(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 114174983 B
(45)授权公告日 2022.10.11

(21)申请号 202080052336.6
(22)申请日 2020.07.24
(65)同一申请的已公布的文献号
申请公布号 CN 114174983 A
(43)申请公布日 2022.03.11
(30)优先权数据
16/527,362 2019.07.31 US
(85)PCT国际申请进入国家阶段日
2022.01.19
(86)PCT国际申请的申请数据
PCT/IB2020/057017 2020.07.24
(87)PCT国际申请的公布数据
WO2021/019401 EN 2021.02.04
(73)专利权人 国际商业机器公司
地址 美国纽约
(72)发明人 I·A·艾尔兰 R·科普兰 A·基尔斯特拉 D·K·西格瓦特 孝寿俊彦
(74)专利代理机构 北京市中咨律师事务所 11247
专利代理师 于静 刘薇
(51)Int.Cl.
G06F 8/40(2006.01)
(56)对比文件
CN 102622260 A,2012.08.01
US 2014082597 A1,2014.03.20
CN 106462434 A,2017.02.22
审查员 江亚琳

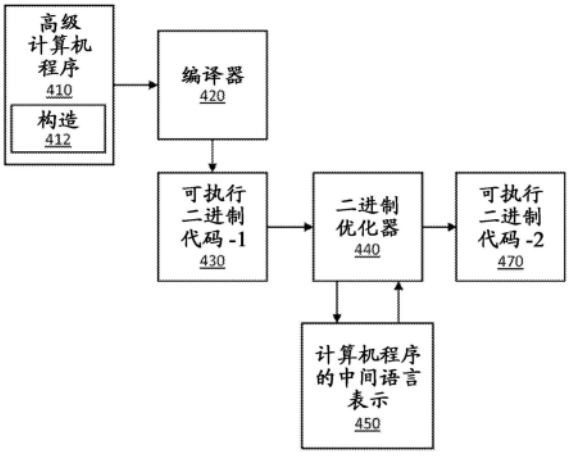
权利要求书5页 说明书16页 附图6页

(54)发明名称

用于高级构造的优化的自动验证的方法和系统

(57)摘要

一种用于高级构造的优化的自动验证的方法,包括通过编译包括高级构造的计算机程序来生成第一可执行代码。编译包括生成用于高级构造的第一组机器指令,并且存储用于高级构造的编译时间信息。该方法还包括优化第一可执行代码,其包括将第一可执行代码转换成中间语言表示。优化还包括生成第二可执行代码。为此,该方法包括从中间语言表示生成用于高级构造的第二组机器指令。如果第一组机器指令和第二组机器指令的行为匹配,则第二组机器指令被包括在第二可执行代码中,否则包括第一组机器指令。



CN 114174983 B

1.一种用于高级构造的优化的自动验证的计算机实现的方法,所述方法包括:

由编译器通过编译计算机程序来生成第一可执行代码,所述计算机程序包括高级构造,所述高级构造是高级语言或机器语言的语句或指令,其足够复杂使得其可执行表示将是多个低级机器指令的复杂代码序列,并且所述编译包括:

生成用于所述高级构造的第一组机器指令;以及

在所述第一可执行代码中存储用于所述高级构造的编译时间信息,所述编译时间信息包括所述高级构造的身份、所述第一组机器指令的位置、所述第一组机器指令的输入和输出的位置;以及

由二进制优化器对所述第一可执行代码进行优化,所述优化包括:

将所述第一可执行代码转换成所述计算机程序的中间语言IL表示,所述IL表示包括基于所述编译时间信息对应于所述高级构造的一组IL指令;以及

基于所述计算机程序的所述IL表示生成第二可执行代码,其中生成所述第二可执行代码包括:

基于所述高级构造的所述IL表示生成用于所述高级构造的第二组机器指令;

基于所述第一组机器指令和所述第二组机器指令在行为上匹配的确定,将所述第二组机器指令包括在所述第二可执行代码中,以及

基于所述第一组机器指令和所述第二组机器指令在行为上不匹配的确定,将所述第一组机器指令包括在所述第二可执行代码中。

2.根据权利要求1所述的计算机实现的方法,其中,所述第一组机器指令和所述第二组机器指令在行为上匹配的所述确定是基于可满足性求解器。

3.根据权利要求1或2所述的计算机实现的方法,其中,确定所述第一组机器指令和所述第二组机器指令在行为上匹配包括:

生成包括多个测试的测试向量;

针对与所述高级构造相关联的所述第一组机器指令和所述第二组机器指令执行所述测试向量中的每个测试;以及

比较所述测试向量中的所述每个测试的输出,其中,基于针对所述第一组机器指令和所述第二组机器指令的所述输出中的每个输出匹配,确定所述行为匹配。

4.根据权利要求3所述的计算机实现的方法,其中,执行所述测试向量中的测试包括:

创建用于所述第一组机器指令的第一包装器函数;

根据要执行的所述测试用输入来执行所述第一包装器函数,并且收集第一组输出;

为所述第二组机器指令创建第二包装器函数;以及

根据要执行的所述测试用所述输入来执行所述第二包装器函数,以及收集第二组输出。

5.根据权利要求4所述的计算机实现的方法,其中,所述第一包装器函数被封装在第一异常处理程序中,并且所述第二包装器函数被封装在第二异常处理程序中。

6.根据权利要求3所述的计算机实现的方法,其中,基于测试的第一输出是所述第一组机器指令的异常,以及所述测试的第二输出是所述第二组机器指令的所述异常,确定所述行为匹配。

7.根据权利要求3所述的计算机实现的方法,其中,所述编译器是编译器的第一实例,

并且具有与所述编译器的第一实例的设置不同的设置的所述编译器的第二实例用于生成所述第二可执行代码。

8.根据权利要求1或2所述的计算机实现的方法，其中，基于对于一组输入，所述第一组机器指令和所述第二组机器指令在行为上不匹配的确定：

生成第三可执行代码，所述第三可执行代码包括所述第一组机器指令、所述第二组机器指令以及检查对所述高级构造的输入的附加函数，并且响应于所述输入来自预定组输入，引起所述第一组机器指令的执行，否则引起所述第二组机器指令的执行。

9.根据权利要求8所述的方法，其中，确定所述第一组机器指令和所述第二组机器指令在行为上匹配包括：

生成包括多个测试的测试向量；

针对与所述高级构造相关联的所述第一组机器指令和所述第二组机器指令执行所述测试向量中的每个测试；以及

比较所述测试向量中的所述每个测试的输出，其中，基于针对所述第一组机器指令和所述第二组机器指令的所述输出中的每个输出匹配，确定所述行为匹配。

10.根据权利要求9所述的计算机实现的方法，其中，执行所述测试向量中的测试包括：

创建用于所述第一组机器指令的第一包装器函数；

根据要执行的所述测试用输入来执行所述第一包装器函数，并且收集第一组输出；

创建用于所述第二组机器指令的第二包装器函数；以及

根据要执行的所述测试用所述输入来执行所述第二包装器函数，以及收集第二组输出。

11.一种用于高级构造的优化的自动验证的系统，包括：

存储器；以及

与所述存储器耦合的处理器，所述处理器被配置为执行用于高级构造的优化的自动验证的方法，所述方法包括：

由编译器通过编译计算机程序来生成第一可执行代码，所述计算机程序包括高级构造，所述高级构造是高级语言或机器语言的语句或指令，其足够复杂使得其可执行表示将是多个低级机器指令的复杂代码序列，并且所述编译包括：

生成用于所述高级构造的第一组机器指令；以及

在所述第一可执行代码中存储用于所述高级构造的编译时间信息，所述编译时间信息包括所述高级构造的身份、所述第一组机器指令的位置、所述第一组机器指令的输入和输出的位置；以及

由二进制优化器对所述第一可执行代码进行优化，所述优化包括：

将所述第一可执行代码转换成中间语言IL表示，所述IL表示包括基于所述编译时间信息对应于所述高级构造的一组IL指令；

基于所述IL表示生成第二可执行代码，所述第二可执行代码包括用于所述高级构造的第二组机器指令；

由所述二进制优化器将所述第一组机器指令和所述第二组机器指令进行比较；以及

基于所述第一组机器指令和所述第二组机器指令在行为上匹配的确定，用所述第二可执行代码替换所述第一可执行代码。

12.根据权利要求11所述的系统,其中,所述第一组机器指令和所述第二组机器指令在所述行为上匹配的所述确定是基于可满足性求解器。

13.根据权利要求11或12所述的系统,其中,确定所述第一组机器指令和所述第二组机器指令在行为上匹配包括:

生成包括多个测试的测试向量;

针对与所述高级构造相关联的所述第一组机器指令和所述第二组机器指令执行所述测试向量中的每个测试;以及

比较所述测试向量中的所述每个测试的输出,其中,基于针对所述第一组机器指令和所述第二组机器指令的所述输出中的每个输出匹配,确定所述行为匹配。

14.根据权利要求13所述的系统,其中,执行所述测试向量中的测试包括:

创建用于所述第一组机器指令的第一包装器函数;

根据要执行的所述测试用输入来执行所述第一包装器函数,并且收集第一组输出;

为所述第二组机器指令创建第二包装器函数;以及

根据要执行的所述测试用所述输入来执行所述第二包装器函数,以及收集第二组输出。

15.根据权利要求14所述的系统,其中,所述第一包装器函数被封装在第一异常处理程序中,并且所述第二包装器函数被封装在第二异常处理程序中。

16.根据权利要求13所述的系统,其中,基于测试的第一输出是所述第一组机器指令的异常,以及所述测试的第二输出是所述第二组机器指令的所述异常,确定所述行为匹配。

17.根据权利要求13所述的系统,其中,所述编译器是编译器的第一实例,并且具有与所述编译器的第一实例的设置不同的设置的所述编译器的第二实例用于生成所述第二可执行代码。

18.根据权利要求11或12所述的系统,其中,基于确定对于一组输入,所述第一组机器指令和所述第二组机器指令在行为上不匹配:

生成第三可执行代码,所述第三可执行代码包括所述第一组机器指令、所述第二组机器指令以及检查对所述高级构造的输入的附加函数,并且响应于所述输入来自预定组输入,引起所述第一组机器指令的执行,否则引起所述第二组机器指令的执行。

19.根据权利要求18所述的系统,其中,确定所述第一组机器指令和所述第二组机器指令在行为上匹配包括:

生成包括多个测试的测试向量;

针对与所述高级构造相关联的所述第一组机器指令和所述第二组机器指令执行所述测试向量中的每个测试;以及

比较所述测试向量中的所述每个测试的输出,其中,基于针对所述第一组机器指令和所述第二组机器指令的所述输出中的每个输出匹配,确定所述行为匹配。

20.一种用于高级构造的优化的自动验证的计算机可读存储介质,所述计算机可读存储介质可由处理电路读取并且存储用于由所述处理电路执行以执行根据权利要求1至10中任一项所述的方法的指令。

21.一种用于高级构造的优化的自动验证的计算机实现的方法,所述方法包括:

