(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 110262818 A
(43)申请公布日 2019.09.20

(21)申请号 201910455888.0

(22)申请日 2019.05.29

(71)申请人 北京达佳互联信息技术有限公司
地址 100085 北京市海淀区上地西路6号1幢1层101D1-7

(72)发明人 刘正阳

(74)专利代理机构 北京成创同维知识产权代理有限公司 11449
代理人 蔡纯 高青

(51)Int.Cl.
G06F 8/65(2018.01)

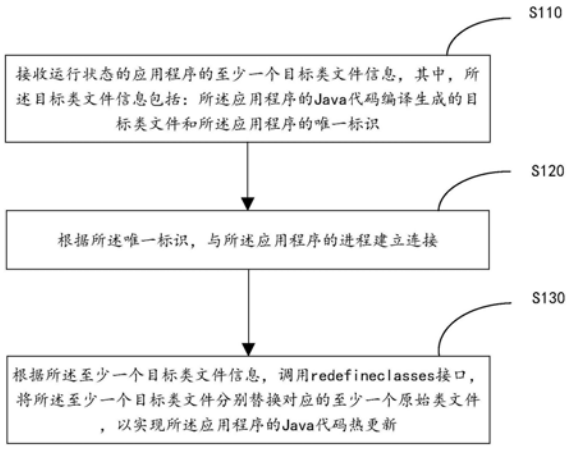
权利要求书3页 说明书13页 附图5页

(54)发明名称

Java代码热更新方法、装置、电子设备及存储介质

(57)摘要

本公开关于一种Java代码热更新方法、装置、电子设备和存储介质。所述Java代码热更新方法包括:接收运行状态的应用程序的至少一个目标类文件信息,其中,所述目标类文件信息包括:所述应用程序的Java代码编译生成的目标类文件和所述应用程序的唯一标识;根据所述唯一标识,与所述应用程序的进程建立连接;以及根据所述至少一个目标类文件信息,调用redefineclasses接口,将所述至少一个目标类文件分别替换对应的至少一个原始类文件,以实现所述应用程序的Java代码热更新。该Java代码热更新方法无需重新启动对应的应用程序即可在jvm执行修改后的Java代码经过编译得到的.class文件,缩短了Java代码更新的耗时,提高了Java代码的更新效率。



CN 110262818 A

1.一种Java代码热更新方法,其特征在于,包括:

接收运行状态的应用程序的至少一个目标类文件信息,其中,所述目标类文件信息包括:所述应用程序的Java代码编译生成的目标类文件和所述应用程序的唯一标识;

根据所述唯一标识,与所述应用程序的进程建立连接;

根据所述至少一个目标类文件信息,调用redefineclasses接口,将所述至少一个目标类文件分别替换对应的至少一个原始类文件,以实现所述应用程序的Java代码热更新。

2.根据权利要求1所述的Java代码热更新方法,其特征在于,所述接收运行状态的应用程序的至少一个目标类文件信息之前,包括:

修改所述应用程序的Java代码,得到所述Java代码的改动部分;

编译所述Java代码的改动部分,得到所述至少一个目标类文件。

3.根据权利要求1所述的Java代码热更新方法,其特征在于,所述唯一标识包括:所述应用程序的进程ID,

则所述根据所述唯一标识,与所述应用程序的进程建立连接,包括:

根据所述应用程序的进程ID,得到所述应用程序对应的服务进程;

与所述应用程序对应的服务进程建立连接。

4.根据权利要求3所述的Java代码热更新方法,其特征在于,所述接收运行状态的应用程序的至少一个目标类文件信息之后,包括:

获取所述至少一个目标类文件的文件名;

存储所述至少一个目标类文件;以及

将所述至少一个目标类文件的存储路径和所述文件名组合成参数。

优选地,所述根据所述至少一个目标类文件信息,调用redefineclasses接口,将所述至少一个目标类文件分别替换对应的至少一个原始类文件,以实现所述应用程序的Java代码热更新,包括:

将所述至少一个目标类文件的所述参数传递到所述应用程序的进程;

将所述至少一个目标类文件的所述参数解析成所述至少一个目标类文件的文件名和存储路径;

根据所述至少一个目标类文件的存储路径读取所述至少一个目标类文件的内容;以及

从虚拟机已加载的所有类文件中,查找与所述至少一个目标类文件同名的至少一个原始类文件。

优选地,所述根据所述至少一个目标类文件信息,调用redefineclasses接口,将所述至少一个目标类文件分别替换对应的至少一个原始类文件,以实现所述应用程序的Java代码热更新,还包括:

调用redefineclasses接口,将所述至少一个目标类文件分别替换对应的至少一个原始类文件,重新加载并执行所述至少一个目标类文件,以实现Java代码热更新。

优选地,所述应用程序的Java代码热更新后,如果所述应用程序的功能不完善,则通过进一步修改所述应用程序的Java代码,实现所述应用程序的Java代码的进一步热更新。

5.一种Java代码热更新装置,其特征在于,包括:

接收单元,被配置为执行接收运行状态的应用程序的至少一个目标类文件信息,其中,所述目标类文件信息包括:所述应用程序的Java代码编译生成的目标类文件和所述应用程

序的唯一标识；

连接单元，被配置为执行根据所述唯一标识，与所述应用程序的进程建立连接；

热更新单元，被配置为执行根据所述至少一个目标类文件信息，调用redefineclasses接口，将所述至少一个目标类文件分别替换对应的至少一个原始类文件，以实现所述应用程序的Java代码热更新。

6.根据权利要求5所述的Java代码热更新装置，其特征在于，所述的Java代码热更新装置，还包括：

修改单元，被配置为执行修改所述应用程序的Java代码，得到所述Java代码的改动部分，编译所述Java代码的改动部分，得到所述至少一个目标类文件。

7.根据权利要求5所述的Java代码热更新装置，其特征在于，所述唯一标识包括：所述应用程序的进程ID，

则所述根据所述唯一标识，与所述应用程序的进程建立连接，包括：

根据所述应用程序的进程ID，得到所述应用程序对应的服务进程；

与所述应用程序对应的服务进程建立连接。

8.根据权利要求7所述的Java代码热更新装置，其特征在于，所述的Java代码热更新装置，还包括：

存储单元，被配置为执行获取所述至少一个目标类文件的文件名，存储所述至少一个目标类文件；

组合单元，被配置为执行将所述至少一个目标类文件的存储路径和所述文件名组合成参数。

优选地，所述根据所述至少一个目标类文件信息，调用redefineclasses接口，将所述至少一个目标类文件分别替换对应的至少一个原始类文件，以实现所述应用程序的Java代码热更新，包括：

将所述至少一个目标类文件的所述参数传递到所述应用程序的进程；

将所述至少一个目标类文件的所述参数解析成所述至少一个目标类文件的文件名和存储路径；

根据所述至少一个目标类文件的存储路径读取所述至少一个目标类文件的内容；以及

从虚拟机已加载的所有类文件中，查找与所述至少一个目标类文件同名的至少一个原始类文件。

优选地，所述根据所述至少一个目标类文件信息，调用redefineclasses接口，将所述至少一个目标类文件分别替换对应的至少一个原始类文件，以实现所述应用程序的Java代码热更新，还包括：

调用redefineclasses接口，将所述至少一个目标类文件分别替换对应的至少一个原始类文件，重新加载并执行所述至少一个目标类文件，以实现Java代码热更新。

优选地，所述应用程序的Java代码热更新后，如果所述应用程序的功能不完善，则通过进一步修改所述应用程序的Java代码，实现所述应用程序的Java代码的进一步热更新。

