Android 自动化测试系统的设计与实现

王自豪，郭燕慧
（北京邮电大学计算机学院，北京 100876）

摘要：Android 自动化测试在当前移动互联网快速发展的背景下正逐渐兴起。现有的 Android 自动化测试方法普遍需要 PC 端配合，无法在 Android 终端独立进行自动化测试，并且不能实现多终端多任务并行执行。针对这些问题，本文设计实现了一个 Android 自动化测试系统，通过基于 Servlet3 异步特性实现 http 长连接的方式取代与 PC 端物理连接，保证实时推送任务消息到 Android 终端，并且通过 Android 系统的 getevent 和 input 工具基于事件流分析与事件注入的方法直接在 Android 终端实现脚本的录制回放，实现多终端多任务并行执行。

关键词：自动化测试；Android; 长连接；脚本录制回放
中图分类号：TP311

Design and Implementation of Android Automated Test System

WANG Zihao, Guo Yanhui
(Computer Science School, Beijing University of Posts and Telecommunications, Beijing 100876)

Abstract: With the rapid development of mobile Internet,Android automated test is gradually on the rise.The existing automate test methods generally require the PC side to carry out and Android device can not do the test independently.Besides,the existing methods can not achieve multi-device to perform multiple tasks in parallel.To solve these problems, this paper designed and implemented an Android automated test system, using http long connection based on the asynchronous nature of Servlet3 to replace PC physical connection to ensure to push task message to the Android device real-time.In additon,by Android system tools getevent and input,this system achieve to record and playback scripts directly on the Android device based on the method of event flow analysis and event injection.Therefore,the system can achieve multi-device to perform multiple tasks in parallel.

Key words: automated test;Android;long connection; script recording and playback

0 引言

随着移动互联网的迅猛发展，智能手机、平板电脑等移动应用终端越来越普及，相应的 APP 应用数量也呈快速增长趋势。Android 系统是由 Google 推出的一款移动操作系统，由于其高度开放性、可用性，占据了当前移动智能终端市场的最大份额。根据应用数据追踪公司 AppFigures 的统计数据显示[1]，2014 年 GooglePlay Store 的 Android 应用总量已经达到 143 万款，Google Play Store 的开发者总数为 38.8 万。面对 Android 应用不断推陈出新，在数量上不断增长的现状，如何保证质量也成了一个越来越关键的问题。通过对 Android 应用的进行大量、广泛、重复的测试成了解决这个问题的主要方法。Android 应用的自动化测试应运而生，取代了传统低效率的人工测试方法，用来检验 Android 应用的兼容性、稳定性等问题[2]。

当前已有不少 Android 自动化测试框架[3]，比如 Monkeyrunner、Sikuli 等，但测试过程基本需要 Android 终端通过 USB 接口连接 PC 端之后，由 PC 端负责脚本录制[4]、启动测试

作者简介：王自豪（1990-），男，硕士研究生，主要研究方向：移动互联网技术
通信联系人：郭燕慧（1974-），女，副教授，主要研究方向：移动互联网安全. E-mail: yhguo@bupt.edu.cn

等关键步骤。这种方式下，测试环境容易受到物理条件限制，符合测试条件的 PC 需要部署一定的开发测试环境，对测试环境要求较高。另外，由于 PC 端物理接口数量限制等因素，无法满足对多台终端多个应用同时进行测试的需求，效率相对较低。本文通过对 Android 自动化测试相关技术的研究，设计实现一个 Android 自动化测试系统，通过基于 Servlet3 异步特性实现 http 长连接[5]，以消息推送[6]的方式将测试任务下发到 Android 终端，通过事件流分析与事件注入方式实现脚本的录制回放，实现 Android 终端的独立测试，并使得系统支持多终端多应用的同时测试，提高 Android 自动化测试的效率。

1 相关技术研究

1.1 长连接技术

普通的连接在通讯双方完成数据交互后，会断开连接，而长连接则会一直保持连接。因此，长连接下的通讯双方可以在需要的时候继续实现数据交互，尤其对于服务端需要主动推送消息到客户端的情形特别适用。

Servlet3 开始支持异步处理，服务端在接收到客户端的连接请求后，Servlet 线程不再是一直处于阻塞状态以等待业务逻辑的处理，而是将请求转移到一个异步线程来处理，并生成对应的异步上下文实例 AsyncContext，其中包含了 ServletRequest 和 SerlvetResponse 对象的引用。当服务端需要向客户端返回数据时，可以从 AsyncContent 中获取 ServletResponse 对象来下发消息数据。只要 AsyncContent 一直保存着，就可以一直复用其中的 ServletResponse 对象，服务端也就可以多次向客户端进行消息数据的推送，即实现了 http 长连接。

在 Android 自动化测试过程中，Android 终端通过 USB 数据线的方式与 PC 端建立物理连接的目的是为了接收外部的测试请求命令，从而保证 Android 终端能够执行自动化测试任务，其功能实现可以通过长连接来替代。Android 终端主动发起 http 请求，服务端接收到请求后基于 Servlet3 异步特性建立 http 长连接。当服务端有测试任务需要下发时，再通过建立的长连接推送到 Android 终端。相比于 USB 数据线连接的方式，这种方式解决了物理连接的弊端，能够支持多终端多任务的并行测试，可以明显提升测试效率。

1.2 脚本录制回放技术

通过脚本进行录制回放是 Android 自动化测试中常用的方法，其基本流程为将人工操作的行为动作记录生成脚本文件，在执行回放过程时，读取文件中的内容，模拟实现脚本中记录的事件。