由编译器通过编译计算机程序来生成第一可执行代码,所述计算机程序包括高级构

造，所述高级构造是高级语言或机器语言的语句或指令，其足够复杂使得其可执行表示将是多个低级机器指令的复杂代码序列，并且所述编译包括：

生成用于所述高级构造的第一组机器指令；以及

在所述第一可执行代码中存储用于所述高级构造的编译时间信息，所述编译时间信息包括所述高级构造的身份、所述第一组机器指令的位置、所述第一组机器指令的输入和输出的位置；以及

由优化器对所述第一可执行代码进行优化，所述优化包括：

将所述第一可执行代码转换成中间语言IL表示，所述IL表示包括基于所述编译时间信息对应于所述高级构造的一组IL指令；

基于所述IL表示生成第二可执行代码，所述第二可执行代码包括用于所述高级构造的第二组机器指令；以及

基于对于一组输入，所述第一组机器指令和所述第二组机器指令在行为上不匹配的确定：

生成第三可执行代码，所述第三可执行代码包括所述第一组机器指令、所述第二组机器指令以及检查对所述高级构造的输入的附加函数，并且响应于所述输入来自预定组输入，引起所述第一组机器指令的执行，否则引起所述第二组机器指令的执行。

22.根据权利要求21所述的计算机实现的方法，其中，确定所述第一组机器指令和所述第二组机器指令在行为上匹配包括：

生成包括多个测试的测试向量；

针对与所述高级构造相关联的所述第一组机器指令和所述第二组机器指令执行所述测试向量中的每个测试；以及

比较所述测试向量中的所述每个测试的输出，其中，基于针对所述第一组机器指令和所述第二组机器指令的所述输出中的每个输出匹配，确定所述行为匹配。

23.根据权利要求22所述的计算机实现的方法，其中，执行所述测试向量中的测试包括：

创建用于所述第一组机器指令的第一包装器函数；

根据要执行的所述测试用输入来执行所述第一包装器函数，并且收集第一组输出；

创建用于所述第二组机器指令的第二包装器函数；以及

根据要执行的所述测试用所述输入来执行所述第二包装器函数，以及收集第二组输出。

24.一种用于高级构造的优化的自动验证的系统，包括：

存储器；以及

与所述存储器耦合的处理器，所述处理器被配置为执行用于高级构造的优化的自动验证的方法，所述方法包括：

由编译器通过编译计算机程序来生成第一可执行代码，所述计算机程序包括高级构造，所述高级构造是高级语言或机器语言的语句或指令，其足够复杂使得其可执行表示将是多个低级机器指令的复杂代码序列，并且所述编译包括：

生成用于所述高级构造的第一组机器指令；以及

在所述第一可执行代码中存储用于所述高级构造的编译时间信息，所述编译时间信息

包括所述高级构造的身份、所述第一组机器指令的位置、所述第一组机器指令的输入和输出的位置;以及

由优化器对所述第一可执行代码进行优化,所述优化包括:

将所述第一可执行代码转换成中间语言IL表示,所述IL表示包括基于所述编译时间信息对应于所述高级构造的一组IL指令;

基于所述IL表示生成第二可执行代码,所述第二可执行代码包括用于所述高级构造的第二组机器指令;以及

基于对于一组输入,所述第一组机器指令和所述第二组机器指令在行为上不匹配的确定:

生成第三可执行代码,所述第三可执行代码包括所述第一组机器指令、所述第二组机器指令以及检查对所述高级构造的输入的附加函数,并且响应于所述输入来自预定组输入,引起所述第一组机器指令的执行,否则引起所述第二组机器指令的执行。

25.根据权利要求24所述的系统,其中,确定所述第一组机器指令和所述第二组机器指令在行为上匹配包括:

生成包括多个测试的测试向量;

针对与所述高级构造相关联的所述第一组机器指令和所述第二组机器指令执行所述测试向量中的每个测试;以及

比较所述测试向量中的所述每个测试的输出,其中,基于针对所述第一组机器指令和所述第二组机器指令的所述输出中的每个输出匹配,确定所述行为匹配。

用于高级构造的优化的自动验证的方法和系统

技术领域

[0001] 本公开总体上涉及计算技术,并且更具体地涉及用于提高可执行程序代码的性能的方法、计算机程序产品和数据处理系统。

背景技术

[0002] 计算机软件包括由数据处理系统执行的指令集。计算机软件提供的指令使数据处理系统能够起到诸如字处理设备、电子表格设备或因特网浏览设备、数据存储库等的作用。存在能够使用计算机软件的各种各样的不同数据处理系统。因此,如本文所使用的,术语“数据处理系统”旨在具有广泛的含义,并且可以包括个人计算机、膝上型计算机、掌上型计算机、手持式计算机、网络计算机、服务器、大型机、工作站、蜂窝电话和类似的无线设备、个人数字助理和可安装计算机软件的其他电子设备。术语“计算机”、“计算机软件”、“计算机程序”、“计算机编程”、“软件”、“软件程序”和相关术语旨在具有类似广泛的含义。

[0003] 通常,现代计算机软件最初使用程序员可理解的语法构造以“高级”计算机编程语言编写以表示在软件中体现的指令。高级计算机编程语言是有用的,因为它们的语法构造使得程序员更容易创建计算机软件,因为它们不需要以数据处理系统将直接理解的语言编写指令。以这种语言编写指令将更加困难,因为这种语言与任何人类语言几乎或没有相似之处。

[0004] 然而,以高级计算机编程语言编写的指令通常不能由数据处理系统直接理解和实施。因此,在数据处理系统可以使用以高级计算机编程语言编写的计算机程序之前,首先将它“编译”成目标数据处理系统将理解的语言。编译是通常由被称为“编译器”的计算机程序执行的过程,其中,高级计算机编程语言的语法构造本质上被翻译成将由目标数据处理系统(可能通过中间软件层)理解的语言的指令。“编译”或“汇编”过程的结果称为“可执行代码”或“二进制代码”,即数据处理系统可直接或通过中间软件层执行的计算机程序代码,目标数据处理系统将理解的指令被表示为“机器”指令。

[0005] 高级计算机编程语言可被视为属于两种宽泛类型之一:静态编译的语言和动态编译的语言。在静态编译的语言中,编译过程在任何代码被执行之前单次执行,并且编译的结果是可由数据处理系统直接实现而无需任何中间软件层的可执行代码。静态编译的语言包括C、C++、FORTRAN、PL/I、COBOL和Ada。在动态编译的语言(诸如Java™)中,源代码首先被编译成可由中间软件层(诸如Java虚拟机(JVM))实现的中间形式。在Java中,该中间形式被称为“字节码”。通常,但不一定,中间软件层将在每次计算机程序运行时执行附加编译,通常将源代码的中间形式转换成可由数据处理系统直接执行的可执行代码。Java和所有基于Java的商标和标志是Oracle和/或其子公司的商标或注册商标。

[0006] 因此,在本领域中需要解决上述问题。

发明内容

[0007] 从第一方面来看,本发明提供了一种用于高级构造的优化的自动验证的计算机实

现的方法，所述方法包括：由编译器通过编译计算机程序来生成第一可执行代码，所述计算机程序包括高级构造，并且所述编译包括：生成用于所述高级构造的第一组机器指令；以及在所述第一可执行代码中存储所述高级构造的编译时间信息；以及由二进制优化器优化所述第一可执行代码，所述优化包括：将所述第一可执行代码转换成所述计算机程序的中间语言(IL)表示，所述IL表示包括基于所述编译时间信息对应于所述高级构造的一组IL指令；以及基于所述计算机程序的所述IL表示生成第二可执行代码，其中，生成所述第二可执行代码包括：基于所述高级构造的所述IL表示生成用于所述高级构造的第二组机器指令；基于所述第一组机器指令和所述第二组机器指令的行为匹配的确定，将所述第二组机器指令包括在所述第二可执行代码中，以及基于所述第一组机器指令和所述第二组机器指令的行为不匹配的确定，将所述第一组机器指令包括在所述第二可执行代码中。

[0008] 从另一方面来看，本发明提供一种系统，包括：存储器；以及与所述存储器耦合的处理器，所述处理器被配置为执行用于高级构造的优化的自动验证的方法，所述方法包括：由编译器通过编译计算机程序生成第一可执行代码，所述计算机程序包括高级构造，并且所述编译包括：生成用于所述高级构造的第一组机器指令；以及在所述第一可执行代码中存储所述高级构造的编译时间信息；以及由二进制优化器优化所述第一可执行代码，所述优化包括：将所述第一可执行代码转换成中间语言(IL)表示，所述IL表示包括基于所述编译时间信息对应于所述高级构造的一组IL指令；基于所述IL表示生成第二可执行代码，所述第二可执行代码包括用于所述高级构造的第二组机器指令；由所述二进制优化器对所述第一组机器指令和所述第二组机器指令进行比较；以及基于所述第一组机器指令和所述第二组机器指令在行为上匹配的确定，用所述第二可执行代码替换所述第一可执行代码。

[0009] 从另一方面来看，本发明提供了一种高级构造的优化的计算机程序产品自动验证，该计算机程序产品包括计算机可读存储介质，该计算机可读存储介质可由处理电路读取并且存储用于由该处理电路执行以便执行用于执行本发明的步骤的方法的指令。

[0010] 从另一方面来看，本发明提供一种存储在计算机可读介质上并且可加载到数字计算机的内部存储器中的计算机程序，该计算机程序包括当所述程序在计算机上运行时用于执行本发明的步骤的软件代码部分。

[0011] 从又一方面来看，本发明提供一种计算机程序产品，包括计算机可读存储介质，所述计算机可读存储介质具有与其体现的程序指令，所述程序指令可由计算机处理器执行以使所述计算机处理器执行用于高级构造的优化的自动验证的方法，所述方法包括：由编译器通过编译计算机程序来生成第一可执行代码，所述计算机程序包括高级构造，并且所述编译包括：生成用于所述高级构造的第一组机器指令；以及在所述第一可执行代码中存储所述高级构造的编译时间信息；以及由二进制优化器优化所述第一可执行代码，所述优化包括：将所述第一可执行代码转换成所述计算机程序的中间语言(IL)表示，所述IL表示包括基于所述编译时间信息对应于所述高级构造的一组IL指令；以及基于所述计算机程序的所述IL表示生成第二可执行代码，其中，生成所述第二可执行代码包括：基于所述高级构造的所述IL表示生成用于所述高级构造的第二组机器指令；基于所述第一组机器指令和所述第二组机器指令的行为匹配的确定，将所述第二组机器指令包括在所述第二可执行代码中，以及基于所述第一组机器指令和所述第二组机器指令的行为不匹配的确定，将所述第一组机器指令包括在所述第二可执行代码中。

[0012] 从又一方面来看，本发明提供了一种用于高级构造的优化的自动验证的计算机实现的方法，所述方法包括：由编译器通过编译计算机程序来生成第一可执行代码，所述计算机程序包括高级构造，并且所述编译包括：生成用于所述高级构造的第一组机器指令；以及在所述第一可执行代码中存储用于所述高级构造的编译时间信息；以及由优化器优化所述第一可执行代码，所述优化包括：将所述第一可执行代码转换成中间语言(IL)表示，所述IL表示包括基于所述编译时间信息对应于所述高级构造的一组IL指令；基于所述IL表示生成第二可执行代码，所述第二可执行代码包括用于所述高级构造的第二组机器指令；以及基于确定对于一组输入，所述第一组机器指令和所述第二组机器指令在行为上不匹配：生成第三可执行代码，所述第三可执行代码包括第一组机器指令、第二组机器指令以及检查对高级构造的输入的附加函数，并且响应于输入来自预定组输入，导致所述第一组机器指令的执行，否则导致所述第二组机器指令的执行。

