(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 103136029 A
(43)申请公布日 2013.06.05

(21)申请号 201310079129.1

(22)申请日 2013.03.12

(71)申请人 无锡江南计算技术研究所
地址 214083 江苏省无锡市滨湖区军东新村 030 号

(72)发明人 张海军 唐大国 郑磊 李茜 叶俊

(74)专利代理机构 北京众合诚成知识产权代理有限公司 11246
代理人 龚燮英

(51)Int.Cl.
G06F 9/45(2006.01)

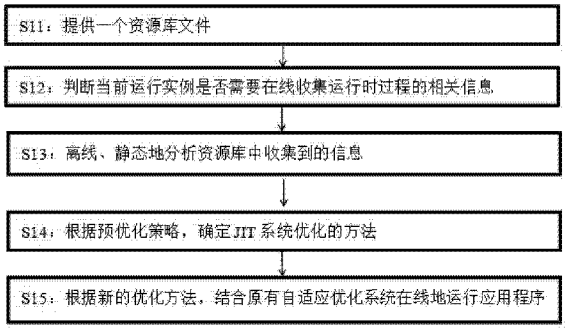
权利要求书1页 说明书5页 附图2页

(54)发明名称

即时编译系统自适应调优方法

(57)摘要

本发明提供了一种即时编译系统自适应调优方法。提供一个跨应用程序多次使用的资源库，其中资源库中的信息，针对每个运行实例是分开的，由此每个运行实例在资源库中的信息互不影响。判断当前运行实例是否需要在线收集即时编译过程的相关信息，即时编译系统通过运行时信息输出向资源库中写入未处理运行时信息，并且/或者从资源库中读取跨应用程序运行的运行时信息。离线、静态地分析资源库中收集到的未处理运行时信息以形成预计算的在线运行策略。通过运行时优化策略读取从资源库中读取计算出来的预计算的在线运行策略，并根据预计算的在线运行策略确定即时编译系统的优化方法。根据确定的优化方法，结合原有自适应优化系统在线地运行应用程序。



CN 103136029 A

1. 一种即时编译系统自适应调优方法，其特征在于包括：

提供一个跨应用程序多次使用的资源库，其中资源库中的信息，针对每个运行实例是分开的，由此每个运行实例在资源库中的信息互不影响；

离线、静态地分析资源库中收集到的未处理运行时信息以形成预计算的在线运行策略；

通过运行时优化策略读取从资源库中读取计算出来的预计算的在线运行策略，并根据预计算的在线运行策略确定即时编译系统的优化方法；

根据确定的优化方法，结合原有自适应优化系统在线地运行应用程序。

2. 根据权利要求 1 所述的即时编译系统自适应调优方法，其特征在于还包括：判断当前运行实例是否需要在线收集即时编译过程的相关信息，并且，即时编译系统通过运行时信息输出向资源库中写入未处理运行时信息。

3. 根据权利要求 1 或 2 所述的即时编译系统自适应调优方法，其特征在于还包括：判断当前运行实例是否需要在线收集即时编译过程的相关信息，并且，即时编译系统从资源库中读取跨应用程序运行的运行时信息。

4. 根据权利要求 1 或 2 所述的即时编译系统自适应调优方法，其特征在于还包括：定期地对资源库中的信息进行清理。

5. 根据权利要求 1 或 2 所述的即时编译系统自适应调优方法，其特征在于，预计算在线运行时策略包括：采取哪些优化选项，什么时候编译某个热方法，明确地内联或者禁止内联特定方法。

6. 根据权利要求 2 所述的即时编译系统自适应调优方法，其特征在于，所述即时编译系统自适应调优方法用于 Java 虚拟机的热点方法。

7. 根据权利要求 6 所述的即时编译系统自适应调优方法，其特征在于，即时编译过程的相关信息包括：方法执行的次数、何时加入到编译队列、方法编译需要的开销、优化执行获得的收益、virtual 方法调用的具体目标。