目前 Android 自动化测试框架的脚本录制回放基本是在 PC 端配合实现的，不能直接在 Android 终端上进行录制生成，灵活性欠佳。因此有必要考虑在 Android 终端直接实现脚本录制回放功能。通过对 Android 系统的研究，发现 Android 自带的 getevent 工具可以获取到 Android 系统中当前输入设备的一些参数和实时事件数据。据此，本文提出一种基于事件流分析的脚本录制方法，通过 Android 系统下的 getevent 工具实现在 Android 终端直接完成脚本录制生成功能。该方法主要通过 getevent 工具监控屏幕点击以及按键的事件流，获取事件流属性并进行相关转换和分析，并将分析后的事件流信息以约定形式写入脚本。

首先，需要进行事件采集，事件采集分为屏幕事件采集和按键事件采集两种。屏幕事件采集主要是指采集屏幕的点击、拖拽、滑动等操作，按键事件采集主要是指采集 Android 终端上的音量、电源等按键操作。通过 getevent -l -t 命令可以获取到格式化后的事件信息，举例如下：

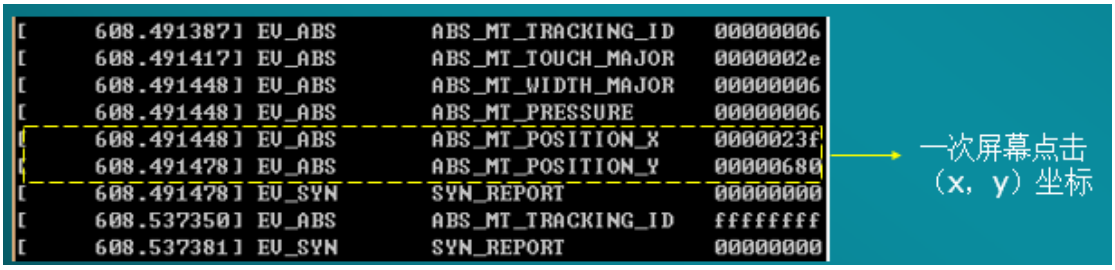

图 1 屏幕事件样图

Fig.1 screen event sample

图 1 表示一次屏幕点击操作过程中 getevent 工具采集到的事件流，其中 ABS_MT_POSITION_X 表示屏幕点击的 x 坐标，ABS_MT_POSITION_Y 表示屏幕点击的 y 坐标，后面是对应的坐标值的十六进制表示形式。

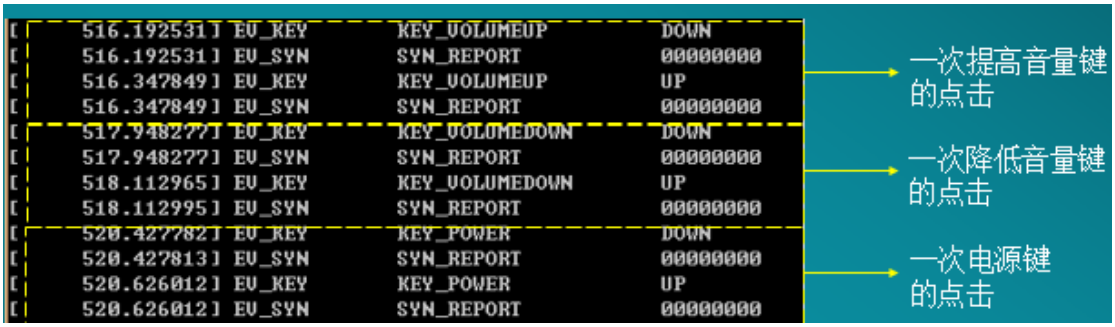

图 2 按键事件样图

Fig.2 key event sample

图 2 表示一次降低音量键、一次提高音量键和一次电源键这三次按键操作过程中 getevent 工具采集到的事件流，其中的 KEY_VOLUMEUP 表示提高音量键操作，KEY_VOLUMEDOWN 表示降低音量键操作，KEY_POWER 表示电源键的按键操作。

通过事件采集部分中的结果，可以看出 getevent 记录的事件流具有一定的格式，但是还有必要对每一列值进行分析处理。每一次事件发生，getevent 都会记录包括时间戳、事件类型、事件编码和事件键值等信息。其中上图中的第一列表示时间戳，第二列表示事件类型，事件类型主要有 EV_SYN（同步事件）、EV_KEY（按键事件）和 EV_ABS（绝对值事件）；第三列包括按键编码和绝对值编码，按键编码主要有 KEY_HOME（HOME 键）、KEY_MENU（MENU 键）、KEY_BACK（BACK 键）和 KEY_SEARCH（SEARCH 键），绝对值编码主要有 ABS_MT_TOUCH_MAJOR（接触面长轴值）、ABS_MT_POSITION_X（接触点 x 轴坐标）和 ABS_MT_POSITION_Y（接触点 y 轴坐标）；第四列表示对应键值，其中十六进制数字表示的是屏幕事件的坐标，其他表示按键操作事件。将这些事件整理后如表 1。

表 1 事件整理表

Tab.1 the table of events

事件类型	*按键编码*	*含义*
同步事件	EV_SYN	同步事件
按键事件	KEY_HOME	HOME 键
	KEY_MENU	MENU 键
	KEY_BACK	BACK 键
	KEY_SEARCH	SEARCH 键
绝对值事件	ABS_MT_TOUCH_MAJOR	接触面长轴值
	BS	

ABS_MT_POSITION_X	接触点 X 轴坐标
ABS_MT_POSITION_Y	接触点 Y 轴坐标
ABS_MT_TRACKING_ID	接触的跟踪 ID，用来分辨多个同时操作

回放过程其实就是事件模拟过程，需要通过事件注入的方法来实现。通过 Android 的 input 工具可以实现事件注入，完成模拟触屏、按键等操作。Android 的 input 工具可以用来向设备发送模拟操作的命令。使用方法如下：

```
usage:input text <string>
      input keyevent <key code number or name>
      input tap <x><y>
      input swipe <x1><y1><x2><y2>
```