[0013] 从又一方面来看，本发明提供一种系统，包括：存储器；以及与所述存储器耦合的处理器，所述处理器被配置为执行用于高级构造的优化的自动验证的方法，所述方法包括：由编译器通过编译计算机程序生成第一可执行代码，所述计算机程序包括高级构造，并且所述编译包括：生成用于所述高级构造的第一组机器指令；以及在所述第一可执行代码中存储所述高级构造的编译时间信息；以及由优化器优化所述第一可执行代码，所述优化包括：将所述第一可执行代码转换成中间语言(IL)表示，所述IL表示包括基于所述编译时间信息对应于所述高级构造的一组IL指令；基于所述IL表示生成第二可执行代码，所述第二可执行代码包括用于所述高级构造的第二组机器指令；以及基于确定对于一组输入，所述第一组机器指令和所述第二组机器指令在行为上不匹配：生成第三可执行代码，所述第三可执行代码包括所述第一组机器指令，第二组机器指令，以及检查对高级构造的输入的附加函数，并且响应于输入来自预定组输入，导致所述第一组机器指令的执行，否则导致所述第二组机器指令的执行。

[0014] 根据本发明的一个或多个实施例，一种用于高级构造的优化的自动验证的计算机实现的方法，包括由编译器通过编译计算机程序生成第一可执行代码。所述计算机程序包括高级构造，并且所述编译包括：生成用于所述高级构造的第一组机器指令；以及在所述第一可执行代码中存储用于所述高级构造的编译时间信息。该方法还包括通过二进制优化器来优化第一可执行代码。优化包括将第一可执行代码转换成计算机程序的中间语言(IL)表示。IL表示包括对应于基于编译时间信息的高级构造的一组IL指令。所述优化进一步包括基于所述计算机程序的IL表示生成第二可执行代码，其中生成所述第二可执行代码包括基于所述高级构造的所述IL表示生成用于所述高级构造的第二组机器指令。基于第一组机器指令和第二组机器指令在行为上匹配的确定，第二组机器指令被包括在第二可执行代码中。替代地，基于第一组机器指令和第二组机器指令在行为上不匹配的确定，第一组机器指令被包括在第二可执行代码中。

[0015] 在一个或多个示例中，确定机器指令的第一集合和机器指令的第二集合在行为上匹配是基于可满足性求解器。确定所述第一组机器指令和所述第二组机器指令在行为上匹配包括：生成包括多个测试的测试向量；针对与所述高级构造相关联的所述第一组机器指令和所述第二组机器指令，执行所述测试向量中的每个测试。该确定进一步包括将测试向量中的每个测试的输出进行比较，其中基于针对第一组机器指令和第二组机器指令的输出

中的每个输出匹配来确定行为匹配。

[0016] 在所述测试向量中执行测试包括:创建用于所述第一组机器指令的第一包装器函数,以及根据要执行的测试用输入执行所述第一包装器函数,以及收集第一组输出。进一步,执行测试包括为第二组机器指令创建第二包装器函数,以及根据要执行的测试用输入来执行第二包装器函数。执行所述测试进一步包含收集第二组输出。在一个或多个示例中,第一包装器函数被封装在第一异常处理程序中,并且第二包装器函数被封装在第二异常处理程序中。在一个或多个示例中,基于测试的第一输出是第一组机器指令的异常并且所述测试的第二输出是第二组机器指令的所述异常来确定所述行为匹配。

[0017] 在一个或多个示例中,存储用于高级构造的编译时间信息包括存储对与高级构造对应的所述测试的输入。

[0018] 在一个或多个示例中,编译时间信息包括高级构造的身份、第一组机器指令的位置、第一组机器指令的输入和输出的位置。

[0019] 在一个或多个示例中,第一编译器是编译器的第一实例,并且具有与第一编译器的设置不同的设置的所述编译器的第二实例用于生成第二可执行代码。

[0020] 根据本发明的一个或多个实施例,一种用于高级构造的优化的自动验证的方法包括由编译器通过编译计算机程序来生成第一可执行代码,该计算机程序包括高级构造。所述编译包括生成用于所述高级构造的第一组机器指令,以及在所述第一可执行代码中存储用于所述高级构造的编译时间信息。该方法进一步包括由优化器对第一可执行代码进行优化。优化包括将第一可执行代码转换成程序的IL表示,转换包括基于编译时间信息将高级构造转换成程序的IL表示。所述优化包括基于所述程序的所述IL表示生成第二可执行代码,所述第二可执行代码包括基于其IL表示的用于所述高级构造的第二组机器指令。所述优化还包括:基于确定对于一组输入,所述第一组机器指令和所述第二组机器指令在行为上不匹配,生成第三可执行代码,所述第三可执行代码包括所述第一组机器指令、所述第二组机器指令、以及检查对所述高级构造的输入的附加函数,以及响应于所述输入来自预定组输入,引起所述第一组机器指令的执行,否则引起所述第二组机器指令的执行。

[0021] 在一个或多个示例中,确定第一组指令和第二组机器指令在行为上匹配包括生成包括多个测试的测试向量,并且针对与高级构造相关联的第一组指令和第二组机器指令执行测试向量中的每个测试。比较测试向量中的每个测试的输出,其中基于针对第一组机器指令和第二组指令的每个输出匹配来确定行为匹配。进一步,在所述测试向量中执行测试包括:创建用于所述第一组机器指令的第一包装器函数,根据要执行的测试用输入执行所述第一包装器函数,以及收集第一组输出。进一步,执行测试包括为第二组机器指令创建第二包装器函数,根据要执行的测试用所述输入来执行第二包装器函数,以及收集第二组输出。

[0022] 还可以至少由系统、计算机程序产品和机器提供上述特征。

[0023] 通过本发明的技术实现了附加特征和优点。本发明的其他实施例和方面在本文中详细描述,并且被认为是要求保护的发明的一部分。为了更好地理解本发明的优点和特征,参考说明书和附图。

附图说明

[0024] 在说明书结尾处的权利要求中特别指出并清楚地要求保护被视为本发明的主题。从下面结合附图的详细描述中，本发明的上述和其他特征以及优点显而易见，其中：

[0025] 图1描绘了根据本发明的一个或多个实施例的云计算环境；以及；

[0026] 图2描绘了根据本发明的一个或多个实施例的云计算机环境的抽象模型层；

[0027] 图3描绘了可用于实现本发明的一个或多个实施例的示例性计算机处理系统的框图；

[0028] 图4描绘了根据本发明的一个或多个实施例的优化系统的框图；

[0029] 图5描绘了根据本发明的一个或多个实施例的使用测试向量自动验证高级构造的优化的方法的流程图；以及

[0030] 图6描绘了根据本发明的一个或多个实施例的用于确定在第二可执行程序中包括哪个可执行代码的方法的流程图。

[0031] 此处所描绘的图是说明性的。在不背离本发明的范围的情况下，可以对这里所描述的图或操作进行许多变化。例如，可以以不同的顺序执行动作，或者可以添加、删除或修改动作。而且，术语“耦接”及其变形描述了在两个元件之间具有通信路径，并且不暗示元件之间的直接连接，在它们之间没有介入元件/连接。所有这些变型被认为是说明书的一部分。

[0032] 在附图和以下公开的实施例的详细说明中，附图中示出的各种元件设置有两个或三个数字参考标号。除了次要例外，每个参考标号的最左边的数字对应于首先示出了其元件的附图。

具体实施方式

[0033] 本文参考相关附图描述了本发明的各种实施例。在不脱离本发明的范围的情况下，可设计本发明的替代实施例。在以下描述和附图中，在元件之间阐述了各种连接和位置关系(例如，上方、下方、相邻等)。除非另有规定，否则这些连接和/或位置关系可以是直接或间接的，并且本发明不打算在这方面进行限制。因此，实体的连接可以指直接的或间接的连接，并且实体之间的位置关系可以是直接的或间接的位置关系。此外，本文描述的各种任务和过程步骤可以并入具有本文未详细描述的附加步骤或功能的更全面的程序或过程中。

[0034] 以下定义和缩写将用于解释权利要求书和说明书。如本文所使用的，术语“包含”、“包括”、“具有”、“拥有”、“含有”或“含”，或其任何其他变体旨在涵盖非排他包含。例如，包含一系列要素的组合物、混合物、工艺、方法、物品或设备不必仅限于那些要素，而是可以包括未明确列出的或这种组合物、混合物、工艺、方法、物品或设备固有的其他要素。

[0035] 此外，术语“示例性的”在本文中用于表示“用作实例、例子或例证”。本文中描述为“示例性”的任何实施例或设计不一定被解释为优于或优于其他实施例或设计。术语“至少一个”和“一个或多个”可以被理解为包括大于或等于一的任何整数，即，一个、两个、三个、四个等。术语“多个”可以被理解为包括大于或等于二的任何整数，即，两个、三个、四个、五个等。术语“连接”可以包括间接“连接”和直接“连接”两者。

[0036] 术语“约”、“基本上”、“大致”及其变型旨在包括与基于提交申请时可用的设备的特定量的测量相关联的误差程度。例如，“约”可以包括给定值的±8％或5％、或2％的范围。

[0037] 为了简洁起见，涉及制造和使用本发明的方面的常规技术可以或可以不在本文中详细描述。具体地，用于实现本文所描述的不同技术特征的计算机系统和特定计算机程序的不同方面是众所周知的。因而，为了简洁起见，许多常规实施细节在本文中仅简要提到或完全省略，而不提供众所周知的系统和/或过程细节。

[0038] 本发明的一个或多个实施例可以使用云计算来实现。尽管如此，提前理解的是，虽然本公开包括关于云计算的详细描述，但是本文所陈述的教导的实现不限于云计算环境。相反，本发明的实施例能够结合现在已知的或以后开发的任何其他类型的计算环境来实现。

[0039] 云计算是服务交付的模型，用于使得能够方便地、按需地网络访问可配置计算资源(例如，网络、网络带宽、服务器、处理、存储器、存储、应用、虚拟机和服务)的共享池，所述可配置计算资源可以以最小的管理努力或与服务的提供商的交互来快速供应和释放。该云模型可以包括至少五个特性、至少三个服务模型和至少四个部署模型。

[0040] 特性如下：

[0041] 按需自助服务：云消费者可单方面根据需要自动提供计算能力，诸如服务器时间和网络存储，而不需要与服务的提供商进行人为交互。

[0042] 广泛的网络接入：能力可通过网络获得并且通过标准机制接入，该标准机制促进异构瘦客户机平台或厚客户机平台(例如，移动电话、膝上型计算机和PDA)的使用。

[0043] 资源池：供应商的计算资源被池化以使用多租户模型服务于多个消费者，其中根据需求动态地分配和重新分配不同的物理和虚拟资源。存在位置独立性的感觉，因为消费者通常不具有对所提供的资源的确切位置的控制或了解，但可能能够以较高抽象级别(例如，国家、州或数据中心)指定位置。

