计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(9):109−115 [doi: 10.15888/j.cnki.csa.005950]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于服务化技术的个人数据资源共享方法①

宋梦蝶[1,2], 王　枫[1], 武延军[1]

[1](中国科学院软件研究所 基础软件国家工程研究中心, 北京 100190)
[2](中国科学院大学, 北京 100049)

摘　要: 随着信息网络的高速发展, 电子产品普及率日益提高, 个人数据资源分散化的存储在各种不同的设备之中, 设备间方便高效的数据共享成为亟待解决的问题. 针对现有的数据共享方法不能满足个人以及协同工作组内数据高效便捷的共享需求, 运用服务化技术实现数据资源共享方法. 本方法采用端对端的数据共享方式, 使局域网环境下参与共享的个人及协同工作组内设备对等的运行数据 Web 服务化, 服务发布和服务发现三个基本模块, 在各模块的配合下实现多设备间的数据资源共享. 相比常用的数据共享方法, 具有无需搭建额外的硬件环境, 共享效率高, 支持无需下载的在线访问, 避免了三方数据泄露风险以及系统资源占用率低等优势. 方法有效的解决了多设备间高效数据共享面临的数据动态变化, 信息一致性维护, 节点异构, 代价可接受等挑战.

关键词: 多设备数据共享; 数据资源服务化; 服务发现

引用格式: 宋梦蝶,王枫,武延军.基于服务化技术的个人数据资源共享方法.计算机系统应用,2017,26(9):109–115. http://www.c-s-a.org.cn/1003-3254/5950.html

Sharing Method of Personal Data Resources Based on Service-Oriented Technology

SONG Meng-Die[1,2], WANG Feng[1], WU Yan-Jun[1]

[1](National Engineering Research Center for Fundamental Software, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)
[2](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: With the rapid development of information technology, the number of electronic products shows a great increment. Personal data resources are distributed in various devices and therefore efficient data sharing among devices becomes a problem. Given the fact that the existing methods can not meet the demand of efficient data sharing in this circumstance, this paper proposes a personal data resources sharing method based on service-oriented technology. It is an end-to-end method of sharing data that every end runs three basic modules, which are data web service module, service publication module and service discovery module. Compared with the existing data sharing scheme, it has advantages of avoiding additional hardware maintenance, high sharing efficiency in Local Area Network, supporting online access, no need to worry about the data leakage by third parties and a low system resource occupancy rate. The method is an effective solution in the data sharing environment to meet the challenges of frequent changes, information consistency maintenance, heterogeneous nodes and acceptable cost.

Key words: cross-device data sharing; data resource as a service; service discovery

引言

随着信息网络的高速发展, 智能手机, 平板电脑, 笔记本电脑, 办公台式机等个人电子设备普及率极高, 是人们日常生活和协同工作环境中密不可分的组成部分. 而随之带来了一个令人备受困扰的问题, 同属个人或同一协同工作组中的数据资源, 包括文档、图片、

① 基金项目: 中国科学院先导专项(XDA06010600)
收稿时间: 2016-12-27; 采用时间: 2017-01-18

音频、视频等, 如何在分散化的多种设备和应用间方便高效的共享?

现有的针对个人和协同工作组的数据共享方法主要有以下几类, 他们都可以实现数据在多设备间的共享, 但也都有着自身的局限性. 使用数据线, U 盘, 移动硬盘等外部设备进行传统的数据传输, 这种共享方式依赖外部设备, 过程繁琐而且易引发数据丢失和计算机病毒的传播等问题; 安装相同的具有数据传输功能的应用软件, 例如微信, QQ, 飞鸽传书和电子邮件等, 这种方式依赖于第三方应用的文件传输功能, 使用下载后浏览的方式会占用设备较多的存储空间以及存在第三方应用数据泄露的风险; 使用云存储[1]的方式是目前较为通用的数据共享方式, 具体表现为云盘, 这种方式依赖于服务商提供的服务和外部网络, 有着共享数据规模大的优势, 但共享效率受网络带宽的影响较大且同样存在云服务商数据泄露的风险; 使用内部的 FTP 服务器等进行数据资源的上传和下载, 可以保证共享的效率且减少数据泄露的风险, 但随之而来的是 FTP 服务器的搭建和维护的成本, 而且集中式存储的通病, 中心服务器的性能瓶颈以及单点失效等问题也需要考虑.

