



(12) 发明专利申请

(10) 申请公布号 CN 104202373 A

(43) 申请公布日 2014. 12. 10

(21) 申请号 201410416522. X

(22) 申请日 2014. 08. 21

(71) 申请人 清华大学深圳研究生院

地址 518055 广东省深圳市南山区西丽大学  
城清华校区

(72) 发明人 李清 江勇 贺菊华 谭亚垒

(74) 专利代理机构 深圳新创友知识产权代理有  
限公司 44223

代理人 江耀纯

(51) Int. Cl.

H04L 29/08 (2006. 01)

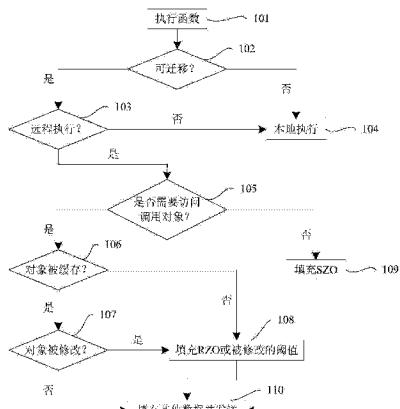
权利要求书2页 说明书12页 附图3页

(54) 发明名称

移动云计算迁移方法及系统

(57) 摘要

本发明公开了一种移动云计算迁移方法及系统,用于将移动端的计算负载向云服务器端迁移。移动云计算迁移方法包括迁移判断步骤和能耗比较步骤,其中所述迁移判断步骤用于判断所述移动端所需要执行的函数是否能够迁移到所述云服务器端执行;所述能耗比较步骤用于比较能够迁移到所述云服务器端执行的函数分别在所述云服务器端与所述移动端执行所需的时间和能耗,并将在所述云服务器端执行所需的时间比较短且能耗比较小的函数迁移到所述云服务器端执行。本发明提出一种轻量级的移动云计算迁移决策框架模型,将移动端所执行的程序中计算耗时耗能的函数迁移到云服务器端执行。



1. 一种移动云计算迁移方法,用于将移动端的计算负载向云服务器端迁移,其特征在于,包括迁移判断步骤和能耗比较步骤,其中:

所述迁移判断步骤用于判断所述移动端所需要执行的函数是否能够迁移到所述云服务器端执行;

所述能耗比较步骤用于比较能够迁移到所述云服务器端执行的函数分别在所述云服务器端与所述移动端执行所需的时间和能耗,并将在所述云服务器端执行所需的时间比较短且能耗比较小的函数迁移到所述云服务器端执行。

2. 根据权利要求 1 所述的移动云计算迁移方法,其特征在于,还包括传输控制步骤,所述传输控制步骤包括:

分析迁移到所述云服务器端执行的函数是否需要访问调用对象的内部对象;

对于不需要访问调用对象的内部对象的函数,所述移动端直接向云服务器传输调用对象;

对于需要访问其调用对象的内部对象的函数,所述移动端向所述云服务器端发送缓存请求,所述缓存请求的对象包括调用对象以及只有通过该调用对象才能直接或间接访问的对象。

3. 根据权利要求 2 所述的移动云计算迁移方法,其特征在于,还包括缓存置换步骤,所述缓存置换步骤是针对需要访问其调用对象的内部对象的函数,包括:

A1 :所述移动端向所述云服务器端发送缓存请求,所述云服务器端检测所述缓存请求的对象是否存在于所述云服务器端的内存中,若存在则结束;若不存在,则进行步骤 A2;

A2 :所述云服务器端检测是否有足够的缓存空间,如果有,则所述移动端直接将所述缓存请求的对象缓存在所述云服务器端,如果没有,则进行步骤 A3;

A3 :所述云服务器端检测内存中重要度最低的对象和其对应的应用程序,检测该对应的应用程序和发送缓存请求的应用程序是否属于同一应用程序,如果是则结束,如果不是,则执行步骤 A4;

A4 :删除重要度最低的对象后所述移动端再将所述缓存请求的对象缓存在所述云服务器端。

4. 根据权利要求 3 所述的移动云计算迁移方法,其特征在于,还包括对象探测步骤:所述缓存置换步骤中所述云服务器端检测所述缓存请求的对象存在于所述云服务器端的内存中结束后,比较所述云服务器端中的对象和所述移动端中的对象,如果所述移动端中的对象存在修改过的值,将所述修改过的值传输到云服务器端。

5. 根据权利要求 1 至 4 任一项所述的移动云计算迁移方法,其特征在于,所述迁移判断步骤包括:

B1 :所述移动端有函数需要执行时,查询所述移动端的数据库是否有该函数的相关信息;

B2 :如果所述移动端的数据库有该函数的相关信息,则可直接判断该函数是否是可迁移函数;如果所述移动端的数据库没有该函数的相关信息,将该函数传输到所述云服务器端;

B3 :查询所述云服务器端上是否有该函数的相关信息,如果有,则可判断该函数是否是可迁移函数,如果没有,则通过云服务器端的用户处理器判断该函数是否是可迁移函数。

6. 一种移动云计算迁移系统,用于将移动端的计算负载向云服务器端迁移,其特征在于,包括移动端和云服务器端,所述移动端包括代码分析模块、调度参数监测模块和在线决策模块,所述云服务器端包括函数分析模块,其中:

所述代码分析模块和所述函数分析模块用于判断所述移动端要执行的函数是否能够迁移到云服务器端执行;

所述调度参数监测模块用于计算能够迁移到所述云服务器端执行的函数分别在所述云服务器端与所述移动端执行所需的时间和能耗,通过比较,如果所述函数迁移到云服务器端执行时所需要的时间比较短并且能耗比较小,则所述在线决策模块确定所述函数为需迁移到云服务器端执行的函数。

7. 根据权利要求 6 所述的移动云计算迁移系统,其特征在于,所述代码分析模块还用于对迁移到所述云服务器端执行的函数是否需要访问调用对象的内部对象进行分析;对于不需要访问调用对象的内部对象的函数,所述移动端直接向云服务器传输调用对象;对于需要访问其调用对象的内部对象的函数,所述移动端向所述云服务器端发送缓存请求,所述缓存请求的对象包括调用对象以及只有通过该调用对象才能直接或间接访问的对象。

8. 根据权利要求 7 所述的移动云计算迁移系统,其特征在于,所述云服务器端还包括数据缓存模块,其中:所述数据缓存模块用于缓存所述移动端的缓存请求的对象,

所述移动端向所述云服务器端发送缓存请求时,所述云服务器端检测所述缓存请求的对象是否存在于所述数据缓存模块中,若存在则不缓存;若不存在,则所述云服务器端检测所述数据缓存模块中是否有足够的缓存空间,如果有,则所述移动端直接将所述缓存请求的对象缓存在所述数据缓存模块,如果没有,则所述云服务器端检测所述数据缓存模块中重要度最低的对象和其对应的应用程序,检测该对应的应用程序和发送缓存请求的应用程序是否属于同一应用程序,如果是则不缓存,如果不是,则删除重要度最低的对象后所述移动端再将所述缓存请求的对象缓存在所述数据缓存模块。