[0044] 快速弹性：能够快速和弹性地提供能力，在一些情况下自动地快速缩小和快速释放以快速放大。对于消费者而言，可用于供应的能力通常显得不受限制并且可以在任何时间以任何数量购买。测量的服务：云系统通过在适合于服务类型(例如，存储、处理、带宽和活动用户账户)的某个抽象级别处利用计量能力来自动控制和优化资源使用。可以监视、控制和报告资源使用，为所利用的服务的提供者和消费者提供透明度。

[0045] 软件即服务(SaaS)：提供给消费者的能力是使用在云基础设施上运行的提供者的应用。可通过诸如web浏览器(例如，基于web的电子邮件)之类的瘦客户端接口从不同客户端设备访问应用。消费者不管理或控制包括网络、服务器、操作系统、存储或甚至单独的应用能力的底层云基础设施，可能的例外是有限的用户特定应用配置设置。

[0046] 平台即服务(PaaS)：提供给消费者的能力是将消费者创建的或获取的使用由提供商支持的编程语言和工具创建的应用部署到云基础设施上。消费者不管理或控制包括网络、服务器、操作系统或存储的底层云基础设施，但是对所部署的应用和可能的应用托管环境配置具有控制。

[0047] 基础设施即服务(IaaS)：提供给消费者的能力是提供处理、存储、网络和消费者能够部署和运行任意软件的其他基本计算资源，所述软件可以包括操作系统和应用。消费者不管理或控制底层云基础设施，而是具有对操作系统、存储、所部署的应用的控制以及对所选联网组件(例如，主机防火墙)的可能受限的控制。

[0048] 部署模型如下：

[0049] 私有云:云基础架构仅为组织运作。它可以由组织或第三方管理,并且可以存在于场所内或场所外。

[0050] 社区云:云基础架构被若干组织共享并支持共享了关注(例如,任务、安全要求、策略、和合规性考虑)的特定社区。它可以由组织或第三方管理,并且可以存在于场所内或场所外。

[0051] 公共云:使云基础架构对公众或大型行业组可用,并且由出售云服务的组织拥有。

[0052] 混合云:云基础架构是两个或更多个云(私有、社区或公共)的组合,这些云保持唯一实体但通过使数据和应用能够移植的标准化或专有技术(例如,云突发以用于云之间的负载平衡)绑定在一起。

[0053] 云计算环境是面向服务的,集中于无状态、低耦合、模块化和语义互操作性。云计算的核心是包括互连节点网络的基础设施。

[0054] 现在参见图1,描述了说明性云计算环境50。如图所示,云计算环境50包括云消费者使用的本地计算设备可以与其通信的一个或多个云计算节点10,本地计算设备诸如例如个人数字助理(PDA)或蜂窝电话54A、台式计算机54B、膝上型计算机54C和/或一些其他计算机系统或设备。节点10可彼此通信。它们可以物理地或虚拟地分组(未示出)在一个或多个网络中,诸如如上所述的私有云、社区云、公共云或混合云、或其组合。这允许云计算环境50提供基础设施、平台和/或软件作为云消费者不需要为其维护本地计算设备上的资源的服务。应当理解,图1中所示的计算装置54A-54C的类型仅旨在是说明性的,并且计算节点10和云计算环境50可以通过任何类型的网络和/或网络可寻址连接(例如,使用网络浏览器)与任何类型的计算机化装置通信。

[0055] 现在参考图2,示出了由云计算环境50提供的一组功能抽象层。应提前理解,图2中所示的组件、层和功能仅旨在是说明性的,并且本发明的实施例不限于此。如所描述的,提供以下层和相应的功能:硬件和软件层60包括硬件和软件组件。硬件组件的示例包括:大型机61、基于RISC(精简指令集计算机)架构的服务器62、服务器63、刀片服务器64、存储设备65、以及网络和网络组件66。在一些实施例中,软件组件包括网络应用服务器软件67和数据库软件68。虚拟化层70提供抽象层,从该抽象层可以提供虚拟实体的以下示例:虚拟服务器71、虚拟存储器72、虚拟网络73(包括虚拟专用网络)、虚拟应用和操作系统74、以及虚拟客户端75。

[0056] 在一个示例中,管理层80可以提供以下描述的功能。资源供应81提供用于在云计算环境内执行任务的计算资源和其他资源的动态采购。计量和定价82在云计算环境内利用资源时提供成本跟踪,并为这些资源的消费开账单或发票。在一个示例中,这些资源可以包括应用软件许可证。安全性为云消费者和任务提供身份验证,以及为数据和其他资源提供保护。用户门户83为消费者和系统管理员提供对云计算环境的访问。服务水平管理84提供云计算资源分配和管理,使得满足所需的服务水平。服务水平协议(SLA)规划和履行85提供根据SLA预期未来需求的云计算资源的预安排和采购。

[0057] 工作负载层90提供可以利用云计算环境的功能的示例。可以从该层提供的工作负载和功能的示例包括:地图和导航91、软件开发和生命周期管理92、虚拟教室教育交付93、数据分析处理94、事务处理95和学习模型处理96,用于执行高级编程指令的编译。

[0058] 参考图3,示出了数据处理系统(通常被称为计算机系统100)的实施例,数据处理

系统通过通信网络与云计算环境50的一个或多个节点10通信以实现本文的教导。计算机系统100具有一个或多个中央处理单元(处理器)121a、121b、121c等(统称为或一般地称为(一个或多个)处理器121)。在一个或多个实施例中,每个处理器121可以包括精简指令集计算机(RISC)微处理器。处理器121经由系统总线133耦合到系统存储器(RAM)134和不同其他组件。只读存储器(ROM)122耦合到系统总线133,并且可以包括控制计算机系统100的某些基本功能的基本输入/输出系统(BIOS)。

[0059] 图3进一步示出了耦接至系统总线133的输入/输出(I/O)适配器127和网络适配器126。I/O适配器127可以是与硬盘123和/或磁带存储驱动器125或任何其他类似组件通信的小型计算机系统接口(SCSI)适配器。I/O适配器127、硬盘123和磁带存储设备125在本文中统称为大容量存储设备124。

[0060] 可在大容量存储器124中存储用于在处理系统100上执行的操作系统140。然而,操作系统140还可以存储在计算机系统100的RAM 134中。根据本发明实施例的操作系统包括例如UNIX™、Linux™、MicrosoftXP™、AIX™和IBM的i5/OS™。

[0061] 网络适配器126将总线133与外部网络136互连,使得计算机系统100能够与其他这样的系统通信。屏幕(例如,显示监视器)135通过显示适配器132连接到系统总线133,显示适配器132可以包括用于改进图形密集型应用的性能的图形适配器和视频控制器。在一个实施例中,适配器127、126和132可以连接到经由中间总线桥(未示出)连接到系统总线133的一个或多个I/O总线。用于连接诸如硬盘控制器、网络适配器和图形适配器之类的外围设备的合适的I/O总线通常包括诸如外围组件互连(PCI)之类的公共协议。附加的输入/输出设备被示为经由用户界面适配器128和显示适配器132连接到系统总线133。键盘129、鼠标130和扬声器131都经由用户接口适配器128互连到总线133,用户接口适配器128可以包括例如将多个设备适配器集成到单个集成电路中的超级I/O芯片。

[0062] 在示例性实施例中,计算机系统100包括图形处理单元141。图形处理单元141是被设计成操纵和改变存储器以加速意图输出到显示器的帧缓冲器中的图像的创建的专用电子电路。一般而言,图形处理单元141在操纵计算机图形和图像处理时是非常高效的,并且具有高度并行的结构,该结构使得其比通用CPU更有效地用于其中并行完成大数据块的处理的算法。

[0063] 因此,如图3中配置的,计算机系统100包括处理器121形式的处理能力、包括RAM134和大容量存储器124的存储能力、诸如键盘129和鼠标130的输入装置、以及包括扬声器131和显示器135的输出能力。在一个实施例中,RAM 134和大容量存储器124的一部分共同存储操作系统以协调图3中所示的不同组件的功能。

[0064] 如本文中所描述,在静态编译程序的情况下,在执行任何生成的可执行代码之前编译计算机程序。利用静态编译的语言,技术问题在于,在编译计算机程序时,编译器程序不具有仅能在运行时(即,当执行计算机程序时)收集的任何信息,并且哪些信息可能对计算机程序的效率具有实质影响。附加的技术问题是编译器程序可能不知道将在其上执行生成的可执行代码的特定数据处理系统,并且因此不能将可执行代码适配于其将在其上运行的数据处理系统的硬件特征。为了解决这样的技术问题,并且为了使可执行代码高效,当要被用于运行可执行代码的数据处理系统被改变时,利用重新编译器来重新编译计算机程序。

[0065] 这种方法的技术问题是用于重新编译计算机程序的编译器也可能已经改变以利用现在正在使用的数据处理系统中的硬件和其他变化。进一步,正被使用的编程语言(诸如COBOL语言)可具有未定义行为的实例。例如,特别是对于诸如COBOL的语言,在编译器的一个版本中使用的数据类型可能具有不符合另一版本中的相同数据类型的规范的不良形成的数据。进一步,问题在于,由于较老的编译器版本的特定低级代码生成选择,由于这个不良形成的数据而导致的任何可观察到的问题可被无意地隐藏/掩蔽。当新的编译器版本生成假定遵循规范的代码时,则由于不良形成的数据而导致的这些问题就会出现。

[0066] 应注意,对于使用二进制值以使得每一位模式对应于值的语言,这可能不是问题。例如,COBOL中使用的常见数据类型包括“紧缩十进制”和“分区十进制”类型。这些二进制编码十进制(BCD)类型具有一系列无效编码。无效编码的特定程序行为不是由语言明确定义的,它是特定无效编码和对该无效数据进行操作的编译器生成的精确机器指令序列的产物。在本描述中,将参考COBOL语言描述本发明的实施例,然而,本发明的实施例也适用于其他编程语言。

[0067] 在COBOL的情况下,由于数据类型的这种未定义行为,用较新的编译器重新编译程序比大多数语言风险更大。未定义行为的结果是不可预测的,并且可以从编译器到编译器或者在同一编译器内的编译器优化级别之间变化。重新编译包含未定义行为的程序可以产生可执行代码,该可执行代码导致与由较老的编译器产生的旧的可执行代码不同的执行(或者甚至是相同的编译器,但当在不同的环境中运行时,或者当使用诸如编译器优化级别的不同编译器选项、或任何其他编译器选项时)。作为诸如COBOL的语言中的可能不良形成数据的典型实例,考虑其中区位十进制数据项中的每个字节的高4位未被适当地初始化的源代码。根据语言规范,这四个位应该是0xF,并且对于带符号项,最低有效字节中的四个位应该是有效符号代码0xA->0xF。另一可能性包括在打包或区域的十进制数据项中不适当地设置符号代码。另一潜在问题产生于COBOL的二进制二的补码数据项被声明为指定数量的十进制数字。因此,有意或无意地,可以通过包括更多的二进制位来‘过度填充’二进制项,然后实际上可以通过所声明的十进制位数来表示。另一示例是不清除(或不经意地设置)偶数打包十进制数据项中的最高有效字节的高4位。甚至精确打包的十进制项仅需要最高有效字节的一半来表示其所有数位,并且因此该字节的高4位应根据语言规范被设置为零。各种其他示例是可能的。

