，而物体标识，定位，传感，控制等新兴产业增长点也在物联网的带动下出现，将成为我国经济跨越式发展的动力。在过去的 50 年间，计算机和互联网的两次革命浪潮我国都没有完全抓住，而这第三次信息技术革命浪潮政府和企业都格外关注，力争抓住这次机遇，实现我国信息产业的崛起和跨越式发展，带动国民经济快速增长。

第三，促进物联网应用开发是推动管理服务模式变革、提升我国社会管理服务能力的重大需求。物联网连接了物理世界和数字世界，利用数字世界的实时感知来控制物理世界，在电力，交通，公共安全等社会服务管理领域有重大意义。物联网相当于信息社会的神经网络，在感知和控制方面起主导作用，必将成为社会服务管理模式变革的核心动力。

第四，本项目是突破信息技术壁垒限制、保障我国国家安全的必然需求。在前两次信息产业浪潮中，西方国家占得先机，控制了芯片制造，基础软件，高端元器件等核心领域，造成我国在信息产业的发展上受制于人。而物联网作为第三次信息浪潮的核心，其应用更加广泛，将渗透到社会的每个角落。加快物联网的发展将是摆脱发达国家的技术垄断的关键道路，也是保障国家安全的必然要求。

整个物联网主要包含传感器网，数字控制系统，应用软件系统。而将这三者结合起来的是互联网。传感器网络主要负责物理状态感知，数字控制系统用来执行，而应用软件系统实现一切跟踪，监控，管理职能。因此从功能来看，应用软件层是物联网的关键。物联网是市场前景不在于传感器或执行机构的数量，而是在于应用，而目前的应用软件层的开发很少，而且大部分是依赖于分布式系统和嵌入式系统的理论研究，现在并不完全适用于物联网。

本项目试图构建一个新型的物联网模型，通过引入虚拟社区管理来管理连入物联网中的元素。本项目的主要工作集中于应用软件层，在软件架构上将资源进行抽象、组合、分配和管理。通过虚拟组织的管理，资源的分配会更加合理，资源的共享会更加充分。在未来的物联网发展中，技术将退居幕后，人才是信息社会的主角，每个人都可以随心所欲的控制控制和使用网络中的资源。虚拟社区的思路就来自于人类社会的组织关系，在物联网中借鉴人类社会的组织关系模型，充分体现了技术的以人为本。

虚拟社区(Virtual community)的定义最早由瑞格尔德(Rheingole)做出,他将其定义为“一群主要藉由计算机网络彼此沟通的人们,他们彼此有某种程度的认识、

分享某种程度的知识和信息、在很大程度上如同对待朋友般彼此关怀,从而所形成的团体。”

虚拟社区拥有不同于现实社区的独特属性：

超时空性：虚拟社区不受物理上距离的限制，地球两端的两个人也可以通过虚拟社区进行交流。时间上也同样不受限制，你所发表的一句话或是一个操作，可能会在几天后被其他人看到并作出回应。

匿名性：在虚拟社区中每个人都只有一个符号。你可以随意选取你的名字，由于不能看到对方的真面目，传统的性别，年龄，相貌，都在虚拟社区里不再重要。正如网上的名言：和你聊天的可能是条狗。

群体流动频繁：在虚拟社区中，人际关系较为松散，社区群体流动频繁，大家可能会被某个社区的人气所吸引而加入一个社区。

例如 QQ 群，BBS 等就是典型的虚拟社区，但是它仅仅局限于信息共享和文件共享等，不能做到深度的资源共享。而本项目中所研究的新型的虚拟社区，不仅仅是一个文字交流的平台，更是一个深入分享资源的平台。在这种虚拟社区上，计算资源，文件资源，包括物联网终端控制权限都是可以共享的，而共享的权限管理掌握在资源所有者的手中。如同人类社会中的组织一样，资源共享建立在虚拟组织（Virtual Organization）的基础上，虚拟组织可以由用户自发创建，谁创建谁管理，在同一个虚拟组织中的人可以有限度的共享资源和信息。[2]

本项目所构建的物联网 2.0，以物联网为基础模型，以虚拟组织为主要管理模式，其设计参考了云计算网格计算的概念，并将流行的虚拟化技术融入其中，最终构建为一个通过网络来远程访问虚拟资源和物理资源的系统。

1.2 技术背景

1.2.1 云计算技术介绍

“云计算是一种基于互联网的计算新方式，通过互联网上异构、自治的服务为个人和企业用户提供按需即取的计算。由于资源是在互联网上，而在电脑流程图中，互联网常以一个云状图案来表示，因此可以形象地类比为云，‘云’同时也是对底层基础设施的一种抽象概念。”[3]

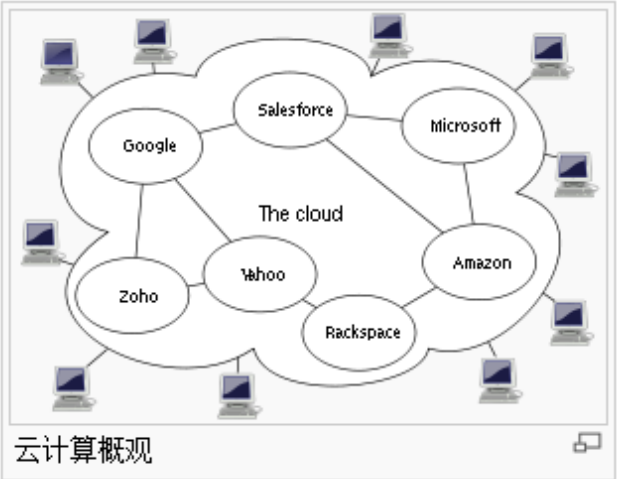

图1.1　云计算概观

图 1.1[注 1]形象的描述了云计算的组织形式。云计算提供三个层面的服务：IaaS，PaaS 和 SaaS，分别为设施即服务，平台即服务，软件即服务。

图 1.2[注 2]说明了这三个不同的服务类型。

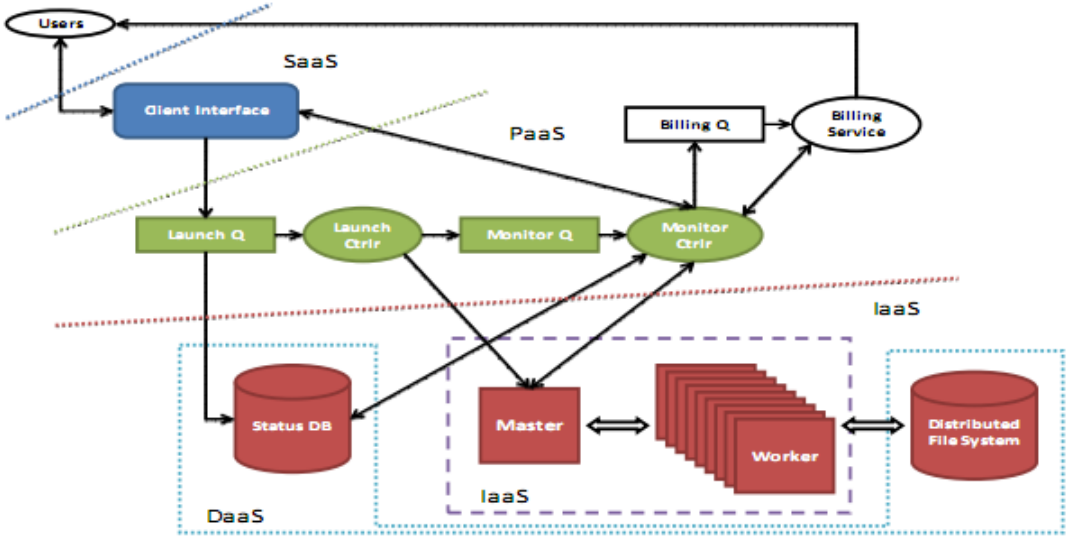

Figure 6.4: Anatomy of cloud service models: The IaaS provides virtualized infrastructure at user's costs. The PaaS is applied at the platform application level. The SaaS provides specific software support for users at web service level. DaaS (Data as a Service) applies the status database and distributed file system. Some providers consider DaaS an integral part of the IaaS.

图1.2　云计算三种服务模型

[注 1] 图片取自《Cloud System Architecture and Datacenter Design in Distributed Systems and Cloud Computing》

[注 2] 图片取自《Cloud System Architecture and Datacenter Design in Distributed Systems and Cloud Computing》

Infrastructure as a Service（IaaS）：

这个模型允许用户租用处理，存储，网络和其他一些资源。用户可以部署和运行自己的 OS 和应用。用户不管理或者控制底层的云设施，但可以控制操作系统，存储设施，部署应用程序，还有可能可以选择网络组件。这个 IaaS 模型包括了存储，计算资源，通信资源作为服务。这一类型的服务有：Amazon-S3 提供存储服务，Amazon-EC2 提供计算资源，Amazon-SQStigong 通信资源。

Platform as a Service(PaaS):

这个模型允许用户在云设施上部署用户建立的应用程序，这些应用程序必须是使用提供商支持的语言和软件工具编写的。用户不需要管理底层的云设施，云提供商在一个良好定义的服务平台上帮助建立整个应用程序的开发，测试和运行。这个 PaaS 模型使得世界各个不同位置的开发者有可能在同一个平台上共同开发软件。这个模型的其他服务方面包括第三方软件管理，整合以及服务监控的解决方案。Google App Engine 和 Microsoft Azure 是这个模型的两个例子。

Software as a Service(SaaS)：

这个模型是指基于浏览器的应用程序，它运行于数千个云客户之间。SaaS 模型把软件应用程序作为服务，而不是让用户购买软件包。因此，在用户来看，不需要对为购买软件许可或服务进行一次性的预投资。在提供商来看，相比于传统的作为用户应用程序的伺服机代价更加低廉。客户的数据存储在云中，既有可能是卖主所有的，也有可能是公共的云平台如 PaaS 和 IaaS。大量的商业软件被作为服务提供。成功的案例有微软的 online sharepoint 和 SalesForce.com 的 CRM software。

通常云计算服务应该具备以下几条特征：

1. 基于虚拟化技术快速部署资源或获得服务
2. 实现动态的、可伸缩的扩展
3. 按需求提供资源、按使用量付费
4. 通过互联网提供、面向海量信息处理
5. 用户可以方便地参与
6. 形态灵活，聚散自如
7. 减少用户终端的处理负担

云计算所提出的集中服务的概念将是未来计算的主旋律，除了能够充分利用大型数据中心提供的计算能力外，如果能够把零散的计算资源也整合起来将是更

加符合绿色计算的要求。而我们希望实现的物联网，不仅仅要求把计算资源进行整合，而且希望能够将各种设备都连入网络统一管理。

1.2.2 网格计算技术介绍

网格是继 Internet，web 技术之后第三次网络技术浪潮革命。前两者实现了计算机网络的连接，构建了完整的信息传送技术，而这两者的技术进步主要体现在网络传输速度，信息流量大小和传输的安全性。而网格技术关注的是如何有效的整合网络上离散的各种资源，利用 Internet 将地理上分散的计算能力，存储能力，带宽资源，软件资源，数据资源，知识资源等整合为一个逻辑整体，从来完成一台单独的计算机无法完成任务，为用户提供一体化的信息和应用服务，最终实现在这个整合的环境下的资源共享和协同工作。

网格技术广泛应用于大规模计算领域，用来替代超级计算机的工作。如分布式超级计算，高吞吐率计算，数据密集型计算，在各种科学项目中发挥了重要的作用。例如在生物领域遗传信息的相关计算，网格技术起到了关键性的作用。而随着网格技术的进一步发展，它已经被推广至应用服务领域，如广泛的协同工作，环境保护，培训和教育，信息集成等。企业和组织内部也可以利用网格技术整合计算资源和数据资源，从而有效的优化资源，降低成本。从网格的客体对象不同，我们可以将网格分为数据网格，计算网格和服务网格数据网格中共享的基本单位的数据，主要解决的问题是数据的共享问题；计算网格的共享内容是计算资源，通过设计好的接口和分布式机制，把计算资源合理的利用起来，通过并行化和分布式算法来解决大规模计算问题；服务网格的共享对象是服务，我们通过一定的手段将不同的资源封装起来为网格用户提供服务。

网格的想法借鉴自电力网络，我们在使用电力的时候，是不需要考虑点是从哪里发出来的，而只需要接入电力网络就可以享受大这个服务。网格的最终目标也是希望我们在使用计算能力的时候可以不需要考虑它的来源，只要接入网络就能够享受。

网格系统有四个特点：一是异构性，加入网格的计算资源有不同的形式，它们来自与不同类型的计算机，有不同的操作系统，不同的计算框架，不同的计算方式和存储方式。因此，一个成熟的网格系统应该能够合理的处理异构性，并能够一致的管理好资源。二是自治性，网格上的资源首先属于本地的用户，因此本地的用户拥有最高的管理权，网格的管理不能够凌驾于用户的决策之上。第三是动态性，网格中的资源很可能随时加入和退出网格，网格必须对这种情况予以足

够的考虑，能够动态的分配资源。第四是自相似性，网格的局部和整体存在一定的自相似性，局部在很多地方具有全局的某些特征，而全局的特征在局部也有体现。[4]

1.2.3 虚拟化技术介绍

虚拟化是一个 IT 领域新兴的概念，它指的是计算元件在虚拟的基础上而不是真是的基础上运行，其主要目的是优化资源，简化管理，提高通用能力。

图 1.3[注 1]表示的就是一个通过虚拟化技术将数据中心的计算资源提供给用户的示意图。

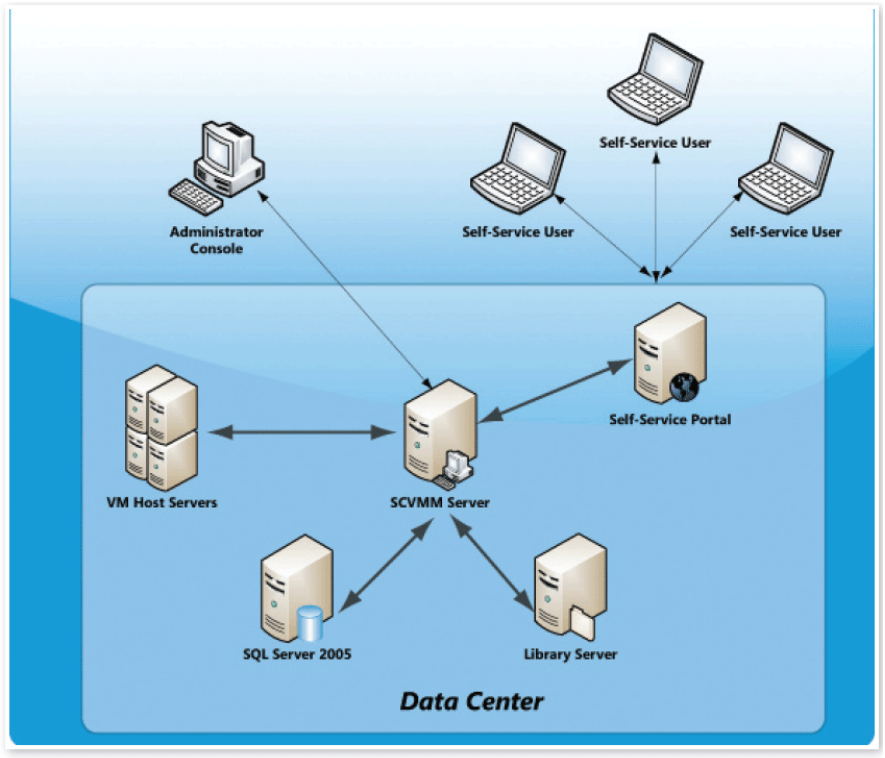

Figure 1: Corporate data center setup

图1.3　虚拟化技术示意

在维基百科中虚拟化的定义为：“虚拟化是一个表现逻辑群组或电脑资源的子集的进程，用户可以用比原本的组态更好的方式来存取这些进程。这些资源的新虚拟部分是不受现有资源的假设方式，地域或物理组态所限制。一般所述的虚拟化资源包括计算能力和资料存储。”[5]