9. 根据权利要求 8 所述的移动云计算迁移系统,其特征在于,所述云服务器端还包括对象探测模块,其中:

所述移动端向所述云服务器端发送缓存请求时,所述云服务器端检测所述缓存请求的对象存在于所述数据缓存模块中时,所述对象探测模块比较所述云服务器端中的对象和所述移动端中的对象,如果所述移动端中的对象存在修改过的值,将所述修改过的值传输到云服务器端。

10. 根据权利要求 6 至 9 任一项所述的移动云计算迁移系统,其特征在于,所述移动端有函数需要执行时,查询移动端的数据库是否有该函数的相关信息;如果所述移动端的数据库有该函数的相关信息,所述代码分析模块判断该函数是否是可迁移函数;如果所述移动端的数据库没有该函数的相关信息,将该函数传输到所述云服务器端,所述函数分析模块判断该函数是否是可迁移函数,并且把结果返回给所述代码分析模块。

## 移动云计算迁移方法及系统

### 技术领域

[0001] 本发明涉及移动互联网技术领域，尤其涉及一种移动云计算迁移方法及系统。

### 背景技术

[0002] 近年来智能移动手持设备迅速发展，同时相应的3G、4G网络大幅度提升了移动网络带宽，促使移动互联网规模急剧膨胀。各种各样的移动应用呈现在人们眼前，诸如各类手机游戏、社交软件等。

[0003] 相比传统互联网客户端，移动手持设备具有独特的便携性优势，同时，移动互联网也存在着无法忽视的局限性：(1)与个人电脑相比，移动手持设备计算能力弱，同时续航能力较差，无法有效支撑计算敏感型应用，如视频游戏、图片处理、语音分析、自然语言处理等；(2)与传统互联网相比，移动互联网带宽仍然较小。虽然3G、4G、WiFi等移动网络技术的发展使移动互联网带宽不断突破，但同时更多移动应用的涌现会对带宽需求提出进一步的挑战。

[0004] 移动云计算(Mobile Cloud Computing)是未来移动互联网发展的主流方向。移动云计算技术可以将移动应用的“计算”任务从终端迁移到云服务器，从而减轻终端设备计算负担，同时降低终端设备的电源消耗并提升其续航能力。移动云计算主要思路是通过把移动设备端的软件中耗能耗时的计算运行单元迁移到云服务器端(拥有强大计算能力的计算机集群)，通过利用云服务器端丰富的资源和计算能力来运行执行这些计算密集型单元，并把结果返回到移动设备。然而，移动设备处在动态变化的移动网络中，网络环境较为复杂，在移植计算的过程中需要不断进行交互。当网络环境变差时，移植计算所需的网络传输会成为移动云计算的瓶颈，严重影响实时应用(如交互式游戏等)的用户体验。此外，移动设备的硬件条件不一，平台差异化较大，同样给移动云计算平台架构的设计带来巨大挑战。因此，如何解决模块从移动端到云服务器端的迁移问题并提升用户体验成为移动云计算的关键技术难题。

### 发明内容

[0005] 为解决上述问题，本发明提供了一种轻量级的移动云计算迁移方法，将移动端执行的程序中耗时耗能的函数迁移到云服务器端执行。

[0006] 为达到上述目的，本发明采用以下技术方案：

[0007] 本发明公开了一种移动云计算迁移方法，用于将移动端的计算负载向云服务器端迁移，包括迁移判断步骤和能耗比较步骤，其中：

[0008] 所述迁移判断步骤用于判断所述移动端所需要执行的函数是否能够迁移到所述云服务器端执行；

[0009] 所述能耗比较步骤用于比较能够迁移到所述云服务器端执行的函数分别在所述云服务器端与所述移动端执行所需的时间和能耗，并将在所述云服务器端执行所需的时间比较短且能耗比较小的函数迁移到所述云服务器端执行。

[0010] 进一步地,所述移动云计算迁移方法还包括传输控制步骤,所述传输控制步骤包括:

[0011] 分析迁移到所述云服务器端执行的函数是否需要访问调用对象的内部对象;

[0012] 对于不需要访问调用对象的内部对象的函数,所述移动端直接向云服务器传输调用对象;

[0013] 对于需要访问其调用对象的内部对象的函数,所述移动端向所述云服务器端发送缓存请求,所述缓存请求的对象包括调用对象以及只有通过该调用对象才能直接或间接访问的对象。

[0014] 更进一步地,所述移动云计算迁移方法还包括缓存置换步骤,所述缓存置换步骤是针对需要访问其调用对象的内部对象的函数,包括:

[0015] A1:所述移动端向所述云服务器端发送缓存请求,所述云服务器端检测所述缓存请求的对象是否存在与所述云服务器端的内存中,若存在则结束;若不存在,则进行步骤A2;

[0016] A2:所述云服务器端检测是否有足够的缓存空间,如果有,则所述移动端直接将所述缓存请求的对象缓存在所述云服务器端,如果没有,则进行步骤A3;

[0017] A3:所述云服务器端检测内存中重要度最低的对象和其对应的应用程序,检测该对应的应用程序和发送缓存请求的应用程序是否属于同一应用程序,如果是则结束,如果不是,则执行步骤A4;

[0018] A4:删除重要度最低的对象后所述移动端再将所述缓存请求的对象缓存在所述云服务器端。

[0019] 更进一步地,所述移动云计算迁移方法还包括对象探测步骤:所述缓存置换步骤中所述云服务器端检测所述缓存请求的对象存在于所述云服务器端的内存中结束后,比较所述云服务器端中的对象和所述移动端中的对象,如果所述移动端中的对象存在修改过的值,将所述修改过的值传输到云服务器端。

[0020] 更进一步地,所述迁移判断步骤包括:

[0021] B1:所述移动端有函数需要执行时,查询所述移动端的数据库是否有该函数的相关信息;

[0022] B2:如果所述移动端的数据库有该函数的相关信息,则可直接判断该函数是否是可迁移函数;如果所述移动端的数据库没有该函数的相关信息,将该函数传输到所述云服务器端;

[0023] B3:查询所述云服务器端上是否有该函数的相关信息,如果有,则可判断该函数是否是可迁移函数,如果没有,则通过云服务器端的用户处理器判断该函数是否是可迁移函数。

