doi:10.6041/j.issn.1000-1298.2016.01.038

# 田间数据传输同步策略与中间件研究

李　林[1,2]　王　竹[1]　呼延正勇[1]　赵明明[1]　曹　津[1]　朱德海[1,2]

(1. 中国农业大学信息与电气工程学院, 北京 100083; 2. 农业部农业信息获取技术重点实验室, 北京 100083)

**摘要**: 移动终端的田间信息采集相比于传统的田间信息采集,具有灵活性强、采集方便等优势,已成为田间信息采集的主要手段之一。但是在进行田间信息采集时存在数据同步问题,主要表现在慢同步频发、无为数据(已经传输过的数据)重传、服务器端变化数据没有及时同步到移动终端等问题。针对这 3 个数据同步的主要问题,基于 Sync ML 协议,提出了慢同步策略,并对该协议进行了改进;设计了主动推送策略和变化数据捕获策略;在此基础上设计并且实现了改进后的移动数据同步中间件,并且把该中间件应用到种业科研育种田间信息采集系统和水稻定位试验打分系统中,试验证明,该中间件提高了数据同步的效率,减少了无为数据的重复传输,表现出良好的实用性和普适性。

**关键词**: 田间信息; 移动数据同步; 中间件; 同步策略; Android

**中图分类号**: S2　　**文献标识码**: A　　**文章编号**: 1000-1298(2016)01-0279-10

# Research on Field Data Transmission Synchronization Strategy and Middleware

Li Lin[1,2]　Wang Zhu[1]　Huyan Zhengyong[1]　Zhao Mingming[1]　Cao Jin[1]　Zhu Dehai[1,2]

(1. *College of Information and Electrical Engineering*, *China Agricultural University*, *Beijing* 100083, *China*

2. *Key Laboratory of Agricultural Information Acquisition Technology*, *Ministry of Agriculture*, *Beijing* 100083, *China*)

**Abstract**: Compared with the traditional field information collection, field information collection with mobile terminal has strong flexibility, easy collection and other advantages, and has become one of the main means of agricultural field data acquisition. But when the field information was collected, data synchronization has disadvantages of slow sync frequency, unnecessary retransmission of data, and server-side changes, which can not synchronize to the mobile terminal timely. The grammar and data synchronization process of the Sync ML protocol was lucubrated, the data synchronization process of mobile data acquisition was analyzed and various problems were encountered. Sync ML protocol was improved, slow synchronization strategy was proposed, and active push synchronization strategy and change data capture strategy were designed. The strategy of combining pull synchronous with push synchronous was designed, the push synchronous was used to push synchronous messages, and the pull synchronous was used to synchronize the data from server. The data synchronization strategy middleware was designed and implemented. Mobile terminal based on Android and C# technology was used. The middleware was used in information acquisition system and rice-positioning-test-score system. The results showed it significantly improved the efficiency of data synchronization and reduced the unnecessary retransmission of data. It makes the field information collection system in favor of the promotion of modern information agriculture, and enhances the work efficiency of field data acquisition and data accuracy. The experimental results verify that the middleware has high practicability and universality, and makes data transmission not increase as the number of the synchronization.

**Key words**: field information; mobile data synchronization; middleware; synchronization strategy; Android

收稿日期: 2015-05-11　修回日期: 2015-08-04

**基金项目**: 国家自然科学基金项目(31471762)

**作者简介**: 李林(1963—),女,教授,博士,主要从事软件工程和软件自动化研究,E-mail: lilincau@126.com

**通信作者**: 朱德海(1962—),男,教授,博士生导师,主要从事 3S 技术在农业、国土资源等领域应用研究,E-mail: zhudehai@cau.edu.cn

## 引言

随着移动互联网技术的发展,利用智能终端的数据获取方式逐渐在农业生产和科研领域广泛应用,基于移动终端的田间信息采集方式相比于传统的田间信息采集,具有灵活性强、信息采集方便等优势,已成为田间信息采集的主要手段之一[1-6]。

数据同步一直被认为是数据传输过程中最重要的环节之一,在学术界,同步复制技术已被认为是移动数据库同步的一个关键技术[7-8],比如:多版本冲突消解方法、两级复制机制、主动复制机制、容错型定额同意方法[9]、三级复制机制[10]、虚拟主副本法[11]、基于双时间印的事务级同步机制[12]、移动复制数据系统冲突检测和消解策略[13]。

外业工作中存在网络弱、数据传输流量大等问题,因此,在信息传输过程中常遇到慢同步频发、无为数据(已经传输过的数据)重传、服务器端变化数据没有及时同步到移动终端等问题。本文针对上述问题,对数据传输网络同步协议进行改进,设计主动推送策略和移动数据同步策略,在 Android 平台上设计实现改进后的移动数据同步中间件[4,14],并且把该中间件应用到种业科研育种田间信息采集系统和水稻定位试验打分系统[15]中,以测试中间件的实用性和普适性。

## 1 移动数据同步策略制定

### 1.1 移动数据同步策略

#### 1.1.1 数据同步总体策略

数据同步包括推送式和拉入式 2 种方式。本文将这 2 种方式结合起来解决数据同步中的问题。推送式同步主要负责向移动终端推送同步通知消息,通知移动终端进行数据同步;拉入式同步根据同步通知,主动向服务器发出同步请求获取更新的数据,在这一过程中,也包括在特定条件下发起的慢同步,数据同步总体策略如图 1 所示。通常,移动终端与服务器端进行数据同步包括在用户登录进入系统时同步和移动终端接收到同步通知消息后发起数据同步 2 种情况。移动终端第 1 次连接服务器端数据库时,用户名密码验证通过后,进行移动终端唯一编号在服务器端的注册,注册完成后进行慢同步。对于变化数据的获取,在服务器端和移动终端都采用了行级触发器来实现,数据表一有变化,就会把变化的类型、变化数据所在表、变化数据唯一标识等信息存储到表示变化数据的信息表 BC_ChangeLog 中(BC_ChangeLog 主要用于记录服务器端和移动终端捕获的变化数据,见 1.3 节)。