从上面的使用说明可以知道 input 主要分为四类操作，text、keyevent、tap 和 swipe，具体含义如表 2。

表 2 input 事件表

Tab.2 the table of input event

事件类型	*含义*
text	发送文字内容
keyevent	模拟按键事件
tap	模拟点击操作
swipe	模拟滑动操作

比如模拟点击屏幕上某一点时，只需使用 input tap x y 这一命令来实现，x 的值表示屏幕点的 x 坐标，y 的值表示 y 坐标。模拟滑动操作时，使用 input swipe x1 y1 x2 y2，其中 x1、y1 表示滑动起始点的坐标，x2、y2 表示滑动结束点的坐标。比如在应用的安装卸载过程中，往往还需要点击屏幕中的“取消/安装”、“完成/打开”等控件。这些动作就可以利用 input tab 命令来实现，根据获取的对应坐标，执行模拟点击操作，从而实现应用安装卸载的自动化。

2 系统设计与实现

2.1 总体架构

系统架构如图 3 所示，主要由 Web 访问服务器、数据服务器、任务调度服务器和 Android 终端组成。

Web 访问服务器主要用于用户访问，提供任务提交功能，是 Android 自动化测试的入口；数据服务器用于存储 Android 终端自动化测试需要的资源文件数据、结果数据以及数据库的维护；任务调度服务器用于任务的分配调度，直接与 Android 终端通过 http 请求进行通信。

客户端部分为 Android 终端设备，是自动化测试的执行主体，包括实现应用自动化安装、运行、卸载过程和模拟触屏、按键操作等。这些也是 Android 自动化测试需要完成的必要操作。

系统运行流程主要如下步骤：

1） 用户访问 Web 服务器，选择系统已有的应用提交测试任务。对于系统没有的应用，用户也可以上传应用至系统，再提交测试任务。新上传的应用数据会同步到数据服务器。