针对现有数据资源共享方法存在的不足, 本文运用服务化技术实现数据资源共享方法, 它采用端对端的数据共享方式, 使局域网环境下参与共享的个人及工作组内各设备在系统级对等的运行数据 Web 服务化, 服务发布和服务发现三个基本模块, 在各模块的配合下实现多设备间的数据资源共享. 本方案较常见的数据共享方法, 具有无需搭建额外的硬件环境, 共享效率高, 支持无需下载的在线访问, 避免了三方数据泄露以及系统资源占用率低等优势, 为多设备乃至多应用间的数据共享提供一种新的思路和方法.

本文的结构如下: 第 1 节为研究背景和相关工作, 第 2 节为个人数据资源共享方法的设计, 第 3 节为系统各模块的具体实现以及运行效果, 第 4 节为方法的功能及性能评估, 第 5 节为本文工作的总结以及对未来工作的规划和展望.

1 研究背景和相关工作

1.1 多设备数据资源共享方法

依赖数据线, U 盘, 移动硬盘等数据传输媒介的数据共享方式是一种传统的数据共享方式, 这种方式有着显而易见的缺点, 它依赖于外部设备, 操作繁琐而且 U 盘等设备经常出现不被主机识别或是病毒感染等意外情况. 因此, 虽然这种方式仍然很大程度上的为人们所使用, 却在近年来越来越多的被其他方式所替代.

安装相同的具有数据传输功能的应用实现多设备间的数据共享也是一种常见的方式, 在局域网和广域网中都有相关的实现, 日常的即时通讯软件, 例如 QQ, 微信, 飞鸽传书等也具有文件传输的功能. 这种方式除了要求安装相同的应用, 我们的数据资源还需要经过其他应用的转发, 无可避免的存在着数据泄露的风险. 另一方面, 数据的接收方通常只能通过下载到本地的方式进行文件的查看, 无法实现数据的在线访问, 灵活性低, 且在外网环境下同样存在着数据传输效率低的问题.

随着云计算技术的发展, 云存储的方式是近年来使用广泛的数据共享方式. 云存储的本质上是一个分布式的文件系统, 常见的设计思想分为集中式元数据管理和一致性哈希管理两种[2]. 前者比较有代表性的系统是 GFS(Google File System)和 HDFS(Hadoop Distributed File System), 后者比较有代表性的是 Amazon S3(Simple Storage Service). 集中式元数据管理是一种主从结构, 主节点存储着一张 key-value 的表格, 记录着数据实际存储的位置, 所有获取数据的请求都需要和主节点通信, 从而获得数据的实际地址. 一致性哈希管理不存在主节点, 是根据一致性哈希算法来解决分布式存储的节点分配问题. 云盘, 是一种云存储的具体表现形式, 是云服务商所提供的数据存储服务, 具有存储空间巨大, 没有冗余存储等优势. 云盘通常需要通过外部网络进行访问, 受网络带宽影响使云盘共享数据的效率较低. 并且在这种条件下, 数据的拥有者失去了对数据的控制, 数据的安全性完全取决于云服务商的可靠性, 基于这种方式的数据共享安全性[3,4]一直以来也是学界研究的热点问题. 近年来国内外数据泄露事件层出不穷, 在 Verizon 最新发布的《2016 数据泄露报告》[5]中也出现了对云服务商的一些建议, 来减少云存储潜在的安全问题, 之前年度的报告中也指出云服务商需注意内部人员因经济目的泄露用户数据的行为.

在数据安全性以及传输效率要求较高的环境下, 搭建内部的 FTP 服务器进行数据资源的共享是一个可行的实现方案. 但针对本文的个人数据资源共享和协

同工作的工作组内的数据共享, 搭建和维护 FTP 服务器的成本较高, 而且无可避免的存在着服务器的性能瓶颈和单点失效等集中式存储的弊端.

本文运用服务化技术解决个人数据资源共享问题, 为多设备间的数据共享提供一种新的思路.