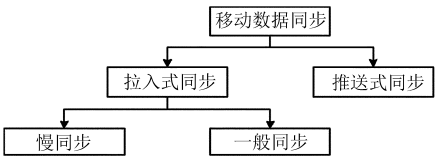


图 1 数据同步总体策略图

Fig. 1 Overall strategy figure of data synchronization

#### 1.1.2 慢同步的改进

在传统的慢同步实现方式[16]中,慢同步发生的频率较高,只要本次同步过程中出现异常,下次就是慢同步。而且在慢同步实现过程中,会造成已经同步过的无为数据的重复传输。

针对此问题,本文基于 Sync ML 协议对慢同步做了如下改进:在消息头加入 SlowTableLog 对象标签,对象包括 TableName、Last 和 Order 属性,TableName 表示数据表的名称,数值类型为 string,Last 表示是否为该业务的最后一个数据表,数值类型为 string,Order 表示数据表在业务中的编号(1,2,…,$n$),同时,在消息头加入 SyncType 标签,取值为 1 时表示一般同步,取值为 0 时表示慢同步。改进的慢同步过程如下:以移动终端存储的 SlowTableLog 对象的 Last 属性为判断依据,如果 Last 值为空,则移动终端发起慢同步,将 Order 值发送到服务器端,从服务器端同步编号为 Order + 1 的数据表。在服务器端,给 Order 赋值数据表对应的编号,TableName 赋值数据表的表名,然后将数据表和 SlowTableLog 对象传输到移动终端,如果是业务的最后一个数据表,除了上述操作外,还需要给 Last 赋值 last。移动终端如果接收到包括 SlowTableLog 对象的 Last 属性为 last 的数据,则在本地更新数据库成功后,将本地存储的 SlowTableLog 对象的 Last 属性设置为 last,终止数据同步,表示慢同步完成。

通过此种改进,明显减少了慢同步发生的次数,只要慢同步完成,移动终端数据库没有接受删除,就不会再发生慢同步。如果在慢同步过程中发生异常,造成同步失败,下次慢同步时接着上次同步失败的断点续传,减少了重复数据的传输。

#### 1.1.3 慢同步策略

在本文中,慢同步策略是在 1.1.2 节 Sync ML 协议中慢同步做出改进的基础上实现的:服务器端建立了一个表 BC_BusinessTable(表 1),记录了某个应用业务下所有的数据表,并且对它们进行了从 1 到 $n$ 的编号(采集信息类不能设置为最后一个编号),用于控制慢同步的进展。在移动终端利用 SharedPreferences 工具存储 SlowTableLog 对象,用于控制慢同步的发生和进展。在数据传输过程中突然断网或者连接失败,再次连接进行同步时,通过 BC_

BusinessTable 的 Order 值和移动终端 SlowTableLog 的 Order 属性以及 SyncHdr 的 SlowTableLog 标签来进行断点续传。慢同步的发起条件由移动终端存储的 SlowTableLog 对象的 Last 属性决定，如果该值为空，则发起慢同步，如图 2 所示，Order 属性用于记录同步到具体哪一个数据表，Last 属性用于标志慢同步是否完成。如果服务器端传递的 Sync ML 数据包中，SlowTableLog 标签的 Last 值为 last，表示同步到最后一个数据表，在移动终端完成数据更新后，将移动终端 SlowTableLog 的 Last 属性赋值 last，标志着慢同步完成。发起慢同步时 Sync ML 消息头中的 SyncType 属性值为 0，服务器根据此来判断发起的数据同步的类型。

**表 1　BC_BusinessTable 表结构**

**Tab. 1　Structure of BC_BusinessTable**

| 字段 | 数据类型 | 记录类型 | 备注 |
|---|---|---|---|
| ID | int | | 主键 |
| TableName | varchar(50) | 表名称 | 不能为空 |
| BusinessName | varchar(100) | 业务名称 | 不能为空 |
| Author | int | 录入者 | |
| CreateTime | datetime | 录入时间 | |
| BusinessType | int | 数据类型 | 1 为基础，2 为业务 |
| Order | int | 排序 | 不能为空 |

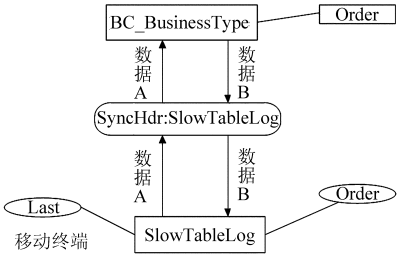


图 2　慢同步策略图

Fig. 2　Strategy graph of slow synchronization

### 1.1.4　一般同步的改进

Sync ML 协议[16]一般同步主要包括服务器端单向同步、服务刷新同步、移动终端单向同步、移动终端刷新同步、双向同步 5 类。这 5 种同步都是由移动终端发起，同步的都是两端最新变化的数据，但是这种同步方式的实效性不高，服务器端有变化数据时不能及时地同步到移动终端。

针对上述问题，本文做了如下改进：将推送式同步加入到一般同步中，服务器每隔 3 min 遍历一次 BC_PrepareSync 表（表 2），向该表有记录的移动设备推送同步消息，移动终端接收到同步消息后，系统在后台自动发起一般同步请求在服务器端同步变化数据。除此之外，用户在非首次登陆时，移动终端也会发起一般同步请求，向服务器端同步上次变化的数据。对于移动终端的变化数据，采用用户手动控制上传的方式实现，避免把没有采集完的部分数据同步到服务器端。

