



(12) 发明专利申请

(10) 申请公布号 CN 104781785 A

(43) 申请公布日 2015. 07. 15

(21) 申请号 201380057733. 2

(74) 专利代理机构 中国专利代理(香港)有限公司 72001

(22) 申请日 2013. 09. 03

代理人 刘靖龙 景军平

(30) 优先权数据

13/604618 2012. 09. 05 US

(51) Int. Cl.

G06F 9/45(2006. 01)

(85) PCT国际申请进入国家阶段日

2015. 05. 05

G06F 9/445(2006. 01)

(86) PCT国际申请的申请数据

PCT/US2013/057892 2013. 09. 03

(87) PCT国际申请的公布数据

W02014/039458 EN 2014. 03. 13

(71) 申请人 微软公司

地址 美国华盛顿州

(72) 发明人 S. 特加尼 A. M. 特鲁芬斯库

Y. 沙班 A. 格巴格辛 A. 巴巴

蔡美琴 S. 拉马斯瓦米

C. L. 费尔南多

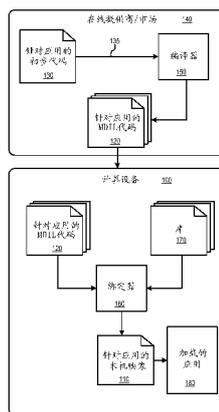
权利要求书2页 说明书17页 附图10页

(54) 发明名称

为应用从中间语言代码生成本机代码

(57) 摘要

本文中所公开的是用于通过从中间语言形式的代码生成本机代码来安装、执行和 / 或更新受管理应用的工具和技术的代表性实施例。根据一个示范性技术, 计算设备接收由在先提供商针对应用所生成的机器相关中间语言代码(MDIL 代码)。附加地, 计算设备通过生成针对应用的本机映像(其包括将 MDIL 代码的一部分与计算设备上的一个或多个库绑定) 来将应用安装在计算设备上。并且, 本机映像被存储在计算设备上以用于在加载应用以用于执行时使用。



1. 一种方法,其包括:

在计算设备处,接收由在线提供商针对应用所生成的机器相关中间语言代码(MDIL 代码);

在计算设备处,通过经由绑定 MDIL 代码生成针对应用的本机映像来将应用安装在计算设备上,生成本机映像包括:

将 MDIL 代码的一部分与计算设备上的一个或多个库绑定;以及

将本机映像存储在计算设备上以用于在加载应用以用于执行时使用。

2. 根据权利要求 1 所述的方法,其进一步包括更新计算设备,更新计算设备包括:

更新计算设备上的库的集合,库的集合包括绑定到 MDIL 代码的所述部分的一个或多个库中的至少一个库;

更新计算设备的至少一个运行时引擎;

在库的集合的更新之后,通过在一个或多个库在计算设备上被更新之后将 MDIL 代码的所述部分与一个或多个库中的至少一个库绑定来自动地生成针对应用的更新的本机映像,针对应用的经更新的本机映像被生成,使得经更新的本机映像可使用经更新的至少一个运行时引擎运行;以及

将经更新的本机映像存储在计算设备上以用于在加载应用以用于执行时使用。

3. 根据权利要求 1 所述的方法,进一步包括在应用的执行期间加载本机映像,其中本机映像的加载由公共语言运行时执行。

4. 根据权利要求 1 所述的方法,其中 MDIL 代码包括在二进制级别下编码的一个或多个机器代码指令和一个或多个伪指令,并且其中机器代码指令是基于针对计算设备的处理器的处理器指令集的。

5. 根据权利要求 4 所述的方法,其中将 MDIL 代码与一个或多个库绑定包括:

至少通过从 MDIL 代码中的伪指令中的伪指令生成本机语言指令来生成可执行代码。

6. 根据权利要求 5 所述的方法,其中生成本机语言指令包括基于伪指令来生成数值字段偏移。

7. 根据权利要求 6 所述的方法,其中伪指令包括识别字段的记号并且本机指令包括用来在被执行时引用字段的数值字段偏移。

8. 根据权利要求 1 所述的方法,进一步包括:

接收针对应用的新的 MDIL 代码;

通过生成针对应用的更新的本机映像来更新计算设备上的应用,其包括将 MDIL 代码的新的一部分与计算设备上的一个或多个库绑定;以及

将经更新的本机映像存储在计算设备上以用于在加载应用以用于执行时使用。

9. 一种包括处理器和存储器的计算设备,存储器存储用于使计算设备执行方法的计算机可执行指令,所述方法包括:

在计算设备处,从在线提供商接收针对应用的至少一个安装包,所述至少一个安装包包括机器相关中间语言文件(MDIL 文件)的集合;

将 MDIL 文件的集合中的至少一个 MDIL 文件和要被绑定到该至少一个 MDIL 文件的一个或多个库提供给计算设备的绑定器;以及

采用绑定器,生成针对应用的本机映像,所述生成包括使用一个或多个库来绑定至少

一个 MDIL 文件的机器相关中间语言代码(MDIL 代码)。

10. 一种方法,其包括:

通过将应用的初步代码编译成机器相关中间语言代码(MDIL 代码)来为应用生成机器相关中间语言文件(MDIL 文件)的集合;

对 MDIL 文件的集合的相应文件签名以便指示相应文件被信任为来自在线市场;

生成识别 MDIL 文件的集合的相应文件的绑定列表;

生成针对应用的安装包,其包括针对应用的 MDIL 文件的集合和绑定列表;以及在在线市场处提供安装包以用于下载。

为应用从中间语言代码生成本机代码

背景技术

[0001] 各种移动设备还支持这样的应用,其能够以不能够被直接地执行但是当使用及时编译(JIT 编译)在移动设备上编译时能够被执行的表示下载。尽管代码的 JIT 编译已被用来在计算机上运行应用,但是使用 JIT 编译来运行应用具有局限性,包括当应用运行时编译所需的附加时间以及在编译期间做出的潜在地非最佳的决定。

发明内容

[0002] 在本文中所描述的其它创新当中,本公开内容呈现了用于通过从中间语言形式的代码生成本机代码来安装、执行和 / 或更新受管理应用的工具和技术的各种代表性实施例。根据一个示范性技术,计算设备接收由在线提供商针对应用所生成的机器相关中间语言代码(MDIL 代码)。计算设备通过生成针对应用的本机映像(包括通过将 MDIL 代码的一部分与计算设备上的一个或多个库绑定)来将应用安装在计算设备上。并且,本机映像被存储在计算设备上以用于在加载应用以执行时使用。

[0003] 根据示范性工具,计算设备从在线提供商接收针对应用的安装包,其中安装包包括机器相关中间语言文件(MDIL 文件)的集合。附加地,计算设备的绑定器(binder)被提供 MDIL 文件的集合中的至少一个文件以及要被绑定到该至少一个文件的一个或多个库。绑定器通过使用一个或多个库来绑定至少一个 MDIL 文件的 MDIL 代码而生成针对应用的本机映像。

[0004] 在另一示范性技术中,在线提供商从针对应用的初步代码生成 MDIL 代码。计算机系统通过将应用的初步代码编译成 MDIL 代码来为应用生成 MDIL 文件的集合,然后对 MDIL 文件的集合的相应文件签名以便指示相应文件被信任为来自在线市场。计算机系统生成识别 MDIL 文件的集合的相应文件的绑定列表,并且还生成针对应用的包括针对该应用的 MDIL 文件的集合和绑定列表的安装包。计算机系统然后在在线市场提供安装包以用于下载。

[0005] 当它接收到安装包时,计算设备接收 MDIL 代码并且生成针对应用的本机映像以便将应用安装在计算设备上。附加地,计算设备的运行时引擎和 / 或由所安装的应用使用的一个或多个库能够在对计算设备的更新期间在计算设备上被更新,并且作为响应应用被自动地更新。应用通过使用被更新的一个或多个库来生成针对应用的更新的本机映像而被更新。经更新的本机映像被生成,使得它可使用计算设备上的经更新的运行时引擎运行。在经更新的本机映像被生成之后,应用通过加载经更新的本机映像而不是先前生成的本机映像而被运行。

[0006] 本发明内容被提供来以简化的形式引入在下面被进一步描述的构思的选择。本发明内容不旨在识别所要求保护的的主题的关键特征或必要特征,它也不旨在被用来限制所要求保护的的主题的范围。技术的前面的和其它的目标、特征以及优点从参考附图进行的以下具体描述将变得更明显。

附图说明

[0007] 图 1 是图示了示范性在线提供商 / 市场以及示范性计算设备的图, 所述示范性计算设备加载通过绑定机器相关中间语言代码 (MDIL 代码) 所安装的应用的本机映像。

[0008] 图 2 是绑定应用的 MDIL 代码以便将应用安装在计算设备上的示范性方法的流程图。

[0009] 图 3 是图示了示范性在线提供商 / 市场以及示范性计算设备的图, 所述示范性计算设备能够通过从由在线提供商 / 市场 (或其它在线提供商 / 市场) 提供和生成的中间语言形式的代码生成一个或多个本机映像来安装一个或多个受管理应用。

[0010] 图 4 是用于使用 MDIL 形式的代码通过本机代码生成来安装和更新应用的示范性方法的流程图。

[0011] 图 5 是生成针对应用的安装包的示范性方法的流程图, 其中安装包能够被提供以用于下载。

[0012] 图 6 是能够为一个或多个安装的应用生成一个或多个更新的本机映像的计算设备的图。

[0013] 图 7 是用于通过从 MDIL 形式的代码生成针对应用的本机映像来更新应用的示范性方法的流程图。

[0014] 图 8 是描绘了所公开的实施例中的至少一些能够采用其被实施的示范性移动设备的示意图。

[0015] 图 9 是图示了针对所公开的实施例中的至少一些适合的实施环境的一般性例子的示意图。

[0016] 图 10 是图示了针对所公开的实施例中的至少一些适合的计算环境的一般性例子的示意图。

具体实施方式

[0017] 本公开内容呈现了用于通过从中间语言形式的代码生成本机代码来安装、执行和 / 或更新受管理应用的工具和技术的各种代表性实施例。例如, 计算设备接收由在线提供商针对应用所生成的 MDIL 代码。计算设备通过生成针对应用的本机映像来安装应用。作为本机映像的生成的一部分, 计算设备将 MDIL 代码的一部分与计算设备上的一个或多个库绑定。计算设备存储本机映像以用于在加载应用以执行时使用。以这种方式, 应用的性能能够受益于该应用的至少一部分的之前的离线编译 (以便产生适于设备的 MDIL 代码)。同时, 当存在对由应用使用的库和 / 或针对应用的 MDIL 代码的改变时这个方法能够便于应用的更新。

[0018] 取决于实施方案, 本文中所描述的技术和工具的各种特征能够被相结合地或单独地使用。本文中所描述的技术和工具的不同特征影响处理的不同阶段, 包括摄取、安装、执行、应用更新以及设备更新。术语摄取一般地指代开发者将应用上传到在线提供商的过程, 所述在线提供商处理应用以用于在一个或多个类型的计算设备上的高效执行, 验证应用, 并且使应用可用于下载。术语安装一般地指代适配应用以便在特定计算设备上运行的过程, 所述特定计算设备将应用转换为更适合于在设备上迅速且安全执行的形式 (例如, 在绑定期间通过解析 (resolve) 对库和其它资源的引用来将 MDIL 代码转换为本机指令、存储能

够被加载以用于执行的本机映像、将应用标记为可信等)。在执行期间,针对应用的可信本机映像能够被迅速地加载并且运行。术语应用更新一般地指代应用被更新(例如,针对应用的 MDIL 代码)的过程,其能够牵涉应用在设备上的重新安装。术语计算设备更新一般地指代跟随在运行时引擎之后由应用所引用的库或其它资源已被更新的过程,其典型地牵涉重新绑定取决于已改变的库或其它资源的应用。

[0019] 用于通过从 MDIL 形式的代码生成本机代码来安装和加载应用的示范性系统

图 1 是图示了示范性在线提供商 / 市场以及示范性计算设备 100 的图,所述示范性计算设备加载通过绑定 MDIL 代码 120 所安装的应用的本机映像 110。在图 1 中,针对应用的初步代码 130 被在线提供商 140 接收。例如,初步代码 130 可以是作为源代码或诸如微软中间语言 (MSIL) 这样的中间语言形式的代码的针对应用的代码。在 135 处,初步代码被提供给编译器 150,所述编译器 150 将初步代码 130 编译成针对应用的 MDIL 代码 120。初步代码 130 可以是 MDIL 代码 120 之外的针对应用的代码的更高级别表示。例如,初步代码 130 采用在比 MDIL 代码 120 更高的级别下的中间语言 (IL),所述 MDIL 代码 120 是在更接近于机器代码的级别下。MDIL 代码 120 可以是机器相关的,使得它包括基于处理器指令集的本机代码。处理器指令集可以是针对 ARM 处理器、x86 处理器或其它处理器的指令集。MDIL 代码 120 还能够包括具有符号引用的未解析的伪指令,所述符号引用能够被解析以便通过绑定来生成本机代码。