[注 1] 图片摘自维基百科

虚拟化技术可以在有限的硬件资源上模拟更大的硬件环境，简化软件的配置和运行，如 CPU 的虚拟化可以让一个 CPU 模拟多个 CPU 运行，从而在一个平台上允许多个操作系统同时运行而互不影响，显著的提高工作效率。

虚拟化的主要目的是对 IT 基础设施进行简化。它可以简化对资源以及对资源管理的访问。终端用户通过受虚拟资源支持的标准接口来对资源访问，通过标准接口，可以在基础设施变化时对用户的破坏降到最低。

实现虚拟化有软件方案和硬件方案。纯软件方案中，客户操作系统通过虚拟机监视器（VMM）来与底层硬件交互。VMM 决定虚拟系统对硬件的访问。因此 VMM 此时相当于传统意义上的操作系统，而客户操作系统则相当于应用软件。这中间物理资源都需要中间层的转换来与客户操作系统交互，因此需要较大的额外开销。VMWare 就提供了一整套解决方案。其主要软件包括：
VMWare-ESX-Server，这个不需要操作系统，本身就可以充当操作系统管理硬件资源；VMWare-GSX-Server 需要安装在一个操作系统之上，其他功能与 ESX 版本基本相同；VMware-WorkStation 也需要安装在操作系统之上，区别是没有提供 web 远程管理和客户端管理的功能。

而在硬件方案上，支持虚拟技术的 CPU 带有特别优化过的指令集来控制虚拟过程，因此可以提供比纯软件高得多的性能和稳定性。虚拟化的硬件支持虚拟操作系统直接在上面运行，从而减少了中间转换的环节，省掉了额外的开销，性能大大提高。同时，还具有支持 64 位系统的能力。目前，Intel 和 AMD 都争相发布支持虚拟化技术的芯片，虚拟化市场前景将非常乐观。

虚拟化可以通过多种方式来实现，它不是一个个体，而是一组技术和模式的结合。下面是虚拟化常用的一些技术和模式。

1. 单一资源的多个逻辑表示：这是最常见的虚拟化模式之一。在实际上只包含一个物理资源，但是向用户呈现的是多个资源的形态。只访问一个虚拟资源的用户不会考虑到其他用户正在与他分享一个实际的资源，也不会受其影响。虚拟机的实现就是这种虚拟化模式。
2. 多个资源的单一逻辑表示：这种模式下包含了多个实际资源，但是虚拟化的整合，他们以统一的接口对外提供服务，表现为单一的逻辑资源。在虚拟化存储中该模式非常常见，如虚拟文件系统等。用户从虚拟文件系统中访问文件时，不会察觉到上面的两个文件可能在物理存储上分布在不同的地方。
3. 在多个资源之间提供单一逻辑表示：这种模式与上一种模式非常类似，但是仍有一些明确的差别。在这种情况下，每一个物理资源都能够提供逻辑所需

要的所有功能，在发生请求时，虚拟资源会根据需要选择一个物理资源来实现。一个常见的例子是，用户提交任务请求后，虚拟资源根据任务负载均衡来讲请求提交给某一个实际的运行环境进行执行，而用户并不关心到底是哪一个物理资源提供了服务，而只关心请求能否得到处理。

4. 单个资源的单一逻辑表示：这是用来表示单一资源的一种简单模式，用另一种不同的形式来表示一个资源，仿佛它是另一种资源一样。这种模式通常用来重用一些基本的功能。

5. 复合和分层虚拟化。这种模式通过上面几种模式的组合实现，使用强大的物力资源池来提供丰富的功能。通常意义的网格计算就需要这种模式的实现。本系统中，通过对远程服务器的虚拟化，让用户连接虚拟机资源，从而提高了可扩展性，能够提供不定量的虚拟资源服务，而不像传统的计算机群，每台单独的计算机只能够为单一的用户提供服务。[6]

1.2.4 证书认证技术介绍

为了确保通信的安全，我们使用 OpenSSL 对通信内容加密以及身份确认。OpenSSL 使用的 PKI 安全机制。

PKI（Public Key Infrastructure ）即"公钥基础设施"，“是一种遵循既定标准的密钥管理平台,它能够为所有网络应用提供加密和数字签名等密码服务及所必需的密钥和证书管理体系，简单来说，PKI 就是利用公钥理论和技术建立的提供安全服务的基础设施。”[7]

PKI 技术是信息安全技术的核心，其基础技术包括加密，数字签名，数据完整性机制，数字信封，双重数字签名等。完整的 PKI 系统必须具有权威的认证机构 CA 中心，CA 中心负责数字证书的申请和签发，必须具备权威性的特性。公钥体制涉及一对密钥（即私钥和公钥），私钥只有用户独立掌握，无需在网上传输，而公钥则是公开的，故公钥体系的密钥管理主要是针对公钥的管理问题，目前比较好的方法就是数字证书机制。在这种机制下，用户可以独立的认证对方的身份，而无需和 CA 中心沟通信息，证书的持有者可以被正确的识别出身份，其发送的信息也可以保证无法更改，无法冒名顶替，无法否认。[8]

PKI 的优势主要体现在：

1. 采用公开密钥密码技术，能够支持可公开验证的无法仿冒的数字签名，从而在确认可追求的服务商有不可替代的优势。

2. 保护机密性。这是 PKI 的最基础的服务，它的在于能够为陌生的两个用户提供保密支持。这在我们的系统中至关重要。
3. 由于数字证书可以由用户独立认证，原理上能够服务范围的无限扩张，突破了传统安全验证方式必须在在线的限制。
4. PKI 提供了证书撤销机制，从而提供了在意外情况下的补救措施。从而使用 PKI 服务的用户不必担心身份被盗窃并恶意使用。
5. PKI 适合大型系统的互连。互相信任的上下级关系和平等的第三方信任关系堵在 PKI 中很好实现，PKI 的各种互联机制使构建一个复杂的网络信任体制成为可能。

本系统使用的就是数字证书，首先由用户向证书中心发出请求，并填写个人信息，证书中心确认之后给用户发送一个数字证书，其中包含了用户的个人信息，如 commonName，在一个证书中心签发的证书中，commonName 都是唯一的，用来确认用户的身份。数字证书中还包含了通讯使用的公钥和私钥。拥有同一个证书中心签发的证书的用户可以安全的进行通讯，并能够保证其信息的完整和不可冒充，即使他们原本彼此并不认识和信任。

1.3 相关工作

在网络环境下远程使用和共享虚拟资源的相关研究早就已经开展，而一些信息产业的高科技公司也致力于在其中发掘市场价值。

Citrix 公司一直致力于虚拟化研究，并为用户提供一整套远程方案。XenDesktop 是他们发布的一款旨在为用户提供和使用台式机相同体验的产品，而他们使用的系统可能是数据中心的某个虚拟机。

同样还有 XenApp，它提供了应用虚拟化的技术，操作系统仍然是用户的真实操作系统，然而可以通过虚拟化的方式按需提供 Office 等软件，而不是在系统上安装这些软件。

而在最近，他们又发布了面向客户 PC 的 XenClient，这是一款采用了 hypervisor 的虚拟化软件。Hypervisor 是一款支持完全虚拟化的基础软件，它能够运行于裸机而不需要操作系统，因此避免了为主席选择 CPU 的问题。同时可以利用 CPU 的虚拟化支持功能，大大提高了性能。主流的服务器虚拟化技术早已经应用了这一裸机基础架构的技术，然而面向客户 PC 的这一技术尚不成熟，因为他们所需要支持的硬件设备大不相同。在服务器架构中无关紧要的 USB，打印机，高级图

形界面等在 PC 上不可忽略，例如 PC 的显卡可以支持 Windows Aero，而这在虚拟化系统中成为了一大难题。

因此 Citrix 发布了支持 PC 必备功能的 hypervisor-XenClient，它采用了 Intel 的 VT-d 技术，在硬件上避免了二次翻译，大大提高了性能，因此可以使用全部操作系统的图形功能。客户端 PC 虚拟化可以实现个人环境向商务环境的切换，如个人操作系统安装了各种各样的应用，即使在最糟糕的情况下（如病毒侵袭），商务操作系统仍能够保持安全状态。此外，XenClient 还可以配合 Citrix 的 XenServer 实现虚拟操作系统环境的自动备份，因此如果 PC 环境发生故障，也可以马上复原备份的虚拟操作系统环境。

1.4 物联网 2.0 设计理念

物联网的概念已经很普及，而物联网 2.0 的理念还没有普遍达成共识。作为物联网 2.0 系统，要具备以下特征：

1. 物物相连的功能。物联网 2.0 是在物联网的基础上的升级改进，因此物物相连的本源功能必须具备。在物联网 2.0 系统中，任何设备都可以连入网络进行远程控制或远程获取信息。
2. 以人为本的设计。虽然名称叫做物联网，但该系统的设计是为了更好的体现以人为本的理念。网络构建的时候必须考虑人性化的设计，技术将在幕后起支持作用，而不是物联网的主体。
3. 组织方式要适合人的认知习惯。人在进行活动时，经常是以组织为单位进行，而组织可以是自发形成，也可以由权威人士发起。在组织内部，信息和资源可以有限度的共享。而每个人都可能加入多个组织，每个资源也可能被多个组织共享。在物联网 2.0 中，组织的功能由虚拟组织来完成。人和资源也可以分别由虚拟组织来管理。
4. 自由的控制终端。当所有的控制器和传感器都连入了物联网，这些资源的拥有者可以通过任意终端连入网络来获取信息并进行控制。因此需要一个统一的远程显示接口和访问接口。只要实现了这个接口，不管终端是一台计算机还是一个嵌入式的手持设备，都可以使用相同的方式来控制。

在该理念的指导下，并考虑到可行性，本系统实现的功能如下：

以虚拟机为资源的示例，以远程登录软件作为远程访问接口的实现，通过虚拟组织管理资源。每个用户在虚拟组织服务器中注册之后，可以创建，加入一个

虚拟组织，创建虚拟组织者自动成为虚拟组织的管理员，而加入一个虚拟组织则需要经过该虚拟组织的管理员批准。一个用户也可以加入多个虚拟组织。虚拟机作为资源，可以由虚拟组织的正式成员创建并添加到本虚拟组织中，然后该虚拟组织的所有成员即可以使用该资源。对于虚拟机来说，使用该资源就是可以远程打开虚拟机并登录到该虚拟机中进行操作。而已经存在的虚拟机资源在经过一定的许可之后可以加入其他的虚拟组织，由多个虚拟组织共享该资源。这一操作的许可可以根据需要由资源的拥有者、虚拟组织的管理员或是虚拟组织的任何成员来批准。

本系统中实现的虚拟组织管理功能由数据库技术实现，而与虚拟机相关的许多操作通过 VirtualBox 的命令接口实现。

1.5 文章结构安排

在本文第一章中，我们介绍了该项目的工程背景以及目前的发展情况。与我们的系统相关的云计算，网格计算，虚拟化技术和证书认证技术都进行了简单的介绍。同时我们介绍了与我们的项目相关的工作，主要是 Citrix 的多款虚拟化及远程工具。

在第二章中，我们介绍了我们系统的三项关键技术，数据库的选择和表单设计，虚拟机技术，以及 ELOP 框架。虚拟机技术将是我们项目能够实现的核心，而我们的系统未来将是 ELOP 框架的一部分，与框架中其他内容的配合将极为重要。

第三章中我们将具体的讨论系统的框架实现细节，系统分为三个部分，每个部分单独介绍其中的技术难点和解决方案。然后综合介绍系统的优势。

在第四章我们对系统进行总体评估，从三个部分的运行状态来研究系统的运行情况。然后讨论系统的缺点和未来的发展方向。

第2章 关键技术分析

2.1 ODBC 数据库

数据库是该系统的重要基础，要管理很多用户的个人信息和虚拟组织信息，就需要使用数据库来存储。在 linux 系统下，最常用的便是 MySQL 数据库，然而如果直接使用 MySQL 数据库的编程接口，将使程序的移植和扩展极为困难，如果要更换底层数据库更是需要将程序做很大的改动，因此我们选择使用 ODBC 接口来完成数据库的操作。

ODBC，全称 Open Database Connectivity，即开放数据库互连，由微软公司开发，作为开放服务结构的一个组成部分发布。它提供了一组访问数据库的标准编程接口，这些 API 以 SQL 语句为基础，完成其大部分的数据库操作任务。基于 ODBC 的应用程序无需关心底层数据库的类型，也不需要和底层数据库打交道，无论是 FoxPro、Access , MYSQL 还是 Oracle 数据库，都可以使用 ODBC 的接口进行访问，保证了程序的通用性和可移植性。

数据在数据库中是以表（table）的形式存在的，每个表在新建时可以指定有哪几列，分别代表什么信息，是否非空等。然后每一行代表一条记录，可以增加删除和修改。

在实验室之前的工作中，针对我们的工作在 ODBC 基础上进一步进行了封装。针对我们的数据库中的数据，在配置文件中写好数据库的表项，即可以把每一项的信息作为参数后直接调用插入函数。这样做的好处是当数据库结构发生改变时，只需要改变配置文件中的数据库表项说明，程序是不需要发生变化的，也就不需要重新编译，保证了程序的通用行。

虚拟组织的数据库是由实验室之前的开发完成的，管理用户和虚拟组织一共使用了 4 个表，分别是 UserInfo，VOInfo，Member，Message。创建这几个表的时候都指定了表的列，命令如下：

```
CREATE TABLE UserInfo(commonName varchar(50) NOT NULL, IP varchar(50) NOT NULL , Identity varchar(50) NOT NULL);
CREATE TABLE VOInfo(VOName varchar(50) NOT NULL, Founder varchar(50) NOT NULL,  Statement varchar(100) NOT NULL);
```

CREATE TABLE Member(user varchar(50) NOT NULL, VO varchar(50) NOT NULL, status varchar(50) NOT NULL, Membership varchar(50) NOT NULL, Reputation varchar(50) NOT NULL, SuccIntr varchar(50) NOT NULL, FailIntr varchar(50) NOT NULL, subject varchar(200) NOT NULL, additional varchar(50) NOT NULL);

CREATE TABLE Message(user varchar(50) NOT NULL, MessageType varchar(50) NOT NULL, Message varchar(300) NOT NULL, time int(50) NOT NULL);

从每个表所包含的项目中即可看出每个表的作用。

UserInfo：包含了用户的 commonName，IP 和身份信息，是用来管理所有已注册用户的数据库。当用户使用证书中心颁发的证书与服务器进行通讯的时候，服务器会将用户的信息加入到该数据库中。并且每次登陆会更新其 IP 信息。

VOInfo：包含了虚拟组织的名称，创建者和说明。这个表仅仅包含了虚拟组织的信息，而不包含其中的用户信息。在用户查询当前已经存在虚拟组织时，将会从该表中查找数据返回给用户。

Member：这个表中的内容最为丰富，因为这个表是用来存储用户和虚拟组织的关系的。每一条记录包含了一个用户对一个虚拟组织的从属关系。而一个用户可以加入多个虚拟组织，这将会在表中体现为多条记录。同时该表中还包含了一些用户的表现记录，如参与任务的成功率等，这些事实验室之前工作中用来评估用户可信度的，在本系统中没有使用，但为了一致性，数据库中的信息仍做了保留。

在管理虚拟机时增加了 2 个数据库，创建时的命令如下：

CREATE TABLE SupporterInfo(commonName varchar(50) NOT NULL, IP varchar(50) NOT NULL, VMleft int(50), Statement varchar(100) NOT NULL);

CREATE TABLE VmInfo(VmName varchar(50) NOT NULL, Supporter varchar(50) NOT NULL, SupporterIP varchar(50) , port int(50) , Statement varchar(100) NOT NULL , VoName varchar(50) NOT NULL);

SupporterInfo 表是用来存储虚拟机提供端的信息的，包括虚拟机提供端的 commonName，IP，剩余资源可分配的虚拟机，描述等。由于虚拟机与社区中心服务器的通讯也是用经过证书认证的 gSoap 通讯，因此提供端也有一个唯一的 commonName，但这个名字只是用来表示，实际通讯时用不到。IP 则是最重要的

信息，因为服务器是单边向提供端发送信息，因此必须保证 IP 的正确以及不常更改。

VmInfo 表用来存储已经注册过的虚拟机信息，VmName 是由一定规则创建的，在整个服务器上是唯一的，是用来指明虚拟机的唯一标示。同时，记录中还包括了提供端的名称，IP，远程端口，描述等。还有一个很重要的是 VoName，即所在的虚拟组织名称。由于在同一个虚拟组织中的用户可以共享虚拟机，因此必须标明某一个虚拟机可以由哪一个虚拟组织中的用户共享。