**表 2　BC_PrepareSync 表结构**

**Tab. 2　Structure of BC_PrepareSync**

| 字段 | 数据类型 | 记录类型 | 备注 |
|---|---|---|---|
| ID | int | | 主键 |
| KeyID | int | BC_ChangeLog 主键 | 不能为空 |
| DevCode | varchar(10) | 移动设备编号 | 不能为空 |
| UserID | varchar(10) | 用户 ID | 不能为空 |
| RoleID | int | 角色名 | 不能为空 |

通过这种改进，对于同步到移动终端的变化数据，极大的提高了其时效性，对于移动终端变化的数据，减少了不必要的重复上传，节省了用户的网络流量。

### 1.1.5　一般同步策略

在本文中，一般同步策略是在 1.1.4 节基础上实现的：一般数据同步包括非慢同步登录时的数据同步和移动终端接收到数据同步通知消息后发起的数据同步 2 种情况。移动终端发起一般同步时，Sync ML 消息头中的 SyncType 属性必须设置为 1，这样服务器才会作为一般同步处理，采用把服务器端变化数据同步到移动终端和移动终端把变化数据同步到服务器分开处理的策略，前者是自动进行，后者必须通过用户控制同步操作来完成。图 3 表示服务器端同步数据到移动终端的实现策略。服务器端接收到同步请求后，通过 BC_PrepareSync 和 BC_ChangeLog 查找需要同步的数据，然后把数据发送到移动终端，移动终端更新完数据库后，把 BC_Status 表和 BC_Map 表（只针对数据采集表中添加的新数据）中数据发送到服务器端，服务器端更新 BC_Map 表，针对 BC_Status 数据删除 BC_PrepareSync 中相关的数据，接着发送给移动终端同步完成状态码 StateCode，移动终端根据此状态码删除 BC_Map 和 BC_Status 中的数据，完成数据同步。

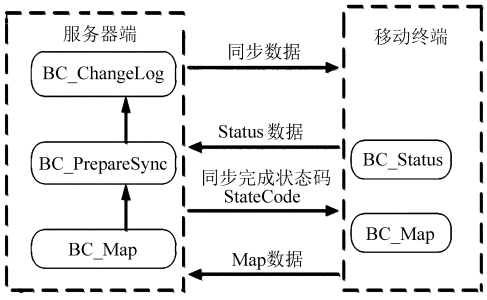


图 3　一般同步实现图

Fig. 3　Realizing graph of general synchronization

对于移动终端同步变化数据到服务器端，主要用到移动终端的 BC_ChangeLog 表，该表记录移动终端采集数据的变化情况。移动终端把其变化的数据同步到服务器端，服务器端将此数据更新至中心数

据库,如果是新增加数据,需要更新 BC_Map 表,完成后返回同步完成标志信息给移动终端,移动终端把表示是否同步到服务器端的标志属性 Flag 设置为 1,表示此次同步完成。

### 1.2 主动推送策略

主动推送指的是服务器端主动向移动终端推送数据的技术,它的优势在于一次可以向很多移动终端同时推送信息,而且数据同步比较及时,使得需要同步的数据及时准确地同步到相应的移动终端。在本研究的主动推送实现中,使用了极光推送技术,调用它的 API 实现推送,考虑到实际应用数据的安全性和敏感性,服务器端只推送同步消息,同步数据通过移动终端发起的同步请求来实现。图 4 是主动推送流程图。

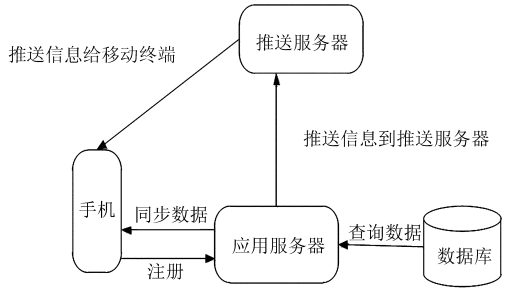


图 4 主动推送流程图

Fig. 4 Implementation flow of active pushing

考虑到实际应用数据的安全性和敏感性,本文设计的是服务器端推送同步通知消息策略,它主要是通过数据库中的表 BC_PrepareSync(表 2)来实现的,该表中存储有需要推送信息的移动终端编号(DevCode)、用户标识(UserID)以及中心数据库记录变化数据的表 BC_ChangeLog 的主键(KeyID)和用户所属的角色标识(RoleID),如表 2 所示,服务器每隔 3 min 遍历一次该表,然后向该表中记录的所有移动终端发送同步通知消息,移动终端收到同步消息后,立即向服务器发起同步请求,把中心数据库需要被同步的数据同步到自己的数据库中,并且发送给服务器完成状态信息,服务器根据移动终端传来的信息清除 BC_PrepareSync 表中的相关数据,并发送完成状态码给移动终端,移动终端如果收到完成状态码为 10,就会删除该表中相应的数据。

移动终端在后台开启一项服务(只要应用系统没有关闭,就一直在运行),专门用于监听服务器推送的数据同步通知消息并且进行处理。移动终端监听消息,更新本地数据库包括发送更新完成的状态信息,都是在系统后台运行,对用户完全透明,提升了用户的使用体验。

### 1.3 变化数据捕获

变化数据捕获技术是为了实现数据仓库、数据集以及其他业务系统中数据的增量更新。在本文的设计中,主要应用触发器法[17]和日志法[17]结合的方法,建立变更日志表,通过给各个表建立触发器,将其变化的数据记录在变更日志表中。

