面向移动应用的后端服务平台①

高嘉泽[1,2,3], 高 强[1,2,3], 吴国全[2], 魏 峻[2,3], 黄 涛[2,3]

[1](中国科学院大学 北京 100049)
[2](中国科学院软件研究所 软件工程技术研究开发中心, 北京 100190)
[3](中国科学院软件研究所 计算机科学国家重点实验室, 北京 100190)

摘 要: 典型的移动应用开发分为服务器端和客户端两部分, 传统的移动应用开发模型需要花费大量人力、物力和时间成本用于开发和维护服务器端功能. 本文提出了一种面向移动应用的后端服务平台, 该平台为移动应用开发提供自定义数据服务、用户管理服务、文件存储服务、地理信息查询服务、消息推送服务等适用于典型移动应用场景的通用服务. 移动开发者可以通过平台提供的 REST 风格 Open API 或者为不同移动终端提供的 native SDK 两种方式利用后端服务平台进行移动应用的开发. 最后, 本文以实际案例说明了面向移动应用的后端服务平台的使用和为移动应用开发带来的优化和改进.

关键词: 移动应用; 后端即服务; RESTful Web 服务; SDK

Backend Service Platform for Mobile Apps

GAO Jia-Ze[1,2,3], GAO Qiang[1,2,3], WU Guo-Quan[2], WEI Jun[2,3], HUANG Tao[2,3]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)
[3](State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Traditional development of mobile application contains server-side and client-side. Developers should spend a lot of resources and time on developing and maintaining the functionality of the server-side. To simplify mobile application development, we present a backend service platform for mobile app development, which offers customer data storage, user management, file upload and download, geographic location query, push notification and other services. Mobile app developers have access to these services using two different ways: RESTful web service or native SDK. Finally, we present a practical case to illustrate the convenient usage of our platform and the improvement this platform bring to mobile app development.

Key words: mobile app; backend as a service; RESTful web service; SDK

1 引言

随着智能手机、平板电脑等移动设备的普及, 人们越来越多的从移动终端访问互联网. 来自埃森哲的《进入移动互联网新时代》报告[1]显示, 到 2014 年, 移动互联网的使用率将首次超过桌面电脑的使用率. 根据移动分析公司 Flurry 的数据, 中国已经成为仅次于美国的全球第二大移动应用使用国[2].

典型的移动应用包括客户端和服务器端两部分. 移动应用的传统开发部署过程主要包括如下五个步骤:

① 应用服务器、数据库服务器等硬件基础设施的购买和准备;

② 服务器端运行环境的选择、安装、配置;

③ 使用 Java, PHP, Python,.NET 等高级编程语言开发服务器端的业务逻辑和服务;

④ 对服务器端暴露给移动端的接口进行部署、监控和管理;

⑤ 编写移动端和服务器端接口的交互代码以及开发移动端的用户界面.

① 基金项目:国家自然科学基金(61173005,61003029)
收稿时间:2013-07-09;收到修改稿时间:2013-08-22

通常一个移动应用的开发时间为 2-6 个月, 其中将有 40%-60%的时间花费在后端开发即服务器端的开发上, 而服务器端的硬件购买和维护成本也达到平均每个移动应用十万至一百万美元. 与此同时, 移动应用开发者普遍缺乏服务器端的开发技能和经验, 这些方面都导致移动应用的开发需要花费大量的人力、物力、时间成本.

针对上述问题, 本文提出并设计实现了一个面向移动应用的后端服务平台, 该后端服务平台提供了满足移动应用开发所需要的共性、通用的服务, 移动开发者可以通过 REST(Representational State Transfer,即表述性状态转移)风格的 Open API 和为不同移动终端平台订制的 native SDK(Software Development Kit,即软件开发工具包)直接访问平台提供的服务器端服务, 从而降低移动应用的开发成本, 使移动应用开发者可以专注于移动端的开发, 无需购置服务器等硬件设备, 无需开发维护服务器端代码.

本文的组织结构如下: 首先介绍移动应用后端服务领域的相关工作, 然后详细讨论面向移动应用的后端服务平台 MobiPlus 的架构设计和模块交互, 接着重点介绍该平台实现中的三点关键技术, 最后介绍一个利用该平台开发的移动应用的实际案例以及下一步工作.

