面向一体化桌面云架构的资源调度与管理策略①

丁 鼎[1]，何 进[2]

[1](北京邮电大学 计算机学院，北京 100876)
[2](北京电信规划设计院 第六设计所，北京 100048)

摘 要：桌面云作为运营商 IT 架构云化的切入点，已在小规模试点工程中取得了成功，但此种创新类项目大规模推广会对现有技术架构及管控体系产生较大冲击，因此在实际应用中，亟需解决如何保障用户体验，实现平滑过渡等问题。立足于运营商营业厅类生产任务型场景，分析了传统架构在大规模推广条件下的种种不足，建立了一套适应未来发展的一体化桌面云架构，并以用户体验为核心，对传统架构中资源调度与管理策略进行了优化。

关键词：桌面云；一体化；资源调度；虚拟化；Credit

Strategy of Resource Scheduling and Management in Integrated Desktop Cloud Architecture

DING Ding[1], HE Jin[2]

[1](School of Computer Science, BUPT, Beijing 100083, China)
[2](Beijing Telecom Planning & Designing Institute Co.Ltd, Beijing 100048, China)

Abstract: Desktop Cloud, as the beginning of clouding on operator IT Architecture, has already succeed in small-scale pilot project, however Desktop Cloud, as the project of innovation, will impact on existing technology infrastructure and control systems, and how to protect the User Experience is one of the most important issues in the course of large-scale promotion. This paper, concerning operator task oriented scene, such as businesshall, analysed the shortcomings of traditional architecture, established integrated desktop cloud architecture, and also setUser Experience as the core, optimized the strategy of resource scheduling and management in traditional architecture.

Key words: desktop cloud; integration; resource scheduling; virtualization; credit

云计算是运营商优化自身 IT 架构的重要策略之一[1]，而如何高效的管理终端则始终困扰着企业信息化人员，尤其是需强管控的营业厅类生产任务型终端正面临来自安全、运维、能耗、管理，以及 TCO 成本等多方面的挑战。

桌面云正是诞生于这种背景下，它是将终端计算机的各个逻辑单元进行“松耦合”的过程，它将传统模式下用户侧的操作系统、应用程序和用户数据转移到数据中心进行运行和保存。用户利用瘦终端（TC）或者“智能终端+软终端”（智能终端含普通台式机、笔记本电脑、智能手机等），鉴权认证后通过优化的网络协议访问数据中心云端服务器和应用程序。本地不保留关键数据，实现用户桌面环境的集中存储、监控与管理。

桌面云架构由终端侧、网络侧和云平台侧三部分组成[2]。终端与网络侧主要为用户提供一种接入手段，云平台侧为桌面云架构的核心，云平台侧主要包含业务层、资源管理调度层、虚拟资源层、物理资源层、网络层、物理层六大部分。按功能可将六大功能模块划分为两大功能域，即：管理功能域与虚拟资源域。其中虚拟资源域主要功能为将物理资源抽象为虚拟资源。管理功能域主要功能为接受用户注册到云平台，并调度相应虚拟资源给用户。

通过引入桌面云，用户的访问模型由二层（终端->应用）变为三层（客户端->桌面云->应用），用户可

① 收稿时间:2011-07-25;收到修改稿时间:2011-09-03

按需获得更充沛的桌面应用程序运行能力，在技术层面利用数据中心提供集中、统一的安全防护能力，有效的提升了企业信息安全级别，同时，利用虚拟化等技术共享后台计算和存储资源，桌面资源利用率提高 1 倍以上。运维管理方面虚拟桌面与瘦终端的统一管理与交付有效推动了终端管理的标准化，同时随用户桌面集中至数据中心，运维效率得到极大提升，并且单用户能耗降至 50W，节能减排效益显著。经济效益方面随着用户规模的放量，时间的递增，整个 TCO 效果愈加明显[3]。

目前，通用桌面云解决方案多立足于本地网，试图借助虚拟化技术将一个或几个数据中心的计算能力整合后交付给最终用户，对于当前用户规模，现有架构基本可以实现预期目标，但并不足以支撑将桌面云作为一种通用解决方案进行推广。