在中心服务器,建立变更日志表 BC_ChangeLog,记录捕获变化的数据,如表 3 所示。ChangeID 表示捕获的变化数据在自己所在表中的唯一标识,ChangeSource 表示变化数据所在的表,由这两项可以捕获到数据库中变化的数据。ChangeTime 表示数据变化时间,ChangeType 表示变化的类型,I 表示增加,U 表示修改,D 表示删除,DevCode 表示移动设备编号,如果移动终端对数据更新,就会记录该设备编号,防止给自己同步,ChangeUser 表示更新该条数据的用户。这个表的所有数据都是通过建立在基础信息表和采集数据表中的触发器来获取的。

**表 3 服务器端 BC_ChangeLog 表结构**

**Tab. 3 Structure of BC_ChangeLog in server**

| 字段 | 数据类型 | 记录类型 | 备注 |
|---|---|---|---|
| ID | int | | 主键 |
| ChangeID | varchar(50) | 变化数据 ID | 不能为空 |
| ChangeTime | time | 变化时间 | 不能为空 |
| ChangeType | char(1) | 变化类型 | 不能为空,值为 I/U/D |
| ChangeSource | varchar(100) | 变化数据所在表 | 不能为空 |
| DevCode | varchar(200) | 移动设备编号 | 服务器端修改时为空 |
| ChangeUser | int | 修改者 ID | 不能为空 |

在移动终端,建立表 BC_ChangeLog,如表 4 所示,用以记录移动数据库捕获的变化数据。跟中心数据库变化日志表不同的是,移动终端没有记录移动设备号,而是增加了一个表示是否把该变化数据同步到中心数据库的标志字段 Flag,Flag 为 0 表示没有同步,Flag 为 1 表示该变化数据已经同步到中心数据库。

**表 4 移动终端 BC_ChangeLog 表结构**

**Tab. 4 Structure of BC_ChangeLog in mobile terminal**

| 字段 | 数据类型 | 记录类型 | 备注 |
|---|---|---|---|
| ID | int | | 主键 |
| ChangeID | int | 数据 ID | 不能为空 |
| TableName | text | 表名 | 不能为空 |
| ChangeType | text | 修改类型 | I/U/D |
| ChangeTime | text | 修改时间 | 不能为空 |
| ChangeUser | int | 修改者 ID | 不能为空 |
| Flag | int | 是否上传 | 0 为未上传,1 为上传 |

## 2 移动数据同步中间件设计

移动终端设备的信息采集系统都设在移动终端的移动数据库中,以保证在不连网的情况下也能进

行信息采集。为了在条件允许的情况下，尽量保持两边数据库的数据一致性，就需要与中心数据库进行同步。在信息采集时，需要用到一些基础数据，这些数据都是在服务器端维护，所以要在有网络的情况下，尽量保证服务器端的相关数据与移动终端相一致，而对于移动终端采集的数据，则通过用户控制同步，以免一条数据经常修改与同步，造成网络流量的浪费。本文设计了一款在 Android 平台上运行的、基于 Sync ML 协议的移动数据同步中间件，中间件便于在一台或多台机器上的多个软件通过网络进行交互。通过它可以实现服务器端的新变化数据及时同步到移动终端，移动终端新变化数据同步到服务器端，该中间件实现了上文阐述的同步策略，并可以应用到任何基于 Android 的信息采集系统中。

### 2.1　中间件系统框架体系结构

本文设计的中间件采用 C/S 架构，整个系统框架分为服务器端和移动终端两部分，如图 5 所示。服务器端中心数据库采用 SQLServer、C#语言实现，移动终端数据库采用 SQLite，开发语言使用基于 Android 的 JAVA，服务器端和移动终端进行数据同步时，利用 HTTP 协议进行网络访问和数据传输，数据以 JSON 形式进行传输。

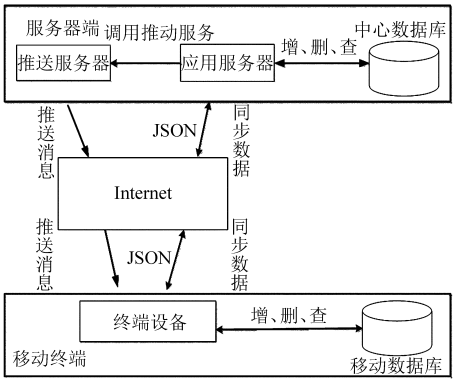


图 5　系统总体架构图

Fig. 5　System's overall architecture diagram

服务器端包括应用服务器、数据库服务器和推送服务器，总体架构如图 6 所示。推送服务器主要用于推送同步通知消息到移动终端，本文使用极光公司的推送服务器，通过调用该公司提供的极光推送 API 向移动终端推送同步消息。应用服务器负责用户身份和设备的验证、数据同步过程中的冲突处理、改进后 Sync ML 协议的实现、JSON 解析与生成、访问和操作数据库服务器等，同时，应用服务器还实现了主动推送策略，通过主动推送策略调用推送 API。数据库服务器主要用于存储数据和编写的触发器。

在移动终端，主要包括移动数据库和各类移动

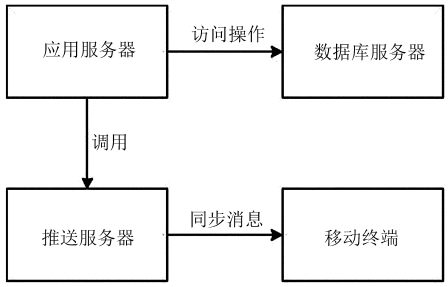


图 6　服务器架构图

Fig. 6　Server's architecture diagram

应用，总体架构如图 7 所示。移动应用可以对移动数据库进行数据的存储、访问以及优化；同时也负责监听服务器推送过来的同步消息并对其处理，即根据自身任务向服务器发起数据同步、管理移动数据库数据和处理同步过程中出现的数据冲突等功能。

