(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 111176986 B
(45)授权公告日 2023.12.29

(21)申请号 201911292192.7
(22)申请日 2019.12.16
(65)同一申请的已公布的文献号
申请公布号 CN 111176986 A
(43)申请公布日 2020.05.19
(73)专利权人 金蝶软件(中国)有限公司
地址 518000 广东省深圳市南山区科技园科技南十二路2号金蝶软件园A座1-8层
(72)发明人 郑政芳
(74)专利代理机构 华进联合专利商标代理有限公司 44224
专利代理师 方高明
(51)Int.Cl.
*G06F 11/36*(2006.01)
*G06F 9/4401*(2018.01)
(56)对比文件
CN 107948317 A,2018.04.20
CN 109783342 A,2019.05.21
US 9405658 B1,2016.08.02
US 2014095932 A1,2014.04.03
CN 110032512 A,2019.07.19
CN 106254436 A,2016.12.21
审查员 郝玉香

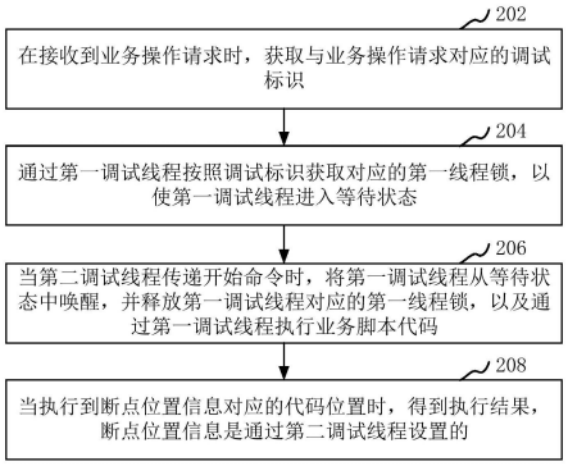
权利要求书2页 说明书11页 附图5页

(54)发明名称

# 线程脚本调试方法、装置、计算机设备和存储介质

(57)摘要

本申请涉及一种线程脚本调试方法、装置、计算机设备和存储介质。该方法包括:在接收到业务操作请求时,获取与业务操作请求对应的调试标识;通过第一调试线程按照调试标识获取对应的第一线程锁,以使第一调试线程进入等待状态;当第二调试线程传递开始命令时,将第一调试线程从等待状态中唤醒,并释放第一调试线程对应的第一线程锁,以及通过第一调试线程执行业务脚本代码;当执行到断点位置信息对应的代码位置时,得到执行结果,断点位置信息是通过第二调试线程设置的。采用本方案能够提高线程脚本调试的安全性。




CN 111176986 B

1.一种线程脚本调试方法,所述方法包括:

在接收到业务操作请求时,获取与所述业务操作请求对应的调试标识;

通过第一调试线程按照所述调试标识获取对应的第一线程锁,以使所述第一调试线程进入等待状态时,触发与所述业务操作请求对应的调试请求,把所述第一调试线程对应的调试线程传递给所述调试请求对应的第二调试线程;

当所述第二调试线程传递开始命令时,将所述第一调试线程从所述等待状态中唤醒,并释放所述第一调试线程对应的第一线程锁,以及

通过所述第一调试线程执行业务脚本代码;

当执行到断点位置信息对应的代码位置时,得到执行结果,所述断点位置信息是通过所述第二调试线程设置的。

2.根据权利要求1所述的方法,其特征在于,所述断点位置信息存储在存储器中,所述存储器中包括至少一个断点位置信息。

3.根据权利要求1所述的方法,其特征在于,在所述当第二调试线程传递开始命令时,将所述第一调试线程从所述等待状态中唤醒之前,所述方法还包括:

通过所述第二调试线程根据所述调试标识获取对应的第二线程锁,以使所述第二调试线程进入等待状态;

当所述第二调试线程接收启动命令时,将所述第二调试线程从所述等待状态中唤醒,并释放所述第二调试线程对应的第二线程锁。

4.根据权利要求1所述方法,其特征在于,所述执行结果通过所述第一调试线程放入存储器中,在所述当执行到断点位置信息对应的代码位置时,得到执行结果之后,所述方法还包括:

通过所述第二调试线程把所述执行结果发送至终端界面进行渲染显示。

5.根据权利要求4所述方法,其特征在于,在所述通过所述第二调试线程把所述执行结果发送至终端界面进行渲染显示之后,所述方法还包括:

获取断点添加指令,所述断点添加指令携带有待添加的断点位置信息;

执行所述断点添加指令,将所述待添加的断点位置信息存储于所述存储器中。

6.根据权利要求1所述方法,其特征在于,所述方法还包括:

接收结束指令;

根据所述结束指令对所述第一调试线程和所述第二调试线程执行关闭操作。

7.一种线程脚本调试装置,其特征在于,所述装置包括:

第一获取模块,用于在接收到业务操作请求时,获取与所述业务操作请求对应的调试标识;

第二获取模块,用于通过第一调试线程按照所述调试标识获取对应的第一线程锁,以使所述第一调试线程进入等待状态时,触发与所述业务操作请求对应的调试请求,把所述第一调试线程对应的调试线程传递给所述调试请求对应的第二调试线程;

释放模块,用于当所述第二调试线程传递开始命令时,将所述第一调试线程从所述等待状态中唤醒,并释放所述第一调试线程对应的第一线程锁,以及所述第一调试线程执行业务脚本代码;

执行模块,用于当执行到断点位置信息对应的代码位置时,得到执行结果,所述断点位

置信息是通过所述第二调试线程设置的。

8.根据权利要求7所述装置,其特征在于,所述装置还包括:

指令添加模块,用于获取断点添加指令,所述断点添加指令携带有待添加的断点位置信息;执行所述断点添加指令,将所述待添加的断点位置信息存储于存储器中。

9.一种计算机设备,包括存储器和处理器,所述存储器中存储有计算机程序,其特征在于,所述处理器执行所述计算机程序时实现权利要求1至6中任一项所述的方法的步骤。

10.一种计算机可读存储介质,其上存储有计算机程序,其特征在于,所述计算机程序被处理器执行时实现权利要求1至6中任一项所述的方法的步骤。

# 线程脚本调试方法、装置、计算机设备和存储介质

## 技术领域

[0001] 本申请涉及调试技术领域,特别是涉及一种线程脚本调试方法、装置、计算机设备和存储介质。

## 背景技术

[0002] 在分布式环境下可进行代码的开发,检查开发的代码是否存在错误,可以通过线程调试来检查。

[0003] 然而,传统的线程调试方法中,调试客户端进行线程脚本调试通过脚本引擎中断解释执行器,进行调试;调试过程中会有多个业务请求,即调试周期中包含多次请求的,在调试过程中导致线程脚本调试受到混用线程环境的干扰,降低线程脚本调试的安全性。

## 发明内容

[0004] 基于此,有必要针对上述技术问题,提供一种能够提高线程脚本调试安全性的线程脚本调试方法、装置、计算机设备和存储介质。

[0005] 一种线程脚本调试方法,所述方法包括:

