一种面向云构软件的云操作系统

王 伟[1,2] 胡长武[1,2] 郭 栋[1,3] 张静轩[1,2] 常进达[1,2] 张礼庆[1,2]
(同济大学计算机科学与技术系 上海 200092)[1]
(同济大学嵌入式系统与服务计算教育部重点实验室 上海 200092)[2] (云件实验室 上海 200000)[3]

摘 要 云计算和互联网的兴起不仅带来了数据中心的变化,也带来了软件的开发、部署、运维和使用上的变革。随着当前云计算和网络环境的不断完善,传统软件如何更好地利用云计算平台并服务于终端用户是计算机软件领域的研究热点,具有广泛的现实意义。近年来,随着实时互联网、微服务、云端渲染、容器等技术和理念发展的不断深入,软件(Software)形态将进一步朝着云件(Cloudware)形态的方向发展。文中主要探究了在云计算和互联网环境下"云件"这一新型软件范型,并针对该软件范型提出了一种新型的云操作系统:GalaxyOS。GalaxyOS可以在不修改传统软件的情况下直接将软件部署到云端运行,并通过浏览器实时投射到终端用户,实现传统软件的新型服务模式。通过采用微服务架构设计,使得该云操作系统具有较好的可扩展、容灾性和灵活配置等特性。通过对实际的GalaxyOS原型系统的实现和交互时延及资源占用进行实验,验证了所提方法在用户体验上的有效性。

关键词 云计算,云件,云操作系统,用户体验
中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.11.006

Cloud Operating System for Cloudware

WANG Wei[1,2] HU Chang-wu[1,2] GUO Dong[1,3] ZHANG Jing-xuan[1,2] CHANG Jin-da[1,2] ZHANG Li-qing[1,2]
(Department of Computer Science and Engineering, Tongji University, Shanghai 200092, China)[1]
(The Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 200092, China)[2]
(Cloudware Labs, Shanghai 200000, China)[3]

Abstract The rise of cloud computing and the Internet not only bring change on the data center, but also lead to transformation in software development, deployment, operation and maintenance. With the continuous improvement of the current cloud computing and the Internet environment, how to make better use of cloud computing platforms, and how to serve the end users in the field of computer software are big challenges. In recent years, with the further development of concepts like micro-services and containers, software will further step forward to the Cloudware. This paper defined the concept of Cloudware paradigm, and discussed how to deploy Cloudware in cloud environment. Then, we proposed a new cloud operation system called GalaxyOS which can directly deploy software on the cloud without any modification, while achieving a new model by the browser services. By using micro-service architecture, we can achieve such characteristics as good performance, scalable deployment, faults tolerance and flexible configuration. Finally, we evaluated GalaxyOS by proposing a user experience oriented framework and carrying out an interactive delay experiment that directly focuses on users' experience, and verified the effectiveness of our platform.

Keywords Cloud computing, Cloudware, Cloud OS, User experience

1 引言

操作系统是计算机系统中最为关键的一层系统软件。长期以来,操作系统发展的主线是面向单机,追求更好地发挥计算机硬件的计算能力,同时为上层应用和用户提供更友好、易用的接口。随着互联网、云计算等技术的快速发展,如何更好地支持互联网和云计算资源成为操作系统发展的一个重要方向[1]。近年来,由于云计算的迅速普及,面向云计算资源的操作系统开始得到广泛的关注,并逐渐成为操作系统发展的新主线[2]。为了更好地管理云上的分布海量资源,同时为互联网时代的新型应用和服务提供支持,操作系统技术正在产生许多重要的变革。

网络环境的不断优化,尤其是5G等相关无线通信技术的兴起,使用户的网络接入能力和网络质量有了大幅度提升,

到稿日期:2016-10-23 返修日期:2016-12-11 本文受国家自然科学基金(61672384),上海科委优秀技术带头人计划课题(15XD1503600)资助。
王 伟(1979-),男,博士,副教授,主要研究方向为并行分布式计算、云计算,E-mail:wwang@tongji.edu.cn;胡长武(1992-),男,硕士生,主要研究方向为分布式计算;郭 栋(1992-),男,硕士,主要研究方向为云计算;张静轩(1993-),男,硕士生,主要研究方向为软件定义网络;常进达(1993-),男,硕士生,主要研究方向为分布式计算、云计算;张礼庆(1994-),男,硕士生,主要研究方向为虚拟化和云计算。

为传统软件逐渐向云计算平台迁移提供了通信保障。通过浏览器的方式获取软件服务将是未来软件发展的重要方向，软件的 Web 化和云化也将成为未来软件的重要形态之一[6]。

利用云计算环境构建软件的开发、部署和运行环境，同时利用先进的互联网技术实现软件的 Web 化，将是云计算环境下软件的发展方向和趋势[3]。在当前云计算环境下，软件将不再是一个简单的代码实体，而是由一系列服务构成的服务综合体，通过网络交付给用户。本文将这样的软件形态称为云件，它是一种“互联网＋软件”的新型软件形态，也将是未来云环境下软件的主要形态，使得在任何时间、任何地点通过浏览器使用任何软件成为可能[4]。

为了验证面向云件的云操作系统的可行性，本文首先阐述了云件的概念和特点，然后提出并实现了一种面向云件服务的新型云操作系统 GalaxyOS。该操作系统基于实时互联网、微服务、云端渲染和容器技术，可以直接将传统软件部署到云端运行，用户通过互联网技术连接到云件，使用浏览器进行交互，整体上实现即搜即用、秒级启动的新型软件服务模式。