但是需要说明的是，并不是一个虚拟机就只对应一条记录，而是一个虚拟机和虚拟组织的对应关系对应一条记录。因此，如果一台虚拟机可以被两个虚拟组织共享，将会在表中体现为两条记录，但是两条记录中的虚拟机名称相同，可以确认为一台虚拟机。

2.2 虚拟机技术

2.2.1 虚拟机选择和简介

我们的项目是物联网的构建，但是由于物联网的构建需要大量的其他工作作为基础，如软件物化，电器设备的网络接口等。而我们的工作重点并不在此，因此选择了虚拟机作为我们整个系统构建的突破口，也就是用虚拟机来暂时代替“物”搭建系统。

之所以我们最终选择了使用虚拟机来作为物联网中物的代表，一方面它便于控制，不涉及硬件环境，从工作量和可行性来看最为合适；另一方面，它也具有一定的代表性。因为未来的电器设备连入网络之后，我们对其进行操作也是通过控制它们的嵌入式操作系统来实现，而虚拟机就可以看作是嵌入在电脑中的嵌入式操作系统，在远程显示，网络构架，资源的抽象描述等方面都有一定的相似性和代表性，因此我们选择了虚拟机作为我们目前我们系统所要控制的终端。

在 linux 平台下，有多款虚拟机的软件可供选择。最常见的包括 VmWare 和 VirtualBox。VmWare 是一款优秀的虚拟机商业软件，有 windows 和 linux 版本。VmWare 功能极为强大，因此其程序也很庞大。VirtualBox 则是一款开源软件，其核心代码由开源社区维护。这款软件则比较小巧，而且功能也完全符合我们的需要。因此最终我们选择了 VirtualBox 作为虚拟机的支持。

VirtualBox 提供了两种管理模式，一种是图行界面的管理方式，另一种是命令行界面的管理，对于一般用户来说，图行界面的管理方式更加美观，并且方便易用。

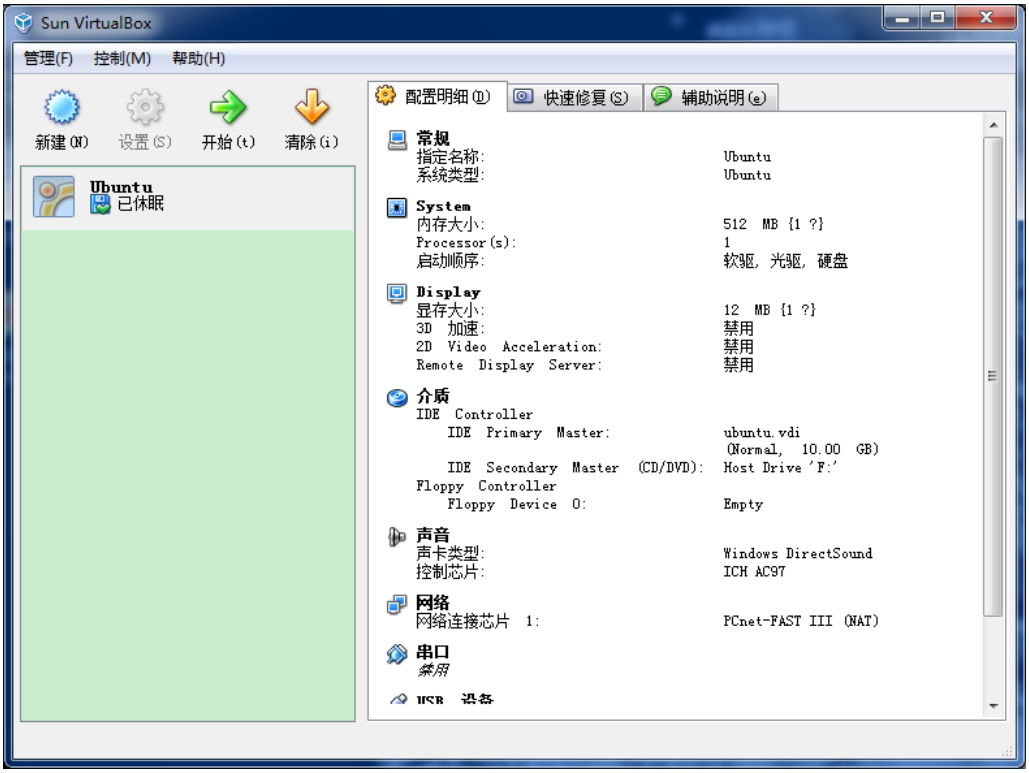

图2.4　VirtualBox虚拟机图形操作界面

这是 VirtualBox 运行时的界面，从图中可以看出，当前系统中只有一台虚拟机，名称为 Ubuntu，右侧是该虚拟机的配置明细，例如内存大小，显存大小，挂载的驱动器，网络连接等都可以配置，对于安装在其中的系统来说，这就是一台完整的计算机。此时只需要点击开始按钮，虚拟机的系统就会启动。

除了这种图形界面的窗口程序，VirtualBox 还可以使用命令行在后台运行。该方式更容易让我们的程序调用，因此在虚拟机提供段的程序我们都使用了这种方式进行控制。与此相关的命令主要有两个，VBoxManage 和 VBoxHeadless。

2.2.2 VirtualBox 的命令行控制方式

VBoxHeadless 的用处是打开一个无头虚拟机。无头虚拟机是指创建一个虚拟机，但是该虚拟机的界面不在窗口中显示出来，而只能通过远程登录的方法访问到。在我们的系统中，虚拟机提供端和使用虚拟机的客户物理上不在一起，因此

正需要使用远程访问来操作虚拟机，而且使用无头虚拟机可以节省本地的计资源消耗，从而为同时开启更多的虚拟机创造了条件。该命令的用法很简单，只需要加上“-s”参数以及虚拟机的名称即可打开一个无头虚拟机。

VBoxManage 用法则很复杂，实际上，这条命令就相当于整个 VirtualBox 程序，VBoxManage 命令后可以接 40 多条二级命令，包括了 VirtualBox 图形界面所能进行的所有操作，甚至还包括了一些因为不常用而没有放入图形界面程序中的操作。

下面介绍我们将会用到的几个重要命令。

```
VBoxManage createvm     --name <name>
                        [--register]
                        [--basefolder <path>
                        [--settingsfile <path>]
                        [--uuid <uuid>]
```

我们使用该命令创建一个新的虚拟机，但其实这条命令仅仅创建了一个配置文件，而没有为虚拟机分配实际的空间。--register 参数是必要的，否则在虚拟机列表里无法找到该虚拟机，但也可以通过 VBoxManage registervm 来实现注册功能。

```
VBoxManage modifyvm             <uuid|name>
                                [--name <name>]
                                [--ostype <ostype>]
                                [--memory <memorysize in MB>]
                                [--vrdp on|off]
                                [--vrdpport default|<ports>]
```

该命令可选参数有几十条，用来配置虚拟机的各种选项，如 cpu，内存，远程访问，网络连接，串口通讯，usb 通讯，声卡等等。在这里只列出了几项最基本的参数，其他参数用法相同。特别要注意的是远程端口“--vrdpport”参数，这个必须对每一个虚拟机指定一个单独的端口，不能重复。否则将无法同时远程访问这两台虚拟机。

```
VBoxManage controlvm            <uuid>|<name>
                                pause|resume|reset|poweroff|savestate|
                                acpipowerbutton|acpisleepbutton|
                                keyboardputscancode <hex> [<hex> ...]|
```

```
                                vrdp on|off |
                                vrdpport default|<ports> |
```

该命令是在运行时可以用来控制虚拟机的一些选项，如远程访问是否打开，但一般情况下，该命令最主要的用途在于控制虚拟机关闭或休眠，如需关闭，使用 poweroff 参数，休眠则使用 savestate 参数。该命令还可以模仿一些其他的实际机器的操作，如直接断电等。

```
VBoxManage createhd             --filename <filename>
                                --size <megabytes>
                                [--format VDI|VMDK|VHD] (default: VDI)
                                [--variant Standard,Fixed,Split2G,Stream,ESX]
                                [--type normal|writethrough] (default: normal)
                                [--comment <comment>]
                                [--remember]
VBoxManage modifyhd              <uuid>|<filename>
                                [--type normal|writethrough|immutable]
                                [--autoreset on|off]
                                [--compact]
VBoxManage clonehd              <uuid>|<filename> <outputfile>
                                [--format VDI|VMDK|VHD|RAW|<other>]
                                [--variant Standard,Fixed,Split2G,Stream,ESX]
                                [--type normal|writethrough|immutable]
                                [--remember] [--existing]
```

这三条命令是用来创建，修改，复制虚拟磁盘的，虚拟磁盘是一个文件（*.vdi），VirtualBox 创建的虚拟机就使用该文件来作为它们的磁盘，因此一个虚拟机中几乎所有的系统信息都存在于该磁盘文件中。但在我们的实际应用中，如果使用第一条命令创建了一个空的磁盘，里面是没有系统的，需要挂载一个虚拟光盘来安装系统。我们还可以用第三条命令来复制一个已经有一些内容（比如安装好了系统）的磁盘，这样就省去了安装系统的操作。“--remember”参数用处在于让复制后的磁盘注册在 VirtualBox 的虚拟资源池中，否则该磁盘将无法被 VirtualBox 虚拟机使用。

```
VBoxManage storagectl           <uuid|vmname>
                                --name <name>
```

```
                                    [--add <ide/sata/scsi/floppy>]
                                    [--controller
<LsiLogic/BusLogic/IntelAhci/PIIX3/PIIX4/ICH6/I82078>]
                                    [--sataideemulation<1-4> <1-30>]
                                    [--sataportcount <1-30>]
                                    [--remove]
VBoxManage storageattach        <uuid|vmname>
                                    --storagectl <name>
                                    --port <number>
                                    --device <number>
                                    [--type <dvddrive|hdd|fdd>
                                     --medium
<none|emptydrive|uuid|filename|host:<drive>>]
                                    [--passthrough <on|off>]
                                    [--forceunmount]
```

这两条命令通常配合使用，前一条是用来为虚拟机添加或删除设备，如需挂载驱动器，一般要指定“--add”参数。但这个只是增加了 IDE 控制器，需要挂载一个已经创建好的虚拟磁盘时，需要使用第二条命令，在 IDE 控制器上添加媒体，即指定“--medium”并在其后加虚拟磁盘的名称。用这种方式就可以将前面用 clonehd 命令复制的磁盘挂载到新的虚拟机上，构成一台完整的虚拟机。

2.2.3 虚拟机的网络连接

我们要让创建的虚拟机能够在网络上访问到，网络连接的配置很重要。VirtualBox 虚拟了多款不同种类的网卡，并为虚拟机提供了多种网络连接方式，我们可以根据实际需要来进行选择。其中 host-only 方式是只允许虚拟机访问 host 主机，而不能访问到外面的网络，因此该方式在我们的系统下不适用。剩下的两种方式主要是 NAT 和桥接。

NAT 方式：host 不需设置，guest 需设置使用 VirtualBox 提供的 DHCP 服务。配置好之后，三种情况的访问情况如下：

Guest -> WAN：由 VirtualBox 的 NAT 提供 WAN 的访问服务。相当于能够正常的连接网络。

Guest -> Host：需要注意的是，如果直接访问 Guest 拿到的网关 IP，会发现这个 IP 似乎是 Host。不过事实上不能直接访问网关 IP 来访问 Host。因为这个 IP 是由 VirtualBox 负责的。只实现了 NAT 的功能。其他的一些功能并不能正常运行(如 FTP)。如果要访问 Host，应该访问 Host 的真实 IP。

Host -> Guest：不可访问。Guest 没有自己的实际网络地址，从 Host 或从 WAN 上都无法直接访问到虚拟机。但是 VirtualBox 提供了一种端口映射的方式可以用来模拟访问虚拟机。

VirtualBox 的配置文件是一个 xml 文件，通过修改该文件就可以完成宿主机到虚拟机的端口映射。如添加以下三行，就会将连接到宿主机 22 端口的 ssh 连接映射到虚拟机的 22 端口，通过这种方式，我们可以在虚拟机中架设 web 服务，ftp 服务等，并通过端口映射让外界通过访问宿主机的 ip 就可以访问虚拟机的服务。

```
<ExtraDataItem name
="VBoxInternal/Devices/pcnet/0/LUN#0/Config/ssh/Protocol" value="TCP"/>
<ExtraDataItem name
="VBoxInternal/Devices/pcnet/0/LUN#0/Config/ssh/GuestPort" value="22"/>
<ExtraDataItem name
="VBoxInternal/Devices/pcnet/0/LUN#0/Config/ssh/HostPort" value="22"/>
```

桥接方式：VirtualBox 还提供了一种网络连接配置方式，就是桥连，在这种情况下，虚拟机将有自己的网络地址，我们首先在 VirtualBox 的配置中选择 bridge 连接，并将虚拟机的 ip 设置为与 host 的 ip 在同一个子网内，然后发送到虚拟机 ip 的数据包都会经过 host 的网卡转发到虚拟机内。这种方式下，无论是虚拟机访问外界还是外界访问虚拟机，都是可以的，因为虚拟机有了实际的 ip 地址。

综合考虑这两种方式，我们最终选择了 NAT 方式连接。因为 NAT 方式已经可以满足我们的应用需求，我们不需要虚拟机有实际的 IP 地址，而且桥接方式下，如果我们再同一个计算机上开设多个虚拟机，就必须要求宿主机的网段内有多个空余 IP，这个要求在很多情况下不能满足，因此在我们的系统中选择了 NAT 方式进行网络连接。

2.3 ELOP 框架

ELOP 是正在开发的一套网络资源管理的中间件，它具有四层的结构，每层之间相互合作，完成从资源的抽象，管理，调度，使用一体化的框架。本系统也是 ELOP 软件包的一个组成部分。

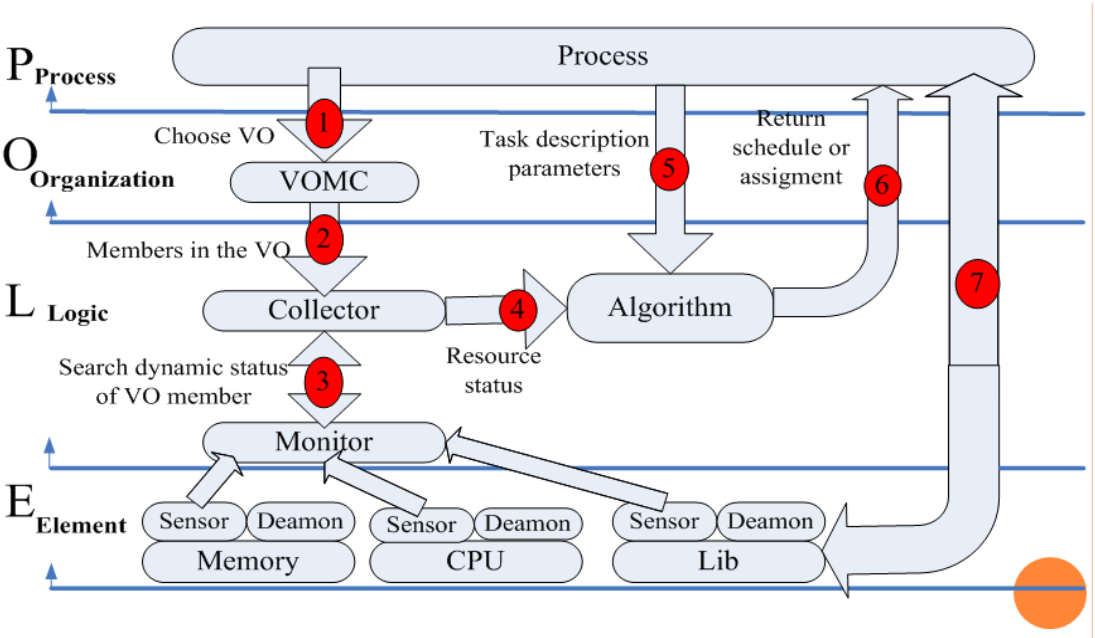

图2.5　ELOP框架示意图

该图形象的表示了 ELOP 四层的含义以及组成部分。

E（Element）代表元素层，网络中的各种资源，如计算资源 CPU，存储资源，显示资源，还包括一些软件资源如链接库等组成了这一层，该层的主要任务是管理资源的抽象表示，并对外提供输入输出端口。

