



论自研智能终端操作系统^{*}

魏颖琪, 梁柏青

(中国电信股份有限公司广东研究院 广州 510630)

摘 要: 智能终端操作系统的持有厂商通过平台掌控移动生态系统, 要在生态系统上获得话语权, 必须参与到操作系统的研发中。分析了自研智能终端操作系统的技术切入点、研发策略以及研发方向。

关键词: 操作系统研发; 深度定制; 智能终端

doi: 10.3969/j.issn.1000-0801.2013.11.025

Research on Smartphone Operation System Development

Wei Yingqi, Liang Boqing

(Guangdong Research Institute of China Telecom Co., Ltd., Guangzhou 510630, China)

Abstract: The smartphone platform owners control mobile ecosystem through platforms they developed. To get influence on ecosystem, the users must be involved in the development of the operating system. The start point of smartphone platform development, development strategy and development direction were analyzed.

Key words: development of operation system, deeply customization, smart device

1 引言

工业和信息化部电信研究院发布的《移动互联网白皮书(2013年)》中首次提及自研操作系统,代表了国家的某种态度。针对手机操作系统的研发有不同的声音,有基于Android定制的MIUI(米柚)操作系统,突出用户体验,强调为中国用户设计;有基于Android定制的百度云系统,集成百度业务,增强云端互联网服务;还有将自研操作系统作为备份的华为公司。对于移动智能终端自研操作系统的研发,需要在系统哪些层面上展开工作,其深度和广度又如何,研发重点在哪里,发展方向是什么?本文将就这些问题一一展开论述。

2 现代移动智能终端操作系统的特点

要探究自研操作系统的切入点,首先需要了解现在移

动智能终端操作系统的特点、趋势和方向。现代智能手机属于应用平台,应用平台具有以下三大要素:

- 除用户外,开发者/开发商是平台另一个重要的目标客户;
- 为开发者提供API/SDK以及开发环境;
- 提供应用的发布及货币化渠道,建立健康的应用生态系统。

移动智能终端操作系统的垂直化整合日趋明显,移动应用同时具备全球化和本土化的特点,自研操作系统应从本土化接入,从本土化中寻找应用生态系统的立足点。

Web化演进是移动智能终端操作系统的重要发展方向,跨平台技术能够降低原生系统的垄断地位,Web是自研操作系统的新机遇,结合云计算能力,创建自己的生存空间。

^{*} 国家“核心电子器件、高端通用芯片及基础软件产品”科技重大专项基金资助项目“移动智能终端操作系统研发”(No. 2012ZX01039002-001)

3 自研操作系统的模式

手机市场已由运营商和手机制造商主导转为平台主导,以创新应用开发为驱动力。手机系统平台是移动生态系统的基础,通过应用开发环境和应用商店繁荣整个生态系统。要在生态系统中获得话语权,必须针对应用平台的三大要素积极参与。

自研操作系统有两大方向:一是完全自研模式,即基于 Linux 内核,从内核开始开发操作系统;二是分支模式,即在现有操作系统上作为分支发展,未来系统成熟时,再考虑分离成独立的操作系统。

3.1 完全自研模式

完全自研模式是以 Linux 内核为基础,自主设计系统架构,搭建从内核到系统应用的整个平台,从零开始搭建,难度大、投资大、风险大。新操作系统将与现有智能手机系统在市场上竞争,现有系统在整个产业链上占据先机 and 优势,对后来者形成障碍。新的操作系统不仅要吸引用户使用,同时也要说服用户放弃现有平台,即放弃 iOS 和 Android,转到新平台上。

3.2 分支模式

分支模式是以现有智能手机操作系统为基础,发展分支系统。选择基础操作系统,要求该系统已经具备完备且繁荣的生态系统,可以站在巨人的肩膀上发展;同时要求该操作系统必须开源开放,有合适的版权许可。

iOS 是闭源系统,其 SDK 对外开放,用户只能安装来自官方应用商店的应用。开发者无法对 UI 框架、中间件、开发环境、系统内核进行定制,即不能进行系统级别的研究;Windows Phone 也属于封闭操作系统。纵观主流智能手机操作系统,能形成有效的网络效益、具有繁荣生态系统、提供开源代码的,唯有 Android。