2 云件的概念

2.1 云件的定义和特性

软件是一种特殊的人工制品，是人类“智力活动”的产物。在信息化社会，软件正变得无处不在，并成为信息时代的重要基础设施。软件技术的发展贯穿于计算机技术发展的整个历史，然而，软件是相对于硬件的一个概念，是一系列按照特定顺序组织的计算机数据和指令的集合，其特性与承载软件的硬件结构密切相关。随着云计算和虚拟化技术的兴起，越来越多的软件逐渐把软件主体放在云端，而客户端只需要通过互联网技术使用云端软件的服务即可，使得软件不再依赖于终端软硬件资源，这样的软件形态更多地体现为一种服务，而这样的软件形态则称为云件(Cloudware)[4]。

云件也是 SaaS 的一种服务方式，是通过互联网技术使用云端的服务，与传统的 SaaS 服务(如网盘、邮箱和在线办公)的主要差别在于，传统 SaaS 软件往往是将桌面软件进行大量改造，大部分需要相应的客户端程序，且大量的计算还需要本地软硬件的支持，比如某些在线制图的工具将传统桌面制图软件用 HTML 5 和 Flash 等相关技术进行了 Web 重构，这是极其繁重的工作，且需要本地渲染的支持；而云件则是将终端的操作系统和运行环境迁移到了云端，使得传统桌面软件不进行任何修改就可进行云化(Cloudalization)，客户端采用统一的交互平台(如浏览器)来实现交互功能，最终实现与本地相同的用户体验但不依赖本地资源的软件模式。

云件主要有如下几个特性[4]：

(1)云件主体云端运行。云件将传统桌面软件部署在云端运行，其依赖的配置、库和相关组件全部由云端的服务提供。

(2)按需资源分配。云件能够按照自身类型分配不同的资源，且能够随时调整资源用量，比如对计算密集型的云件可以提供多核支持，进而满足不同用户的需求，实现弹性云件。

(3)云端渲染，终端显示。对于图形化交互的云件，尤其是具有 GPU 需求的云件，在云端实现云端渲染，然后将渲染后的结果传给终端，而不受限于终端的硬件资源。

(4)毋须安装，快速启动。云件的启动需要达到本地启动软件的速度，能够秒级启动大型软件，且不用事先安装相关插件等额外组件。

(5)通过网络交付。云件的服务全部通过网络交付，保证数据和云件状态能够实时保存在云端，终端只需要针对某次连接会话做必要的缓存即可。

(6)统一交互平台。云件需要有统一化的交互平台，交互平台需要有广泛的终端适应性和普遍性，以保证相同云件可以使用不同终端进行交互，且具有相同的交互方式和体验。

(7)文件透明传输。由于云件产生的文件全部存储在云端，需要相应的机制将不同云件和终端文件系统连接，实现互相透明的访问，使文件对用户不存在远程和本地的区别。

从上述特征可以看到，云环境下的软件不再是一个简单的代码实体，而是由一系列微服务构成的服务综合体，通过互联网进行交付，体现为一种更加契合云环境的软件形态，这就是云件的本质；同时，云件是一种不需要下载安装即可使用的应用，它实现了应用“触手可及”的梦想，用户通过搜索即可打开云件应用；也体现了“用完即走”的理念，用户不用关心是否安装太多应用的问题。云件应用将无处不在，随时可用，但又无需安装和卸载。

2.2 云件作为一种软件范型

我们还可以从软件范型的角度看待前面提出的云件概念，即云件范型[5]。一般来说，软件范型包括 4 个方面的内容[6]：被构建和执行的软件主体是什么(模型)，如何运行这些软件结构或主体(运行时系统支持)，如何开发软件主体(工程技术)，以及这些被构建和执行的软件主体所表现出来的效果(软件质量保证)。

(1)云件模型

云件模型是为了明确云件主体的形式、结构、行为及交互过程的一种规范，它决定了云件技术所采用的规范和技术特点(如编程语言、部署方法和运行机制等)。云件模型既能兼容传统软件，也能利用新兴的特性来指导云件的构建，例如采用面向对象的技术或服务计算的技术。云件模型主要包括 3 个方面：云件的客户端、云件的云端以及两者之间的交互关系。

(2)云件操作平台

软件操作平台实现了软件模型的基本元素及彼此之间的关系。与之对应，云件操作平台提供了一个运行空间来操作云件主体及彼此的交互，它让传统软件具备了云件的特性，同时也能用一种更加智能和自动化的方式来管理云件主体。

云件与传统软件的不同之处在于，云件的主体运行在云端，而传统软件则运行在客户端。因此，对云件来讲，关键问题在于如何与用户交互。云件主体的计算和存储都在云服务器上，因此客户端仅需要一个交互环境即可。近年来，软件网络化是软件变革的一种趋势，云件可以被认为是一个提供交互服务组件的浏览器，但它的操作是在客户端。浏览器交互实际是输入与输出的虚拟化过程，仅需要鼠标和键盘这样的设备。输入就是将数据发送到云服务器，然后返回到客户端进行渲染，可以达到与本地软件相同的效果。通过这种方法，用户可以像使用本地软件一样使用云件，但他们是用浏览器实现这些操作的。在任何时候、任何地点，云件都可以返回当

前最新的状态,不用依赖于客户端的操作系统和运行状态。

(3)云件工程方法

云件工程可以控制云件开发的整个生命周期,包括需求分析、设计、实现、部署、维护及更新。云件工程方法遵循"全生命周期的软件体系结构"的核心准则。软件体系结构作为一张蓝图,控制着云件每一步的开发。为了支持云件应用的在线开发,云件及其交互都是基于软件体系结构来实现和管理的。

开发:在传统软件开发过程中,开发人员需要自己构建相应的软件开发环境,如 IDE 和编译工具链等。随着 Git 和任务管理系统的兴起,云件的开发应当体现为云端开发过程,利用云端 IDE 和编译完成软件的整个开发任务,同时利用云端协作软件进行任务追踪,从代码编写和软件工程两个角度对软件开发过程进行云化。