L（Logic）表示逻辑层，这一层中主要负责对下层元素的逻辑关系的管理，资源的收集分类，应用程序所需资源的组合等。一些资源调度的算法也在这一层实现。

O（Organization）代表组织层，这一层中管理元素以及用户的组织关系，虚拟组织就是这一层的重要内容。除了虚拟组织外，该层还需要负责任务级的调度，分析任务并根据权限以及需求等进行分配。

P（Process）代表执行层，在这一层里负责应用程序的执行。应用程序包括用户提交的应用程序以及终端自身所需要运行的程序。

ELOP 是一个在云计算环境下完整的组织管理平台，在物联网应用中，也可以作为很好的一个中间件，将连入网络的“物”抽象为一个元素，通过合理的逻辑组合，组织关系调度，是资源能够充分的发挥出它的作用。

第3章　系统框架

3.1　系统框架

在我们实现的远程访问和共享虚拟机的系统里，主要由三部分组成，一个是虚拟社区服务器，一个是虚拟机的提供端，一个是用户终端。

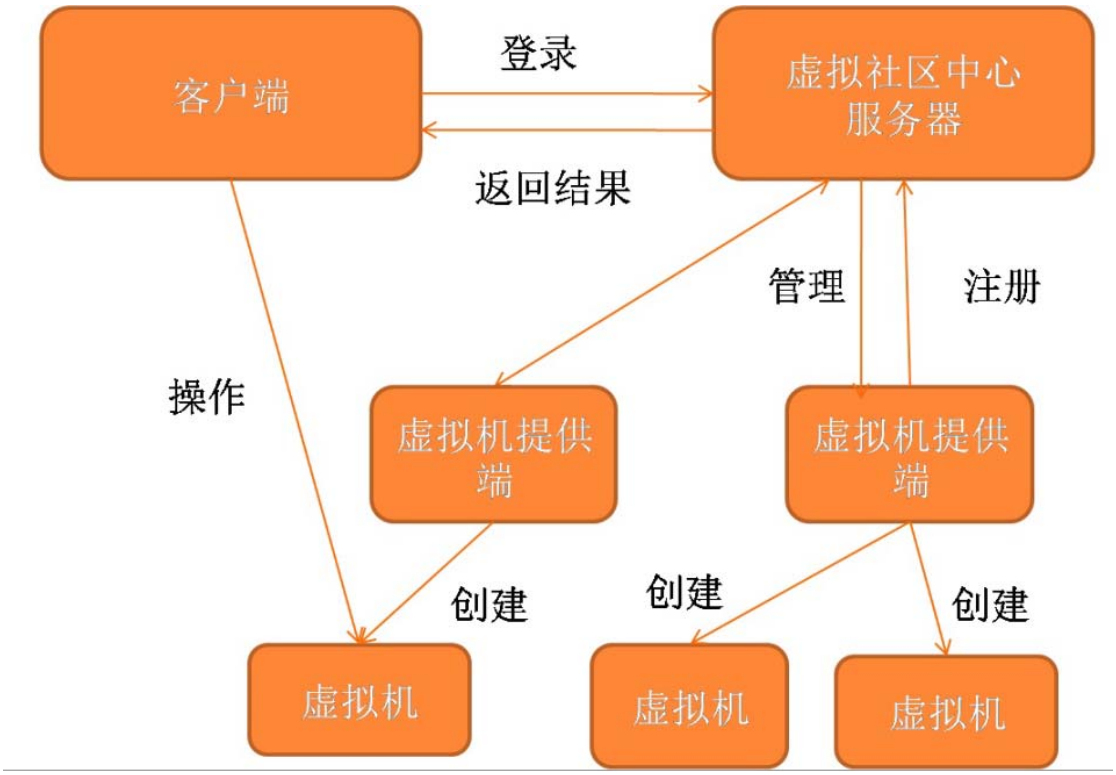

图3.6　系统整体框架

简要的工作流程如下：

客户在任何终端登陆虚拟社区中心服务器，社区中心服务器验证用户身份，返回用户所需要的数据，如果用户需要在虚拟机提供端提供一台虚拟机，可以向社区中心服务器发出申请，社区中心服务器经过审核（或验证用户交费信息）后，想虚拟机提供端发出创建虚拟机的请求，虚拟机提供端将会创建虚拟机，并将虚拟机的信息向中心服务器报告，存入数据库。用户需要操作虚拟机的时候，也是像社区中心服务器提交请求，社区中心服务器向提供端递送信息，提供端开启虚

拟机，并打开远程登录端口，等待用户连接，然后经过身份认证，用户就可以开始远程使用这台虚拟机了。

整个系统在 linux 系统下，主要使用 C++语言编写，辅助以 shell 命令编写完成。程序编译后分为客户端程序，社区中心服务器程序，虚拟机提供端程序三个可执行程序。环境依赖关系较为复杂，每部分都需要的依赖组件有 OpenSSL 和 gSoap。在社区中心服务器端，额外需要安装 ODBC，MySQL 等。在客户端需要安装 rdesktop 远程桌面服务。在服务器提供端，要安装 VirtualBox 程序，在目前情况下必须安装非开源版本的 VirtualBox 以保证其远程访问功能的正常使用。

3.2 各部分实现

3.2.1 客户端实现

客户端主要使用 C++语言编写完成。目前客户端程序仍是一个演示程序，只提供了命令行的访问方式，用户通过键入命令来进行操作。

程序在接收到用户输入的命令后，首先进行分析，确认用户的命令内容，用户的命令分为两种，一种是虚拟组织操作命令，一种是虚拟机操作命令。如果用户的命令式虚拟组织操作命令，则使用以前实验室中虚拟组织管理的操作方式，将命令传送到社区中心服务器。如果是虚拟机操作命令，则按照作者定义的格式将信息发送到社区中心服务器。

通讯方面：在本系统中，我们使用了 OpenSSL 来管理证书认证以及对通讯进行加密。通讯方面使用了 gsoap，从而避免了自行编写 socket 程序，简化了工作并提交了稳定性。更重要的一点是，gsoap 支持 OpenSSL 的加密通讯。

本系统使用 C/S 架构的通讯模式，一方为 Client 端，具有主动发送功能并可以接受返回信息。，一方为 Server 端，不能主动发送信息，负责监听端口收取 Client 端发来的信息并作出回应。连接均为非持久连接，在一次通讯中，Client 端将一个字符串序列通过 gsoap 发送到 Client 端，Client 接受信息，交给适当的函数处理，并给出返回字符串，再经由 gsoap 发送回，然后连接就中断，下一次新的信息传送将会开启一个新的连接。

在本系统中，在客户端只有一个发送端，其接收端在社区中心服务器。在社区中心服务器还有一个发送端，接收端则位于虚拟机提供段。

3.2.2 社区中心服务器实现

社区中心服务器在整个系统中起了核心的作用，负责了所有用户和资源信息的存储，以及用户间，用户与资源间的信息交互。它主要由两部分程序组成：一是等待用户传送信息并处理，二是发送请求到虚拟机提供端。

接受用户的信息端，服务器接受到客户端发送的请求后，仍然是分为两种情况，一种是虚拟组织操作命令，另一种是虚拟机相关操作命令。下面主要介绍后一种情况。

下面介绍用户的几种主要请求情况：

1. 新建虚拟机

在我们的系统中，同一个虚拟组织内部可以深度共享资源，因此虚拟机在虚拟组织内是可以共享的。而一个用户可以加入多个虚拟组织，并使用这几个虚拟组织中的虚拟机。因此用户在要求新建虚拟机的时候必须指定将该虚拟机共享于哪一个虚拟组织内。

在客户端的设计中，用户此时发送的信息包括：

事件：CreateVM（创建虚拟机） 虚拟组织名称：VoName 设计虚拟机名称： VmName

其中，VmName 主要是为了以后功能扩展时使用，而目前虚拟机为了防止重名，采用的命名方式是请求者的用户名（commonName）加数字，由于用户的 commonName 都是唯一的，因此虚拟机的名称也不会出现重复的情况。

步骤 1：检查用户的身份。根据具体应用的不同，可以新建虚拟机的用户要求也不一样，这一步主要是看用户的身份是否满足在资源池中为他新建虚拟机的要求。

步骤 2：检查虚拟组织。检查该用户是否是所申请的虚拟组织的正式成员，以及该虚拟组织是否有资格共享虚拟机。

步骤 3：查找可用的虚拟机提供端。在我们的数据库设计中，包含有已注册的虚拟机提供端信息，例如剩余可创建的虚拟机数量。在数据库中查找一个有剩余资源并可用的虚拟机提供端。

步骤 3：向虚拟机提供端提交请求。具体的请求格式将在虚拟机提供端详述。如果申请成功，虚拟机提供端将会返回该虚拟机的访问端口信息。端口信息直接对应一个可以远程访问的虚拟机。

步骤 4：向数据库中添加数据，包括该虚拟机的名称，提供端 IP，端口信息，虚拟组织信息等。

2. 开启虚拟机

在客户端的设计中，开启虚拟机发送的信息包括：

事件：OpenVm 虚拟机名称：VmName

在创建虚拟机的时候，虚拟机名称就已经确定了，而且是唯一的，因此用户只需要指定虚拟机的名称就可以了。

服务器在接收到指令之后，进行以下步骤。

步骤 1：检查用户身份，这一步同创建虚拟机相同，不过可能需要根据应用的不同而设定开启虚拟机所需要的身份和创建虚拟机所需要的身份的不同，在目前实现的系统中，所有的正式用户都有权利创建和开启虚拟机。

步骤 2：在管理虚拟组织成员的数据库中搜索，找到该用户加入的所有虚拟组织。

步骤 3：在管理虚拟机的数据库中搜索，找到前面找到的虚拟组织中是否有用户所申请开启的虚拟机的名称。

步骤 4：找到对应的虚拟机之后，查询其对应的提供端 IP，向提供端发送开启虚拟机的请求。具体的请求格式将在提供端详述。如果一切正常，提供端会发回一个确认信息。

步骤 5：将虚拟机的 IP 和端口信息返回给客户端，由客户端自主去连接虚拟机。

3. 关闭（休眠）虚拟机

关闭虚拟机是比较简单的一个步骤，基本与开启虚拟机相同。只不过发送的指令是：

事件：CloseVm(SleepVm)　虚拟机名称：VmName

其后的操作与开启虚拟机基本对应。

除了以上操作外，为了适应虚拟组织中共享资源的要求，增加了分享虚拟机的操作。虚拟机的分享则不需要虚拟机提供端的参与，仅仅在社区中心服务器修改数据库，增加一条虚拟机续虚拟组织的对应关系记录即可。

3.2.3 虚拟机提供端实现

虚拟机提供端程序使用 C++语言编写完成。虚拟机提供端在整个系统中扮演了一个资源提供者或者资源池的角色。在一个完整的物联网系统中，单一的虚拟机提供端将由一系列的资源提供端，电器终端等代替。

虚拟机提供端设计如下：

SupporterConnection 类，该类负责处理社区中心服务器发送过来的请求。

VmControl 类，控制本台计算机上的所有的虚拟机的操作。

Vm 类，该类是一个虚拟机的抽象，其中包含了开启，关闭虚拟机等操作接口，因此在将虚拟机替换为多资源的时候，该类将继续抽象为更基本的元素类，代表一个可分配的元素。

SupporterConnection 类负责接收服务器发送过来的请求并处理，而请求主要有以下几种情况。

1. 新建虚拟机

   在接收到新建虚拟机的请求之后，提供端首先检查本地资源是否允许继续新建虚拟机。在这一步，应该是检查用户的需求和现在的提供能力之间是否匹配，但是目前为了简化系统，所有的虚拟机被定制为相同的大小，享有相同的资源。因此不需要针对用户的请求进行检查而只是操作 VmControl 类，新建一个虚拟机。

2. 打开虚拟机

   调用 VmControl 类方法，打开对应的虚拟机，并返回确认信息。

3. 关闭虚拟机

   调用 VmControl 类方法，关闭对应的虚拟机。

由上面三点可见，虚拟机提供的核心操作都封装在 VmControl 中。下面介绍 VmControl 类的工作原理。

该类负责管理本地所有的虚拟机资源，而每个虚拟机都被抽象成一个 Vm 类，所以 VmControl 中包含了 Vm 对象的一个序列，同时记录了每个虚拟机的名称和对象的对应关系。当中心服务器发送请求开启或关闭虚拟机时，会先将名称对应到相应的 Vm 对象上，然后调用其开启关闭的方法。而创建虚拟机则比较复杂，由于安装系统所需要的时间比较久，而且安装好的系统不经过配置也很难使用，我们采用的解决办法是首先安装配置好一个干净的系统，将其称为 baseVm，当用户需要创建新的虚拟机的时候，使用 VirtualBox 的复制虚拟机的方式复制一个相同的虚拟机出来。

步骤 1：根据本地的命名规则，创建一个新的虚拟机，并在 virtualbox 的虚拟机列表中注册。

命令：”VBoxManage createvm –name $VmName –register”

步骤 2：使用 VBoxManage 中的 CloneHd 命令复制一个虚拟机磁盘出来并在 virtualbox 的资源池中注册。

命令：”VBoxManage clonehd $oldhdname $newhdname –remember”

步骤 3：设置新建的虚拟机，增加 IDE 控制器，设置内存，显存，远程端口等信息。

命令 1：”VBoxManage modifyvm $vmname --ostype Ubuntu --memory 386 --vram 12 –vrdpport $port”

命令 2：”VBoxManage storagectl --name \"IDE Controller\" --add ide”

步骤 4：将复制好的虚拟机磁盘文件挂载到 IDE 控制器上。

命令：“VBoxManage storageattach --storagectl \"IDE Controller\" --port 0 --device 0 --type hdd –medium $hdname”

经过以上四个步骤，一个与原虚拟机基本完全相同的虚拟机就复制出来了，如果要针对用户的需求做不同的定制，只需要准备几个不同版本的虚拟机系统以供复制即可。

第4章 系统评估和未来展望

4.1 系统评估

在系统搭建完成之后，我们对系统进行了功能测试，即远程创建，登录，关闭虚拟机以及在虚拟组织中共享虚拟机的测试。

测试环境：

系统：Ubuntu10.04，内存 1.5G，CPU 双核 1.6MHz。

软件：VirtualBox 版本 3.1.6。

我们所实现的系统为三端的软件程序，但为了方便测试，我们将三个程序都在同一台实际计算机上运行，但是在运行环境上三部分程序完全分开仅仅通过本地网络回环（127.0.0.1）进行通信，因此完全可以模拟实际上分布在三个不同地点并通过网络相连的三部分。

4.1.1 客户端评估

客户端程序运行时，首先读取配置文件，配置文件中包含证书的用户名密码，以及中心服务器的 IP 和端口，IP 选择实际社区中心服务器的 IP，端口则为预先规定好的 8889 端口。实验环境下配置文件如下：

```
[TERMINAL]
ServerIP=127.0.0.1:8889
admin=alice
password=123456
```

客户端程序读取该配置文件后，自动读取名称为 Bobcert.pem 的证书，并用该证书的身份信息与服务器进行通讯，该证书中用户的 commonName 是 alice。