[0068] 为了解决这样的技术问题,使用在此被称为二进制优化器的二进制到二进制优化工具,其还可以被称为二进制转换器。二进制优化器将编译的可执行代码中的旧二进制模块(具有旧版本/较旧优化级别)转换成另一种形式,并且使用不同的(例如,较新的)编译器技术从该形式优化二进制模块。应注意,在这种情况下,“转换”不指将二进制代码转换回源代码。相反,二进制转换器或二进制优化器解构并分析原始二进制代码(来自较旧版本和/或较低优化级别),以便将其转换成程序的中间语言(IL)表示,从而优化并然后使用新的优化和编译技术来生成新的二进制模块。因为二进制优化器使用编译的二进制可执行代码作为输入,所以假设即使在未定义行为的情况下,优化版本也具有相同行为。然而,此方法限于较旧版本的编译器。编译器的较新版本使用更积极的优化,使得过于难以分析二进制。

[0069] 例如,考虑生成高级构造的专门内联扩展。“高级构造”是高级语言或机器语言(诸如IBM ®z/Architecture®指令集)的语句或指令,其足够复杂使得其可执行表示将是多个低

级机器指令的复杂代码序列。高级构造的示例是:COBOL语言的INSPECT,以及IBMz/Architecture指令集中的ED、EDMK。在若干情况下,COBOL编译器的较新版本生成专用代码序列,以表示高级构造,而不是用运行时库调用或较慢的指令序列以更一般但较不高效的方式进行处理。例如,在COBOL中,“INSPECT”语句可搜索和变换字符串。在COBOL的较早版本中,通常通过调用COBOL运行时例程来执行非微不足道的INSPECT语句。在较新的版本中,在二进制可执行代码本身中内联地生成更快的实现。与必须处理INSPECT的每个可能版本的运行时例程不同,内联副本可以专用于必须执行的特定版本。类似地,与先前使用的通用硬件指令ED和EDMK相比,使用专用代码序列可以更快地进行从数字编辑到数字数据类型的转换。ED和EDMK指令采用图案作为输入,并且必须处理任何可能的图案。生成专用于将在可执行代码中使用的特定模式的序列是更快的。IBM和z/Architecture是在全球许多司法管辖区中注册的国际商业机器公司的商标。

[0070] 在由较新的、更高效的编译器进行编译期间,对应于高级构造的这样的操作通常在编译器优化程序时在高级处表示,并且仅在代码生成期间被转换成低级专门序列。这允许编译器优化利用关于高级构造的语义的信息。

[0071] 这些高级构造产生的编码序列可以显著变化,这取决于高级构造的细节。仅基于编译的二进制来标识这样的代码序列的范围和目的是过于困难的技术问题。现有技术的二元优化器在恢复相应的IL表示的能力方面受到限制。这样的二进制优化器能够转换用较旧版本的编译器编译的代码,其中第一可执行代码可以容易地转换成IL表示,并且其中编译器不处于活跃开发中。在这种情况下,二元优化器可优化高级构造,因为高级构造的IL表示易于确定且不会改变。但是这对于正在开发的任何编译器并不成立,并且因此使用新的编译器优化技术。这可导致生成更多或不同类型的这些序列(甚至对于现有的高级构造,如INSPECT或数字编辑)。

[0072] 根据本发明的一个或多个实施例,本发明的一个或多个实施例解决了这样的技术问题并且促进由较新版本的编译器编译的代码被二进制优化器优化。它允许二进制优化器生成新的、更有效的代码序列。其在每次修改编译器以生成新代码序列时都不要求二进制优化器的改变的情况下这样做,并且促进二进制优化器保存现有行为。

[0073] 图4示出了根据本发明的一个或多个实施例的优化系统的框图。如本文所使用的,“编译计算机程序”包括从采用源语言的计算机程序生成二进制可执行文件,即可由处理器执行的机器指令。而且,如在此使用的,“优化可执行代码”是指通过产生新的二进制可执行文件来从二进制可执行文件中优化计算机程序,该新的二进制可执行文件包括将第一二进制可执行文件转换成IL表示。应注意,如本文所使用的IL可为可表示程序的任何语言。例如,IL可以是内部语言,它可以是计算机程序的源语言,它可以是具有内部注释的源语言,它可以是另一二进制语言,或任何其他类型的表示。所有这些都是在第一可执行代码和第二可执行代码之间的中间的示例,第二可执行代码是第一可执行代码的优化。

[0074] 在所描绘的场景中,计算机程序410由编译器420编译以生成第一二进制可执行代码430。计算机程序410可以包括一个或多个高级构造412。应当理解,尽管仅示出了一个构造412,但是程序410可以包括一个或多个构造412。

[0075] 当代码要被优化时,例如,对于新硬件,使用二进制优化器440。二进制优化器440可以执行可执行代码430的转换,并且进一步执行优化和最终代码生成以生成第二可执行

代码470。应当注意，在一个或多个示例中，二进制优化器440可以用于其他可能性，诸如用于产生第一可执行代码430的IL(或甚至源)表示，该表示被传递到第二编译器(未示出)以用于最终代码生成(或附加的较低级别优化)。例如，第二编译器是比编译器420更新的编译器。可替代地，第二编译器是使用不同设置(诸如正在使用的优化级别)的编译器420的实例。

[0076] 二进制优化器440接收第一可执行代码430作为输入，并转换可执行代码430以输出计算机程序的IL表示450。二进制优化器440随后优化计算机程序的IL表示450以生成第二二进制可执行代码470。即，编译器420产生可包括将进一步描述的附加编译时间信息的第一可执行代码430，并且二进制优化器440消耗第一可执行代码430和编译时间信息。由编译器420存储的任何编译时间信息是用于优化以产生经优化的第二可执行代码470的目的。

[0077] 如前所述，利用编译器420的不同版本，特别是较新的编译器技术，分析第一二进制可执行代码430在技术上是有挑战性的，因为运行时命令可将这样的分析作为移动目标。现有技术的二进制优化器不能将二进制可执行代码430转换成IL表示450，并且因此不能从编译器420通常生成的其可执行表示转换高级构造412。

[0078] 为了解决该技术问题并使得可以在用这样的(例如，较新的)编译器编译的程序上使用二进制优化器440，如之前提到的，本发明的一个或多个实施例促进在编译的二进制可执行代码430中包括附加的编译时间信息。该添加的信息可用于传送关于程序410的含义的信息。

[0079] 在本发明的一个或多个实施例中，编译器420在第一可执行代码430中包括关于程序410中的每个高级构造412的编译时间信息，以允许二进制优化器440将该构造412转换成包括在程序的IL表示450中的该结构的IL表示，优化高级构造的IL表示，以及为该高级构造生成潜在地包括在第二可执行代码470中的新指令序列。

[0080] 因为优化，所以新优化指令可能在一些输入上不具有与旧指令相同的行为。为了解决这个问题，本发明的一个或多个实施例使用测试向量：一组自动生成的输入来比较针对高级构造生成的新和旧指令序列的行为。如果两个序列针对测试向量中的每个输入给出相同的输出，则认为在第二可执行代码470中包括用于高级构造的新指令是安全的。进一步，因为在优化期间将新指令序列的行为与旧序列进行比较，所以相对于高级构造，经优化代码(470)保存bug-for-bug兼容性。

[0081] 图5描绘了根据本发明的一个或多个实施例的使用测试向量来自动验证高级构造的优化的方法的流程图。该方法包括在510使用编译器420编译程序410。编译包括在512发出关于程序410中的高级构造412的每个内联扩展的编译时间信息。该信息包括被扩展的高级构造412的标识符。例如，“编辑并标记”或“检查清点”。该信息还包括由第一可执行代码430中的内联扩展生成的该组指令的位置。由于像由编译器420执行的指令调度那样的优化，这些指令可能不形成连续范围。

[0082] 对于到高级构造412的每一输入，编译时间信息还包含关于发现该输入的位置的信息。例如，输入到编辑并标记操作中的值可以在寄存器中或存储器中。对于高级构造的每个输出，信息还包括关于该输出被放置在哪里的信息，诸如在寄存器中或存储器中。输入/输出读取/写入的存储器和/或寄存器分别使用存储器地址或寄存器地址进行标识。该信息存储在可执行二进制代码430中。

[0083] 在520，使用编译器420来生成第一可执行代码430，并且该第一可执行代码430可以用于执行，直到决定使用二进制优化器440来优化该代码为止。例如，一旦较新的二进制优化器或较新的硬件可用，就决定对代码进行优化，但是，可以出于若干其他原因执行优化。在530，二进制优化器440将第一可执行代码430转换成程序的IL表示450。在转换期间，在532，使用存储在可执行二进制代码430中的发出的编译时间信息，二进制优化器440生成其IL表示中的高级构造412。

[0084] 针对高级构造412生成的机器指令未在IL表示450中明确表示。二进制优化器440的默认行为通常将每个原始输入机器指令表示为待优化的IL表示450中的对应结构。然而，对于本文讨论的更高级构造(如检查和数字编辑)，要防止生成高级构造的IL表示450的方式。以这种方式表示可以使得二进制优化器440很难或不可能确定在这些输入指令的范围内进行的总体较高水平操作。因此，当在二进制430中存在关于跨输入指令的范围的这样的较高级操作的信息时，二进制优化器440选择类似的高级IL表示来封装正在进行的较高级操作。

[0085] 进一步，对于程序410的优化，在540，二进制优化器440被用于从高级构造412的IL表示450生成用于高级构造412的新优化指令。新指令将与由编译器420生成的指令进行比较。

[0086] 此时，对于每个高级构造412，基于第一二进制可执行代码430中的编译时间信息并且基于来自完成步骤540的信息，可获得以下信息。编译时间信息包括高级构造412的身份、编译器420为高级构造412生成的旧指令的位置以及每个编译器420的旧指令的输入和输出的位置；来自完成步骤540的信息包括二进制优化器440为高级构造412生成的新指令和每个二进制优化器440的新指令的输入和输出的位置。

[0087] 该方法进一步包括在550使用编译时间信息来为每个高级构造412生成两个包装器函数，比如F_new和F_old。应当理解的是，“F_new”和“F_old”是示例名称，并且在其他示例中可以使用其他名称/身份。创建包装器函数包括在552处配置输入和输出参数以及作为包装器函数的一部分的指令。函数的输入是高级构造412的输入。这些函数的返回值是高级构造412的输出。在内部，每个函数被实现为F_old的旧指令和F_new的新指令的副本，旧指令分别是每个编译器420的那些指令，并且新指令分别是每个二进制优化器440的那些指令。

[0088] 此外，在560，对于每种类型的高级构造412，定义过程以创建测试向量。测试向量是比较由编译器420和二进制优化器440生成的代码序列的行为的测试列表。测试向量包括对高级构造的有效和无效输入。如果二进制优化器440提供(通过查看之前的操作)无效输入不能到达该点，则测试向量可以被可选地约束为省略某些无效输入。测试向量可包括基于高级构造512的类型的预定测试集合，其中高级构造412的类型基于来自编译时间信息的身份来确定。

