

一种基于 Android 平台的云客户端实现方案

金思宇¹, 吴振宇¹, 沈苏彬²

(1. 南京邮电大学 物联网学院, 江苏 南京 210003;
2. 南京邮电大学 计算机学院, 江苏 南京 210003)

摘要: 由于受到存储资源和计算资源匮乏的影响, 移动设备无法进行大文件的存储操作和计算密集型任务的执行操作, 这种资源受限的问题一直是影响移动设备用户体验的关键问题之一。为此移动设备生产厂商长期以来试图通过不断改良硬件来解决这一问题。文中试图通过利用云计算技术解决移动设备存储和计算资源受限的问题。首先成功地设计并开发了支持 OpenStack、AWS 和阿里云进行云存储操作的云客户端原型, 接着提出了一种移动应用计算减荷的解决方案。利用云存储技术和本地文件浏览器将本地的大文件上传到云端; 通过拦截进程间通信, 将耗时任务减荷到云端运行。通过实验验证了云客户端原型可以成功地将移动设备上的大文件保存到云端, 提高移动设备的运行效率。

关键词: 云计算; 云存储; 移动云; Android 客户端

中图分类号: TP301

文献标识码: A

文章编号: 1673-629X(2016)11-0019-06

doi: 10.3969/j.issn.1673-629X.2016.11.005

A Cloud Client Implementation Scheme Based on Android Platform

JIN Si-yu¹, WU Zhen-yu¹, SHEN Su-bin²

(1. School of IoT, Nanjing University of Posts and Telecommunications,
Nanjing 210003, China;
2. School of Computer, Nanjing University of Posts and Telecommunications,
Nanjing 210003, China)

Abstract: Because mobile applications which need lots of storage space or computing power can not be efficiently executed, resource-constrained problem is always faced by mobile devices, and it affects the user experience. For this mobile device manufacturers have tried to improve the hardware continuously to solve this problem. It tries to use the cloud computing to solve this problem in this paper. By storing big files and offloading computationally intensive tasks to the cloud, it can improve the performance of mobile device. Moreover, a prototype of cloud client is designed which can supports three different cloud services, such as OpenStack, AWS and Aliyun. The cloud storages and file explorers are used to store big files in the cloud, using IPC interception to offload computationally intensive tasks to cloud. By the experiment, it concludes that cloud client can store big files to cloud and improve the performance of mobile device.

Key words: cloud computing; cloud storage; mobile cloud; Android client

0 引言

以智能手机为代表的移动设备由于其安装了丰富的应用得到了飞速的普及和发展。随着各种各样的移动应用的涌现, 用户对移动设备的存储和计算能力的要求越来越高。但移动设备硬件的发展速度常常无法满足移动应用对于存储和计算资源的要求, 一些应用由于受到移动设备存储空间小、处理能力低、内存有

限、网络连接不稳定和电池使用时间短等限制而无法在移动设备上运行^[1]。

目前, 移动设备的生产厂商力图不断通过改良移动设备的硬件来提高移动设备的运行效率和计算能力。这导致了两方面的问题: 由于尺寸限制, 硬件层面的改良很难完全满足应用的需要; 一味增加硬件的性能, 在运行对移动设备性能要求不高的应用时造成了资源浪

收稿日期: 2016-01-15

修回日期: 2016-04-20

网络出版时间: 2016-10-24

基金项目: 国家自然科学基金资助项目(61502246); 江苏省未来网络前瞻性研究资助项目(BY2013095-1-08); 南京邮电大学自然科学基金(NY211115)

作者简介: 金思宇(1991-), 男, 硕士研究生, 研究方向为云计算与云存储; 吴振宇, 讲师, 研究方向为不确定性人工智能、数据挖掘; 沈苏彬, 博士生导师, 研究方向为计算网络、下一代电信网及网络安全。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20161024.1105.020.html>

费,增加了设备厂商的成本,最终导致移动设备用户成本的增加。

近年来,云计算技术发展迅速,它使用户可以在任何时间任何地点按需获得所需资源。驱使云计算诞生和发展的因素有:网络计算的发展、数据存储和传输中高新技术的出现、WEB2.0 的出现。其中虚拟化技术的发展对云计算影响巨大^[2]。