2 相关工作

BaaS(Backend as a Service)即为移动应用开发者提供整合云后端的边界服务. 根据调研机构 IDC 发布移动领域的季度报告显示: 近 60%的合作开发者希望在应用中提供云服务. 移动应用开发者需要的云服务有推送通知、地理位置信息检索、图片文件分享、用户管理、通信、评星和评价等. 虽然每个 BaaS 系统的目的都是向移动开发者提供基本类似的功能套件, 但是也有很多提供商专注在移动开发行业的特定领域. 比如, iKnode 定位于.net 体系框架; CodeCloud 则是托管 node.js 和 SQL Lite 平台.

本文介绍的面向移动应用的后端服务平台 MobiPlus 为移动应用提供灵活高可用的移动开发技术堆栈, 以 REST API 的创建和管理为基础, 向移动客户端提供获取云资源的方式, 并提供便于移动开发者使用的 native SDK, 使开发者无需购买和准备应用服务器以及数据库服务器, 从编写复杂繁琐的服务器端业务逻辑代码中解脱出来, 同时无需对自己的服务器端功能进行运维, 从而节省了开发维护服务器端代码的人力和时间成本, 极大的提高了移动应用的开发效率, 缩短了移动应用的开发周期, 帮助开发者在短时间内高效的开发出优质的移动应用.

3 后端服务平台的设计与实现

面向移动应用的后端服务平台 MobiPlus 为移动应用开发提供了面向丰富场景的数据、通信和管理服务. 下面将从该平台的架构设计、模块交互和关键技术几个方面来具体介绍该平台的设计与实现.

3.1 架构设计

MobiPlus 平台的体系结构如图 1 所示, 它从平台架构视图和移动应用开发者视图两个角度说明了该平台的系统构成.

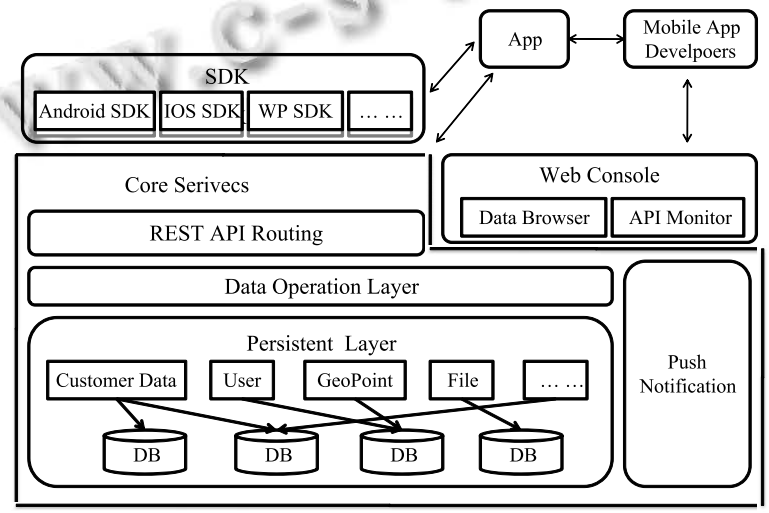

图 1 面向移动应用的后端服务平台体系结构图

从平台架构的角度来分析，面向移动应用的后端服务平台分成三大组件: native SDK 组件, web 管理后台组件(Web Console), 核心服务组件(Core Services). 其中, 核心服务组件是实现云端数据服务的主要组件, 对于云端数据, 平台主要实现如下服务:

① 自定义类型的数据(Customer Data). 开发者可根据移动应用的需求, 定义数据类型和数据结构, 平台为开发者提供了丰富的数据类型可供使用, 同时也提供了包括如表 1 所示的基本的条件查询、复合查询、关联查询、统计、集合操作等复杂的数据服.

表 1 自定义数据的操作举例

操作关键字	操作举例	操作类型
$lt	where={"score":{"$lt":1000}}	数字比较条件查询
$select	where={"hometown":{"$select":{"query":{"className":"Team","where":{"winPct":{"$gt":0.5}}},"key":"city"}}}	复合查询
$inQuery	where={"post":{"$inQuery":{"where":{"image":{"$exists":true}},"className":"Post"}}}	关联查询
AddUnique	{“fruit":{"__op":"AddUnique","objects":[“apple",“pear"]}}	集合操作

② 账号用户数据(User). 通过对移动应用使用场景的分类概括可以发现, 大部分移动应用都需要用户注册账号并且应用内容与账号紧耦合, 因此平台为开发者提供了一系列账号用户相关的服务, 包括注册、登录、登出、重设密码、与第三方社交网络账号授权连接等. 平台同时提供账号用户数据的安全访问机制, 账号用户相关操作均需要通过平台授权的有效令牌(session token)才可进行.