本文通过对现有架构进行分析，结合运营商生产任务型终端场景，建立了一套符合运营商实际，并能充分解决目前终端管理困扰的一体化桌面云架构，并以用户体验为核心对现有资源调度策略进行优化，为桌面云的顺利推广奠定了基础。

1　一体化桌面云架构

造成传统桌面云局限性的最主要原因是现有架构没有站在整体的角度去匹配用户需求。首先，从管理角度来看，现有架构下终端依旧为分点、分散管理，缺乏一种标准的办法和手段来支撑全集团对终端的统一管控；其次，从风险角度来看，桌面云的引入实际上是在用户与后台系统之间增加了一层，它的可靠性、稳定性直接决定了用户体验，因此，不能简单的将桌面云作为普通的 IT 系统考虑，它的重要性级别应约等于后台 CRM 类业务系统，同时随终端集中，服务中断风险对现有架构提出了新的考验；再次，从资源共享的角度，桌面云是一种 IaaS 服务[4]，它将数据中心的计算能力抽象后提供给用户，分散架构分区域建立多片云，但云之间难以交互，这与云计算资源整合、资源共享的理念相违背；最后，从成本的角度，分散架构难以形成规模效应、无法发挥云计算集约性优势，容易照成资源的浪费。

通过上文分析，集中化架构是提升桌面云能力，实现统一管控的一种有效手段。集中的最大瓶颈在于网络，单用户虚拟终端带宽需求在 100-200kbps 之间，同时虚拟终端对网络的时延、抖动要求较高，对于运营商十万级的终端规模，现有 DCN 网尚无集中承载所有虚拟终端的能力。因此需对现有桌面云架构进行充分解耦，分析各个功能模块的集中需求，采用分级集中的方式实现标准、统一。

虚拟资源域的集中需求主要来自于场景、运维与成本三个方面。营业厅类生产任务型终端主要任务在于与后台 CRM 类业务系统的信息交互，目前运营商的 CRM 类业务系统已实现分省集中部署，电子渠道类系统甚至已全国集中，因此将虚拟终端集中至数据中心后，可以将系统间的大数据量交互消化在数据中心内部，可极大的改善用户体验。运维方面，虚拟终端集中后，原有分散的运维方式得到了极大的改善，运维工作可集中在数据中心内部完成，提升了运维响应速度的同时，极大的降低了运维队伍规模。成本方面，集中后产生的规模效应可有效的改善运营成本。但受限于虚拟终端的带宽需求与用户 SLA 与 QoS 要求，尚不具备将虚拟终端进行物理全国集中的条件，同时考虑部分业务系统分省集中部署的现状，考虑对虚拟资源域分省集中。

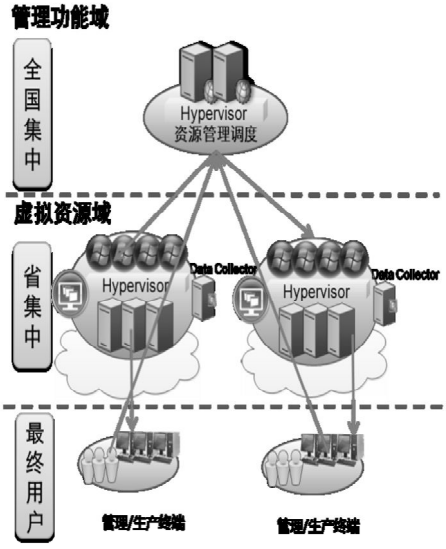

图 1　一体化桌面云架构

管理功能域的集中需求主要来自管理、安全、服务可靠性三部分。桌面云对运营商生产任务型终端管理的最大提升在于提供一种标准、统一的管理手段，管理是运营商引入桌面云的最主要驱动。将管理功能集中后，可提供统一的终端管理手段，实现完整的终

端资产管理；其次，管理功能集中使标准终端桌面镜像、统一补丁分发成为可能，有效提升了终端的标准化程度，规避了敏感信息泄露的风险；最后，管理功能集中后可实现虚拟资源的跨省、跨区域调度，规避了某个数据中心宕机导致服务中断的风险。同时管理功能集中并无网络等方面因素制约，因此考虑将管理功能域全国集中。

