计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(12):257−261 [doi: 10.15888/j.cnki.csa.006084]

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

基于 JSON 的离线数据同步策略及应用①

穆鑫鑫[1,2,3], 蒋同海[1], 程　力[1], 马玉鹏[1]

[1](中国科学院 新疆理化技术研究所, 乌鲁木齐 830011)
[2](新疆民族语音语言信息处理实验室, 乌鲁木齐 830011)
[3](中国科学院大学 计算机与控制学院, 北京 100049)

摘　要: 针对智能移动应用的特殊性及其在离线情况下的数据同步问题, 提出了一种数据同步方案, 使用 JSON 技术设计数据交换协议, 移动端离线数据存放在 SQLite 数据库中、使用基于时间戳的冲突检测算法提高同步的准确性, 并采用增量同步方式保证同步的效率和准确性. 将该策略应用在智慧安防系统中, 结果表明, 基于 JSON 离线数据同步效率相比传统基于 XML 的方案提高约 8%.

关键词: 数据同步; JSON; SQLite; 混合移动应用

引用格式: 穆鑫鑫,蒋同海,程力,马玉鹏.基于 JSON 的离线数据同步策略及应用.计算机系统应用,2017,26(12):257–261. http://www.c-s-a.org.cn/1003-3254/6084.html

Offline Data Synchronization Strategy Based on JSON and its Application

MU Xin-Xin[1,2,3], JIANG Tong-Hai[1], CHENG Li[1], MA Yu-Peng[1]

[1](Xinjiang Technical Institute of Physics and Chemistry, Chinese Academy of Sciences, Urumqi 830011, China)
[2](Xinjiang Key Laboratory of Minority Speech and Language Information Processing, Urumqi 830011, China)
[3](School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: In view of the particularity of intelligent mobile applications and its data offline synchronization problems, we put forward a scheme of data synchronization, using JSON technology to design data exchange protocol and the SQLite database to store the mobile terminal offline data, using the conflict detection algorithm based on time stamp to improve the accuracy of synchronization as well as the incremental synchronization mode to ensure the efficiency and accuracy of synchronization. This proposed method is applied to an intelligent security system, and the results show that the efficiency of offline data synchronization based on JSON is about 8% higher than that of the traditional XML based scheme.

Key words: data synchronization; JSON; SQLite; hybrid mobile applications

引言

随着移动互联网的跨越式发展, 移动智能应用的也呈现井喷式增长, 通常系统所有的信息都保存在服务端 (Server-side), 客户端 (Mobile device) 数据暂存在客户端, 为了保障数据的一致性, 一般需要进行客户端和服务端数据的冲突检测以及双向的数据同步, 所以, 数据同步问题也成了最近国内外移动计算研究领域中的热点. 然而移动应用在以下几个方面有特殊性: 无线网络带宽限制, 移动应用资源有限 (如存储), 移动性 (设备和网络), 网络中断, 电池容量有限[1]. 为了在上述各种情况下很好地进行数据在客户端和服务端间的同步, 研究者们提出了许多方法和策略. MY Choi 等提出了一种基于信息摘要 (Message digest) 的数据同步算法 SAMD[2], 包括移动端数据库、同步数据服务器和服务端数据库服务器三个组件, 和传统关系型数据库管理系统 (RDBMS) 中数据同步框架类似 (如图 1), 该

① 基金项目: 中国科学院西部之光人才培养计划项目 (XBBS201319); 新疆维吾尔自治区青年科技创新人才培养工程基金项目 (2014721033)
收稿时间: 2017-03-08; 修改时间: 2017-03-27; 采用时间: 2017-03-29