9.一种电子设备，其特征在于，包括：

处理器；

用于存储所述处理器可执行指令的存储器；

其中，所述处理器被配置为执行所述指令，以实现如权利要求1至4中任一项所述的Java代码热更新方法。

10.一种存储介质，当所述存储介质中的指令由电子设备的处理器执行时，使得所述电子设备能够执行如权利要求1至4中任一项所述的Java代码热更新方法。

Java代码热更新方法、装置、电子设备及存储介质

技术领域

[0001] 本公开涉及软件技术领域，尤其涉及Java代码热更新方法、装置、电子设备及存储介质。

背景技术

[0002] 在Java应用程序的开发过程中，为了不断完善Java应用程序的功能，开发人员经常需要对Java代码进行多次修改。相关技术中，在对Java类中的方法进行修改时，需要重新启动相关的应用程序来实现Java代码的更新，例如，在tomcat容器中，对每一个应用程序建立一个单独的classLoader(类加载器)来加载该应用程序的类文件。当某个应用程序的类文件更新的时候，首先销毁原来的classLoader加载的所有的已加载类，然后新建一个classLoader来加载更新后的类文件。重新启动相关的应用程序操作流程较为复杂且耗时，这降低了Java代码的更新效率。同时，在对Java类中的方法进行修改时，需要重新编译修改后的相关应用程序的全部Java代码，进而在jvm(Java虚拟机)中执行更新后的Java代码。频繁地重新编译修改后的相关应用程序的全部Java代码，增加了内存需求，占用更多CPU资源，这进一步降低了Java代码的更新效率。

发明内容

[0003] 本公开提供一种Java代码热更新方法、装置、电子设备及存储介质，以至少解决相关技术中需要重新启动相关的应用程序来实现Java代码的更新导致Java代码的更新效率低的问题。本公开的技术方案如下：

[0004] 根据本公开实施例的第一方面，提供一种Java代码热更新方法，包括：

[0005] 接收运行状态的应用程序的至少一个目标类文件信息，其中，所述目标类文件信息包括:所述应用程序的Java代码编译生成的目标类文件和所述应用程序的唯一标识；

[0006] 根据所述唯一标识，与所述应用程序的进程建立连接；

[0007] 根据所述至少一个目标类文件信息，调用redefineclasses接口，将所述至少一个目标类文件分别替换对应的至少一个原始类文件，以实现所述应用程序的Java代码热更新。

[0008] 可选地，所述接收运行状态的应用程序的至少一个目标类文件信息之前，包括：

[0009] 修改所述应用程序的Java代码，得到所述Java代码的改动部分；

[0010] 编译所述Java代码的改动部分，得到所述至少一个目标类文件。

[0011] 可选地，所述唯一标识包括:所述应用程序的进程ID，

[0012] 则所述根据所述唯一标识，与所述应用程序的进程建立连接，包括：

[0013] 根据所述应用程序的进程ID，得到所述应用程序对应的服务进程；

[0014] 与所述应用程序对应的服务进程建立连接。

[0015] 可选地，所述接收运行状态的应用程序的至少一个目标类文件信息之后，包括：

[0016] 获取所述至少一个目标类文件的文件名；

[0017] 存储所述至少一个目标类文件;以及

[0018] 将所述至少一个目标类文件的存储路径和所述文件名组合成参数。

[0019] 可选地,所述根据所述至少一个目标类文件信息,调用redefineclasses接口,将所述至少一个目标类文件分别替换对应的至少一个原始类文件,以实现所述应用程序的Java代码热更新,包括:

[0020] 将所述至少一个目标类文件的所述参数传递到所述应用程序的进程;

[0021] 将所述至少一个目标类文件的所述参数解析成所述至少一个目标类文件的文件名和存储路径;

[0022] 根据所述至少一个目标类文件的存储路径读取所述至少一个目标类文件的内容;以及

[0023] 从虚拟机已加载的所有类文件中,查找与所述至少一个目标类文件同名的至少一个原始类文件。

[0024] 可选地,所述根据所述至少一个目标类文件信息,调用redefineclasses接口,将所述至少一个目标类文件分别替换对应的至少一个原始类文件,以实现所述应用程序的Java代码热更新,还包括:

[0025] 调用redefineclasses接口,将所述至少一个目标类文件分别替换对应的至少一个原始类文件,重新加载并执行所述至少一个目标类文件,以实现Java代码热更新。

[0026] 可选地,所述应用程序的Java代码热更新后,如果所述应用程序的功能不完善,则通过进一步修改所述应用程序的Java代码,实现所述应用程序的Java代码的进一步热更新。

[0027] 根据本公开实施例的第二方面,提供一种Java代码热更新装置,包括:

[0028] 接收单元,被配置为执行接收运行状态的应用程序的至少一个目标类文件信息,其中,所述目标类文件信息包括:所述应用程序的Java代码编译生成的目标类文件和所述应用程序的唯一标识;

[0029] 连接单元,被配置为执行根据所述唯一标识,与所述应用程序的进程建立连接;

[0030] 热更新单元,被配置为执行根据所述至少一个目标类文件信息,调用redefineclasses接口,将所述至少一个目标类文件分别替换对应的至少一个原始类文件,以实现所述应用程序的Java代码热更新。

[0031] 可选地,所述的Java代码热更新装置,还包括:

[0032] 修改单元,被配置为执行修改所述应用程序的Java代码,得到所述Java代码的改动部分,编译所述Java代码的改动部分,得到所述至少一个目标类文件。

[0033] 可选地,所述唯一标识包括:所述应用程序的进程ID,

[0034] 则所述根据所述唯一标识,与所述应用程序的进程建立连接,包括:

[0035] 根据所述应用程序的进程ID,得到所述应用程序对应的服务进程;

[0036] 与所述应用程序对应的服务进程建立连接。

[0037] 可选地,所述的Java代码热更新装置,还包括:

[0038] 存储单元,被配置为执行获取所述至少一个目标类文件的文件名,存储所述至少一个目标类文件;

[0039] 组合单元,被配置为执行将所述至少一个目标类文件的存储路径和所述文件名组

合成参数。

[0040] 可选地，所述根据所述至少一个目标类文件信息，调用redefineclasses接口，将所述至少一个目标类文件分别替换对应的至少一个原始类文件，以实现所述应用程序的Java代码热更新，包括：

[0041] 将所述至少一个目标类文件的所述参数传递到所述应用程序的进程；

[0042] 将所述至少一个目标类文件的所述参数解析成所述至少一个目标类文件的文件名和存储路径；

[0043] 根据所述至少一个目标类文件的存储路径读取所述至少一个目标类文件的内容；以及

[0044] 从虚拟机已加载的所有类文件中，查找与所述至少一个目标类文件同名的至少一个原始类文件。

[0045] 可选地，所述根据所述至少一个目标类文件信息，调用redefineclasses接口，将所述至少一个目标类文件分别替换对应的至少一个原始类文件，以实现所述应用程序的Java代码热更新，还包括：

[0046] 调用redefineclasses接口，将所述至少一个目标类文件分别替换对应的至少一个原始类文件，重新加载并执行所述至少一个目标类文件，以实现Java代码热更新。

[0047] 可选地，所述应用程序的Java代码热更新后，如果所述应用程序的功能不完善，则通过进一步修改所述应用程序的Java代码，实现所述应用程序的Java代码的进一步热更新。

[0048] 根据本公开实施例的第三方面，提供一种电子设备，包括：

[0049] 处理器；

[0050] 用于存储所述处理器可执行指令的存储器；

[0051] 其中，所述处理器被配置为执行所述指令，以实现如上所述的Java代码热更新方法。

[0052] 根据本公开实施例的第四方面，提供一种存储介质，当所述存储介质中的指令由电子设备的处理器执行时，使得所述电子设备能够执行如上所述的Java代码热更新方法。

[0053] 根据本公开实施例的第五方面，提供一种计算机程序产品，包括计算机程序产品，所述计算机程序包括程序指令，当所述程序指令被移动终端执行时，使所述移动终端执行上述Java代码热更新方法的步骤。

[0054] 本公开的实施例提供的技术方案至少带来以下有益效果：

[0055] 1)通过代理服务程序agent调用redefineclasses接口，将至少一个目标类文件分别替换对应的至少一个原始类文件。在代理服务程序agent的代理程序中得到一个Instrumentation实例，通过该实例在至少一个原始类文件加载前将该至少一个原始类文件替换为对应的至少一个目标类文件以改变该至少一个原始类文件的.class文件的字节码，在.class文件的字节码改变后重新加载该至少一个目标类文件。重新加载该至少一个目标类文件后在jvm中执行该至少一个目标类文件，以实现Java代码热更新。Java代码修改后，无需重新启动对应的应用程序即可在jvm执行修改后的Java代码经过编译得到的.class文件，看到修改后的代码的最终执行效果，缩短了Java代码更新的耗时，提高了Java代码的更新效率；