2） 任务调度服务器接收到 Web 服务器的任务请求，根据系统的调度算法，选择合适的 Android 终端进行任务下发。

3） Android 终端接收到测试任务，进行自动化测试，执行包括下载、安装、运行和卸载应用这一整个流程，并将结果数据打包上传回调度服务器。

4） 调度服务器接收 Android 终端的测试结果数据分析处理后，将结果数据转移到数据服务器，并更新数据服务器中数据库信息。

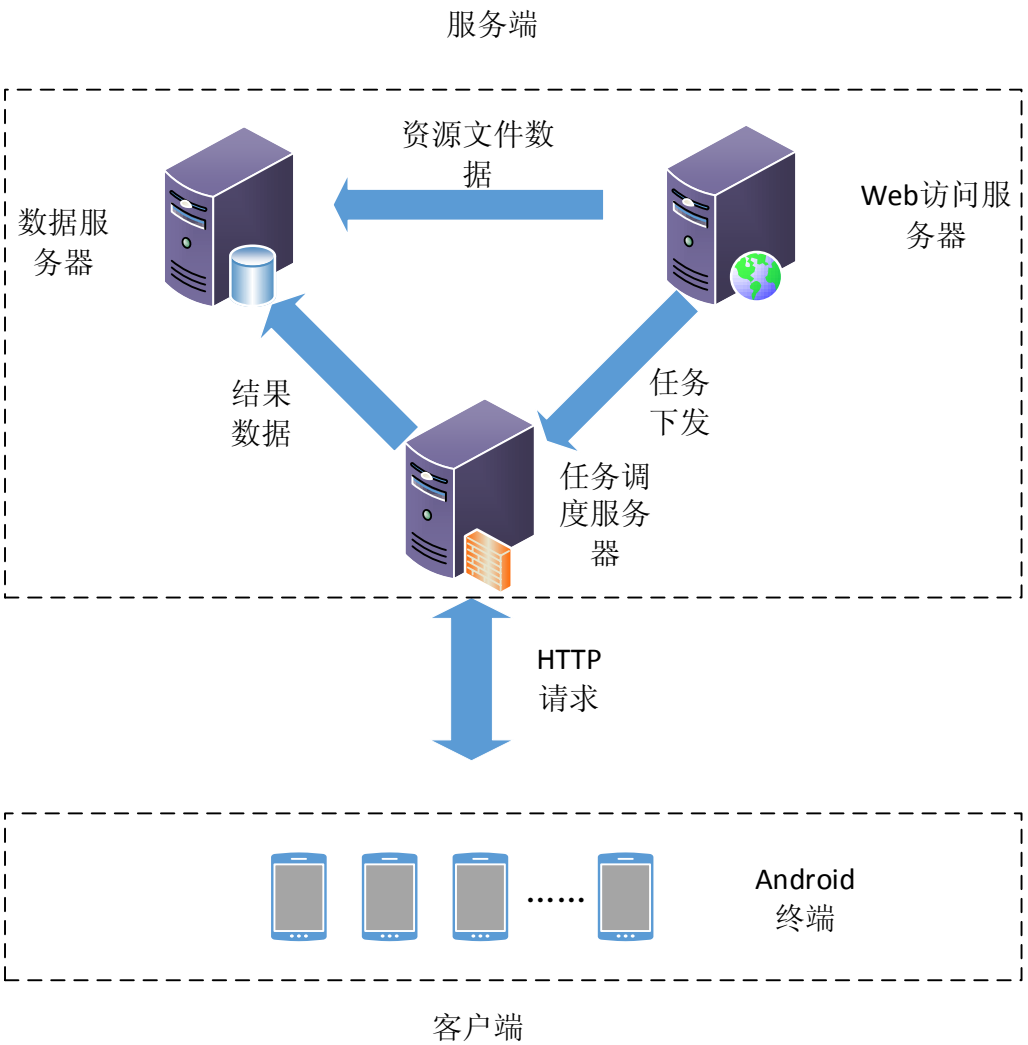

图 3 系统架构图

Fig.3 the structure of system

2.2 详细设计

系统从功能模块上主要可以分成终端管理模块、任务管理模块、终端执行模块、结果处理模块和数据存储模块，如图 4。

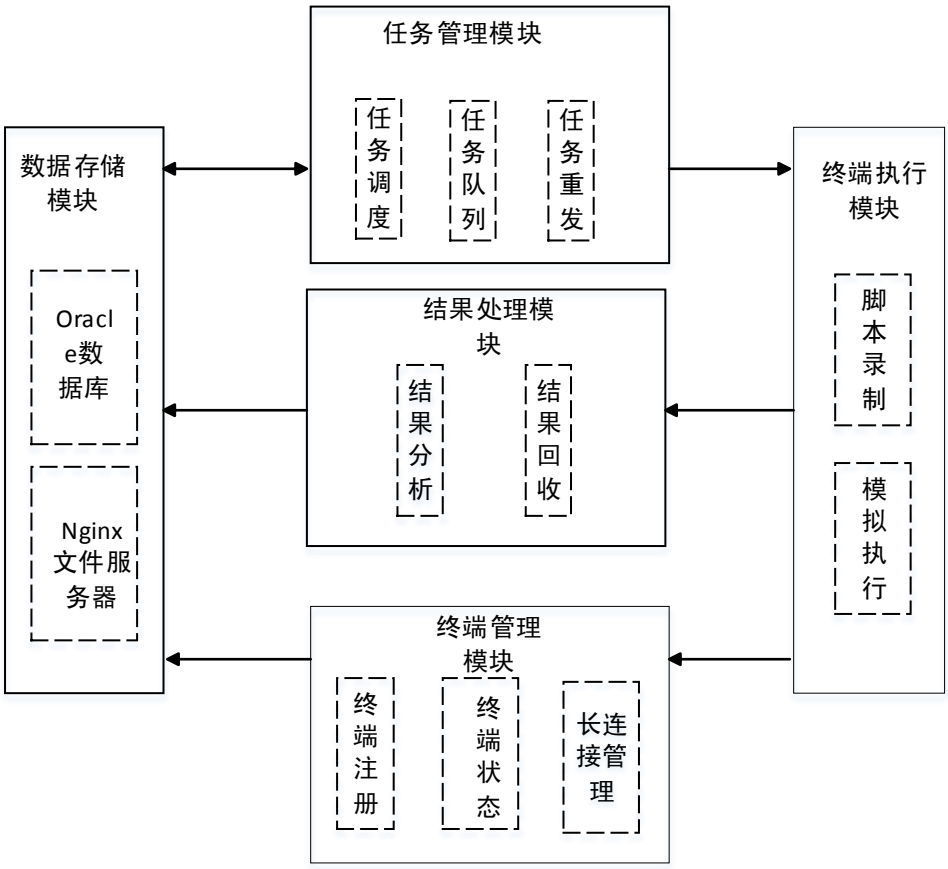

图 4 功能模块图

Fig.4 functional modules

2.2.1 终端管理模块

终端管理模块是维护服务端与 Android 终端通信的关键模块，主要实现终端注册、Http 长连接管理和终端状态管理功能。

1）终端注册

Android 终端若要接入系统进行测试，首先需要在服务端进行注册。终端注册子模块用来接收终端提交的基本信息，包括终端号码、机型、分辨率、系统版本、运营商信息等。服务端在接收终端信息后会生成唯一识别的终端 ID 返回给注册终端，用于后续终端的鉴别。注册时，Android 终端会将基本信息以 JSON 格式提交到服务端，服务端解析成功后返回终端 ID，并将终端信息保存到数据库中，完成终端注册流程。

终端注册信息 JSON 格式例子如下：

```
{
    'deviceNumber':'15110002345',
    'deviceArea':'Beijing',
    'deviceImei':'123456789012345',
    'deviceResolution':'960x480',
    'deviceOsVersion':'4.4.2',
    'deviceCarrier': 'YD',
    'phoneModel': 'Google Nexus4'
}
```

其中的参数意义如表 3。

表 3 终端注册信息表

Tab.3 device register information

参数	*含义*
deviceNumber	终端号码
deviceArea	终端地域
deviceImei	终端 IMEI
deviceResolution	终端分辨率
deviceOsVersion	终端系统版本
deviceCarrier	终端运营商
phoneModel	终端机型

2）长连接管理

Android 自动化测试中需要服务端向 Android 终端下发测试任务，Android 终端根据接收的任务信息获取相应的测试资源执行自动化测试。因此，Android 终端与服务端之间需要建立连接，提供通信的通道。普通的 http 连接为短连接，即一次请求一次响应后就断开连接，下次需要重新建立连接。这样频繁创建销毁连接对服务器会造成较大负载和资源浪费。而 Http 长连接只需要创建一次，之后服务端可以向客户端持续发送消息，因此基于 Servlet3 的异步特性实现 http 长连接。这样的连接会持久保持，每一个 Android 终端都会有各自的连接。为了保证连接正确，需要对这些 Http 长连接进行管理，确保服务端的任务消息发送到正确的 Android 终端。服务端使用 Java 中 key-value 形式的数据结构 Map 来维护连接，将终端注册获取的终端 ID 作为唯一识别码对应 Map 结构中的 key 值，将终端与服务端对应的 Http 长连接对象作为 Map 结构中的 value 值。每次服务端接收到终端连接请求，会根据终端 ID 查看 Map 结构中是否已经存在连接，防止重复连接。如果连接存在，则更新为新的连接，释放原有连接资源；若不存在，说明是第一次建立连接，将新连接加入到 Map 中。同样的，在有测试任务下发时再从中获取对应的连接。