云计算具有虚拟化、分布式和动态可扩展性等特征。虚拟化指对云进行的管理、扩展、迁移和通过虚拟平台进行的备份等操作都可以在虚拟层完成;分布式指的是计算所用的物理节点是分布式的;动态扩展指的是通过动态扩展虚拟化层达到承载应用的目的。虚拟化技术通过将物理资源适应逻辑进行统一表示,打破了原有物理资源的壁垒。

云存储和计算减荷技术可以从软件的角度解决移动设备资源受限的问题。云存储将数以万计的廉价存储设备结合成一个庞大的存储资源池,用户可以按照自己的需求使用池中的存储资源^[3],它的三个特征是:分布于网络(互联网或局域网)、易于扩展和易于管理^[4]。计算减荷可以将资源敏感的计算任务从移动设备迁移到资源丰富的云端,增强移动应用的性能、减少电量消耗。

文中分别从存储和计算的角度出发,研究如何将云中的资源提供给移动设备使用。基于 Android 平台创新性地设计并实现了一种支持三种主流云平台的移动云客户端,提出了一种适用于 Android 设备的进程间通信拦截方案,利用该方案设计开发了客户端的云存储功能和计算减荷功能。

1 相关技术分析

1.1 移动云架构

移动云的定义是:一种云计算技术和移动设备的整合,旨在在存储、计算、能源和情景感知方面使移动设备具有充足的资源^[5]。

移动云计算是用云计算技术和移动计算技术共同发展而来,这个跨学科的技术也被称为 mobicloud computing^[6]。

移动云计算具有两种不同的架构:一是以基础设施为基础的,二是专用的云。在基于基础设施的云计算架构中,硬件基础设施仍然是固定的,并且向移动用户提供服务。相对的,专用的云指的是一组移动设备组成的云平台,其基于局域网或因特网向其他移动设备提供云服务^[7]。文中讨论的移动云是基于基础设施的云。

移动设备可以通过两种方式接入到云中,分别是通过移动网络和热点接入,如图 1 所示。

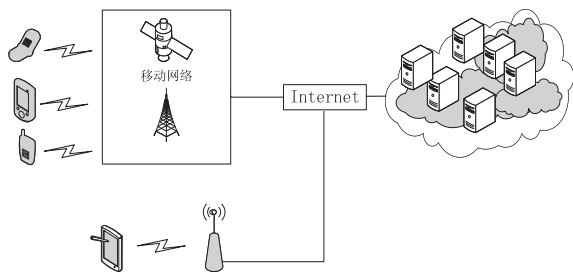


图 1 移动云架构

与 3G 移动网络相比,WiFi 具有更低的延迟和能耗^[7],所以在接入网络方式上用户应首选 WiFi。文中对移动网络环境下的上传下载操作进行了相应优化。

1.2 主流云平台介绍

OpenStack 为公有云和私有云提供了可扩展的弹性云平台,可以实现简单、大规模可扩展的云。图 2 展示了 OpenStack 的基本架构。

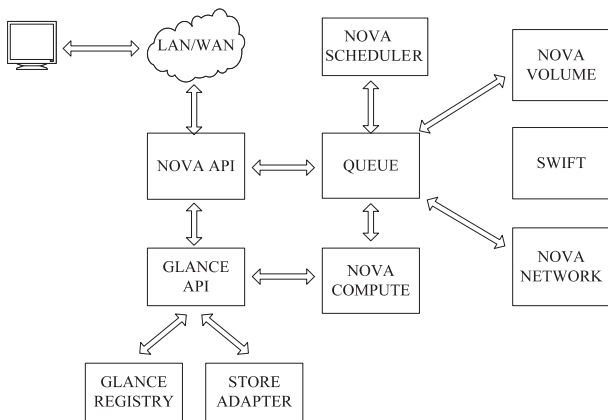


图 2 OpenStack 架构图

其中,Nova 作为云平台的控制器,具有 Nova-API,负责根据用户的请求部署资源;Queue,是通过 RabbitMQ 实现的消息队列,用于实现远程程序请求;Nova 计算模块,负责开始或终止计算节点上的虚拟机实例;Nova 网络模块,负责为虚拟机实例管理 IP 地址以及建立虚拟局域网;Nova 中卷管理模块负责关键虚拟机的卷的创建和挂载等操作;Nova 调度模块,负责根据可用的计算节点调度计算资源;Nova 数据库^[8]。Swift 对象存储系统非常易于扩展,其包含一个代理服务器、一个对象服务器、一个账户服务器、一个容器服务器。Glance 是镜像存储组件,用于在虚拟机的镜像中创建用于查找和检索的系统。Dashboard 实现一个基于 Web 的用户接口。