[0024] 本发明另外还公开了一种移动云计算迁移系统,用于将移动端的计算负载向云服务器端迁移,包括移动端和云服务器端,所述移动端包括代码分析模块、调度参数监测模块和在线决策模块,所述云服务器端包括函数分析模块,其中:

[0025] 所述代码分析模块和所述函数分析模块用于判断所述移动端要执行的函数是否能够迁移到云服务器端执行;

[0026] 所述调度参数监测模块用于计算能够迁移到所述云服务器端执行的函数分别在

所述云服务器端与所述移动端执行所需的时间和能耗,通过比较,如果所述函数迁移到云服务器端执行时所需要的时间比较短并且能耗比较小,则所述在线决策模块确定所述函数为需迁移到云服务器端执行的函数。

[0027] 进一步地,所述代码分析模块还用于对迁移到所述云服务器端执行的函数是否需要访问调用对象的内部对象进行分析;对于不需要访问调用对象的内部对象的函数,所述移动端直接向云服务器传输调用对象;对于需要访问其调用对象的内部对象的函数,所述移动端向所述云服务器端发送缓存请求,所述缓存请求的对象包括调用对象以及只有通过该调用对象才能直接或间接访问的对象。

[0028] 更进一步地,所述云服务器端还包括数据缓存模块,其中:所述数据缓存模块用于缓存所述移动端的缓存请求的对象,所述移动端向所述云服务器端发送缓存请求时,所述云服务器端检测所述缓存请求的对象是否存在与所述数据缓存模块中,若存在则不缓存;若不存在,则所述云服务器端检测所述数据缓存模块中是否有足够的缓存空间,如果有,则所述移动端直接将所述缓存请求的对象缓存在所述数据缓存模块,如果没有,则所述云服务器端检测所述数据缓存模块中重要度最低的对象和其对应的应用程序,检测该对应的应用程序和发送缓存请求的应用程序是否属于同一应用程序,如果是则不缓存,如果否,则删除重要度最低的对象后所述移动端再将所述缓存请求的对象缓存在所述数据缓存模块。

[0029] 更进一步地,所述云服务器端还包括对象探测模块,其中:

[0030] 所述移动端向所述云服务器端发送缓存请求时,所述云服务器端检测所述缓存请求的对象存在于所述数据缓存模块中时,所述对象探测模块比较所述云服务器端中的对象和所述移动端中的对象,如果所述移动端中的对象存在修改过的值,将所述修改过的值传输到云服务器端。

[0031] 更进一步地,所述移动端有函数需要执行时,查询移动端的数据库是否有该函数的相关信息;如果所述移动端的数据库有该函数的相关信息,所述代码分析模块判断该函数是否是可迁移函数;如果所述移动端的数据库没有该函数的相关信息,将该函数传输到所述云服务器端,所述函数分析模块判断该函数是否是可迁移函数,并且把结果返回给所述代码分析模块。

[0032] 本发明与现有技术相比的有益效果包括:本发明的移动云计算迁移方法通过数学能耗比较模型,通过数学方法计算能够迁移到所述云服务器端执行的函数在云服务器端及移动端的时间和能耗,并通过比较判定是否将函数迁移到云服务器端执行,从而实现不忽略移动端的计算资源而只将一些在云服务器端执行更省时和节能的函数迁移到云服务器端执行,同时也节约传输资源。

[0033] 在优选方案中,本发明通过分析函数是否需要访问调用对象的内部对象,而决定向云服务器端传输或缓存相应所必须的对象,从而不浪费传输资源并减少电源的消耗;更优选的方案中,通过采用缓存置换步骤,将移动端的缓存请求的对象缓存到云服务器端,而当云服务器端缓存空间不足时,置换掉重要度低的对象,这样在多次缓存请求执行后,云服务器端会存储重要度较高的一些对象,对于执行频率高的对象所对应的函数需要执行时而不需再重复传输,从而保证函数的快速执行,并且提高云服务器端缓存的 cache 命中率,使移动端的计算负载向云服务器端迁移的过程中传输数据量大幅度压缩。另外,本发明还提供了一套简单方便的 API,能够动态去识别并执行可迁移函数,从而不再需要人为判定函数

是否能够在云服务器端执行；本发明还提出一个利用序列化机制的对象探测模块，通过该模块，可以精准快速地找到调用对象哪些阈值被修改，从而保证函数在云服务器端的执行。

### 附图说明

[0034] 图 1 是本发明优选实施例中的移动云计算迁移方法中的一个应用程序中的有向权重图；

[0035] 图 2 是本发明优选实施例的移动云计算迁移方法的流程图；

[0036] 图 3 是本发明优选实施例的移动云计算迁移方法中的缓存置换步骤的流程图。

### 具体实施方式

[0037] 下面对照附图并结合优选的实施方式对本发明作进一步说明。

[0038] 本发明优选实施例的移动云计算迁移方法是提供一个将移动端程序中计算耗时耗能的函数迁移到云服务器端执行的框架，主要包括以下步骤：

[0039] 第一，迁移判断步骤，即判断函数是否能够迁移到云服务器端执行。移动应用程序的开发者众多，所开发的程序也各不相同，因此，本优选实施例为识别程序中可以迁移到云服务器端执行的代码块提供一套完整的 API 框架，这是一个封装了自动判断函数是否可迁移流程的控制逻辑的骨架。此 API 的基类 Base 提供了一个 Invoke 函数，该函数的第一个参数是被执行的函数的名称，后面的参数为被执行的函数所需的参数。应用程序开发者只需在应用程序中继承 Base 基类，在调用程序开发者自己的函数时通过继承 Base 基类中的 Invoke 函数以间接调用这些函数。

[0040] 对于某函数是否应该迁移到云服务器端执行，主要通过 Base 基类里自带的单例类执行控制器控制。当移动端的一个应用程序启动的时候，每当它需要去执行一个自定义函数，移动端的执行控制器会去查询数据库，去查询这个函数的相关信息去判断这个函数是否是可迁移的函数，如果不能找到相关的信息，执行控制器将该函数传输到云服务器端，让云服务器端，即当一个首次执行的应用第一次执行自己的函数时，将这些函数全部迁移到服务器执行，通过返回的异常分析它们是否需要访问本地资源，若需要访问本地资源即不能迁移到云服务器端执行，而不需要访问本地资源的函数可以迁移到云服务器端执行。所以通过返回的异常即可判断该函数是可迁移函数 (Remotable Method, RM) 还是不可迁移函数 (Unremotable Method, URM)。

[0041] 第二，能耗比较步骤，即对函数进行能耗分析。一个应用程序可以被看做由若干个 RM 函数和 URM 函数构成，用一个图来表示就是： $G = (V, E)$ ， $V$  表示该应用程序有  $V$  个函数， $E$  表示函数与函数之间是否存在执行关系。如图 1 所示，可以用一个有向权重图来表示一个应用程序  $G$ ，对每个点  $u, v \in V$ ， $e_{uv}$  是指点  $u$  到  $v$  的一条有向边，表示函数  $u$  执行完后接着函数  $v$  时需要传输的数据量。