3）终端状态管理

终端状态管理是指服务端根据 Android 终端向服务端定时发送消息的心跳机制，动态更新服务端记录的终端状态。终端状态主要有空闲、工作、离线三种状态，分别表示终端在线并且没有在执行测试任务、终端正在执行测试任务、终端未上线，如表 4。服务端维护一个 key-value 形式的 Map 结构用来保存终端的心跳信息，其中将终端 ID 作为 key 值，终端自定义对象作为 value 值。自定义对象中包含两个属性，终端状态值和上次心跳接收时间。每次接收到 Android 终端发送的心跳信息时，都会根据终端 ID 查询 Map 结构更新上次心跳接收时间为当前时间，同时比较终端状态值是否发生了变化。若状态发生变化，则更新为新的状态值，这一过程如图 5。此外，还有一个定期扫描 Map 结构的线程，将当前时间与终端的上次状态接收时间比较，若大于一定阈值，将终端状态更新为离线。终端状态管理能够保证有效获取终端的实际状态，方便任务调度过程中的终端选择。

表 4 终端状态表

Tab.4 the table of device status

终端状态表	*含义*
0	离线，终端未上线
1	空闲，终端当前无任务
2	工作，终端当前正在执行

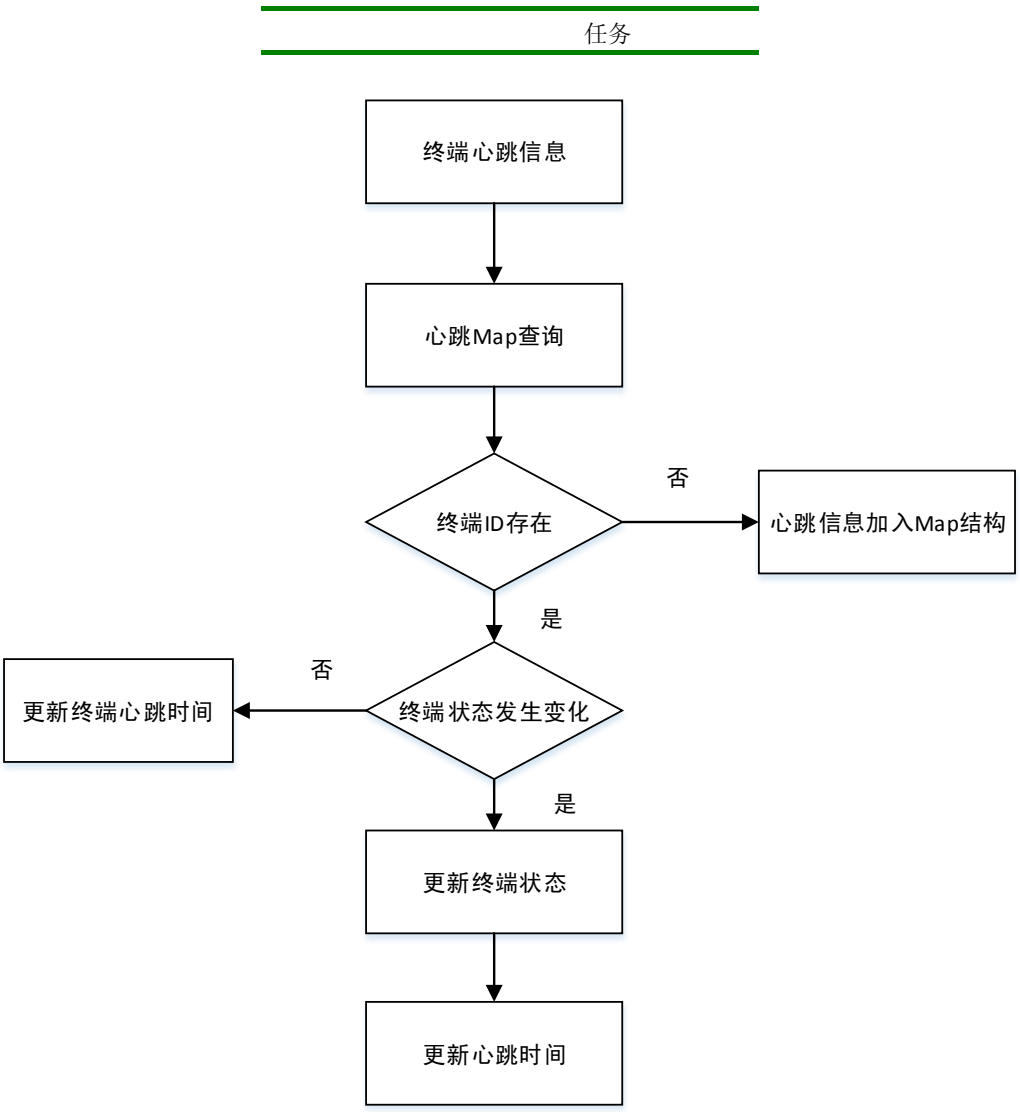

图 5 心跳更新流程

Fig.5 the process of head-beat update

2.2.2 任务管理模块

任务管理模块主要负责任务调度、任务队列和任务重发。该模块负责了测试任务分发，保证任务与测试终端的对应关系，任务优先级的动态变化和失败任务的重发处理。

1）任务调度

任务调度主要负责维护 Android 自动化测试设备资源池，在接收到任务请求后，根据调度算法，选择合适的测试设备进行任务下发。系统中自动化测试执行主体是 Android 终端，每一个 Android 终端由于其自身配置差异，性能上有所区别，对应的在执行测试任务时间上会有不同。系统在实际使用过程中，往往需要同时对一大批 Android 应用进行测试。任务调度模块会将每一个测试任务下发到具体的 Android 设备终端，在设备终端的选取过程中需要考虑作为任务执行节点的 Android 终端的自身能力与负载情况，防止性能较差的 Android 终端分配到耗时较高的任务导致任务整体执行时间偏大，即出现“拖后腿”现象，或者任务分配不均，部分 Android 终端任务量较重，部分 Android 终端任务量较小，造成资源浪费和系统效率低下。