亚马逊 AWS 最早通过网页的形式为用户提供 IT 资源服务,可以说是云计算的鼻祖。阿里独立开发的飞天平台是阿里云服务的基础,其可以有效地管理数据中心中的服务器集群,很好地做到了隐藏层硬件实现的细节,使用户便于使用存储、计算资源。

1.3 Android 系统架构

Android 平台基于 Linux 内核,其将应用、应用系

统内核和中间件进行了分层。其基本架构见图 3。

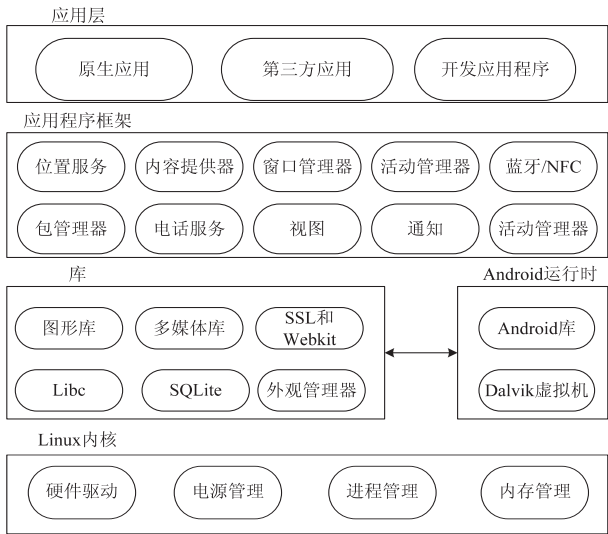


图 3 Android 系统架构

Linux 内核层:主要负责保障系统运行的安全性、稳定性,管理基本的内存使用,管理程序的进程、网络堆栈和处理驱动模块。Android 系统的 Linux 内核层放弃了使用虚拟内存文件,提高了文件系统的工作效率^[9]。

Android 运行时及其他库层:Android 平台上每个应用都有自己专有的 Dalvik 虚拟机实例。经过优化使每个虚拟机占用很小的内存。

应用框架层:使开发人员有权限访问框架中所有 API。

应用程序层:即系统自带的用于实现一些重要基础功能的应用,开发新应用时可以很方便地与这些核心应用交互^[10]。

2 云客户端总体设计

将文中设计的移动云客户端功能视图划分为如下模块:用户认证、云存储、信息管理、计算减荷和其他功能。

2.1 用户认证模块

用户认证模块主要负责获取用户输入的账号密码和与云端认证模块交互,判断认证结果后返回给客户端。其首先检验账号密码的合法性并判断移动设备目前的联网状态,接着和云平台交互进行用户认证操作,最后用户认证模块对认证结果进行判断,如果认证成功则向客户端返回成功认证的消息,若认证不成功则判断失败原因,如用户账号密码错误或是网络因素,返回相应的失败原因给客户端。

2.2 云存储模块

云存储模块主要负责根据用户选择的本地或云端文件完成对应的上传或下载操作。由于在移动网络下进行云存储操作相比 WiFi 环境具有低带宽、高时延的

缺陷,所以针对移动网络进行了优化设计。

数据上传模块中,首先实现对本地设备文件进行浏览的功能。其次,允许用户选择需要上传的文件,并选择是执行直接上传操作还是压缩上传操作。针对移动网络环境,文中优化了上传机制,在该环境下会优先推荐用户使用压缩上传功能,将待上传文件压缩成 ZIP 格式再进行上传。数据上传模块支持用户多选在不同路径下的文件,在一次上传操作中打包将这些文件上传云端。

数据下载模块,首先会遍历云端已有的文件和文件存储结构,接着将遍历结果保存在客户端的数据库中。和数据上传模块一样,数据下载模块也考虑移动网络环境下的优化。通过实验发现,在用户进行了上传操作之后云端存储的文件才有可能发生变化。所以数据下载模块在客户端软件关闭后初次启动或发生了上传操作的情况下,才会查询云端文件存储结构。查询结果会保存在数据库中,方便用户下次下载操作时使用。