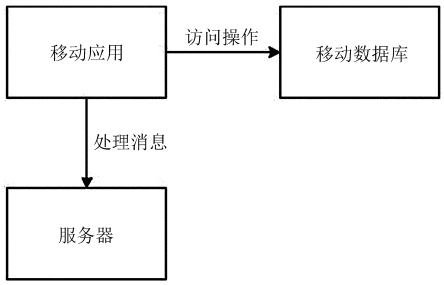


图 7　移动终端架构图

Fig. 7　Mobile's architecture diagram

### 2.2　中间件功能设计

图 8 展示了中间件的功能组织结构。服务器端主要包括同步日志管理、推送管理、同步数据操作管理、同步会话管理、数据传输管理、数据冲突管理 6 个模块。其中同步日志管理模块主要用于实现服务器端中心数据库中变化数据的捕获；推送管理模块主要用于实现服务器端推送同步消息到移动终端；同步数据操作管理模块主要实现在同步过程中对数据的操作；同步会话管理模块主要实现服务器端和移动终端建立同步会话，即对移动终端发起的同步请求做出响应；数据传输管理模块负责对数据传输提供服务；数据冲突管理模块主要解决移动终端数据同步到服

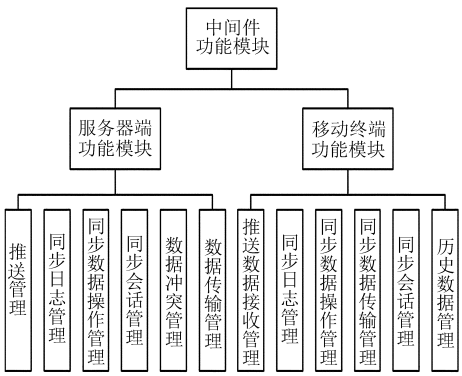


图 8　中间件功能图

Fig. 8　Function diagram of middleware

务器端,与服务器端数据发生冲突或者不同移动终端更新同一条数据时发生冲突的问题。

移动终端主要包括同步日志管理、历史数据管理、同步数据传输管理、同步数据操作管理、同步会话管理和推送数据接收管理 6 个模块。其中同步日志管理主要是通过 SQLite 数据库中提供的触发器机制,给数据采集类数据表建立行级触发器;历史数据管理主要针对移动终端设备存储空间有限,实现对移动数据库进行有效管理;同步数据传输管理主要是负责对移动终端的数据传输提供服务;同步数据操作管理主要实现数据同步过程中对数据的操作,包括 JSON 解析、JSON 生成、生成同步数据、操作同步数据、操作文件数据、Map 与 Status 数据操作和数据冲突处理;同步会话管理主要负责与服务器端建立同步会话连接,它包括数据解压缩、JSON 解析、发起数据同步 3 个功能;推送数据接收管理主要是负责接收处理服务器端推送过来的同步消息。

## 3 数据同步中间件的实现与测试

### 3.1 数据同步中间件的实现

本文实现的数据同步中间件包括服务器端和移动终端,这两块分别采取了不同的技术和工具进行开发。

在服务器端,以 Windows XP 操作系统作为平台,采用了.NET 技术,利用 C#开发语言,数据库采用 Miscrosoft 公司的 SQL Server 2008 作为中心数据库,所用的开发工具为微软的 Visual Studio 2010。

在移动终端,开发平台采用 Windows XP 操作系统,运行和调试的平台采用开源操作系统 Android 4.0,开发语言为 JAVA 语言,数据库采用开源嵌入式数据库 SQLite 3,开发工具为 Sun 公司的 Eclipse 3.0。移动终端调试和测试的硬件工具有联想 A789、Giant 等移动终端设备。

当实现服务器端推送技术时,使用极光公司的极光推送工具,在服务器端配置 C#版本的推送服务,调用推送 API,在移动终端配置 Android 版本的接收程序,监听服务器端推送消息。

#### 3.1.1 慢同步实现

本文对 Sync ML 协议中的慢同步做了改进并实现。通过移动终端控制发起慢同步,实现了大数据的分包传输、断网后再次同步时的断点续传等,具体实现流程如图 9 所示。在移动终端通过 SlowSync (String userName, String password, int order) 函数发起慢同步。

#### 3.1.2 一般同步实现

本文实现的一般同步也是在移动终端发起,主要是在移动终端接收到服务器端推送过来的同步通知和非首次登录的情况下发生,通过移动终端的函数 CommanSync(String username, String password)发起,实现流程如图 10 所示。

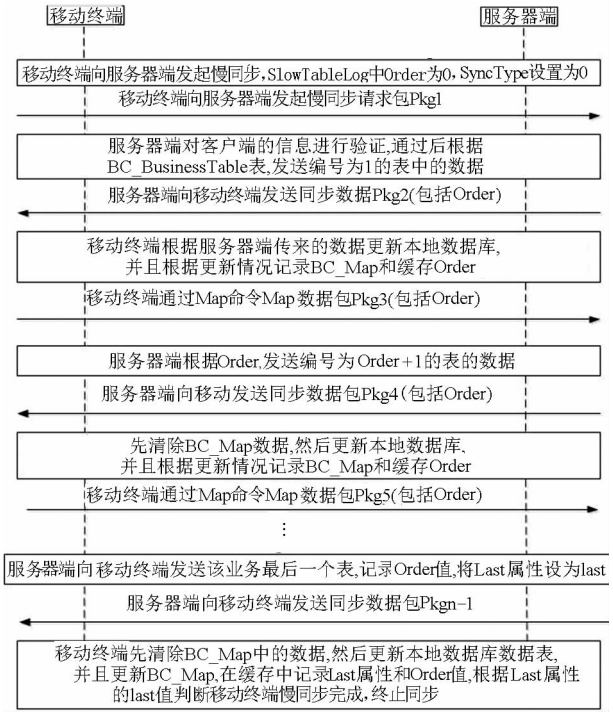


图 9 慢同步实现流程图

Fig. 9 Implementation flow of slow synchronization


图 10 一般同步实现流程图

Fig. 10 Implementation flow of general synchronization

#### 3.1.3 数据冲突管理

数据冲突管理[19],主要是对移动终端和服务器端数据进行修改或者删除时,对其所产生的冲突进行处理。在服务器端,对数据采集类信息进行修改或者删除时,会发生冲突,本文中间件做了相应的规则处理。首先判断上次是否删除掉该条数据,若已删除,则无法更新;若没有删除,则比较上次更新时间和本次要更新的时间,如果小于阈值 7 d,对于修