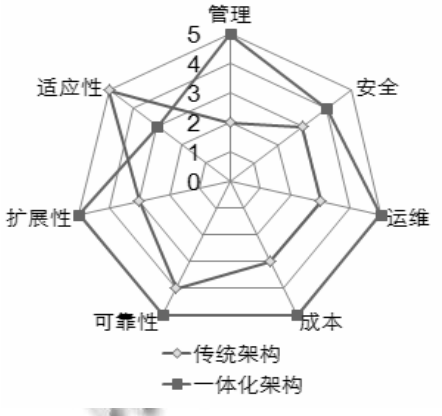

图 2 一体化架构与传统架构对比分析

2 一体化桌面云架构下资源的调度与管理

2.1 资源池创建原理

一个高可用、可动态扩展的计算资源池是桌面云解决方案成功的基础，计算资源池是借助 Hypervisor[5]（虚拟资源层）的能力通过将物理服务器虚拟成多个 VM(虚拟机)而形成的。

Hypervisor 采用混合型虚拟化架构模式，如下图。其中 Hypervisor 为介于硬件与操作系统之间的一个薄软件层，实际上为一个经过裁剪改造的 linux，主要功能为虚拟硬件资源，为服务器上运行的虚拟机提供一个运行时环境。hypervisor 负责各虚拟机间 CPU、内存的管理与调度，但不能处理网络、IO、存储等请求。这些请求借助其上运行的 HOST OS 的 I/O 设备驱动能力实现。HOST OS 是唯一运行于 Hypervisor 之上的虚拟机，它具有访问物理 I/O 资源的权限，负责与其他虚拟机的交互。Hypervisor 同时负责多个 VM 间物理资源的动态调配。

2.2 物理硬件资源层资源调度

云计算是一种破坏性的创新，它通过整合物理资源成功消除了传统应用系统“烟囱式”架构的种种弊端，但它在带来资源共享、按需分配等好处的同时，也对现有技术架构、运维体系产生了巨大的冲击，如何保障桌面云化后用户体验不降低已成为以桌面云为代表的云计算解决方案成败的关键。由于对虚拟资源的调度与管理很大程度上决定了桌面云平台所提供的服务质量，本文的另一个研究重点是设计一套适用于一体化桌面云架构的资源调度与管理策略，以便将合适的虚拟资源交付给最终用户，并通过资源的动态调配来保障用户体验。

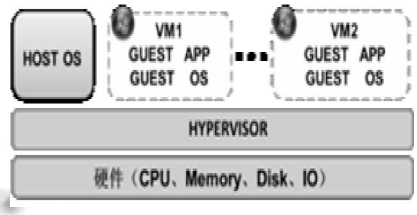

图 3 混合型虚拟化架构模式

2.2.1 传统桌面云架构下物理硬件资源层资源调度

在桌面云架构中，包含对底层物理硬件资源以及资源池中的虚拟资源两个层面的调度与管理。对于底层物理硬件资源的调度与管理主要由 Hypervisor 实现。Hypervisor 对资源调度实际上是一个将 VCPU 映射至物理 CPU 的过程，Hypervisor 以 VCPU 为单位来对硬件资源进行调度,首先 Hypervisor 为 GUEST VM 分配相应 VCPU(一个 GUEST VM 可对应多个 VCPU，但同一 VCPU 只能分配给唯一的 GUEST VM)，之后 GUEST OS 通过自有的线程调度算法将核心线程映射到 VCPU 上。最终 VCPU 由 Hypervisor 上的调度算法映射至物理 CPU 上。

Hypervisor 资源调度算法 Credit[6]是一个按比例共享物理 CPU 资源的非抢占式调度算法，它的核心思想是将物理 CPU 资源公平的分配给每一个 VCPU。Credit 算法通过为 GUEST VM 定义二元组（weight，cap）来标识虚拟机对物理 cpu 占用的优先级。其中 weight 值表示当前虚拟机可以占用物理 cpu 时间的比例，cap 值表示虚拟机可占用的最大物理 CPU 时间。