部署:云件的部署其实就是微服务的部署。目前以 Docker为代表的微服务容器技术发展得越来越成熟。Docker 提供了一系列容器部署工具,为开发者提供了一种新颖、便捷的软件集成测试与部署之道。云件的部署应当以服务发布的形式体现,不同的构件可以单独部署,也可以集成部署,提供向下兼容的服务部署形式,保证云件的不中断运行,这也是云服务的基本需求。

(4)云件质量保证

云端软件通常借助网络,以一种在线和同步的形式来服务大量用户。云件质量框架不仅定义了各种质量属性的定量和定性测量方法(如性能、可靠性和可用性),也在这些属性之间进行综合权衡。为了保证云件质量,要求在云件开发(如测试、验证和确认)和云件运行(如在线更新、自动系统管理)时都有质量保证机制。由于云件尚处于早期阶段,本文主要关注它在部分关键性能方面的表现,例如云件启动时间、网络延时和用户体验等;未来将逐步涉及到更全面的云件质量保证机制的整体研究。

3　GalaxyOS 云操作系统

相对于传统桌面软件运行在桌面操作系统之上(例如 Windows),云件也需要一个统一的分布式操作系统作为运行环境。本文提出了一个面向云件服务的云操作系统:GalaxyOS,它是云件开发、测试、部署和运维的集成操作平台,既面向开发者提供云件开发工具和云件运行环境,也面向用户提供云件服务。

3.1　GalaxyOS 的整体架构

GalaxyOS 的整体架构如图 1 所示,主要包括 4 个层次 6 个部分。

(1)KaaS 层(Kernel as a Service):运行在标准的 IaaS 层之上,通过 Linux 内核池提供上次服务所需的内核服务,同时还包括内核管理、内核监控、内核调度等服务。

(2)CaaS 层(Container as a Service):提供标准的容器服务。由于 GalaxyOS 采用全微服务架构,因此所有的服务均封装成容器镜像提供服务功能。该层还包括容器的编排、调度和监控等服务。

(3)NaaS 层(Networking as a Service):提供分布式系统所需的网络服务,目前主要提供 Iptables 和 NAT 等服务。

(4)XSaaS 层(X Streaming as a Service):提供交互式远程视屏传输服务,向下通过通讯协议(例如 X11 协议)与 Container Service 通信,向上以 TCP 协议与 Web Service 通信。

(5)STaaS 层(Storage as a Service):提供分布式存储服务,为使用云件过程中所需要的文件存储提供服务。

(6)OaaS 层(Orchestration as a Service):提供总体控制协同服务,同时提供云件的 Open API,为上层的应用场景提供服务接口。该部分还包括 Web 前端服务、配置服务、CIP 协议服务、工作流服务等。

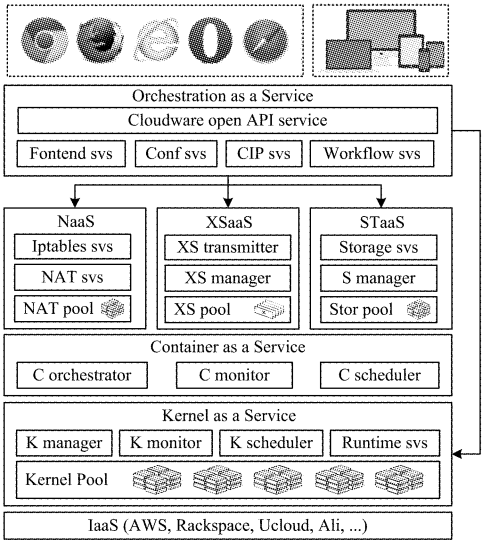


图 1　GalaxyOS 的整体架构

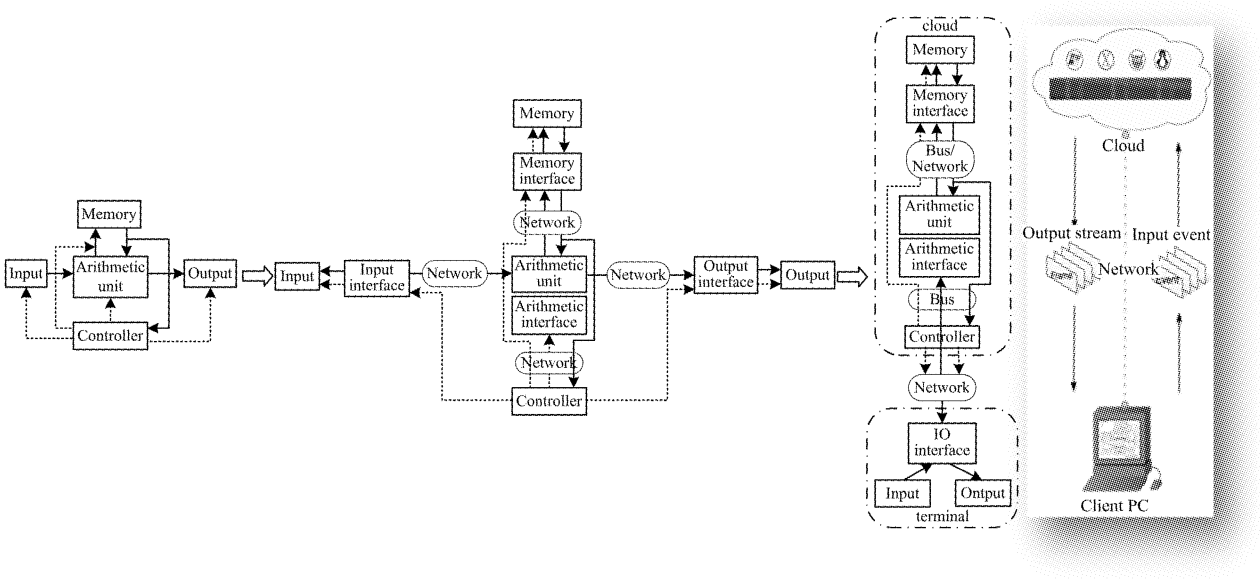
3.2　计算与存储分离的设计理念

传统的计算机本身是一个虚拟化的典范,它的计算实体是基于图灵-哥德尔可计算理论和冯·诺依曼体系结构来构建的,基本实现思路是通过将最简单的加法运算和与非逻辑运算进行组合与迭代来实现各种复杂运算功能。