[0006] 在接收到业务操作请求时,获取与所述业务操作请求对应的调试标识;

[0007] 通过第一调试线程按照所述调试标识获取对应的第一线程锁,以使所述第一调试线程进入等待状态;

[0008] 当第二调试线程传递开始命令时,将所述第一调试线程从所述等待状态中唤醒,并释放所述第一调试线程对应的第一线程锁,以及

[0009] 通过所述第一调试线程执行业务脚本代码;

[0010] 当执行到断点位置信息对应的代码位置时,得到执行结果,所述断点位置信息是通过所述第二调试线程设置的。

[0011] 在其中一个实施例中,所述断点位置信息存储在存储器中,所述存储器中包括至少一个断点位置信息。

[0012] 在其中一个实施例中,所述断点位置信息存储在存储器中,所述存储器中包括至少一个断点位置信息。

[0013] 在其中一个实施例中,在所述当第二调试线程传递开始命令时,将所述第一调试线程从所述等待状态中唤醒之前,所述方法还包括:

[0014] 通过所述第二调试线程根据所述调试标识获取对应的第二线程锁,以使所述第二调试线程进入等待状态;

[0015] 当所述第二调试线程接收启动命令时,将所述第二调试线程从所述等待状态中唤醒,并释放所述第二调试线程对应的第二线程锁。

[0016] 在其中一个实施例中,所述执行结果通过所述第一调试线程放入存储器中,在所述当执行到断点位置信息对应的代码位置时,得到执行结果之后,所述方法还包括:

[0017] 通过所述第二调试线程把所述执行结果发送至终端界面进行渲染显示。

[0018] 在其中一个实施例中,在所述通过所述第二调试线程把所述执行结果发送至终端界面进行渲染显示之后,所述方法还包括:

[0019] 获取断点添加指令,所述断点添加指令携带有待添加的断点位置信息;

[0020] 执行所述断点添加指令,将所述待添加的断点位置信息存储于所述存储器中。

[0021] 在其中一个实施例中,所述方法还包括:

[0022] 接收结束指令;

[0023] 根据所述结束指令对所述第一调试线程和所述第二调试线程执行关闭操作。

[0024] 一种线程脚本调试装置,所述装置包括:

[0025] 第一获取模块,用于在接收到业务操作请求时,获取与所述业务操作请求对应的调试标识;

[0026] 第二获取模块,用于通过第一调试线程按照所述调试标识获取对应的第一线程锁,以使所述第一调试线程进入等待状态;

[0027] 释放模块,用于当第二调试线程传递开始命令时,将所述第一调试线程从所述等待状态中唤醒,并释放所述第一调试线程对应的第一线程锁,以及所述第一调试线程执行业务脚本代码;

[0028] 执行模块,用于当执行到断点位置信息对应的代码位置时,得到执行结果,所述断点位置信息是通过所述第二调试线程设置的。

[0029] 在一个实施例中,装置还包括:

[0030] 指令添加模块,用于获取断点添加指令,所述断点添加指令携带有待添加的断点位置信息;执行所述断点添加指令,将所述待添加的断点位置信息存储于所述存储器中。

[0031] 一种计算机设备,包括存储器和处理器,所述存储器存储有计算机程序,所述处理器执行所述计算机程序时实现以下步骤:

[0032] 在接收到业务操作请求时,获取与所述业务操作请求对应的调试标识;

[0033] 通过第一调试线程按照所述调试标识获取对应的第一线程锁,以使所述第一调试线程进入等待状态;

[0034] 当第二调试线程传递开始命令时,将所述第一调试线程从所述等待状态中唤醒,并释放所述第一调试线程对应的第一线程锁,以及

[0035] 通过所述第一调试线程执行业务脚本代码;

[0036] 当执行到断点位置信息对应的代码位置时,得到执行结果,所述断点位置信息是通过所述第二调试线程设置的。

[0037] 一种计算机可读存储介质,其上存储有计算机程序,所述计算机程序被处理器执行时实现以下步骤:

[0038] 在接收到业务操作请求时,获取与所述业务操作请求对应的调试标识;

[0039] 通过第一调试线程按照所述调试标识获取对应的第一线程锁,以使所述第一调试线程进入等待状态;

[0040] 当第二调试线程传递开始命令时,将所述第一调试线程从所述等待状态中唤醒,并释放所述第一调试线程对应的第一线程锁,以及

[0041] 通过所述第一调试线程执行业务脚本代码;

[0042] 当执行到断点位置信息对应的代码位置时,得到执行结果,所述断点位置信息是

通过所述第二调试线程设置的。

[0043] 上述线程脚本调试方法、装置、计算机设备和存储介质，在接收到业务操作请求时，获取与业务操作请求对应的调试标识；通过第一调试线程按照调试标识获取对应的第一线程锁，以使第一调试线程进入等待状态；当第二调试线程传递开始命令时，将第一调试线程从等待状态中唤醒，并释放第一调试线程对应的第一线程锁，以及通过第一调试线程执行业务脚本代码；当执行到断点位置信息对应的代码位置时，得到执行结果，断点位置信息是通过第二调试线程设置的。第一调试线程只有在获取调试标识对应的第一线程锁后进入等待状态，且调试标识与第一线程锁之间存在对应关系，第一调试线通过第二调试线程唤醒后执行业务脚本代码，即需要通过第一调试线和第二调试线之间的协调交互，执行到断点位置信息对应的代码位置时得到执行结果，完成脚本调试，避免其它业务请求对应线程的干扰，提高了线程脚本调试间的安全性。

## 附图说明

[0044] 图1为一个实施例中线程脚本调试方法的应用环境图；

[0045] 图2为一个实施例中线程脚本调试方法的流程示意图；

[0046] 图3为一个实施例中线程脚本调试原理框图；

[0047] 图4为另一个实施例中线程脚本调试方法的流程示意图；

[0048] 图5为一个实施例中线程脚本调试装置的结构框图；

[0049] 图6为另一个实施例中线程脚本调试装置的结构框图；

[0050] 图7为一个实施例中计算机设备的内部结构图。

## 具体实施方式

[0051] 为了使本申请的目的、技术方案及优点更加清楚明白，以下结合附图及实施例，对本申请进行进一步详细说明。应当理解，此处描述的具体实施例仅仅用以解释本申请，并不用于限定本申请。

[0052] 本申请提供的线程脚本调试方法，可以应用于如图1所示的应用环境中。其中，终端102通过网络与服务器104通过网络进行通信。服务器104接收到终端102到业务操作请求时，获取与所述业务操作请求对应的调试标识；通过第一调试线程按照所述调试标识获取对应的第一线程锁，以使所述第一调试线程进入等待状态；当第二调试线程传递开始命令时，将所述第一调试线程从所述等待状态中唤醒，并释放所述第一调试线程对应的第一线程锁，以及通过所述第一调试线程执行业务脚本代码；当执行到断点位置信息对应的代码位置时，得到执行结果，所述断点位置信息是通过所述第二调试线程设置的。其中，终端102可以但不限于是各种个人计算机、笔记本电脑、智能手机、平板电脑，服务器104可以用独立的服务器或者是多个服务器组成的服务器集群来实现。