客户端为命令行的操作方式，首先要配置运行时的环境变量，由于目前的程序还在测试阶段，需在开始程序时手动设置，在未来的正式版本的系统中，将无需手动设置。

在命令行下输入

export ELOPHOME=/home/chenwei/project/elop

该命令将设置 ELOPHOME 这个全局变量为后面的路径，该路径需设置为实际的程序文件所在文件夹。

然后在命令行下输入：./Terminal

客户端程序就顺利启动起来了，会出现一个命令提示符，此时可以输入各种控制命令，如果对这些命令不熟悉，可以输入 help 命令进行提示，如下图：

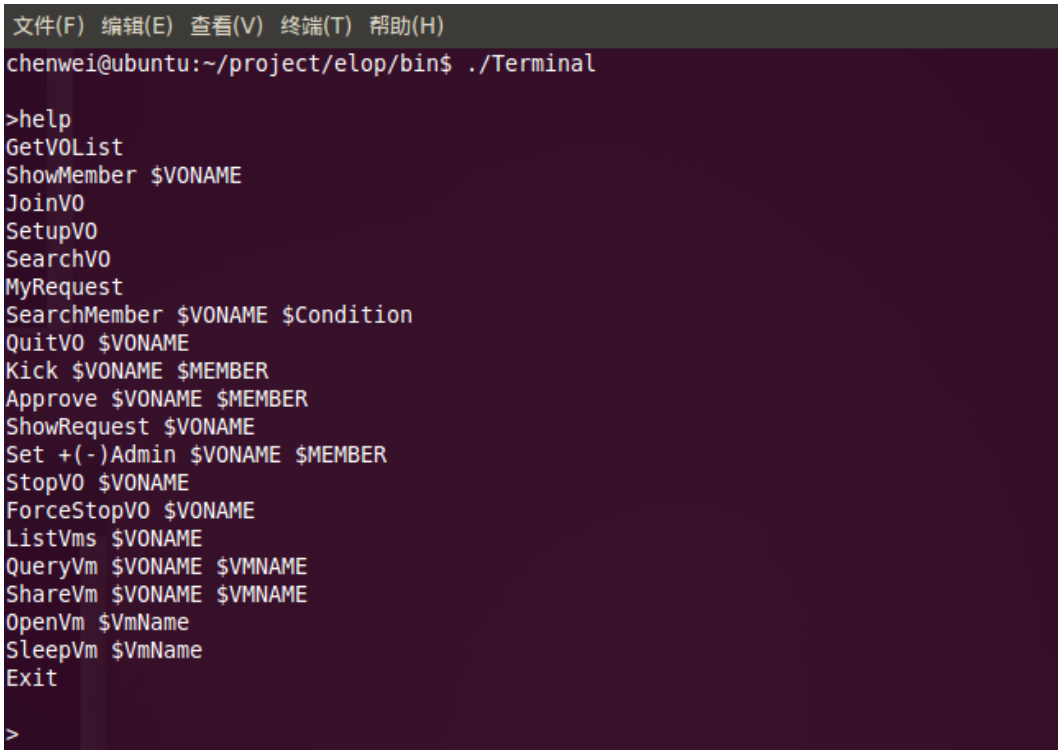

图4.7　客户端程序运行界面

通过提示的这些命令，我们可以完成各种与虚拟组织以及虚拟机相关的功能。比如在社区中心没有任何虚拟组织及虚拟机注册的空白状态时，我们输入 GetVOList 以及 ListVms 都将显示没有记录。如图 4.2：

图4.8 客户端程序界面（服务器中无记录）

此时我们可以自行创建 VO，输入命令 SetupVO，系统将会提一些问题，按照提示进行回答，并最终确认提交，系统就会处理该创建请求，一般情况下请求可以通过，一个虚拟组织就创建完成了，此时提交请求者就自动成为了该虚拟组织的创建者。如图 4.3：

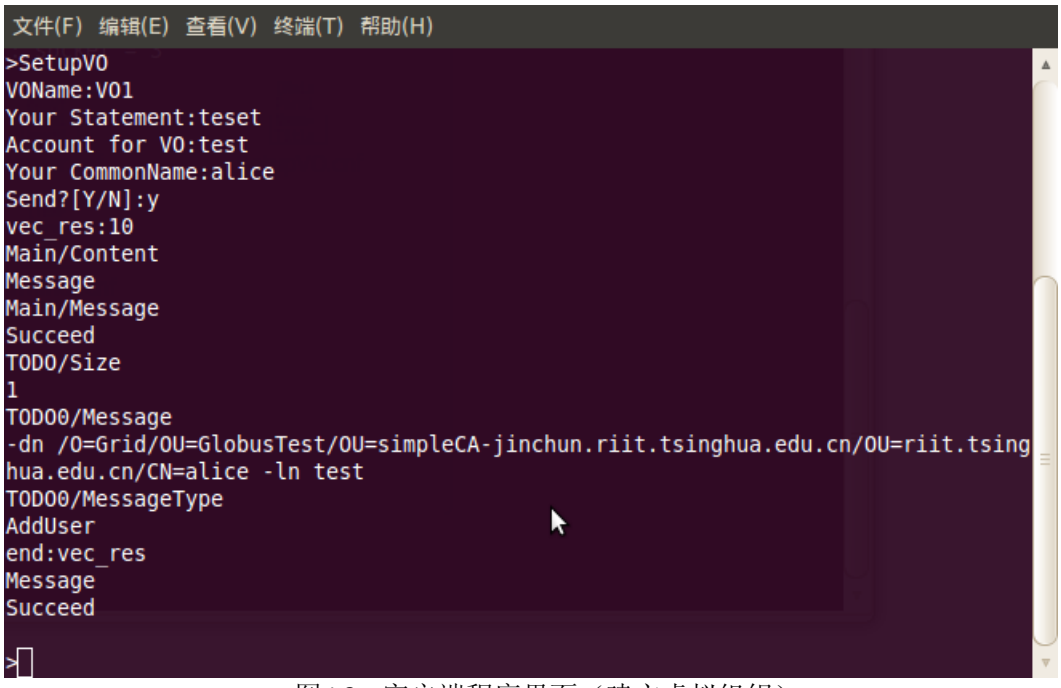

图4.9 客户端程序界面（建立虚拟组织）

下面我们为该虚拟组织添加一台虚拟机，由该虚拟组织的正式成员（包括创建者）提交请求，命令为 QueryVm，此时服务器在收到请求后返回确认信息。

创建虚拟机成功之后，可以尝试开启该虚拟机，使用命令 OpenVm，在等待几秒钟后，本地将会开启一个 rdesktop 程序访问远程的虚拟机。如图 4.4：

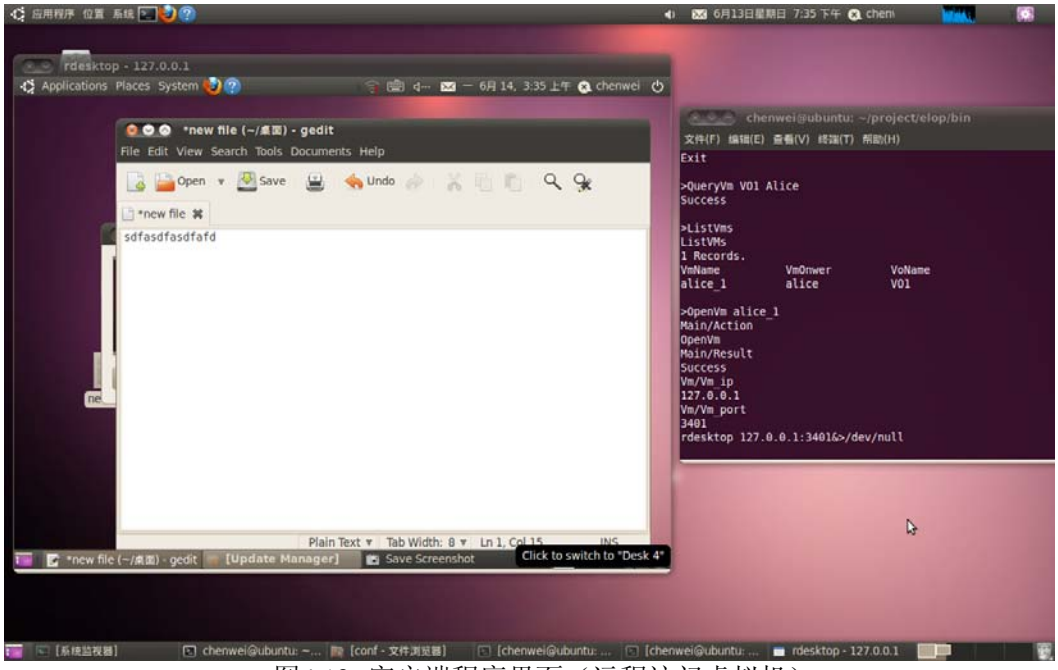

图4.10 客户端程序界面（远程访问虚拟机）

下面来看一下多个用户在虚拟组织的管理之下的一些特性。

我们保留前面一个用户 alice 建立的虚拟组织，然后使用另一个用户 bob 登陆客户端，首先他尝试打开 alice 建立的虚拟机，系统将会返回错误信息，因为 bob 不属于 alice 建立的那个虚拟组织，无法使用该虚拟机。如图 4.5

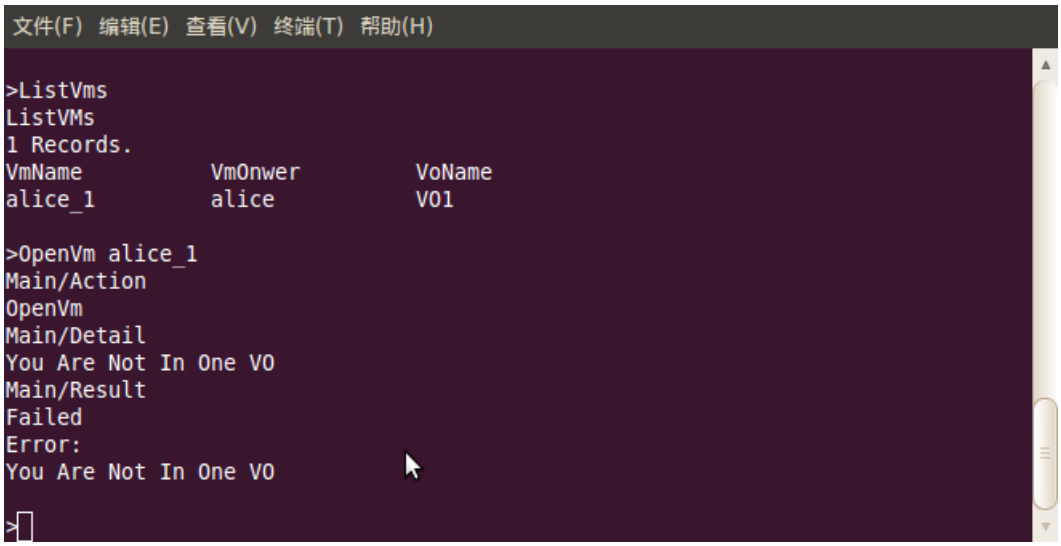

图4.11 客户端程序界面（无法打开虚拟机）

Bob 此时申请加入 VO1，提交申请后使用 alice 登录批准该申请，再次用 bob 登录，此时 bob 已经是 VO1 组织中的正式成员，再次尝试开启虚拟机则可以成功。如图 4.6。

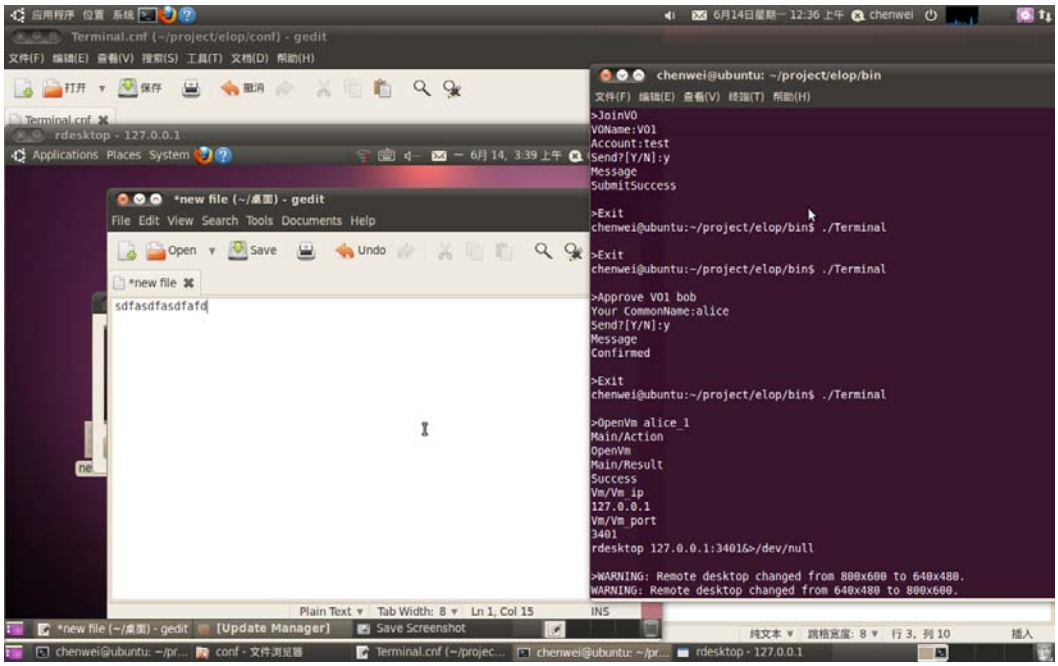

图4.12 客户端程序界面（加入虚拟组织并开启虚拟机）

然后我们让 bob 退出 VO1，此时他就不能够再使用虚拟机 alice_1，然后 bob 自行创建虚拟组织 VO2，并使用 ShareVm 命令将虚拟机 alice_1 共享到虚拟组织 VO2，在目前的系统中系统会自动批准该请求，然后使用 ListVms 命令可以查看目前数据库中的虚拟机情况。如图 4.7。

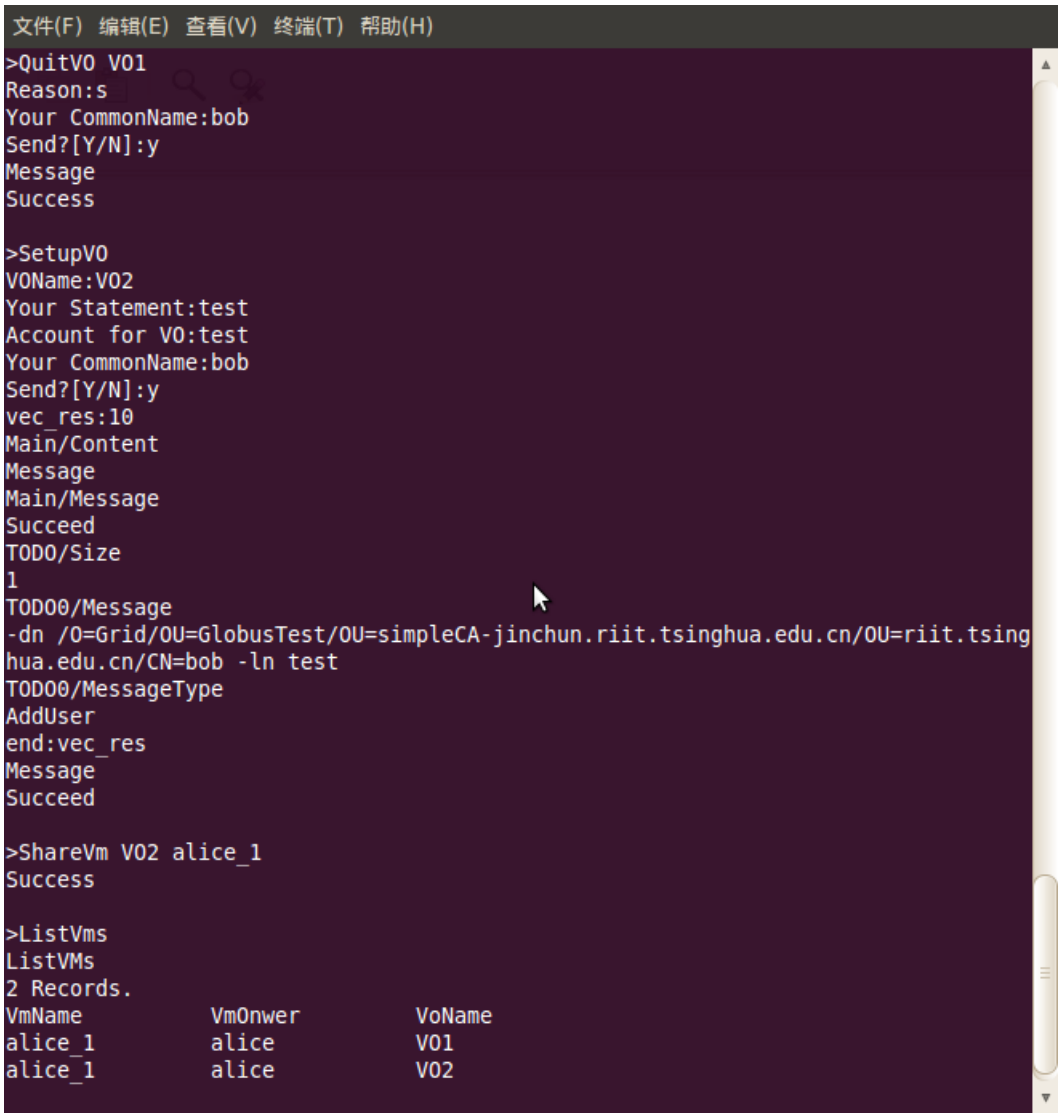

图4.13 客户端程序界面（退出虚拟组织，新建虚拟组织，共享虚拟机）

可以看到，alice_1 虽然只是一个虚拟机，但在记录中保存了两份，因为在两个不同的虚拟组织中的人都可以使用该虚拟机。也就是说，显示的记录并不是虚拟机的列表，而是虚拟机与虚拟组织的对应关系。