[0020] 针对应用的 MDIL 代码 120 在计算设备 100 处被接收。作为应用的安装的一部分,MDIL 代码 120 被提供给绑定器 160,所述绑定器 160 将 MDIL 代码 120 绑定到设备上的一个或多个库 170。一个或多个库 170 可以是针对类、类型等的代码库和 / 或框架库。通过在计算设备 100 上绑定 MDIL 代码 120,绑定器 160 生成应用的本机映像 110。本机映像 110 包括针对应用的本机代码并且被存储在设备处。在 180 处,本机映像被加载来执行应用。

[0021] 用于绑定 MDIL 代码以便将应用安装在计算设备上的示范性方法

图 2 是绑定应用的 MDIL 代码以便将应用安装在计算设备上的示范性方法 200 的流程图。在各种实施方案中,图 2 的所图示的方法块可以被合并、划分成子块或者省略。

[0022] 在图 2 中,在 210 处计算设备从在线提供商接收针对应用的 MDIL 代码。例如,计算设备能够接收包括针对应用的代码的一个或多个文件,其中代码采用为机器相关的 IL 的形式,诸如如美国专利申请公开 No. : US 2011/0258615 中所描述的机器相关中间语言的形式或另一机器相关中间语言的形式代码。因为 MDIL 代码在计算设备处从在线提供商被接收,所以 MDIL 代码可以是从小通过在线提供商或在云中别处的编译生成的。例如,在线提供商编译针对应用的初步代码以便生成针对应用的 MDIL 代码。

[0023] 在一些实施方案中,针对应用的安装包被接收并且该安装包包括针对应用的 MDIL 代码。针对应用的安装包能够包括诸如针对应用的资源文件这样的针对应用的其它信息。针对应用的 MDIL 代码能够被存储在计算设备上以用于应用的安装并且用于应用的后续更新。

[0024] 在 220 处,应用通过生成应用的本机映像(这包括绑定应用的 MDIL 代码)而被安装在计算设备上。例如,绑定器能够将 MDIL 代码绑定到在计算设备上可用的一个或多个库以便生成针对应用的可执行的本机映像。应用可以是使用诸如 .NET 框架的公共语言运行时这样的运行时或执行引擎或使用其它运行时引擎执行的受管理应用。运行时引擎和 / 或执

引擎可以是应用最终在其上运行的软件层。在一些实施方案中,受管理应用是使用包括一个或多个共享库和一个或多个受管理代码运行时环境的框架来执行的应用。例如,库是在应用当中被共享的基类库。在一些实施方案中,计算设备上的库可以是本机和 / 或机器代码中的库的映像,其能够被加载到存储器中以用于由也被加载到存储器中的受管理应用使用。在一些实施方案中,所生成的本机映像能够由绑定器生成,使得本机映像可由被用来在计算设备上运行应用的运行时引擎或执行引擎运行。例如,绑定器能够在绑定期间使用和 / 或引用针对应用的运行时引擎,以便绑定适于使用运行时引擎执行。

[0025] 在一些实施方案中,绑定器的绑定解析 MDIL 代码并且生成针对应用的本机代码,其被包括在针对应用的本机映像中。本机代码可以是可在特定处理器上执行并且使用来自处理器指令集的指令的机器代码。处理器指令集可以是针对 ARM 处理器、x86 处理器或其它处理器的指令集。在一些实施方案中,在生成本机映像时,对于针对应用接收到的文件的集合中的每个 MDIL 文件,相应的 MDIL 文件以及将被绑定到文件的 MDIL 代码的一个或多个库被提供给绑定器。例如,到 MDIL 文件和 / 或库文件的一个或多个文件路径能够被提供给绑定器。绑定器将相应的 MDIL 文件绑定到被提供给绑定器的库以便生成针对应用的本机映像。本机映像能够包括诸如可由计算设备的处理器执行的机器代码这样的本机代码。(附加于本机映像,针对应用的至少一些代码能够用 IL 形式保持,以用于在应用被启动之前或当应用被启动时的及时编译或其它编译。当资源频繁地改变时或者当类型值或其它引用在应用被安装并且剩余的 MDIL 代码被绑定到设备上的库时不能够被解析时这可能是有用的)。

[0026] 在一些实施方案中,应用能够通过包管理器被安装在设备上。例如,在应用的安装期间,针对应用的接收到的安装包能够被包管理器解包。如果安装包和 / 或在该安装包中包括的文件中的一个或多个被压缩,则包管理器能够对这样的文件进行解压。如果一个或多个文件未被压缩则包装管理器不对这样的文件进行解压。包管理器能够提供将被绑定到绑定器的 MDIL 文件中的每一个以及将被绑定到相应的 MDIL 文件的库。在一些实施方案中,包管理器通过对 MDIL 文件和 / 或一个或多个库的引用提供给绑定器来提供 MDIL 文件和 / 或一个或多个库。绑定器将相应的 MDIL 文件绑定到针对相应的 MDIL 文件的所提供的库来生成应用的本机映像。本机映像能够由绑定器生成,使得本机映像可使用在计算设备上可用的运行时引擎运行以便运行应用。

[0027] 在 230 处,应用的本机映像被存储在计算设备上以用于在加载应用以用于执行时使用。例如,所生成的本机映像被加载来执行应用(并且不可执行的 MDIL 代码未被用来执行应用)。在一些实施方案中,本机映像在被加载以用于执行之前被确定为可信的。在其它实施方案中,在本机映像被加载以用于执行之前计算设备不做出关于本机映像是否可信的任何确定。

[0028] 从提供的 MDIL 代码生成本机代码以用于应用安装和执行的示范性系统

图 3 是图示了示范性在线提供商 / 市场以及示范性计算设备 350 的图,示范性计算设备能够通过从诸如由在线提供商(或其它在线提供商)提供和生成的 MDIL 这样的中间语言形式的代码生成一个或多个本机映像来安装一个或多个受管理应用。在图 3 中,在线提供商 305 使用编译器 310 来将初步代码 315 编译成一个或多个 MDIL 文件 320,其是包括 MDIL 形式的代码的文件。一个或多个 MDIL 文件 320 的集合被包装成针对应用的安装包 325。MDIL 文件中的每一个能够被签名为可信的,如在 330 处所示出的。包括诸如资源文件或应

用内容这样的支持信息的一个或多个文件 335 能够被包括在安装包 325 中。安装包 325 被在线提供商 305 存储并且被提供用于下载,如在 340 处所示出的。在安装包 325 从在线提供商 305 被下载之前,安装包 325 它本身能够被签名为可信的,如在 345 处所示出的。

[0029] 因此,安装包 325 作为在线市场中的一类应用的一部分被提供。安装包 325 的不同版本能够针对不同类型的计算设备被提供。例如,不同类型的 MDIL 代码(适配用于不同类型的处理器)针对不同类型的计算设备被包括在不同的安装包中。出于后向兼容性的目的,可用于不同类型的计算设备的安装包还能够包括具有非 MDIL 代码(例如,XAP 文件中的 MSIL 代码)的版本。

[0030] 在 355 处,计算设备 350 从在线提供商下载一个或多个安装包(包括安装包 325)。作为安装应用的一部分,计算设备 350 的包管理器 360 对包括针对应用的支持信息的一个或多个文件 335 和一个或多个 MDIL 文件 320 进行解包。包管理器还调用绑定器 365。对于一个或多个 MDIL 文件 320 的集合中的每个 MDIL 文件,包管理器 360 将诸如 MDIL 文件 322 这样的相应的 MDIL 文件提供给绑定器 365,连同提供要被与相应的 MDIL 文件绑定的一个或多个库(诸如一个或多个库 370)。绑定器 365 将相应的 MDIL 文件绑定到针对相应的 MDIL 文件所提供的的一个或多个库以便生成本机代码。针对 MDIL 文件提供的库(诸如一个或多个库 370)被包括在计算设备 350 上的库的集合 375 中。使用从绑定一个或多个 MDIL 文件 320 生成的本机代码,绑定器生成针对应用的本机映像 380。本机映像 380 能够由绑定器 365 生成,使得本机映像 380 可使用能够运行应用的计算设备上的运行时引擎运行。本机映像 380 作为将应用安装在设备上的一部分被存储,如在 385 处所示出的。计算设备 350 还能够安装一个或多个其它应用并且存储与相应的其它应用相对应的生成的本机映像。

[0031] 如在 390 处所示出的,由绑定器 365 从可信代码生成的本机映像 380 能够被签名为可信代码。本机映像 380 能够在它被存储和 / 或加载以用于在执行应用时使用之前被签名。在 395 处,本机映像 380 被加载来执行应用。当被加载时,本机映像的本机代码被计算设备 305 执行来提供应用的功能性。

[0032] 用于使用 MDIL 形式的代码通过本机代码生成来安装并且更新应用的示范性方法

图 4 是用于使用 MDIL 形式的代码通过本机代码生成来安装和更新应用的示范性方法 400 的流程图。在图 4 中,如在 410 处所示出的,包括 MDIL 代码的安装包被生成。在一些实施方案中,MDIL 代码能够被包括在一个或多个文件中。在一些实施方案中,初步代码能够被编译以便生成 MDIL 代码。

[0033] 在一些实施方案中,针对应用的 MDIL 代码包括基于处理器指令集的本机代码,但是还能够包括伪指令。例如,伪指令包括在 MDIL 被生成时不能够被解析的符号引用和 / 或字段密钥。因此,在一些实施方案中,伪指令能够包括对字段、类型等的符号引用。例如,符号引用可以是在不用具体地指示字段的偏移或类型的大小的情况下识别要被使用的字段或类型的记号(token)。在一些实施方案中,符号引用可以是能够被绑定器解释来确定要针对该符号引用被生成的本机代码的记号。例如,对于诸如引用对象的字段的字段记号这样的记号,特定计算设备上的绑定器能够(1)在其通过库在计算设备上可用时展示(layout)所述对象并且(2)在为计算设备生成可执行指令时确定针对对象的字段的适当的偏移。因此,在一些实施方案中,针对 MDIL 代码中的对象的字段偏移可以是符号的而不是被硬编码为数值偏移(诸如在对应的本机代码中)。

[0034] 伪指令能够抽象当伪指令被绑定到计算设备上的适当库时将被确定和 / 或解析的实施细节。在一些实施方案中, MDIL 代码包括包含符号引用的伪指令, 所述符号引用能够被绑定器转换以便产生可由处理器执行的本机指令。例如, MDIL 代码能够包括伪指令, 所述伪指令能够被绑定器转换成机器指令以用于访问由诸如 .NET 框架的公共语言运行时 (CLR) 等这样的运行时引擎管理的对象的字段。在一些实施方案中, MDIL 代码中的伪指令可以与本机语言形式的机器代码类似, 使得寄存器已经被分配但是对于一个或多个库和 / 或类来说偏移尚未被包括。例如, 在如从伪指令转换的本机指令中包括的字段偏移能够由绑定器确定并且能够取决于可用的和 / 或在计算设备上的库的版本。在一些实施方案中, MDIL 代码能够包括能够被绑定器转换成用于查找通用类型或方法的实例化或实例的机器指令的伪指令。

[0035] 参考图 4, 在 420 处, 计算设备接收针对应用的 MDIL 代码。例如, 计算设备能够接收针对应用的包括 MDIL 文件的安装包, 所述 MDIL 文件包括针对应用的 MDIL 代码。在一些实施方案中, 安装包能够包括绑定列表, 所述绑定列表包括应用的将被绑定来生成针对应用的本机映像的 MDIL 文件的列表以及将被绑定到相应的 MDIL 文件的一个或多个库的集合的列表。在一些实施方案中, 要被绑定到应用的 MDIL 代码的库中的一个或多个是在应用的源代码中被声明和 / 或包括的库。要被绑定到应用的 MDIL 代码的一个或多个其它库可以是在应用的源代码中未被声明和 / 或包括的库。

[0036] 在 430 处, 应用通过生成针对应用的本机映像而被安装在计算设备上。在一些实施方案中, 应用可以由包管理器安装。例如, 包管理器能够在使用安装包的文件来将应用安装在计算设备上之前确定安装包被签名为可信的, 并且如果安装包未被签名为可信的, 则包管理器不使用该安装包来将应用安装在计算设备上。包管理器能够提供将被绑定到绑定器的 MDIL 文件中的每一个以及将被绑定到相应的 MDIL 文件的库。在一些实施方案中, 代替或附加于确定整个安装包是否被签名为可信, MDIL 文件能够在被用来生成针对应用的本机映像之前被确定为被签名为可信的, 并且被确定为未被签名为可信的 MDIL 文件不被用来生成针对应用的本机映像。在一些实施方案中, 包管理器能够被提供并且使用绑定列表来自动地确定并且识别将被绑定的 MDIL 文件以及将被绑定到针对应用的相应的 MDIL 文件的代码的相应的库。绑定列表可以是机器可读格式的形式, 诸如编程语言、脚本语言或标记语言这样的形式。例如, 绑定列表能够被包括在文件中, 所述文件包括使用可扩展标记语言 (XML) 组织的信息。