即时编译系统自适应调优方法

技术领域

[0001] 本发明涉及计算机技术领域，更具体地说，本发明涉及一种即时编译系统自适应调优方法。

背景技术

[0002] 在 Java 编程语言和环境中，即时编译(即时编译 compiler，just-in-time compiler)系统是一个把 Java 的字节码(包括需要被解释的指令的程序)转换成可以直接发送给处理器的指令的程序。当写好一个 Java 程序后，源语言的语句将由 Java 编译器编译成字节码，而不是编译成与某个特定的处理器硬件平台对应的指令代码(比如，Intel 的 Pentium 微处理器或 IBM 的 System/390 处理器)。字节码是可以发送给任何平台并且能在那个平台上运行的独立于平台的代码。

[0003] 即时编译优化系统一般都有动态运行时环境支持，如 Java 程序运行在 Java 虚拟机环境。和传统的静态编译的程序相比，即时编译优化都是发生在程序运行时，能够得到程序的复杂运行轨迹和全局运行时信息，这些信息给即时编译系统的编译优化带来了更多机会。

[0004] 动态特性的程序具有静态编译没有的运行时信息，利用编译和优化都发生在程序运行时的特性，采用较为复杂的指导程序进行何种优化的全局自适应运行时优化技术，收集动态编译信息与性能监测信息，获得更好的执行效率，更高的性能。

[0005] 但是，现有的即时编译系统普遍存在着如下缺陷：一方面，即时编译依赖的信息都是在程序运行时监测收集，但当程序执行完成后这些信息将直接被丢弃，下一次程序再执行的时候即时编译要再重新监测收集运行时优化需要的信息，以前收集的运行时信息不能够得到有效地利用，浪费了已有信息改进提高程序性能的机会。由于即时编译本地代码需要占用程序运行时间，要编译出优化程度更高的代码，所花费的时间可能更长；而且想要编译出优化程度更高的代码，解释器需要帮助编译器监控收集运行时信息，这对解释执行的速度也有影响。

[0006] 另一方面，即时编译系统都会提供相当多的优化特性和参数调节手段。这些选项在不同平台上对性能的影响效果也不同，许多用户面对如此多的可选参数往往都会无所适从，普通用户使用的优化选项通常在 3 个以内，高级用户使用的优化选项通常在 5 个以内，极少能用到 8 个以上，很少用户愿意尝试不同的优化选项，更细粒度的优化控制几乎不可能。

发明内容

[0007] 本发明所要解决的技术问题是针对现有技术中存在上述缺陷，提供一种结合运行时动态收集信息和静态离线信息的自适应调优方法，以便进一步提高即时编译系统运行的性能的即时编译系统自适应调优方法。

[0008] 根据本发明，提供了一种即时编译系统自适应调优方法，其包括：提供一个跨应用

程序多次使用的资源库，其中资源库中的信息，针对每个运行实例是分开的，由此每个运行实例在资源库中的信息互不影响；离线、静态地分析资源库中收集到的未处理运行时信息以形成预计算的在线运行策略；通过运行时优化策略读取从资源库中读取计算出来的预计算的在线运行策略，并根据预计算的在线运行策略确定即时编译系统的优化方法；根据确定的优化方法，结合原有自适应优化系统在线地运行应用程序。

[0009] 优选地，所述即时编译系统自适应调优方法还包括：判断当前运行实例是否需要在线收集即时编译过程的相关信息，并且，即时编译系统通过运行时信息输出向资源库中写入未处理运行时信息。

[0010] 优选地，所述即时编译系统自适应调优方法还包括：判断当前运行实例是否需要在线收集即时编译过程的相关信息，并且，即时编译系统从资源库中读取跨应用程序运行的运行时信息。

[0011] 优选地，所述即时编译系统自适应调优方法还包括：定期地对资源库中的信息进行清理。

[0012] 优选地，预计算在线运行时策略包括：采取哪些优化选项，什么时候编译某个热方法，明确地内联或者禁止内联特定方法。

[0013] 优选地，所述即时编译系统自适应调优方法用于 Java 虚拟机的热点方法。

[0014] 优选地，即时编译过程的相关信息包括：方法执行的次数、何时加入到编译队列、方法编译需要的开销、优化执行获得的收益、virtual 方法调用的具体目标。

[0015] 本发明公开的动静结合的即时编译系统自适应调优方法中，充分利用离线收集的信息，自动分析程序运行的性能瓶颈，降低了一些热点方法运行时动态信息收集的开销，提前对这些方法进行高性能的优化，解决了现有即时编译系统运行时过程中收集的性能信息数据的浪费问题，达到了进一步改进优化应用程序效率的效果。