[0042] 为计算每个函数在执行时候所需的时间及能耗，定义如下参数：

[0043] a)  $T_v^l(t)/T_v^r(t)$  表示函数  $v$  分别在移动端 / 云服务器端所执行所需要的时间 ( $t$  表示在  $v$  的第  $t$  次执行)

[0044] b)  $T_{uv}(t)/\varepsilon_{uv}(t)$  表示函数  $u$  执行完后去执行  $v$  所需要传输数据量所消耗的时间和

能源

[0045] c)  $p_i/p_1$  表示处理器休闲以及激活状态下所需要消耗的单位能耗

[0046] d)  $\varepsilon_v^l(t)$  表示函数 v 在移动端执行所需要消耗的能源

[0047] e)  $p_{tx}/p_{rx}$  表示移动端传输和接收数据所需要的单位能耗

[0048] f)  $R(t)/S(t)$  表示上传 / 下载率

[0049] 对函数 v, 定义  $\omega_t(t) \in \{0, 1\}$ 。  $\omega_v(t) = 1$  时表示函数 v 在移动端中执行 ; 等于 0 则表示在云服务器端执行。如果函数 v 在第 t 次执行时在本地执行, 则有  $\varepsilon_v^l(t) = p_l T_v^l(t)$ 。

当  $\omega_u(t) = \omega_v(t)$ , 有  $T_{uv}(t) = 0$ ,  $\varepsilon_{uv}(t) = 0$ , 因为当函数 u 和函数 v 在同一个环境下执行时, 其传输能耗和时间均为 0。否则当  $\omega_u(t) = 1$  时,  $T_{uv}(t) = e_{uv}/R(t)$ ,  $\varepsilon_{uv}(t) = p_{tx} T_{uv}(t)$ , 而  $\omega_v(t) = 1$  时,  $T_{uv}(t) = e_{uv}/S(t)$   $\varepsilon_{uv}(t) = p_{rx} T_{uv}(t)$ 。移动端在空闲等待状态下的能耗为 :

[0050]  $\varepsilon_i(t) = p_i [(1 - \omega_v(t)) T_v^r(t) + |\omega_u(t) - \omega_v(t)| T_{uv}(t)]$ 。

[0051] 因此函数 v 第 t 次执行时能耗和时间如下 :

[0052]  $\varepsilon_v(t) = \varepsilon_i(t) + \omega_v(t) \varepsilon_v^l(t) + |\omega_u(t) - \omega_v(t)| \varepsilon_{uv}(t)$

[0053]  $T_v(t) = \omega_v(t) T_v^l(t) + (1 - \omega_v(t)) T_v^r(t) + |\omega_u(t) - \omega_v(t)| T_{uv}(t)$

[0054] 假如 v 是一个 RM 函数, 如果  $\varepsilon_v(t)_{w_v(t)=0} < \varepsilon_v(t)_{w_v(t)=1}$  且  $T_v(t)_{w_v(t)=0} < T_v(t)_{w_v(t)=1}$ , 可判断当前执行的函数应该迁移到云服务器端执行, 从而节省执行时间和能耗。

[0055] 第三, 传输控制步骤, 即确定传输数据类型。一个对象的 Shallow Size 是指系统分配的用来维护该对象最基本的内存大小, 它不包含任何这个对象里面内部数据的引用。一个对象的 Retained Size 是指该对象以及通过该对象才能直接或间接访问的对象的总大小。该对象被释放时, 它从内部访问的其他对象也将被回收, 因此该对象的 Retained Size 就是系统回收该对象时所能回收的最大的内存值。这意味着一个对象的 Retained Size 要大于这个对象的 Shallow Size。

[0056] 在把函数迁移到云服务器端执行的过程中, 采用的技术主要是 java 反射机制, 而在执行前必须把调用该函数的那个对象传输到云服务器端, 表 1 是用 android 的 DDMS 工具获取的部分进程中对象大小的分析表, 从表中可以看出, 应用程序的一个堆对象的 Retained Size 和 Shallow Size 差距特别大, 且上一步骤也已说明, 传输数据量越大,  $e_{uv}$  就越大, 从而延长函数执行的时间并增加能耗, 因此提出通过传输一个对象的 Shallow Size 以减少数据传输量。

[0057] 表 1 典型应用程序的堆对象大小

[0058]

应用程序名称	堆对象计数	最大	最大	最小
--------	-------	----	----	----

[0059]

		Retained Size (KB)	Shallow Size (B)	Shallow Size (B)
Angry Bird	18	3288	200	16
Defender	35	2979	180	21
Lunar Lander	22	3288	221	18

[0060] 由于 RM 函数在执行时有可能需要访问其调用对象的内部对象,如果只传输一个对象的 Shallow Size 则不能保证该函数的正常执行。因此对于不需要访问调用对象的内部对象的函数,只需传输该对象的 Shallow Size 就能保证程序的正常执行;但对于需要访问调用对象的内部对象的函数,就必须传输该对象的 Retained Size。且一个进程一旦启动,如果不被禁止就将一直重复执行下去,使得该进程中的很多函数被不断重复执行,造成同一个对象的 Retained Size 被反复传输到云服务器端。因此,为进一步减少数据传输量,提出将需要把 Retained Size 传输到云服务器端的对象缓存到云服务器端。

[0061] 因此,当 RM 函数在执行时不需要访问调用对象的内部对象时,只需传输对象的 Shallow Size;而当 RM 函数在执行时需要访问调用对象的内部对象时,则需传输对象的 Retained Size。而判断 RM 函数在执行时是否需要访问其调用对象的内部对象,本发明优选是采用代码技术实现的,即通过对象包含的属性是否空指针来判断:如果不需某个对象,由于初始化该对象时对象里的属性都是空指针,这样不会产生异常;而如果需要这个对象,就会访问空指针,从而产生异常。故通过是否产生异常的判断就能判断出是否需要访问某个调用对象的内部对象。

[0062] 第四,缓存置换步骤,即对缓存请求的对象进行缓存。由于在程序执行过程中,大的对象数据可能会被反复传输,因此为了减少数据传输量,本发明提出一个新的缓冲策略以缓冲这些数据。传统的缓存策略对函数的执行频率并不敏感,但一个应用程序中各函数的执行频率是不尽相同的。因此如果用传统缓存策略统一处理这些函数,就无法保证它们需要的调用对象能在云服务器端长期缓存。针对这种情况,本发明优选实施例提出一个基于重要度的 LRU(Importance-based LRU, IB-LRU) 缓存策略。