[0037] 在一些实施方案中, 包管理器通过对 MDIL 文件和 / 或一个或多个库的引用提供给绑定器来将 MDIL 文件和 / 或一个或多个库提供给绑定器。绑定器将相应的 MDIL 文件绑定到针对相应的 MDIL 文件的所提供的库以便生成应用的本机映像。在高级别下, 绑定器读取 MDIL 指令, 将 IL 指令转换为本机指令, 并且通过已经具有本机形式的指令。例如, 绑定器能够将 MDIL 代码绑定到计算设备上的一个或多个库以便生成针对应用的可执行的本机映像。本机映像能够由绑定器生成, 使得本机映像可使用计算设备上的运行时引擎运行, 所述运行时引擎可被绑定器利用、用于运行应用。运行时引擎可以是能够在计算设备上最终运行应用的软件层。在一些实施方案中, 绑定器能够使用来自脚本的信息、用户输入和 / 或其它信息被运行。在绑定代码的一些实施方案中, MDIL 代码被提供给绑定器。绑定器解析伪指令以便生成对应的本机指令。例如, 根据伪指令中的符号引用 (例如, 字段记号), 绑定器

能够通过基于类型布局确定并且指定针对字段的数值偏移来生成本机代码。在一些实施方案中,MDIL 伪指令能够被用来创建可能在长度上不同于伪指令的本机指令。替换地,本机指令可能不在长度上不同于伪指令。在一些实施方案中,针对应用的 MDIL 代码中的至少一些可能接近于在应用的本机映像中包括的机器代码;然而,MDIL 代码的其它部分能够提供允许 MDIL 代码的一部分被绑定到可能在一个或多个版本中可用的库的一定级别的抽象。通过绑定,MDIL 代码能够被用来生成使用库的特定版本执行的可执行代码。例如,将 MDIL 代码绑定到库的第一集合能够生成针对应用的可执行代码。或者,将 MDIL 代码绑定到其中库中的一个或多个具有不同版本的集合库也能够生成针对应用的可执行代码。一个或多个库的版本的集合能够具有库的重叠版本。

[0038] 在一些实施方案中,绑定 MDIL 代码生成能够由处理器在没有进一步编译的情况下直接地执行的代码。例如,将 MDIL 代码绑定到库包括从 MDIL 代码中的指令生成本机指令。在一些实施方案中,MDIL 代码包括伪指令,伪指令包括符号引用,所述符号引用能够被绑定器转换和 / 或绑定来产生可由处理器执行的本机指令。例如,绑定器能够将伪指令转换成本机代码指令以用于访问对象的字段,其能够由计算设备上的运行时引擎管理。在一些实施方案中,MDIL 代码的部分可以与所生成的本机代码类似,使得寄存器被分配,但是在 MDIL 代码中对于一个或多个库和 / 或类来说偏移尚未被包括。在一些实施方案中,绑定器能够将伪指令转换成本机代码指令以用于查找通用类型或方法的实例化或实例。

[0039] 在一些实施方案中,MDIL 代码符号指令和所生成的对应的本机指令相对于虚拟方法时隙指派和 / 或对象字段布局是一致的。例如,MDIL 代码和可执行本机代码相对于对象字段布局可能是一致的,使得用于访问包括字段记号的对象的 MDIL 代码的一部分能够被用来使用数值字段偏移来生成本机指令(例如,机器指令)。在一些实施方案中,绑定器能够从符号引用生成数值偏移以便生成一个或多个可执行机器指令。例如,MDIL 代码中的符号引用被绑定器转换成特定字段引用,使得该特定字段引用包括数值大小和 / 或偏移。

[0040] 在绑定 MDIL 代码的一些实施方案中,绑定器能够解释包括记号的符号应用以便确定要针对该符号引用被生成的本机代码。例如,对于引用对象的字段的字段记号,绑定器能够在对象可能在计算设备上可用时展示对象并且确定针对对象的字段的适当的偏移以用于生成针对计算设备的可执行指令。在绑定 MDIL 代码的一些实施方案中,能够绑定 MDIL 代码的绑定器比链接器(linker)更复杂。例如,在绑定 MDIL 代码时,绑定器能够调整结果得到的本机代码中的一个或多个跳转,以便跳转将到达它们的预定目标。

[0041] 附加于本机映像中的本机代码,针对应用的至少一些代码能够用 IL 形式保持。在应用被启动之前或当应用被启动时,IL 形式的代码在及时编译或其它编译期间被进一步处理。这在运行时引擎、库或其它资源频繁地改变时可能是有用的。或者,它在类型值或引用在绑定过程期间不能够被解析时可能是有用的,在所述绑定过程中剩余的 MDIL 代码被绑定到计算设备上的库。

[0042] 在 440 处,本机映像被存储用于在加载应用以用于执行时使用。例如,当用户在计算设备上启动应用时,应用的本机映像被诸如 .NET 框架的 CLR 这样的执行引擎或运行时或其它运行时引擎加载。在一些实施方案中,计算设备能够包括针对应用的 MSIL 代码,其能够在计算设备上使用及时编译被编译来执行应用,但是不是使用 MSIL 代码来执行应用,而是针对应用的本机映像被加载并且执行来运行应用。在一些实施方案中,当本机映像被加

载时,被应用的如包括在本机映像中的代码使用的一个或多个库映像或组件被加载。例如,本机映像能够被加载以用于使用计算设备的运行时引擎执行。针对应用的本机映像能够在它被存储之前被绑定器签名,以便示出应用是可信的。

[0043] 在 450 处,计算设备至少通过更新该计算设备上的库的集合而被更新。例如,计算设备的库的集合中的库的一个或多个能够被更新、代替或者改变。这可能破坏将被绑定到库的集合的先前版本的应用的操作。因此,在一个或多个库被更新、代替或者改变之后,计算设备上的应用中的一个或多个能够被更新,使得针对相应应用的更新的本机映像能够被生成。针对应用的更新的本机映像能够通过将针对应用的 MDIL 代码(其甚至在安装之后也被以 MDIL 形式存储在计算设备上)绑定到被更新了的一个或多个库的集合中的至少一个而被生成。针对应用的 MDIL 代码在库被更新之后到计算设备上的库中的一个或多个的重新绑定能够允许本机映像在更新设备和 / 或更新设备的库之后恰当地引用如在计算设备上可用的一个或多个库。同时,每当库改变时,针对应用的代码的完整重新编译是不需要的,因为针对应用的 MDIL 代码能够被再使用。当经改变的库被设备上的应用中的许多使用时,使用针对相应应用的先前存储的 MDIL 代码来重新绑定能够大大地简化更新针对相应应用的本机映像的过程。

[0044] 例如,假设七个应用各自依赖于已被改变的库。对于七个应用中的每一个,云中的离线编译(例如,通过在线提供商)能够使用比在设备上出于更新的目的的编译或及时编译更复杂和 / 或费时的编译来为该应用产生高效的 MDIL 代码。不是在库改变时重新编译七个应用中的每一个(这将是费时的并且可能导致不太高效的本机映像),而是相应应用被重新绑定到计算设备上的库,这快得多,因为对编译优化的“繁重”在云中先前完成了。

[0045] 在一些实施方案中,计算设备被更新使得更新包括更新计算设备的运行时引擎。例如,计算设备的使用针对应用的本机映像来运行一个或多个安装的应用的运行时引擎能够被更新、代替或者改变。运行时引擎可以是最终运行更新的应用的软件层,如此当运行时引擎被改变时针对更新的应用的更新的本机映像能够由绑定器生成以便可使用如对绑定器可用并且在计算设备上可用的经更新的运行时引擎执行。在一些实施方案中,计算设备的运行时引擎和库两者在设备更新时被改变并且应用能够被更新,以便它能够用经更新的运行时引擎和更新的库被恰当地执行和 / 或运行。

[0046] 生成针对应用的能够被提供用于下载的安装包的示范性方法

图 5 是生成针对应用的能够被提供用于下载的安装包的示范性方法 500。在各种实施方案中,图 5 的所图示的方法块可以被合并、划分成子块或者省略。在图 5 中,针对应用的初步代码被接收。例如,应用的开发者能够在初步语言和 / 或状态下发送针对应用的代码,并且针对应用的初步代码能够被诸如在线市场这样的在线提供商接收到。在一些实施方案中,针对应用的初步代码是针对应用的源代码。例如,源代码可以用诸如 C++、C#、Visual Basic®、J#、J++ 或其它类似编程语言这样的一种或多种编程语言编写的代码。在一些实施方案中,针对应用的初步代码是诸如 IL 这样的中间表示的形式。例如,针对应用的源代码能够通过编译器和 / 或生成器被编译成 IL 形式的中间表示。在一些实施方案中,IL 可以是微软中间语言(MSIL)的形式。在一些实施方案中,IL 不包括为机器相关的本机指令。然而,IL 形式的代码能够被编译成包括为机器相关的本机指令的 MDIL 形式的代码。

[0047] 在一些实施方案中,针对应用的其它信息以及针对应用的初步代码被接收。例如,

被应用在它被执行时使用的针对应用的资源文件能够被接收。针对应用的资源文件可以是包括声音信息、音乐信息、图形信息、视频信息、其它媒体信息、数据库信息、文本信息等的一个或多个文件。在一些实施方案中,初步代码能够被包装在诸如 ZIP 文件或 XAP 文件这样的—个或多个压缩文件中和 / 或在其中接收。替换地,初步代码未被包装在一个或多个压缩文件中和 / 或未在其中接收。在一些实施方案中,具有针对应用的 MDIL 代码的一个或多个文件被在线提供商接收,但是从开发者接收到的 MDIL 文件能够作为不是由在线提供商生成的非置信文件被丢弃。替换地,在线提供商不丢弃由在线提供商所接收到的任何 MDIL 文件。在一些实施方案中,初步代码能够在—个或多个组件中被接收到。例如,初步代码(诸如在组件中包括的 MSIL 形式的代码)能够被编译成在二进制级别下的代码。包括 MSIL 代码的组件可以是 MSIL 组件。在一些实施方案中,MSIL 组件能够包括—个或多个文件。

[0048] 在 520 处,应用的初步代码被编译来产生 MDIL 代码,其是机器相关的中间语言形式的代码。作为一般规则,IL 文件内的决定或引用在编译过程期间被解析,但是对库、外部源、外部类型等的决定或引用未被解析。MDIL 文件因此是(a)组件指令、本机代码或其它设备相关指令和(b)将稍后被解析的伪代码指令或其它设备无关指令的混合物。因此,从初步代码编译的 MDIL 代码可以是比初步代码的语言更低级别的语言的中间语言。例如,初步代码是在更高级别下(离机器代码更远)的中间语言的形式,并且 MDIL 代码在更接近于机器代码的较低级别下。即使如此,MDIL 代码也典型地在未被绑定的情况下不是直接地可执行的,因为 MDIL 代码包括不为本机代码的符号引用。替代地,MDIL 代码能够通过绑定器被转换为可执行机器代码。在一些实施方案中,针对应用的初步代码不是直接地可执行的而是在它被及时编译器编译时能够被编译成可执行代码。例如,MSIL 形式的针对应用的代码能够被及时编译来根据 MSIL 代码运行应用。在一些实施方案中,MSIL 形式的针对应用的代码能够被提供给编译器以便为该应用生成 MDIL 代码。典型地,MSIL 代码不包括本机指令,它也不包括在 MDIL 代码中包括的将被绑定器解析和 / 或绑定以便生成针对应用的本机代码的伪指令。

[0049] MDIL 代码能够被包括在一个或多个 MDIL 文件中。在一些实施方案中,MDIL 文件能够具有名称和 / 或文件扩展名,使得“.mdil”被包括在 MDIL 文件的名称和 / 或文件扩展名中。在其它实施方案中,MDIL 文件能够具有另一命名约定,使得“.mdil”未被包括在 MDIL 文件的名称和 / 或文件扩展名中。

[0050] 在一些实施方案中,附加于伪指令,当初步代码被编译时,编译器产生针对 MDIL 代码的一个或多个组件。例如,在组件中包括的 MDIL 代码能够被编译成在二进制级别下的代码。包括 MDIL 代码的组件可以是 MDIL 组件。在一些实施方案中,MDIL 组件能够包括—个或多个文件。

[0051] 在应用代码在目标计算设备(应用将在其上被执行)处被接收之前将初步代码编译成针对应用的 MDIL 代码能够提供各种优点。例如,编译时间能够被从它将应用安装在目标计算设备上花费的时间中去除。从应用在目标设备上的安装中去除编译还能够允许代码的编译时间优化在云中完成,而那个时间没有被加到它将应用安装在目标设备上花费的时间。特别地,每当库被改变时和 / 或当运行时引擎被改变时这对于取决于库的多个应用能够帮助避免目标计算设备上的费时的重新编译。

[0052] 在一些实施方案中,初步代码的编译能够产生针对应用的包括针对应用的本机代

码并且不包括 MDIL 代码的一个或多个本机代码文件。

[0053] 在一些实施方案中,对于针对应用接收到的每个 MSIL 组件文件,对应的 MDIL 文件被编译器生成。初步代码能够被编译来生成依赖针对第一处理器(针对第一类型的目标计算设备)的指令集的 MDIL 代码,并且依赖针对第二处理器(针对第二类型的目标计算设备)的指令集的不同 MDIL 代码能够被生成。例如,编译针对应用的初步代码能够产生能够被绑定以便在具有第一处理器的设备上运行的 MDIL 代码,并且能够产生针对应用的能够被绑定以便在具有不同的第二处理器的另一设备上运行的不同的 MDIL 代码。例如,第一处理器可以是 ARM 处理器并且第二处理器可以是 x86 处理器。