改操作，只有管理员、信息录入者、最后修改者可以修改，如果是删除操作，只有信息录入者可以删除本条数据，其他角色的用户无权删除本条数据，如果上次修改时间距离本次要修改的时间间隔大于时间阈值 7 d，则管理员、信息录入者、最后修改者都可以对本条数据做修改或者删除操作，但是除了这些角色外的其他用户，只有查看的权限，无权变更数据。在移动终端，本文对数据冲突处理比较简单。如果已经同步过到服务器端的数据，则服务器端同步过来的数据将替换移动终端的数据，如果是移动终端没有同步过的数据，则对服务器端同步过来的数据不做任何处理。

### 3.2　数据同步中间件的测试

本文在 Android 平台上实现了移动数据同步中间件，为了验证该中间件的实用性和普适性，将该中间件分别应用在 2 个不同的实际项目中，即种业科研育种田间信息采集系统（测试 1）和水稻定位试验打分系统（测试 2）。

#### 3.2.1　田间信息采集系统

利用智能手机到野外进行信息采集时，田间信息采集系统的数据同步部分需满足：服务器端一有新维护的信息采集指标数据，该数据快速准确地同步到移动终端；移动终端除首次登陆外，每次正常登陆时只下载服务器端新维护的数据，不能把所有数据都下载一遍；移动终端采集的数据有图片、视频等比较占存储空间的数据，因此需对移动终端采集类信息的历史数据进行管理，提高移动终端存储空间的利用效率。对于上述问题，中间件给出了解决方案：提供了主动推送策略，通过使用极光推送技术，当服务器端有变化数据，就推送同步消息到移动终端，使移动终端可以立即启动同步服务，获取最新数据；该中间件主要策略是在服务器端有专门的 BC_ChangeLog 数据变化日志表，记录变化的数据，表 BC_PrepareSync 记录该变化数据还有哪些设备还未同步。移动终端每次只同步服务器端中还没有同步的数据；此中间件提供了移动终端历史数据管理功能，解决了移动终端对采集类历史数据的管理问题，提高了存储空间利用率。

将中间件应用到该系统时，需要做如下配置和改动即可使用：在服务器端，对中心数据库中信息采集业务下的每个基础表和信息采集类表都必须针对 insert、update 和 delete 3 种操作建立操作后的触发器，用以捕获数据库中该业务的变化数据，同时，在表 BC_BusinessTable 中维护该业务的数据表，主要是维护各个数据表的表名和编号（信息采集类表的编号不可以设置成最后一个），用于实现慢同步。

该中间件应用到种业科研育种田间信息采集系统后，慢同步的运行效果如图 11 所示，初次运行会给用户友好提示：“该设备首次使用，需要同步大量数据，比较耗时且费流量，建议用 wifi 登陆，请耐心等待…”；系统登陆后自动向极光推送服务器注册确认，设备唯一标识注册成功后，给用户一个提示，如图 12 所示；因为一般同步在系统后台执行，用户看不到，所以在填写采集信息时，如果系统当前是跟服务器同步过的最近数据，则会给用户一提示说明，如图 13 所示。


图 11　慢同步效果图

Fig. 11　Effect drawing of slow synchronization


图 12　注册设备号图

Fig. 12　Registering device number

田间信息采集系统以前使用的同步策略是每次登陆时才能从服务器端获取新数据，而且所有数据都一起传输到移动终端，由移动终端根据移动数据库决定是否更新本地数据库，如果没有重新登陆的动作，就不会同步到服务器端的最新数据。将中间件系统应用到该系统后，通过慢同步的断点续传和一般同步只同步变化数据的方法，明显地提高了数据同步的效率，减少了无为数据的重复传输。同时，通过推送同步消息让移动终端同步变化数据的方式，极大地提高了服务器端新维护数据同步到移动终端的时效性。以同步 BS_Variety、BS_Pow_Crop 和 BS_Pow_


图 13 同步完成效果图

Fig. 13 Effect drawing of complete synchronization

Userl 3 张表的数据为例,使用中间件前后同步数据量的对比如图 14 所示,使用中间件后明显减少了无为数据的重复传输。图 15 表示使用中间件前后同步变化数据时效性,在 8 h 内每隔 3 min 向数据表 TG_KeyWord 增加 1 条数据,使用中间件后明显提高了数据同步的效率。

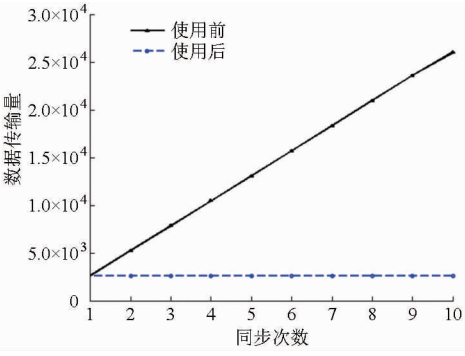


图 14 测试 1 传输数据量对比

Fig. 14 Contrast of transmitted data amounts (Test No. 1)