③ 文件图片(File). 针对移动应用中文件和图片的使用特点, 平台主要支持了面向小文件和图片上传服务, 同时为每个文件和图片生成可通过 HTTP 协议访问的 URL, 方便开发者获取云端资源.

④ 地理位置数据(GeoPoint). LBS(Location Based Service)类型的移动应用也在移动应用市场占据了大量份额, 平台为移动开发者实现了上传地理位置经纬度的服务, 此外, 还可以在上传的地理位置集合中进行地理位置的查询, 例如查询以指定点为圆心指定距离为半径的圆内覆盖的地理位置; 查询以指定点构成的矩形内覆盖的点等.

从移动应用开发者的视图来看, 开发者可以通过 native SDK 为每一个移动终端平台开发移动应用, 开发者也可以用直接访问 REST 风格 API 的方式完成想要实现的操作. 此外, 该平台为移动开发者提供了 web 形式的后台管理工具, 开发者可在该工具下可视化的查看移动应用的各种类型数据, 包括应用自定义类型的数据(Customer Data),账号用户数据(User), 地位位置数据(GeoPoint),文件图片数据(File)等, 同时可以对这些数据进行可视化编辑. 此外, 后台管理工具还提供了对移动应用使用情况的监控统计数据的可视化界面, 使移动应用开发者对移动应用的运维情况一目了然.

3.2 Web 模块交互

核心服务组件是平台向其他组件和外部提供服务的出口, 同时也是平台处理全部服务流程的框架. 核心服务组件内部模块实现如图 2 所示. 各个模块的功能和模块间协作方式如下:

① 权限模块(Auth Module)对从移动端发来的请求进行权限检查, 只有平台授权过的移动应用发来的请求才能通过该模块并把请求交给下一层模块处理.

② 路由解析和应答模块(REST Resolver and Responser)根据请求中 REST 风格的 URI、HTTP 方法的语义(POST,GET,PUT,DELETE)以及请求包含的参数, 根据 JAX-RS(JSR 311)标准[3]将该请求路由给下一层模块相应的处理方法.

③ 解析模块(Parse Module)将请求中包含的参数按照指定格式解析成下层模块可读的数据对象, 在解析过程中, 也会对参数进行语法合法性等检查, 从而确保来自移动端的请求是有效的.

④ 数据模块(Data Module)将根据解析模块的解析结果执行对云端数据的各种类型的操作, 如自定义数据的插入, 账号用户登录, 上传图片, 查询附近有哪些地理位置信息等.

⑤ 组装模块(Assemble Module)对数据模块的执行结果进行组装, 如插入删除更新操作是否成功、查询操作的查询结果等. 如果在之前某一模块发生错误或异常, 错误异常信息也将交由组装模块进行组装, 最后通过路由解析和应答模块返回到移动端.

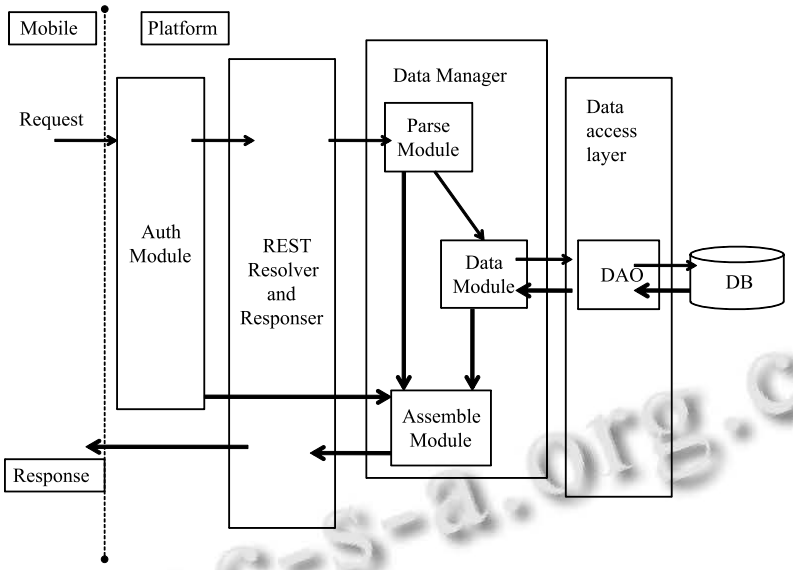

图 2 核心服务组件内部模块图

3.3 关键技术

3.3.1 可伸缩的数据交换协议

面向移动应用的后端服务平台采用 JSON 数据交换格式进行移动端和服务端的数据传输. 采用 JSON 格式进行数据传输, 有如下几方面的优势: 首先, JSON 格式是一种轻量级的数据交换格式, 不需要任何辅助类型定义信息, 键值对的方式即可完整描述数据自身的类型和内容, 这一点对于移动应用是非常重要的, 因为移动设备访问网络时常受限于网络带宽和网络延时等因素[4]; 其次, JSON 格式是一种跨语言的数据交换格式, 每一种编程语言都可以便捷的解析和生成 JSON 格式的数据.