[0054] 在 530 处,应用的 MDIL 代码被签名为可信的。例如,从初步代码生成的 MDIL 代码文件能够被用发行者密钥签名以便指示经签名的文件包括可信代码。在一些实施方案中,通过编译初步代码而生成的本机代码能够被签名为可信的。例如,由编译器生成的本机代码文件能够被用发行者密钥签名以便指示经签名的文件包括可信代码。

[0055] 在 540 处,包括 MDIL 代码的安装包被生成。例如,MDIL 代码能够被包装到容器中,诸如能够被下载以用于安装应用的一个或多个文件。在一个实施方案中,一个或多个 MDIL 代码文件中的 MDIL 代码被包装成诸如 ZIP 文件或 XAP 文件这样的压缩文件。压缩的安装包能够降低要针对应用从在线提供商被下载的数据的量。替换地,安装包不是压缩的形式。不同类型的 MDIL 代码(针对不同类型的处理器适配的)能够针对不同类型的计算设备被包括在不同的安装包中。出于后向兼容性的目的,可用于不同类型的计算设备的安装包还能够包括具有非 MDIL 代码(例如,XAP 文件中的 MSIL 代码)的版本。

[0056] 在一些实施方案中,安装包能够包括诸如针对应用的资源文件这样的针对应用的其它信息。例如,能够被应用在它被执行时使用的针对应用的资源文件能够被接收。针对应用的资源文件可以是包括声音信息、音乐信息、图形信息、视频信息、其它媒体信息、数据库信息、文本信息等的一个或多个文件。安装包还能够包括被包括在该安装包中的文件的列表。文件的列表能够包括关于在安装包中包括的文件的信息。例如,安装包能够包括文件,所述文件包括使用可扩展标记语言(XML)组织的信息。XML 文件能够包括被包括在安装包中的文件的列表。对于所列举的文件中的每一个,XML 文件能够包括与所列举的文件相关联的信息。例如,XML 文件能够列举在安装包中包括的 MDIL 代码文件中的每一个并且指示 MDIL 代码文件中的每一个能够被绑定。XML 文件还能够指示将被绑定到相应列举的 MDIL 代码文件的代码的一个或多个库。

[0057] 在一些实施方案中,所生成的安装包本身被签名为可信的。例如,安装包能够被用发行者密钥签名以便指示它包括可信的代码和/或其它信息。在一些实施方案中,安装包能够被加密。

[0058] 在 550 处,安装包被提供用于下载。例如,在线提供商和/或在线提供商的伙伴能够存储安装包以用于下载。在线提供商可以是一个或多个应用能够从其被下载以用于安装到一个或多个计算设备上的在线市场。应用能够被包括在可用于从在线市场下载的一类应用中。例如,应用能够由计算设备使用互联网和/或云来下载。应用能够被下载使得针对应用的安装包被下载。在一些实施方案中,应用已被购买并且授权被下载到计算设备之后应用能够被下载到计算设备。在一些实施方案中,应用能够在未被购买的情况下被下载。例如,应用能够被免费提供和下载。

[0059] 为安装的应用生成更新的本地映像的示范性系统

图 6 是能够为一个或多个安装的应用生成一个或多个更新的本地映像 605 的计算设备 600 的图。在图 6 中,计算设备包括针对计算设备上的一个或多个安装的应用的一个或多个生成的本地映像 610。计算设备 600 包括更新的库 620,其包括在被绑定到应用的相应的 MDIL 代码以便生成一个或多个本地映像 610 之后被更新了的一个或多个库。因为被绑定到一个或多个本地映像 610 的库中的至少一个被更新了,所以计算设备 600 通过为所安装的应用生成一个或多个更新的本地映像 605 来更新安装在它上的应用。计算设备还包括更新的运行时引擎 670。

[0060] 在为所安装的应用生成一个或多个更新的本地映像 605 时,绑定器 625 为一个或多个安装的应用中的每一个生成更新的本地映像。如图 6 中所示出的,对于具有被绑定到已被改变和 / 或更新的至少一个库的本地映像的每个安装的应用,包管理器 630 给绑定器 625 提供相应应用的 MDIL 文件以用于重新绑定来生成针对正被更新的应用的更新的本地映像,诸如经更新的本地映像 635。经更新的本地映像能够由绑定器 625 生成,使得本地映像可使用经更新的运行时引擎 670 执行。

[0061] 为了生成经更新的本地映像 635,对于正被更新的应用的每个 MDIL 文件,绑定器 625 被提供应用的诸如 MDIL 文件 640 这样的相应的 MDIL 文件以及诸如将被绑定到该 MDIL 文件的一个或多个库 645 这样的。被提供用于绑定到 MDIL 文件的代码的一个或多个库能够包括已被更新并且被包括在更新的库 620 的集合中的至少一个库。在一些实施方案中,绑定到正被更新的应用的 MDIL 文件的代码的一个或多个库尚未被更新。

[0062] 当被生成时,针对应用的经更新的本地映像(诸如经更新的本地映像 635)作为针对计算设备上的所安装的应用的一个或多个更新的本地映像 605 中的至少一个被存储。在经更新的本地映像被用来执行应用之前,经更新的本地映像还能够被签名为可信代码,如在 650 处所示出的。在 655 处,计算设备加载针对应用的生成的经更新本地映像来执行应用,而不是加载针对应用的先前生成的本地映像。

[0063] 附加于处理对设备上的库的更新,当应用已改变(例如,应用的 MDIL 代码已改变)时设备能够更新应用。在这种情况下,设备通过生成针对应用的更新的本地映像(包括将 MDIL 代码的新的部分与设备上的(未改变的)一个或多个库绑定)来更新应用。针对应用的经更新的本地映像能够由绑定器生成,使得本地映像可使用经更新的运行时引擎运行。

[0064] 如在 660 处所示出的,先前生成的本地映像在经更新的本地映像的生成和 / 或计算设备 600 的更新期间能够被暂时存储,直到应用和 / 或计算设备 600 的更新被验证为成功的为止。如果对计算设备和 / 或对应用的更新失败,则先前生成的本地映像能够从暂时存储装置被恢复并且再次用作由计算设备加载来执行应用的本地映像。

[0065] 用于使用 MDIL 形式的代码通过本地代码生成来更新应用的示范性方法

图 7 是用于通过从 MDIL 形式的代码生成针对应用的本地映像来更新应用的示范性方法 700。在各种实施方案中,图 7 的所图示的方法块可以被合并、划分子块或者省略。在图 7 中,一个或多个库的集合在计算设备上被更新,如在 710 处所示出的。例如,计算设备上的一个或多个库的一部分或全部被改变、修改或否则更新。在一些实施方案中,一个或多个库的集合被更新并且设备的运行时引擎作为计算设备更新的一部分被更新。在一些实施方案中,对于计算设备上的一个或多个应用,被绑定到针对相应应用的 MDIL 代码(以便

在安装期间生成针对相应应用的本机映像)的一个或多个库中的诸库中的至少一个已被改变、修改或否则更新。在一些实施方案中,库被更新和/或改变使得针对应用的被绑定到库的本机映像的代码不再恰当地对应于针对库的至少一个偏移。例如,针对对象的至少一个字段的偏移在库等中已被改变等。计算设备上的库的集合能够作为操作系统更新的一部分被更新。或者,计算设备上的库能够作为对框架的更新和/或执行引擎更新的一部分被更新。例如,计算设备能够使用框架来执行受管理应用,其诸如 .NET 框架,并且作为框架的一部分的库被改变和/或更新和/或作为框架的一部分的运行引擎被改变和/或更新。在 .NET 框架中,基类库能够被改变或者更新为框架库的不同和/或较新版本。在一些情况下,并非计算设备上的所有库被更新和/或改变。在其它情况下,计算设备上的所有库被更新和/或改变。

[0066] 在 720 处,计算设备生成针对应用的更新的本机映像。例如,通过从 MDIL 代码为应用生成本机映像而被安装在计算设备上的应用能够被更新,使得在一个或多个库中的至少一个在计算设备上已被更新之后新的本机映像(例如,更新的本机映像)通过将针对应用的 MDIL 代码绑定到一个或多个库而针对应用被生成。经更新的本机映像的生成可以与在应用被原先安装时本机映像的生成类似。例如,经更新的本机映像能够像在上面针对安装过程所描述的那样使用计算设备的经更新的库中的至少一个针对应用被生成。因此,经更新的本机映像的生成在针对应用的第一本机映像已经针对应用被生成并且在第一本机映像的生成期间绑定的库中的一个或多个在计算设备上已被更新和/或改变之后发生。在一些实施方案中,当运行时引擎已经连同计算设备的库被更新时,正被更新的应用的 MDIL 代码的绑定能够完成,使得所生成的本机映像可使用如在计算设备上可用的经更新的运行时引擎和经更新的库执行。

[0067] 或者,当针对应用的 MDIL 代码被更新但是库和/或运行时引擎尚未改变时,更新的本机映像能够被更新。例如,计算设备接收针对应用的新的 MDIL 代码作为应用更新的一部分。所接收到的针对应用的新的 MDIL 代码能够像本文中所描述的那样被安装,诸如使用上面的方法 200。例如,计算设备通过为应用生成更新的本机映像(包括将新的 MDIL 代码的一部分与计算设备上的(未改变的)一个或多个库绑定)来更新应用。由绑定器针对应用生成的本机代码能够被生成使得它可使用在设备上可用于运行应用的运行时引擎运行。

[0068] 在一些实施方案中,对于正被更新的应用,针对应用存储在计算设备上的针对应用的 MDIL 代码能够被绑定到设备的一个或多个库。例如,绑定器能够将 MDIL 代码绑定到计算设备上可用的一个或多个库以便生成针对应用被更新的可执行本机映像。绑定解析 MDIL 代码并且为应用生成被包括在针对应用的经更新的本机映像中的本机代码。在一些实施方案中,在生成经更新的本机映像时,对于针对应用的 MDIL 文件的集合中的每个 MDIL 文件,相应的 MDIL 文件以及将被绑定到文件的 MDIL 代码的一个或多个库被提供给绑定器。绑定器将相应的 MDIL 文件绑定到被提供给绑定器的库以便生成针对应用的经更新的本机映像。经更新的本机映像能够包括本机代码,诸如可由计算设备的处理器执行的机器代码,并且经更新的本机映像典型地像在 MDIL 代码中那样不包括未绑定伪指令。在一些实施方案中,所生成的经更新的本机映像能够被签名为可信代码。

[0069] 在一些实施方案中,经更新的本机映像代替针对应用的先前生成的本机映像作为被加载以用于执行应用的本机映像。经更新的本机映像被加载来执行应用,因为经更新的

本机映像包括可使用计算设备的经更新的库中的至少一个执行的本机代码。先前生成的本机映像可能是不稳定的和 / 或不可靠的,使得它不恰当地或如预期那样运作,因为它可能不恰当地引用在计算设备上可用的库中的一个的早先版本。

[0070] 在一些实施方案中,当更新的本机映像正被生成时,先前生成的本机映像被暂时存储和 / 或备份直到应用和 / 或对计算设备的更新被验证为成功的为止。例如,用来更新计算设备的库的对计算设备的更新能够触发针对应用的本机映像的再生和 / 或更新。如果对计算设备的更新或针对应用的本机映像的更新失败,则失败的更新能够被回滚并且应用的被存储的先前生成的本机映像在被执行时能够被用作应用的本机映像。在一些实施方案中,应用数据也被保存。例如,应用的用户能够在应用被更新之前和在应用被更新之后生成用户数据和 / 或应用数据,用户数据和 / 或应用数据能够被保存,使得经更新的本机映像的生成不改变来自应用的先前版本的用户数据和 / 或应用数据。

[0071] 在 730 处,计算设备为安装在设备上的一个或多个其它应用生成更新的本机映像。例如,对于安装在计算设备上的每个附加应用(其具有通过将 MDIL 代码绑定到被更新了了的库中的一个或多个而被生成的本机映像),针对相应应用的更新的本机映像通过将 MDIL 代码绑定到所更新的一个或多个库而被生成。在方法 700 的各种实施方案中,如在 730 处所示出的虚线指示块 730 能够被从该方法中省略或者包括在该方法中。在一些实施方案中,在设备的一个或多个库被更新(诸如在框架更新中)之后,设备的应用能够被自动地更新,并且针对应用的更新的本机映像能够被生成,以便应用使用经更新的库来恰当地执行。如果安装的应用不使用已被更新和 / 或改变的共享库,则应用不必被更新,并且更新的本机映像不必针对应用被生成,因为针对应用的先前生成的本机映像能够采用如在设备上可用的库恰当地运作。在一些实施方案中,当运行时引擎已经连同计算设备的库一起被更新时,针对应用的经更新的本机映像能够被生成,使得应用可使用如在计算设备上可用的经更新的运行时引擎和经更新的库在计算设备上执行。