[0056] 2)在Java代码更新过程中,无需重启相应的应用程序,因而在代码更新过程中,相应的应用程序实现在线更新、可以持续提供对外的服务,从而提高了Java代码对应的应用程序的服务质量和用户体验;

[0057] 3)该Java代码热更新方法中,目标类文件是开发人员修改应用程序的Java代码后,将Java代码的改动部分重新编译生成的。在对Java代码进行修改时,只需要重新编译相关应用程序的Java代码改动部分,减少了Java代码的编译计算量,进而降低了内存需求,占用较少的CPU资源,这进一步提高了Java代码的更新效率。

[0058] 应当理解的是,以上的一般描述和后文的细节描述仅是示例性和解释性的,并不能限制本公开。

附图说明

[0059] 此处的附图被并入说明书中并构成本说明书的一部分,示出了符合本公开的实施例,并与说明书一起用于解释本公开的原理,并不构成对本公开的不当限定。

[0060] 图1是根据一示例性实施例示出的Java代码热更新方法的流程图;

[0061] 图2是根据一示例性实施例示出的Java代码热更新方法的应用场景图;

[0062] 图3是根据一示例性实施例示出的Java代码热更新方法的流程图;

[0063] 图4是根据一示例性实施例示出的Java代码热更新装置框图;

[0064] 图5是根据一示例性实施例示出的Java代码热更新装置框图;

[0065] 图6是根据一示例性实施例示出的一种执行Java代码热更新方法的装置的框图;

[0066] 图7是根据一示例性实施例示出的一种执行Java代码热更新方法的装置的框图。

具体实施方式

[0067] 为了使本领域普通人员更好地理解本公开的技术方案,下面将结合附图,对本公开实施例中的技术方案进行清楚、完整地描述。

[0068] 需要说明的是,本公开的说明书和权利要求书及上述附图中的术语“第一”、“第二”等是用于区别类似的对象,而不必用于描述特定的顺序或先后次序。应该理解这样使用的数据在适当情况下可以互换,以便这里描述的本公开的实施例能够以除了在这里图示或描述的那些以外的顺序实施。以下示例性实施例中所描述的实施方式并不代表与本公开相一致的所有实施方式。相反,它们仅是与如所附权利要求书中所详述的、本公开的一些方面相一致的装置和方法的例子。

[0069] 在Java应用程序的开发过程中,开发人员每次修改代码后需要将代码重新编译并封装为应用程序,然后重新启动应用程序,才能看到修改后的代码的最终执行效果。某些应用程序重新编译后,重新启动需要较长时间,例如,web应用的web层开发等无法分割成模块开发的应用程序。对于此种应用程序,每次即使修改很少量的代码都需要重新编译封装整个应用程序,并且,重新启动web应用服务器。

[0070] 在企业级的开发中,通常修改普通的应用程序,从修改代码到关闭并重新启动web服务器,这一过程需要耗时约5分钟。对于较大的应用程序,这一过程可能需要耗时约10分钟或者更多时间。然而,在程序编写过程中,开发人员需要定期查看当前代码的运行结果并做出相应修改,以保证代码按照开发人员的意愿呈现相应效果。因此,需要不断地关闭并重

新启动web服务器,这使得代码编译和重启应用程序的时间大大提升。另外,对于大型服务性应用程序来说,在代码更新过程中,往往需要重新编译代码和重新启动应用程序,在重启服务的过程中,需要中断对外的服务,从而影响相应应用程序的服务质量。

[0071] 为此,本发明提供了一种Java代码热更新方法,图1是根据一示例性实施例示出的Java代码热更新方法的流程图。

[0072] 在步骤S110中,接收运行状态的应用程序的至少一个目标类文件信息,其中,所述目标类文件信息包括:所述应用程序的Java代码编译生成的目标类文件和所述应用程序的唯一标识。

[0073] 在该步骤中,接收处于运行状态的应用程序的至少一个目标类文件信息,其中,该目标类文件信息包括:应用程序的Java代码编译生成的目标类文件和应用程序的唯一标识。应用程序的唯一标识用来唯一标识相应的应用程序的服务进程。这里的目标类文件可以是开发人员修改应用程序的Java代码后,将全部Java代码重新编译生成的。

[0074] 优选地,目标类文件还可以是开发人员修改应用程序的Java代码后,将Java代码的改动部分重新编译生成的。

[0075] 具体地,首先,开发人员查找需要修改功能的Java代码。然后,修改该Java代码。由于目前所有的Java应用程序均基于jvm在服务器上运行,所以开发人员还需要将Java代码的改动部分编译为jvm所能识别的格式,即.class文件。

[0076] 与普通程序不同,.class文件并不是本地服务器的可执行程序。当需要运行.class文件时,首先应运行jvm,然后再把.class文件加载到jvm中运行,jvm中负责加载.class文件的工具叫做classLoader,而jvm在加载过程中使用的接口称为jvmti接口。

[0077] 另外,接收运行状态的应用程序的至少一个目标类文件信息。可以理解的是,接收的至少一个目标类文件信息可以来自一个应用程序,对于一个应用程序A来说,如果应用程序A的Java代码有多处修改,则可以将应用程序A的这多处Java代码改动部分编译为一个目标类文件,也可以将应用程序A的这多处Java代码改动部分编译为多个目标类文件。

[0078] 可选地,接收运行状态的多个应用程序的多个目标类文件信息,对于应用程序B和C来说,如果应用程序B和C的Java代码分别有多处修改,则可以分别将应用程序B的这多处Java代码改动部分编译为一个目标类文件,也可以分别将应用程序B的这多处Java代码改动部分编译为多个目标类文件;可以将应用程序C的这多处Java代码改动部分编译为一个目标类文件,也可以将应用程序C的这多处Java代码改动部分编译为多个目标类文件。

[0079] 可以理解的是,应用程序的个数以及每个应用程序的Java代码的改动部分编译成的目标类文件的个数可以为任意值,不应对本公开实施例构成任何限制。

[0080] 在步骤S120中,根据所述唯一标识,与所述应用程序的进程建立连接。

[0081] 在该步骤中,根据应用程序的唯一标识,与相应的应用程序的进程建立连接。

[0082] 每个应用程序的唯一标识包括:应用程序的进程ID,则根据应用程序的唯一标识,与相应的应用程序的进程建立连接,包括:根据应用程序的进程ID,得到该应用程序对应的服务进程。与该应用程序对应的服务进程建立连接。

[0083] Java的com.sun.tools.attach包中的VirtualMachine类允许通过给attach方法传入一个jvm的pid(进程id),远程连接到jvm上,例如,将目标类文件对应的应用程序的进程ID传入attach方法,远程连接到对应的应用程序所在服务器的jvm上。可以通过

loadAgent方法向jvm注册一个代理服务程序agent，在该agent的代理程序中会得到一个Instrumentation实例，该实例可以在.class文件加载前改变.class文件的字节码，可以在.class文件的字节码改变后重新加载。在调用Instrumentation实例的方法时，attach、loadAgent等方法会使用ClassFileTransformer接口中提供的方法进行处理。

[0084] 在步骤S130中，根据所述至少一个目标类文件信息，调用redefineclasses接口，将所述至少一个目标类文件分别替换对应的至少一个原始类文件，以实现所述应用程序的Java代码热更新。

[0085] 在该步骤中，根据至少一个目标类文件信息，调用redefineclasses接口，使用至少一个目标类文件分别替换对应的至少一个原始类文件，以实现该至少一个目标类文件对应的应用程序的Java代码热更新。

[0086] 具体地，根据至少一个目标类文件在测试机本地的存储信息，调用redefineclasses接口，将至少一个目标类文件分别替换对应的至少一个原始类文件。在代理服务程序agent的代理程序中得到一个Instrumentation实例，通过该实例在该至少一个原始类文件加载前改变.class文件的字节码，在.class文件的字节码改变后重新加载至少一个目标类文件即将该至少一个原始类文件替换为对应的至少一个目标类文件。重新加载该至少一个目标类文件后在jvm中执行该至少一个目标类文件，以实现Java代码热更新。