策略虽然保证了数据同步的安全性和准确性, 但由于加入了中间层同步服务器来进行数据同步服务, 所以在性能上大打折扣, 不能满足移动应用的需求; D Sethia 等人提出了一种基于元组时间戳的同步策略 MRDMS[3], 由移动端和服务端两部分组成, 因为省去了中间同步数据服务器的处理, 相比文献[2]中的 SAMD 省去了同步数据服务器处理数据时间, 不足是 MRDMS 采用的基于元组的时间戳, 要维护每个元组的时间戳组成的时间戳表, 虽然, 这样做同步的数据更加精确, 但是维护大量时间戳表会消耗很多时间和系统资源, 而移动端系统资源本身很有限, 所以该方法的效率有待提高; 本文提出一种基于 JSON 和 SQLite 数据库的离线数据同步策略, 设计了基于 JSON 的数据交换协议, 利用时间戳技术检测数据冲突, 使用增量同步方式保证传输数据的高效性, 避免不必要的冗余数据传输, 使用 HTTP POST 请求进行数据传输, 并将该同步策略应用于混合移动应用“智慧安防”中, 结果表明, 该策略不仅能保证离线情况下的数据完整性, 而且在离线数据同步效率上优于传统的同步方法.

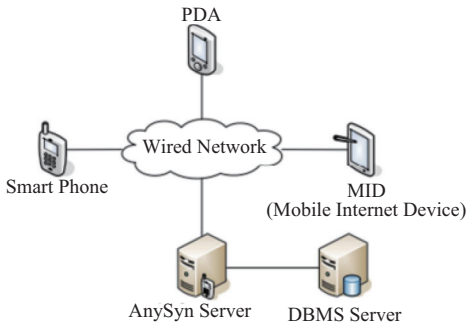

图 1 经典数据同步框架

1 问题描述

互联网的高速发展和移动计算技术的不断发展, 使得移动智能设备在日常生活中扮演的角色越来越重要, 移动应用的数量急剧增长, 通常, 系统数据保存在服务端, 应用端也会保存少量备份, 当数据在不同的移动终端中被修改后, 就要将修改的结果同步到服务端, 在网络连接正常的情况下, 可以进行实时的数据同步. 但是, 由于各种原因会发生移动终端离线的情况, 在网络连接中断的时候, 数据的一致性和完整性就显得格外重要. 离线情况下的数据同步算法应当满足如下几个要求[4]:

(1) 应当保证最少的数据传输量.

(2) 离线后的数据改变在网络重新连接之后要同步到服务端.

(3) 数据同步算法要能很容易地用现有技术实现.

(4) 数据传输格式要简单高效.

基于以上各方面的综合考虑, 本文提出了一种基于 JSON 和 SQLite 的离线数据同步策略 (ODSSJS), 后面部分将对 ODSSJS 做详细说明.

2 同步策略

离线数据同步策略主要包括: 移动端数据存储策略、数据交换协议、冲突检测策略和同步算法, 下面对各个组成部分做详细解释.

2.1 移动端存储策略

目前移动应用中使用的存储数据的方法主要有 Android 存储类 SharedPreference、文件存储、HTML5 Local Storage 存储以及 SQLite 数据库. 然而不同的存储方式有各自特殊的特点和适用范围, 如表 1 所示.

表 1 移动端数据存储方式

名称	特点	适用范围	使用难度
文件存储	同传统的文件操作类似, 通过数据流进行文件的读写	文本、多媒体资源存储	一般
SharedPreference	Android中轻量级的存储类, 数据通过键值对的方式保存和查询	少量、非结构化数据存储	较简单
HTML5 Local Storage	HTML5中新加入的特性, 解决了cookie存储空间不足的问题	少量、结构化数据存储	较简单
SQLite	轻量级的关系型数据库, 以表结构存储数据, 通过SQL操作数据库	结构化较复杂数据的存储, 需要插入和查询等操作	较难

由于在安防系统中数据中会有图片等多媒体数据, 另外, 离线情况下的数据同步是本文的关注点, 所以综合考虑各种因素之后, 选择 SQLite 数据库存储移动端数据.

2.2 数据交换协议

数据交换协议指的是数据交换的格式, 以往的移动应用数据交换多数采用 XML 格式的数据 (如文献[8]), XML 用标签对来包含数据, 然后通过 XML 解析

器来进行解析. JSON(JavaScript object notation) 是一种比 XML 使用更加容易的数据交换格式, 可以直接使用 JavaScript 来解析, 所以, 相比 XML , JSON 使用更加便捷而且效率要比 XML 高[9]. 从简洁和高效两个方面考虑, 本文数据交换采用 JSON 格式, 具体做法为: 我们将要同步的数据按照事先定义的协议格式表示成 JSON 对象, 然后将其转换成 JSON 字符串写入文本文件中, 压缩后发送到要同步的一端, 同步端将数据解压, 解析其中的 JSON 对象, 然后将数据通过 SQL 插入到移动端 SQLite 数据库或者服务端数据库中.