[0072] 在 740 处,经更新的本机映像被存储以用于在加载应用来在计算设备处执行应用时使用。例如,代替在经更新的本机映像的生成之前生成的应用的本机映像,经更新的本机映像被加载。经更新的本机映像被加载来执行应用,因为经更新的本机映像能够恰当地引用它使用的库。通过经由重新绑定已经下载的 MDIL 代码为应用生成更新的本机映像,使用本机映像来运行应用的计算设备能够被更新,而不必下载该应用的更新的版本,不用根据针对该应用的新近下载的信息重新安装应用,并且不用重新编译该应用。

[0073] 示范性移动设备

图 8 是描绘了在 802 处一般性地示出的包括各种可选的硬件和软件构件的示范性移动设备 800 的系统图。一般而言,移动设备中的构件 802 能够与另一构件进行通信,但是为了便于说明,并非所有连接被示出。移动设备可以是各种计算设备(例如,蜂窝电话、智能电话、平板计算机、手持计算机、个人数字助理(PDA)等)中的任一个,并且能够允许与一个或多个移动通信网络 804(诸如蜂窝网络或卫星网络)的无线双向通信。

[0074] 所图示的移动设备 800 能够包括控制器或处理器 810(例如,信号处理器、微处理器、ASIC 或其它控制和处理器逻辑电路)以用于执行如信号编码、数据处理、输入 / 输出处理、功率控制和 / 或其它功能这样的任务。操作系统 812 能够控制构件 802 的分配和使用并且支持一个或多个应用程序 814 和 / 或一个或多个软件程序 815,其诸如能够实施本文中

所描述的技术的一个或多个以用于在设备上从 IL 代码为应用生成本机代码的应用程序和 / 或软件程序。应用程序能够包括公共移动计算应用和软件(例如,电子邮件应用、日历、联系人管理器、web 浏览器、消息传送应用、运行时引擎)或任何其它计算应用。用于访问应用商店、在线市场或在线提供商的功能性 813 还能够被用于获取和更新针对应用程序 814 的代码和 / 或其它信息。

[0075] 所图示的移动设备 800 能够包括存储器 820。存储器 820 能够包括非可拆卸存储器 822 和 / 或可拆卸存储器 824。非可拆卸存储器 822 能够包括 RAM、ROM、闪速存储器、硬盘、或其它众所周知的存储器存储技术。可拆卸存储器 824 能够包括闪速存储器或订户识别模块(SIM)卡(其在 GSM 通信系统中是众所周知的)、或其它众所周知的存储器存储技术,诸如“智能卡”。存储器 820 能够被用于存储用于运行操作系统 812 以及应用 814 和应用 815 的数据和 / 或代码。示例性数据能够包括 web 页面、文本、图像、声音文件、视频数据、或要经由一个或多个有线或无线网络被发送到一个或多个网络服务器或其它设备和 / 或从其接收的其它数据集。存储器 820 能够被用来存储订户标识符(诸如国际移动订户身份(IMSI))和设备标识符(诸如国际移动设备标识符(IMEI))。这样的标识符能够被传送到网络服务器以便识别用户和设备。

[0076] 移动设备 800 能够支持一个或多个输入设备 830(诸如触摸屏 832、话筒 834、相机 836、物理键盘 838 和 / 或轨迹球 840)以及一个或多个输出设备 850(诸如扬声器 852 和显示器 854)。其它可能的输出设备(未示出)能够包括压电或其它触觉输出设备。一些设备能够服务多于一个的输入 / 输出功能。例如,触摸屏 832 和显示器 854 能够被组合在单个输入 / 输出设备中。输入设备 830 能够包括自然用户接口(NUI)。NUI 是使得用户能够以“自然的”方式与设备交互的任何接口技术,免于由诸如鼠标、键盘、遥控器等这样的输入设备所强加的人工约束。NUI 方法的例子包括依靠语音辨识、触摸和触针辨识、在屏幕上和与屏幕邻近的手势辨识、空中手势、头和眼跟踪、话音和语音、视觉、触摸、手势以及机器智能的那些。NUI 的其它例子包括使用加速度计 / 陀螺仪的运动手势检测、面部辨识、3D 显示器、头、眼以及视线跟踪、沉浸式增强现实以及虚拟现实系统,其中的全部都提供更自然的接口,以及用于使用电场感测电极来感测脑活动的技术(EEG 和有关方法)。因此,在一个特定例子中,操作系统 812 或应用 814 能够包括语音辨识软件作为允许用户经由话音命令来操作设备 800 的话音用户接口的一部分。进一步地,设备 800 能够包括允许经由用户的空间手势的用户交互(诸如检测并且解释手势以便将输入提供给游戏应用或其它应用)的输入设备和软件。

[0077] 无线调制解调器 860 能够被耦合到天线(未示出)并且能够支持处理器 810 与外部设备之间的双向通信,如本领域中很好地理解的那样。调制解调器 860 被一般性地示出,并且能够包括用于与移动通信网络 804 进行通信的蜂窝调制解调器和 / 或其它基于无线电的调制解调器(例如,蓝牙 864 或 Wi-Fi 862)。无线调制解调器 860 典型地被配置用于与一个或多个蜂窝网络通信,诸如用于在单个蜂窝网络内、在蜂窝网络之间或在移动设备与公用交换电话网(PSTN)之间进行数据和话音通信的 GSM 网络。

[0078] 移动设备能够进一步包括至少一个输入 / 输出端口 880、电源 882、卫星导航系统接收机 884(诸如全球定位系统(GPS)接收机)、加速度计 886 和 / 或物理连接器 890,所述物理连接器 890 可以是 USB 端口、IEEE 1394(火线)端口和 / 或 RS-232 端口。所图示的构

件 802 不是必需的或包括一切的,因为任何构件能够被删除并且其它构件能够被添加。

[0079] 示范性实施环境

图 9 图示了描述的实施例、技术以及工艺可以在其中被实施的适合的实施环境 900 的一般性的例子。

[0080] 在示范性环境 900 中,各种类型的服务(例如,计算服务)由云 910 来提供。例如,云 910 能够包括将基于云的服务提供给经由诸如因特网这样的网络连接的各种类型的用户和设备的计算设备的合集,所述计算设备可以被集中定位或分布。实施环境 900 能够被以不同的方式用来实现计算任务。例如,一些任务(例如,处理用户输入以及呈现用户界面)能够在本地计算设备(例如,连接的设备 930、940、950)上被执行,然而其它任务(例如,要被用在后续处理中的数据的存储)能够在云 910 中执行。

[0081] 在示范性环境 900 中,云 910 给针对连接的设备 930、940、950 的服务提供各种屏幕能力。连接的设备 930 表示具有计算机屏幕 935(例如,中等尺寸屏幕)的设备。例如,连接的设备 930 可以是诸如台式计算机、膝上型电脑、笔记本、上网本等这样的个人计算机。连接的设备 940 表示具有移动设备屏幕 945(例如,小尺寸屏幕)的设备。例如,连接的设备 940 可以是移动电话、智能电话、个人数字助理、平板计算机等。连接的设备 950 表示具有大屏幕 955 的设备。例如,连接的设备 950 可以是电视屏幕(例如,智能电视)或连接到电视的另一设备(例如,机顶盒或游戏控制台)等等。连接的设备 930、940 以及 950 中的一个或多个能够包括触摸屏能力。触摸屏能够以不同的方式接受输入。例如,当对象(例如,指尖套或触针)使跨越表面运行的电流失真或中断时电容式触摸屏检测到触摸输入。作为另一例子,触摸屏能够使用光学传感器在来自光学传感器的光束被中断时检测触摸输入。与屏幕的表面的物理接触对于要被一些触摸屏检测的输入是不必要的。没有屏幕能力的设备还能够被用在示范性环境 900 中。例如,云 910 能够为没有显示器的一个或多个计算机(例如,服务器计算机)提供服务。

[0082] 服务能够由云 910 通过服务提供商 920 或者通过在线服务的其它提供商(未描绘)提供。例如,云服务可以是对特定连接的设备(例如,连接的设备 930、940、950)的屏幕尺寸、显示能力和/或触摸屏能力定制的。

[0083] 在示范性环境 900 中,云 910 至少部分地使用服务提供商 920 和一个或多个在线提供商 925 来将本文中所描述的技术和解决方案提供给各种连接的设备 930、940、950。例如,服务提供商 920 能够为各种基于云的服务提供集中式解决方案。服务提供商 920 能够管理针对用户和/或设备(例如,针对连接的设备 930、940、950 和/或它们相应的用户)的服务订阅。云 910 能够提供资源以用于下载、发送或者接收如本文中所讨论的针对一个或多个应用的一个或多个安装包。例如,针对应用的中间语言代码能够在云 910 中由在线提供商 925 中的至少一个编译。如在 965 处所示出的,本机映像由连接的设备 930 根据从云 910 下载的 IL 代码针对应用而生成。

[0084] 示范性计算环境

图 10 描绘了所描述的创新可以在其中被实施的适合的计算环境 1000 的一般性的例子。计算环境 1000 不旨在关于使用或功能性的范围提出任何限制,因为创新可以用多种多样的通用或专用计算系统加以实施。例如,计算环境 1000 可以是各种计算设备中的任一个(例如,台式计算机、膝上型计算机、服务器计算机、平板计算机、媒体播放机、游戏系统、移

动设备等)。

[0085] 参考图 10, 计算环境 1000 包括一个或多个处理单元 1010、1015 和存储器 1020、1025。在图 10 中, 这个基本配置 1030 被包括在虚线内。处理单元 1010、1015 执行计算机可执行指令。处理单元可以是通用中央处理单元(CPU)、专用集成电路(ASIC)中的处理器或任何其它类型的处理器。在多处理系统中, 多个处理单元执行计算机可执行指令以便增加处理能力。例如, 图 10 示出了中央处理单元 1010 以及图形处理单元或协处理单元 1015。有形存储器 1020、1025 可以是可被(一个或多个)处理单元访问的易失性存储器(例如, 寄存器、高速缓存、RAM)、非易失性存储器(例如, ROM、EEPROM、闪速存储器等)或两者的某种组合。存储器 1020、1025 以适合于由(一个或多个)处理单元执行的计算机可执行指令的形式存储软件 1080, 其实施本文中所描述的一个或多个创新。

[0086] 计算系统可以具有附加的特征。例如, 计算环境 1000 包括存储装置 1040、一个或多个输入设备 1050、一个或多个输出设备 1060 以及一个或多个通信连接 1070。诸如总线、控制器或网络这样的互连机制(未示出)互连计算环境 1000 的构件。典型地, 操作系统软件(未示出)为在计算环境 1000 中执行的其它软件提供操作环境, 并且协调计算环境 1000 的构件的活动。

[0087] 有形存储装置 1040 可以是可拆卸的或非可拆卸的, 并且包括磁盘、磁带或磁盒、CD-ROM、DVD, 或能够被用来以非暂时性方式存储信息并且能够在计算环境 1000 内被访问的任何其它介质。存储装置 1040 存储针对软件 1080 的指令, 所述软件 1080 实施诸如从 IL 代码为一个或多个应用生成本机代码这样的本文中所描述的一个或多个创新。

[0088] (一个或多个)输入设备 1050 可以是诸如键盘、鼠标、笔或轨迹球这样的触摸输入设备、话音输入设备、扫描设备, 或将输入提供给计算环境 1000 的另一设备。对于视频编码, (一个或多个)输入设备 1050 可以是相机、视频卡、TV 调谐器卡或以模拟形式或数字形式接受视频输入的类似设备、或将视频样本读取到计算环境 1000 中的 CD-ROM 或 CD-RW。(一个或多个)输出设备 1060 可以是显示器、打印机、扬声器、CD 刻录器、或从计算环境 1000 提供输出的另一设备。

[0089] (一个或多个)通信连接 1070 使能实现通过通信介质到另一计算实体的通信。通信介质在已调制数据信号中输送诸如计算机可执行指令、音频或视频输入或输出或其它数据这样的信息。已调制数据信号是这样的信号, 所述信号使其特性中的一个或多个以这样的方式被设置或改变以便将信息编码在所述信号中。作为例子而非限制, 通信媒体能够使用电、光学、RF 或其它载体。

[0090] 尽管为了方便呈现所公开的方法中的一些的操作被按照特定有顺序的次序描述, 但是应理解, 除非特定排序是在下面所阐述的特定语言所要求的, 否则这个描述的方式包含重新布置。例如, 顺序地描述的操作在一些情况下可以被重新布置或者同时执行。而且, 为了简单起见, 附图可能不示出所公开的方法以其能够与其它方法相结合地使用的各种方式。

[0091] 所公开的方法中的任一个能够作为存储在一个或多个计算机可读存储媒体(例如, 非暂时性计算机可读媒体, 诸如一个或多个光学媒体盘、易失性存储器构件(诸如 DRAM 或 SRAM)或非易失性存储器构件(诸如闪速存储器或硬盘驱动器))上并且在计算机(例如, 任何在商业上可获得的计算机, 包括智能电话或其它包括计算硬件的移动设备)上执行的