[0087] 在本公开的实施例中，通过代理服务程序agent调用redefineclasses接口，将至少一个目标类文件分别替换对应的至少一个原始类文件。在代理服务程序agent的代理程序中得到一个Instrumentation实例，通过该实例在至少一个原始类文件加载前将该至少一个原始类文件替换为对应的至少一个目标类文件以改变该至少一个原始类文件的.class文件的字节码，在.class文件的字节码改变后重新加载该至少一个目标类文件。重新加载至少一个目标类文件后在jvm中执行该至少一个目标类文件，以实现Java代码热更新。Java代码修改后，无需重新启动对应的应用程序即可在jvm执行修改后的Java代码经过编译得到的.class文件，看到修改后的代码的最终执行效果，缩短了Java代码更新的耗时，提高了Java代码的更新效率。同时，在Java代码更新过程中，无需重启相应的应用程序，因而在代码更新过程中，相应的应用程序实现在线更新、可以持续提供对外的服务，从而提高了Java代码对应的应用程序的服务质量和用户体验。

[0088] 优选地，目标类文件是开发人员修改应用程序的Java代码后，将Java代码的改动部分重新编译生成的。在对Java代码进行修改时，只需要重新编译相关应用程序的Java代码改动部分，减少了Java代码的编译计算量，进而降低了内存需求，占用较少的CPU资源，这进一步提高了Java代码的更新效率。

[0089] 代码测试是重现应用程序故障、定位故障根源，并最终解决应用程序的故障问题的过程。当测试一个项目代码时，单步执行到错误点时，需要修改代码，修改的代码可能是各种各样的类型。如果想要测试修改后的代码以查看代码修改后应用程序的执行效果，相关技术中的方案是重启应用程序，然后再单步执行到上一个测试现场，这会消耗很多时间、增加很多重复的操作步骤，使得应用程序的开发效率降低。所以在代码测试的过程中实现代码的热更新具有重大意义。

[0090] 图2是根据一示例性实施例示出的Java代码热更新方法的应用场景图。开发人员在本地开发机210修改应用程序的Java代码，将Java代码的改动部分重新编译生成的jvm可

以识别的.class文件。将编译生成的.class文件同步到测试机以测试应用程序的Java代码经过修改后是否按照开发人员的意愿呈现出相应效果。测试机包括:测试机代理220和测试机服务230。在代码测试过程中,应用程序在测试机持续运行。可以提前在测试机通过loadAgent方法向测试机的jvm注册一个代理服务程序agent。开发人员通过代理服务程序agent实现在测试机上运行的应用程序的远程热更新。

[0091] 图3是根据一示例性实施例示出的Java代码热更新方法的流程图。

[0092] 下面结合图2所示的代码测试应用场景,详细说明图3所示的Java代码热更新方法。

[0093] 具体步骤包括:

[0094] 在步骤S301中,在测试机运行待测试的所述多个应用程序。

[0095] 在该步骤中,在测试机服务230运行待测试的多个应用程序。例如,在测试机服务230上运行应用程序D、应用程序E和应用程序F。

[0096] 在步骤S201中,获取所述多个应用程序在测试机上运行的进程ID。

[0097] 在该步骤中,获取多个应用程序在测试机上运行的进程ID。例如,获取在测试机服务230上运行的应用程序D的进程ID1、应用程序E的进程ID2和应用程序F的进程ID3。

[0098] 在步骤S101中,分别修改运行状态的所述多个应用程序的Java代码,得到所述多个应用程序的Java代码的改动部分。

[0099] 在步骤S102中,分别编译所述多个应用程序的Java代码的改动部分,得到所述多个目标类文件。

[0100] 在步骤S103中,上传所述多个目标类文件。

[0101] 在步骤S101-步骤S103中,开发人员在本地开发机210分别修改运行在测试机上的多个应用程序的Java代码,例如,修改Java代码的方法body,将多个应用程序的Java代码的改动部分分别重新编译生成jvm可以识别的.class文件格式的多个目标类文件。

[0102] 将多个目标类文件信息即步骤S101-步骤S103中编译生成的多个目标类文件和步骤S201中获得的对应的多个应用程序的进程ID同步到测试机以分别测试多个应用程序的Java代码经过修改后是否按照开发人员的意愿呈现出相应效果。

[0103] 例如,开发人员在本地开发机210分别修改应用程序D、应用程序E和应用程序F的Java代码。并且分别编译应用程序D、应用程序E和应用程序F的Java代码的改动部分,得到应用程序D的多个目标类文件、应用程序E的多个目标类文件和应用程序F的多个目标类文件。将应用程序D的多个目标类文件和进程ID1、应用程序E的多个目标类文件和进程ID2以及应用程序F的多个目标类文件和进程ID3分别通过web页面或者快捷插件上传到测试机的测试机代理220。

[0104] 在步骤S202中,获取所述多个目标类文件的类名。

[0105] 在步骤S203中,存储所述多个目标类文件。

[0106] 在步骤S204中,将所述多个目标类文件的类名和存储路径组合成参数。

[0107] 在步骤S202-步骤S204中,当接收到多个目标类文件信息后,测试机代理220获取多个目标类文件的类名,将这多个目标类文件的类名和存储路径组合成参数。

[0108] 例如,当分别接收到应用程序D的三个目标类文件后,使用ASM框架(java字节码操纵框架)下的classReader类来解析这三个目标类文件的.class文件,得到这三个目标类文

件的内容。通过className类获取这三个目标类文件的类名。将获取的这三个目标类文件的类名转换为jvm可以识别的类名后，使用转换后的类名将这三个类文件存储。例如，将应用程序D的目标类文件1、目标类文件2和目标类文件3的类名“a1/b1/c1”、“a2/b2/c2”和“a3/b3/c3”转换为“a1.b1.c1”、“a2.b2.c2”和“a3.b3.c3”。分别使用类名“a1.b1.c1”、“a2.b2.c2”和“a3.b3.c3”将应用程序D的目标类文件1、目标类文件2和目标类文件3分别存储在存储路径1、存储路径2和存储路径3。

[0109] 将类名“a1.b1.c1”和存储路径1组合成目标类文件1的参数agentArgs1，将类名“a2.b2.c2”和存储路径2组合成目标类文件2的参数agentArgs2，将类名“a3.b3.c3”和存储路径3组合成目标类文件3的参数agentArgs3。

[0110] 在步骤S205中，根据所述多个应用程序的进程ID，分别与所述多个应用程序的进程建立连接。

[0111] 在该步骤中，测试机代理220根据多个应用程序的进程ID，分别得到多个应用程序对应的服务进程。例如，根据应用程序D、应用程序E和应用程序F的进程ID，通过findProcess方法查找测试机上运行的jvm的所有的进程列表，查找到应用程序D、应用程序E和应用程序F的具体进程。

[0112] 分别与多个应用程序对应的服务进程建立连接。例如，对于应用程序D、应用程序E和应用程序F，代理服务程序220通过给VirtualMachine.attach方法传入进程ID1，与运行在jvm的应用程序D的具体进程建立连接；代理服务程序220通过给VirtualMachine.attach方法传入进程ID2，与运行在jvm的应用程序E的具体进程建立连接；代理服务程序220通过给VirtualMachine.attach方法传入进程ID3，与运行在jvm的应用程序F的具体进程建立连接。

[0113] 在步骤S206中，将所述多个目标类文件的所述参数传递到所述多个应用程序的进程。

[0114] 在步骤S302中，将所述多个目标类文件的参数解析成所述多个目标类文件的类名和存储路径。

[0115] 在步骤S206和步骤S302中，将多个目标类文件的参数传递到多个应用程序的进程。将多个目标类文件的参数解析成多个目标类文件的类名和存储路径。

[0116] 例如，将目标类文件1的参数agentArgs1、目标类文件2的参数agentArgs2和目标类文件3的参数agentArgs3分别传递到应用程序D、应用程序E和应用程序F的服务进程。