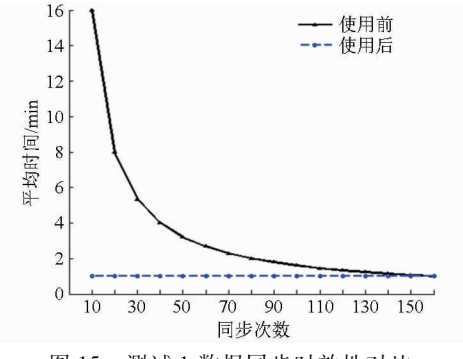


图 15 测试 1 数据同步时效性对比

Fig. 15 Contrast of data synchronization timeliness (Test No. 1)

### 3.2.2 水稻定位试验打分系统

利用智能手机到野外进行信息采集时,水稻定位试验打分系统的数据同步部分需满足:首次登陆失败后,再次登陆时接着上次同步失败的断点进行数据同步,不能重新从头同步数据;同一用户具有多台移动设备,每个设备都及时准确地同步到变化数据;对于同一条数据,被服务器端和不同移动设备识别为同一条数据;对于移动终端采集的水稻生长指标数据,由于不同指标是在不同时间采集的,为了防止同一条数据多次修改,多次上传,造成流量浪费,由用户主动控制移动终端的此类信息同步到服务器端。对于上述问题,中间件也给出了很好的解决方案:中间件实现的慢同步策略很好地解决了不能重新从头同步数据,而且还避免了不必要的慢同步的发生;中间件通过在服务器端记录用户所有设备向特定设备推送其相应同步消息的方式,解决了同一用户多台移动设备都能准确地同步到相关数据的问题;中间件通过改进的 Map 指令,给映射信息中增加表名和设备号,很好地解决了对于同一信息的识别问题;中间件提供了上传同步方式,让用户自己控制什么时候将采集的数据同步到服务器端,防止没有采集完的数据被同步到服务器端。

将中间件应用到该系统时,需要做如下配置:在服务器端给基础信息类数据表 BS_Pow_User、TG_TrialPoint_Code、TG_Variety_Trial、TG_TrialPoint、TG_BS_Authentication 和信息采集类数据表 TG_RiceDWTrialGrade 建立 insert、update 和 delete 的后触发触发器,用以捕获变化数据,同时将上面的数据表在表 BC_BusinessTable 中进行业务维护,并且对它们进行编号。在移动终端针对信息采集类数据表 TG_RiceDWTrialGrade 建立行级后触发触发器,用以捕获变化数据。与上文应用到信息采集系统类似,在移动终端需要配置接收服务器推送过来消息的程序,针对各个数据表编写操作它们的方法。

将中间件应用到水稻定位试验打分系统后,通过慢同步的断点续传和一般同步只同步最新变化数据,与使用前相比,同步的数据量明显减少,避免了无为数据同步,节省了网络流量。同时,由于服务器端主动推送同步消息,服务器端新维护的数据也能及时准确地同步到移动终端,极大地提高了新维护数据同步的时效性。用 TG_TrialPoint_Code、TG_Variety_Trial、G_TrialPoint 3 张表的数据同步为例,总共同步 10 次,每次同步时新增 5 条数据,得到如图 16 所示的折线图。

图 17 表示了使用中间件前后同步变化数据时效性的对比,在 8 h 内每隔 3 min 向数据表 TG_KeyWord 增加 1 条数据。

## 4 结束语

总结了移动信息采集的业务流程,提炼出此类

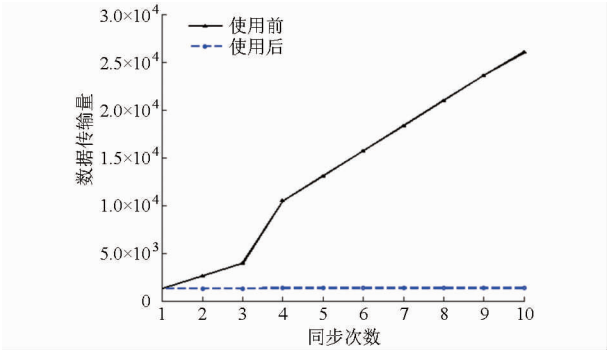


图 16　测试 2 传输数据量对比

Fig. 16　Contrast of transmitted data amounts (Test No. 2)

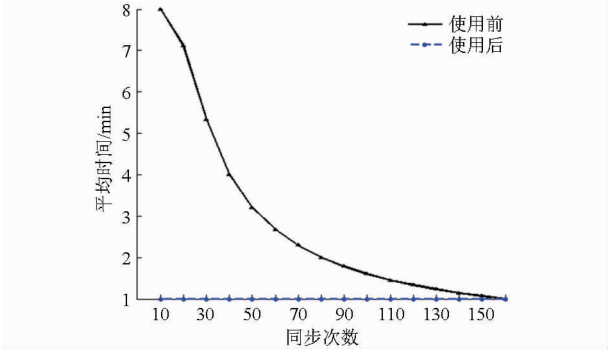


图 17　测试 2 数据同步时效性对比

Fig. 17　Contrast of data sync timeliness (Test No. 2)

应用中移动数据同步的需求，并在实际应用中遇到的数据传输同步问题进行研究的基础上，对 Sync ML 协议和移动数据同步策略进行改进和设计。把这些改进和设计实现的中间件应用到种业科研育种田间信息采集系统和水稻定位试验打分系统，测试并验证了该中间件实现的数据同步策略的完善性和普适性。比较了系统使用中间件前后的数据传输同步的效果。试验结果表明，中间件明显提高了数据同步的效率，减少了无为数据的重复传输，有利于田间信息采集系统在现代信息农业中的推广，提高了田间数据采集人员的工作效率和数据的准确率。本文方法具有较好的稳定性和实用性，使得数据传输量不随同步次数的增加而增加，并且在野外网络不稳定的情况下同样适用。