与传统基于标准的冯·诺依曼体系结构的操作系统不同,面向云件的 GalaxyOS 使得计算机系统的输入、输出、存储和计算都不在一个单一的计算机系统中,它们可能分布在互联网的各个地方,通过网络连接在一起。比如,输入和输出部署在终端机器上,而云件的存储、控制和计算则在云端服务器上,从而使得云件的输入、输出和计算都自成系统,通过互联网并基于相应网络协议实现通信,本文将这样的计算模型称为松耦合冯·诺依曼计算模型[5]。该模型是 GalaxyOS 的设计理论基础。

如果将计算看作是指令存储、指令执行和结果展示,则传统操作系统是将三者局限于一台单机,这种做法可以看作是一种极端;而早期的瘦客户机技术将指令存储、执行与结果展示进行部分分离,包括"透明计算"[7]这种将计算与存储彻底分离的理念,可以说向前迈进了一步;而云计算时代,面向云件的 GalaxyOS 模式则将指令存储、指令执行与结果展示完全彻底分开,这便是另一种极端,甚至可以按需定制。

GalaxyOS 设计的关键在于如何将冯·诺依曼机的 5 个模块进行解耦。解耦的关键在于各个模块需要自成系统,并能够独立运行,各个部件之间通过相应的网络协议进行通信,其设计理念如图 2 所示。



(a)标准冯·诺依曼体系结构 (b)松耦合冯·诺依曼计算模型 (c)云件实现

图 2

GalaxyOS 操作系统作为云件系统中最为重要的系统软件，一方面直接管理各种计算资源，另一方面作为“虚拟机”为云件应用程序提供运行环境。在此意义上，GalaxyOS 操作系统体现了“软件定义的系统”的特征。当前出现的软件定义的网络、软件定义的存储等技术，如同设备互联技术、磁盘存储技术之于单机操作系统一样，本质上反映了“网络化操作系统”对网络化、分布式设备的管理技术需求；而 GalaxyOS 则反映了“云操作系统”对应用软件“触手可及”、“用完即走”理念的需求，将成为“云化操作系统”核心的底层支撑技术。

GalaxyOS 通过“软件定义”的途径，一方面实现了资源虚拟化，达到物理资源的共享和虚拟资源的隔离；另一方面实现了管理功能的可编程，打破了传统硬件配置能力有限的桎梏，为用户的软件需求提供高效灵活、随需而变的支撑。

3.3 GalaxyOS 的控制平面和数据平面

基于上述设计理念，GalaxyOS 的各服务之间的通信过程可进一步分为控制平面和数据平面，整体结构如图 3 所示。

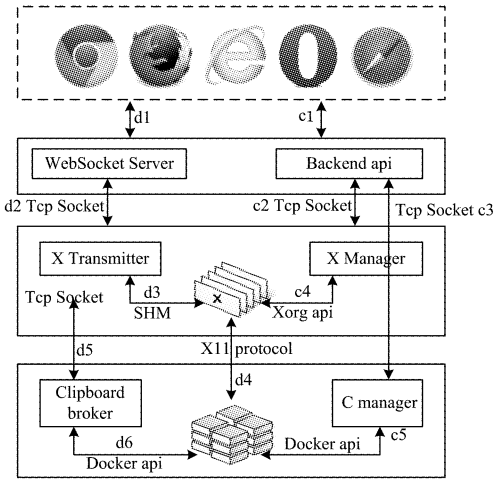


图 3 GalaxyOS 的控制平面和数据平面

其中，c1－c5 为控制平面通道，d1－d6 为数据平面通道。控制平面主要实现云件的生命周期控制，如云件的运行、关闭和重启等。c1 为用户通过浏览器调用 Backend API 的过程，基于 HTTP Restful API 将用户的请求发送到云端。对于云件 PaaS 平台而言，主要的请求包括 X Service 请求（分辨率调整、鼠标键盘事件等）和 Container Service 请求（应用程序运行、关闭和重启），分别通过 c2 和 c3 基于 TCP 协议与 X manager 和C manager 通信。X manager 利用 Xorg 提供的如 Xlib 和 XCB 等库对 Xorg 程序进行控制，C manager 则通过Docker提供的 API 和相应的 language binding 实现对 docker 的控制。

数据平面主要关注交互层的数据流，d1 为浏览器和 WebSocket Server 的通信过程，通过 WebSocket 协议将用户的鼠标和键盘等输入发送到 WebSocket Server，WebSocket Server 实时将云件的 H.264 视频流传输到浏览器；d2 为 WebSocket Server 和 X service 的数据交互，基于 TCP 协议，对上述的事件和图像进行转发，交由 X Transmitter 进一步处理；d3 为 X Transmitter 和 Xorg 的交互过程，X transmitter 将鼠标键盘事件发送给 Xorg，同时将 Xorg 的帧缓存中发生变化的图像经过 H.264 编码压缩后发送给 WebSocket Server，由 WebSocket Server 进一步发送到终端浏览器。