服务端发送给移动端的 JSON 数据格式定义为:

{“success”: “true”,

“message”: “服务器操作异常时的提示信息”,

“obj”: {plans:[巡检任务], items:[巡检明细]}”, //服务端数据

}

移动端上传给服务端的 JSON 数据格式定义为为:

{“success”: “true”,

“message”: “服务器操作异常时的提示信息”,

“obj”: {items:[巡检任务], appendix:[资源信息], station:[定位信息]}”, //客户端数据

}

“success”: 表示数据传输成功与否, true 代表成功, false 代表失败.

“message”: 由数据发送端传给接受端的关于数据操作中的具体情况, 如数据库链接错误时提示数据库连接错误.

“obj”: 协议中的数据主要存放在 obj 中, 上传协议和下载协议中的 obj 具体内容略有差别.

下载协议 (服务端发送给移动端)obj 中包含:

“plan”: 存放具体任务的信息.

“item”: 存放任务的明细, 这里就是要同步数据存放的地方.

上传协议 (移动端发送给服务端)obj 中的字段有:

“item”: 和下载协议一样, 存放任务的明细, 存放主要同步数据存.

“appendix”: 存放要上传服务端的其他资源信息, 如图片等多媒体资源.

“station”: 位置信息, 为了在 PC 端应用中将具体设备等显示在地图上.

2.3 同步算法描述

目前实现数据同步的方式主要有基于触发器、时间戳、标志位以及日志处理等方式, 其中触发器方式需要消耗的系统资源很大, 对于移动客户端来说资源有限, 触发器的方式不是很适用[8]; 而读日志的方式也需要耗用额外的时间和资源来进行日志解析, 所以效率较低, 在移动设备资源有限的情况下, 读日志的方式效率低下; 基于时间戳和基于标志位的方式类似, 但时间戳能明确给出数据的修改时间, 对于要使用数据修改时间的系统能获取操作时间是很重要的, 所以, 本文的离线数据同步策略中采用的是时间戳方式来检测数据的冲突.

离线数据的同步可以分为上传和下载两个部分. 上传由 MC(移动客户端) 发起, 当 MC 连接到 S(服务端) 时, 将离线操作数据发送给 S; 而下载可以有不同的实现方式, 可以由 MC 发起, 也可以是先由 MC 将离线操作数据上传后, S 将差异数据返回给 MC.

数据同步方式有全量同步和增量同步两种方式, 全量同步一次性同步数据库中的所有数据, 而增量同步则只对两个数据库中的差异数据进行同步. 对于离线数据的同步, 要保证传输最少的数据量[10], 所以本文中的离线同步采用增量同步方式.

MC 和 S 数据表中数据冲突检测需要的字段为:

T_MC	T_S	IS_VALID
MC修改时间	S修改时间	是否删除

(1) 数据下载算法

BEGIN

① 同步.

② MC 将本地最大的 T_MC 作为参数向 S 发出同步请求.

③ S 将 T_S > T_MC 的数据从数据表抽取出来, 按下载协议格式组装成 JSON 文件 F, 将 F 压缩成压缩文件 D 发送给 MC.

④ MC 收到压缩文件 D, 解压出 JSON 文件 F.

⑤ 如果 F 为空, 则转到 END.

⑥ 如果 F 不为空, 则 MC 解析 JSON 文件 F 内容:

a) 将 F 中的 obj 中携带的数据转换成 SQL 操作.

b) 创建 SQLite 数据库 DB.

c) 新建事先设计的 SQLite 数据表 T.

d) 通过事先封装的 Angular 服务 (service)API 将每条记录插入 T 中.

⑦ 数据写入完成后向服务器发送 SUCCESS 消息.

END

(2) 数据上传算法

BEGIN

① MC 将待上传数据按照同步协议格式组装成 JSON 文件 F.

② MC 将 F 压缩成 D 发送给 S.

