(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 104598217 B
(45)授权公告日 2018.06.29

(21)申请号 201410601088.2

(22)申请日 2014.10.30

(65)同一申请的已公布的文献号
申请公布号 CN 104598217 A

(43)申请公布日 2015.05.06

(30)优先权数据
2831711 2013.10.31 CA

(73)专利权人 国际商业机器公司
地址 美国纽约

(72)发明人 M·米垂恩 V·沃克舒利

(74)专利代理机构 中国国际贸易促进委员会专利商标事务所 11038
代理人 鲍进

(51)Int.Cl.
G06F 8/41(2018.01)

(56)对比文件
CN 101438529 A,2009.05.20,
US 2010058303 A1,2010.03.04,
US 2013086563 A1,2013.04.04,
US 7996671 B2,2011.08.09,
US 7747989 B1,2010.06.29,

审查员 卢素斋

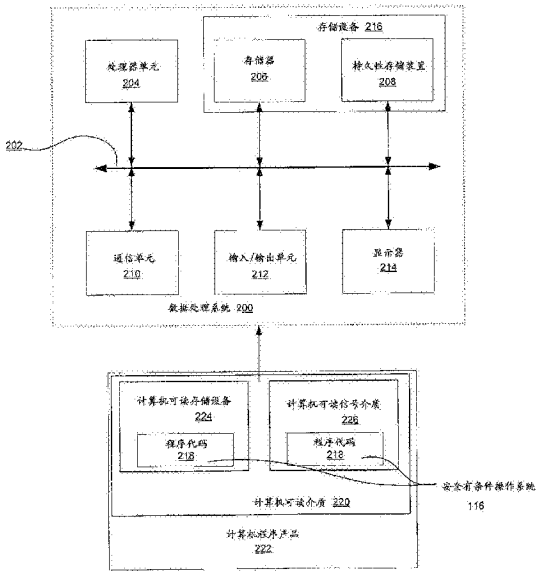
权利要求书2页 说明书11页 附图5页

(54)发明名称

用于执行安全有条件加载和有条件存储操作的方法和设备

(57)摘要

本发明公开涉及用于执行安全有条件加载和有条件存储操作的方法和设备。提供了一种用于在存储访问不能被证明是安全的时进行安全有条件操作的计算机实现的方法的说明性实施例包括由增强编译器接收用于事务的部分源代码，由所述增强编译器分析接收到的所述部分源代码以确定由所述增强编译器分析的所述部分源代码是否是用于变换的候选。响应于确定由所述增强编译器分析的所述部分源代码是用于变换的候选，变换被分析的所述部分源代码以在变换后代码的第一部分中使用有条件操作，其中相应的有条件操作使用硬件事务存储器在硬件中调用重试操作，及在变换后代码的第二部分中添加指向原始代码部分的分支，其中所述分支的代码是包含所述原始代码部分的恢复部分。



CN 104598217 B

1.一种用于在存储访问不能被证明是安全的时进行安全有条件操作的计算机实现的方法，所述计算机实现的方法包括：

由增强编译器接收用于事务的部分源代码；

由所述增强编译器分析接收到的所述部分源代码；

确定由所述增强编译器分析的所述部分源代码是否是用于变换的候选；

响应于确定由所述增强编译器分析的所述部分源代码是用于变换的候选，变换被分析的所述部分源代码以在变换后代码的第一部分中使用有条件操作，其中相应的有条件操作使用硬件事务存储器在硬件中调用重试操作，其中变换被分析的所述部分源代码以在变换后代码的第一部分中使用有条件操作还包括在所述变换后代码部分的第一部分中规定恢复例程；

在变换后代码的第二部分中添加指向原始代码部分的分支，其中所述分支的代码是包含所述原始代码部分的恢复部分；以及

执行包括变换后代码的所述第一部分和变换后代码的所述第二部分的事务。

2.如权利要求1所述的计算机实现的方法，其中变换后代码的所述第二部分包括到恢复例程的分支以供在所述事务内推测性执行的变换后代码的所述第一部分中的有条件操作失败时使用，所述方法还包括：

为所述有条件操作设置条件代码；

在所述有条件操作中的规定条件不为真时，利用所述有条件操作把位于第一存储位置的值加载到第二存储位置中；

当所述事务失败时，分支离开所述事务到所述恢复例程；

确定所述分支是否被预测得不好；

响应于确定分支不是被预测得不好，执行所述原始代码部分；及

响应于确定分支是被预测得不好，分支离开所述恢复例程。

3.如权利要求1所述的计算机实现的方法，还包括：

当所述变换后代码的有条件操作失败时，使用所述原始代码部分作为所述分支的目标，其中所述恢复部分使得能够从所述第一部分的失败的有条件操作得体地离开。

4.如权利要求2所述的计算机实现的方法，其中为所述有条件操作设置所述条件代码还包括：

保留原始源代码部分的条件设置代码。

5.如权利要求2所述的计算机实现的方法，其中，响应于确定分支是被预测得不好，分支离开所述恢复例程还包括：

使用*ptr作为所述有条件操作的自变量。

6.如权利要求1所述的计算机实现的方法，其中变换被分析的所述部分源代码还包括：

变换后代码在第一部分中包括所述变换后代码的有条件操作代码部分以代替无条件操作，并且在第二部分中包括相应的恢复段，所述恢复段包括与所述有条件操作代码部分相关联的原始分支代码。