附图说明

[0016] 结合附图，并通过参考下面的详细描述，将会更容易地对本发明有更完整的理解并且更容易地理解其伴随的优点和特征，其中：

[0017] 图 1 示意性地示出了根据本发明优选实施例的即时编译系统自适应调优方法的配置示意图。

[0018] 图 2 示意性地示出了根据本发明优选实施例的即时编译系统自适应调优方法的流程图。

[0019] 图 3 示意性地示出了根据本发明优选实施例的即时编译系统自适应调优方法的一个具体示例的流程图。

[0020] 需要说明的是，附图用于说明本发明，而非限制本发明。注意，表示结构的附图可能并非按比例绘制。并且，附图中，相同或者类似的元件标有相同或者类似的标号。

具体实施方式

[0021] 为了使本发明的内容更加清楚和易懂，下面结合具体实施例和附图对本发明的内容进行详细描述。

[0022] 本发明提供一种动静结合的自适应调优方法，该框架综合考虑即时编译系统优化

过程的“成本效益”问题和已存在的虚拟机自适应优化系统，成功地将离线获取的信息和在线收集的信息结合起来，进而指导热点程序是否能够提前优化，采用何种优化手段，何时进行编译，指导调用的短小方法是否需要立即内联等，从而使得即时编译的自适应优化系统获得更高的性能。

[0023] 通过动静结合的自适应调优框架，在执行过程中结合跨程序运行收集的离线运行时动态信息和在线的运行时动态信息，自适应地指导应用程序更早实施相应的高效优化，从而更快地为虚拟机选取最好的优化路径和方法，提高程序运行的自适应调优的运行效率。

[0024] 下面将具体描述本发明的优选实施例。

[0025] 图 2 示意性地示出了根据本发明优选实施例的即时编译系统自适应调优方法的流程图。

[0026] 如图 2 所示，根据本发明优选实施例的即时编译系统自适应调优方法包括 ：

[0027] 第一步骤 S11 ：提供一个跨应用程序多次使用的资源库 100。

[0028] 具体地，资源库 100 可以跨应用程序多次使用，保证应用程序间收集的信息不会浪费。

[0029] 第二步骤 S12 ：判断当前运行实例是否需要在线收集即时编译过程的相关信息。并且，即时编译系统可以根据需要通过运行时信息输出 10 向资源库 100 中写入未处理运行时信息 101，并且可以根据需要从资源库 100 中读取跨应用程序运行的运行时信息。

[0030] 资源库 100 中的信息，针对每个运行实例是分开的 ；每个运行实例在资源库 100 中的信息是互不影响的。

[0031] 向资源库 100 中写入或者从资源库 100 中读取运行时信息是可选的，特殊的情况都可以被忽略，由此所述第二步骤 S12 是可选的。

[0032] 第三步骤 S13 ：离线、静态地分析资源库 100 中收集到的未处理运行时信息 101 以形成预计算的在线运行策略 102。

[0033] 具体地，从即时编译系统中输出的未处理运行时信息 101 是丰富的，以便在资源分析阶段分析有效的运行时信息。

[0034] 而且，资源分析是离线分析的，不占用即时编译系统的开销。

[0035] 第四步骤 S14 ：通过运行时优化策略读取 20 从资源库 100 中读取第三步骤 S13 计算出来的预计算的在线运行策略 102，并根据预计算的在线运行策略 102，确定即时编译系统的优化方法。

[0036] 具体地，预计算在线运行时策略 102 提供给即时编译系统的是短小并明确的优化策略，比如包括采取哪些优化选项，什么时候编译某个热方法，明确地内联或者禁止内联某个（短小）方法等。

[0037] 第五步骤 S15 ：根据第四步骤 S14 确定的优化方法，在线地运行应用程序。

[0038] 优选地，可定期地对资源库 100 中的信息进行清理，从而清除资源库 100 中的无用信息。

[0039] 本发明上述实施例提出的动静结合的自适应调优框架不影响即时编译系统在线运行时信息的继续收集。而且，本发明上述实施例提出的动态结合的自适应调优方法明显地改进提高了程序自适应优化的性能。

[0040] 图 3 示意性地示出了根据本发明优选实施例的即时编译系统自适应调优方法的一个具体示例的流程图。其中以 Java 虚拟机的热点方法探测为例说明了即时编译系统自适应调优方法。