为了能够快速获取变化图像，X Transmitter 采用了与 Xorg 共享内存的方式获取对应的渲染位图，并使渲染位图经过 H.264 编码成为视频数据流；d4 为云件 x11 客户端应用程序和 Xorg 通信的数据通道，该过程完全基于 X11 protocol 实现；d5 为剪贴板扩展的数据通道，以实现用户在客户端以及不同云件之间的数据拷贝过程，客户端数据可以通过 d5 发送到 clipboard broker，然后通过 d6 的数据通道发送到对应云件应用程序的剪贴板缓冲区。

4 原型系统与性能评测

本节对提出的 GalaxyOS 进行了实际的用户行为模拟测试，并对测试的结果进行了统计和建模分析；基于这些结果，对云件操作系统的交互时延进行性能分析。为此，本文基于

上述内容开发并实现了一套实际的云件服务平台 CloudwareHub[1)]，来验证 GalaxyOS 的理念与实际意义。

4.1　云件 PaaS 平台的运行环境

CloudwareHub 本身提供的是微服务容器运行环境，为了实现规模化部署、容灾和灵活配置，系统的部署和运行也以微服务形式架设在 IaaS 云计算系统上。为了验证本文所提方法的有效性，CloudwareHub 原型系统运行在 2 台云主机上，配置 2 核心、4GB 内存和 20GB 硬盘。Container Service 构建在 Linux 环境下，基于 Ubuntu Server14.04，安装 Docker 工具组和 Xorg 驱动以及编程库。上述所有内容均在 Ucloud[2)] 上进行相应部署，同时其还提供了 Matlab 等云件服务，运行效果如图 4 和图 5 所示。

图 4　CloudwareHub 中云件的运行界面

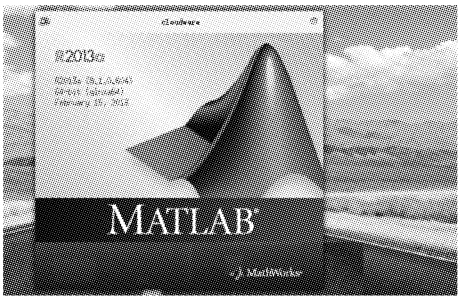


图 5　运行 Matlab 云件的情况

同时，本文开发了一款名为 Fornax 的云件管理系统，以实现云件的自动化部署以及更为有效地提高资源利用率，如图 6 所示。

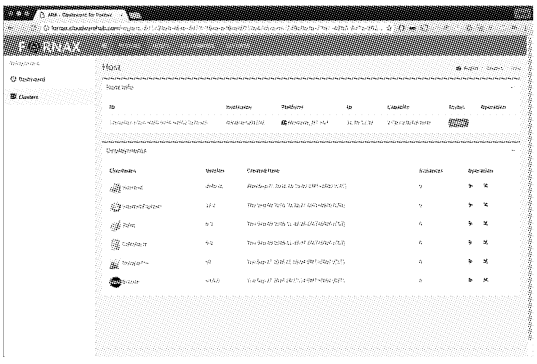


图 6　Fornax 管理界面

4.2　交互时延实验

为了评估 GalaxyOS 的实际效果，本文对提出的云件平台 CloudwareHub 进行实际的用户行为模拟测试，并对测试的结果进行了统计和建模分析；然后基于这些结果，对 GalaxyOS性能影响最大的交互时延因素做进一步的深入分析。

由于 GalaxyOS 本身是基于网络的服务，其用户体验与网络环境有很大的关系，较高的网络时延将极大程度地降低用户体验。在局域网环境下，网络延迟不太明显，但是在广域网环境中，网络的不稳定性可能导致云件的使用体验较差。为了测试 GalaxyOS 的体验指数，本文通过实验测量了在不同网络环境下不同 GalaxyOS 随机操作的交互时延。

为了测量不同云端网络带宽和不同云件的时延，本文采用 50Fps 的 FFRM 模式进行视频传输，并选取了 4 种不同渲染复杂度的云件进行了测试，包括 gedit，eclipse，rstudio 和 supertuxkart。对上述云件进行随机操作，测量在不同云端网络带宽下从用户输入事件到云端输出图像至终端的交互时延，得到如图 7 所示的结果。

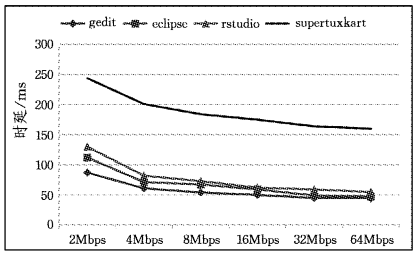


图 7　不同云件在不同云端网络下的交互时延

分析结果可知，当渲染的数据量不大时，延时基本是固定的，此时数据量和带宽并不是造成时延的主要原因，主要延时来源于 TCP 通信过程和内存拷贝的过程。渲染复杂度对时延会造成一定影响，但是并不是线性关系，主要是由于渲染本身所耗费的时间以及云端的流数据压缩造成的时延并不与最终编码后的帧大小呈线性关系。总体上，较为复杂的渲染会导致时延的增加。

4.3　编码器对系统资源占用和体验效果的影响

进一步，本文在终端采用带有 i7-4750 CPU 的机器运行 53.0.2785.116 版本的 Chrome 浏览器；云端机器带有 2 核心虚拟化 e5-2670 CPU，云端网络为 2Mbps，H264 编码帧尺寸为 1400×900。在此硬件基础上，验证不同编码器对系统资源和用户体验效果的影响。