计算机可执行指令被实施。如应该容易地理解的,术语计算机可读存储媒体不包括通信连接性,诸如已调制数据信号。用于实施所公开的技术的计算机可执行指令以及在所公开的实施例的实施期间创建和使用的任何数据中的任一个能够被存储在一个或多个计算机可读媒体(例如,非暂时性计算机可读媒体,其不包括传播信号)上。计算机可执行指令可以是专用软件应用或例如经由 web 浏览器或其它软件应用(诸如远程计算应用)被访问或者下载的软件应用的一部分。这样的软件能够例如在单个本地计算机(例如,任何适合的在商业上可获得的计算机)上或在使用一个或多个网络计算机的网络环境(例如,经由因特网、广域网、局域网、客户端-服务器网络(诸如云计算网络)或其它这样的网络)中被执行。

[0092] 为了清楚,仅基于软件的实施方案的特定选择的方面被描述。在本领域内众所周知的其它细节被省略。例如,应理解,所公开的技术不限于任何特定计算机语言或程序。例如,所公开的技术能够通过用 C++、C#、J++、Java、Perl、JavaScript、Adobe Flash 或任何其它适合的编程语言编写的软件被实施。同样地,所公开的技术不限于任何特定计算机或特定类型的硬件。适合的计算机和硬件的特定细节是众所周知的并且不必在本公开内容中详细地阐述。

[0093] 还应该很好地理解,本文中所描述的任何功能性能能够至少部分地由一个或多个硬件逻辑构件而不是软件来执行。例如,但非限制,能够被使用的说明性类型的硬件逻辑构件包括现场可编程门阵列(FPGA)、特定程序集成电路(ASIC)、特定程序标准产品(ASSP)、片上系统型系统(SOC)、复杂可编程逻辑器件(CPLD)等。

[0094] 此外,基于软件的实施例(包括例如用于使计算机执行所公开的方法中的任一个的计算机可执行指令)中的任一个能够通过适合的通信装置被上传、下载或远程地访问。这样的适合的通信装置例如包括因特网、万维网、内联网、软件应用、电缆(包括光纤电缆)、磁通信、电磁通信(包括 RF、微波以及红外通信)、电子通信,或其它这样的通信装置。

[0095] 所公开的方法、装置以及系统不应该被解释为以任何方式限制。替代地,本公开内容单独地或在与彼此的各种组合和子组合中针对各种公开的实施例的所有新颖和非显而易见的特征和方面。所公开的方法、装置以及系统不限于任何特定方面或特征或其组合,所公开的实施例也不要求存在任何一个或多个特定优点或者问题被解决。鉴于所公开的发明的原理可以被应用于的许多可能的实施例,应认识到,所图示的实施例仅是本发明的优选例子,并且不应该被视为限制本发明的范围。相反地,本发明的范围由以下权利要求定义。我们因此要求落入这些权利要求及其等同物的范围内的全部作为我们的发明。