2.3 信息管理

信息管理模块主要负责收集云端和本地设备的运行信息和资源信息,并展示给用户。

云资源信息管理模块主要负责与云平台的管理软件通信,收集存储和计算资源的使用情况,并在移动设备上向用户展示。本地设备信息管理模块主要负责收集移动设备当前的存储、内存、电量和 CPU 使用率情况,并用图表向用户展示。用户在使用客户端时,可以通过这些信息决定是否需使用云存储服务或将计算减荷到云端。

2.4 计算减荷

计算减荷模块主要负责将本地移动设备中计算密集型任务卸载到云端运行。目前计算减荷的解决方案可以分为两种:一种是在移动应用开发时使用可以进行计算减荷的框架和模式,这种框架需要对 Android 系统或应用本身进行修改,如文献[11-12];另一种是在不改变移动应用本身和 Android 系统本身的前提下,通过在移动设备上的客户端软件,监测移动应用的行为,当其需要执行计算密集型任务时,将该任务卸载到云端。

文中设计的移动客户端软件更加倾向于第二种计算减荷解决方案。因为移动设备的流行很大程度上是由于其具有丰富多样的移动应用,要求每一个移动应用的开发者都遵循一个统一的框架开发,来开发支持计算减荷的应用,在生产环境下的成本是巨大的。

文中的计算减荷模块是一种基于 Android 系统组件接口的计算减荷方式,客户端将监听 Android 系统中 Activity 组件对 Service 组件的调用并利用动态代理

拦截 Binder 请求,根据移动设备目前的运行情况或提示用户决定是否需要卸载计算任务到云端。

2.5 其他功能

其他功能主要包括用户反馈模块和关于程序模块。客户端程序在运行时产生的日志文件都会被重定向到设备本地文件中,用反馈模块将运行日志和客户反馈信息一同发送给开发者。关于程序模块主要负责显示客户端的开发者信息和版本号。