1.2 服务发布和服务发现方式

服务发布模块和服务发现[6]模块是负责在共享环境中将数据服务化后得到的数据资源地址进行发布和发现.

Google 发起的“The Physical Web”[7]项目是使物联网环境下的每个智能设备用 URL(Uniform Resource Locator, 统一资源定位符)来标识自己, 之后用户使用 BLE(Bluetooth Low Energy, 低功耗蓝牙)这种服务发现技术获取到一个所有服务的 URL 列表, 这样一来, 使用智能设备的体验就和在网站上使用各种超链接差不多了. 该项目的 Android 端应用可以在 Google Play Store 里找到, 它的开源实现里提供了 BLE, mDNS, SSDP 三种 URL 发布和发现的方式. 本文借鉴了这种物联网中的服务发布和资源定位的方法, 针对个人数据共享环境和协同工作环境的具体场景和网络条件, 采用了 SSDP 实现服务发布和发现.

SSDP[8](Simple Service Discovery Protocol, 简单服务发现协议)是构成 UPnP(Universal Plug and Play, 通用即插即用)的核心协议, 是一个应用层协议, 建立在 HTTPMU(HTTP over Multicast UDP)和 HTTPU(HTTP over Unicast UDP)之上. 当一个设备接入网络中后, 它会向网络里通过 SSDP 在 HTTPMU 之上多播它的服务, 类似的, 当一个 UPnP 的控制点被加入到网络中, 它通过 SSDP 在 HTTPMU 上搜索设备, 而每一个在多播端口上监听的设备都会通过 SSDP 在 HTTPU 上进行单播的应答. 具体而言, SSDP 协议的几种典型消息及其应用场景为: 当一个客户端接入网络的时候, 它可以向一个特定的多播地址的 SSDP 端口使用 M-SEARCH 方法发送 ssdp:discover 消息; 当设备监听到这个保留的多播地址上由控制点发送的消息的时候, 设备会分析控制点请求的服务, 如果自身提供了控制点请求的服务, 设备将通过单播的方式响应控制点的请求; 当一个设备接入网络的时候, 它应当向一个特定的多播地址的 SSDP 端口使用 NOTIFY 方法发送 ssdp:alive 消息; 当一个设备计划从网络上卸载的时候, 它也应当向一个特定的多播地址的 SSDP 端口使用 NOTIFY 方法发送 ssdp:byebye 消息. 理解协议的基本场景对有利于理解服务发布及发现模块的实现.

REST 是 Roy Fielding 在 2000 年他的博士论文[9]提出的一种面向资源的软件架构风格, REST 和 SOAP 的比较在文献[10]中有详细阐述, 二者作为数据 Web 服务化的方式也分别被文献[11]以及文献[12]所采用, 基于 REST 而非 SOAP 的 UPnP 早在文献[13]被提出, 文章指出可以通过服务发现 URL 的方式获取更多的信息. 本文服务发现及发布模块实现了 SSDP 协议的基本使用场景, 配合 REST 而非 SOAP 对资源定位及访问.

2 个人数据资源共享方法的设计

实现局域网环境下个人以及协同工作组的多设备间高效的数据资源共享, 主要面临以下几个方面的挑战[14]:

1) 共享数据动态变化

共享的数据内容动态变化, 例如协同工作组内一份设计文档, 在一段时间内会不断进行改动, 需保证共享的各方查看的是最新内容而非历史版本.

2) 信息一致性维护

共享数据的各方关系上是对等的, 获得的共享数据资源是一致的, 共享资源的添加和删除需要及时的被各节点感知.

3) 节点异构

各终端设备是多样的, 智能手机, 平板电脑, 笔记本电脑以及工作台式机, 不同设备的系统和资源都不同, 方法应该具有通用性, 适用于不同的系统和平台.

4) 代价可接受

代价不能过高, 尤其是对于移动端这类资源较为稀缺的节点, 实现和运行数据共享方法不能占用设备过多的资源.