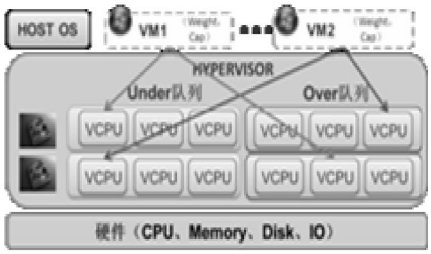

图 4 credit 算法

每个 VCPU 有 over，under 两个优先级，Credit 算法为每个物理 CPU 维护两个双链表队列，即 Under 队列与 Over 队列。队列上的元素为被分配到该物理 CPU 上运行的 VCPU，队列按照先到先服务方式排序。Credit 算法通过监测每个 VCPU 的 credit 值消耗（由对应 VM 的 weight 与 cap 计算得出）来调整队列上 VCPU 的服务顺序。当 credit>0 时，表示该 VCPU 请求的资源尚未被足额分配，仍可被物理 CPU 调度，处于 under 优先级；当 credit<0 时，表示分配给该 VCPU 的时间片已经被消耗完，不能再被 CPU 调度，处于 over 优先级。

当一个 VCPU 被调度执行时，Credit 算法会每隔 10ms 触发一次时间中断，并将当前运行的 VCPU 的 credit 值减掉 100，如果 credit 值>0,则继续执行，否则将它插入到 over 队列的尾部。每隔 30ms，如果 under 队列上 VCPU 排序仍没有变化，Credit 算法会对 under 队列重新排序，将当前运行的 VCPU 放至 under 队列队尾，即当一个 VCPU 的 credit 值够多时，它最多可以运行 3 个调度时长。如果当前 VCPU 均处于 over 队列时，Credit 算法支持从其他物理 CPU 队列中调度处于 under 优先级的 VCPU 运行。只有当前所有 VCPU 均处于 over 队列时，才会对所有活动 VCPU 重新赋值。

2.2.2 物理硬件资源层资源调度优化

Credit 算法实现了将物理 CPU 资源公平的分配给用户，但在实际场景中，用户的优先级并不相同，不同用户的 SLA 与 QoS 存在差异。以营业厅为例，存在主管、VIP 营业员、普通营业员三类用户角色。这三类角色业务紧迫性程度： VIP 营业员>普通营业员>主管，因此结合用户场景紧迫程度高的用户需被优先响应。

本文针对营业厅类生产任务型场景提出了一种改进型 Credit 算法，在新的算法中，为物理 CPU 额外维护一个 boost 队列，位于该队列的 VCPU 处于就绪状态，即，当前映射在物理 CPU 上的 VCPU 结束运行后，优先调度位于 booest 队列上的 VCPU。同时为每个 GUEST VM 增加一个属性：优先权值（factor），用于标识被调度 VM 上承载业务的紧迫程度。当 VCPU 结束运行应被移到 under 队尾时，检测其对应 factor 值是否大于 0，如果大于 0，则将该 VCPU 放入 booest 队列，Factor 值减 1，boost 队列按 factor 值大小进行排序，只有当 boost 队列为空时，才会调度 under 队列中的 VCPU，factor 值刷新频率与 credit 值刷新频率保持一致。改进后，只要一个 VM 的优先级较高（即 factor 值>0），当它自身时间片消耗完后，不会被放置与 under 队尾，而是按其 factor 值放置于 boost 队列相应位置，从而保障优先级较高的用户可以抢占比它优先级低的 VCPU, 实现此类高级别用户延迟的降低。

2.2.3 虚拟资源调度优化

资源池建立之后,如何将虚拟资源（即 VM）高效、合理的分配给用户便成了最大问题。资源调度的目标是动态平衡资源池中各节点的工作负载，同时保证服务的高可用性。在一体化的桌面云架构中，将以此资源调度过程划分为四个步骤：资源请求、资源探测、资源选择、资源监控[7]。

为统一管理这些虚拟资源，将每个可用的虚拟资源抽象成一个可调度单元（Unit），每个 Unit 由一个二元组 V（C，M）确定，其中 C 表示 CPU 大小，M 表示内存大小。则一台物理服务器 Si 最多能虚拟的 Unit 数为：

$$Unit = \min\left|\frac{Ci}{C}, \frac{Mi}{M}\right|$$

单数据中心可用资源总数为：

$$\mathrm{Unit}_{total} = \sum \min\left|\frac{Ci}{C}, \frac{Mi}{M}\right|$$

同时引入分组的思想，将具有相同使用特点的资源分组，从而避免因统一调度大量无规则资源对云平台性能的消耗，即为可调度单元引入新属性 F，每个 Unit 由一个三元组 V（C，M，F）决定，其中 F 表示资源的核心特性，结合上文对生产任务型终端的分类，生产任务型场景下共包含三类不同特性的虚拟资源。当用户登入云平台，并输入相应能力需求即完成了资源请求。云平台接到用户资源请求之后，会首先通过安装在每台服务器节点上的数据收集模块探测节点负载信息（即：计算每个资源节点剩余的不同特性的可用资源数量）与服务器响应时间。