[0089] 在570，对于测试向量中的每个测试，执行F_new和F_old包装器函数，并收集相应的输出。在580，比较来自两个包装器函数的输出，以确定要在第二可执行代码470中包括用于该高级构造412的哪些指令。根据本发明的一个或多个实施例，F_new和F_old的执行被包裹在异常处理程序中以用于收集输出。对包装器函数的无效输入可导致硬件异常。这是可接受的，只要旧指令和新指令引起相同的硬件异常。测试向量用于确保旧指令和新指令的

行为基本相同。

[0090] 基于针对高级构造412的测试的输出的比较，确定要在第二可执行代码470中包括用于该高级构造412的哪些指令。进一步，在该方法中，如果输出不匹配，则即使来自测试向量的单个输出不匹配，在580,确定要包括在第二二进制可执行代码470中的用于高级构造412的指令。如果新指令和旧指令不提供相同的输出，则存在若干选项。从这些选项中选择这些选项中的哪一个是基于如进一步描述的若干因素。一旦基于该选择的每个高级构造的指令在行为与第一可执行程序430中的高级构造的对应指令匹配，二进制优化器440就在590生成并发出包括用于每个所选高级构造的对应指令的第二可执行程序470。

[0091] 图6示出了根据本发明的一个或多个实施例的用于确定用于高级构造412的哪些指令以包括在第二可执行程序470中的方法的流程图。如果在610，F_new和F_old针对测试向量中的每个输入给出相同的输出，则在620，新指令被包括在第二二进制可执行程序470中。

[0092] 在本发明的一个或多个实施例中，如果用于高级构造412的新指令不匹配旧指令的行为，则在630，可提示用户选择哪些指令以包括在第二可执行程序470中。根据本发明的一个或多个实施例，在运行优化器440之前为优化器440设置用于选择哪些指令包括在第二可执行程序470中的选项(如果出现这种情况的话)。替换地或附加地，在本发明的一个或多个实施例中，在630，如果用于高级构造412的、测试结果不匹配于来自编译器420的原始指令的新指令被发送到第二可执行代码中。这可以是在630处执行的选择的默认行为。在一个或多个示例中，再次执行来自测试向量的测试以确保行为匹配。

[0093] 替换地或附加地，基于来自新指令的性能增益来促进选择。例如，如果来自高级构造412的新代码序列的预期性能增益与旧代码序列的预期性能增益相比足够大，则用户可选择更积极的选项。这里，可以基于执行时间、计算机资源使用或任何其他度量来确定足够大的性能增益。如果对旧代码和新代码的此类度量的比较导致至少预定阈值改善，则可认为新代码具有足够大的性能增益。例如，就执行时间而言，如果执行时间提高至少10％、25％、50％或任何其他此类比率(或其他类型的阈值)，则新代码可被视为具有足够大的性能增益，并且新代码(甚至具有不同行为)可由用户选择。

[0094] 替换地或附加地，如果新代码序列仅给出(某些类型的)无效数据的不同结果，并且用于检测(那些类型的)无效数据的运行时测试的成本小于来自新代码序列的性能改进，则二进制优化器440可以生成新的(第三)二进制可执行代码，该新的(第三)二进制可执行代码包括由运行时测试所保护的旧的和新的代码序列两者。如果测试通过，则使用新的代码序列。如果测试失败，则使用旧的代码序列。这提供了整体性能改进，同时仍然保证相同的结果。

[0095] 进一步，在本发明的一个或多个实施例中，代替使用测试向量，使用布尔可满足性(SAT)求解器或其他这样的工具来确定用于高级构造412的新代码和用于该高级构造412的旧代码的匹配。因为匹配测试作为优化的一部分由二进制优化器440执行，并且因为SAT求解器可能需要大量时间，所以与如上所述的测试向量的使用相比，使用二进制优化器440的优化可被认为是太长。

[0096] 更进一步，在一个或多个示例中，如果编译时间信息由于某种原因不存在于第一可执行代码430中，则另一工具(诸如将第一可执行代码430转换回源的真实反编译器(未示

出))或二进制优化器440可分析并发现较高级操作,并随后继续本文描述的测试向量方法。

[0097] 因而,本发明的一个或多个实施例提供实际应用以改进诸如COBOL的语言的静态编译代码的优化。因此,本发明的一个或多个实施例提供了对计算技术(特别是编译器和优化器)的改进。本发明的一个或多个实施例有助于使用测试向量验证高级构造的优化。

[0098] 例如,验证包括使用启用智能二进制的编译器来编译程序。“启用智能二进制的”编译器发出并存储关于每个内联扩展的信息,包括在所生成的二进制代码中扩展的高级构造。所存储的信息包括由内联扩展生成的一组指令的标识。所述信息还包括:对于至高级构造的每个输入,关于待输入的一个或多个输入参数位于何处的信息。例如,输入参数的位置可以被提供为存储器中的位置地址、文件中的位置或任何其他这样的引用。所述信息还包括:对于所述高级构造的每个输出,关于所述输出被放置在哪里的信息。例如,输出的位置可被提供为存储器中的位置地址、文件中的位置或存储一个或多个输出参数的任何其他这样的引用。

[0099] 通过优化器(诸如二进制优化器)优化包括所生成的代码(旧指令)的程序包括将所生成的(二进制)代码转换成IL表示。所生成的代码可包括用于在程序的源代码中使用的高级构造的一个或多个指令。进一步,响应于程序被优化,生成另一二进制可执行代码(新指令),其可包括用于高级构造的经修改的指令集。在一个或多个示例中,编译时间信息包括高级构造的身份、旧指令的位置、旧指令的输入和输出的位置。

[0100] 验证还包括由计算机为每个高级构造创建F_new和F_old的包装器函数,其中,对这些函数的输入是对高级构造的输入,并且这些函数的返回值是高级构造的输出。在本发明的一个或多个实施例中,生成过程以创建包括被设计为比较旧指令和新指令的行为的测试列表的测试向量,其中测试向量包括有效和无效输入。F_new和F_old函数中的每个函数针对测试向量中的每个测试执行以收集相应的输出,其中测试作为程序优化的一部分发生。响应于由计算机确定F_new和F_old函数针对测试向量中的每个输入给出相同的输出,提供在优化程序中使用新指令是安全的(即,匹配旧行为)的指示。响应于由计算机确定F_new和F_old函数不对测试向量中的每个输入给出相同的输出,计算机可以促进从一组预定选项中进行选择。

[0101] 本发明可以是系统、方法和/或计算机程序产品。计算机程序产品可包括其上具有用于使处理器执行本发明的各方面的计算机可读程序指令的计算机可读存储介质(或多个介质)。

[0102] 计算机可读存储介质可为可保留和存储供指令执行装置使用的指令的有形装置。计算机可读存储介质可以是,例如但不限于,电子存储设备、磁存储设备、光存储设备、电磁存储设备、半导体存储设备、或者上述的任意合适的组合。计算机可读存储介质的更具体示例的非穷尽列表包括以下各项:便携式计算机盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦式可编程只读存储器(EPROM或闪存)、静态随机存取存储器(SRAM)、便携式紧凑盘只读存储器(CD-ROM)、数字通用盘(DVD)、记忆棒、软盘、诸如穿孔卡之类的机械编码设备或具有记录在其上的指令的槽中的凸出结构、以及上述各项的任何合适的组合。如本文所使用的计算机可读存储介质不应被解释为暂时性信号本身,例如无线电波或其他自由传播的电磁波、通过波导或其他传输介质传播的电磁波(例如,穿过光纤电缆的光脉冲)或通过电线发射的电信号。

[0103] 本文中所描述的计算机可读程序指令可以经由网络(例如,互联网、局域网、广域网和/或无线网络)从计算机可读存储介质下载到相应的计算/处理设备,或者下载到外部计算机或外部存储设备。网络可以包括铜传输电缆、光传输纤维、无线传输、路由器、防火墙、交换机、网关计算机和/或边缘服务器。每个计算/处理设备中的网络适配器卡或网络接口接收来自网络的计算机可读程序指令,并转发计算机可读程序指令以存储在相应计算/处理设备内的计算机可读存储介质中。

[0104] 用于执行本发明的操作的计算机可读程序指令可以是汇编指令、指令集架构(ISA)指令、机器指令、机器相关指令、微代码、固件指令、状态设置数据、或以一种或多种程序设计语言的任何组合编写的源代码或目标代码,这些程序设计语言包括面向对象的程序设计语言(如Smaltalk、C++等)和常规的过程式程序设计语言(如“C”程序设计语言或类似的程序设计语言)。计算机可读程序指令可以完全地在用户计算机上执行、部分在用户计算机上执行、作为独立软件包执行、部分在用户计算机上部分在远程计算机上执行或者完全在远程计算机或服务器上执行。在后一种情况下,远程计算机可通过任何类型的网络(包括局域网(LAN)或广域网(WAN))连接至用户计算机,或者可连接至外部计算机(例如,使用互联网服务提供商通过互联网)。在一些实施例中,包括例如可编程逻辑电路、现场可编程门阵列(FPGA)或可编程逻辑阵列(PLA)的电子电路可以通过利用计算机可读程序指令的状态信息来使电子电路个性化来执行计算机可读程序指令,以便执行本发明的各方面。

[0105] 下面将参照根据本发明实施例的方法、装置(系统)和计算机程序产品的流程图和/或框图描述本发明。应当理解,流程图和/或框图的每个方框以及流程图和/或框图中各方框的组合,都可以由计算机可读程序指令实现。

[0106] 这些计算机可读程序指令可被提供给通用计算机、专用计算机或其他可编程数据处理装置的处理器以产生机器,使得经由计算机或其他可编程数据处理装置的处理器执行的指令创建用于实现在流程图和/或框图的或多个框中指定的功能/动作的装置。也可以把这些计算机可读程序指令存储在计算机可读存储介质中,这些指令使得计算机、可编程数据处理装置、和/或其他设备以特定方式工作,从而,其中存储有指令的计算机可读存储介质包括包含实现流程图和/或框图中的或多个方框中规定的功能/动作的方面的指令的制造品。

[0107] 也可以把计算机可读程序指令加载到计算机、其他可编程数据处理装置、或其他设备上,使得在计算机、其他可编程装置或其他设备上执行一系列操作步骤,以产生计算机实现的处理,使得在计算机、其他可编程装置或其他设备上执行的指令实现流程图和/或框图中的或多个方框中规定的功能/动作。

[0108] 附图中的流程图和框图示出了根据本发明的不同实施例的系统、方法和计算机程序产品的可能实现方式的架构、功能和操作。对此,流程图或框图中的每个框可表示指令的模块、段或部分,其包括用于实现指定的逻辑功能的一个或多个可执行指令。在一些备选实现中,框中标注的功能可以不按照图中标注的顺序发生。例如,取决于所涉及的功能,连续示出的两个块实际上可以基本上同时执行,或者这些块有时可以以相反的顺序执行。也要注意的是,框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合,可以用执行规定的功能或动作或执行专用硬件与计算机指令的组合的专用的基于硬件的系统来实现。