[0117] Java类一般打包在jar格式的压缩包中。本公开实施例中，提前在测试机通过loadAgent方法向测试机的jvm注册代理服务程序agent，可以将代理服务程序agent打包在jar格式的压缩包中。代理服务程序agent的jar格式的压缩包的路径为agentPath。通过为attach.loadAgent方法分别上传目标类文件1的参数agentArgs1和agentPath、目标类文件2的参数agentArgs2和agentPath以及目标类文件3的参数agentArgs3和agentPath，分别得到目标类文件1、目标类文件2和目标类文件3的类名。通过Paths.get获取目标类文件1、目标类文件2和目标类文件3的存储路径。

[0118] 在步骤S303中，根据所述多个目标类文件的存储路径读取所述多个目标类文件的内容。

[0119] 在步骤S304中，从虚拟机已加载的所有类文件中，查找与所述多个目标类文件同

名的多个原始类文件。

[0120] 在步骤S303和步骤S304中,根据多个目标类文件的存储路径读取该多个目标类文件的内容。从虚拟机已加载的所有类文件中,查找与多个目标类文件同名的类对象。

[0121] 例如,根据多个目标类文件的存储路径读取该多个目标类文件的内容。

[0122] 通过instrumentation.getAllLoadedClasses遍历测试机上运行的jvm中已加载的.class文件(已加载的.class文件可以存储在缓存中),找到与该多个目标类文件的文件名相同的类对象即待替换的原始类文件。

[0123] 在步骤S305中,分别调用redefineclasses接口,将多个所述目标类文件分别替换对应的多个原始类文件,以分别实现所述多个应用程序的Java代码热更新。

[0124] 在该步骤中,测试机服务230分别调用redefineclasses接口,分别将每一个应用程序的至少一个目标类文件分别替换对应的至少一个原始类文件。

[0125] 例如,分别调用redefineclasses接口,在代理服务程序agent的代理程序中得到一个Instrumentation实例,通过该实例在每一个应用程序的至少一个原始类文件加载前将该至少一个原始类文件替换为对应的至少一个目标类文件以改变该至少一个原始类文件的.class文件的字节码,在.class文件的字节码改变后重新加载该至少一个目标类文件。

[0126] 例如,通过Java agent机制调用java.lang.instrument.Instrumentation类的redefineClasses方法完成每个应用程序的至少一个原始类文件的类重定义,实现将每个应用程序的至少一个原始类文件替换为对应的至少一个目标类文件。

[0127] 分别通过classLoader将.class文件格式的每个应用程序的至少一个目标类文件加载到测试机上运行的jvm中运行,以实现对应的应用程序的Java代码热更新。

[0128] 多个目标类文件在测试机上运行的jvm中运行的过程中,测试机测试对应的多个应用程序的功能否按照开发人员的意愿呈现出相应效果。经过代码测试,如果对应的多个应用程序的功能没有按照开发人员的意愿呈现出相应效果,则进一步执行步骤S101以进一步热更新对应的多个应用程序的Java代码。

[0129] 根据本公开的实施例,在对多个应用程序的功能进行测试时,开发人员在本地开发机210分别修改运行在测试机上的多个应用程序的Java代码,将多个应用程序的Java代码的改动部分分别重新编译生成jvm可以识别的.class文件格式的多个目标类文件。将多个目标类文件信息即编译生成的多个目标类文件和对应的多个应用程序的进程ID同步到测试机以分别测试多个应用程序的Java代码经过修改后是否按照开发人员的意愿呈现出相应效果。测试过程中只需要重新编译相关应用程序的Java代码的改动部分,减少了Java代码的编译计算量,进而降低了内存需求,占用较少的CPU资源,这进一步提高了Java代码的更新效率,进而提高了应用程序的开发效率。

[0130] 另外,测试机服务230分别调用redefineclasses接口,将每一个应用程序的至少一个目标类文件分别替换对应的至少一个原始类文件。分别通过classLoader将.class文件格式的每一个应用程序的至少一个目标类文件加载到测试机上运行的jvm中运行,以实现对应的应用程序的Java代码热更新。每一个应用程序的至少一个目标类文件在测试机上运行的jvm中运行的过程中测试对应的应用程序的功能否按照开发人员的意愿呈现出相应效果。经过代码测试,如果对应的多个应用程序的功能没有按照开发人员的意愿呈现出相

应效果，则进一步执行步骤S310以进一步热更新对应的多个应用程序的Java代码。测试过程中利用java.lang.instrument.Instrumentation类的redefine功能，无需重启相应的应用程序即可实现Java代码的热更新，缩短了Java代码更新的耗时，提高了Java代码的更新效率，进而提高了应用程序的开发效率。

[0131] 图4是根据一示例性实施例示出的Java代码热更新装置框图。包括：接收单元410、连接单元420和热更新单元430。

[0132] 接收单元410，被配置为执行接收运行状态的应用程序的至少一个目标类文件信息，其中，所述目标类文件信息包括：所述应用程序的Java代码编译生成的目标类文件和所述应用程序的唯一标识。

[0133] 连接单元420，被配置为执行根据所述唯一标识，与所述应用程序的进程建立连接。

[0134] 热更新单元430，被配置为执行根据所述至少一个目标类文件信息，调用redefineclasses接口，将所述至少一个目标类文件分别替换对应的至少一个原始类文件，以实现所述应用程序的Java代码热更新。

[0135] 在本公开的实施例中，该Java代码热更新装置接收处于运行状态的应用程序的至少一个目标类文件信息，其中，该目标类文件信息包括：应用程序的Java代码编译生成的目标类文件和应用程序的唯一标识。这里的目标类文件可以是开发人员修改应用程序的Java代码后，将全部Java代码重新编译生成的。目标类文件还可以是开发人员修改应用程序的Java代码后，将Java代码的改动部分重新编译生成的。每个应用程序的唯一标识包括：应用程序的进程ID，根据应用程序的进程ID，得到应用程序对应的服务进程。与应用程序对应的服务进程建立连接。根据至少一个目标类文件信息，调用redefineclasses接口，使用至少一个目标类文件分别替换对应的至少一个原始类文件，以实现至少一个目标类文件对应的应用程序的Java代码热更新。Java代码修改后，热更新单元430无需重新启动对应的应用程序即可在jvm执行修改后的Java代码经过编译得到的.class文件，看到修改后的代码的最终执行效果，缩短了Java代码更新的耗时，提高了Java代码的更新效率。

[0136] 图5是根据一示例性实施例示出的Java代码热更新装置框图。包括：修改单元510、接收单元520、连接单元530、存储单元540、组合单元550、和热更新单元560。

[0137] 修改单元510，被配置为执行修改所述应用程序的Java代码，得到所述Java代码的改动部分，编译所述Java代码的改动部分，得到所述至少一个目标类文件。

[0138] 接收单元520，被配置为执行接收运行状态的应用程序的至少一个目标类文件信息，其中，所述目标类文件信息包括：所述应用程序的Java代码编译生成的目标类文件和所述应用程序的唯一标识。

[0139] 连接单元530，被配置为执行根据所述唯一标识，与所述应用程序的进程建立连接。

[0140] 可选地，所述唯一标识包括：所述应用程序的进程ID，则所述根据所述唯一标识，与所述应用程序建立连接，包括：

[0141] 根据所述应用程序的进程ID，得到所述应用程序对应的服务进程；与所述应用程序对应的服务进程建立连接。

[0142] 存储单元540，被配置为执行获取所述至少一个目标类文件的文件名，存储所述至

少一个目标类文件；

[0143] 组合单元550，被配置为执行将所述至少一个目标类文件的存储路径和所述文件名组合成参数。

[0144] 热更新单元560，被配置为执行根据所述至少一个目标类文件信息，调用redefineclasses接口，将所述至少一个目标类文件分别替换对应的至少一个原始类文件，以实现所述应用程序的Java代码热更新。

[0145] 可选地，将所述至少一个目标类文件的所述参数传递到所述应用程序的进程；将所述至少一个目标类文件的所述参数解析成所述至少一个目标类文件的文件名和存储路径；根据所述至少一个目标类文件的存储路径读取所述至少一个目标类文件的内容；以及从虚拟机加载的所有类文件中，查找与所述至少一个目标类文件同名的至少一个原始类文件。

[0146] 调用redefineclasses接口，将所述至少一个目标类文件分别替换对应的至少一个原始类文件，重新加载并执行所述至少一个目标类文件，以实现Java代码热更新。

