基于 Android 平台多线程断点续传技术研究

石建华　聂文芳　文晓棠
江西师范大学软件学院　江西　南昌　330022

【摘　要】在Android平台的实际开发过程中，如何保证用户访问互联网资源的流畅，以及避免用户重复下载互联网资源造成带宽的浪费是一个常见的问题。针对这个问题，作者提出了一种基于Android平台多线程断点续传的技术，文章首先给出了多线程断点续传技术总体架构，然后分析了多线程断点续传的执行过程，最后详细描述了实现过程和关键代码。

【关键词】Android　多线程　断点续传　移动互联网
中图分类号:TP31 文献标识码:B 文章编号:1009-4067(2012)06-114-02

引言

随着移动互联网技术的发展，智能便携式设备受到消费者欢迎。由于现在移动互联网带宽有限等特征，用户体验更加重注使用的流畅性和较少的流量耗费满足用户需求。在实际移动开发过程中，如果使用多线程断点续传技术可以提高用户使用体验。本文基于实际需求，详细阐述了Android平台下，多线程断点续传技术总体架构、执行过程和关键代码。

1 Android平台介绍

Android是Google于2007年11月5日提出的基于Linux平台的开源手机操作系统。它包括操作系统、用户界面和应用程序。Android系统具有如下特点：(1)开放性：Android允许任何移动终端厂商加入到Android平台。(2)应用程序无界性：基于Android的应用程序可以通过标准API访问移动设备核心功能。(3)应用程序是在同等条件下创建的：移动终端上的应用程序可以被扩展或替换。(4)应用程序可以轻松地嵌入网络：应用程序可以很轻松地潜入HTML、JavaScript和样式表，还可以通过WebView组件显示网络内容。(5)应用程序可以并行运行：Android是一个完全的多任务系统，应用程序可以并行运行。

2 总体架构设计及流程图

多线程断点续传总体架构图和流程图如图1图2所示，

2.1 总体架构设计

(1)服务器端

服务器端架设在web服务器中，服务器端向客户端提供访问资源。当客户端的连接请求达到时，服务器端接受客户端请求，根据请求发过来的信息做出响应。若服务器端存在客户请求的资源，那么响应的信息以Http响应头的方式向客户端发出，相应信息包含访问资源名称、大小等。

(2)客户端

客户端是基于Android操作系统的移动设备。用户通过客户端获得访问资源的URL，客户端向服务器端根据获得URL与服务器端建立连接。若服务器端存在访问资源，那么客户端的下载引擎就开始进入下载状态，通过查询数据库选择进行普通下载还是断点下载，数据下载进度在客户端实时显示。

(3)数据库

采用Android集成好的SQLite数据库，在数据库中创建名称为download的数据库，该数据库中有一个download表，用于收集客户端下载数据的进度信息，表的结构为download(id,url,threadNo,vbegin,vend,progress)，其中url表示访问的资源位置，threadNo表示进程号，vbegin表示资源下载起始位置，vend表示资源下载结束位置，progress表示资源下载进度。下载引擎开始下载后，会实时地更新SQLite数据库中表的数据信息。资源下载完毕，下载信息记录从数据库表中删除(单个线程下载任务完成，该线程对应的记录也会删除)。

2.2 多线程断点续传流程分析

用户从客户端获得访问资源URL，并且与服务器建立连接，服务器向客户端返回匹配资源信息。客户端通过接收到响应信息获得该资源的大小、名称等，再根据URL提取目标文件的名称，并在对应的目录创建文件，大小和访问资源大小一样。下载引擎通过查询数据库中downLoad表，若数据库表存在该资源下载记录则选