[0109] 已经出于说明的目的呈现了本发明的各种实施例的描述，但并不旨在是详尽的或者限于所公开的实施例。在不脱离所描述的实施例的范围和精神的情况下，许多修改和变化对本领域普通技术人员将是显而易见的。这里使用的术语被选择来最好地解释实施例的原理、实际应用或对在市场中找到的技术的技术改进，或者使得本领域普通技术人员能够理解这里公开的实施例。

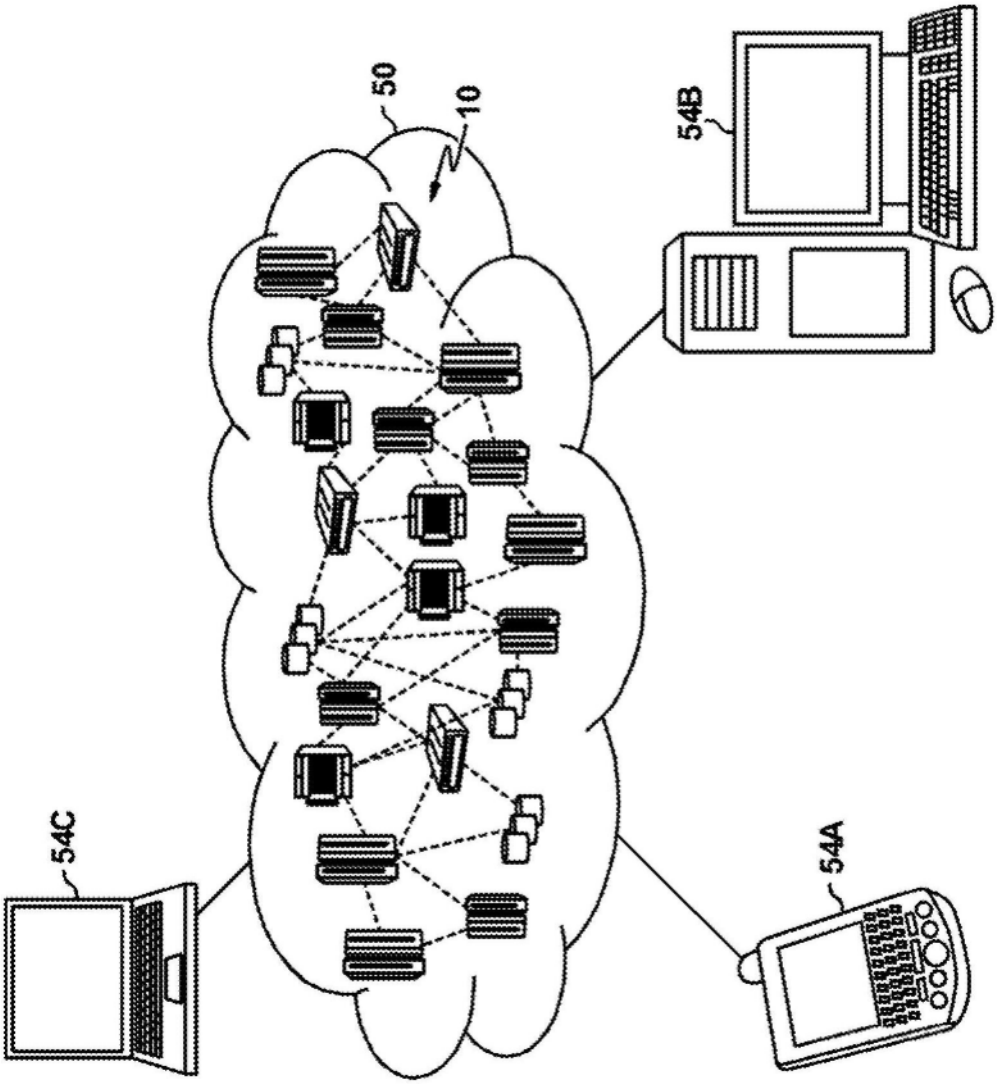


图1

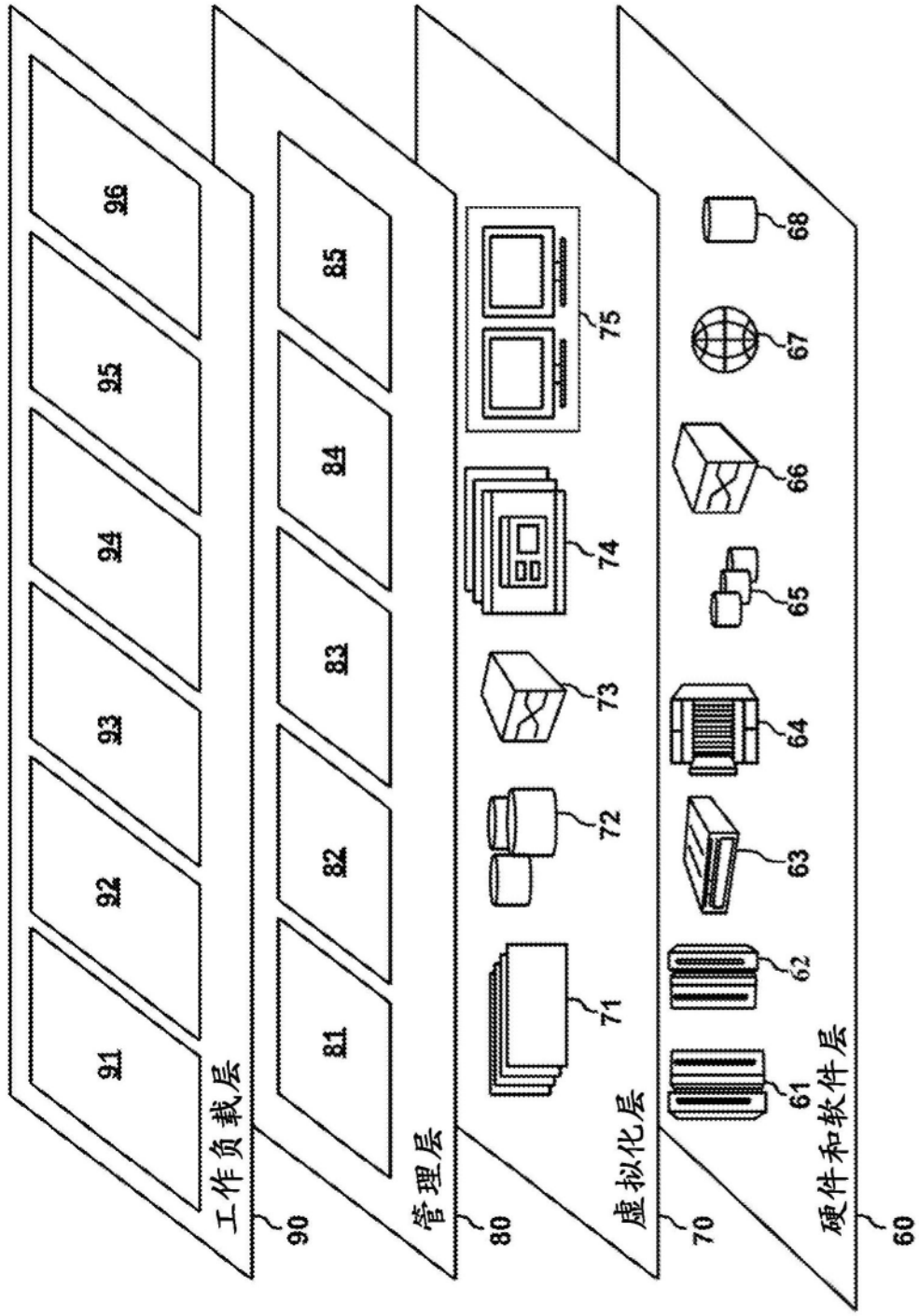

工作负载层
90
91
92
93
94
95
96
管理层
80
81
82
83
84
85
虚拟化层
70
71
72
73
74
75
硬件和软件层
60
61
62
63
64
65
66
67
68

图2

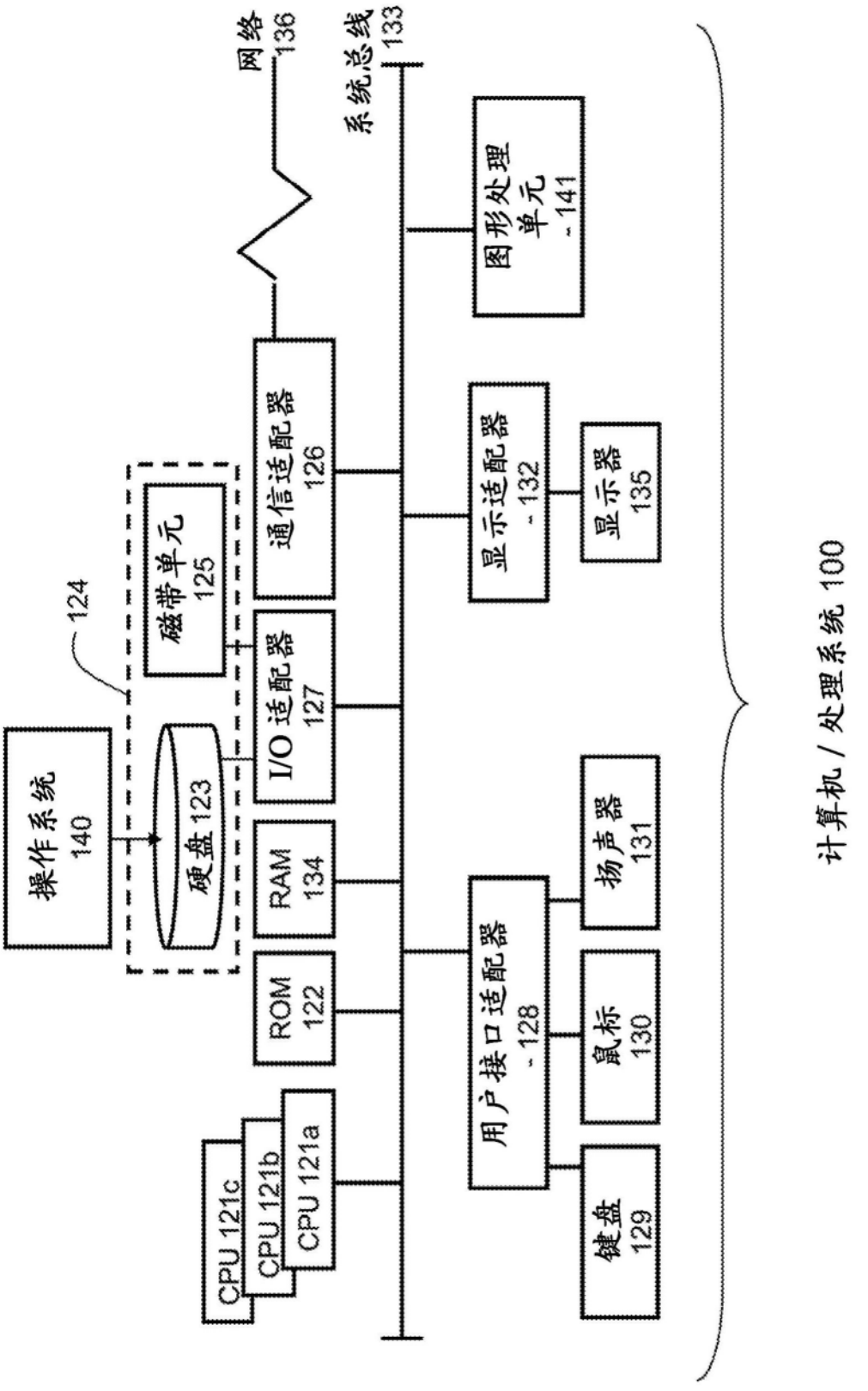

网络
136
系统总线
133
124
操作系统
140
硬盘 123
磁带单元
125
通信适配器
126
I/O 适配器
127
RAM
134
ROM
122
CPU 121c
CPU 121b
CPU 121a
图形处理
单元
141
显示适配器
132
显示器
135
用户接口适配器
128
扬声器
131
鼠标
130
键盘
129
计算机 / 处理系统 100