[0147] 在本公开的实施例中，根据所述至少一个目标类文件信息，调用redefineclasses接口，使用至少一个目标类文件分别替换对应的至少一个原始类文件，以实现至少一个目标类文件对应的应用程序的Java代码热更新。对Java代码热更新时对应的应用程序运行的容器没有限制，对相应的应用程序提供的对外服务的种类也没有限制。无论使用tomcat容器还是使用类似jetty等容器都可以实现本公开实施例的Java代码热更新。

[0148] 在本公开的一个可选的实施例中，所述应用程序的Java代码热更新后，如果所述应用程序的功能不完善，则通过进一步修改所述应用程序的Java代码，实现所述应用程序的Java代码的进一步热更新。

[0149] 图6是根据一示例性实施例示出的一种用于上述Java代码热更新方法的Java代码热更新装置600的框图。例如，Java代码热更新装置600可以是移动电话，计算机，数字广播终端，消息收发设备，游戏控制台，平板设备，医疗设备，健身设备，个人数字助理等。

[0150] 参照图6，装置600可以包括以下一个或多个组件：处理组件610，存储器620，电源组件630，多媒体组件640，音频组件650，输入/输出(I/O)的接口660，传感器组件670，以及通信组件680。

[0151] 处理组件610通常控制装置600的整体操作，诸如与显示，电话呼叫，数据通信，相机操作和记录操作相关联的操作。处理组件610可以包括一个或多个处理器690来执行指令，以完成上述的方法的全部或部分步骤。此外，处理组件610可以包括一个或多个模块，便于处理组件610和其他组件之间的交互。例如，处理组件610可以包括多媒体模块，以方便多媒体组件540和处理组件610之间的交互。

[0152] 存储器620被配置为存储各种类型的数据以支持在设备600的操作。这些数据的示例包括用于在装置600上操作的任何应用程序或方法的指令，联系人数据，电话簿数据，消息，图片，视频等。存储器620可以由任何类型的易失性或非易失性存储设备或者它们的组合实现，如静态随机存取存储器(SRAM)，电可擦除可编程只读存储器(EEPROM)，可擦除可编程只读存储器(EPROM)，可编程只读存储器(PROM)，只读存储器(ROM)，磁存储器，快闪存储器，磁盘或光盘。

[0153] 电源组件630为装置600的各种组件提供电力。电源组件630可以包括电源管理系

统，一个或多个电源，及其他与为装置600生成、管理和分配电力相关联的组件。

[0154] 多媒体组件640包括在所述装置600和用户之间的提供一个输出接口的屏幕。在一些实施例中，屏幕可以包括液晶显示器(LCD)和触摸面板(TP)。如果屏幕包括触摸面板，屏幕可以被实现为触摸屏，以接收来自用户的输入信号。触摸面板包括一个或多个触摸传感器以感测触摸、滑动和触摸面板上的手势。所述触摸传感器可以不仅感测触摸或滑动动作的边界，而且还检测与所述触摸或滑动操作相关的持续时间和压力。在一些实施例中，多媒体组件640包括一个前置摄像头和/或后置摄像头。当设备600处于操作模式，如拍摄模式或视频模式时，前置摄像头和/或后置摄像头可以接收外部的多媒体数据。每个前置摄像头和后置摄像头可以是一个固定的光学透镜系统或具有焦距和光学变焦能力。

[0155] 音频组件650被配置为输出和/或输入音频信号。例如，音频组件650包括一个麦克风(MIC)，当装置600处于操作模式，如呼叫模式、记录模式和语音识别模式时，麦克风被配置为接收外部音频信号。所接收的音频信号可以被进一步存储在存储器620或经由通信组件680发送。在一些实施例中，音频组件650还包括一个扬声器，用于输出音频信号。

[0156] I/O接口660为处理组件610和外围接口模块之间提供接口，上述外围接口模块可以是键盘，点击轮，按钮等。这些按钮可包括但不限于：主页按钮、音量按钮、启用按钮和锁定按钮。

[0157] 传感器组件670包括一个或多个传感器，用于为装置600提供各个方面的状态评估。例如，传感器组件670可以检测到设备600的打开/关闭状态，组件的相对定位，例如所述组件为装置600的显示器和小键盘，传感器组件670还可以检测装置600或装置600一个组件的位置改变，用户与装置600接触的存在或不存在，装置600方位或加速/减速和装置600的温度变化。传感器组件670可以包括接近传感器，被配置用来在没有任何的物理接触时检测附近物体的存在。传感器组件670还可以包括光传感器，如CMOS或CCD图像传感器，用于在成像应用中使用。在一些实施例中，该传感器组件670还可以包括加速度传感器，陀螺仪传感器，磁传感器，压力传感器或温度传感器。

[0158] 通信组件680被配置为便于装置600和其他设备之间有线或无线方式的通信。装置600可以接入基于通信标准的无线网络，如WiFi，运营商网络(如2G、3G、4G或5G)，或它们的组合。在一个示例性实施例中，通信组件680经由广播信道接收来自外部广播管理系统的广播信号或广播相关信息。在一个示例性实施例中，所述通信组件680还包括近场通信(NFC)模块，以促进短程通信。例如，在NFC模块可基于射频识别(RFID)技术，红外数据协会(IrDA)技术，超宽带(UWB)技术，蓝牙(BT)技术和其他技术来实现。

[0159] 在示例性实施例中，装置600可以被一个或多个应用专用集成电路(ASIC)、数字信号处理器(DSP)、数字信号处理设备(DSPD)、可编程逻辑器件(PLD)、现场可编程门阵列(FPGA)、控制器、微控制器、微处理器或其他电子元件实现，用于执行上述方法。

[0160] 在示例性实施例中，还提供了一种包括指令的存储介质，例如包括指令的存储器620，上述指令可由装置600的处理器690执行以完成上述方法。可选地，存储介质可以是非临时性计算机可读存储介质，例如，所述非临时性计算机可读存储介质可以是ROM、随机存取存储器(RAM)、CD-ROM、磁带、软盘和光数据存储设备等。

[0161] 在示例性实施例中，还提供了一种计算机程序产品，包括计算机程序产品，所述计算机程序包括程序指令，当所述程序指令被移动终端执行时，使所述移动终端执行上述

Java代码热更新方法的步骤:接收运行状态的应用程序的至少一个目标类文件信息,其中,所述目标类文件信息包括:所述应用程序的Java代码编译生成的目标类文件和所述应用程序的唯一标识;根据所述唯一标识,与所述应用程序的进程建立连接;根据所述至少一个目标类文件信息,调用redefineclasses接口,将所述至少一个目标类文件分别替换对应的至少一个原始类文件,以实现所述应用程序的Java代码热更新。

[0162] 图7是根据一示例性实施例示出的一种用于上述Java代码热更新方法的Java代码热更新装置700的框图。例如,装置700可以被提供为一服务器。参照图7,装置700包括处理组件710,其进一步包括一个或多个处理器,以及由存储器720所代表的存储器资源,用于存储可由处理组件710的执行的指令,例如应用程序。存储器720中存储的应用程序可以包括一个或一个以上的每一个对应于一组指令的模块。此外,处理组件710被配置为执行指令,以执行上述Java代码热更新方法。

[0163] 装置700还可以包括一个电源组件730被配置为执行装置700的电源管理,一个有线或无线网络接口740被配置为将装置700连接到网络,和一个输入输出(I/O)接口750。装置700可以操作基于存储在存储器720的操作系统,例如Windows ServerTM,Mac OS XTM,UnixTM,LinuxTM,FreeBSDTM或类似。

[0164] 本领域技术人员在考虑说明书及实践这里公开的发明后,将容易想到本公开的其它实施方案。本公开旨在涵盖本公开的任何变型、用途或者适应性变化,这些变型、用途或者适应性变化遵循本公开的一般性原理并包括本公开未公开的本技术领域中的公知常识或惯用技术手段。说明书和实施例仅被视为示例性的,本公开的真正范围和精神由下面的权利要求指出。

[0165] 应当理解的是,本公开并不局限于上面已经描述并在附图中示出的精确结构,并且可以在不脱离其范围进行各种修改和改变。本公开的范围仅由所附的权利要求来限制。