针对以上几点, 本文运用服务化技术实现个人数据资源共享方法. 为了应对数据的动态变化并保证实时获取数据的最新版本, 对资源的共享和定位是使用数据资源地址而非传输数据本身来实现的, 数据内容本身会发生变化, 但只要数据地址不变, 使用该地址获取的就是当前共享的最新版本; 各节点对等的运行服务发布和服务发现模块来得到一致的资源地址列表, 资源的添加和删除可以从实时得到的资源列表中体现, 从而维护了信息一致性; 方法的模块化架构以及模块

间的低耦合设计适用于各类异构节点, 各模块的功能实现上不限于特定的技术和运行平台; 针对移动端等资源受限的平台, 各模块选用轻量级实现方案, 以保证运行共享方法的低代价.

方法采用了模块化的架构, 是将参与数据共享的各个节点从功能上分为多个不同的单元, 每个单元是一个模块. 各节点对等的运行本地数据 Web 服务化, 服务发布和服务发现三个基本模块, 来实现端对端的共享, 系统架构设计如图 1 所示.

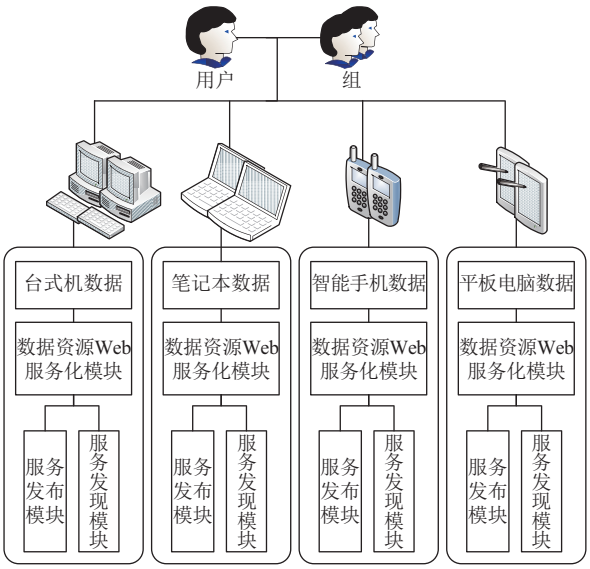

图 1 系统架构设计图

数据 Web 服务化模块, 实现设备或应用中的本地数据到 Web 资源的转化, 使本地数据资源可以通过网络上的 URL, 实质上是一个 RESTful API(Application Programming Interface, 应用程序编程接口)来访问. RESTful API 是符合 REST 设计风格的 Web API, 其核心在于 URL 仅仅是用来标识资源的 URI(Uniform Resource Identifier, 统一资源标识符), 而由 HTTP 协议支持的 POST, DELETE, PUT 和 GET 操作来分别表示对资源的增删改查, 执行本模块得到的数据资源地址是符合 REST 风格的.

服务发布模块, 负责将数据地址的 URL 发布到网络中, 使协同工作的设备可以定位并使用数据资源. 本模块的技术实现上需要根据具体的网络环境对技术方案加以选择. 举例来说, 针对没有网络覆盖的环境, 可以选择 BLE, WiFi-Direct 等技术, 而对于个人多设备, 家庭网络, 办公网络中的多设备数据资源地址的发布可以使用简单服务发现协议 SSDP, 下一章中的实现部分采用的就是 SSDP.

服务发现模块, 负责搜索定位已发布的数据资源地址 URL, 和服务发布模块进行配合, 采用和服务发布模块相同的技术, 实时获取到当前环境下所有共享数据的最新版本.

图 2 描述了系统各个模块间的调用关系. 用户打开一个设备中的实现本方法的系统应用, 默认开启服务发现模块, 可以得到一个当前共享环境下各设备所有在线的数据资源 URL 列表, 其中的每一个 URL 都是一个数据资源的 RESTful API, 点击 URL 可以实现在线查看或下载数据资源. 用户发布本设备的数据资源 URL 到共享环境中, 首先需要通过一个认证模块, 该模块主要是模拟验证用户对设备的操作权限, 类似于 Windows 下的 Administrator 验证或是 Linux 下的 ROOT 用户验证, 通过验证的用户表明有发布本机数据服务的权限, 可以选择开启本机的数据 Web 服务化模块, 实质上是开启 HTTP 服务来启动文件服务器的功能, 此时设备中的本地数据地址转化为 Web 中的 URL. 而其他设备要想获取到数据地址的 URL, 需要本机开启服务发布模块将要发布的地址多播到环境中, 这样一来其他设备的服务发现模块就可以定位并访问本机共享的数据资源了.