[0053] 可以理解，本申请所使用的术语“第一”、“第二”等可在本文中用于描述各种调试线程，但这些调试线程不受这些术语限制。这些术语仅用于将第一个调试线程与另一个调试线程区分。举例来说，在不脱离本申请的范围的情况下，可以将第一调试线程称为第二调试线程，且类似地，可将第二调试线程称为第一调试线程。第一调试线程和第二调试线程两者都是调试线程，但其不是同一个调试线程。

[0054] 在一个实施例中,如图2所示,提供了一种线程脚本调试方法,以该方法应用于图1中的服务器104为例进行说明,包括以下步骤:

[0055] 步骤202,在接收到业务操作请求时,获取与业务操作请求对应的调试标识。

[0056] 其中,业务操作请求是用来获取业务数据的一种请求,业务操作请求可以是一种测试请求。例如,在服务器中新建数据表,要测试新建数据表中保存的数据的准确性,终端向服务器发出测试数据准确性的测试请求。终端可以向服务器同时发起多个业务操作请求,每一个业务操作请求对应的调试标识不同,在一个业务操作过程中,每个业务操作请求会存在至少一次请求,每次请求对应的调试标识是相同的。

[0057] 调试标识是用于标识业务操作请求对应请求线程的标识,不同的操作请求可对应于不同的请求线程,调试标识与请求线程之间存在对应关系,调试标识用于标识当前业务操作请求对应的请求线程。例如调试标识可以是一连串数字组合或者字母组合、数字字母组合的字符串等。例如,当服务器接收到业务操作请求A时,获取与业务操作请求A(Thread A)对应的请求线程1的调试标识1,调试标识1为“Thread A id=123456”。在此对调试标识的形式不做限定。

[0058] 具体地,在启动调试模式(Debugger)下,当服务器接收到终端发送的业务操作请求时,业务操作请求对应的响应信息为预设响应信息时,生成或调用与所接收的业务操作请求对应的请求线程,获取与业务操作请求对应请求线程的调试标识,并通过请求线程把业务操作请求接入对应的调试线程。其中,响应信息是服务器对接收终端的业务操作请求的处理后得到的处理结果;服务器可以通过响应协议对终端的业务操作请求进行处理;接收的响应信息的形式可以是二进制数值,例如,返回的响应消息可以是0000或0001。

[0059] 步骤204,通过第一调试线程按照调试标识获取对应的第一线程锁,以使第一调试线程进入等待状态。

[0060] 其中,线程锁是线程访问共享资源的权限。例如,有十个线程要访问同一个数据库,数据库有一个对应线程锁,其中一个线程获取线程锁后,可以访问数据库,其他九个线程要等到有线程锁的这个线程释放锁之后,获取到线程锁才能访问数据库。

[0061] 调试线程是在通过容器选择的调试(debugger)模式下,根据接收的调试指令对调试任务依次进行处理的处理单元。例如,终端在接收到数据库是否连接成功的测试指令时,执行对应的测试代码,包括检测连接的数据库名称是否是目标数据库名称、获取的登录密码是否是预设的登录密码等。容器启动需要启动一个对应的进程,一个进程可以包括多个调试线程,例如,在debugger模式下,调试线程可以包括第一调试线程、第二调试线程,第一调试线程可用于获取脚本文件中的数据,第二调试线程可以用于执行对应的线程脚本,在此,对第一调试线程和第二调试线程不做具体限定。其中,容器(Docker)是一个开源的应用引擎容器引擎,可以把开发的应用以及依赖包到一个可移植的存储镜像(registry)的地方后发布到终端,在分布式环境下容器中可以选择debugge模式和非debugger模式。

[0062] 具体地,在开发过程中选择容器中debugger模式,通过请求线程把业务操作请求接入对应的第一调试线程,获取业务操作请求对应的调试标识;通过第一调试线程根据调试标识获取对应的第一线程锁,以使第一调试线程进入等待状态。

[0063] 步骤206,当第二调试线程传递开始命令时,将第一调试线程从等待状态中唤醒,并释放第一调试线程对应的第一线程锁,以及通过第一调试线程执行业务脚本代码。

[0064] 其中,第二调试线程是在debugger模式下与第一调试线程对应的调试线程,第一调试线程的调试标识与第二调试线程对应的调试标识相同。

[0065] 具体地,在debugger模式下,当第一调试线程按照调试标识获取对应的第一线程锁,以使第一调试线程进入等待状态时,触发与业务操作请求对应的调试请求,并把第一调试线程对应的调试线程传递给调试操作请求对应的第二调试线程,当第二调试线程传递开始命令时,将第一调试线程从等待状态中唤醒,并释放第一调试线程对应的第一线程锁,以及通过第一调试线程执行业务脚本代码;其中,开始命令可通过终端用户界面输入开始控制指令。例如,第一调试线程为线程A,第二调试线程为线程B,当线程A获取对应的第一线程锁(lock)时,进入等待wait状态,线程B获取与线程A对应的调试标识,当线程B执行Inspect开始命令时,线程B给线程A注入需要callable将线程A从等待状态中唤醒,并释放线程A对应的第一线程锁。

[0066] 步骤208,当执行到断点位置信息对应的代码位置时,得到执行结果,断点位置信息是通过第二调试线程设置的。

[0067] 其中,断点是程序代码的追踪点,断点位置信息是追踪点在程序代码中的具体位置。例如,为了观察代码第10行和第30行上变量或表达式的值,在代码的第10行和第30行分别设置断点,即,第10行和第30行就是断点位置信息。一个程序的代码可以设置多个断点,也就是说可以有多个断点位置信息。断点位置信息可以通过单击添加断点按钮,双击目标断点位置添加,也可以通过ADD指令添加断点。断点位置信息可以通过第二调试线程获取调试界面输入的断定位置信息设置指令。

[0068] 具体地,当第二调试线程传递开始命令时,将第一调试线程从等待状态中唤醒,并释放第一调试线程对应的第一线程锁,以及通过第一调试线程执行业务脚本代码,第二调试线程获取对应的线程锁,进入等待状态;当第一调试线程执行到断点位置信息对应的代码位置时,得到执行结果,并把执行结果放入共享缓存中,第一调试线程进入等待状态,第二调试线程被唤醒;其中,执行结果可为当前帧信息。例如,第一调试线程执行对应的Thread类中的run()方法,获取业务脚本代码的断点位置信息,根据断点位置信息执行线程脚本得到执行结果并结束;当存在多个断点位置信息,可以通过第二调试线程获取执行指令,例如stepOver,继续执行下一行,直到执行完所接受的断点位置信息结束或接受终止指令后结束。