然后采用动态优先权（Priority）算法，选择 Priority 值最高节点，将其上与用户特性相匹配的虚拟资源交付给用户。动态优先权（Priority）算法即将服务器分组,给每个组定义不同优先权，用户请求会分配给优先权最高的服务器组（在同一组内，采用轮询或比率算法，分配用户的请求）。每组服务器的 Priority 值由当前节点的负载情况与服务器响应时间共同决定。动态优先权算法的核心思想是用户请求由当前负载最小且

响应最短的资源节点承载。

其中，F 表示用户需求特性权重，LT 表示服务器响应时间，μ 表示时间优先权转换比。当用户请求到 $Priority = Unit_{total} \times F + LT \times \mu$ 达时，Priority 值动态更新。

最后云平台会跟根据运行在每个节点上的 Agent 反馈的状态信息对每个服务器节点进行实时监控，一旦某个节点发生故障，云平台会将用户桌面迁移至其他资源节点。

3 结语

云计算对整个 IT 产业的影响日益深远，它已成为运营商解决自身 IT 问题的重要手段。云计算已经不再局限于一种技术手段，它开始更多的从服务模式、产业链构成、管控架构方面改变企业。桌面云作为云计算破坏式创新的产物从终端管理薄弱环节入手，优化 IT 产品和服务，并推动传统 IT 运营体系的变革。本文搭建起一套适应未来发展的一体化桌面云架构，并在此架构下，结合运营商营业厅类生产任务型场景，通过引入业务重要性级别对 hypervisor 层资源调度算法进行优化，同时在此基础上对虚拟资源调度策略进行改进，以提升用户体验。本文所设计的架构与优化策略已成功应用于国内某运营商，有效推动了其终端云化进程，解决了终端管理面临的种种难题。

参考文献

1 杨林凤.基于 XEN 网络虚拟化的性能研究.上海:复旦大学,2010.40–47.

2 顾振宇,张申生,李晓勇.Xen 中 Credit 调度算法的优化.微型电脑应用,2009.3–5.

3 肖斐.虚拟化云计算中资源管理的研究与实现.西安:西安电子科技大学,2010.30–43.

4 王新佳,田晨,熊桂喜.基于多级队列算法的 ITS 资源调度策略.计算机工程,2003:4–5.

5 董耀祖.基于 x86 架构的系统虚拟机技术与应用.上海:上海交通大学,2006.50–56.

6 鲁松.计算机虚拟化技术及应用.北京:机械工业出版社,2008.78–85.

7 周铁成.虚拟化技术在数据中心架构中的应用研究.现代计算机,2009:3–4.

（上接第 17 页）

参考文献

1 孙咏.基于 OCP 软件应用架构的设计与实现.北京:中国科学院研究生院,2009.

2 阎宏.JAVA 与模式.北京:电子工业出版社,2002.41–44.

3 Shalloway A.熊节译.设计模式精解.北京:清华大学出版社,2004.

4 Gamma E, Helm R, Johnson R.李英军译.可复用面向对象软件的基础.北京:机械工业出版社,2000.

5 Martin J, McClure Carma.软件维护---问题与解答.北京:机械工业出版社,1990.21–39.

6 邹娟,田玉敏.软件设计模式的选择与实现.计算机工程,2004,30(10):79–81.

7 Shalloway A, James R.设计模式解析.徐言声译.北京:人民邮电出版社,2007.

8 Gamma E.设计模式--可复用面向对象软件基础(中译本).北京:机械工业出版社,2000.1–21.

9 张跃平.符合开-闭原则的一种设计模式及应用.大连交通大学学报.2007,28(1):38–40.

10 Shalloway A. Design patterns explained.vPearson Education, 2002.