本节阐述了个人数据资源共享方法的架构设计以及各模块间的关系, 下一章中将详细说明各模块的实现以及在不同平台下系统运行效果.

3 系统实现

3.1 模块实例化

本部分阐述数据 Web 服务化模块和服务发布及发现模块分别在移动端和桌面端的实例化方法.

3.1.1 数据 Web 服务化模块

数据 Web 服务化模块是通过针对不同平台的特点开发 HTTP 文件服务器来实例化的.

移动端的数据 Web 服务化模块使用了 NanoHttpd[15], 主要是因为移动端资源稀缺, 而 NanoHttpd 是适用于移动端的轻量级服务器, 它的 Github 官方描述中指出它是一个极小型的 Java 编写的 web 服务器, 一个千行左右的 Java 文件就可以创建出一个 HTTP 服务器, 在实现上采用的是 Java 的 TCP 网络编程以及多线程技术. 首先创建 ServerSocket 和一个主线程, 运行

ServerSocker.accept()进行请求监听, 每当接收到客户端发来的请求, 就新建一个线程对请求进行处理. 请求的处理过程按照 HTTP Request 的请求头和主体的规范进行解析, 将读取的数据流获得的信息存储到相应的数据结构, 例如因为 HTTP 请求头部最长为 8192 个字节, 所以读取 socket 数据流的前 8192 个字节来提取请求头部的字段. 对请求进行响应的过程也是依照 HTTP 协议以及资源的自身类型特征有针对性地返回响应内容, 支持 MIME 格式的各类文件, 对于请求 URL 是文件夹的情况, 实现上会遍历文件夹中的各个文件并返回资源列表供使用者进一步查看和选取. NanoHttpd 解决了在移动端运行文件服务器资源占用率过大的问题.