[0069] 上述线程脚本调试方法中,通过在接收到业务操作请求时,获取与业务操作请求对应的调试标识;通过第一调试线程按照调试标识获取对应的第一线程锁,以使第一调试线程进入等待状态;当第二调试线程传递开始命令时,将第一调试线程从等待状态中唤醒,并释放第一调试线程对应的第一线程锁,以及通过第一调试线程执行业务脚本代码;当执行到断点位置信息对应的代码位置时,得到执行结果,断点位置信息是通过第二调试线程设置的。第一调试线程只有在获取调试标识对应的第一线程锁后进入等待状态,且调试标识与第一线程锁之间存在对应关系,第一调试线通过第二调试线程唤醒后执行业务脚本代码,即需要通过第一调试线和第二调试线之间的协调交互,执行到断点位置信息对应的代码位置时得到执行结果,完成脚本调试,避免其它业务请求对应线程的干扰,提高了线程脚本调试的安全性。

[0070] 在一个实施例中,断点位置信息存储在存储器中,该存储器中包括至少一个断点

位置信息。

[0071] 其中,存储器可用于存储程序和数据,存储器可以是缓冲存储器、寄存器、主存储器。通过设置多个断点位置信息,第一调试线程可以听过执行到每个断点位置信息对于的业务脚本代码获取的对应执行结果,提高线程调试的准确性。

[0072] 在一个实施例中,执行结果通过第一调试线程放入存储器中,在当执行到断点位置信息对应的代码位置时,得到执行结果之后,方法还包括:

[0073] 通过第二调试线程把执行结果发送至终端界面进行渲染显示。

[0074] 其中,执行结果可以是业务脚本代码中,函数变量在执行结果中的数值或者函数表达式在执行结果中的实际值。例如,变量a=5,a+b=8。

[0075] 具体地,第二调试线程被唤醒,通过交互接口从共享缓存中读取执行结果,并把执行结果发送至终端界面进行渲染显示;其中,交互接口是用于传输数据的传输介质,传输的数据可以是断点位置信息、返回线程脚本的执行结果、调用的函数方法等。通过对执行结果进行渲染显示可以对调试结果进行检验,提高线程脚本调试的准确性。

[0076] 在一个实施例中,在接收到业务操作请求时,获取与业务操作请求对应的调试标识;通过第一调试线程按照调试标识获取对应的第一线程锁,以使第一调试线程被挂起进入等待状状态;当第二调试线程传递开始命令时,将第一调试线程从等待状态中唤醒,第一调试线程释放并释放对应的第一线程锁,以及通过第一调试线程从存储器中获取断点位置信息后,执行业务脚本代码;当执行到断点位置信息对应的代码位置时,得到执行结果,通过第二调试线程把执行结果发送至终端界面进行渲染显示,第二调试线程进入等待状态,等待接受下一个执行命令,其中,执行命令可以是恢复执行、执行下一行、跟踪进函数、指定执行到第n行、设置断点、查看表达式、停止等指令。

[0077] 在一个实施例中,在通过第二调试线程把执行结果发送至终端界面进行渲染显示之后,方法还包括:

[0078] 获取断点添加指令,断点添加指令携带有待添加的断点位置信息;执行断点添加指令,将待添加的断点位置信息存储于存储器中。

[0079] 具体地,断点添加指令用于通过调试交互界面输入断点位置信息添加指令。当获取到调试界面输入的断点添加指令时,执行断点添加指令,把添加的断点位置信息存储在存储器中,通过断点添加指令添加断点位置信息,可以针对调试过程中出现的异常错误进行调试,提高线程调试的效率。

[0080] 在一个实施例中,如图3所示,提供了一种线程脚本调试原理框图,包括业务操作请求302、请求线程304、第一调试线程306、缓存308、第二调试线程310、调试请求线程312、调试命令请求314。

[0081] 服务器请求线程304接收业务操作请求302后,并获取对应的调试标识;把业务操作请求302接入第一调试线程306,第一调试线程306获取调试标识对应的第一线程锁,第一调试线程306进入等待状态;在接收业务操作请求302后,触发业务操作请求302对应的调试命令请求314的调试请求线程312,并把请求线程304对应的调试标识传递给调试请求线程312,当第二调试线程310传递开始命令时,将第一调试线程306从等待状态中唤醒,并释放第一调试线程306程对应的第一线程锁,,第一调试线程306从缓存308中读取调试的断点位置信息,通过第一调试线程306执行业务脚本代码,当执行到断点位置信息对应的代码位置

时,得到执行结果;通过第二调试线程把所述执行结果发送至终端界面进行渲染显示其中,请求线程304同时可以接收至少一个业务请求操作,调试请求线程312同时可以接收至少一个调试请求操作。通过第一调试线程306和第二调试线程310之间协调交互,完成脚本调试,避免其它业务操作请求对应线程的干扰,提高了线程脚本调试的隔离性和安全性。

[0082] 在另一个实施例中,如图4所示,提供了一种线程脚本调试方法,以该方法应用于图1中的终端102为例进行说明,包括以下步骤:

[0083] 步骤402,在接收到业务操作请求时,获取与业务操作请求对应的调试标识。

[0084] 步骤404,通过第一调试线程按照调试标识获取对应的第一线程锁,以使第一调试线程进入等待状态。

[0085] 步骤406,通过第二调试线程根据调试标识获取对应的第二线程锁,以使第二调试线程进入等待状态。

[0086] 步骤408,当第二调试线程接收启动命令时,将第二调试线程从等待状态中唤醒,并释放第二调试线程对应的第二线程锁。

[0087] 步骤410,当第二调试线程传递开始命令时,将第一调试线程从等待状态中唤醒,并释放第一调试线程对应的第一线程锁,以及通过第一调试线程执行业务脚本代码。

[0088] 步骤412,当执行到断点位置信息对应的代码位置时,得到执行结果,断点位置信息是通过第二调试线程设置的。

[0089] 上述线程脚本调试方法中,接收到业务操作请求时,获取与业务操作请求对应的调试标识;通过第一调试线程按照调试标识获取对应的第一线程锁,以使第一调试线程进入等待状态;当第二调试线程传递开始命令时,将第一调试线程从等待状态中唤醒,并释放第一调试线程对应的第一线程锁,以及通过第一调试线程执行业务脚本代码;当执行到断点位置信息对应的代码位置时,得到执行结果,断点位置信息是通过第二调试线程设置的。第一调试线程只有在获取调试标识对应的第一线程锁后进入等待状态,且调试标识与第一线程锁之间存在对应关系,第一调试线程通过第二调试线程唤醒后执行业务脚本代码,即需要通过第一调试线程和第二调试线程之间的协调交互,执行到断点位置信息对应的代码位置时得到执行结果,完成脚本调试;第二调试线程与第一调试线程之间通过交互接口传输断点位置信息,不需要通过端口传输断点位置信息,也避免其它线程的干扰,提高了线程脚本调试的隔离性和安全性。

[0090] 在一个实施例中,方法还包括:

[0091] 接收结束指令;根据结束指令对第一调试线程和第二调试线程执行关闭操作。

[0092] 其中,结束指令用于结束本次线程调试。例如,结束指令可为stop指令。

[0093] 具体地,当第二调试线程结束指令传递给第一调试线程后,移除存储在缓存中的断点位置信息;第一调试线程屏蔽断点位置信息,执行完业务脚本代码得到执行结果,通过第二调试线程把执行结果发送至终端界面进行渲染显示,关闭第一调试线程和第二调试线程,避免其它线程的干扰,影响执行结果的准确性。