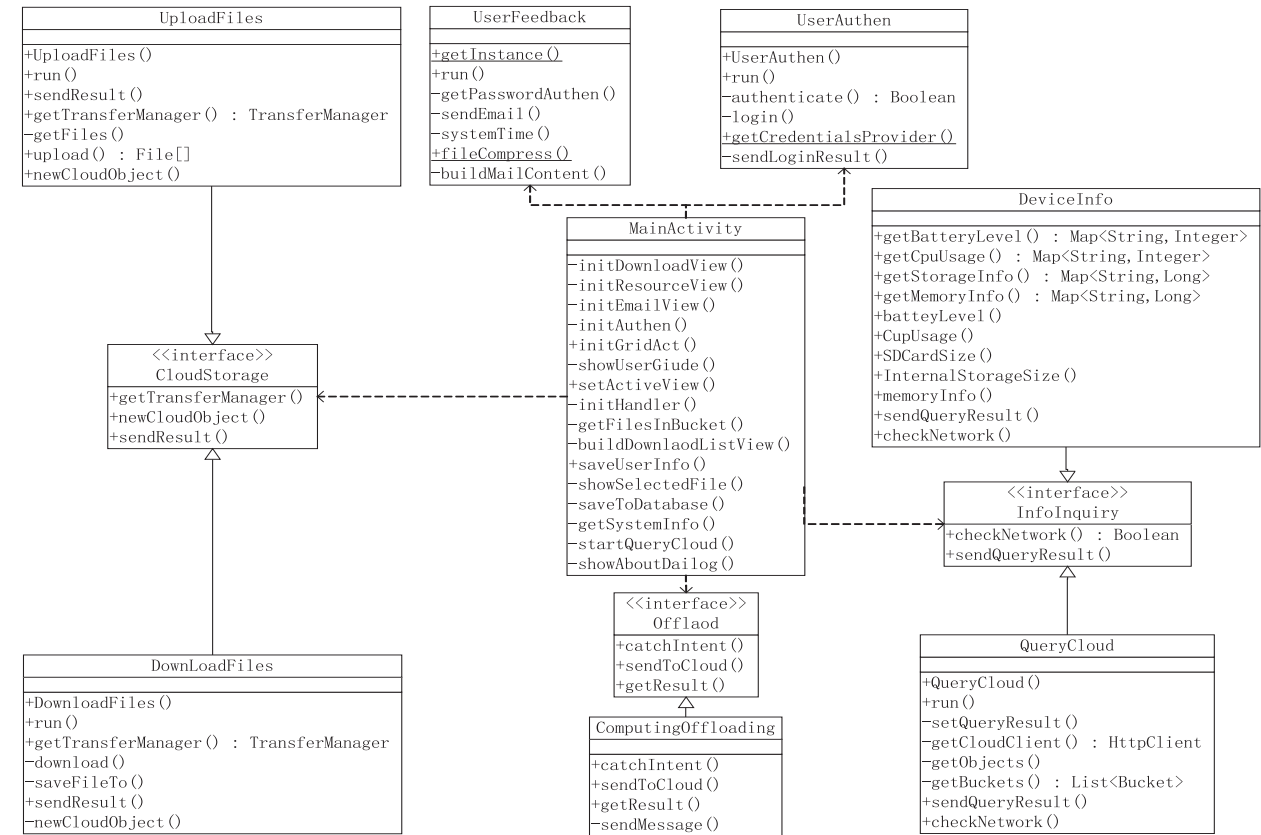


图 4 云客户端类图

为了支持三种不同的云平台而开发三套不同的上传下载和登录逻辑会造成大量的代码冗余问题,不利于维护并且导致程序臃肿。文中通过对客户端软件的基本操作进行抽象,提取出三个对应的接口,分别是 CloudStorage、Offload 和 InfoInquiry,分别表示云存储、计算减荷和信息查询模块。程序的实现中,通过实现相应的接口,自定义的类可以提供相应操作所需方法,如 CloudStorage 接口由 UploadFiles 和 DownloadFiles 两个类实现,这两个类执行上传和下载逻辑。这种设计体现了经典的开闭原则,主 UI 类中使用 setter 方法将用户的数据传递给相应的类,每个类在完成数据处理之后,提供相应的 getter 方法或 Android 的消息传递机制将结果返回给主 UI 类。

3.2 云平台接口分析

OpenStack 是当今最流行的开源云平台,而 AWS

3 云客户端方案实现

3.1 UML 建模

从实现视图划分,云客户端具有 UI、云端交互、信息采集、数据展示和辅助功能模块。文中实现的云客户端基于 Android 系统使用 Java 语言开发。根据面向对象的思想,对开发中需要的对象进行抽象得到客户端中主要的类结构关系,如图 4 所示。这里简要列出了主要的类和这些类中主要的方法,省略了一些非主要功能函数和类。

和阿里云分别是国内外最流行的公有云平台。为了增加客户端的实用性和通用性,文中设计兼容了这三种最主流的云平台,也是文中的创新点之一。针对 OpenStack、AWS 和阿里云三种不同的平台,以实现云存储功能为目的,对三种云提供的接口进行了研究。在接口的支持方面,接口最为完善和强大的是 AWS 平台,其支持 Android、iOS、Java、Python、PHP 等几乎所有常用的语言和平台。其次是 OpenStack 平台,与 AWS 相比 OpenStack 只是缺少了对于 iOS 操作系统的支持,但是值得一提的是 OpenStack 官方指定的 Java SDK 为 Apache Jclouds,也正是文中客户端选用的 Java SDK。Apache Jclouds 是 Apache 基金会推出的一个开源项目,旨在支持所有主流的云平台。通过使用 Jclouds SDK 可以和 30 种不同的云平台进行交互,包括 Amazon、Azure、OpenStack 和 Google 等。目前,其对各种云

平台的计算服务和存储服务提供了成熟的支持,其提供的计算服务接口可以允许用户一次性开启多台虚拟机,并在其中安装制定的软件;存储服务可以简化用户对存储容器的管理,为用户提供一个存储容器的直观的视图。

相比于上述两个云平台,阿里云提供的接口十分有限,在文中进行开发和实验时其仅支持 Java、Python、.NET 和 PHP 四种接口。

在接口的使用效率方面,做的最好的依然是 AWS,利用其为 Android 提供的 SDK,用户可以通过多种方式进行身份认证,并且认证成功后 AWS 将返回一个用于缓存认证结果的对象,使用该对象可以创建对应于该账户的证书缓存,再使用传输管理器对象提供的上传和下载方法即可实现数据的上传下载。利用 Apache 提供的 Jclouds SDK 与 OpenStack 的 Swift 组件交互实现云存储功能也十分简单。首先传入 Swift 组件的地址或 URL 和认证所需要的用户名密码获取 Swift API 对象,接着调用上传、下载或创建容器方法并传入创建好的用户选项对象即可完成相应操作。