[0041] Java 虚拟机的编译对象一般通过解释执行时调用计数器统计方法执行的次数，当方法执行超过一个确定的阀值时，就会向即时编译器提交一个该方法的编译请求。在 Java 虚拟机执行过程中，如果能够根据程序上次运行热点方法的信息统计到资源库中，下次程序运行根据资源库中统计的信息确定该方法尽早执行，则该方法的统计信息过程将被节省。如果能够确定多个明确编译方法，则程序执行的效率将大大提高。方法内联的道理同样如此，如果能够将一些需要内联的代码较短、执行频率较高的 Java 方法尽早内联，将那些会导致方法逆优化的内联方法禁止内联，必定也能够改善 Java 虚拟机的性能。

[0042] 具体的流程结构，如图 3 所示，包括 ：

[0043] 初始地，如上所述，提供一个跨应用程序多次使用的资源库 100。具体地，资源库 100 可以跨应用程序多次使用，保证应用程序间收集的信息不会浪费。随后执行下述步骤。

[0044] 步骤 S21 ：执行至少一遍的运行时信息收集，并将这些收集的信息合并存储到资源库 100 的未处理运行时信息 101 中。

[0045] 具体地，在程序执行过程中，根据优化方向的不同需要通过虚拟机输出相关方法的运行时信息(即时编译过程的相关信息)，如 ：方法执行的次数、何时加入到编译队列、方法编译需要的开销、优化执行获得的收益、virtual 方法调用的具体目标等，并将这些收集的信息合并存储到资源库中。

[0046] 步骤 S22 ：静态且离线地分析资源信息库。具体地说，离线、静态地分析资源库 100 中收集到的未处理运行时信息 101 以形成预计算的在线运行策略 102。

[0047] 具体地，资源信息分析是基于成本效益模型来构建在线运行策略(预计算的在线运行策略 102)，它是静态且离线的，不占用应用程序运行的开销。继续以热点方法编译时机指导策略为例。Java 虚拟机编译对象的确定一般是通过解释执行时调用计数器统计方法的执行次数，当方法执行超过一个确定的阀值时，就会向即时编译器提交一个该方法的编译请求。在 Java 虚拟机执行过程中，如果能够根据程序上次运行时热点方法的信息统计到资源库中，下次程序运行根据资源库中统计的信息确定该方法尽早执行，则在运行过程中该方法的信息统计过程能够得到节省。如果能够确定多个明确编译方法，则程序执行的效率将大大提高。

[0048] 以尽早确定编译方法，节省相关方法信息监测收集的开销为目的，基于 Java 虚拟机，本发明设计并实现了一套离线选择算法。

[0049] 首先基于编译的热点方法构建一个模型，它基于三个数据 ：

[0050]

Count(M)： 根据资源库中运行时信息估计方法执行的次数。 Cost(M)： 表示方法 M 编译需要的时间。 Speedup(M)：表示方法 M 相对解释执行版本执行带来的预期加速。

[0051] 这三个数据是基于离线测试估计的数据。其中，Speedup(M) 是根据个方法运行过程中手机的数据估算而来，估算公式如下 ：

[0052] $speedup=(N_I-N_N)*(T_I/T_N)-C_T$

[0053] 该公式中，N_I 和 N_N 分别指某方法在解释模式和编译模式执行的次数，T_I 和 N_N 分别指某方法在解释模式和编译模式下执行每次需要的平均时间，而 C_T 则代表该方法的编译开销。

[0054] 基于这几个定义的数量，就能够根据收集的复杂未处理运行时信息判断热点方法何时进行编译优化，该方法描述如下。该算法描述中，Timeall 则表示程序执行预估计的执行时间。s1、s2 和 s3 都是预设置的较小的常量经验值，用于控制编译队列里待编译方法的数量，因为如果编译开销过重，则它带来的收益将得不偿失。而算法中用到的 Threshold 则是 Java 虚拟机中触发热点方法即时编译而设置的默认阈值。