7.一种用于在存储访问不能被证明是安全的时进行安全有条件操作的设备，所述设备包括：

通信架构；

连接到所述通信架构的存储器，其中所述存储器包含计算机可执行程序代码；

连接到所述通信架构的通信单元；

连接到所述通信架构的输入/输出单元；

连接到所述通信架构的显示器；及

连接到所述通信架构的处理器单元，其中所述处理器单元执行所述计算机可执行程序代码以指示所述设备：

由增强编译器接收用于事务的部分源代码；

由所述增强编译器分析接收到的所述部分源代码；

确定由所述增强编译器分析的所述部分源代码是否是用于变换的候选；

响应于确定由所述增强编译器分析的所述部分源代码是用于变换的候选，变换被分析的所述部分源代码以在变换后代码的第一部分中使用有条件操作，其中相应的有条件操作使用硬件事务存储器在硬件中调用重试操作，其中变换被分析的所述部分源代码以在变换后代码的第一部分中使用有条件操作还包括在所述变换后代码部分的第一部分中规定恢复例程；

在变换后代码的第二部分中添加指向原始代码部分的分支，其中所述分支的代码是包含所述原始代码部分的恢复部分；以及

执行包括变换后代码的所述第一部分和变换后代码的所述第二部分的事务。

8.如权利要求7所述的设备，其中变换后代码的所述第二部分包括到恢复例程的分支以供在所述事务内推测性执行的变换后代码的所述第一部分中的有条件操作失败时使用，所述所述处理器单元执行所述计算机可执行程序代码以进一步指示所述设备：

为所述有条件操作设置条件代码；

在所述有条件操作中的规定条件不为真时，利用所述有条件操作把位于第一存储位置的值加载到第二存储位置中；

当所述事务失败时，分支离开所述事务到所述恢复例程；

确定所述分支是否被预测得不好；

响应于确定分支不是被预测得不好，执行所述原始代码部分；及

响应于确定分支是被预测得不好，分支离开所述恢复例程。

9.如权利要求7所述的设备，其中所述处理器单元执行所述计算机可执行程序代码以进一步指示所述设备：

当所述变换后代码的有条件操作失败时，使用所述原始代码部分作为所述分支的目标，其中所述恢复部分使得能够从所述第一部分的失败的有条件操作得体地离开。

10.如权利要求8所述的设备，其中所述处理器单元执行所述计算机可执行程序代码以进一步指示所述设备：

保留原始源代码部分的条件设置代码。

11.如权利要求8所述的设备，其中所述处理器单元执行用于响应于确定分支是被预测得不好、分支离开所述恢复例程的计算机可执行程序代码以进一步指示所述设备：

使用*ptr作为所述有条件操作的自变量。

用于执行安全有条件加载和有条件存储操作的方法和设备

技术领域

[0001] 本公开一般关于涉及数据处理系统中有条件加载(conditional-load)和有条件存储(conditional-store)操作的程序代码实例，并且尤其是关于在数据处理系统中不能证明存储访问(storage access)是安全的条件下执行安全的有条件加载和有条件存储操作。

背景技术

[0002] LOAD_ON_COND和STORE_ON_COND操作形式的有条件操作通常为对程序代码编译器的分支预测器来说难以可靠预测的数据驱动的分支提供显著的性能优势。在有些情况下，LOAD_ON_COND和STORE_ON_COND操作会对程序的性能提供相当大的改进，这是因为与预测失误的分支相关联的损失被除去了。

[0003] 例如，利用以下代表LOAD_ON_COND操作的常见代码模式的代码片段：

```
x = 0;
if (a > 10)                          ## 数据驱动的并且预测得不好
```

[0004]
```
{
x = *ptr;
}
```

[0005] 该代码片段的典型实现是：

```
LOAD_IMM      Rx, 0          ## x=0
COMPARE       Ra, 10         ## Cond <= (Ra > 10)
```

[0006]
```
BRANCH        GT, Label      ## 分支被预测得不好
LOAD          Rx, 0(Rptr)    ## 把*ptr 加载到 X 中
Label:
```

[0007] 该典型实现可以进一步被变换成简化形式：

[0008] LOAD_IMM Rx,0 ##x＝0

[0009] COMPARE Ra,10; ##

[0010] LOAD_ON_COND Rx,GT,0(Rptr) ##如果Ra>10，就把*ptr加载到X中 由此，该变换通过把*ptr指定给x的预测而避免了分支预测。