(1)使用 X264 编码器[3)]，速率控制模式设置为恒定速率因子，速率因子设为 23。云件终端的 CPU 状态周期采样图如图 8 所示。

图 8　云件终端 CPU 资源的 Profiling

云端 CPU 的使用情况和网络的使用情况分别如图 9 和图 10 所示。

1) http://www.cloudwarehub.com

2) https://www.ucloud.cn

3) http://www.videolan.org/developers/x264.html

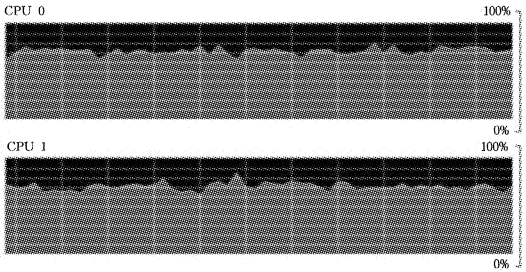


图 9 云端 CPU 资源的使用情况

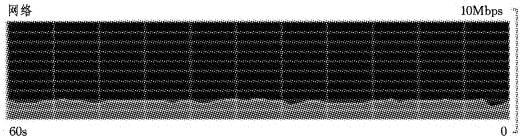


图 10 云端网络的使用情况

可以看出,终端的帧率(Frame Rate)为 11fps～13fps,平均解码时间为 45mspf (millisecond per frame),平均数据传输与编码时间之和为 90mspf;云端 CPU 的使用率约为 80%,网络比特率稳定在 1.9Mbps。

(2)使用 Openh264 编码器[1),速率控制模式设为 RC_QUALITY_MODE。所得到的云件终端的 CPU 状态周期采样图如图 11 所示。

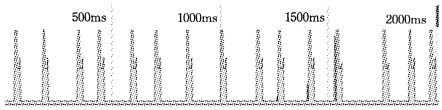


图 11 云件终端 CPU 资源的 Profiling

云端 CPU 的使用情况和网络的使用情况分别如图 12 和图 13 所示。

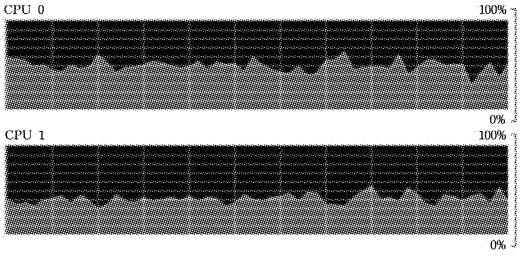


图 12 云端 CPU 资源的使用情况

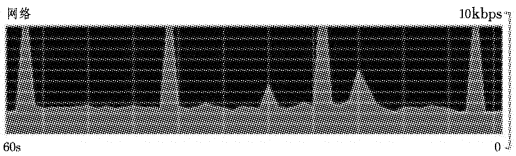


图 13 云端网络的使用情况

可以看出,终端的帧率(Frame Rate)为 8fps～10fps,平均解码时间为 19mspf,平均数据传输与编码时间之和为 112mspf;云端 CPU 的使用率约为 42%,网络比特率平均为 56kpbs,IDR 帧率为 3.2fpm。

在终端接收到的帧率大致相同的情况下,使用 openh264 编码器传输的帧被解码的时间约为使用 x264 编码器所传输帧的 42%,前者所占用的 CPU 资源约为后者的 50%,前者所占用的带宽仅为后者的 2.95%,可见 openh264 编码器的压缩效率明显高于 x264 编码器。但同时,openh264 编码器的压缩时间比 x264 编码器高出了 24.4%。总体来说,使用 openh264 编码器能更为合理地利用资源,给用户带来更好的体验。

总之,GalaxyOS 的用户体验由许多因素决定,提高云端性能、提高网络质量并降低渲染数据的传输量是提高 GalaxyOS 用户体验的关键所在。如何解决网络的固有延时问题和在抖动网络的情况下保证 GalaxyOS 的用户体验,是未来 GalaxyOS 研究的主要方向和问题。

5 相关工作

5.1 软件技术的发展和应用软件虚拟化

自 1946 年第一台计算机 ENIAC 诞生起,计算机软件逐渐在人类社会中起到至关重要的作用,并快速发展。总体来看,软件的技术架构主要经历了单机软件、C/S 软件、B/S 软件和 SaaS 软件等几个阶段,而计算机软件技术的变化更多地是随着计算机体系结构和计算机网络技术等的发展而发展。

随着浏览器成为互联网的入口,越来越多的软件提供商,特别是传统应用软件提供商,采用 SaaS 的方式通过浏览器来提供软件服务。其中一种思路是通过改造和重构将传统应用软件转变成基于 SaaS 的 Web 应用,进而实现 SaaS 化,但该方法需要对软件的源代码进行重构,相当于重新开发目标软件,耗时耗力。另一种思路是直接将传统应用软件放到服务器端,同时将软件的图形用户界面投射到浏览器中,进而让用户通过浏览器来完成与软件的交互,即应用软件虚拟化(Application Software Virtualization,ASV)技术,该技术已经成为传统软件走向云端的核心技术之一。

目前,研究人员通过多种方式来实现应用软件虚拟化。一种是为传统应用软件创建一个 Web 用户界面(UI),然后在这个用户界面和用户之间提供一个中介机制来完成用户和软件之间的交互[8-9]。该方法需要软件提供商为每一种软件都开发一套专属的 Web 用户界面,并封装好后提供给用户,通用性较差。另一种方法是为应用软件设计一套工具库来动态产生 Web 用户界面[10]。这种方法解决了通用性问题,但是兼容性较差,能够支持的应用软件有限。还有一种目前较为流行的方法是通过桌面虚拟化的方式提供应用虚拟化,将整个桌面通过虚拟化的方式提供给用户,用户可以通过浏览器和虚拟桌面中的软件进行交互[11]。虽然用户可以直接在虚拟桌面中使用软件,但该方法将整个桌面提供给用户,效率较低,而且需要足够的网络带宽来保障用户体验。近来,Chen 和 Hsu 提出了一种桌面应用服务(Desktop Application Service,DAS)框架[12],通过服务的方式提供整套解决方案。该方法仅将应用软件的窗口投射到浏览器中与用户进行交互,