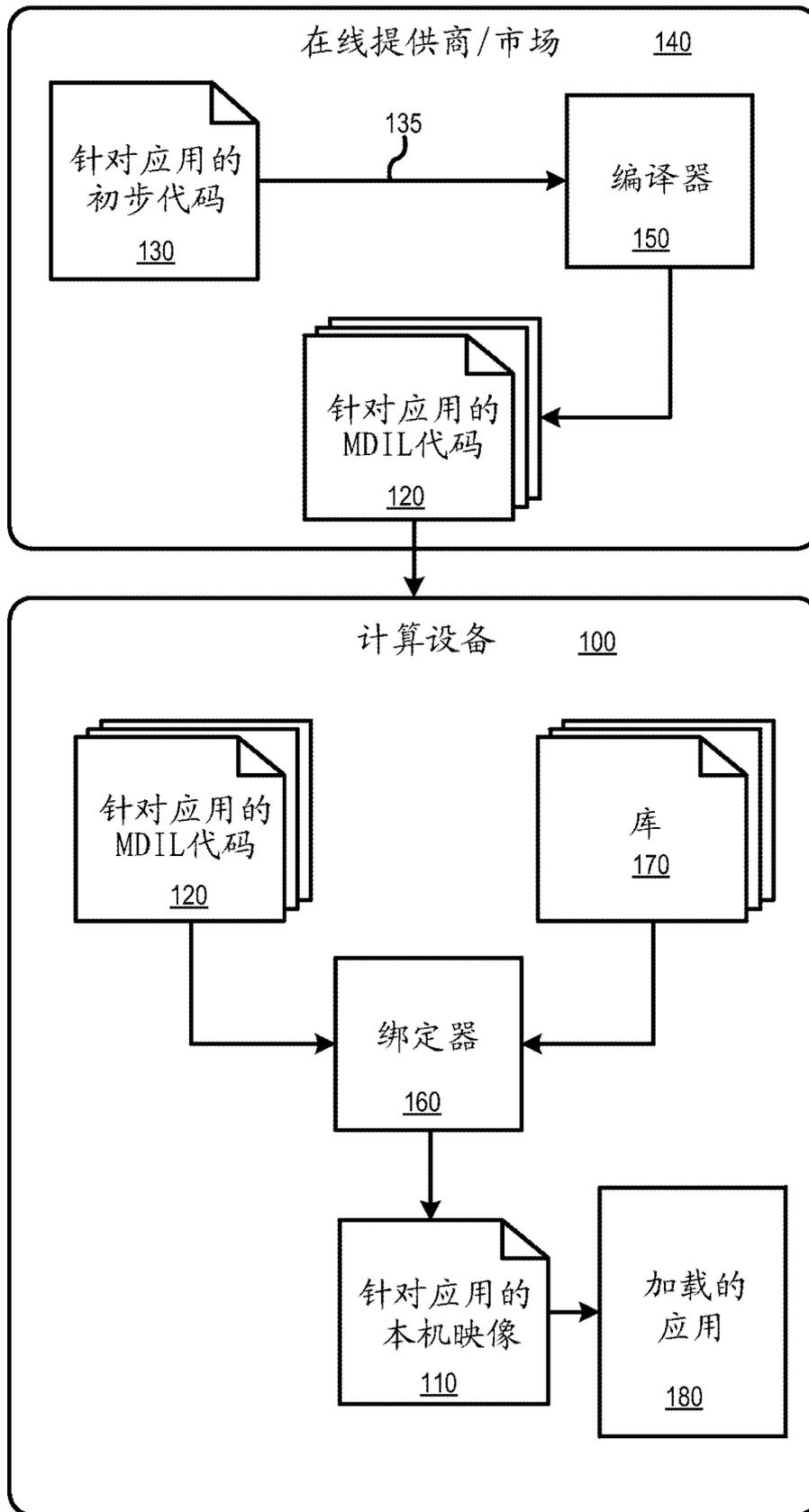


图 1

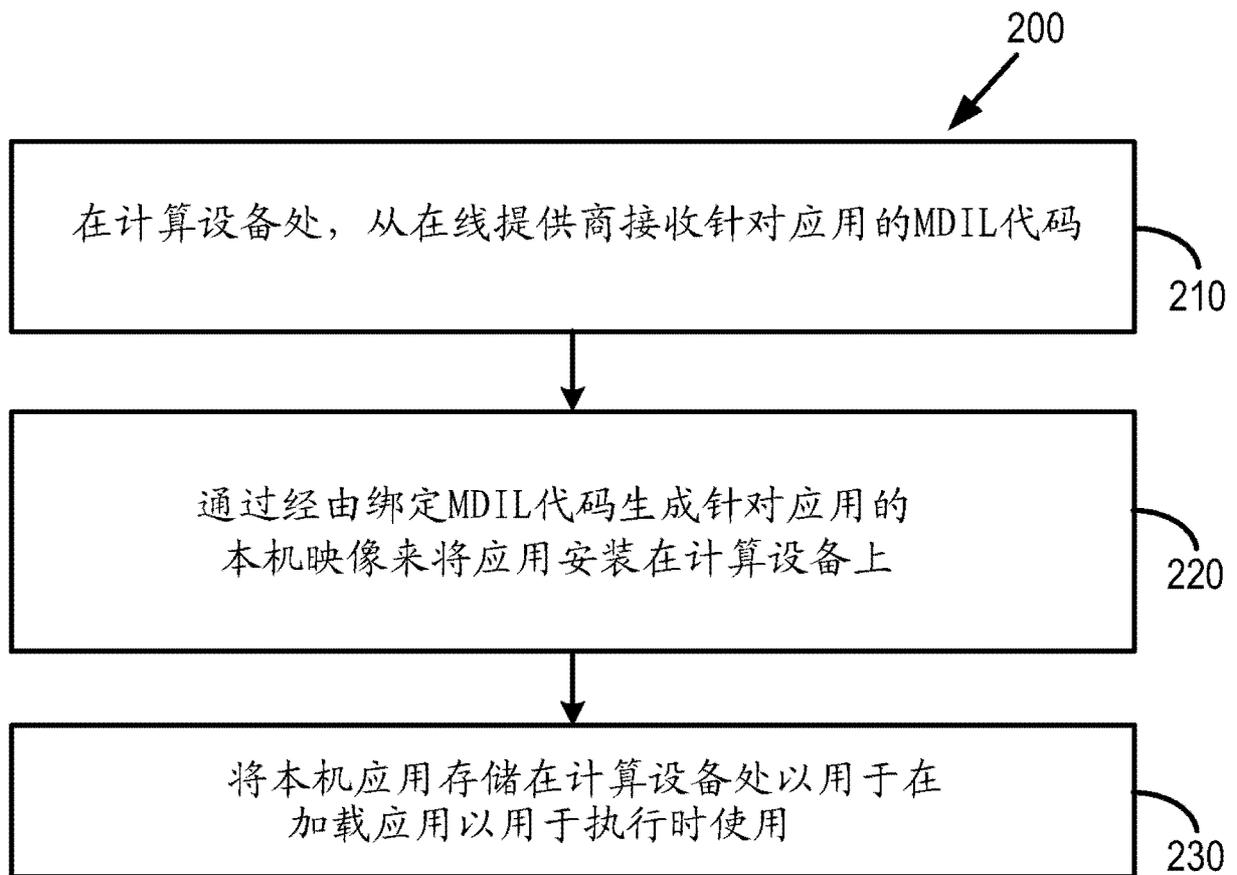


图 2

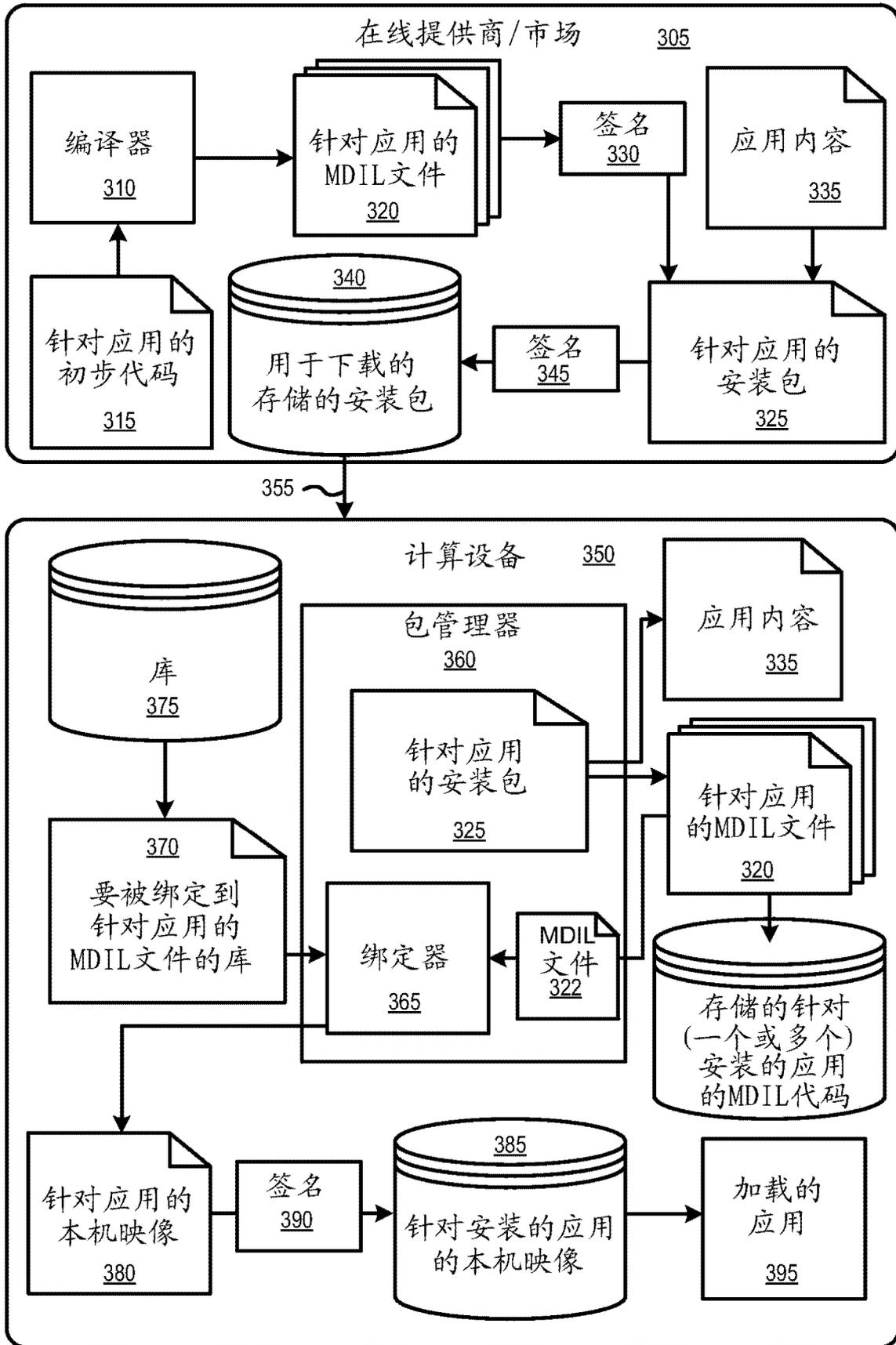


图 3

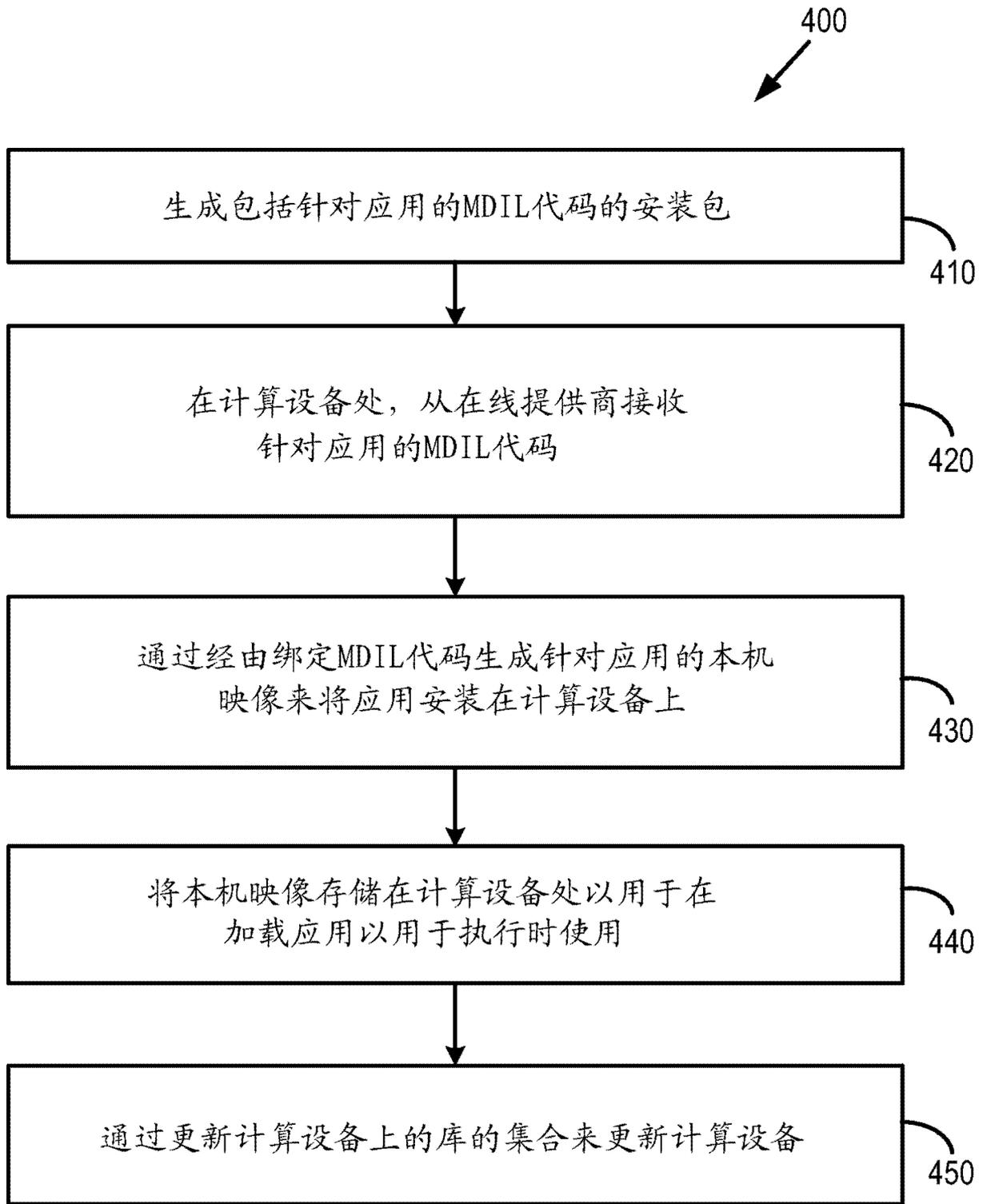


图 4

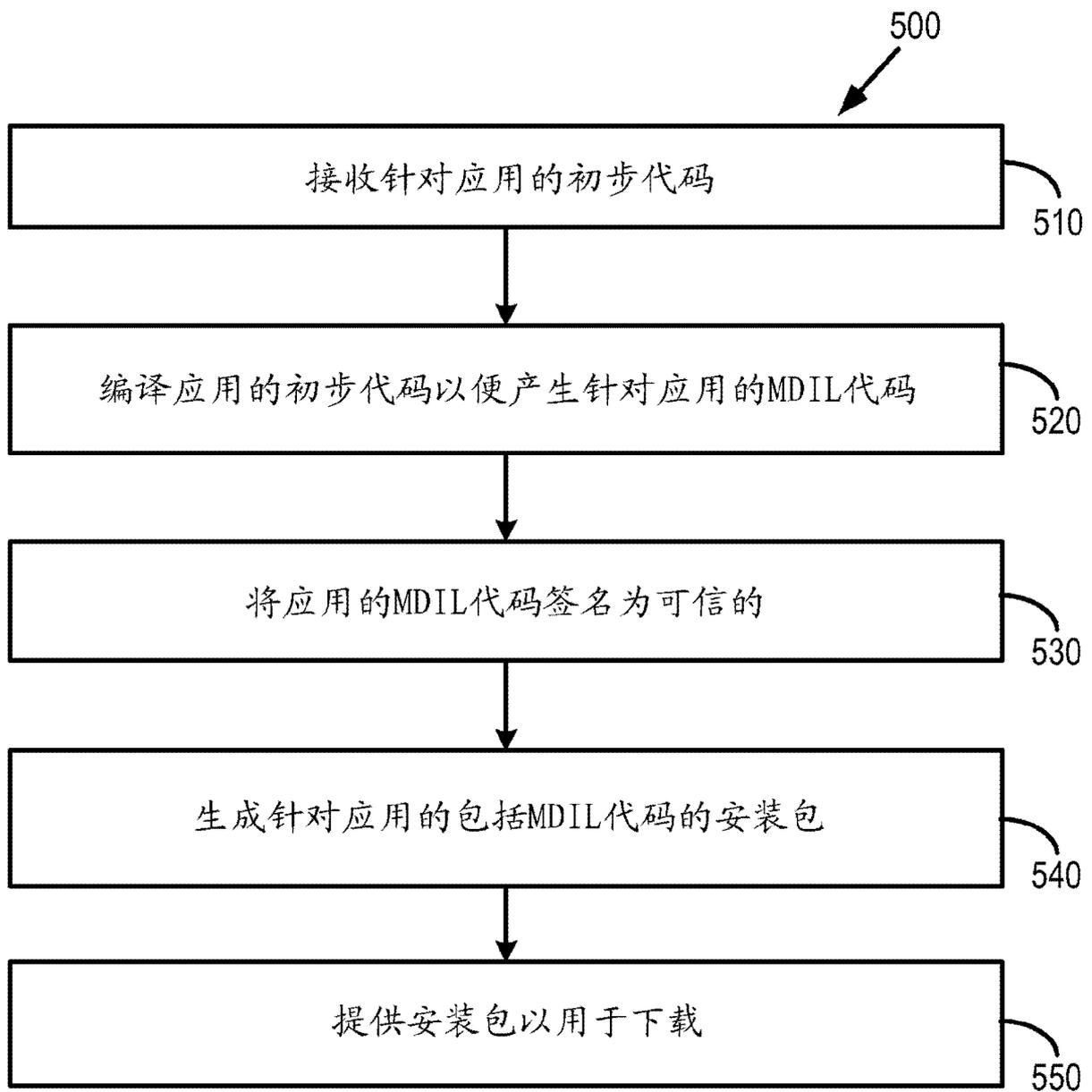


图 5