[0063] 当一个应用程序运行时,如果该应用程序的函数执行频率比较高,说明该应用的对象数据对服务器来说比较重要,应该优先缓存这些数据。而一个应用程序中的某个对象被函数调用频率较高时,说明该实例类对象在当前的应用中具有更高的优先级在云服务器端中缓存。

- [0064] 为计算一个函数的调用对象的重要度,定义以下一些变量:
- [0065] Count( $app_j$ ) 表示第  $j$  个应用程序自启动开始请求的云服务器端执行次数;
- [0066] Count( $object_i$ ) 表示调用对象  $i$  自其应用程序启动后在云服务器端执行的次数;
- [0067] Count( $server$ ) 表示自云服务器端启动后移动端请求在云服务器端上执行的次数;
- [0068]  $I(object_i, app_j)$  表示对象  $i$  对于应用程序  $j$  的重要度;
- [0069]  $I(app_j, server)$  表示应用程序  $j$  对于云服务器端的重要度;
- [0070]  $I(object_i, server)$  表示对象  $i$  对于云服务器端的重要度;

[0071] Start( $app_j$ ) 表示应用程序  $j$  启动时, 已经在云服务器端执行的函数数量。

[0072] 具体计算逻辑如下:

$$[0073] Count(app_j) = \sum_{i=1}^{n^2} Count(object_i) \cdots \cdots (1)$$

$$[0074] I(object_i, app_j) = \frac{Count(object_i)}{Count(app_j)} \cdots \cdots (2)$$

$$[0075] Count(server) = \sum_{j=1}^{n^2} Count(app_j) \cdots \cdots (3)$$

$$[0076] I(app_j, server) = \frac{Count(app_j)}{Count(server) - Start(app_j)} \cdots \cdots (4)$$

[0077] 基于重要度的 LRU (Importance-based LRU, IB-LRU) 缓存策略算法具体如下:

[0078] 输入:

[0079] 触发移动云计算迁移的应用  $app_j$

[0080] 迁移方法所属对象  $object_i$

[0081] 迁移到云服务器端的方法  $m$

[0082] 输出:

[0083] 是否需要缓存对象  $object_i$

[0084] 1: 初始化 CacheFlag 为 false

[0085] 2: if 缓存空间已满 then

[0086] 3: 执行缓存置换算法 IB-RS

[0087] 4: 设置 CacheFlag 为缓存置换算法 IB-RS 的返回值

[0088] 5: end if

[0089] 6: if CacheFlag 为 true then

[0090] 7: 在云服务器端缓存  $object_i$

[0091] 8:  $object_i$  调用次数  $Count(object_i)$  加 1

[0092] 9:  $app_j$  调用次数  $Count(app_j)$  加 1

[0093] 10: if  $Count(app_j)$  为 1 then

[0094] 11:  $Start(app_j)$  赋值为  $Count(server)$

[0095] 12: end if

[0096] 13:  $Count(server)++$

[0097] 14: end if

[0098] 15: return

[0099] 本发明优选实施例中提出一种缓存置换算法 (Importance Based-Replace Strategy, IB-RS 算法), 缓存置换算法是移动端请求在云服务器端执行远程函数时需缓存调用对象, 而云服务器端缓存空间不足时的缓存置换策略。缓存置换算法的主要思路是: 当移动端应用请求一个远程函数执行, 云服务器端会接受相关执行数据, 并通过分析判断是否需要将该函数的调用对象缓存在云服务器端。如果判断需要在云服务器端缓存该对象, 则先检查云服务器端缓存是否有足够的剩余空间, 如果有剩余空间, 则直接缓存此对象; 如果空间不足则执行 IB-RS 算法, 以寻找对云服务器重要度最低的对象。找到对应对象数据

时,先判断该对象是否与待缓存对象属于同一个应用程序。由于理论上应用程序运行到一个稳定期时,已缓存对象的重要度要高于未缓存对象,因此如果待缓存对象与云服务器端的最小重要度对象属于同一个应用程序,则默认不需要置换,否则将最小重要度对象从缓存中删除。如果被删除对象的应用程序没有其它对象在缓存中,表明这个应用程序可能被停止,就把该应用程序的所有数据清空。最后,把待缓存对象缓存到云服务器端,并修改相应的数据。

- [0100] 缓存置换算法具体如下 :
- [0101] 输入 :
- [0102] 需要被缓存的迁移方法所属对象  $object_{cache}$
- [0103] 迁移到云服务器端的方法  $m$
- [0104] 临时保存缓存中对象  $object$  的最小重要性  $I_{min}$
- [0105] 临时保存拥有最小重要性的对象的应用程序名  $app_{obj}$
- [0106] 输出 :
- [0107] 1 :for 每一个在云服务器端缓存的对象 do
- [0108] 2 :根据之前的变量定义规则计算出每一个
- [0109] 3 :对象它相对于云服务器端的重要性  $I(object_i, server)$
- [0110] 4 : $I_{min}$  记录最小的值  $I(object_i, server)$ ,  $app_{obj}$  为
- [0111] 5 :对应的  $app_{obj}$  所属的应用  $app_j$
- [0112] 6 :if  $object_{cache}$  和  $object_i$  同属于一个 app then
- [0113] 7 :return false
- [0114] 8 :end if
- [0115] 9 :把缓存中重要性低的对象  $object_i$  置换出去
- [0116] 10 :if 缓存中不再有  $app_{obj}$  程序的对象 then
- [0117] 11 :Count( $app_{obj}$ ) 清空为 0
- [0118] 12 :Start( $app_{obj}$ ) 清空为 0
- [0119] 13 :end if
- [0120] 14 :return true
- [0121] 综上所述,本发明优选实施例中的缓存置换步骤具体流程如下 :
- [0122] 1) 移动端向云服务器端发送一个缓存请求 ;
- [0123] 2) 在服务器中检查该请求对象是否在云服务器端的内存中,若存在则结束 ;
- [0124] 3) 如果缓存请求的对象不在云服务器端的内存中,就判断云服务器端的缓存中是否还有剩余空间,如果有剩余空间,直接缓存该对象并返回 ;
- [0125] 4) 如果云服务器端的缓存空间已经被填满,则遍历云服务器端缓存中所有的对象,根据基于重要度的 LRU 缓存规则计算这些对象的重要度 ;
- [0126] 5) 删除重要度最低的对象。在重要度相同的情况下,按 IB-LRU 算法测量进行置换 ;
- [0127] 6) 如果被删除对象的应用程序已经没有其它任何对象在缓存中,则删除该应用程序的相关信息 ;
- [0128] 7) 在云服务器端上缓存请求的对象 ;