本文结合动态任务调度[7]算法思想，设计了一种基于动态评估与负载预测的任务调度算

法。根据节点的任务执行情况动态评估 Android 终端节点能力，并综合 Android 终端节点的实际负载情况和任务执行进度进行预测，合理选择任务的执行节点，从而缩短任务的整体完成时间。

在对 Android 终端节点进行评估时，需要通过统计 Android 终端节点已经完成的任务的执行时间和任务数，计算平均任务完成时间，再与两次评估之间完成的任务的平均任务完成时间相比较，获得其变化值λ ，公式如下：

$$\lambda = \frac{\frac{\sum_{i=1}^{n} T_i}{n}}{\frac{\sum_{j=1}^{m} T_j}{m}} \tag{1}$$

其中 n 为完成的任务总数，T_i 为单个任务完成所用时间，m 为两次评估之间完成的任务数，T_j 为两次评估之间完成的单个任务所用时间。新的评估值计算公式如下：

$$S = \lambda S' = \lambda \frac{\frac{\sum_{i=1}^{n} T_i}{n}}{\frac{\sum_{j=1}^{m} T_j}{m}} S' \tag{2}$$

其中的 S' 表示的原来的评估值。

负载预测时需要借助于 Android 终端节点自身任务完成情况的统计数据，通过分析当前 Android 终端节点上的任务数和各个任务已经执行的时间，与 Android 终端节点任务完成任务平均时间来比较。当前各个任务已经执行时间之和，与平均任务时间、任务数的乘积越接近，越说明快有任务要完成，即预测负载将要下降。Android 终端节点负载下降平均等待时间公式如下：

$$T_w = \frac{n\bar{T} - \sum_{i=1}^{n} T_i}{n} = \bar{T} - \frac{1}{n}\sum_{i=1}^{n} T_i \tag{3}$$

其中的 T_w 表示 Android 终端节点任务的平均等待时间，$\bar{T}$ 表示统计的任务平均完成时间，n 表示节点当前的任务数量。

Android 终端节点选择是本调度算法中的关键部分，即当任务来临后，根据 Android 终端节点的任务执行能力和负载预测情况综合考虑，选择合适 Android 终端节点作为任务的执行者。其中主要目的是使任务下发后的等待时间和任务执行所使用的时间之和尽可能小。对于负载有剩余或者空闲的 Android 终端节点，时间为任务在该 Android 终端节点上执行所使用的时间，即只需要考虑该节点的任务执行能力；对于负载较高的节点，时间为 Android 终端节点负载下降所用的时间和任务执行所需要的时间，这里负载下降时间即为上文中的 T_w。计算 Android 终端节点选择的结果参考值公式如下：

$$R = \frac{k_1}{S} + k_2 T_w \tag{4}$$

其中 k_1 为 Android 终端节点评估值部分的系数，用来描述任务量情况；k_2 表示节点任务平均等待时间的系数，评估值 S 越大，等待时间 T_w 越小，Android 终端节点的结果参考值越小，表示任务预期完成所用时间越少。根据符合条件的 Android 终端节点的计算结果参考值进行排序，选择结果值最小的作为任务执行节点。

2）任务队列

任务下发后首先会进入任务队列，因此需要对任务队列进行管理，其中主要是针对任务优先级进行管理。每一个任务都有各自的优先级，用来区分任务的重要程度与紧急程度。高优先级任务进入队列会比原先队列中的低优先级任务具有更高的任务下发可能性。因此，当高优先级任务持续进入队列，低优先级任务很有可能会无法获得任务下发的可能性，出现“饥饿”现象。为解决这一问题，系统采用动态优先级策略，任务优先级不再取决于初始优先级，而是根据初始优先级和睡眠时间共同决定。这里定义睡眠时间为任务下发后所经过的时间，这样低优先级任务在无法改变初始优先级的情况下，随着睡眠时间增长优先级也会相应增加，获得任务下发的机会。定义任务实际优先级公式如下：

$$Priority = Min(10, InitPriority + sleepTime * 0.5) \tag{5}$$

其中 Priority 即为任务实际优先级，InitPriority 为初始优先级，sleepTime 为睡眠时间，单位为小时。系统定义优先级区间为[0,10]，即最高优先级为 10，最低优先级为 0。因此上述公式中取动态优先级结果和最高优先级 10 这两数的最小值作为实际优先级，这样可以避免优先级无限增大。对于优先级相同任务，优先下发提交时间较早的任务。

3）任务重发

任务重发是指任务失败后，系统重新下发任务到 Android 终端进行测试。任务失败原因可分为以下三种：

1.网络通信异常，任务未能成功下发到指定 Android 终端

2.Android 终端设备自身问题导致该任务无法执行成功

3.测试任务本身存在问题，终端无法执行

对于这三种情况，对应的任务重发机制也会有所区别。

对于第一种情况，网络通信异常导致的任务下发失败。具体表现为任务下发后，Android 终端状态没有发生变化，通常情况下终端接收到任务后终端状态会由空闲变为工作状态，没有发生该状态变化说明终端未收到任务。对此，系统会选定原来终端进行任务重发。如果第二次仍旧同样原因失败，说明该 Android 测试终端连接故障，不能接收测试任务，做相应标记后，选择另一随机终端下发测试任务。

对于第二种情况，Android 终端设备自身无法执行任务导致，具体表现为终端状态变化为空闲—工作—离线或者空闲—工作—空闲并且没有上传结果数据。此时，系统会选定另一相同机型配置 Android 终端进行任务重发，若同样失败，说明可能是该机型原因导致，再选定另一机型终端进行任务测试。如果结果仍是失败，即第三种情况，该测试任务本身存在问题，终端无法执行成功。