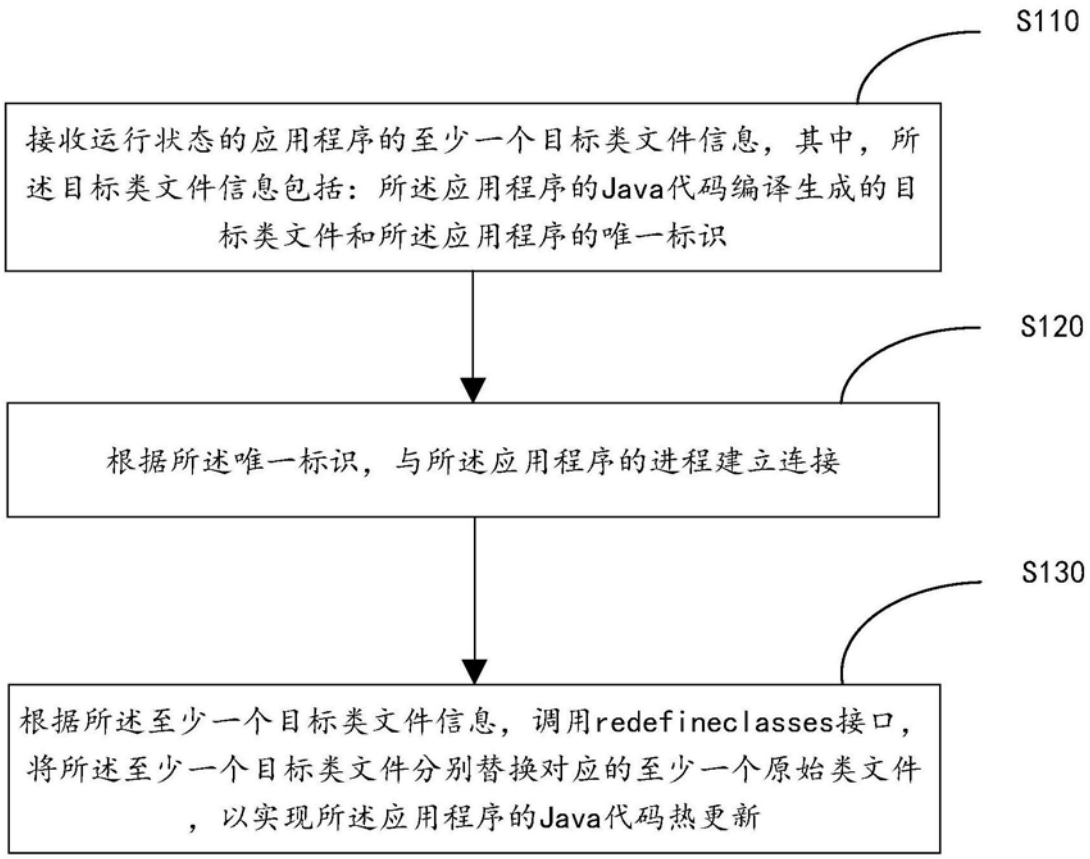

S110
接收运行状态的应用程序的至少一个目标类文件信息，其中，所述目标类文件信息包括：所述应用程序的Java代码编译生成的目标类文件和所述应用程序的唯一标识
S120
根据所述唯一标识，与所述应用程序的进程建立连接
S130
根据所述至少一个目标类文件信息，调用redefineclasses接口，将所述至少一个目标类文件分别替换对应的至少一个原始类文件，以实现所述应用程序的Java代码热更新