③ S 收到数据 D, 响应 SUCCESS(上传成功).

④ 如果 D 为空, 则转到 END.

⑤ 如果 D 不为空, 转到⑥.

⑥ S 将 D 加压为 JSON 文件 F 后解析.

a) 将 F 中的 obj 中的数据记录和对应的表中的记录作对比.

b) 如果 T_MC > T_S, 则插入新数据到服务端数据表中.

c) 更新 T_MC 和 T_S.

d) 否则转到 a).

⑦ 删除 D 和 F.

END

整个离线数据同步过程时序图如图 2 所示.

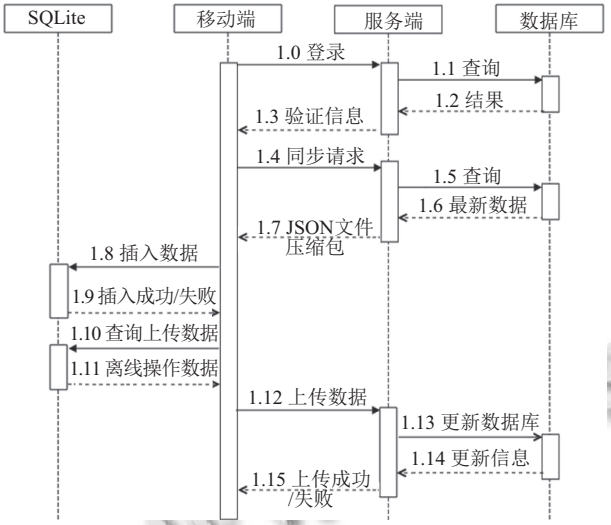

图 2 离线数据同步时序图

3 应用及分析

3.1 应用场景

在安防系统中, 需要对园区设施设备等进行巡检, 可以通过如扫描 NFC 卡的方式来检查设施和设备, 而为此开发的移动应用需要在有网络的情况下才能将检测数据上传到服务器, 但园区通常很大, 很多绝大部分区域都没有网络覆盖, 这就需要在离线的状态下对设备设施进行检查, 然后将数据在有网络环境下上传到服务端, 因此, 离线数据的同步策略就变得尤为重要.

笔者将本文中的离线数据同步策略作为“智慧安防”系统的离线数据同步策略, 整个系统分为移动端、PC 端和服务端, 系统架构如图 3 所示, PC 端界面如图 4 所示, 移动端, 其中移动端采用混合开发框架 (Hybrid) Cordova 来开发, 相比原生移动应用 (Native) 和 web 移动应用, Hyper 应用开发周期短, 可扩展性良好, 还能提供对 SQLite 数据库操作的第三方 JavaScript 库如 cordova-sqlite-storage, 大大减轻了开发难度, 然而混合移动应用也有不足的地方, 就是是运行速度不如 Native 应用快, 因为它是通过第三方 API 来调用 Android native API 去操作系统硬件的, 要经过中间层的额外性能损耗, 所以, 很自然其性能不如 Native APP[10]; 移动端和 PC 端的 UI 及前端业务流程都使用目前流行的 Angular 框架来实现, 同时, 利用第三方库 cordova-sqlite-storage 封装了用于操作 SQLite 数据库的服务 (SQLiteService), 通过该 service 来进行数据的增删改查 (CRUD) 将十分方便; 系统后端使用 SSH 整合框架来实现, 使用 Oracle 数据库作为服务端数据管理系统.

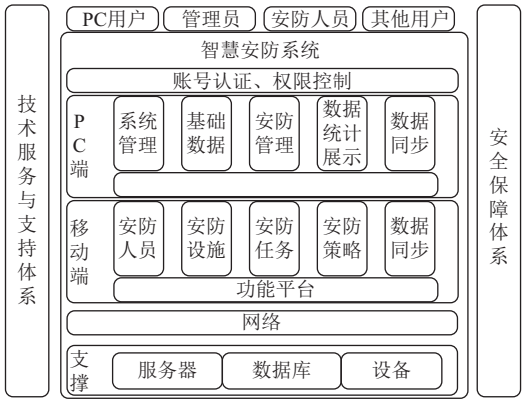

图 3 智慧安防系统架构图

图 4 PC 端界面

系统在前端实现离线数据同步协议中的数据存储、离线数据上传以及向服务端请求同步等功能, 在

服务端实现数据的冲突检测和提供同步数据的下载.