[0129] 8) 如果该对象是这个应用程序第一个缓存到云服务器端的对象,则初始化 Start( $app_j$ )。

[0130] 第五,对象探测步骤,即同步更新缓存数据。在将调用对象传输到云服务器端时,如果该对象已经缓存在云服务器端,为保证当前执行的 RM 函数在云服务器端执行时访问的对象保持一致,需要比较云服务器端的对象和移动端对象以找出那些在移动端修改过的值,再把这些值传输到云服务器端。该模块主要使用了 java 中序列化的操作,通过对对象的序列化找出被修改过的值,然后与云服务器端进行同步更新。

[0131] 以上是本发明优选实施例的移动云计算迁移方法的步骤,总结具体流程图如图 2,进一步说明如下:

[0132] 步骤 101 :执行函数;

[0133] 步骤 102 :通过迁移判断步骤判断被执行的函数是否是可迁移函数,如果是,执行步骤 103,如果否,执行步骤 104;

[0134] 步骤 103 :通过能耗比较步骤对函数进行能耗分析,分析是否应该迁移到云服务器端执行,如果是,执行步骤 105,如果否,执行步骤 104;

[0135] 步骤 104 :将函数在移动端本地执行;

[0136] 步骤 105 :通过传输控制步骤判断函数执行时是否需要访问其调用对象的内部对象,如果是,执行步骤 106,如果否,执行步骤 109;

[0137] 步骤 106 :通过缓存置换步骤判断函数需要的对象是否已经缓存在云服务器端,如果是,执行步骤 107,如果否,执行步骤 108;

[0138] 步骤 107 :通过对象探测步骤判断函数在移动端的对象对比缓存在云服务器端的对象是否被修改过,如果是,执行步骤 108,如果否,执行步骤 110;

[0139] 步骤 108 :填充函数的对象的 Retained Size 或者被修改过的阈值;

[0140] 步骤 109 :填充函数的对象的 Shallow Size;

[0141] 步骤 110 :将填充的数据传输到云服务器端。

[0142] 如图 3 所示,为本发明优选实施例的移动云计算迁移方法的步骤中的缓存置换步骤流程图,即对步骤 106 的进一步细化,具体说明如下:

[0143] 步骤 201 :移动端向云服务器端发出缓存请求;

[0144] 步骤 202 :云服务器端检测是否已经在缓存中,如果是,执行步骤 207,如果否,执行步骤 203;

[0145] 步骤 203 :检测云服务器端的缓存空间是否满了,即是否不够存储缓存请求所需的内存,如果是,执行步骤 204,如果否,执行步骤 206;

[0146] 步骤 204 :找出对云服务器重要度最低的对象  $I_{min}$  及其对应的应用程序名  $app_{obj}$ ;

[0147] 步骤 205 :移动端执行的函数对应的应用程序是否与  $app_{obj}$  属于同一 app,如果是,执行步骤 207;如果否,执行步骤 206;

[0148] 步骤 206 :删除对云服务器重要度最低的对象  $I_{min}$  及其对应的应用程序名  $app_{obj}$  数据,缓存发出缓存请求的对象;

[0149] 步骤 207 :结束。

[0150] 综合缓存置换步骤流程图中的步骤 201-207 以及移动云计算迁移方法步骤流程图中的步骤 106,缓存置换步骤流程图中的“ $\dots \rightarrow$  步骤 202  $\rightarrow$  是  $\rightarrow$  步骤 207”、“ $\dots \rightarrow$  步

骤 206 → 步骤 207”在图 2 中的步骤 106 中所得到的结论是对象已经被缓存在云服务器端，即执行步骤 107；而“……→ 步骤 205 → 是 → 步骤 207”在图 2 中的步骤 106 中所得到的结论是对象没有被缓存在云服务器端，即执行步骤 108。

[0151] 以下对本发明优选实施例的移动云计算迁移系统作进一步说明，移动云计算迁移方法实施需要移动云计算迁移系统中的各个模块配合工作才得以完成。移动云计算迁移系统包括移动端和云服务器端。

[0152] 移动端包括调度参数监测模块、代码分析模块、在线决策模块、传输协议模块。

[0153] 调度参数监测模块主要包括能耗模型，通过该模型，主要去获取移动端对于某一个函数的执行的历史能耗和运行时间。当一个函数执行的时候，如果它在移动端执行，把执行完所消耗的电源以及所需要的 CPU 执行时间记录下来，更新到移动端的数据库中；如果在云服务器端执行，则在其执行结果从云服务器端返回结果时把它从开始执行移动端所消耗的能源以及总执行时间记录下来，更新到数据库中，具体数学变量在前述移动云计算迁移方法的第二步中已经详述，在此不再赘述。

[0154] 代码分析模块主要负责对函数代码的分析。当函数第一次执行时，并不清楚该函数是否能迁移到云服务器端、是否需要去访问调用对象的数据。该模块主要借助 java 的异常机制来实现相关判断及决策。移动端通过把函数迁移到云服务器端执行，然后通过云服务器端进行代码分析，并把分析的结果通过传输协议模块返回到移动端，并保存到数据库中，以便给下次执行时提供数据参考。

[0155] 在线决策模块主要负责在移动端函数执行的时候，通过获取代码分析模块的分析数据以及调度参数监测模块对该函数的历史执行数据进行分析，然后通过能耗模型的数学规则对是否需要把该函数迁移到云服务器端执行进行决策。

[0156] 传输协议模块主要负责移动端与云服务器端的数据交互问题，具体协议的格式包括七个字段：第一个字段表示需要在云服务器端开启多少个镜像来并行执行该函数；第二个字段 Flag 是一个标识字段，Flag 为 2 表示当前执行的函数不需要访问它调用对象的内部属性，此时第三个字段就是其调用对象的 Shallow size，Flag 为 0 表示该函数是第一次执行或该函数的调用对象没有缓存到云服务器端，Flag 为 1 表示该函数的调用对象已经缓存在云服务器端，第三个字段为该对象的名称，第四个字段为该对象可能在移动端被修改的属性值，为让云服务器端和移动端的数据保持一致性，必须将已经被修改的属性值更新到云服务器端。如果 Flag 不是 1，则第四个字段将为空。第五个字段是被调用函数的名称；第六个字段表示该函数的参数类型列表；第七个字段表示该函数具体的参数值。传输协议的主要流程是，应用程序发起需迁移的函数到云服务器端执行的时候，根据云服务器端所做的决策填充协议的数据，然后把响应的数据传输到云服务器端，并等待云服务器端的返回结果，再把结果返回给云服务器端进行处理。