Google 提供了 Android 开源项目(AOSP),同时还维护一个 Android 的私有分支,该分支比释放在 AOSP 的公开代码提早 6 个月,只提供给特定的合作伙伴,确保他们在市场上领先。AOSP 是由 Google 完全掌控的开源项目。“开源”只是冰山看得见的顶部。Google 使用封闭的管控模式控制产品发展方向,Android 是专制的开源社区。Google 几乎不接纳非 Google 员工的提交。由于 Google 的严格管控,并同时维护公开和私有两个版本,参与 AOSP 社区代码贡献并不现实。因此作为 Android 分支的自研操作系统不参与 AOSP 社区贡献,但为了自我检查、自我验证和自我发

展,可在国内建设相关的开源社区,开放给开发者,提供 beta 测试服务,吸引开发者和极客对系统进行验证,并移植到更多的硬件机型上。

基于 Android 的自研操作系统将针对 Android 某版本进行研发。然而,Android 版本升级速度飞快,每个新版本不仅对原有版本进行改进和优化,同时引入新的创新,增加新的功能。为保持市场竞争力,吸纳 Android 最新的创新,兼容新版本应用,充分利用 Android 生态系统,作为分支发展的自研操作系统应采取版本回归机制。事实上,Android 对它的上游 Linux 内核也同样采取版本回归机制,新的 Android 版本不是在原有的 Android 内核上进行发展,而是在新的 Linux 内核版本上进行开发,使得新版本 Android 内核可以吸纳 Linux 内核的最新成果。

需要注意,自研系统与上游 Android 版本相比,不可避免地出现滞后的现象。

4 作为 Android 分支的自研操作系统

根据 Android 架构,如图 1 所示,从下至上分析自研操作系统的参与点。



图 1 Android 系统架构

4.1 Android 内核层

Android 平台架构的最底层是基于 Linux 内核的 Android 内核,实现设备驱动、资源访问、电源管理并进行其他的 OS 工作。

4.1.1 Android 内核和 Linux 内核

Android 内核并不完全等同于 Linux 内核,Android 操作系统的内核是 Linux 内核的一个分支,具备典型的 Linux 调度功能,此外,Google 为了能让 Linux 在移动设备上良好地运行,对其进行了修改和扩充,如图 2 所示。Android 去除了 Linux 中的本地 X Window System,也不支持标准的 GNU 库,这使得 Linux 平台上的应用程序移植到 Android 平台上变得困难。Google 还在 Android 内核中加入了自己开发的名为“wakelocks”的移动设备电源管理功能,该功能用于管理移动设备的电池性能,但该功能并没有被加入 Linux 内核的主线开放和维护中。

Android 内核的最大特点是增加了硬件抽象层

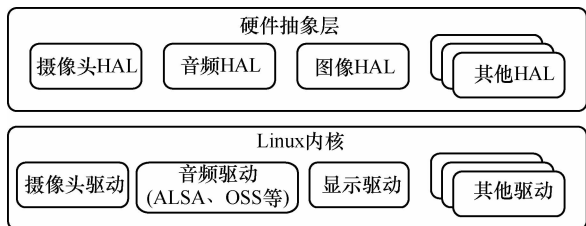


图2 Android架构最底层:Android内核层

(HAL),以提供广泛硬件适配和快速商业化能力。Android的HAL提供从内核中抽离的硬件功能,以*.so档的形式存在,把Android框架与内核隔开,通过HAL适配底层的不同硬件,使得上层软件开发得以独立进行,增强系统的可移植性。

HAL可保护硬件提供商知识产权,避开Linux的GPL束缚。思路是把控制硬件的动作都放到了AndroidHAL中,而Linux驱动仅完成一些简单的数据交互工作,甚至把硬件寄存器空间直接映射到用户空间,使驱动从系统空间的内核开发转为用户空间开发,绕过Linux内核的GPL许可限制,而在用户空间的程序只需遵循Android的Apache 2.0许可。这使得厂商可以将产品快速推向市场,不会因为Linux社区的流程而错过市场档期。