通常，在任务重发的整个过程中，同一终端同一任务最多执行两次。任务重发处理的这几种情况可以归纳为如图 6。

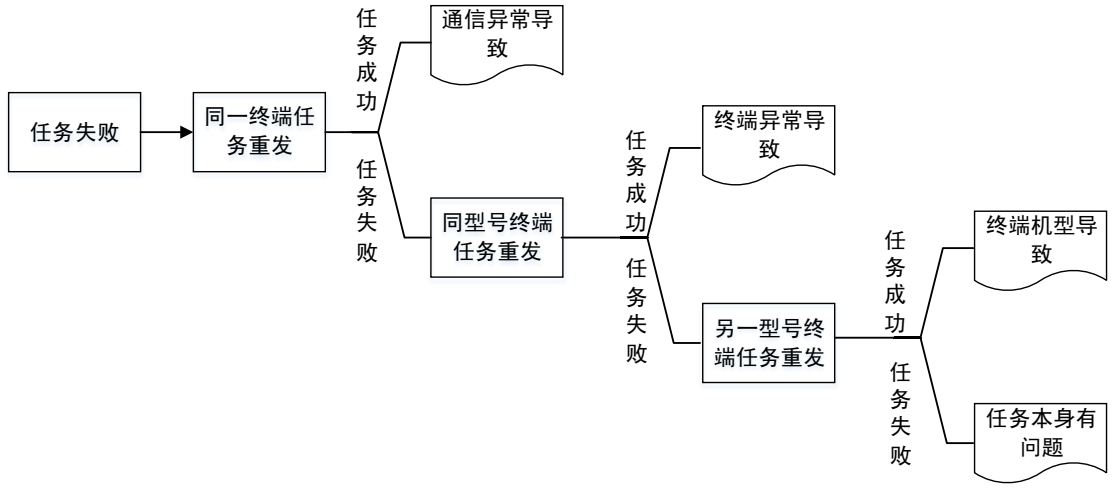

图 6 任务重发分析图

Fig.6 the analysis of task resend

2.2.3 终端执行模块

终端模拟执行模块主要实现脚本录制和事件的模拟执行功能。脚本录制主要通过 getevent 工具对事件流监控分析来生成对应的脚本文件，模拟触屏、按键等操作则是通过 Android 的 input 工具向设备发送模拟操作的命令方式实现事件注入。这一模块其实主要完成脚本的录制回放功能，利用 getevent 实现脚本录制，利用 input 模拟事件执行，达到回放目的。回放过程指根据事先录制好的脚本进行操作的回放，重现录制过程。基本流程是：按照脚本中记录的顺序，读取脚本语句，根据脚本语句构造对应的模拟事件，使程序按脚本中的命令运行。这里的事件回放就需要使用 input 工具来实现，根据脚本中语句动作的分析，转换成 Android 系统可识别的事件，再利用 input 工具将事件模拟重现，从而达到 Android 自动化测试的目的，如图 7。

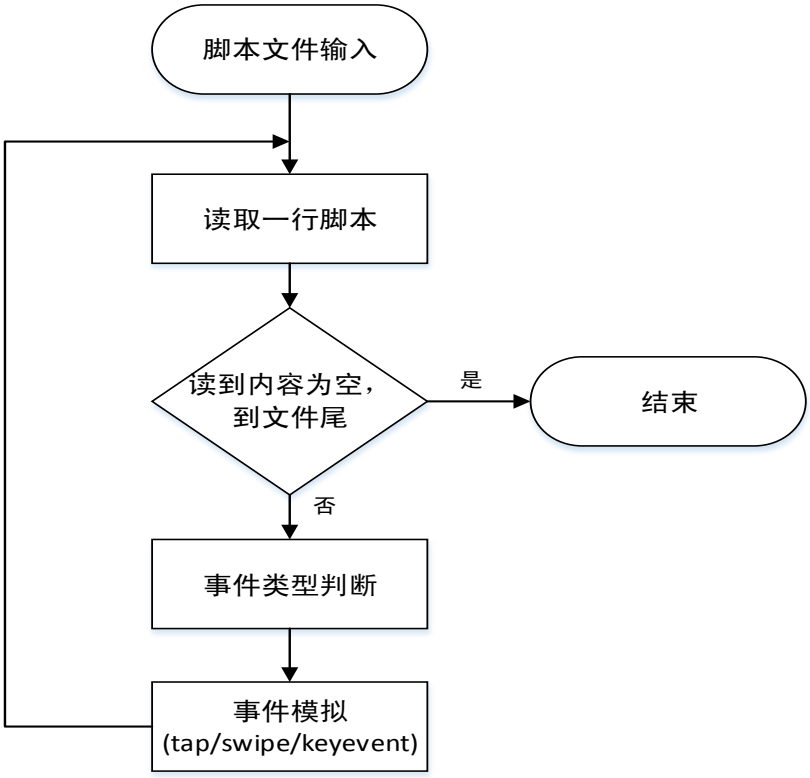

图 7 脚本回放流程图

Fig.7 the process of script playback

2.2.4 结果处理模块

结果处理模块负责对 Android 自动化测试任务结果数据的接收、处理和分析，是整个系统流程的最后关键部分，主要分为结果文件接收、结果文件解压、结果文件分析和结果文件保存这么几个步骤，如图 8。