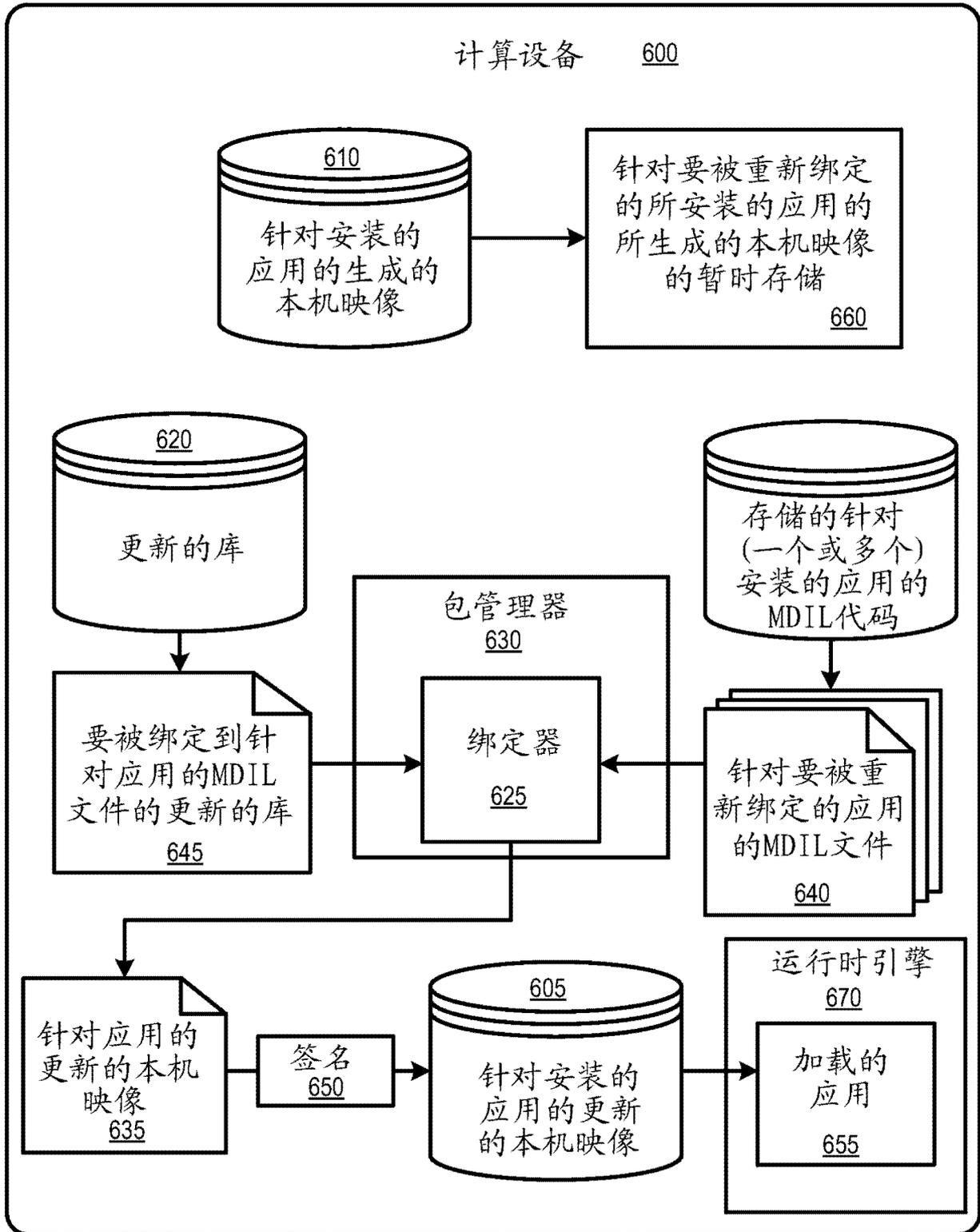


图 6

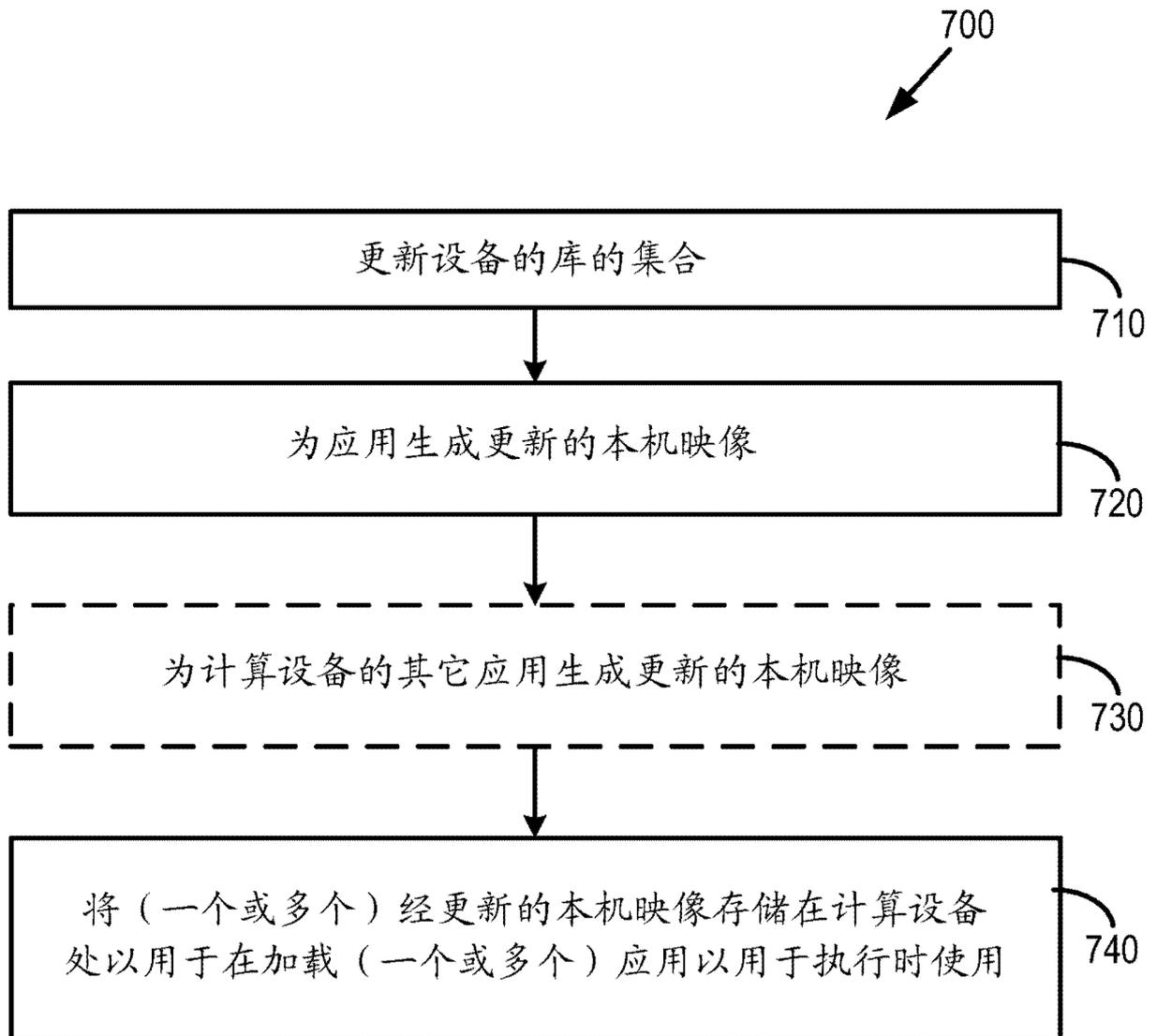


图 7

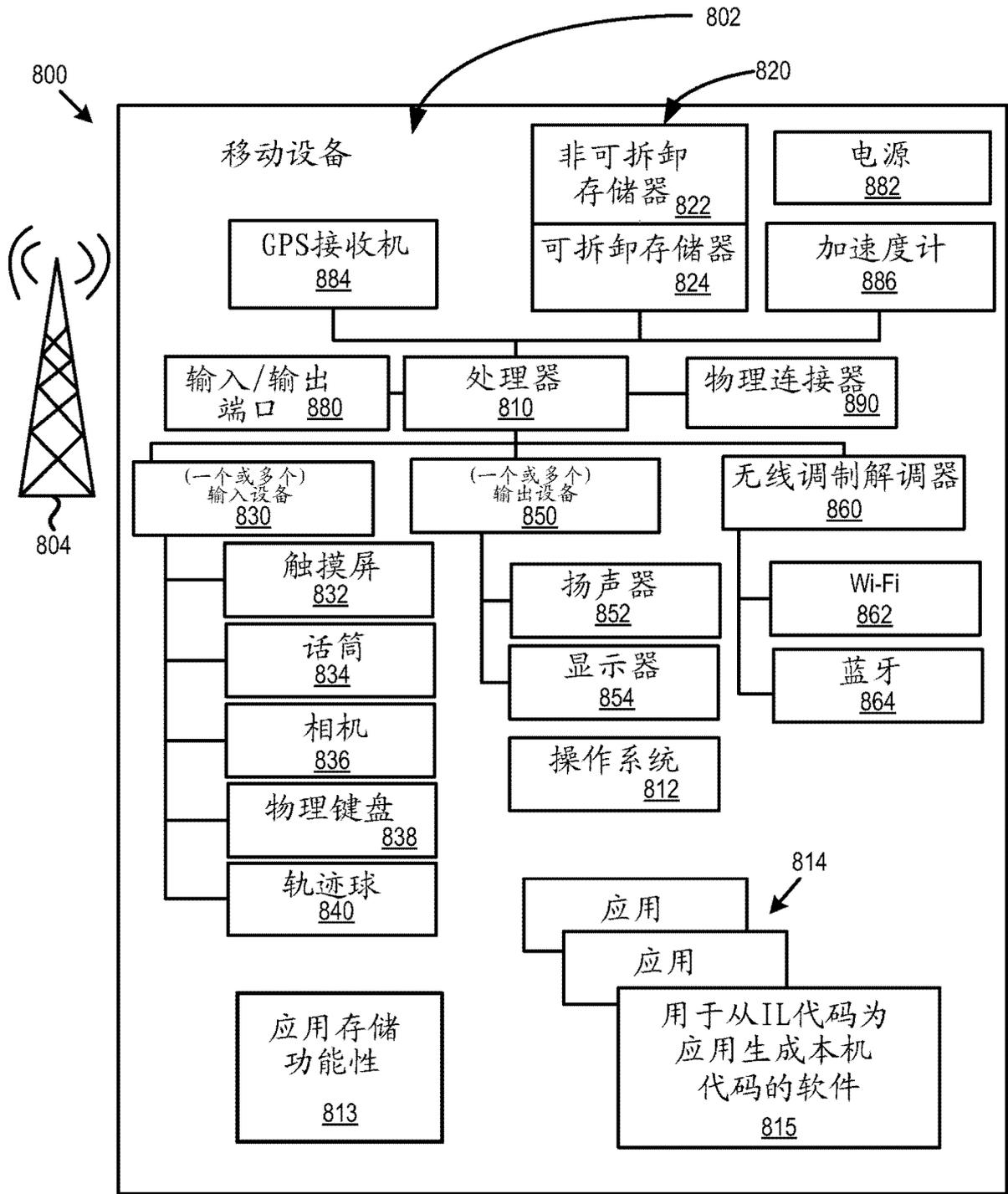


图 8

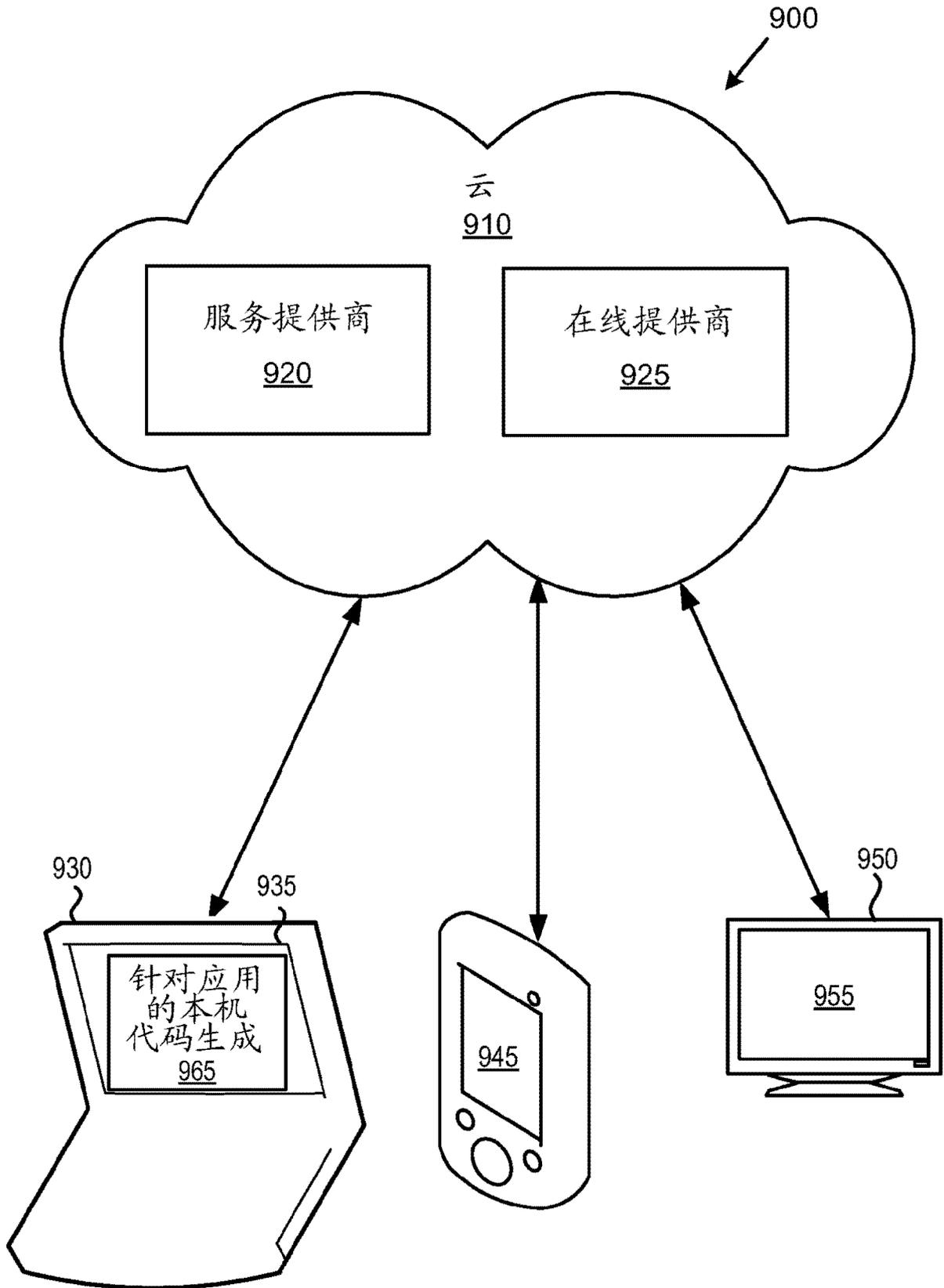
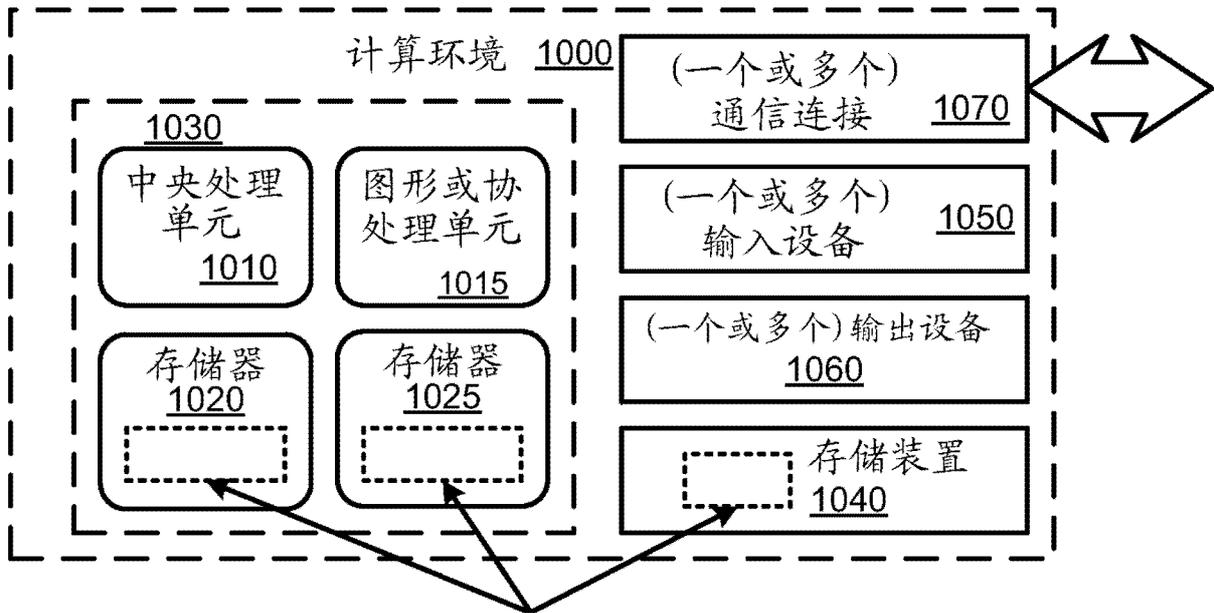


图 9



软件1080，其实施用于从IL代码为应用生成本机代码的描述的技术

图 10