1) http://www.openh264.org

较好地解决了网络带宽的问题,但该方法本质上还是虚拟桌面的变种,无法解决服务器端资源的利用率问题,因此可扩展性和服务质量(如交互性延迟)无法得到保障。

这方面类似的工作还有远程桌面(Remote Desktopping),其将桌面的图像信息通过网络传回给客户端,给用户的感觉就像是在使用,本地桌面一样。远程桌面可以使用,也可以不使用虚拟化技术。这方面的工具包括 VNC,Windows Remote Desktop, TeamViewer, LogMeIn, Chrome Remote Desktop Extension 等,同时也有不少试图改善和优化这方面技术的研究工作[17]。

5.2　网络化操作系统和云操作系统的发展

随着近 20 年来互联网的快速发展,操作系统面对的计算平台正在从单机平台和局域网平台向互联网平台转移。操作系统不仅需要提供网络支持能力,更重要的是需要解决如何管理互联网平台上的庞大的计算资源和数据资源以及如何更好地利用分布式的计算能力等诸多问题[1]。

在互联网流行之后,出现了互联网操作系统的提法,各种组织和个人都曾经提出或者尝试开发过被称作 Internet OS 的软件和系统。如,著名操作系统专家、曾在 Amiga 个人计算机上首次引入多任务概念的 Carl Sassenrath 就曾推出过基于他所发明的 REBOL 语言的 REBOL Internet Operating System:IOS。与 Internet OS 比较接近的另一个概念是 WebOS[13],这是一个从 1997 年就开始出现的术语,主要思想是在浏览器中实现类似操作系统的功能,从而使用户可以随时、随地通过任意计算机上的浏览器访问自己的操作系统桌面。在 2000 年前,UC Berkeley 就实现了 WebOS 的原型系统,后来又出现了 VirtualOS, YouOS, G.ho.st 等流行的 WebOS。严格来讲,WebOS 不能被称作操作系统,它们只是在浏览器中采用 Web 页面展示技术来实现类似于操作系统界面的功能。Tim O'Reilly 在 2010 年发表了关于 Internet OS 现状的看法(The State of Internet Operating System),即现代的 Internet OS 应当包括如下的功能:搜索、多媒体访问、通信机制、身份识别和社交关系图、支付机制、广告、位置、时间、图形和语音一识别、浏览器。

随着云计算技术的发展,更好地支持互联网和云计算资源成为了操作系统发展的重要方向[1],面向云计算资源的操作系统开始得到广泛的关注,并逐渐成为操作系统发展的新主线[14-16]。但目前云操作系统的研究主要面向大型数据中心,没有针对桌面软件演化。特别是随着云件这一新型软件范型的出现,亟需新型的云操作系统。

结束语　云计算和互联网技术的发展和成熟,不仅带来了数据中心的变革,也影响到软件的开发、部署和运行方式,进而影响着软件的使用方式。在这样的环境下,利用云件的方式去开发部署软件将是未来云环境下软件的主要形式。云件本身体现了一切皆服务的理念,其微服务的设计理念更容易应用到云环境中。我们相信传统软件逐渐向云件方向转型是大势所趋,而本文所提出的 GalaxyOS 云操作系统正是适应这种背景的软件新平台,具有重要的研究意义。目前,云件和云件操作系统的技术和理论尚处于探索和初期发展阶段,我们将会持续对相关技术和理论加以探索和改进。

本文采用容器技术在云端管理软件运行实例,仅讨论了应用实例共享一个物理资源(宿主机)的情景,宿主机在典型配置下的最大承载能力以及业务之间可能存在的相互性能干扰情况,对于此类技术的大规模应用至关重要,是项目组未来的研究方向之一。另外,本文在性能评测中仅给出了有限场景下的操作时延分析,未来也会继续深入研究,特别是对交互密集型应用、流媒体播放及 3D 视频类应用的支持情况;同时也会进一步评价典型应用场景下基于视频流的远程执行模式对网络带宽的占用情况及总的网络流量情况。在实际情况中,大量桌面应用有很多依赖(如对 OS、类库、本地文件、端口资源等的依赖),部分应用的功能依赖于 GuestOS 操作系统能力或本地硬件能力(如与本地输入法的结合、本地设备的结合),部分应用场景中需要解决云端设备驱动的整合问题,解决该类问题也是云件走向实际应用场景所必须面对的。

参 考 文 献

[1] MEI H,GUO Y. Network-oriented operating systems:status and challenges[J]. Scientia Sinica Informationis,2013,43(3):303-321. (in Chinese)
梅宏,郭耀. 面向网络的操作系统——现状和挑战[J]. 中国科学:信息科学,2013,43(3):303-321.

[2] MICHAEL A,ARMANDO F,et al. A view of cloud computing [J]. Communications of the ACM,2010,53(4):50-58.

[3] STEFAN W,EDDY T,WOUTER J. Comparing PaaS offerings in light of SaaS development[J]. Computing,2014,96(8):669-724.

[4] GUO D,WANG W,ZHANG J X,et al. Towards Cloudware Paradigm for Cloud Computing[C]// The 9th IEEE International Conference on Cloud Computing. San Francisco,USA,2016.

[5] GUO D,WANG W,ZHANG J X,et al. Cloudware:An Emerging Software Paradigm for Cloud Computing[C]// Proceedings of The Internetware '16. Beijing,China,2016.

[6] MEI H,HUANG G,XIE C T. Internetware:A Software Paradigm for Internet Computing[J]. IEEE Computer,2012,45(6):26-31.

[7] ZHOU Y Z,ZHANG Y X. Transparent Computing:Concepts, Architecture, and Implementation [M]. CENGAGE Learning Press,2010.