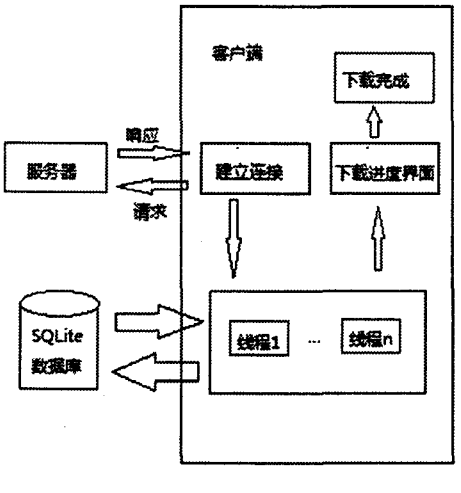

图1 总体架构图

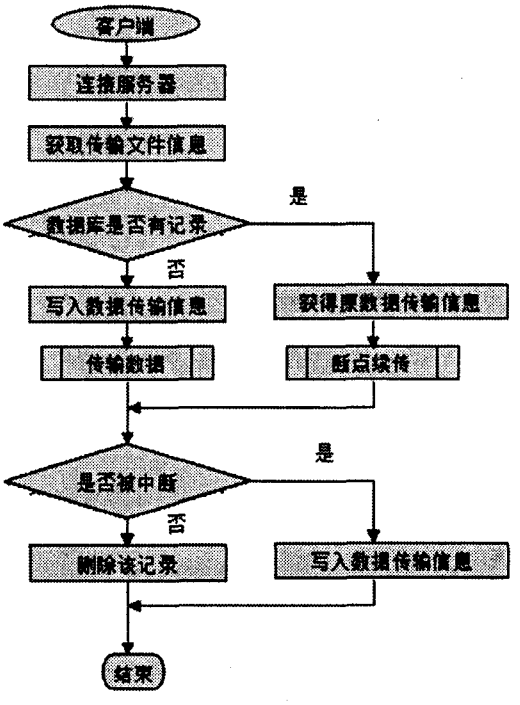

图2 流程图

择断点下载，若没有下载记录进行普通下载，下载过程中下载引擎会向数据库实时写入信息。如果文件下载完成，数据库删除该资源下载的信息，具体的流程如图2所示。

3 功能实现及关键代码

(1)与服务器建立连接

客户端向服务器端发送请求，服务器端对客户端发出的请求做出响应，如果存在客户端访问的资源，那么服务器端返回一个HttpURLConnection连接。

```
URL u=new URL(url); //url为访问的资源路径
HttpURLConnection conn=(HttpURLConnection)u.openConnection();
int total=conn.getContentLength();//Content-Length头返回文件大小
```

(2)初始化文件存储对象

获得下载文件长度后，初始化本地文件。本地文件大小要和下载文件大小一样，采用RandomAccessFile类型的对象raf来保存下载信息，RandomAccessFile支持随机读写，通过调用skip()方法，可以为不同的线程指定写入的位置，多个线程可以同时向raf写入数据。

```
fileName=url.substring(url.lastIndexOf('/')+1);//获得下载文件名称
file=new File(Environment.getExternalStorageDirectory()
        ,"/download/"+fileName);
RandomAccessFile raf=new RandomAccessFile(file,"rwd");
raf.setLength(total);//设置文件存储对象大小
```

(3)计算线程下载数据的信息

建立连接后得到下载资源的大小，就可以根据需要计算出线程个数以及每个线程下载的详细信息。默认设置线程的数量maxThread为4。前3个线程下载数据大小是一样的，最后一个线程的数据下载大小为数据总大小减前3个线程下载数据大小。下载起始位置=线程的排序位置*线程下载的数据长度，下载结束位置=下载起始位置+线程下载的数据长度，其中最后一个线程下载结束位置就是文件的大小。例如要下载一个9M的资源a.mp3，线程数为4，那么，前3条线程下载的数据长度都为2M，最后一个线程下载3M，线程开始下载和结束下载的位置如图3所示，