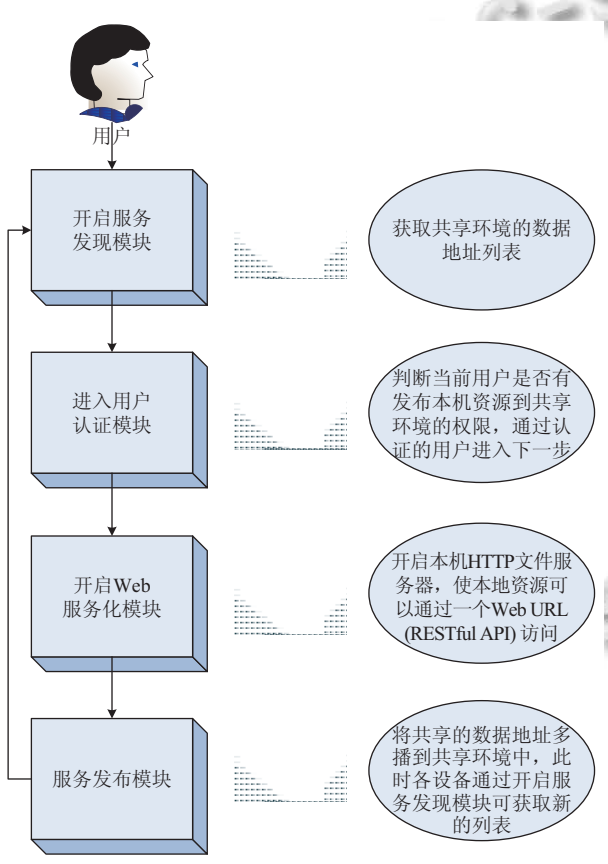

图 2 系统各个模块间的调用关系图

桌面端的数据 Web 服务化模块的实现上, 采用 Node.js 来创建 HTTP 文件服务器. Node.js 可以不依赖 Apache 或 Nginx 等服务器, 其本身内置的服务器功能就有着公认的良好性能, 而且它具有跨平台性, 编写的应用适用于 Windows, Linux 等多种桌面系统. 使用 Node.js 搭建文件服务器, 既可以利用 http, fs 等原生模块进行创建, 也可以使用 Express 开发框架的 express.static 中间件, 本文中为了更灵活且高性能的实现数据 Web 服务化模块的功能, 采用原生模块从头搭建文件服务器, 参考文献[16]中的实现细节, 加入了 MIME 类型支持, 缓存机制, 文件压缩以及断点续传支持等提高文件服务器性能的因素, 同时添加解码机制支持中文 URL. MIME 类型支持是通过一张后缀和格式的映射表进行支持的, 针对文件不同的后缀, 返回不同的 Content-Type; 缓存机制是对于某些类型的文件, 返回的响应头里添加 Expires 头和 Cache-Control: max-age 头, 设置一个超期时间, 浏览器对期限内的文件将不向服务器端发送新的请求, 而是直接返回浏览器缓存的副本; 与此同时, 添加对请求头中 If-Modified-Since 字段的解析支持, 如果服务器端的文件在这个时间后没有发生修改, 则直接返回 304 未修改状态码, 只有修改后的才发送文件, 减少流量的消耗; 同样可减少流量消耗的还有文件压缩, 调用的是 Node.js 的 zlib 模块, 实现对于大文件采用 GZip 压缩; 断点续传支持主要通过 HTTP 的 Range 支持, Server 通过声明支持 Range, Client 通过在请求头中加入 Range 字段来发起请求, Server 接收到 Client 的请求后只返回请求区间的内容, 支持断点续传; 最后添加了中文资源 URL 的解码处理, 支持数据资源的中文地址.

3.1.2 服务发布和服务发现模块

服务发布和服务发现模块针对的是个人多设备和协同工作环境的多设备间的数据资源地址的发布和发现, 采用了简单服务发现协议 SSDP 来实例化.

移动端的服务发布和服务发现模块抽取了 Google Physical Web 项目的 Android 端开源实现里的基于 SSDP 协议的搜索和发布的部分代码, 和 NanoHttpd 实现的文件服务器加以整合, 既可以从移动端的文件列表里选择文件进行发布, 也可搜索到其他设备发布的文件地址的 URL.

桌面端的服务发布和服务发现模块采用了 node.js 模块 peer-ssdp, 使用 npm install peer-ssdp 安装该模块, 调用模块中的基础函数可方便高效的实现桌面端数据地址 URL 的搜索和发布.

3.2 系统运行效果

针对智能手机, 笔记本电脑, 办公台式机运行本系统, 实现典型办公设备间的数据共享.

移动端手机型号为 Nexus 6, Android 版本为 5.1,

Android SDK Version 24, 移动端搜索获取共享数据资源地址列表如图 3 所示. 台式机, 操作系统为 Ubuntu 14.04 LTS, Memory 12GB, Processor Intel Core i7-2600@3.40GHz*8, OS type 64-bit, Node.js 版本 4.5.0. 笔记本电脑, 操作系统为 Windows 10, Memory 4GB, Processor Intel(R) Core(TM) i3-3217U CPU@1.80GHz, OS type 64-bit, Node.js 版本 4.5.0. 桌面端 BS 架构使用 Express 框架进行搭建, 桌面端搜索获取共享数据资源地址列表如图 4 所示.

图 3 移动端搜索获取共享数据资源地址列表

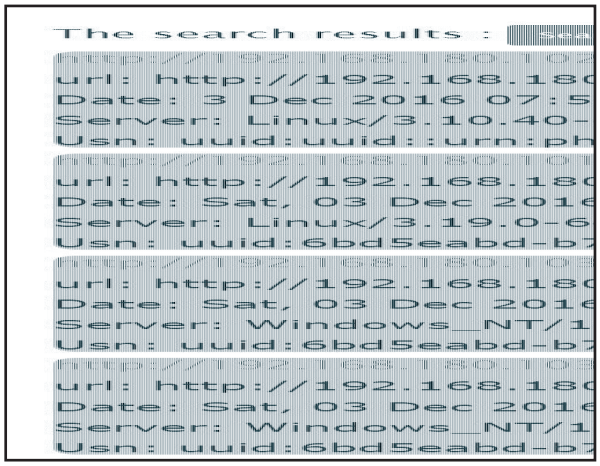

图 4 桌面端搜索获取共享数据资源地址列表

数据资源地址列表中的资源分别来自 Android 移动端, Windows 笔记本和 Ubuntu 台式机, 点击列表中的 URL 可以实现跨设备访问数据资源.