3.2 实验结果及分析

实验在 WiFi 环境下进行, 移动端 Android 版本是 4.4.0, 通过对文献[2]中基于同步服务器的同步方法 (SAMD)、文献[8]中基于 XML 的同步策略和本文基于 JSON 的离线同步策略 (ODSSJS) 三种方法同步不同数据量的离线数据时的效率进行了对比, 结果如图 5 所示.

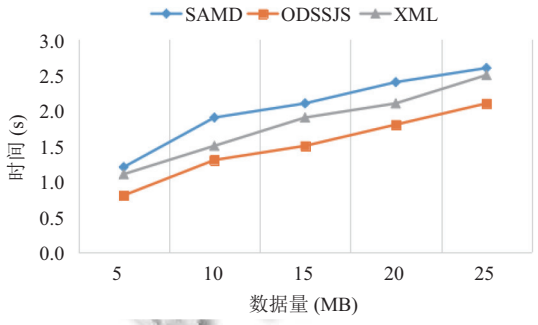

图 5 同步效率对比图

由图 5 可以看出, SAMD 在三种策略中是效率最差的, 原因在于 SAMD 采用如图 1 的架构, 通过中间层数据同步服务器来进行数据同步, 这样做就在中间层处理同步数据过程中确实损失了效率; 而文献[8]中基于 XML 的同步策略虽然也是用增量同步的方式保证传输数据量最少, 但在携带相同数据的情况下, 相比 JSON 格式, XML 总数据量更大, 解析效率低, 所以整体效率不如本文的 ODSSJS.

4 结束语

本文提出了一种基于 JSON 的离线数据同步方案, 设计了基于 JSON 的数据交换协议, 通过 HTTP POST 传输数据, 使用 SQLite 存储移动离线数据, 通过时间戳技术来检测数据冲突, 同时, 采用增量同步的方式保证数据同步过程的高效. 该方案已经在“智慧安防”混合移动应用中实现, 可靠性和高效性完全达到移动应用离线数据同步的要求. 下一步研究可在进一步提高同步效率及在其他设备 (如 POS 机、PDA 设备) 中的扩展应用等方面进行.

参考文献

1 Imam AA, Basri S, Ahmad R. Data synchronization between mobile devices and server-side databases: A systematic literature review. Journal of Theoretical & Applied Information Technology, 2015, 81(2): 364–382.

2 Choi MY, Cho EA, Park DH, *et al*. A database synchronization algorithm for mobile devices. IEEE Trans. on Consumer Electronics, 2010, 56(2): 392–398. [doi: 10.1109/TCE.2010.5505945]

3 Sethia D, Mehta S, Chowdhary A, *et al*. MRDMS-mobile replicated database management synchronization. Proc. of International Conference on Signal Processing and Integrated Networks. Noida, India. 2014. 624–631.

4 郝平, 林原冲. 一种移动网络下基于双时间戳的数据增量同步研究. 计算机应用与软件, 2016, 33(4): 143–145, 226.

5 知乎. Android 开发时, 如何实现和服务器的数据同步? https://www.zhihu.com/question/19708808. [2016-06-14].

6 陈俊龙. Android 平台的数据传输与同步的设计与实现[硕士学位论文]. 北京: 北京邮电大学, 2015.

7 Imam AA, Basri S. Data synchronization patterns in mobile application design. Journal of Theoretical and Applied Information Technology, 2015, 81(2): 364–382.

8 刘宇, 戴鸿君, 郭凤华, 等. Android 平台可增量同步的网络应用协议. 计算机工程, 2011, 37(18): 59–61. [doi: 10.3969/j.issn.1000-3428.2011.18.020]

9 w3schools. JSON introduction. http://www.w3schools.in/json/intro/. [2016-11-01].

10 Carmatec. Hybrid apps vs native apps in the mobile app development world. http://www.carmatec.com/blog/hybrid-apps-vs-native-apps. [2015-08-13].