图3

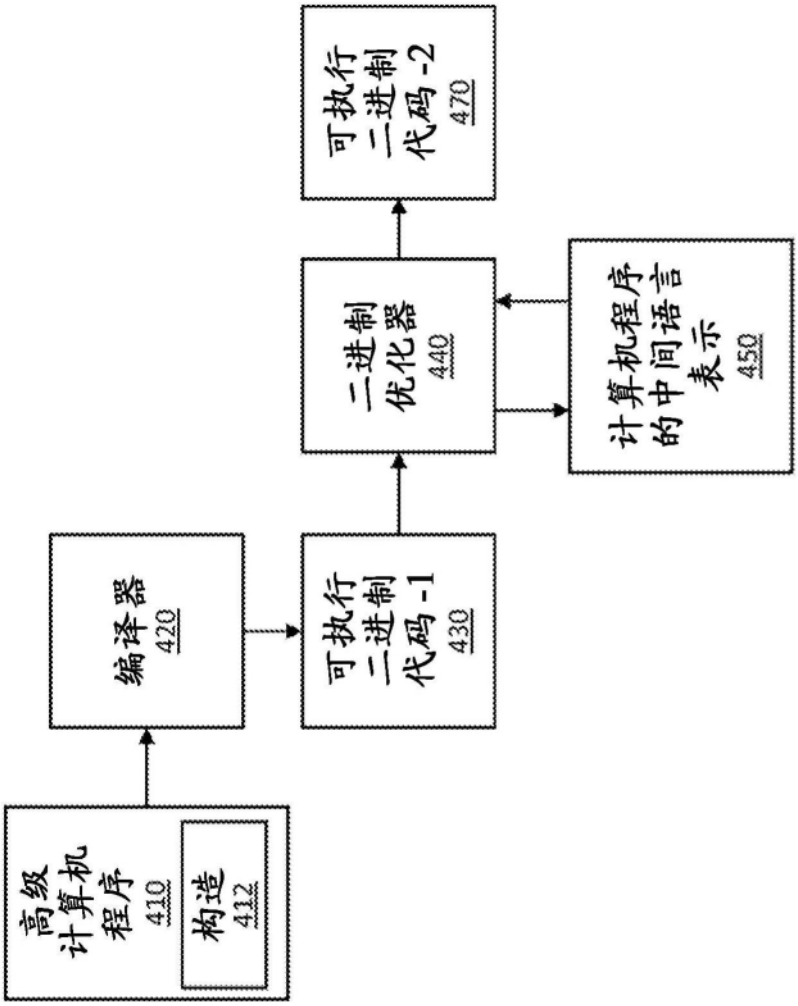

高级计算机程序
410
构造
412
编译器
420
可执行二进制代码-1
430
二进制优化器
440
计算机程序的中间语言表示
450
可执行二进制代码-2
470

图4

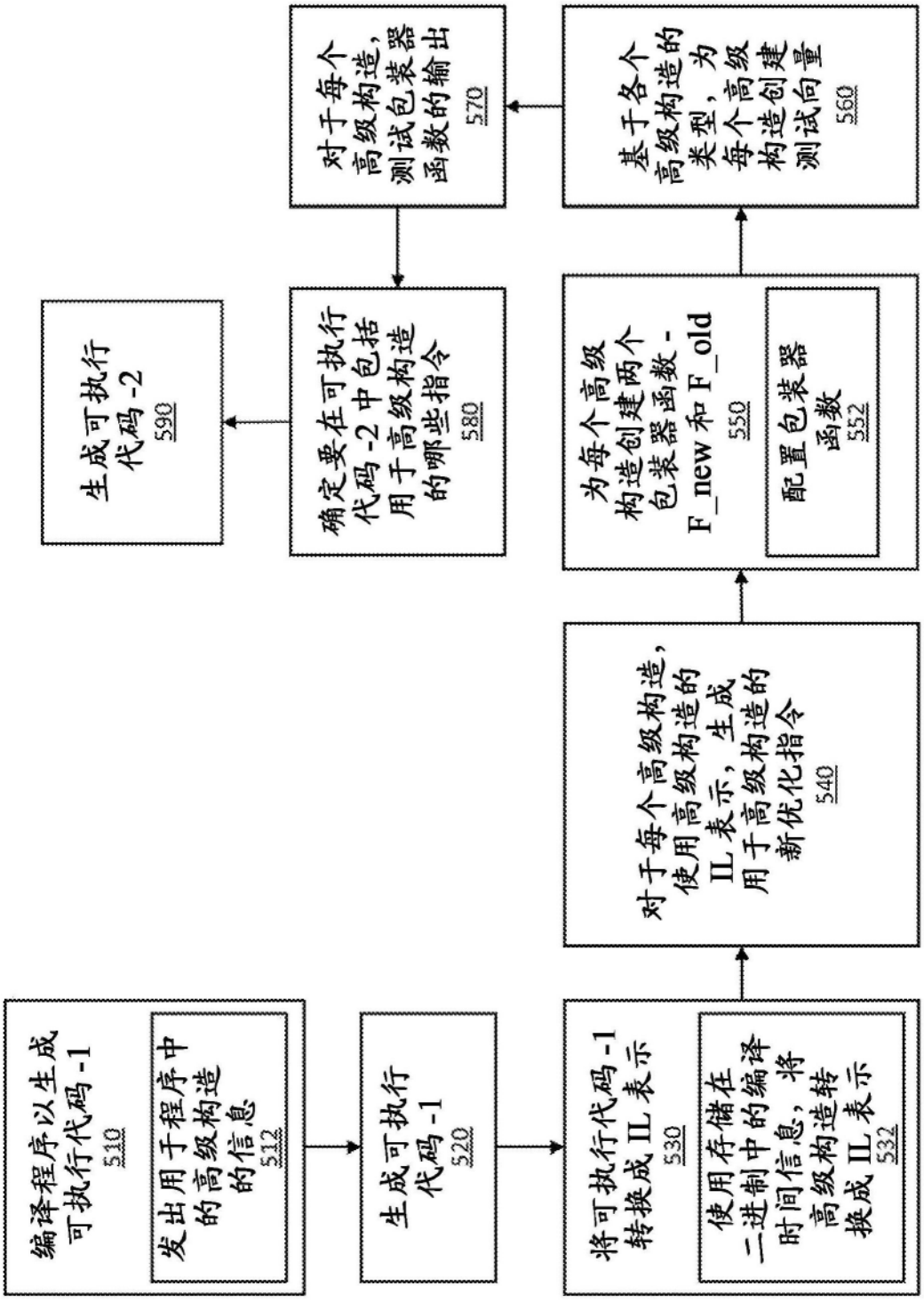

编译程序以生成可执行代码-1
510
发出用于程序中的高级构造的信息
512
生成可执行代码-1
520
将可执行代码-1转换成IL表示
530
使用存储在二进制中的编译时间信息，将高级构造转换成IL表示
532
对于每个高级构造，使用高级构造的IL表示，生成用于高级构造的新优化指令
540
为每个高级构造创建两个包装器函数-F_new和F_old
550
配置包装器函数
552
基于各个高级构造的类型，为每个高级构造创建测试向量
560
对于每个高级构造，测试包装器函数的输出
570
确定要在可执行代码-2中包括用于高级构造的哪些指令
580
生成可执行代码-2
590

图5

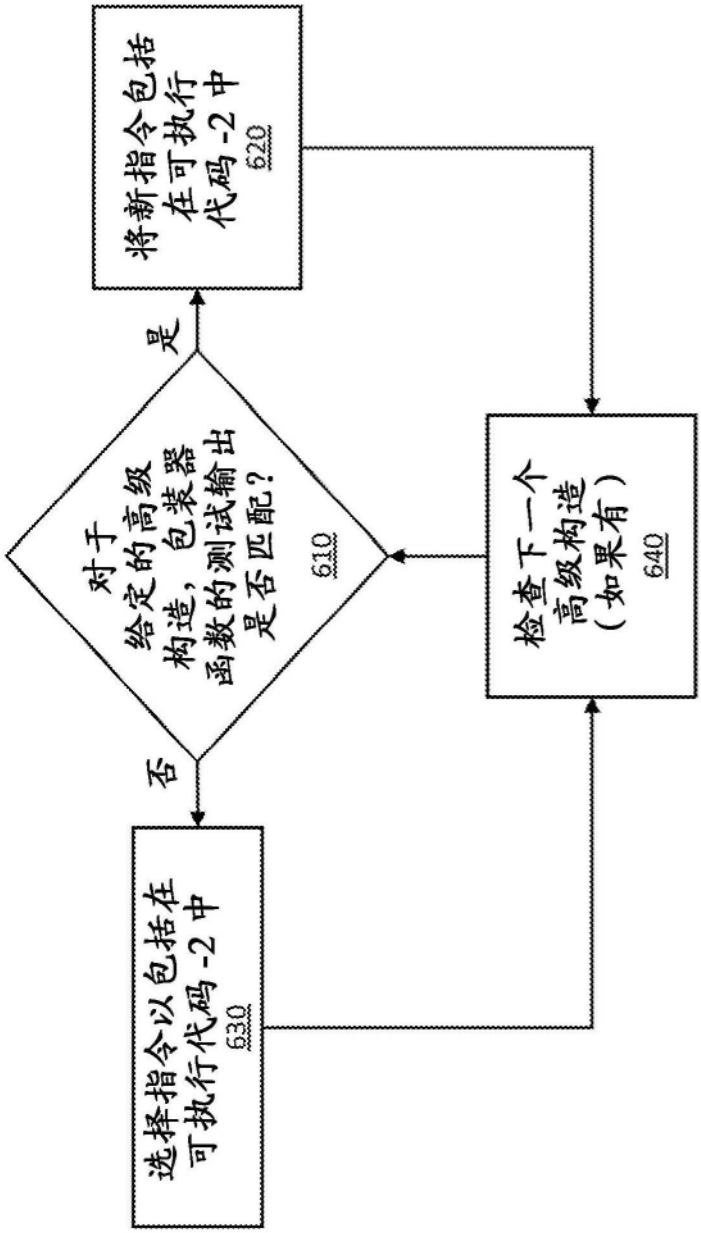

将新指令包括在可执行代码-2中
620
是
对于给定的高级构造，包装器函数的测试输出是否匹配？
610
否
检查下一个高级构造（如果有）
640
选择指令以包括在可执行代码-2中
630

图6