4 方法评估

4.1 功能评估

本文的方法适用于局域网环境下个人多设备和协同工作组内多设备间的数据共享, 以应对共享数据动态变化, 信息一致性维护, 节点异构, 代价可接受等多设备间高效数据共享面临的挑战.

对于共享数据动态变化, 方法对资源的共享和定位是使用数据资源地址而非传输数据本身. 开启应用服务发现模块可以实时获取数据资源地址列表, 每个列表项唯一标识了一个共享环境下的数据资源, 包括数据资源的地址以及描述信息, 通过点击数据资源的地址可以在线查看或下载频繁改动的数据的最新版本.

对于信息一致性维护, 各设备开启服务发现模块获得的数据资源地址列表是一致的, 每个设备通过服务发布模块可以共享新的数据资源以及删除已发布的数据资源, 而资源地址的添加和删除会实时的被其他设备的服务发现模块捕获, 保证各设备同一时刻获得的共享数据资源地址列表是无差别的.

对于节点异构, 本文的方法具有通用性, 智能手机, 平板电脑, 笔记本电脑和台式机都可采用不同的技术方案完成方法设计中的模块化架构, 稳定运行系统应用.

对于代价可接受, 针对移动端等资源相对稀缺的平台, 采用 NanoHttpd 等资源占用率小的技术方案来实现方案中的模块, 使运行系统给设备造成的代价维持低水平, 这在下一小节的性能评估里可以更清楚的反映.

最后, 方法具有可扩展的优势, 整个系统运行的核心是实现, 发布, 发现一套跨设备访问数据资源的 API. 其具有支持开发的特征, 基于这套 API 实现一个面向多设备的文档查看器, 图片浏览器, 音乐和视频播放器将会是一件更轻松有趣的事情.

4.2 性能评估

实现上述功能的代价是运行本地系统应用, 需要考虑在移动端资源相对稀缺的条件下, 系统应用的运行不能占用移动端过多的资源. 采用 Android 性能测试工具 Emmagee 对移动端应用在一次完整的使用过程占用内存比和占用 CPU 率进行了测量, 图 5 是应用在使用过程当中 10 次采样的占用内存比和占用 CPU 率, 曲线的高点发生在每次运行服务发现模块实时搜索数据资源地址列表, 可见运行应用给移动端带来的代价很小, 占用率基本在 5% 以下.

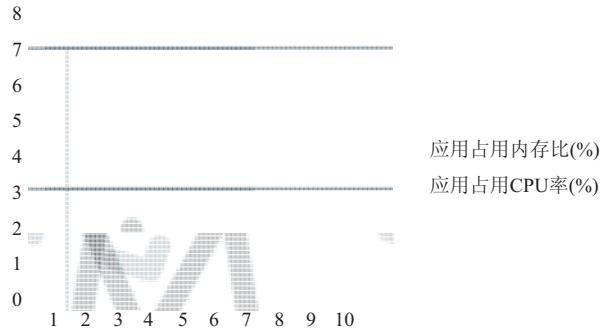

图 5 移动端应用运行过程中占用内存比和 CPU 率

同时, 将本应用与局域网内具有代表性的数据传输应用飞鸽传书进行对比, 二者运行期间占用内存比基本相当, 本应用的 CPU 占用率相对更小, 如图 6 所示.