通过HAL,确保Android平台在不同硬件上加载,运行在Android平台上的应用可在不同硬件配置的不同手机上使用,并可在平板、手机、车载、机顶盒等多种形态设备上运行。

某种程度上,Android的HAL以及芯片turnkey模式,促使手机制造业向PC制造方向发展。由于Android系统差异化空间被Google不断挤压,差异化主要集中在UI和应用本土化上,手机制造商不得不以价格和推出市场时间作为主要竞争点。

4.1.2 自研系统不需要修改内核

如果对内核进行修改,将带来如下问题:

- 版权限制,Linux内核及基于其开发的Android内核是GPL,在上面修改必须公开代码。
- 硬件适配问题,OEM厂商需要针对系统进行专门适配,在初期不具备生产规模的情况下生产成本较高。
- 若Linux内核或Android内核发生较大修改,可能需要大规模的重开发。

Firefox OS是Web操作系统,是利用HAL实现平台在设备上快速移植的成功案例。它的底层核心称为Gonk,包

括Linux核心和HAL,而HAL则来自Android的AOSP。与以往OEM适配第一款平台手机的艰辛相比,利用AndroidHAL,很快就将Firefox OS移植到现有Android设备上,在2012年移动世界大会中,Mozilla在三星Galaxy II高端智能手机硬件上演示了新系统。

一致性是应用平台成功的关键,应用可以运行在所有承载该平台的设备(不同厂商、不同型号)上,因此硬件的广泛适配非常重要,没有需求也没有必要对Android内核进行修改。

4.2 Android基础库与运行环境

Android次底层是基础库和Android应用运行环境。

4.2.1 基础库和开源项目

基础库提供了Android平台的基础计算能力,是C/C++库。如基础的C语言函数库(libc)、基础图像处理库(将2D/3D图像显示到屏幕的surface manager和OpenGL ES)、基础字体渲染库(freetype),这些都来源于独立的大型开源项目。Android平台对开源项目进行筛选,大部分直接使用,小部分进行修改,如Android采用OpenCore作为基础多媒体框架,直接采用skia作为核心图形引擎(2005年被Google收购);而Bionic是Android改良的libc版本,大小只有原来的一半。

4.2.2 Web引擎和WebKit组件

基础库归属各大开源项目,通常对于各基础字体、图像、多媒体、数据库没有修改需求。最大可能进行修改的是Web/Widget引擎。Web化演进是目前移动智能终端操作系统发展的趋势,可减少原生操作系统的羁縻,直接开发和部署应用,将以HTML5技术为基础发展Web平台,作为现有移动生态系统的补充。Web平台研发涉及本层的WebKit组件,有如下3种研发模式,如图3所示,具体介绍如下:

- 在本层增加新的Web引擎;
- 在本层WebKit模块中进行修改;
- 基于WebKit在上面的应用框架层中开发。

通用的能力库开发策略适用于对如下3种方式进行

应用	Widget/HTML5应用
应用框架	Widget/HTML5引擎
库和运行环境	WebKit (Android自带)
Android内核	

图3 在应用框架层基于WebKit进行研发的模式

选择。

- 要完全重新实现某个基础能力,首先要回答的问题是为何要从头开始,重新开发所涉及的技术基础和成本是否能够满足?
- 在某开源项目中进行修改,必须符合该开源项目的版权要求,同时应争取将修订合并到上游项目中,否则必须对新开源版本进行重新适配,存在因开源版本改动过大或架构调整而导致原有修改需大量重写的风险。该风险由不同团队开发内容互相渗透、缺乏严格模块化边界所致。
- 在某开源项目的基础上进行开发,即由应用框架层提供。当开源项目升级时,将获得该开源项目升级带来的所有新能力。

综合分析,应采用第 3 种方式,确保模块化边界清晰,不在本层 WebKit 模块进行修改。

4.3 应用框架层和应用层

互联网厂商和运营商参与移动操作系统开发,目的是借助移动终端推广自己的核心业务,将自有业务下沉到平台中,作为系统级别的基础应用/服务和基础能力提供给用户和开发者,充分利用新硬件带来的新信息输入/输出内容和方式,而将硬件创新留给硬件厂商和 OEM;充分利用各大开源社区提供的大型软件库能力,将库能力的开发留给开源社区,将选择开源项目留给 Google。因此,互联网厂商和运营商基于 Android 自研操作系统的研发主要集中在这两层上。