[0094] 在一个实施例中,第一调试线程为Main-Debug-Thread,第二调试线程为CMD-Debug-Thread。在调试模式下,服务器的请求服务线程接收到业务操作请求时,获取与业务操作请求对应的调试标识;通过请求服务线程把业务操作请求接入线程Main-Debug-Thread中,线程Main-Debug-Thread根据业务操作请求对应的调试标识获取对应的第一线

程锁,并进入等待状态;当服务器的请求服务线程接收业务操作请求对应的调试操作请求时,通过请求服务线程把调试操作请求接入线程CMD-Debug-Thread,CMD-Debug-Thread获取业务操作请求对应的调试标识并获取调试标识对应的第二线程锁,进入等待状态;当CMD-Debug-Thread执行启动操作时,CMD-Debug-Thread由等待状态进入唤醒状态,将Main-Debug-Thread从等待状态中唤醒,并释放Main-Debug-Thread对应的第一线程锁,Main-Debug-Thread开始执行业务脚本代码,当Main-Debug-Thread执行到断点位置信息对应的代码位置时,得到执行结果,把执行结果放入缓存中;Main-Debug-Thread由唤醒状态进入等待状态,CMD-Debug-Thread从缓存中获取执行结果,并将执行结果发送至终端界面进行渲染显示。

[0095] 应该理解的是,虽然图2-4的流程图中的各个步骤按照箭头的指示依次显示,但是这些步骤并不是必然按照箭头指示的顺序依次执行。除非本文中有明确的说明,这些步骤的执行并没有严格的顺序限制,这些步骤可以以其它的顺序执行。而且,图2-4中的至少一部分步骤可以包括多个子步骤或者多个阶段,这些子步骤或者阶段并不必然是在同一时刻执行完成,而是可以在不同的时刻执行,这些子步骤或者阶段的执行顺序也不必然是依次进行,而是可以与其它步骤或者其它步骤的子步骤或者阶段的至少一部分轮流或者交替地执行。

[0096] 在一个实施例中,如图5所示,提供了一种线程脚本调试装置500,包括:第一获取模块502、第二获取模块504、锁释放模块506和执行模块508,其中:

[0097] 第一获取模块502,用于在接收到业务操作请求时,获取与业务操作请求对应的调试标识。

[0098] 第二获取模块504,用于通过第一调试线程按照调试标识获取对应的第一线程锁,以使第一调试线程进入等待状态。

[0099] 在一个实施例中,第二获取模块504还用于通过第二调试线程根据调试标识获取对应的第二线程锁,以使第二调试线程进入等待状态。

[0100] 释放模块506,用于当第二调试线程传递开始命令时,将第一调试线程从等待状态中唤醒,并释放第一调试线程对应的第一线程锁,以及第一调试线程执行业务脚本代码。

[0101] 在一个实施例中,释放模块506还用于当第二调试线程接收启动命令时,将第二调试线程从等待状态中唤醒,并释放第二调试线程对应的第二线程锁。

[0102] 执行模块508,用于当执行到断点位置信息对应的代码位置时,得到执行结果,断点位置信息是通过第二调试线程设置的。

[0103] 在上述线程脚本调试装置中,通过在接收到业务操作请求时,获取与业务操作请求对应的调试标识;通过第一调试线程按照调试标识获取对应的第一线程锁,以使第一调试线程进入等待状态;当第二调试线程传递开始命令时,将第一调试线程从等待状态中唤醒,并释放第一调试线程对应的第一线程锁,以及通过第一调试线程执行业务脚本代码;当执行到断点位置信息对应的代码位置时,得到执行结果。断点位置信息是通过第二调试线程设置的,第一调试线程只有在获取调试标识对应的第一线程锁后进入等待状态,且调试标识与第一线程锁之间存在对应关系,第一调试线通过第二调试线程唤醒后执行业务脚本代码,即需要通过第一调试线和第二调试线之间的协调交互,执行到断点位置信息对应的代码位置时得到执行结果,完成脚本调试,避免其它业务请求对应线程的干扰,且线程之间

交互不需要暴露调试端口;提高了线程脚本调试的隔离性和安全性。

[0104] 在另一个实施例中,如图6所示,提供了一种线程脚本调试装置500,除包括第一获取模块502、第二获取模块504、锁释放模块506和执行模块508,还包括:指令添加模块510、渲染模块512和接收模块514,其中:

[0105] 指令添加模块510,用于获取断点添加指令,断点添加指令携带有待添加的断点位置信息;执行断点添加指令,将待添加的断点位置信息存储于存储器中。

[0106] 具体地,调试过程中出现的错误,通过断点添加指令添加对应断点,根据新添加的断点位置信息及时处理错误,提高线程调试的效率。

[0107] 渲染模块512,用于通过第二调试线程把执行结果发送至终端界面进行渲染显示。

[0108] 接收模块514,用于接收结束指令;根据结束指令对第一调试线程和第二调试线程执行关闭操作。

[0109] 关于线程脚本调试装置的具体限定可以参见上文中对于线程脚本调试方法的限定,在此不再赘述。上述线程脚本调试装置中的各个模块可全部或部分通过软件、硬件及其组合来实现。上述各模块可以硬件形式内嵌于或独立于计算机设备中的处理器中,也可以以软件形式存储于计算机设备中的存储器中,以便于处理器调用执行以上各个模块对应的操作。

[0110] 在一个实施例中,提供了一种计算机设备,该计算机设备可以是终端,其内部结构图可以如图7所示。该计算机设备包括通过系统总线连接的处理器、存储器、网络接口、显示屏和输入装置。其中,该计算机设备的处理器用于提供计算和控制能力。该计算机设备的存储器包括非易失性存储介质、内存储器。该非易失性存储介质存储有操作系统和计算机程序。该内存储器为非易失性存储介质中的操作系统和计算机程序的运行提供环境。该计算机设备的网络接口用于与外部的终端通过网络连接通信。该计算机程序被处理器执行时以实现一种线程脚本调试方法。该计算机设备的显示屏可以是液晶显示屏或者电子墨水显示屏,该计算机设备的输入装置可以是显示屏上覆盖的触摸层,也可以是计算机设备外壳上设置的按键、轨迹球或触控板,还可以是外接的键盘、触控板或鼠标等。

[0111] 本领域技术人员可以理解,图7中示出的结构,仅仅是与本申请方案相关的部分结构的框图,并不构成对本申请方案所应用于其上的计算机设备的限定,具体的计算机设备可以包括比图中所示更多或更少的部件,或者组合某些部件,或者具有不同的部件布置。

[0112] 在一个实施例中,提供了一种计算机设备,包括存储器和处理器,存储器中存储有计算机程序,该处理器执行计算机程序时实现以下步骤:

[0113] 在接收到业务操作请求时,获取与业务操作请求对应的调试标识;

[0114] 通过第一调试线程按照调试标识获取对应的第一线程锁,以使第一调试线程进入等待状态;