此外, 面向移动应用的后端服务平台还需要提供一种通用的、可动态伸缩的数据交换、构造和存储方式, 以满足移动应用需求和业务的变更. 算法一描述了一种可以允许移动应用开发者无需预定义或更改数据结构和类型, 可以直接在传输数据过程中更新数据结构的服务器端解析和构造方式. 算法一通过递归的方式处理 JSON 形式数据对象的每一组键值对, 并可以自动识别平台支持的键值对中值的类型, 递归调用完成也就构造出了数据对象每一个字段类型和值的完整信息, 这些信息将由平台进行维护和更新, 从而使开发者并不需要关心数据对象的结构定义和更改, 仅需要根据业务需求和变更使用新数据即可.

算法一 服务器端的数据解析构造算法

输入: JSON 格式字符串

输出: 动态生成的数据对象及其类型

```
Object parseJSON (String jsonString)
{
    JsonElement je= JsonParser.parse(jsonString);
    RETURN buildObject(je);
}
Object buildObject(JsonElement je)
{
     IF (je.isJsonNull )
          RETURNnull;
     IF (je.isJsonArray)
          FOR each item in je
           array.add(buildObject (item));
           RETURN array;
     IF (je.isJsonObject())
          FOR each entry
           object.append(entry.key,buildObject (entry.value));
         RETURN object;
     IF (je.isJsonPrimitive())
          JsonPrimitive jp=je.getAsJsonPrimitive();
          IF (jp.isBoolean())
            RETURN jp.getAsBoolean();
          ELSE IF(jp.isNumber())
            RETURN jp.getAsDouble();
          ELSE IF (jp.isString())
             RETURN jp.getAsString();
}
```

3.3.2 面向资源的 REST API

面向移动应用的后端服务平台为移动开发者提供了一组 REST 风格的 API. REST 风格的 Web 服务是 Roy Fielding 于 2000 年在他的博士论文[5]中提出的. REST 风格的 web 服务是一种可以传输 XML 格式和 JSON 格式数据的无状态服务, 并且显式的使用了 HTTP 的方法语义, 即 POST GET PUT DELETE 四种方法,同时 REST 风格 web 服务的 URI 是一种面向资源

的目录结构.

面向移动应用的后端服务平台以 REST API 的创建和管理为基础, 向移动客户端提供获取云资源的方式, 这种方式可以为移动应用提供通用的、便于开发和使用的 web 服务, 这种面向资源的 API 风格可以直观的表述自身的语义, 同时使每种开发者自定义数据的操作 API 皆可自动生成. 以一个笔记应用为例, 我们将笔记对象命名为 note,如表 2 所示, 对 note 数据的增删改查可通过直观的 URI 进行.

表 2　REST API 的 URI 和对应语义

URI	HTTP Verb	功能
/class/note	POST	创建笔记
/class/note/123456	GET	获取笔记
/class/note	GET	查询笔记
/class/note/123456	PUT	更新笔记
/class/note/123456	DELETE	删除笔记

3.3.3 数据同步和性能优化

智能移动设备在使用中往往受限于网络的稳定性和可用性, 面对这种情况, 移动应用在开发过程中需处理大量的网络连接异常状况并确保移动应用在无法访问网络时的可用性和用户体验[6]. 针对此类问题, 面向移动应用的后端服务平台提供了数据同步框架, 使得移动应用在没有网络连接时仍可正常使用, 并在重新获得网络连接后自动完成移动端到云端的数据同步, 此外平台还优化了 SDK 中对网络服务的请求从而达到性能优化.

数据同步框架包括移动端同步 API(Client Sync API)、移动端同步服务(Client Sync Service)和云端同步 API(Cloud Sync API). 当开发者希望某些数据是在离线情况下可操作的时候, 只需调用 Client Sync API, 这些数据将首先被 Client Sync API 写进移动端数据同步缓存中, 接着 Client Sync Service 将判断当前的网络连接状况, 如果当前设备处于离线状态, 数据同步缓存中的数据将被 Client Sync Service 不断更新到设备的本地持久化存储中; 如果当前设备的网络连接可用, Client Sync Service 会将数据同步缓存中的数据和设备本地持久化存储中的数据发送给 Cloud Sync API, 开发者可在 Cloud Sync API 中定制解决数据同步冲突的策略, 最终将数据同步至平台后端数据存储中.