由于 Android 应用使用 Java 语言进行开发,所以基于阿里云的云存储开发文中选择其提供的 Java SDK。但在实验中发现,其在 Android 设备上运行时会报无相应方法错误。进一步实验发现,这一问题是由于新建阿里云 OSSClient 内部通过 SDK 中的 ThreadSafeClientConnManager 构造,该方法其实生成一个 HttpClient 对象,为此阿里云 SDK 和 Android 平台都集成了 Apache 的 HttpClient 类库,导致阿里云封装的代码中调用的方法在 Android 上运行时无法找到。最后,使用了一个第三方的 SDK 与阿里云进行交互,同时,阿里已经陆续增加了其云存储 SDK,将逐步支持更多的平台。

3.3 主要功能实现

云存储模块分为数据上传和数据下载两个功能。在实现数据上传功能的过程中,文中首先实现了一个基于 Android 移动设备文件浏览器,可以向用户展示本地所有文件的绝对路径和文件名。在用户选择了相应的文件后,该文件的绝对路径和文件名将被传递到上传模块中,上传模块判断用户是否选择压缩操作,并生成相应的压缩文件对象,接着从用户认证模块获取已经通过认证的云平台传输管理器对象,调用其提供的上传方法进行文件上传操作。

文件下载功能,首先需要通过信息查询模块,查询云端目前保存的文件以及保存文件的存储结构。为了节省移动设备的资源,查询操作只有在客户端运行后首次进行下载操作或进行了上传操作之后才会运行,并且客户端会将最新的查询结果保存在本地 SQLite

数据库中作为缓存。这里使用 SQLite 这一 Android 提供的轻量级数据库,相比于传统的 SQL 数据库,SQLite 更加轻量更加适合嵌入式设备,在运行时仅仅需要百 kb 级的内存^[13]。并且 Android 为其提供了 SQLite Helper 类,不仅可以使使用传统的 SQL 语句,而且可以使使用面向对象的方法对其进行操作。用户通过查看数据库获得云中文件存储结构信息并选择下载。

信息查询模块通过和云平台的管理模块进行通信,获得云中目前的存储和计算资源使用情况;通过与 Android 交互,查询移动设备的内存、存储、CPU 和电量信息,并将这些信息汇总到图表中展示给用户。

传统的计算减荷解决方案通过接收应用发送的 Intent 请求或监听 Activity 和 Service 之间的调用实现。这种解决方案的前提是移动设备上所有的应用都会发送标准的 Intent 请求并且移动云客户端可以准确拦截所有的 Intent 请求。通过实验发现,由于 Android 系统本身的碎片化现象和 Android 应用的多样化,不同品牌的设备和不同开发者开发的应用,对同一个 Intent 请求做出的响应是不同的,并且开发者并不会遵从同样的开发原则。使用一个客户端接收所有应用发送 Intent 请求可行性较低,这并不是一种通用的解决方案。文中提出通过拦截并监听计算密集型任务的 Binder 调用进行计算减荷。Binder 是 Android 的 IPC 机制,由用户层的 libbinder.so 和 binder 驱动实现。

常见 IPC 拦截解决方案是在主调组件中注入并加载用于拦截的模块,在主调组件 GOT 表中确定被调组件地址,重定向到新的地址^[14]。由于 Android 和 Linux 的内存布局不同,所以常见的方法在 Android 平台上无法实现。为此,文中通过实验提出如下适合 Android 的拦截方式:

首先,运行注入程序,通过 ptrace 停止并附加系统服务;注入内核代码加载共享库接触附加服务,调用共享库中函数;该函数将自己的地址和被替换的按时地址暴露到 Android Property;注入程序通过 Android Property 获得新函数和替换函数的地址并再次附加系统服务,定位 got 文件中原始函数的地址并将其替换成新地址;最后附加系统服务并重新运行。

通过上述步骤将活动对服务的调用请求由客户端发送到云端。云端运行的虚拟在接收到调用后会启动其本地的服务,执行该计算任务,执行完毕后将执行结果返回给客户端,再由客户端返回到作为服务调用方的活动组件中。