[8] DE LUCIA A,et al. Developing Legacy System Migration Methods and Tools for Technology Transfer[J]. Software:Practice and Experience,2008,38(13):1333-1364.

[9] MENG X,et al. Legacy Application Migration to Cloud[C]// Proc. 2011 IEEE Int'l Conf. on Cloud Computing (CLOUD). 2011:750-751.

[10] KARAMPAGLIS Z,et al. Secure Migration of Legacy Applica-

tions to the Web[M]// Information Technology and Open Source Applications for Education, Innovation, and Sustainability. Springer, 2014: 229-243.

[11] WANG S T, et al. Development of Web-Based Remote Desktop to Provide Adaptive User Interfaces in Cloud Platform[J]. Int'l J. Computer, Information, Systems and Control Eng., 2014, 8(8): 1195-1199.

[12] CHEN B, HSU H, HUANG Y. Bringing Desktop Applications to the Web[J]. IT Professional, 2016, 18(1): 34-40.

[13] VAHDAT A, ANDERSON T, DAHLIN M, et al. Webos: Operating System Services For Wide Area Applications[C]// Proceedings of the Seventh IEEE Symposium on High Performance Distributed Systems. 1997: 52-63.

[14] DAN S, JAMES C, et al. A way forward: enabling operating system innovation in the cloud[C]// Proceedings of the 6th USENIX conference on Hot Topics in Cloud Computing (HotCloud'14). 2014.

[15] RAFAEL M, RUBÉN S, et al. IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures[J]. IEEE Computer, 2012, 45(12): 65-72.

[16] FABIO P, PETER B, et al. Toward a Cloud Operating System [C]// Proceedings of the 2010 IEEE/IFIP Network Operations and Management Symposium Workshops (NOMS Workshop). 2010.

[17] JAFFER S, KEDIA P, BANSAL S. Improving Remote Desktopping through Adaptive Record/Replay[C]// Proceedings of the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environment (VEE 2015). 2015: 161-172.

(上接第 21 页)

[9] BOHNET J, DÖLLNER J. Monitoring code quality and development activity by software maps[C]// Proceedings of the 2nd Workshop on Managing Technical Debt. New York, USA: ACM, 2011: 9-16.

[10] BARTON B, STERLING C. Manage project portfolios more effectively by including software debt in the decision process[J]. Cutter It Journal, 2010, 23(10): 19-24.

[11] HOLVITIE J, LEEPPÄNEN V. DebtFlag: Technical debt management with a development environment integrated tool[C]// Proceedings of the 4th International Workshop on Managing Technical Debt. Piscataway, USA: IEEE, 2013: 20-27.

[12] GARCIA J, POPESCU D, EDWARDS G, et al. Identifying architectural bad smells[C]// Proceedings of the 13th European Conference on Software Maintenance and Reengineering. Piscataway, USA: IEEE, 2009: 255-258.

[13] SLINGER S. Code Smell Detection in Eclipse[J]. Delft University of Technology, 2005, 10(1): 78-83.

[14] MORGENTHALER J D, GRIDNEV M, SAUCIUC R, et al. Searching for build debt: Experiences managing technical debt at Google[C]// Proceedings of the Third International Workshop on Managing Technical Debt. Piscataway, USA: IEEE, 2012: 1-6.

[15] GREENING D R. Release Duration and Enterprise Agility[C]// Proceedings of the 46th Hawaii International Conference on System Sciences. Piscataway, USA: IEEE, 2013: 4835-4841.

[16] ZAZWORKA N, IZURIETA C, WONG S, et al. Comparing four approaches for technical debt identification[J]. Software Quality Journal, 2014, 22(3): 403-426.

[17] GUO Y, SEAMAN C. A portfolio approach to technical debt management[C]// Proceedings of the 2nd Workshop on Managing Technical Debt. New York, USA: ACM, 2011: 31-34.

[18] ZAZWORKA N, SEAMAN C, SHULL F. Prioritizing design debt investment opportunities[C]// Proceedings of the 2nd Workshop on Managing Technical Debt. New York, USA: ACM, 2011: 39-42.

[19] SCHMID K. A formal approach to technical debt decision making[C]// Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures. New York, USA: ACM, 2013: 153-162.

[20] EISENBERG R J. A threshold based approach to technical debt [J]. ACM SIGSOFT Software Engineering Notes, 2012, 37(2): 1-6.

[21] FERNÁNDEZ-SÁNCYEZ C, GARBAJOSA J, IDAL C, et al. An analysis of techniques and methods for technical debt management: a reflection from the architecture perspective[C]// Proceedings of the Second International Workshop on Software Architecture and Metrics. Piscataway, USA: IEEE, 2015: 22-28.

[22] BOHNET J, DÖLLNER J. Monitoring code quality and development activity by software maps[C]// Proceedings of the 2nd Workshop on Managing Technical Debt. New York, USA: ACM, 2011: 9-16.

[23] LETOUZEY J L. The SQALE method for evaluating technical debt[C]// Proceedings of the Third International Workshop on Managing Technical Debt. Piscataway, USA: IEEE, 2012: 31-36.

[24] HOLVITIE J, LEPPÁNEN V. DebtFlag: Technical debt management with a development environment integrated tool[C]// Proceedings of the 4th International Workshop on Managing Technical Debt. Piscataway, USA: IEEE, 2013: 20-27.

[25] BARTON B, STERKUBG C. Manage project portfolios more effectively by including software debt in the decision process[J]. Cutter IT Journal, 2010, 23(10): 19.

[26] FALESSI D, SHAW M A, SHULL F, et al. Practical considerations, challenges, and requirements of tool-support for managing technical debt[C]// Proceedings of the 4th International Workshop on Managing Technical Debt. Piscataway, USA: IEEE, 2013: 16-19.