[0055] 步骤 S23 ：确定明确且短小的在线优化策略。

[0056] 具体地，动静结合的自适应调优用来指导 Java 虚拟机优化的预计算在线运行策略是短小且明确的。例如，用来指导进行何种优化，在什么时候进行优化。指导热点方法何时进行优化的策略可可分为 3 种 ：normal 指正常达到计数器阈值时才进行编译，earlycompile 和 exclude 则分别表示提前编译和不需要进行编译，进一步地还有 reinterpret、recompile 等操作表示当前编译的本地方法发生 uncommon_trap 时，根据附加信息重新选择是立即编译还是要重新根据在线信息进行自适应编译优化。优化策略还能够指导不同目标机器的堆栈分配 ；指导方法具体内联哪个 virtual 调用的方法等。

[0057] 步骤 S24 ：优选地，定期地对资源库 100 中的信息进行清理，从而清除资源库 100 中的无用信息。

[0058] 具体地，动静结合的自适应调优框架中资源库的信息和计算的在线运行策略在不同机器或不同时间下运行，需要收集的信息可能会有变化。因此，资源库中不常用的老的、静态信息需要定期清除，防止资源库变得臃肿。

[0059] 本发明公开的动静结合的即时编译系统自适应调优方法中，充分利用离线收集的信息，自动分析程序运行的性能瓶颈，降低了一些热点方法运行时动态信息收集的开销，提前对这些方法进行高性能的优化，解决了现有即时编译系统运行时过程中收集的性能信息数据的浪费问题，达到了进一步改进优化应用程序效率的效果。

[0060] 此外，需要说明的是，除非特别指出，否则说明书中的术语“第一”、“第二”、“第三”等描述仅仅用于区分说明书中的各个组件、元素、步骤等，而不是用于表示各个组件、元素、步骤之间的逻辑关系或者顺序关系等。

[0061] 可以理解的是，虽然本发明已以较佳实施例披露如上，然而上述实施例并非用以限定本发明。对于任何熟悉本领域的技术人员而言，在不脱离本发明技术方案范围情况下，都可利用上述揭示的技术内容对本发明技术方案作出许多可能的变动和修饰，或修改为等同变化的等效实施例。因此，凡是未脱离本发明技术方案的内容，依据本发明的技术实质对以上实施例所做的任何简单修改、等同变化及修饰，均仍属于本发明技术方案保护的范围内。