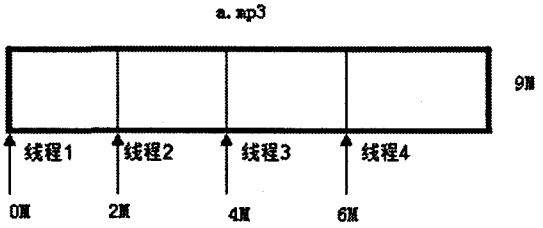

图3 多线线程下载图

实现的核心代码如下所示：

```
int perSize=total/maxThread;//total文总大小,maxThread线程个数
for(int i=0;i<maxThread;i++){
int begin=i*perSize; int end=begin+perSize;
if(i==maxThread-1){
end=total;//最后一个必须把余下全部下完
} }
```

(4)下载方式分类

下载方式分为两种：普通下载和断点下载。普通下载：开始进行下载，首先就会去查询数据库，通过资源的URL判断本次下载是否是首次下载，如果是首次下载则采用普通下载。

```
URL u = new URL(url);
HttpURLConnection conn = (HttpURLConnection) u.openConnection();
conn.setRequestProperty("Range","bytes="+(begin+progress)+"-"+(end-1));
InputStream in=conn.getInputStream();
RandomAccessFile raf=new RandomAccessFile(file,"rwd");
raf.skipBytes(begin+progress);//跳过begin个字节
byte[] buffer=new byte[1024];
int len=-1;
SQLiteDatabase db=dbHelper.getWritableDatabase();
while ((len = in.read(buffer)) > 0) {
raf.write(buffer,0,len);//把文件的数据写到物理文件中
progress+=len;      //记录下载进度
synchronized (totalProgress) {
totalProgress+=len;//计算总进度}
String sql="update download set progress=? where url=? and threadNo=?";
db.execSQL(sql, new Object[]{progress,url,threadNo});}
String sql="delete from download where url=? and threadNo=?";
db.execSQL(sql, new Object[]{url,threadNo});//删除下载任务记录
```

续传部分：如果客户端本次发送的请求，通过查询数据库中发现已有载记录，则使用断点下载。首先从download表中通过url查找当前资源每个线程下载的信息：线程号threadNo，下载起始位置vbegin，下载结束位置vend，下载进度progress。由于下载进度要实时更新到界面，因此还需要计算总进度。下载总进度的计算方法如下，

totalProgress = total-sum(vend-vbegin-progress)，下载总进度为totalProgress，件总大小为total，sum(vend-vbegin-progress)表示还需要下载字节数。关键代码如下，

```
String sql="select sum(vend-vbegin-progress) from download where url=?";
Cursor c=db.rawQuery(sql, new String[]{url});
if(c.moveToNext()){
    int balance=c.getInt(0);
    totalProgress=total-balance;
    progressBar.setProgress(totalProgress);//设置下载进度 }
while(cursor.moveToNext()){
    int threadNo=cursor.getInt(cursor.getColumnIndex("threadNo"));
    int begin=cursor.getInt(cursor.getColumnIndex("vbegin"));
    int end=cursor.getInt(cursor.getColumnIndex("vend"));
    int progress=cursor.getInt(cursor.getColumnIndex("progress"));
    Download Taskdt=new DownloadTask(threadNo,
                begin, end,progress);
    new Thread(dt).start();  //开始下载,放到新线程中运行}
```

4 总结

本文对多线程断点续传技术体系结构进行了分析，介绍了多线程下载断点续传的执行过程，并且对该技术实现的细节和关键代码进行了详述，在实际开发过程中，应用该技术可以明显提高用户的使用体验，能够以最小的流量耗费满足用户的功能需求。

参考文献

[1] 杨丰盛.Android应用开发揭秘[M].北京:机械工业出版社,2010.

[2] 林城.GOOGLE ANDROID 2. x应用开发实战[M].北京:清华大学出版社,2011.

[3] 余志龙、陈昱勋、郑名杰、陈小凤、郭秩均,Google Android SDK开发范例大全(第2版),北京:人民邮电出版社,2010

[4] Android Developers, http://www.Android.com

[5] SQLite Documentation, stinctive Features, 2008.12

[6] Chris Newman.SQLite[M].Sam Publishing,2004.

[7] http://www.sqlite.org.

作者简介

石建华，江西师范大学2010级硕士研究生，工程师，主研领域：服务计算。