4.3.1 业务提供方式

应用平台研发要面向开发者提供业务能力,而不仅是单个应用的提供。基础应用/服务是比较特殊的应用,它们集成到系统中,是系统自带的应用/服务。除了向用户提供应用/服务外,其特点是允许第三方应用调用或激活。例如,电话呼叫是典型的基础应用,第三方应用可以调用电话呼叫,并自动填入被叫号码。基础应用/服务和基础能力的调用方式如图 4 所示,具体描述如下:

- 在应用框架层以 API 的方式,即基础能力的方式,提供给第三方;
- 在应用层,作为应用、服务或者提供者,通过激活/调用等进程间的通信方式提供给第三方,即以基础应用/服务的方式提供。

具体采用哪种方式要根据所提供业务的特点,一般原则是根据用户使用流程(即操作流程)是否具有独立性进

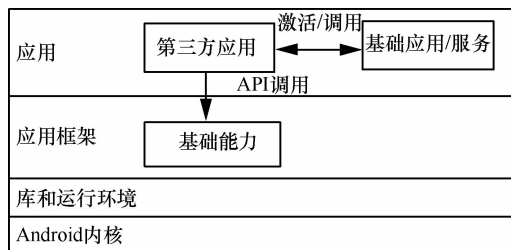


图 4 基础应用/服务和基础能力的调用方式

行判断。

例 1 运营商提供的多方通话功能。用户使用多方通话的操作流程是多样且非唯一的,如用户可以在发起前明确所有与会方,然后同时向多方用户发起呼叫请求,也可以逐个依次将各方接入。所选择的与会方可以来自手机联系人列表,也可以来自第三方企业通讯录。由于多方通话功能可嵌入不同的应用场景,有多种可能的操作流程,因此适合作为基础能力,以 API 的方式提供给第三方开发者。

例 2 某支付功能,如网银支付、支付宝或运营商小额支付。用户具体使用某种支付的操作流程是一致的,包括确定被支付方、确定支付金额以及必要的安全校验等,这些流程不因具体支付场景的不同而发生变化,适合以基础应用/服务的方式提供给第三方调用。

如两者皆可,作为一般原则,尽可能放在上层;为保持业务的完整性,尽可能放在应用层,而非应用框架层,这样做的好处如下。

- 易于维护和管理。以更新升级为例,如果放在应用层,可通过简单的应用升级方式进行;如果放在应用框架层,则需集成在操作系统中,在下一操作系统版本升级中进行。
- 安全性更高。业务对外暴露的接口越少,受到恶意攻击的机会越低。

4.3.2 向本地驻留应用提供 API 的方式

作为基础能力开放(即以 API 方式)提供给开发者,用以开发驻留在设备上的应用,包括原生应用和 Web 应用。开放 API 的方式有如下 3 种,如图 5 所示。

- 本地原生 API 方式,由应用框架层中的基础能力提供给本地原生应用。
- 本地 Web API 方式,将基础能力封装为 Web 接口,以 HTML5 标准格式提供给本地 Web 应用。
- 云端 Web API 方式,通过云 API,将云能力开放给本地应用,这些应用可以是原生应用,也可以是 Web 应用。

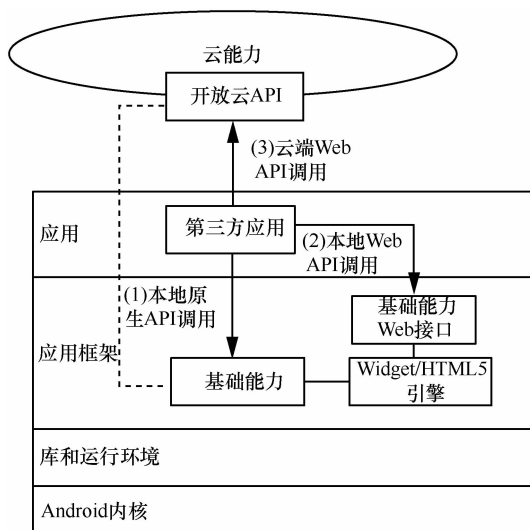


图5 向本地驻留应用开发 API 的 3 种方式

需要注意,云端 HTML5 接口方式可能会形成相互割裂的生态圈。不同互联网商或运营商会封装出不同的云能力 API,这些 API 只是在格式上遵循标准 HTML5,具体接口函数并不一致,如对于功能相同的语音 API,运营商 A 和运营商 B 就很有可能不同。