使用 OpenVm 命令尝试打开该虚拟机，此时 bob 又能够成功的开启该虚拟机。如图 4.8。

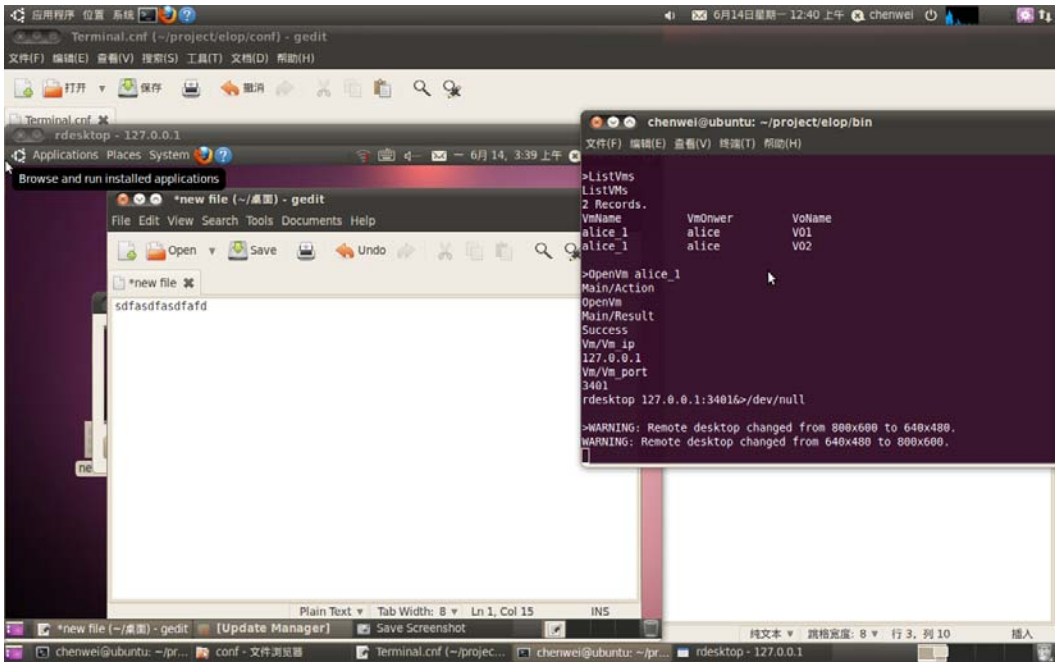

图4.14 客户端程序界面（再次成功打开虚拟机）

上述实验说明，在虚拟组织的管理下，用户可以创建，开启虚拟机，并选择在什么情况下共享该虚拟机。实现了在用户间资源的深度共享。

4.1.2 社区中心服务器评估

社区中心服务器端也有与客户端类似的配置文件，程序在启动后首先读取配置文件，配置文件样例如下：

```
[SERVER]
keyfile=voservercert.pem
admin=voserver
password=123456
port=8889
terminaluser=grrr
terminalpass=123456
supporterport=9890
 [ODBC]
username=root
password=chenwei
dns=SERVERDB
```

[ADDRESS]
globus=
soap=
odbc_ini=
cert=/cert/

这是配置文件的前一部分，SERVER 域中 keyfile，admin，password 与客户端的功能相同，port 是本地监听的端口，该端口必须与客户端配置的 server 的端口一致，否则将无法通讯。接下来的 terminaluser 和 terminalpass 是该服务器用来做发送端时使用的证书用户名和密码。这个主要是用来和虚拟机提供段通讯，要创建虚拟机和开启虚拟机时都是由服务器发起，因此需要使用类似于客户端的程序。

第二个 ODBC 域用来记录本地数据库的一些信息，数据库有用户名密码，dns 是指选择的哪一个数据库。用这些信息 ODBC 就可以打开本地的数据库进行读写了。

第三个 ADRESS 域中式用来指明一些运行时所需文件的路径的。例如 cert 就是指证书文件的存放路径，在整体程序目录下的 cert 目录下。

除了上面的这些之外，还针对每一个表有一个配置信息，如下：

[DB_SupporterInfo]
size=4
item0=commonName
item1=IP
item2=VMLeft
item3=Statement
item2default=
item3default=
item0default=
item1default=

这个就是 SupporterInfo 的配置信息，我们把每个表的列信息存在该配置文件中，就可以在程序里不涉及具体的数据库信息，从而提高通用性。我们的程序一共使用了六个表，因此有六段如上形式的配置信息。

在读取完配置文件后，程序会自动打开监听端口进行监听，收到客户端发来的信息就会自动进行处理。

等待状态（如图 4.9）：

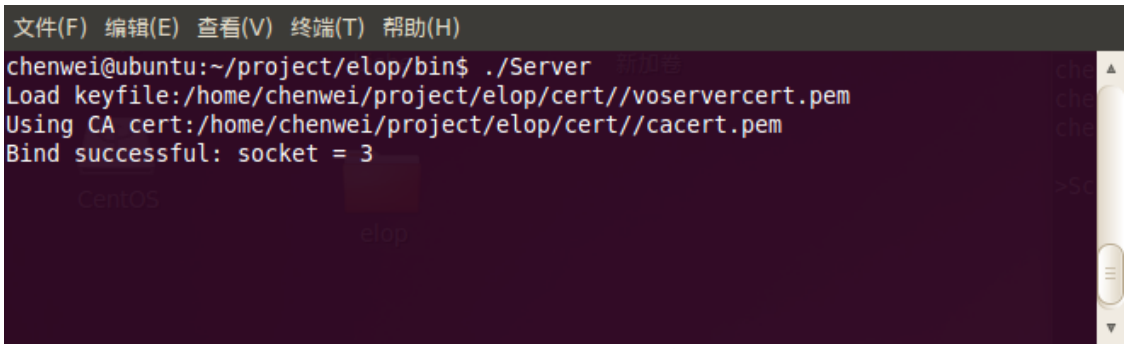

图4.15 社区中心服务器程序界面

社区中心服务器在收到用户的请求后，会调用相关函数并做处理，然后再上图的界面中输出调试信息。

4.1.3 虚拟机提供端评估

虚拟机提供端的配置文件如下：

```
[Supporter]
basevmcount=1
basevm0name=ubuntu-1
basevm0port=3391
[ADDRESS]
cert=/cert/
[SERVER]
keyfile=Tomcert.pem
admin=Tom
password=123456
port=9890
```

后面两个 ADDRESS 和 SERVER 域中的信息与前面服务器端的含义基本相同，Supporter 域中的信息则和虚拟机的建立有关，里面指明了用来复制其他虚拟机的原始虚拟机的信息。

Basevmcount 是指这种原始虚拟机的数量，目前只设定了一台，如果需要支持用户对不同操作系统虚拟机的要求，将需要多个原始虚拟机。Basevmname 就是原始虚拟机的名称。该名称是指在 VirtualBox 中注册的虚拟机的名称，而不是前

面所述的在社区中心服务器的数据库中所存的虚拟机名称，VirtualBox 中的虚拟机名称对于客户端和服务器端都是不可见的，因此只要本地保证不重复就可以。

程序运行后，等待界面和服务器端基本相同，在收到服务器的请求后，也会在后台执行指令并输出调试信息。

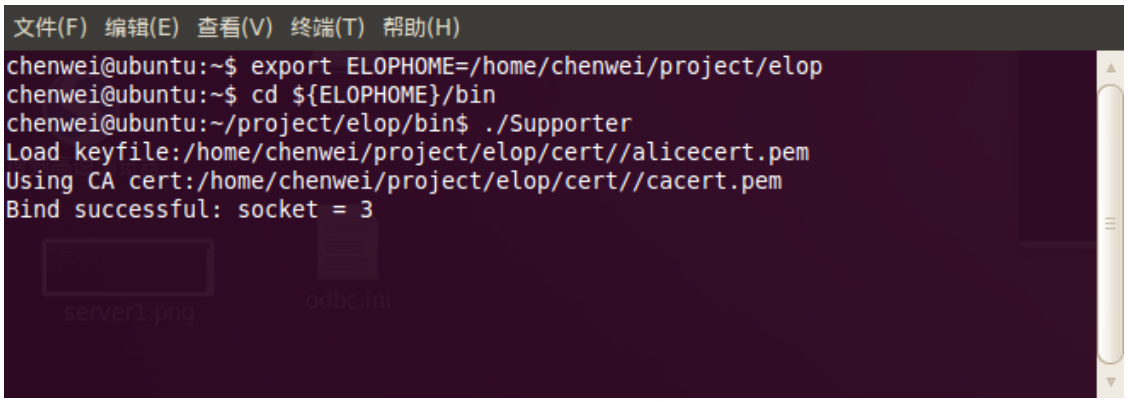

图4.16 虚拟机提供端程序界面

由于虚拟机的提供端将会需要同时开启多台虚拟机，我们来测试一下同时运行的虚拟机对系统的要求和对性能的影响。

图 4.11 为不运行任何虚拟机时系统资源使用情况，可以发现此时内存使用量在 20%左右，CPU 占用量也在 20%以下。

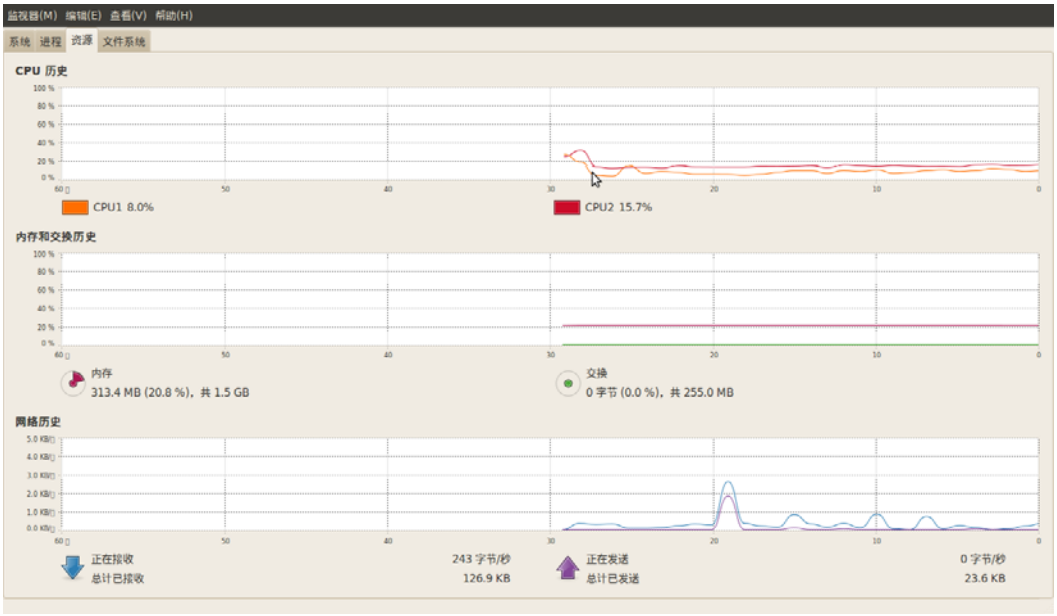

图4.17 系统监视器（空闲状态）

然后我们使用 VirtualBox 打开一个 WinXP 虚拟机，再次检查系统资源使用情况。如图 4.12。

图4.18 系统监视器（窗口方式开启一个虚拟机）

此时系统占用了迅速增加，内存使用占到了 60%。

当我们开启了三个虚拟机的时候，系统资源几乎耗尽，对输入设备响应变慢。系统资源使用情况如图 4.13：

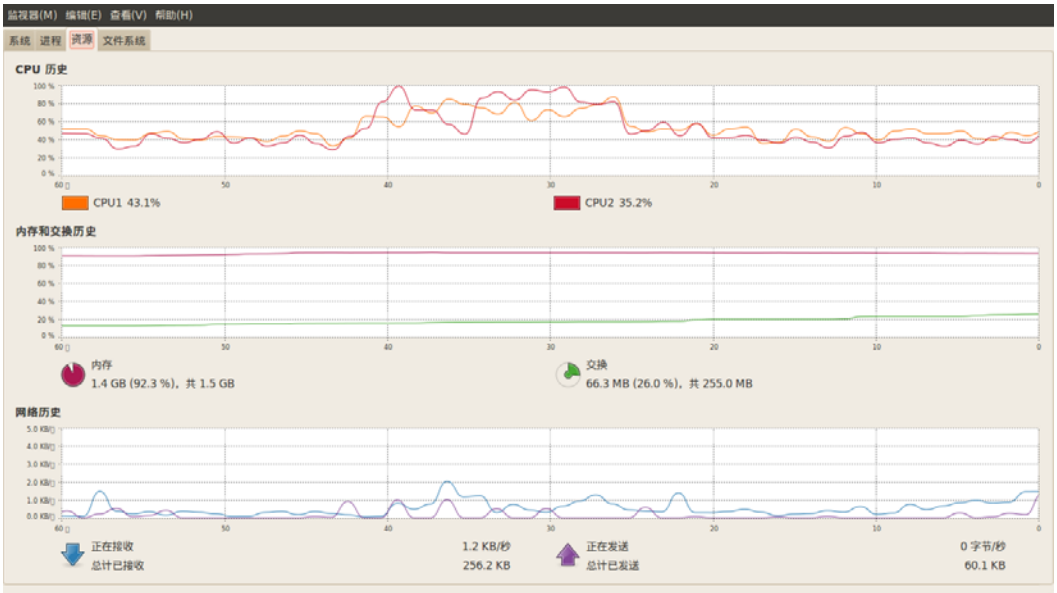

图4.19 系统监视器（窗口方式开启三个虚拟机）

此时内存几乎全部耗尽，交换分区也被大量使用。系统对外部输入基本无反应。

然后我们再尝试使用 VBoxHeadless 模式启动虚拟机，查看资源消耗情况，此时本地将不显示虚拟机的界面。如图 4.14

图4.20 系统监视器（无头模式开启一个虚拟机）

此时内存占用在 50%左右，相比于使用图形界面开启一个虚拟机时要更省资源。

开启三个虚拟机之后资源占用情况如图 4.15：

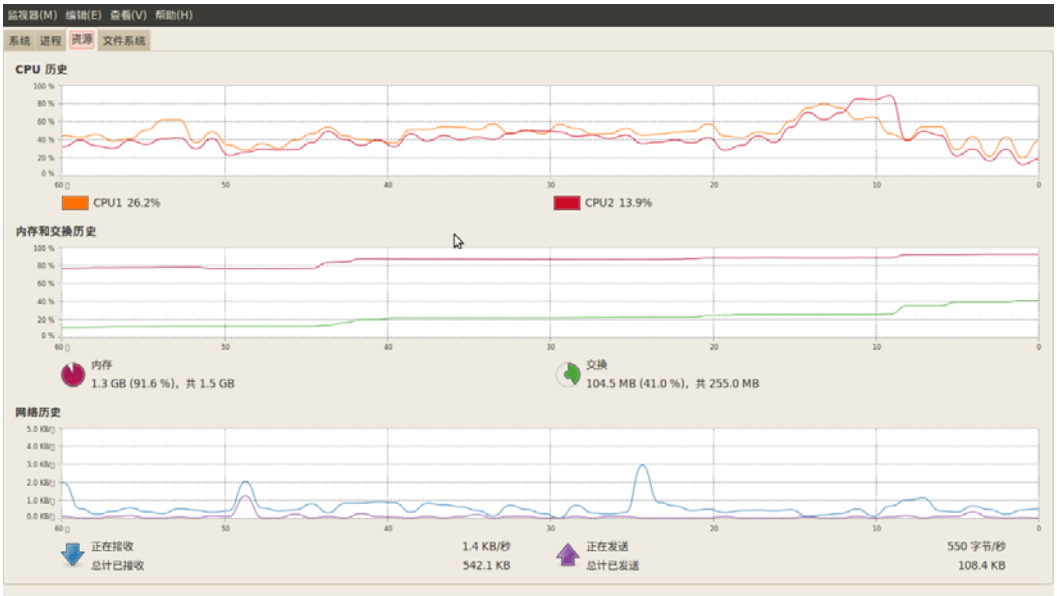

图4.21 系统监视器（无头模式开启三个虚拟机）

可以发现，用 VBoxHeadless 开启三个虚拟机时系统资源消耗相比于使用图形界面的要少，此时计算机对于输入仍然能够做出较快响应。

因此可以得出初步结论，使用 VBoxHeadless 模式的确有利于节省本地资源，在有限的资源下开启更多的虚拟机。更适合于在服务器中提供虚拟机服务时使用。

4.2 未来展望

该系统目前仍不够完善，只是在框架下完成了一个小功能的演示，作为一个基于虚拟社区的物联网使能环境，我们还需要完成的主要工作有：

1. 做好物联接口，目前我们都是在软件层面上进行测试，而要做好物联网的环境，必须规定好软件与物理系统之间的接口。
2. 扩展资源的概念。在 ELOP 软件包中，虚拟机是作为一个元素来处理的，而在之后的系统中，很关键的一点就是元素的种类将需要大大扩展。比如家用电器，或者一个显示器，或者仅仅是一个灯泡，我们将使用统一的方式来进行管理。
3. 提供一个移动终端的访问方式。我们的系统的一大优势就在于，我们只要登录到服务器就可以享受到后台的巨大的资源库，本地可以几乎没有计算能力，只要有一个操作和显示的界面。因此一个手持的设备将是一个很好的选择。它要有良好的操作体验和显示效果，并且有稳定的网络连接，而不需要强大的计算能力。