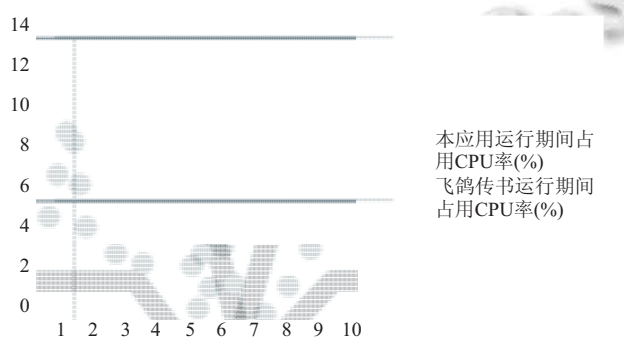

图 6 方法与其他应用的 CPU 占用率对比

5 总结与展望

本文运用服务化技术实现个人数据资源共享方法, 使局域网环境下参与共享的个人设备对等的运行数据 Web 服务化, 服务发布和服务发现三个基本模块, 方便地实现多设备间数据资源共享. 相较常见的数据共享方法具有无需搭建额外的硬件环境, 共享效率高, 支持无需下载的在线访问, 避免了三方数据泄露风险以及占用系统资源少等优势. 同时实现了多个平台的系统应用, 证明了方法有效的解决了多设备间高效数据共享面临的数据动态变化, 信息一致性维护, 节点异构, 代价可接受等挑战.

基于本方法模块化的设计, 我们将会在之后工作中针对不同平台的特点继续优化三个基本模块的实现, 通过提高每个模块的运行效率来提高整个系统的性能. 同时, 鉴于方法可以获得一套跨设备访问数据资源的 API, 具有支持开发的特征, 之后的工作会基于此实现跨设备的文档查看器, 图片浏览器, 音乐和视频播放器等有趣的应用. 最后, 本文的方案不仅限于数据资源的共享, 事实上当共享的不再是数据资源的地址而是操作软件, 硬件的 API 时, 相同的架构可以轻松的实现远程操控等功能, 这些都将是我们下一步的研究内容.

参考文献

1 Armbrust M, Fox A, Griffith R, *et al*. A view of cloud computing. Communications of the ACM, 2010, 53(4): 50–58. [doi: 10.1145/1721654]
2 杨红星. 云平台跨域分布式共享文件系统的设计与实现[硕士学位论文]. 杭州: 浙江大学, 2015. 21–36.
3 王中华, 韩臻, 刘吉强. 云环境下多用户文件共享方案. 计算机研究与发展, 2014, 51(12): 2614–2622. [doi: 10.7544/issn1000-1239.2014.20131178]
4 薛矛, 薛巍, 舒继武, 等. 一种云存储环境下的安全存储系统. 计算机学报, 2015, 38(5): 987–998.
5 Verizon. Verizon's 2017 data breach investigations report. http://www.verizonenterprise.com/verizon-insights-lab/dbir/2016/.
6 Meshkova E, Riihijärvi J, Petrova M, *et al*. A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. Computer Networks, 2008, 52(11): 2097–2128. [doi: 10.1016/j.comnet.2008.03.006]
7 Physical Web. The Physical Web is an open approach to enable quick and seamless interactions with physical objects and locations. http://physical-web.org/.
8 Arunachalam K, Ganapathy G. Research on UPnP protocol stack for applications on a home network. International Journal of Engineering and Technology, 2016, 8(4): 1728–1736. [doi: 10.21817/ijet/2016/v8i4/160804413]
9 Fielding RT. Architectural styles and the design of network-based software architectures[Ph. D. thesis]. Irvine: University of California, Irvine, 2000.
10 Wagh K, Thool R. Comparative study of SOAP Vs REST web services provisioning techniques for mobile host. Journal of Information Engineering and Applications, 2012, 2(5): 12–16.
11 Gao L, Zhang CH, Sun L. RESTful web of things API in sharing sensor data. Proc. of 2011 International Conference on Internet Technology and Applications. Wuhan, China. 2011. 1–4.
12 王孝满, 周晓明, 毛宇光. 数据资源服务化技术的研究与实现. 计算机技术与发展, 2011, 21(3): 79–82, 86.
13 Newmarch J. A RESTful approach: Clean UPnP without SOAP. Proc. of the 2nd IEEE Consumer Communications and Networking Conference. Las Vegas, NV, USA. 2005. 134–138.
14 方亚芬, 梁冠宇, 贺也平. 面向个人自组织网络的硬件资源管理. 计算机系统应用, 2016, 25(12): 1–8. [doi: 10.15888/j.cnki.csa.005472]
15 NanoHttpd. https://github.com/NanoHttpd/nanohttpd.
16 朴灵. 深入浅出 Node.js. 北京: 人民邮电出版社, 2013: 150–176.