4.3.3 私有云接口的代理方式

有时,基于某种原因不能直接暴露云端 API,如要隐藏云和终端之间的用户注册登记流程、隐藏云服务的 IP 位置等,即云 API 是非公开的而是私有的。

通过基础能力或者基础应用/服务进行封装后,即以云能力代理的方式向第三方应用提供,如图 6 所示。

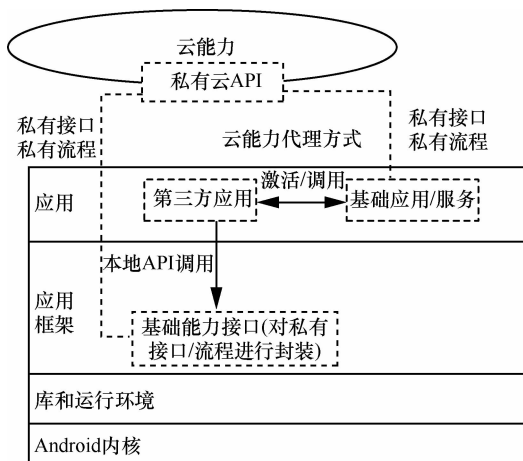


图6 私密云接口的代理方式

4.3.4 非本地驻留的应用方式

除了驻留本地的原生应用或 Web 应用,还有驻留云端的应用。云端应用以 Web 链接方式提供。对用户而言是软

件即服务 (SaaS) 平台,而对开发者而言是平台即服务 (PaaS) 平台,统称为云应用平台。云应用平台提供云应用的开发环境,利用 Web 跨平台能力可部署到不同操作系统的终端上。云应用平台除了提供应用开发/调测环境,还要提供应用发布/运营/支撑环境,由此满足应用平台的三大要素,构筑完整的云应用生态系统,如图 7 所示。

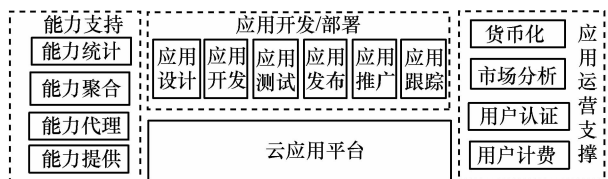


图7 云应用平台框架

云应用平台包括能力支持、应用开发/部署以及应用运营支撑 3 部分。

能力支持部分为云应用平台提供所需能力,包括能力提供、第三方能力代理、能力聚合以及能力统计等。能力统计有助于云应用平台了解各能力的使用情况,作为扩容或与第三方能力提供者分成的依据。

应用开发/部署部分用于帮助开发者开发应用,提供快速开发工具、应用组件模板、通信能力等,同时帮助开发者进行应用发布,并在发布后进行应用跟踪,如 bug 反馈、崩溃报告等。云应用开发平台面向不同类型的开发者,如个人开发者、企业开发者。有一类开发者,具备很少甚至零编程技巧,更多的是对内容的编辑和填充,对于这部分开发者,可考虑通过模板实现零代码或少量代码的开发方式。云应用平台可与应用商店合并,提供统一的应用发布渠道。

应用运营支撑部分帮助开发者在应用发布后进行应用支撑维护,实现货币化,包括用户认证、计费、市场分析和货币化等。计费包括多种方式,如点击下载付费、广告、免费使用增值付费、应用内付费等,提供多种货币化途径。

云应用平台涵盖了应用生态链条的各个环节,是独立且完整的生态系统,以 HTML5 的方式与终端操作系统解耦,是互联网厂商和运营商围绕自己的核心业务参与应用创建、建立生态系统的重要方向,但目前仍处于发展的早期阶段。