[0157] 移动端在移动云计算迁移方法中调度工作的步骤为：一个函数执行时，在线决策模块先通过调度参数监测模块的数据进行分析，如果该函数对象已经缓存到云服务器端，则还需要通过代码分析模块对该函数的调用对象进行分析，并通过上述数据对该函数是否迁移进行决策，如果需要迁移，则通过传输协议模块把所需要的数据通过网络传输到云服务器端，并把执行后的结果返回到传输协议模块，解析后把相应的数据结果传输到调度参数监测模块进行保存，以便为下次执行提供参考。

[0158] 云服务器端包括计算模块、数据缓存模块、调度云执行模块、传输协议模块。

[0159] 计算模块主要用来负责移动端上请求在云服务器端执行计算功能的需求任务,通过用户处理器实现。在程序执行时,该模块需要从系统内存中调用该请求程序的代码库文件,然后根据传输过来的数据状态进行执行。执行结果通过传输协议模块返回到移动端,执行的历史记录会被调度参数监测模块收集,以指导云服务器端数据的缓存置换算法。

[0160] 数据缓存模块主要负责保存并维护移动端上传库文件和执行所需数据,以降低二次调度时的传输代价。在实现过程中,对于程序库文件,在某个应用程序第一次执行的时候,需要把该程序的相关的库文件全部传输到云服务器端并保存。对于函数反复执行所需要的数据,主要通过 IB-LRU 算法来进行缓存管理。

[0161] 调度云执行模块主要负责对移动端的函数是否在云服务器端执行进行调度,此模块主要根据执行的历史记录以及该函数的对象分析数据进行决策。

[0162] 传输协议模块负责移动云计算传输协议的云服务器端部分,接收移动端上传的数据并提交给其他模块,同时将计算结果反馈给移动端。

[0163] 云服务器端在移动云计算迁移方法中调度工作的步骤为:在移动端某个程序需要执行函数时,调度云执行模块根据能获取的数据分析该函数是否在云服务器端执行,当需要请求云服务器端执行时,通过移动端的传输协议模块与云服务器端的传输协议模块进行交互,然后根据请求执行的具体情况与计算模块、数据缓存模块进行交互。

[0164] 另外,在本发明更优选的实施例中,移动云计算迁移方法过程中,移动云计算传输协议的实现主要由安卓平台的网络编程套接字实现,为 C/S 框架。云服务器端首先实现一个 Server 线程,通过不断监听网络端口查看是否有移动端进行请求,如果有,新建一个 ClientSocket 并通过 ClientSocket 与移动端进行数据通信,ClientSocket 把收到的数据传输给云服务器端的用户处理器进行处理,并且把处理结果通过 ClientSocket 返回给移动端。移动端在每次需要与云服务器端进行数据传输时,对云服务器端请求连接,连接建立后把传输协议模块填充好的数据传输到云服务器端,并把执行结果返回到传输协议模块进行相应的处理。

[0165] 而本发明优选实施例中的移动云计算迁移方法的实现也同时为应用程序开发人员提供了一套完整的 API,应用程序开发人员在编写自定义类的时候通过继承 API 中的 Remoteable 类,然后编写开发所需要的功能函数。对于所有继承 Remoteable 的类,函数调用同一通过反射基质,使用 Remoteable 中的 execute(函数名、参数、……) 来实现。通过调用该方法去执行应用程序开发人员自定义函数时,本发明的移动云计算系统会自动去识别该函数是否可迁移,以及是否需要迁移到云服务器端。通过使用本发明的 API,应用程序开发人员不需要去主动注释函数是否可以迁移,从而实现了应用开发和计算迁移的解耦合。

[0166] 进一步地,本发明优选实施例在判断函数是否能够迁移到云服务器端执行过程的具体实施方式如下:当一个应用程序启动的时候,每当它需要去执行一个自定义的函数时,执行控制器会去查询数据库,去查询这个函数的相关信息去判断这个函数是否是可迁移函数,如果不能找到相关的信息,执行控制器默认会把它传输到云服务器端,让云服务器端去判断。首先,执行控制器会把 Flag 置为 0,然后把相关的其余的数据包信息填充完整,然后把它发送到云服务器端。当云服务器端接收到相关的信息的时候,当它判断 Flag 是 0 的时候,会去查询数据库,找寻该函数的相关的信息,如果找不到,则云服务器端就识别该操作

是去做函数识别。然后云服务器端的用户处理器会调用函数分析器去分析这个函数是否是可迁移函数,如果是可迁移函数,则去判断该函数是否需要去访问该函数的调用对象的内部属性,然后把这些信息收集起来(如果不是可迁移函数,则也收集该函数的执行结果)并返回到移动设备端的执行控制器,当收到这些返回信息的时候,执行控制器会去检查该函数是否是可迁移函数,如果是,则此次执行结束,并把该函数的分析数据保存到本地数据库中。如果不是,执行控制器在保存分析数据之后,再在本地执行该函数。

[0167] 更进一步地,如果执行控制器在进行函数识别时能在数据库中找到相关函数信息,应判定该函数是否是可迁移函数。如果不是,则直接在本地执行该函数;否则执行控制器调用能耗模型去计算其能耗优化的相关数据,然后通过分析这些数据判断此次执行是否需要在云服务器端执行。如果不需要,则直接在移动端本地执行;否则还要判断该函数是否需要访问它的调用对象的属性值。如果不需要,则传输它的调用对象的 Shallow Size 到云服务器端执行;否则判断该调用对象是否已经缓存在云服务器端。如果不在,则传输该对象的 Retained Size 到云服务器端;否则用对象探测器判断该对象是否在本地移动端被修改过,这可能导致与云服务器端缓存的对象是不一致。如果不一致,还需要在数据传输协议包中填充被修改的域值。经过上述一系列判断,把填充好的数据包发送到云服务器端,等待云服务器端的执行,并返回结果。

[0168] 以上内容是结合具体的优选实施方式对本发明所作的进一步详细说明,不能认定本发明的具体实施只局限于这些说明。对于本发明所属技术领域的技术人员来说,在不脱离本发明构思的前提下,还可以做出若干等同替代或明显变型,而且性能或用途相同,都应当视为属于本发明的保护范围。