[0115] 当第二调试线程传递开始命令时,将第一调试线程从等待状态中唤醒,并释放第一调试线程对应的第一线程锁,以及

[0116] 通过第一调试线程执行业务脚本代码;

[0117] 当执行到断点位置信息对应的代码位置时,得到执行结果,断点位置信息是通过第二调试线程设置的。

[0118] 在一个实施例中,处理器执行计算机程序时还实现以下:

[0119] 断点位置信息存储在存储器中,存储器中包括至少一个断点位置信息。

[0120] 在一个实施例中,处理器执行计算机程序时还实现以下步骤:

[0121] 在当第二调试线程传递开始命令时,将第一调试线程从等待状态中唤醒之前,方法还包括:

[0122] 通过第二调试线程根据调试标识获取对应的第二线程锁,以使第二调试线程进入等待状态;

[0123] 当第二调试线程接收启动命令时,将第二调试线程从等待状态中唤醒,并释放第二调试线程对应的第二线程锁。

[0124] 在一个实施例中,处理器执行计算机程序时还实现以下步骤:

[0125] 执行结果通过第一调试线程放入存储器中,在当执行到断点位置信息对应的代码位置时,得到执行结果之后,方法还包括:

[0126] 通过第二调试线程把执行结果发送至终端界面进行渲染显示。

[0127] 在一个实施例中,处理器执行计算机程序时还实现以下步骤:

[0128] 在通过第二调试线程把执行结果发送至终端界面进行渲染显示之后,方法还包括:

[0129] 获取断点添加指令,断点添加指令携带有待添加的断点位置信息;

[0130] 执行断点添加指令,将待添加的断点位置信息存储于存储器中。

[0131] 在一个实施例中,处理器执行计算机程序时还实现以下步骤:

[0132] 接收结束指令;

[0133] 根据结束指令对第一调试线程和第二调试线程执行关闭操作。

[0134] 在一个实施例中,提供了一种计算机可读存储介质,其上存储有计算机程序,计算机程序被处理器执行时实现以下步骤:

[0135] 在接收到业务操作请求时,获取与业务操作请求对应的调试标识;

[0136] 通过第一调试线程按照调试标识获取对应的第一线程锁,以使第一调试线程进入等待状态;

[0137] 当第二调试线程传递开始命令时,将第一调试线程从等待状态中唤醒,并释放第一调试线程对应的第一线程锁,以及

[0138] 通过第一调试线程执行业务脚本代码;

[0139] 当执行到断点位置信息对应的代码位置时,得到执行结果,断点位置信息是通过第二调试线程设置的。

[0140] 在一个实施例中,计算机程序被处理器执行时还实现以下:

[0141] 断点位置信息存储在存储器中,存储器中包括至少一个断点位置信息。

[0142] 在一个实施例中,计算机程序被处理器执行时还实现以下步骤:

[0143] 在当第二调试线程传递开始命令时,将第一调试线程从等待状态中唤醒之前,方法还包括:

[0144] 通过第二调试线程根据调试标识获取对应的第二线程锁,以使第二调试线程进入等待状态;

[0145] 当第二调试线程接收启动命令时,将第二调试线程从等待状态中唤醒,并释放第二调试线程对应的第二线程锁。

[0146] 在一个实施例中,计算机程序被处理器执行时还实现以下步骤:

[0147] 执行结果通过第一调试线程放入存储器中,在当执行到断点位置信息对应的代码位置时,得到执行结果之后,方法还包括:

[0148] 通过第二调试线程把执行结果发送至终端界面进行渲染显示。

[0149] 在一个实施例中,计算机程序被处理器执行时还实现以下步骤:

[0150] 在通过第二调试线程把执行结果发送至终端界面进行渲染显示之后,方法还包括:

[0151] 获取断点添加指令,断点添加指令携带有待添加的断点位置信息;

[0152] 执行断点添加指令,将待添加的断点位置信息存储于存储器中。

[0153] 在一个实施例中,计算机程序被处理器执行时还实现以下步骤:

[0154] 接收结束指令;

[0155] 根据结束指令对第一调试线程和第二调试线程执行关闭操作。

[0156] 本领域普通技术人员可以理解实现上述实施例方法中的全部或部分流程,是可以通过计算机程序来指令相关的硬件来完成,所述的计算机程序可存储于一非易失性计算机可读取存储介质中,该计算机程序在执行时,可包括如上述各方法的实施例的流程。其中,本申请所提供的各实施例中所使用的对存储器、存储、数据库或其它介质的任何引用,均可包括非易失性和/或易失性存储器。非易失性存储器可包括只读存储器(ROM)、可编程ROM(PROM)、电可编程ROM(EPROM)、电可擦除可编程ROM(EEPROM)或闪存。易失性存储器可包括随机存取存储器(RAM)或者外部高速缓冲存储器。作为说明而非局限,RAM以多种形式可得,诸如静态RAM(SRAM)、动态RAM(DRAM)、同步DRAM(SDRAM)、双数据率SDRAM(DDRSDRAM)、增强型SDRAM(ESDRAM)、同步链路(Synchlink)DRAM(SLDRAM)、存储器总线(Rambus)直接RAM(RDRAM)、直接存储器总线动态RAM(DRDRAM)、以及存储器总线动态RAM(RDRAM)等。

[0157] 以上实施例的各技术特征可以进行任意的组合,为使描述简洁,未对上述实施例中的各个技术特征所有可能的组合都进行描述,然而,只要这些技术特征的组合不存在矛盾,都应当认为是本说明书记载的范围。

[0158] 以上所述实施例仅表达了本申请的几种实施方式,其描述较为具体和详细,但并不能因此而理解为对发明专利范围的限制。应当指出的是,对于本领域的普通技术人员来说,在不脱离本申请构思的前提下,还可以做出若干变形和改进,这些都属于本申请的保护范围。因此,本申请专利的保护范围应以所附权利要求为准。