4 测试和分析

文中涉及的客户端测试环境如下:

(1)移动设备:Android 4.3 系统,高通骁龙 800 四

核处理器,运行内存 3 GB,电池 3 200 mAh,外部存储 32 G。

(2)云平台:OpenStack G 版部署 Intel core i3,内存 8 G,磁盘 500 GB 的物理机上,AWS 和阿里云均使用其官方提供的平台。

在移动设备上运行文中设计的客户端软件,其主界面如图 5 所示。在列表试图中包括了用户认证、云资源管理、数据上传和下载、运行状态等 8 个不同的按钮。



图 5 客户端主界面

为了验证客户端设计的正确性,在移动和热点接入环境下,在实验移动设备上运行客户端原型。首先在移动设备上完成用户认证操作,获取云端资源的使用权,接着选取本地测试文件 testCloudStorage 分别上传文件到 OpenStack、AWS 和阿里云,然后选择云中保存的测试文件下载到本地的操作也可以正确完成。

通过实验发现,三种不同的云平台可以提供可伸缩的对象存储服务,从存储效率上,由于 AWS 的云服务器位于境外,所以通信速率要低于另外两个云平台。由于三种云提供的都是基于对象存储服务,所以都需要在全球唯一的 bucket 中对数据对象进行存储。

5 结束语

文中对 Andriod 系统架构和云计算技术进行了阐述,研究在 Android 移动设备上使用云存储服务和云计算减荷服务的方法,通过开发移动云客户端,实现了在移动设备上使用 OpenStack、AWS 和阿里云三种不

同的云平台提供的存储资源,并提出了一种移动设备计算减荷的解决方案。通过对开发的客户端原型进行测试,结果表明该设计方案合理可行,开发的客户端能够为移动用户提供高效率的云存储服务。接下来的研究和开发将针对计算减荷这一课题展开,通过实验开发具有移动设备计算减荷功能的客户端原型。

参考文献:

- [1] Vallina-Rodríguez N, Crowcroft J. Energy management techniques in modern mobile handsets[J]. IEEE Communications Surveys & Tutorials, 2013, 15(1): 179-198.
- [2] Zhang S, Zhang S, Chen X, et al. Cloud computing research and development trend[C]//Second international conference on future networks. [s. l.]: IEEE, 2010: 93-97.
- [3] Amazon simple storage service[EB/OL]. 2011-12-08. <http://aws.amazon.com/s3/>.
- [4] 武永卫, 黄小猛. 云存储[J]. 中国计算机学会通讯, 2009, 5(6): 44-52.
- [5] Othman M, Madani S A, Khan S U. A survey of mobile cloud computing application models[J]. IEEE Communications Surveys & Tutorials, 2014, 16(1): 393-413.
- [6] Huerta-Canepa G, Lee D. A virtual cloud computing provider for mobile devices[C]//Proceedings of 1st ACM Workshop on mobile cloud computing & services: social networks and beyond. [s. l.]: ACM, 2010: 6.
- [7] 吴吉义, 平玲娣, 潘雪增, 等. 云计算: 从概念到平台[J]. 电信科学, 2009(12): 23-30.
- [8] Rimal B P, Choi E, Lumb I. A taxonomy and survey of cloud computing systems[C]//Fifth international joint conference on INC, IMS and IDC. [s. l.]: IEEE, 2009: 44-51.
- [9] Meier R. Professional Android 4 application development[M]. [s. l.]: John Wiley & Sons, 2012.
- [10] 韩超. Android 系统原理及开发要点详解[M]. 北京: 电子工业出版社, 2010.
- [11] Ghorpade S, Chavan N, Gokhale A, et al. A framework for executing android applications on the cloud[C]//International conference on advances in computing, communications and informatics. [s. l.]: IEEE, 2013: 230-235.
- [12] Kovachev D, Yu T, Klamra R. Adaptive computation offloading from mobile devices into the cloud[C]//IEEE 10th international symposium on parallel and distributed processing with applications. [s. l.]: IEEE, 2012: 784-791.
- [13] 郭霖. 第一行代码—Android[M]. 北京: 人民邮电出版社, 2014.
- [14] Chen E Y, Itoh M. Virtual smartphone over IP[C]//IEEE international symposium on world of wireless mobile and multimedia networks. [s. l.]: IEEE, 2010: 1-6.