   图 4.16 所示的是实验室之前开发的一个简单的触屏设备，有网络连接的功能，但是这个设备太大而且效果并不理想，我们将需要开发一个更加便携的手持设备来完成这一功能。

图4.22 嵌入式手持终端

4. 扩展程序的应用环境。目前我们的开发都是基于 linux 系统，而以后为了能够是该系统更加实用，我们必须让它能够运行在各种系统各种平台上，包括上面所提的嵌入式设备中。

插图索引

图 1.1　云计算概观.....4
图 1.2　云计算三种服务模型.....4
图 1.3　虚拟化技术示意.....7
图 2.1　VirtualBox 虚拟机图形操作界面.....16
图 2.2　ELOP 框架示意图.....21
图 3.1　系统整体框架.....23
图 4.1　客户端程序运行界面.....30
图 4.2　客户端程序界面（服务器中无记录）.....31
图 4.3　客户端程序界面（建立虚拟组织）.....31
图 4.4　客户端程序界面（远程访问虚拟机）.....32
图 4.5　客户端程序界面（无法打开虚拟机）.....33
图 4.6　客户端程序界面（加入虚拟组织并开启虚拟机）.....33
图 4.7　客户端程序界面（退出虚拟组织，新建虚拟组织，共享虚拟机）34
图 4.8　客户端程序界面（再次成功打开虚拟机）.....35
图 4.9　社区中心服务器程序界面.....37
图 4.10　虚拟机提供段程序界面.....38
图 4.11　系统监视器（空闲状态）.....38
图 4.12　系统监视器（窗口方式开启一个虚拟机）.....39
图 4.13　系统监视器（窗口方式开启三个虚拟机）.....39
图 4.14　系统监视器（无头模式开启一个虚拟机）.....40

图 4.15　系统监视器（无头模式开启三个虚拟机）.....40
图 4.16　嵌入式手持终端.....42

参考文献

[1] 棠棣：ITU 互联网报告 2005：物联网[EB/OL] . http://chiong.cn/2009/10/itu-internet-report-2005-the-internet-of-things-1/

[2] 曹军威: 赛百平台及其技术挑战[J]. 国际学术动态: 华中科技大学, 2010 年第二期: 38-42

[3] Kai Hwang, Albert Zomaya, Jack Dongarra (2010). *Cloud System Architecture and Datacenter Design in Distributed Systems and Cloud Computing: unpublished work*

[4] Ian Foster, Carl Kesselman, Steven Tuecke (2001). *The anatomy of the grid.* Euro-Par 2001 Parallel Processing

[5] 虚拟化-维基百科 [EB/OL] http://zh.wikipedia.org/zh/虚擬化

[6] Martin F.Maldonado：虚拟化概述：模式的观点[EB/OL] http://www.ibm.com/developerworks/cn/grid/gr-virt/index.html

[7] 荆继武 林璟锵 冯登国：PKI 技术(信息安全国家重点实验室信息安全丛书) [M]北京：科学出版社 2008

[8] Peter Gutmann (2002). *PKI: It's Not Dead, Just Resting.* Computer Volume35, Issue 8(Augest 2002) 41-49

致 谢

感谢我的导师曹军威老师，他在我的论文编写过程给了我极大的帮助。在选题开题，学术研究，项目编程开发，论文完成等各个方面都离不开曹老师的悉心教导。曹老师在学术态度，工作态度上积极认真也极大地影响了我，让我能够顺利的完成项目的研究开发。

还要感谢王震师兄，万宇鑫师兄在我的论文编写和项目开发方面给予的极大帮助，项目的顺利进展离不开他们的帮助。

同时还要感谢在论文编写过程中给了我很多帮助的实验室的其他师兄师姐，以及与我共同开发此项目的同学。

声 明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____　　日　期：_____

附录 A 外文资料的调研阅读报告（或书面翻译）

第一章：分布式和云计算系统模型[1]

1.1 规模可变的计算到大规模并行计算

在过去的 60 年里，计算的形势经历了一系列的平台和环境的变化。我们重新审视在机器的设计制造，操作系统平台，网络连接，应用程序计算量等方面的发展变化。一个并行分布式计算系统并不是使用一个中心的计算机来解决计算问题，而是使用多个计算机通过 Internet 进行合作解决大规模的问题。分布式计算是数据密集，以网络为中心的计算模式。我们将会发现现代的并行分布式系统的杀手级应用。这些大规模的应用已经对我们文明的各个方面引起了令人瞩目的变化。

1.1.1 从并行计算到分布式计算

电子计算机已经经历了 5 代发展。每一代大概继续了 20 年。相邻的两代会有 10 年左右的重叠期。自 1950-1970 年，极少数主机，比如 IBM360 和 CDC6400，被用来满足大型商业机构和政府组织的需求。在 1960-1980 年，低成本的小型计算机，像 DEC 的 PDP11 和 VAX 系列，在小型商业机构和大学校园里普及起来。在 1970-1990，使用 VLSI 微型处理器制造的个人计算机开始广泛应用。在 1980-2000，大型的可移动计算机和各种设备开始伴随着有线和无线应用出现。从 1990 年开始，我们开始被隐藏早云网络中的大规模的并行分布式计算系统覆盖。在现代社会中，他们为大众提供了 web 级服务以及实现了决定性的任务。

并行计算的级别：

在我们继续关注计算发展的趋势前，让我们首先回顾一下并行计算的类型。在硬件非常庞大以及昂贵的 50 年前，大多数计算机都被设计为以“位”为单元的方式。位级别的并行计算慢慢将“位”为单元的运算处理改变为“字”级别的处理。在这些年里我们的 CPU 经历了 4 位，8 位，16 位，32 位和 64 位的发展变化。下一步发展变化将是指令级的并行计算（ILP）。再过去的 30 年里当我们从让处理器每次同时只处理 1 个指令改变为每次同时处理多个指令时，我们已经通过流水线技术，超大规模技术，VLIW（长指令字）技术，和多线程技术实现了 ILP。ILP 需要分支预测，动态规划，推测和高级别的编辑器支持来使它有效地工作。

数据级的并行计算（DLP）因为 SIMD（单指令多数据）和使用数组或向量方式存储指令的向量机而流行起来。DLP 需要更多的硬件支持和编辑器协助。自从多核处理器和片上多处理器的引入，我们开发出了任务级的并行计算（TLP）。一个现代的处理器包括了以上的所有并行方式。先进的硬件和编译器都对 BLP，ILP，DLP 有良好的支持。然而，TLP 距离成功还有很远的距离，这主要是因为在多核处理器以及片上多处理器上进行编程和高效率代码的编译有相当难度。当我们从并行计算转移到分布式处理上来时，我们将会发现计算的粒度发展到了工作级并行（JLP）。应该说粗粒度的并行计算是建立于于细粒度的并行计算上的。

计算发展趋势：

大体上计算趋势是越来越依赖于网络上的共享资源。如图 1.1，我们看到两条系统的发展轨迹：分布式计算系统（DCS）和高性能计算系统（HPC）。在 HPC 一边，同样的超级计算机（超大规模并行处理器，MPP）正在慢慢的被出于分享计算资源需求而产生的合作计算机群取代。这个计算机群经常是一系列物理上紧密连接的计算机节点。集群和 MPP 将会在第三章和第八章讲到。在 DCS 一边，由于分布式文件共享和内容递送应用的需求而产生了 P2P 网络。一个有很多客户机组成的 p2p 网络将会在第 5 章和第 8 章学到。其中的客户机在世界范围内广泛分布。

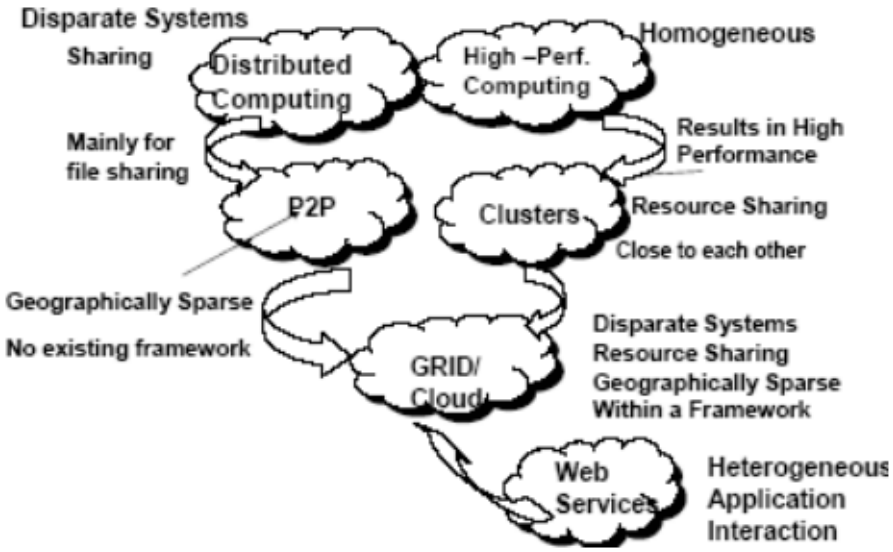

Figure 1.1 The evolutional trend towards web-scale distributed computing and integrated web services to satisfy heterogeneous applications.

分布式计算家族

自从 90 年代中叶，建造 P2P 网络以及集群网络的技术就被纳入了很多国家项目，目标是建成网域的计算设施，被称作计算网格或数据网格。我们将会在第 4 章接触到网格计算技术。最近又掀起了一股研究网络云资源来进行网络尺

度的超级计算的热潮。网络云计算是从桌面计算到服务性计算的转变的必然结果。服务型计算使用服务器集群和大型数据库组委数据中心。我们在 1.2 将介绍多种并行和分布式计算家族。网格和云计算是不同的系统，他们强调在硬件，软件以及数据上共享的资源。

这些大型的分布式计算系统的设计理念，实现技术，和案例研究在本书中将涉及到。大型分布式系统将致力于开发一些用于很多计算机的高等级的并行技术或协作技术。在 2009 年，目前最大的计算集群拥有 224162 个处理器核心。这个最大的计算网格连接了各地的数十到数百个机器的服务集群。一个典型的 P2P 系统可能同时包括了到数百万个客户机。试验用的有数千个处理节点的云计算集群已经建成。最近几年的实际系统的案例研究也将会在全书中涉及。

第六章：云系统构建和数据中心设计

概要：云计算通过大型数据中心向用户提供了虚拟资源。这样的动机是提供符合需求的计算服务以及更加有效的升级网络应用。这一章包括了云计算系统构建和数据中心设计的设计原理，实现技术等。我们从数据中心的构建和管理开始，然后将展示云平台的设计原理。包括分层平台设计，虚拟化支持，资源提供，和设施管理。众所周知的云服务模型 SaaS，laaS，PaaS 都是云平台支持下的。我们呈现主要云设施提供商如 Google，Amazon，IBM,Microsoft，Salesforce.com 等提供的云系统的案例研究。这些平台下的服务应用将在第 7 章讲到。

6.1 云计算和服务模型

再过去的二十年中，这个世界的经济中心从制造业转向了服务业。在 2010 年，美国经济的 80%是有服务业提供的而只有 15%是制造业，5%农业。云计算将会在未来的服务业中扮演一个重要的角色。我们已经介绍过云计算的基本概念，接下来，我们将从各个角度学习云计算。这一章重点在云构架和设施设计。下一章将聚焦于云服务和应用的开发。因为虚拟云平台需要构建在数据中心之上，我们将学习到数据中心在支持云计算中的角色和设计。

6.1.1 公共，私有，和混合云模型。

云计算提供了一个虚拟平台，它拥有弹性的动态按需求结合的硬件，软件和数据资源。这是为了完成从桌面计算到使用服务集群和大型数据中心作为数据库的以服务为目的的计算的转变云计算。云计算要求低成本以及对于提供者和用户都足够简单明了。云计算试图通过同时提供很多不同的应用来完成多任务计算。这种计算被传送至数据存储的位置，而不是复制数据到数百万个桌面

计算机。云计算避免了大规模的数据移动，从而完成了更好的网络带宽使用。更进一步的是，机器虚拟化是的在使用云平台的时候更加有性价比。

云计算的概念已经从集群，网格，公共计算的概念以及提供软件服务中发展起来。集群和网格计算完成使用并行的很多计算机来解决大型问题。公共计算和 SaaS 提供了计算资源作为服务，按次收费的概念。云计算提供了将多样的资源将服务递送给终端用户。这是一种新型的计算模式，在这种模式下，设施通过大型数据中心或服务中心提供服务。云计算模型使用户可以在任何位置任何时间使用它们已经连接到网络的设备享受到这些资源。

一些人认为云计算是在数据中心进行的中心化运算。但我们认为云计算实际上是在各个数据中心进行的分布式并行计算。在这种意义下，云平台实际上是分布式系统。图 6.1 像我们展示了三种云的类型，私有，共有，混合，它们部署在 intranet 或公开的 Internet 上。注意到这些云都是在整个因特网上建立的。他们绝不是在某一个地方集中，就像在任何银行系统中有的很多分支办公室。

一个共有云实现在 Internet 上，可以由为此服务付过费用的用户使用。很多公司已经建立了公共云服务，像 Google App Engine，Amazon AWS，Microsoft Azure，IBM Blue Cloud，和 salesforce.com。这些商业提供商提供了一个公共的远程界面来在它们拥有的设施中创建和管理虚拟机。私有云建立在一个 intranet 域中，是一个单独的组织所有。它们不是为了通过 Internet 提供公共界面出售计算能力，而是为本地用户提供一个灵活的私有设施来在他们的许可范围内运行服务。一个混合云是包含私有和共有的云，私有云可以通过外部公共云补充拥有计算能力的本地设施来成为一个混合云。

例如，RC2 是一个 IBM 建立的私有云，RC2 连接了分布在各地 8 个 IBM 研究中心的计算和 IT 资源。这个晕的核心是服务集群，或者虚拟机集群。集群的节点主要是用来作为计算节点。一小部分控制节点用来管理和监视整个云的活动。用户工作的调度需要把工作分配到不同的虚拟集群中。通道节点提供了外部世界服务的接入点。这些通道节点也可以用作整个云平台的安全控制。