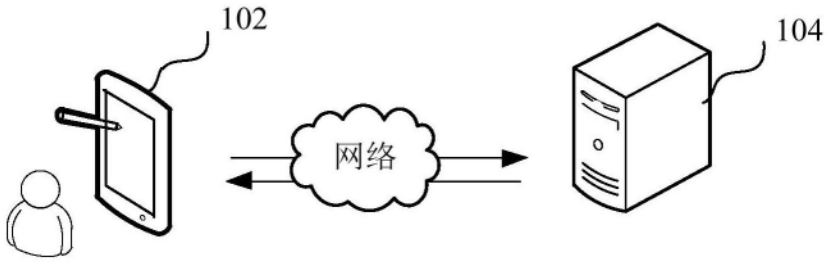

102
104
网络

图1

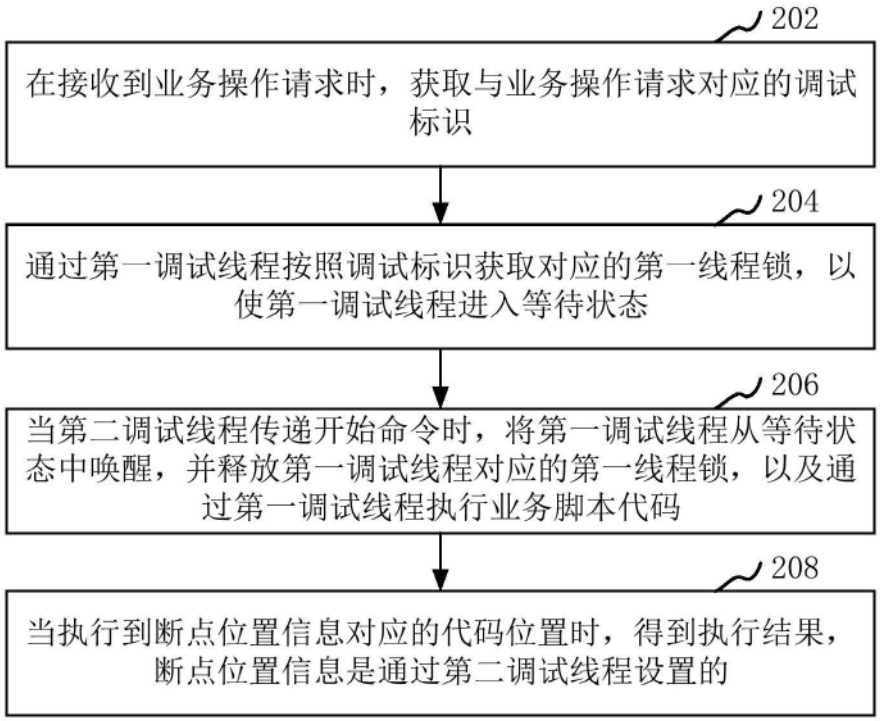

202
在接收到业务操作请求时，获取与业务操作请求对应的调试标识
204
通过第一调试线程按照调试标识获取对应的第一线程锁，以使第一调试线程进入等待状态
206
当第二调试线程传递开始命令时，将第一调试线程从等待状态中唤醒，并释放第一调试线程对应的第一线程锁，以及通过第一调试线程执行业务脚本代码
208
当执行到断点位置信息对应的代码位置时，得到执行结果，断点位置信息是通过第二调试线程设置的