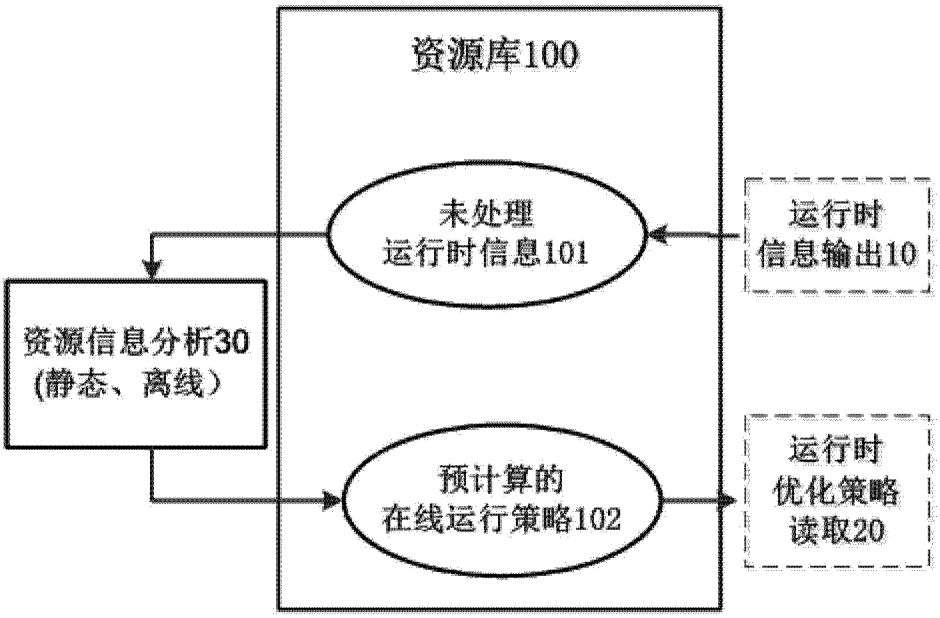

资源库100
未处理
运行时信息101
运行时
信息输出10
资源信息分析30
(静态、离线)
预计算的
在线运行策略102
运行时
优化策略
读取20

图 1

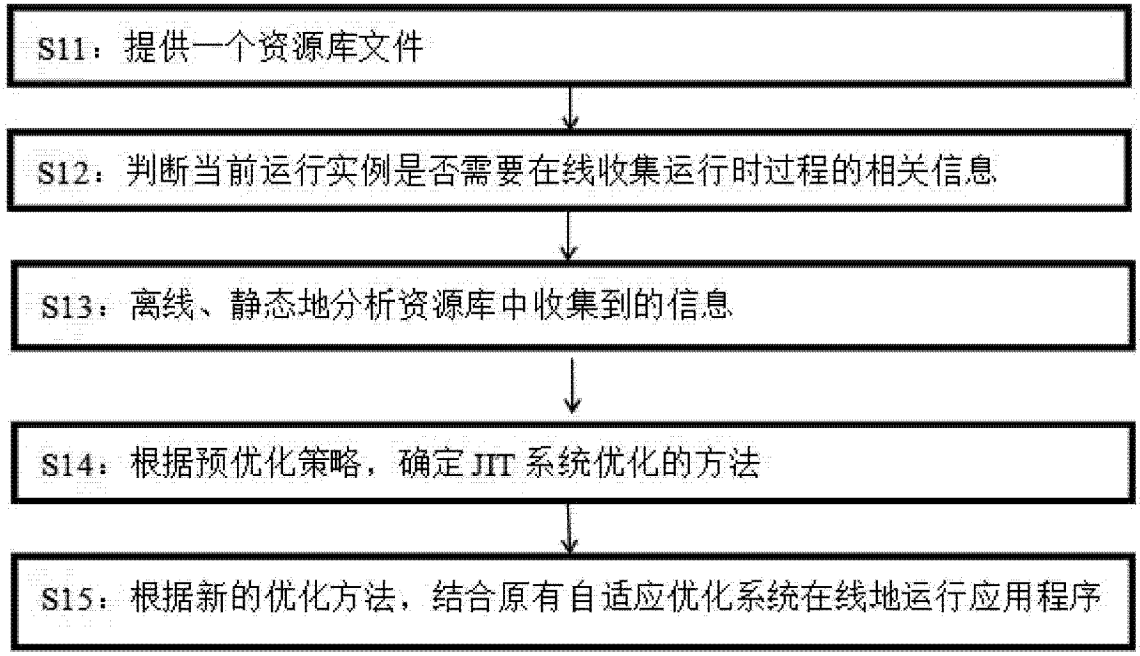

S11：提供一个资源库文件
S12：判断当前运行实例是否需要在线收集运行时过程的相关信息
S13：离线、静态地分析资源库中收集到的信息
S14：根据预优化策略，确定 JIT 系统优化的方法
S15：根据新的优化方法，结合原有自适应优化系统在线地运行应用程序

图 2

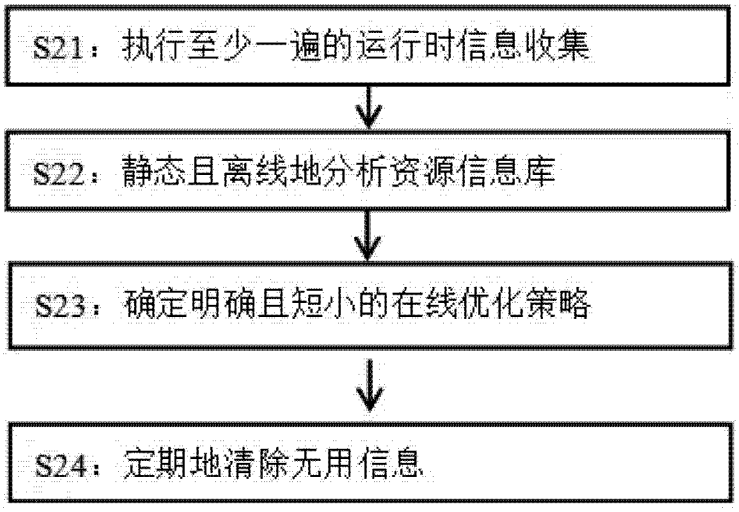

S21：执行至少一遍的运行时信息收集
S22：静态且离线地分析资源信息库
S23：确定明确且短小的在线优化策略
S24：定期地清除无用信息

图 3