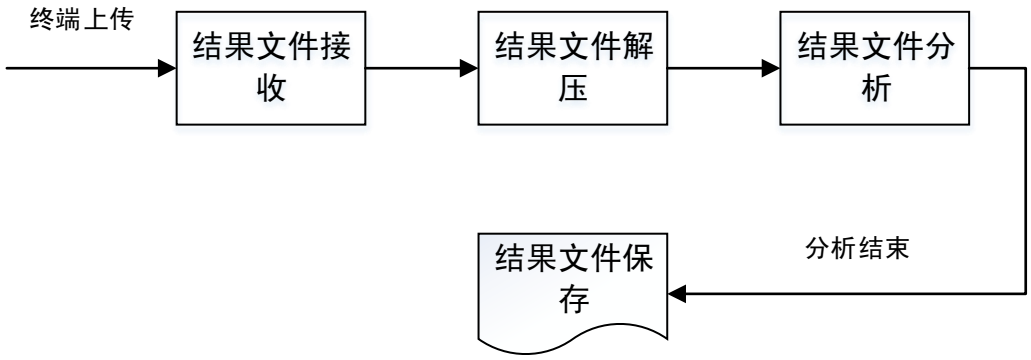

图 8 结果处理流程图

Fig.8 results processing

Android 终端在执行完测试任务后会将结果数据打成压缩包以 zip 文件形式上传到服务端，结果处理模块接收到该文件后执行解压操作。结果文件主要为记录任务信息的文本文件和测试过程中的屏幕截图。结果文件分析过程中从文本文件读取任务 ID、测试的应用名、应用版本、测试终端 ID 以及测试量等相关任务信息，同时对文件中的屏幕截图与预置的图片通过相似度对比的方式判定测试流程是否正确。

2.2.5 数据存储模块

数据存储模块主要负责数据库维护和文件存储与下载服务。系统采用关系型数据库 Oracle 对系统数据信息进行存储，并通过 Java 持久层框架 MyBatis 来进行增删查改操作，同时采用 Nginx 文件服务器提供文件访问下载服务。

数据库中涉及的比较重要的表有终端信息表、任务信息表、应用信息表、脚本资源表和图片资源表。终端信息表主要保存 Android 终端的机型、系统版本、分辨率等基本信息；任务信息表主要保存待测试应用信息、测试终端信息、任务起止时间以及任务量等；应用信息表主要保存应用名、应用版本、存储链接等信息；脚本资源表、图片资源表主要保存关联的应用信息和存储链接。

数据存储模块还需要负责系统结果文件的存储、资源文件的对外下载。资源文件主要包括应用文件、脚本文件和图片文件，需要提供给 Android 终端访问下载，作为 Android 自动化测试必要的资源。资源文件存储部分采用 Nginx 作为文件服务器，开启目录浏览功能来提供 http 文件下载服务。配置文件 nginx.conf 主要涉及的部分内容如下：

```
……
server{
        listen 80;
    server_name 192.168.100.78;
    location / {
    root /share/;
    autoindex on;
    autoindex_exact_size off;
```

```
        autoindex_localtime on;
            }
}
……
```

这里的 server_name 绑定了服务器的 ip 地址，root 后面配置的是允许文件目录浏览的根目录，autoindex on 表示打开了目录浏览功能。

3 实验分析

本节主要是针对 Android 自动化测试系统进行实验测试，验证系统的可用性与性能。系统环境的硬件配置为：3 台同配置 PC 服务器，CPU：Intel i3 3.5GHz；硬盘：500G SATA；内存：4GB RAM，分别作为 Web 访问服务器、数据服务器和任务调度服务器，以及 10 台 Android 终端测试机。

实验过程中通过每次下发一定量的任务，统计任务执行情况数据。实验共进行 4 批次任务，任务量分别为 50，100，150 和 200，结果统计数据如表 5。

表 5 结果统计表

Tab.5 results

任务量/个	*任务到达率*	*任务成功率*	任务重发次数	*任务完成用时/min*
50	100%	100%	1	20.3
100	100%	100%	3	38.6
150	100%	99.3%	4	59.7
200	100%	99.5%	3	79.4

任务到达率是指任务下发到 Android 测试终端的成功率，反映的是终端与系统的长连接稳定性。这 4 次实验过程中所有任务都下发到了 Android 终端，说明系统长连接稳定性不错。任务成功率表示 Android 终端在执行自动化测试过程中的任务成功情况，因为有任务执行失败的可能性，4 次实验中仅在任务量为 150 和 200 时分别只有 1 个任务最终未能成功执行。任务重发次数是指测试过程中的任务失败重发的次数。任务完成用时是指从任务开始到最后一个任务完成所用的时间。相比于普通的 Android 自动化测试方法只能串行执行，不支持多终端并行测试，单终端执行 50 个任务大概需要 3 小时左右，而本文系统实现了多终端多应用任务的并行执行，使得任务整体执行速度有明显提高，在 10 个终端并行执行下 50 个任务仅需 20.3 分钟即可完成。

4 结论

本文论述了 Android 自动化测试系统的设计与实现过程，针对常见的 Android 自动化测试方法中需要 PC 端环境配合，Android 终端不能独立测试等问题，通过长连接技术实现服务端向 Android 终端推送任务消息取代 Android 终端与 PC 端 USB 数据线物理连接的方法，并且实现在 Android 终端独立进行脚本录制回放功能，使得 Android 自动化测试能够实现多终端多任务的并行执行，显著提高了 Android 自动化测试的效率。

[参考文献] (References)

[1] Ariel.App Stores Growth Accelerates in 2014[OL].[2015-1-13].http://blog.appfigures.com/app-stores-growth-accelerates-in-2014/
[2] 高巍.Android 操作系统软件自动化测试方案的设计与实施[D].北京：北京邮电大学，2012
[3] 蔡增柱.基于 Android 移动平台测试相关技术研究[D].广州：华南理工大学，2013

[4] 张灿，薛云志，陈军成.一种基于 Android 平台 GUI 录制回放工具的设计与实现[J].计算机应用与软件，2012，29(12)：6-9
[5] 贾贝.基 Http 长连接的移动终端管理平台的设计及实现[D].北京：中国科学院大学，2013
[6] 冼学辉.基于 Web 的实时信息推送技术的研究[D].北京：华北电力大学，2013
[7] 马莉，唐善成，王静.云计算环境下的动态反馈作业调度算法[J].西安交通大学学报，2014，48(7)：77-82