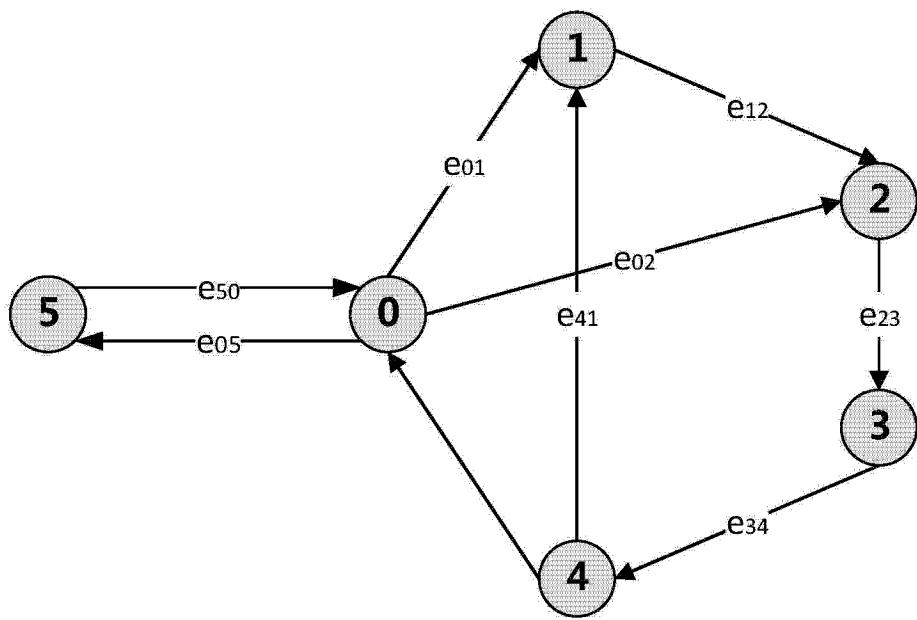


图 1

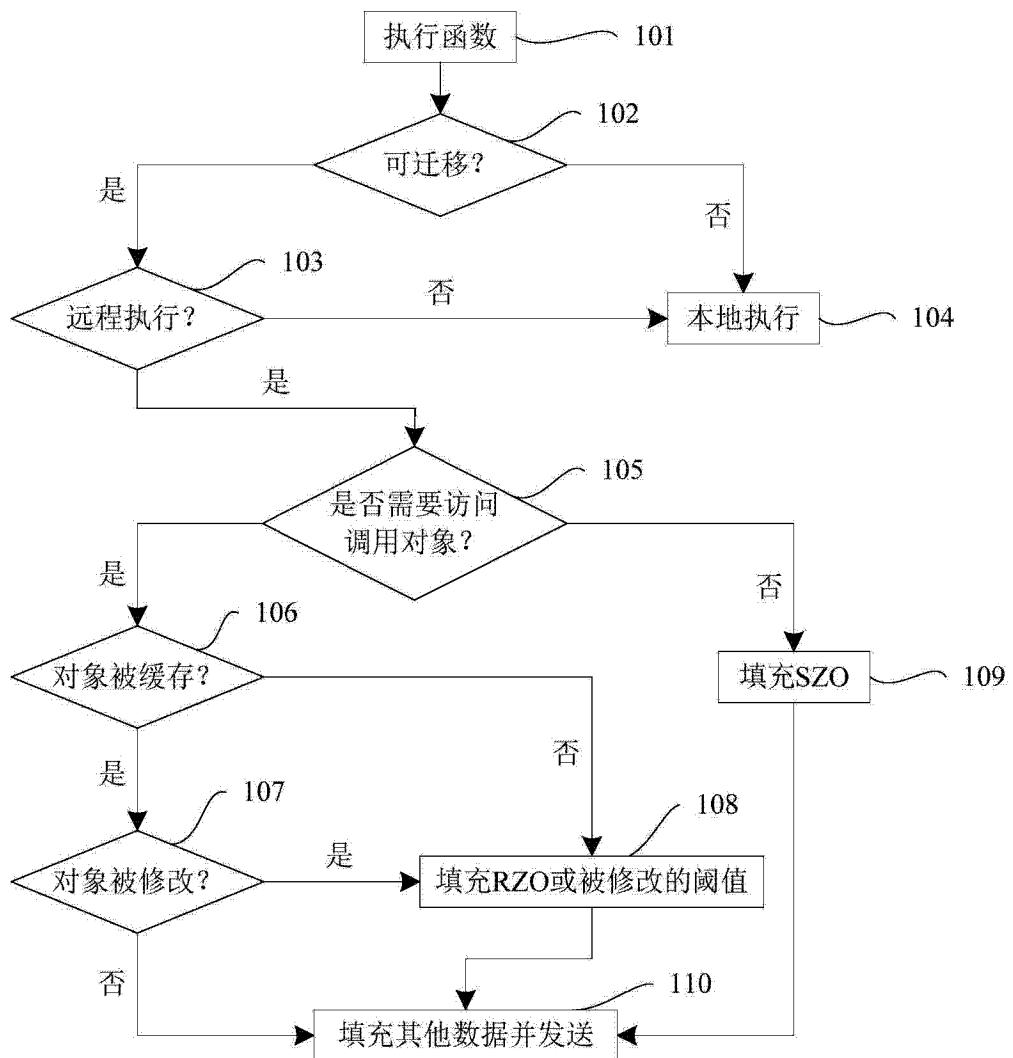


图 2

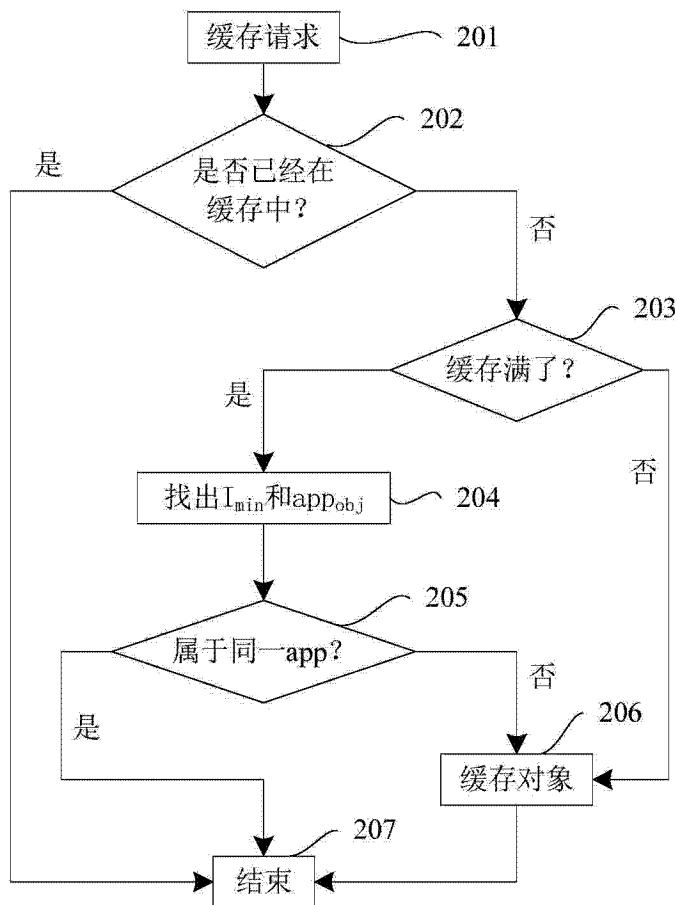


图 3