4.3.5 Home UI 的创新

UI 创新是终端用户界面和用户体验差异化的重要手段,不仅是自研操作系统,也是各大手机制造商品牌体现

的关键,如 HTC 的 Sense UI、华为的 Emotion UI,而小米的 MIUI 更强调适合国人的使用习惯。

三星有专门负责软件方面用户体验的部门,完全从用户使用角度考虑,挖掘用户需求,从细节入手,持续改善 UI 设计,开发新的交互方式,最终只有一个目标——提高用户使用三星手机的便利性^[3]。而三星的 CEO 权五铨在就任讲话中提到,他们将会“特别专注于提供新的用户使用经验”,将会通过加强/改善软件、UI、设计和解决方案这几个方向达成。

Android 市场占有率越来越高,而 Google 实施越来越严格的控制,在手机同质化现象日趋严重的背景下,通过独特的设计、便捷的操作,从使用习惯上绑定用户,从用户直接体验方面建立品牌效应。

5 从分支走向独立

分支发展的自研操作系统利用上游操作系统 Android 的生态环境,确保新平台上市时有足够应用可供用户选择。为了降低上游操作系统的影响力,大力发展以 HTML5 为代表的 Web 应用生态系统,充分利用云端能力。平台实际承接了两个应用生态系统,分别是原生应用生态系统和 Web 应用生态系统。当 Web 应用生态系统不断发展并趋向成熟,不再仅作为原生应用生态系统的补充而是可替代时,则可以考虑将原生部分从自研操作系统中去除,成为独立、完整的自研操作系统。两个应用生态系统的对比如图 8 所示。

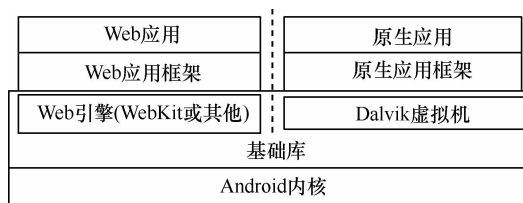
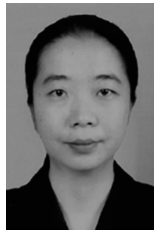


图 8 两个应用生态系统对比

6 结束语

自研操作系统应首先立足于应用平台这一智能终端操作系统的关键特征,将自有业务和能力部署到平台上,为广大的开发者所使用。同时,Web 化演进是目前操作系统发展的重要趋势,降低底层操作系统对生态系统的掌控,直接控制应用,建立自有生态系统。

[作者简介]



魏颖琪,女,中国电信股份有限公司广东研究院高级工程师、一级产品业务架构师,主要从事移动操作系统、移动互联网应用研发工作。

梁柏青,男,中国电信股份有限公司广东研究院高级工程师、政企客户开发部技术总监,主要从事移动终端、云计算、移动互联网技术研发工作。

(收稿日期:2013-08-25)

(上接第 135 页)

- 趋势分析。针对系统获取到的大量 J2EE 应用服务中间件运行数据统计分析,进而实现中间件运行状态趋势预测以及对策预防等高级功能。
- 机器学习智能故障处理。应用机器学习技术,对故障处理操作建模,分析每次故障处理控制信息,操作目标参数回归,最终实现自动故障处理。

参考文献

- 高张,康小军. 提高 Tomcat 服务器运行性能的研究. 计算机与数字工程,2008(10):153~154
- 孙仁鹏,何森. Tomcat 性能监视和调优研究. 软件导论,2012(12):56~59
- 林泊,周明辉,刘天成等. 一个 J2EE 应用服务器的 Web 容器集成框架. 软件学报,2006(5):26~28
- 田珂,谢世波,方马. J2EE 数据持久层的解决方案. 计算机工

程,2003(22):102~103

- 5 窦蕾,袁臻,刘冬梅. 基于构件的中间件技术 J2EE. 计算机科学,2004(6):98~102

[作者简介]



吴一鸣,男,南宁供电局信息中心助理工程师,从事电力信息系统运维工作。

田永涛,男,南宁供电局信息中心工程师,从事电力信息系统开发维护工作。

(收稿日期:2013-10-01)