这些云模型需要不同级别的性能，数据保护和安全措施。不同的服务级别协议被用来满足提供商和付费用户的需求。云计算开发了很多已经存在的技术。比如，网格计算是云计算的关键，在云系统中，网格在更好的利用研究设施方面和资源共享有共同的目标。网格更加着重于递送存储和计算资源然而云计算重点在于节约抽象的服务和资源。

例如，电子邮件服务可以为外部用户提供用户接口。这个应用可以从网络云服务器比如电子邮件存储器中获得服务。同时也存在一些服务节点是用来支持整个云计算集群正确运行的。这些节点叫做运行时支持服务节点。例如，在

某些特殊的应用中可能有分布式的锁定服务。最后，也可能存在某些独立的服务节点，这些节点为集群中的其他节点提供独立的服务。例如，新闻服务可能需要气象信息，这应该是由某些服务节点提供信息的。

既然云的主要概念就是高性能价格比，我们后面将主要会考虑公共云，除非特别指明。很多可执行的应用程序代比他们要处理的 web 级数据小得多，云计算在执行过程中会避免大规模的数据移动。这会导致网络上更小的流量和更好的网络占用量。云也会减轻 IO 问题。云的性能和他的服务质量已经在越来越多的实际应用中被证明了。我们将会在 6.2.5 节中建立云计算的性能模型包括数据保护，安全措施，服务能力，容错能力和运行代价。

6.1.2 云生态系统和实现技术

由 IBM 建立的世界范围的云服务市场在 2012 年将肯那个达到 1260 亿美元的规模，这包括附件服务，基础设施服务和商业服务。作为服务工厂的网络云搭建在多个数据中心之上。我们敬爱那个会引入云生态系统，代价模型和实现技术。这些对于让读者了解云计算后面的动机和在使云计算服务成为现实的道路上已经解决的障碍很重要。

云设计目标：

尽管围绕由数据中心或大型 IT 公司来进行的集中运算和存储服务代替桌面计算的争论一直没有停止，云计算社区还是在如何让云计算被普遍接受上达到了共识，我们列出以下六个云计算的设计目标：

1．把计算从桌面转移到数据中心：这种计算机处理，存储，软件的转移把桌面和本地服务通过网络移动到了数据中心。

2．供和云生态：提供商与客户和终端用户签署 SLAs 来提供云服务。这种服务必须是节省资源并且拥有高的计算，存储和能量消耗效率。付费模型是基于账单到期支付政策的。

3. 平台和软件以及设施服务的性能必须是可以随着用户的数量可变的。

4. 保护：你是否可以信赖数据中心来处理你的私人信息和记录？这个问题必须解决以保证云能够成功的成为可信赖的服务。

5. 云服务：云计算的服务质量必须标准化以解决用户的疑虑，云在不同的提供商直接必须有通用性。

6. 标准和界面：这是指解决数据在数据中心和云提供商之间的不通用问题。必须有一种普遍接受的 API 和接入协议来支持高度便携和灵活的虚拟应用。

我们将会在在这一章里学习这里面的大部分问题，还有安全和性能的问题留在第七章。下面让我们从分析云生态系统作为开始。

云生态系统和代价模型：

在传统的 IT 计算中，用户必须拥有他们自己的计算机和外设，这是主要的费用。此外，他们还需要面临很多使用和保养计算机系统是的额外费用，包括人工和服务花费。下图（a）向我们展示了在传统的 IT 中，在固定的主要投资之上的可变的运作代价。注意到固定费用是最主要的费用，而随着用户数量的增加比例会有略微的减小。但是运行费用可能会随着用户数量增加而急剧增加。另一方面，云计算提供了每次使用付费的商业模式。用户的工作对于数据中心是额外的。

要使用云则不需要第一笔费用来购买昂贵的机器。如下图(b)中所示，用户只有可变的费用需要承担。综上，云计算无论是对一般小用户还是大型的企业都会降低费用。计算经济学向我们展示出了在传统的 IC 用户和云用户之间的鸿沟。不必首先购买昂贵的机器为开设一家公司减轻了很大的负担。事实上，只需要付运行费用而不必在固定器材上投资对大量的小用户格外有吸引力。这就是为什么云计算对大部分企业和大型机器用户来说显得很诱人。

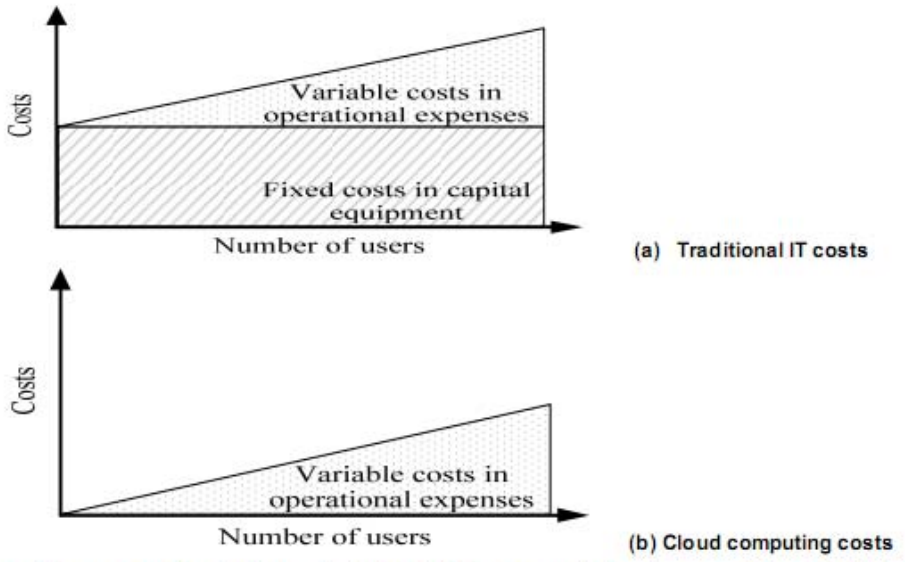

Figure 6.2 Computing economics between traditional IC users and cloud users, where traditional users must acquire expensive computers out front, while the cloud users only pay for the service provide with no major investment on large servers or expensive computer equipment.

建立私有云的云生态系统：

随着网络上各种云的出现，在提供商，用户和技术之间的生态系统开始出现。这个生态系统已经在公共云系统中进化。网络中存在的不断增长的云计算资源使很多组织对于使用自己的网络设施建立他们的 IaaS 产生了浓厚的兴趣。由于公有的云牵扯其中，私有云和混合云并不排外。一个私有或混合云可以允许用户使用网络服务界面远程连接得到他们的资源。

下图像我们展示了 Sotomayer 提出的私有云的生态系统。他们为私有云提出了一个四层的生态系统发展模型：在客户端，顾客需要一个灵活的平台；在云管理层，云管理者通过 IaaS 平台提供虚拟的资源。在虚拟设施管理层（VI），管理员在多个服务器集群上分配虚拟机；最后在虚拟机管理层，虚拟机管理者负责在每个单独的服务机上安装虚拟机。一个云工具的生态系统试图覆盖从云管理到虚拟设备管理各个方面。在一个已经存在的虚拟设备管理层基础上把云管理解决方案综合起来会很困难，因为在两者直接缺少一个公开的标准的接口。

很多小的云提供商会出现在大型 IT 产业周围，例如 GoGrid，FlexiScale 和 ElasticHosts。越来越多的刚成立的公司会在云资源的基础上构建他们的 IT 策略。这样便不需要或很少需要管理他们的 IT 设施。我们需要一个灵活的公开的架构，可以让各个组织构建自己的私有或混合云系统。VI 管理是面向终端的，例如 VI 工具包括 Ovirt，VMware vSphere 和 Platform VM Orchestrator。这些工具支持动态的配置，并支持在一个物理资源池中自动平衡的分配负担，服务器的联合以及基础设施的度量和划分。

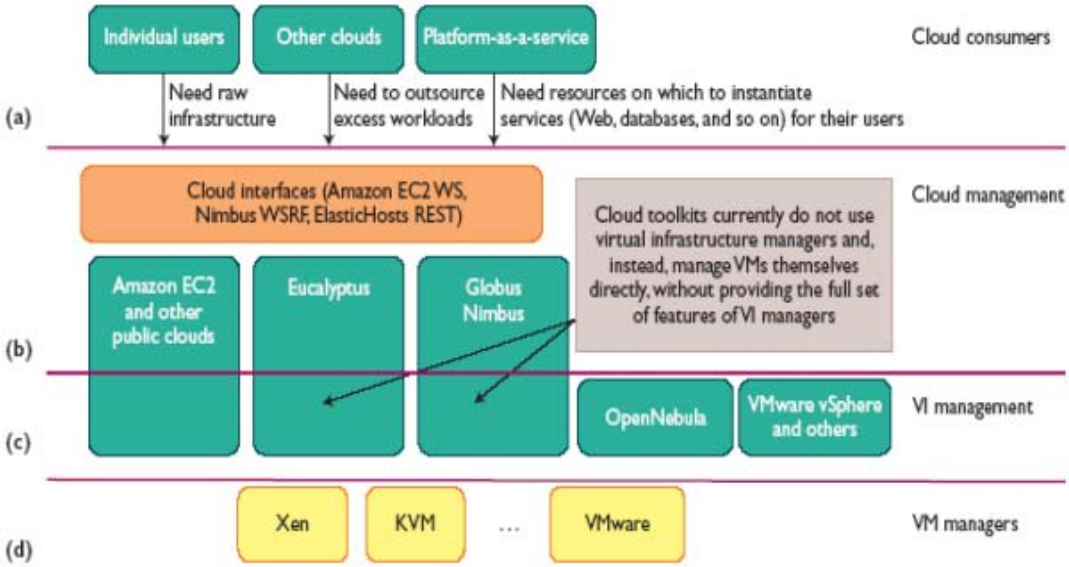

Figure 6.3 Cloud ecosystem for building private clouds. (a) Consumers demand a flexible platform. (b) Cloud manger provides virtualized resources over an IaaS platform. (c) Virtual infrastructure (VI) manager allocates VMs to server clusters. (d) The VM managers handle VMs installed on individual servers. (Courtesy of Sotomayor, Montero, and Foster, *IEEE Internet Computing*, Sept. 2009. [71])

6.1.3 流行的云服务模型

通过云系统提供的服务基本可以分为三种不同的模型：IaaS，PaaS 和 SaaS。这三种模型都允许用户从网络中获得服务，完全依赖于云服务提供商的设施。这些模型基于不同提供商和用户之间的协议（SLAs）。在一般意义上，云计算

的 SLA 签署是为了保证服务性能，数据保护和安全方面。这三个云模型在下图中进行了举例说明。

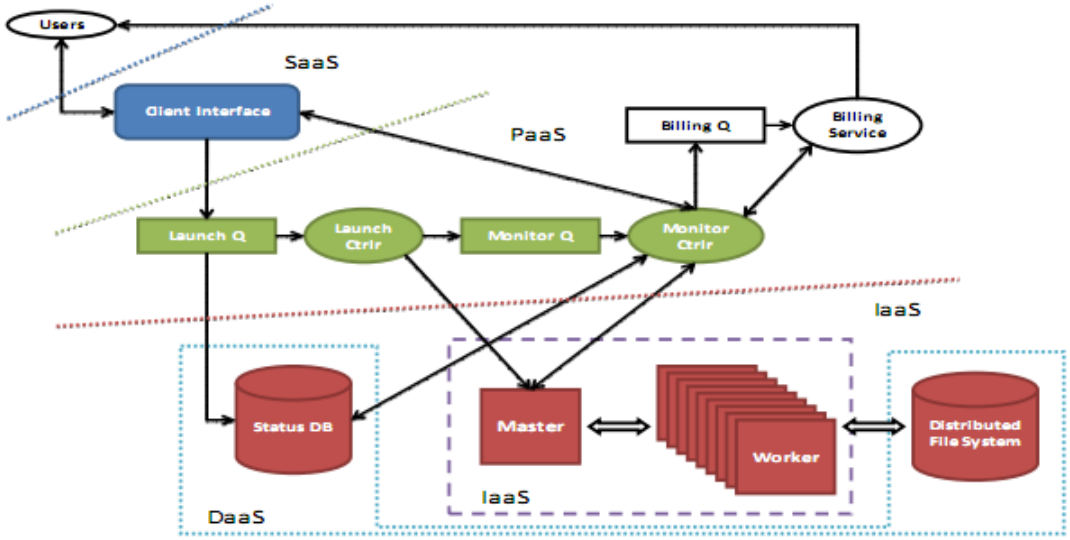

Figure 6.4: Anatomy of cloud service models: The IaaS provides virtualized infrastructure at user's costs. The PaaS is applied at the platform application level. The SaaS provides specific software support for users at web service level. DaaS (Data as a Service) applies the status database and distributed file system. Some providers consider DaaS an integral part of the IaaS.

Infrastructure as a Service（IaaS）：

这个模型允许用户租用处理，存储，网络和其他一些资源。用户可以部署和运行自己的 OS 和应用。用户不管理或者控制底层的云设施，但可以控制操作系统，存储设施，部署应用程序，还有可能可以选择网络组件。这个 IaaS 模型包括了存储，计算资源，通信资源作为服务。这一类型的服务有：Amazon-S3 提供存储服务，Amazon-EC2 提供计算资源，Amazon-SQStigong 通信资源。

Platform as a Service(PaaS):

这个模型允许用户在云设施上部署用户建立的应用程序，这些应用程序必须是使用提供商支持的语言和软件工具编写的。用户不需要管理底层的云设施。云提供商在一个良好定义的服务平台上帮助建立整个应用程序的开发，测试和运行。这个 PaaS 模型使得世界各个不同位置的开发者有可能在同一个平台上共同开发软件。这个模型的其他服务方面包括第三方软件管理，整合以及服务监控的解决方案。Google App Engine 和 Microsoft Azure 是这个模型的两个例子。

Software as a Service(SaaS)：

这个模型是指基于浏览器的应用程序，它运行于数千个云客户之间。SaaS 模型把软件应用程序作为服务，而不是让用户购买软件包。因此，在用户来看，不需要对为购买软件许可或服务进行一次性的预投资。在提供商来看，相比于传统的作为用户应用程序的伺服机代价更加低廉。客户的数据存储在云中，既

有可能是卖主所有的，也有可能是公共的云平台如 PaaS 和 IaaS。大量的商业软件被作为服务提供。成功的案例有微软的 online sharepoint 和 SalesForce.com 的 CRM software。

为了帮助用户认识一些企业的云应用系统，我们将会讲述三个世纪的云应用，分别于 HPC，新闻媒体和商业交易相关。使用云服务的优势在这些应用中一目了然。

（1） 为了通过 DNA 序列分析发现新型药物，Eli Lily 公司使用了亚马逊的 AWS 平台的服务。它们连接了高性能的生物分析机器而不需要购买昂贵的超级计算机，这个 IaaS 的应用大大减少了药物的开发时间，并节约了大量的资金。

（2） 另一个很好的 例子是 New York Times 使用亚马逊的服务来从数百万的档案文件和新闻报纸中重新寻找有用的信息。纽约时报用这个系统节约了时间和花费，并使工作变得系统化，因而从中获利巨大。

（3） 第三个例子是 Pitney Bowes，一家电子商务公司。他们通过 Microsoft Azure 平台为他们的客户提供了完成 B2B 交易的机会，他们还是用了.net 和 sql 服务。最终，他们在客户数量上有了显著的增长。

参考文献（或书面翻译对应的原文索引）

[1] Kai Hwang, Albert Zomaya, Jack Dongarra (2010). *Cloud System Architecture and Datacenter Design in Distributed Systems and Cloud Computing: unpublished work*

综合论文训练记录表

学生姓名		学号		班级	
论文题目					
主要内容以及进度安排	指导教师签字： 考核组组长签字： 年　　月　　日				
中期考核意见	考核组组长签字： 年　　月　　日				

指导教师评语	指导教师签字：_____ 年　　月　　日
评阅教师评语	评阅教师签字：_____ 年　　月　　日
答辩小组评语	答辩小组组长签字：_____ 年　　月　　日

总成绩：_____

教学负责人签字：_____

年　　月　　日