图2

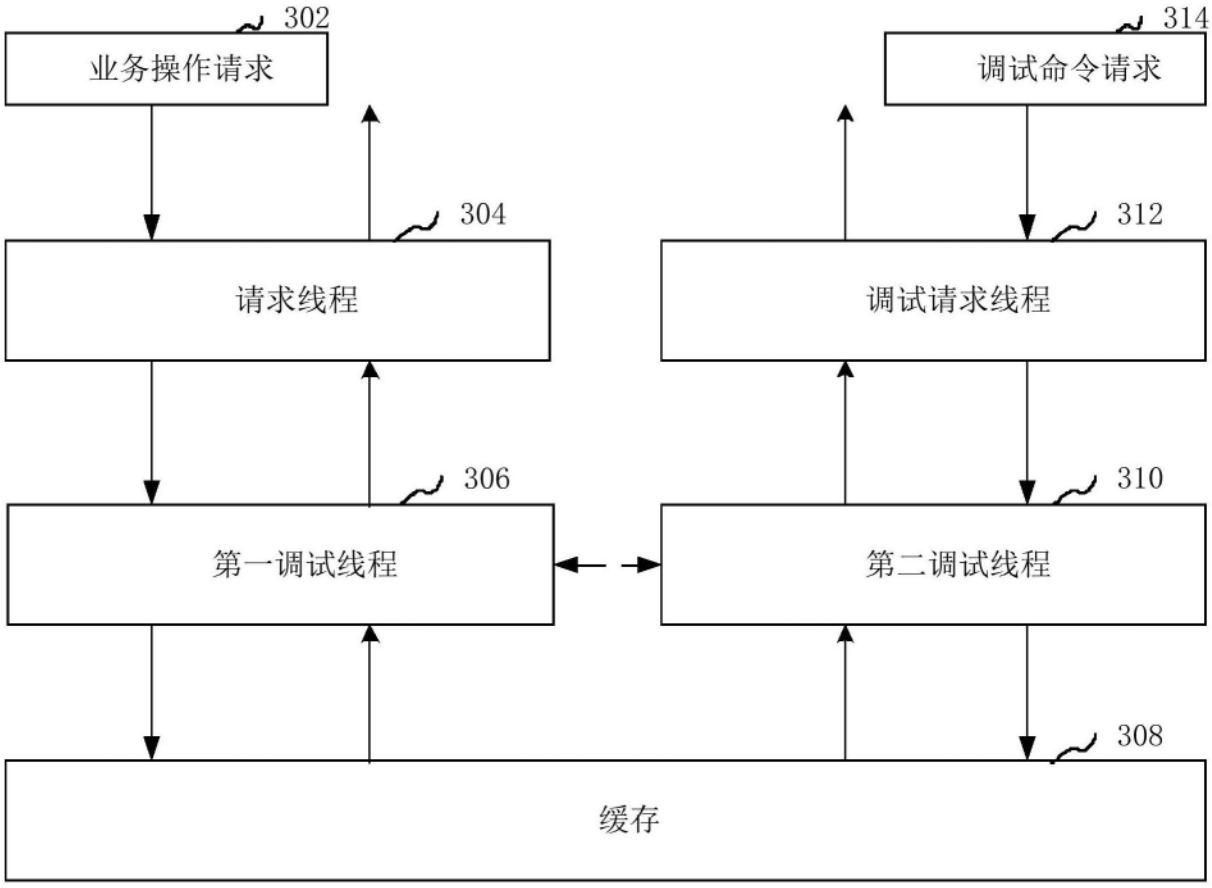

302
业务操作请求
314
调试命令请求
304
请求线程
312
调试请求线程
306
第一调试线程
310
第二调试线程
308
缓存

图3

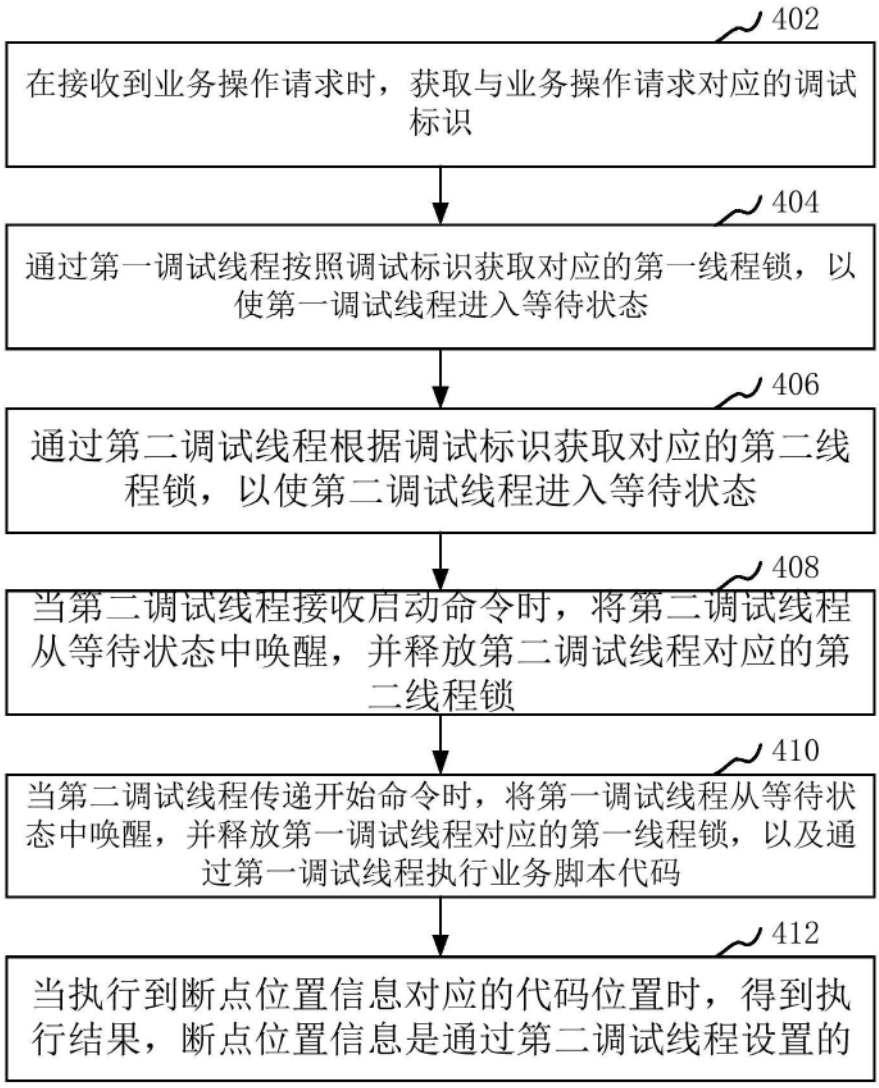

402
在接收到业务操作请求时，获取与业务操作请求对应的调试标识
404
通过第一调试线程按照调试标识获取对应的第一线程锁，以使第一调试线程进入等待状态
406
通过第二调试线程根据调试标识获取对应的第二线程锁，以使第二调试线程进入等待状态
408
当第二调试线程接收启动命令时，将第二调试线程从等待状态中唤醒，并释放第二调试线程对应的第二线程锁
410
当第二调试线程传递开始命令时，将第一调试线程从等待状态中唤醒，并释放第一调试线程对应的第一线程锁，以及通过第一调试线程执行业务脚本代码
412
当执行到断点位置信息对应的代码位置时，得到执行结果，断点位置信息是通过第二调试线程设置的

图4

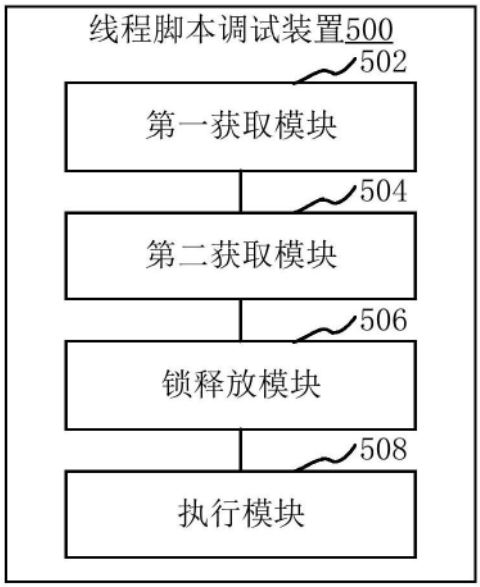

线程脚本调试装置500
502
第一获取模块
504
第二获取模块
506
锁释放模块
508
执行模块

图5

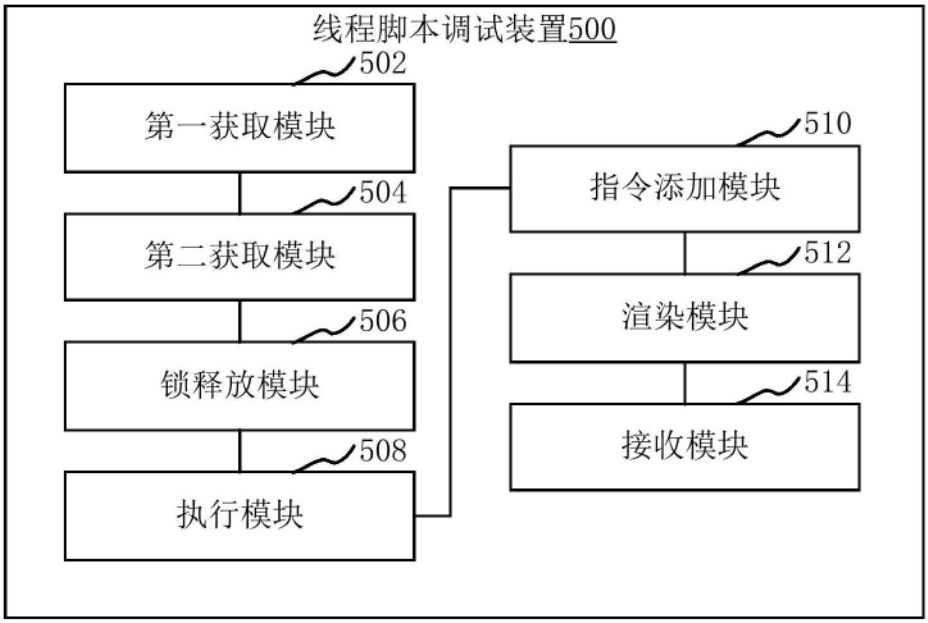

线程脚本调试装置500
502
第一获取模块
504
第二获取模块
506
锁释放模块
508
执行模块
510
指令添加模块
512
渲染模块
514
接收模块

图6

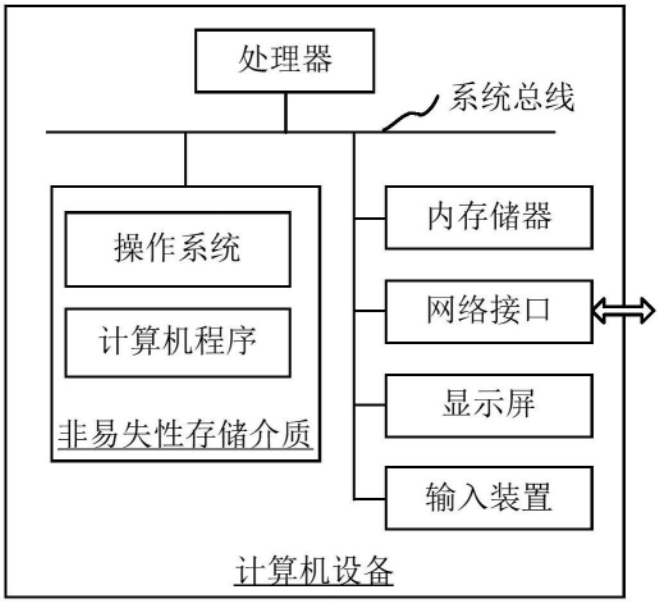



图7