此外, 对于连续发生的数据更新操作, SDK 将对这一系列更新操作进行合并, 并通过一次网络服务请求将合并后的操作发送给平台, 平台接收请求后执行批量处理. 这种策略将减少移动端网络连接和请求的次数, 从而达到对移动设备的能耗优化和移动应用的性能优化.

4　案例分析

本节通过一个实际案例说明面向移动应用的后端服务平台的功能使用, 以及对移动应用开发带来的改进. “代驾爽”是我们依托 MobiPlus 后端服务平台开发的一个帮助有代驾需求的人士寻找代驾司机的移动应用.

“代驾爽”的功能包括: 根据地理位置寻找附近的代驾司机; 显示该用户的全部代驾记录以及每条代价记录的时间、地点、费用信息等. 此外, 还可以将代驾的体验分享至新浪微博. “代驾爽”应用界面如图 3 所示.

图 3　代驾爽应用界面

“代驾爽”应用案例使用了 MobiPlus 后端服务平台的多种服务, 如表 3 中举例所示, 代驾客户在代驾完成后对该代驾司机提交评价, 评价的文字内容、打分、本次代驾记录的 id、被评价的司机 id 这些自定义数据封装在请求中, REST API 的 URI 和 POST 方法说明这是一个对评价数据(comment)的插入操作; 当代驾客户查询附近空闲状态的代驾司机时, 会请求平台的地理位置查询服务, 请求中的参数表示以当前客户所在经纬度为圆心 1 公里为半径的圆形范围内并且状态为空闲的司机, 这个 GET 方法的请求面向司机(driver)数据集合.

表 3 代驾爽应用使用的平台服务示例

curl 工具表述的 REST API 使用示例	API 功能
curl -X POST \ -H "Application-Id: 123456" \ -H "Rest-Api-Key: djs" \ -H "Content-Type: application/json" \ -d '{"context":"不错", "stars":4.5, "orderId":"51009b8970fca3425981e4ca", "driverId":"50ffa1dfd2c3b8f4b01cc973"}' \ http:// localhost:8080/mobiplus/api/class/comment	提交评价
curl -X GET \ -H "Application-Id: 123456" \ -H "Rest-Api-Key: djs" \ --data-urlencode 'where={"location": {"$within":{"$center":[[38,138],1]}},"status":3}' \ http:// localhost:8080/mobiplus/api/class/driver	查询附近空闲司机

面向移动应用的后端服务平台全面的支撑了“代驾爽”应用的功能需求, 同时, 开发者无需开发和维护服务器端代码, 从而极大的减少了整个应用开发所需的人力资源和时间成本, 开发周期由传统开发模型下的四周减少为两周.

5 总结与展望

随着移动互联网的爆发式增长和智能终端的普及, 移动应用的开发将面临更大的需求和挑战. 面向移动应用的后端服务平台旨在为移动应用开发提供完整的后端服务, 使开发者可以专注于移动端, 在更短的时间和更少的成本下开发出成功的移动应用. 面向移动应用的后端服务平台深入移动应用的设计趋势和使用习惯, 为移动应用开发提供通用数据服务和普遍场景的特定服务. 下一步, 面向移动应用的后端服务平台将进一步完善功能、优化性能, 为移动应用开发提供完整的后端支撑.

参考文献

1 Accenture, Accenture Outlook 2011 October Supplement Insight the New Age of Mobility. 2011.

2 Electric technology, apps and the new global village. http: //blog.flurry.com//bid/91911/electric-technology-apps-and-the-new-global-village. 2012-11-28.

3 Reference implementation for building RESTful Web services. http: //jsr311.dev.java.net/ nonav/releases/1.1/index.html. 2008-10-21.

4 Chun BG, Curino C, Sears R. Mobius: Unifided messaging and data serving for mobile apps. Proc. of the 10th International Conference on Mobile Systems, Applications, and Services. ACM New York, NY, USA. 2012. 141–154.

5 Fielding RT. Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine. 2000.

6 Burckhard S, Fahndrich M, Leijen D, Wood BP. Cloud types for eventual consistency. Proc. of the 26th European conference on Object-Oriented Programming. Springer-Verlag Berlin, Heidelberg. 2012. 283–307.