## 参　考　文　献


1　王虎，杨耀华，李绍明，等. 基于移动端作物大田测试数据采集技术研究与实现[J]. 中国农业科技导报，2013，15(4)：156－162.
Wang Hu, Yang Yaohua, Li Shaoming, et al. Research and implementation of field crop test data collection based on mobile phone [J]. Journal of Agricultural Science and Technology, 2013, 15(4): 156－162. (in Chinese)

2　黄锦，李绍明. 基于手机的玉米育种田间数据采集系统设计[J]. 农机化研究，2014，36(6)：193－197.
Huang Jin, Li Shaoming. Design on field information acquisition of maize breeding system based mobile phone[J]. Journal of Agricultural Mechanization Research, 2014, 36(6): 193－197. (in Chinese)

3　Han Y Q, Shan X, Zhu D, et al. Design and implementation of locust data collecting system based on Android[M]// Advances in Computer Science and Education Applications. Berlin Heidelberg: Springer, 2011,202: 328－337.

4　尚明华，秦磊磊，王风云，等. 基于 Android 智能手机的小麦生产风险信息采集系统[J]. 农业工程学报，2011，27(5)：178－182.
Shang Minghua, Qin Leilei, Wang Fengyun, et al. Information collection system of wheat production risk based on Android smartphone[J]. Transactions of the CSAE, 2011, 27(5): 178－182. (in Chinese)

5　叶思菁，朱德海，姚晓闯，等. 基于移动 GIS 的作物种植环境数据采集技术[J]. 农业机械学报，2015，46(9)：325－334.
Ye Sijing, Zhu Dehai, Yao Xiaochuang, et al. Mobile GIS based approach for collection of crop planting environment data[J]. Transactions of the Chinese Society for Agricultural Machinery, 2015, 46(9): 325－334. (in Chinese)

6　刘永华，沈明霞，熊迎军，等. 基于两级预测的温室 WSN 系统数据传输方法[J]. 农业机械学报，2014，45(12)：329－334.
Liu Yonghua, Shen Mingxia, Xiong Yingjun, et al. Data transmission of WSN system in greenhouse based on two-level prediction [J]. Transactions of the Chinese Society for Agricultural Machinery, 2014, 45(12): 329－334. (in Chinese)

7　李萍萍，陈美镇，王纪章，等. 温室物联网测控管理系统开发与数据同步研究[J]. 农业机械学报，2015，46(8)：224－231.
Li Pingping, Chen Meizhen, Wang Jizhang, et al. Development of monitoring management system and data synchronization for greenhouse IOT[J]. Transactions of the Chinese Society for Agricultural Machinery, 2015, 46(8): 224－231. (in Chinese)

8　Bernard G, Ben-Othman J, Bouganim L, et al. Mobile databases: a selection of open issues and research directions[J]. ACM Sigmod Record, 2004, 33(2): 78－83.

9　Byun S, Moon S. Resilient data management for replicated mobile database systems[J]. Data & Knowledge Engineering, 1999, 29(1): 43－55.

10　靖树峰，钟锡昌，张倪. 一种移动数据库系统的同步方案设计[J]. 计算机应用研究，2006，23(6)：193－195.
Jing Shufeng, Zhong Xichang, Zhang Ni. Designing of synchronization system in mobile DBMA[J]. Application Research of Computers, 2006, 23(6): 193－195. (in Chinese)

11　Lim J B, Hurson A R. Transaction processing in mobile, heterogeneous database systems[J]. IEEE Transactions on Knowledge and Data Engineering, 2002, 14(6): 1330－1346.

12　丁治明，孟小峰，王珊. 复制的移动数据库系统事务级同步处理策略[J]. 软件学报，2002，13(2)：258－265.

Ding Zhiming, Meng Xiaofeng, Wang Shan. A novel transactional synchronization scheme for replicated mobile database systems [J]. Journal of Software, 2002, 13(2): 258 - 265. (in Chinese)

13 丁治明, 孟小峰. 移动复制数据库系统冲突检测及消解策略[J]. 计算机学报, 2002, 25(3): 297 - 305.
Ding Zhiming, Meng Xiaofeng. Conflict detection and resolution strategy in replicated mobile database systems[J]. Chinese Journal of Computers, 2002, 25(3): 297 - 305. (in Chinese)

14 Ye S, Zhu D, Yao X, et al. Development of a highly flexible mobile GIS-based system for collecting arable land quality data[J]. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2014, 7(11): 4432 - 4441.

15 孟志军, 王秀, 赵春江, 等. 基于嵌入式组件技术的精准农业农田信息采集系统的设计与实现[J]. 农业工程学报, 2005, 21(4): 91 - 96.
Meng Zhijun, Wang Xiu, Zhao Chunjiang, et al. Development of field information collection system based on embedded COM-GIS and pocket PC for precision agriculture[J]. Transactions of the CSAE, 2005, 21(4): 91 - 96. (in Chinese)

16 李欣慧, 侯紫峰, 贺志强. 基于 SyncML 协议的异构数据源同步方法[J]. 计算机应用研究, 2006, 23(6): 190 - 192, 195.
Li Xinhui, Hou Zifeng, He Zhiqiang. Heterogeneous data source synchronization with SyncML[J]. Application Research of Computers, 2006, 23(6): 190 - 192, 195. (in Chinese)

17 徐福亮, 周祖德. 变化数据捕获技术研究[J]. 武汉理工大学学报, 2009, 31(5): 740 - 743.
Xu Fuliang, Zhou Zude. Research on change-data-capture technology[J]. Journal of Wuhan University of Technology, 2009, 31(5): 740 - 743. (in Chinese)

18 Huyan Zhengyong, Xu Likui, Fang Shuai, et al. Field information acquisition system research based on offline speech recognition [J]. International Journal of Database Theory and Application, 2014, 7(2): 45 - 58.