图1

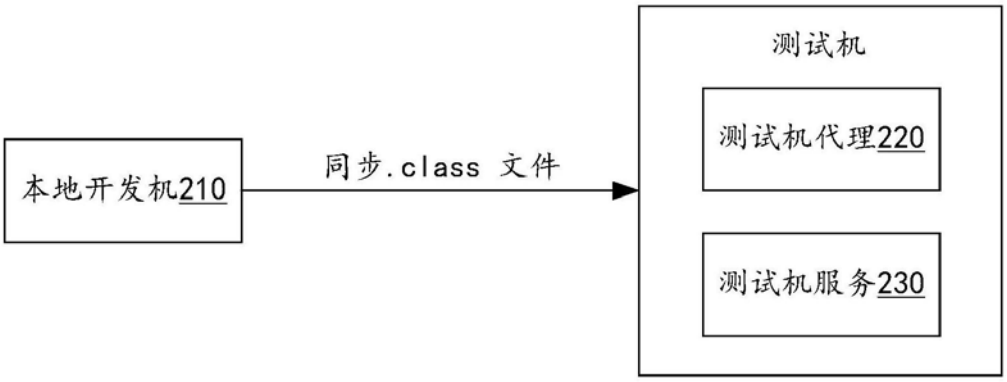

测试机
本地开发机210
同步.class 文件
测试机代理220
测试机服务230

图2

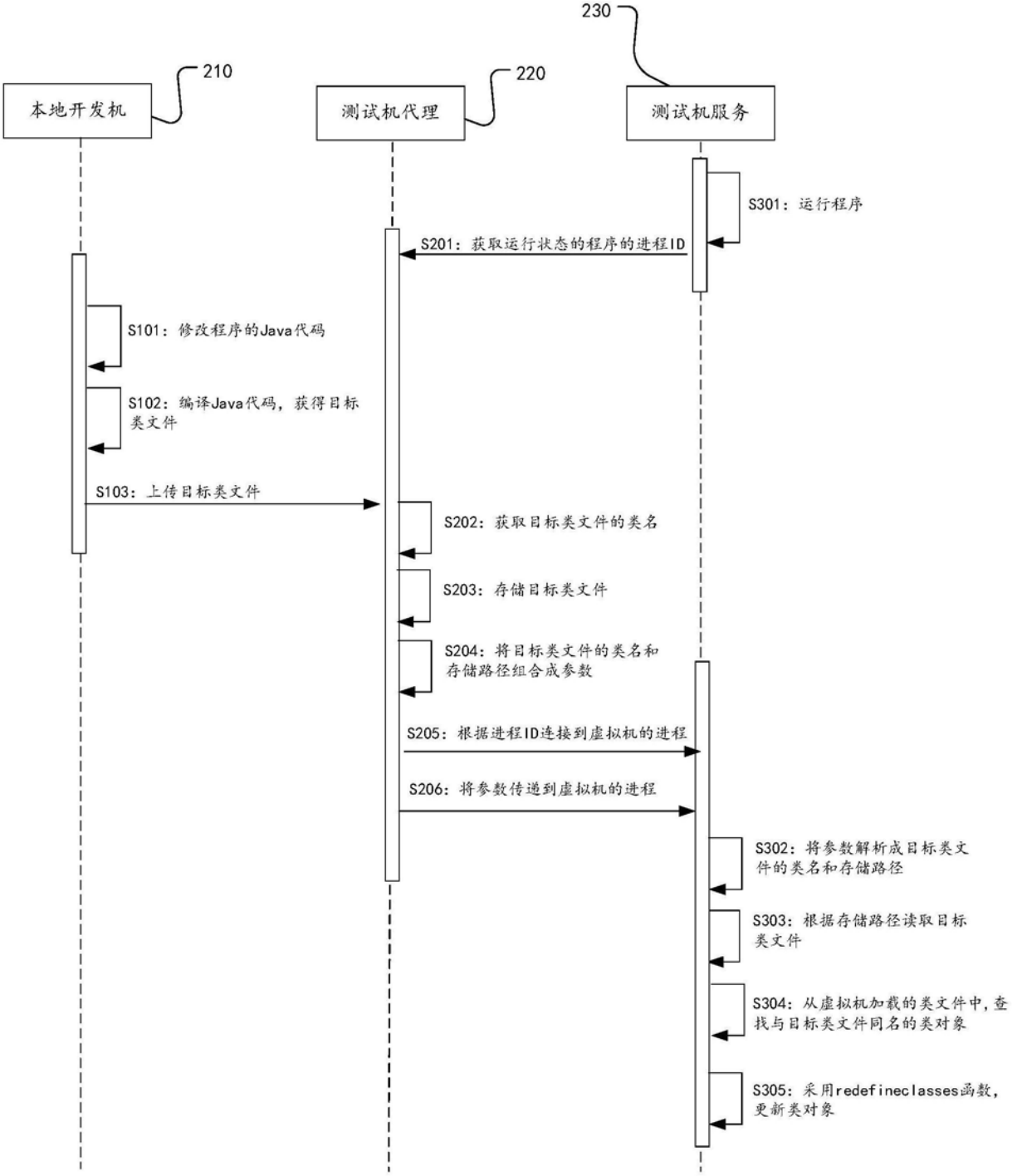

230
210
220
本地开发机
测试机代理
测试机服务
S301：运行程序
S201：获取运行状态的程序的进程ID
S101：修改程序的Java代码
S102：编译Java代码，获得目标类文件
S103：上传目标类文件
S202：获取目标类文件的类名
S203：存储目标类文件
S204：将目标类文件的类名和存储路径组合成参数
S205：根据进程ID连接到虚拟机的进程
S206：将参数传递到虚拟机的进程
S302：将参数解析成目标类文件的类名和存储路径
S303：根据存储路径读取目标类文件
S304：从虚拟机加载的类文件中，查找与目标类文件同名的类对象
S305：采用redefineclasses函数，更新类对象

图3

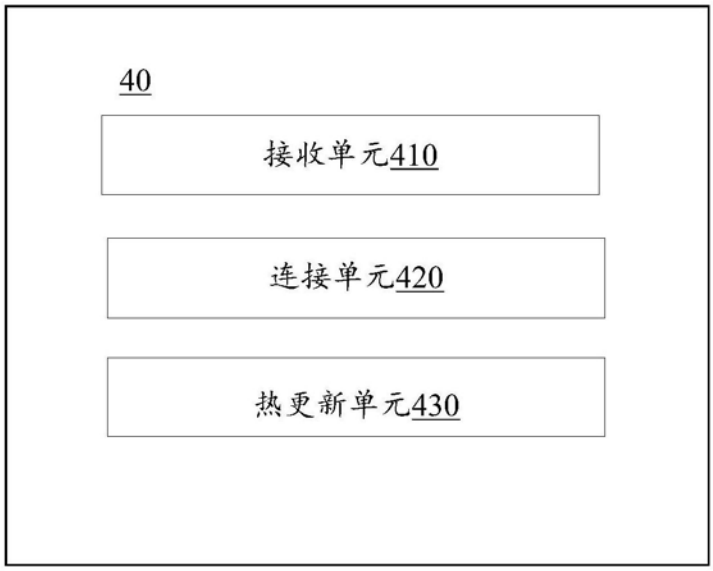

40
接收单元410
连接单元420
热更新单元430

图4


50
修改单元510
接收单元520
连接单元530
存储单元540
组合单元550
热更新单元560

图5

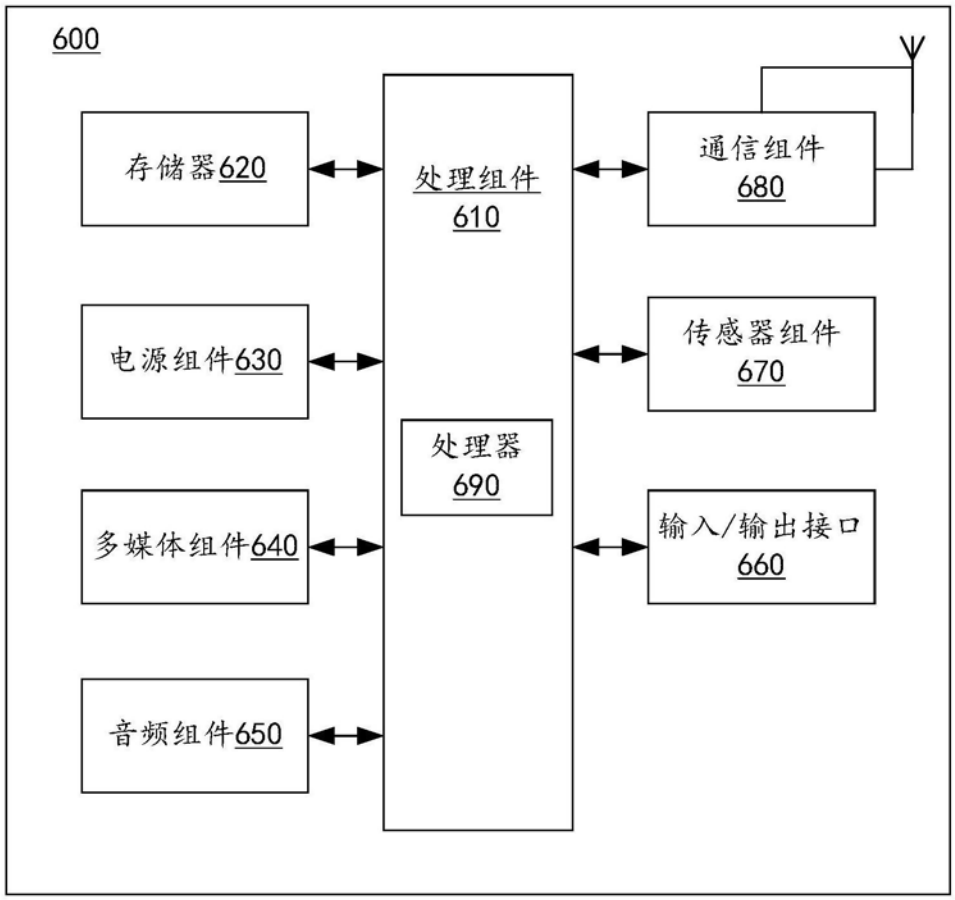

600
存储器620
处理组件
610
通信组件
680
电源组件630
传感器组件
670
处理器
690
多媒体组件640
输入/输出接口
660
音频组件650

图6

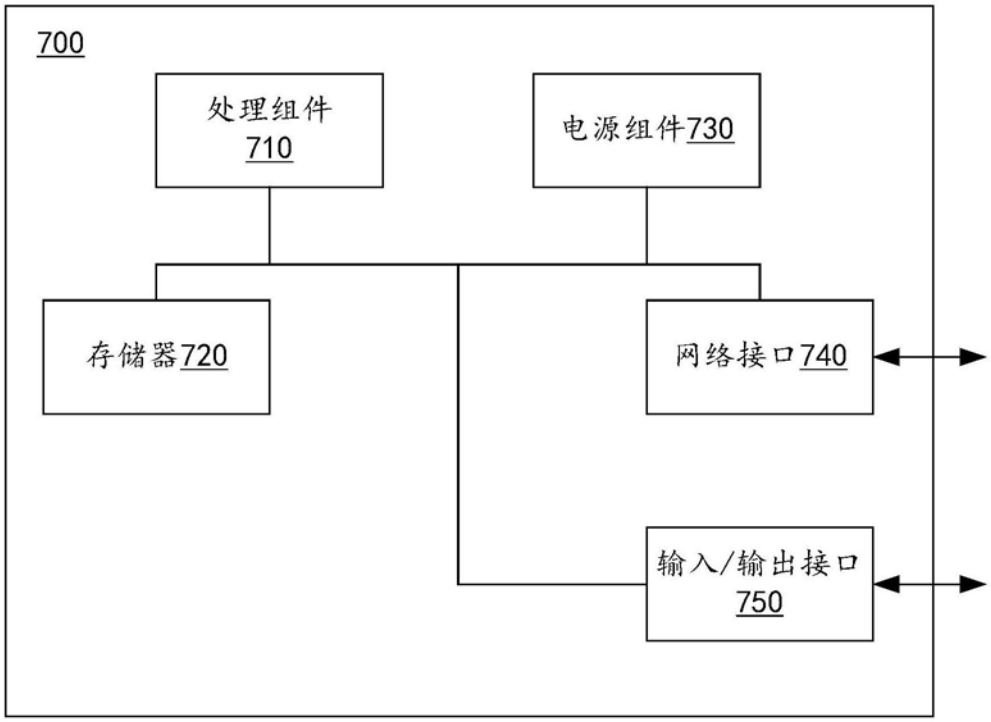


图7