[0011] 但是，对于变换的使用，存在重要的限制。独立于有条件操作的结果，LOAD_ON_COND只能在加载的源被证明不造成非法访问(access violation)的情形下被安全地使用。LOAD_ON_COND的实现通常独立于相应的条件而隐含地执行加载操作，并且因此即使在加载的相应条件为假时都造成非法访问。例如，在以下实现中：

[0012] LOAD_IMM Rt,1

[0013] COMPARE Rptr,NULL ##设置条件代码

[0014] LOAD_ON_COND Rt,NE,0(Rptr) ##对于条件代码NE，把存储位置0

[0015] 的值(Rptr)加载到Rt中

[0016] 当Rptr为NULL时，非法访问将在页面0不可读的系统上发生。

[0017] 因此，为了让LOAD_ON_COND变换正确地工作，编译器必须具有足够多关于*ptr的上下文信息，以便独立于相应的加载条件为真还是为假而证明加载的源将不造成非法访问异常。

[0018] 以类似的方式，STORE_ON_COND只能在其中存储操作要对其执行的位置不造成非法访问的情形下被安全地使用。例如，利用以下实现：

[0019] MOVE 0(array),H'0'

[0020] COMPARE Ra,10

[0021] STORE_ON_COND Rx,GT,0(array)

[0022] 当数组的地址无效时，非法访问将在使用该示例实现的系统上发生。

[0023] 有些之前尝试过的解决办法使用硬件事务存储器，作为检测对物理存储器的坏访问的机制。在之前尝试过的解决办法的另一个例子中，使用其中在区域中检查null(空值)的推测性消除(speculative elimination)的技术(可以在http://pharm.ece.wisc.edu/papers/vee10.pdf获得)。但是，该技术只在存在虚拟存储器子系统，诸如在Java虚拟机的实现中用来检测null引用，的时候有用。在另一个之前尝试过的解决办法中，新的硬件逻辑被用来改进分支预测。

发明内容

[0024] 根据一种实施例，用于在存储访问不能被证明是安全的时进行安全有条件操作的计算机实现的方法的说明性实施例的计算机实现的过程由增强编译器接收用于事务的部分源代码，由所述增强编译器分析接收到的所述部分源代码以确定由所述增强编译器分析的所述部分源代码是否是用于变换的候选。响应于确定由所述增强编译器分析的所述部分源代码是用于变换的候选，变换被分析的所述部分源代码以在变换后代码的第一部分中使用有条件操作，其中相应的有条件操作使用硬件事务存储器在硬件中调用重试操作，及在变换后代码的第二部分中添加指向原始代码部分的分支，其中所述分支的代码是包含所述原始代码部分的恢复部分。

[0025] 根据另一种实施例，用于在存储访问不能被证明是安全的时进行安全有条件操作的计算机程序产品包括计算机可记录类型的介质，该介质包含存储在其上的计算机可执行程序代码。计算机可执行程序代码包括用于由增强编译器接收用于事务的部分源代码的计算机可执行程序代码；用于由所述增强编译器分析接收到的所述部分源代码的计算机可执行程序代码；用于确定由所述增强编译器分析的所述部分源代码是否是用于变换的候选的计算机可执行程序代码；用于响应于确定由所述增强编译器分析的所述部分源代码是用于变换的候选，变换被分析的所述部分源代码以在变换后代码的第一部分中使用有条件操作的计算机可执行程序代码，其中相应的有条件操作使用硬件事务存储器在硬件中调用重试操作，以及用于在变换后代码的第二部分中添加指向原始代码部分的分支的计算机可执行程序代码，其中所述分支的代码是包含所述原始代码部分的恢复部分。

[0026] 根据另一种实施例，一种用于在存储访问不能被证明是安全的时进行安全有条件操作的设备，所述设备包括：通信架构；连接到所述通信架构的存储器，其中所述存储器包含计算机可执行程序代码；连接到所述通信架构的通信单元；连接到所述通信架构的输入/输出单元；连接到所述通信架构的显示器；及连接到所述通信架构的处理器单元。所述处理

器单元执行所述计算机可执行程序代码以指示所述设 备：由增强编译器接收用于事务的部分源代码并由所述增强编译器分析接收到的所述部分源代码以确定由所述增强编译器分析的所述部分源代码是否是用于变换的候选。所述处理器单元执行所述计算机可执行程序代码以响应于确定由所述增强编译器分析的所述部分源代码是用于变换的候选，指示所述设备变换被分析的所述部分源代码以在变换后代码的第一部分中使用有条件操作，其中相应的有条件操作使用硬件事务存储器在硬件中调用重试操作，并且在变换后代码的第二部分中添加指向原始代码部分的分支，其中所述分支的代码是包含所述原始代码部分的恢复部分。

附图说明

[0027] 为了更完整地理解本公开，现在结合附图和具体描述来参考以下简要描述，其中相同的标号代表相同的部分。

[0028] 图1是可操作用于本公开的各种实施例的示例性网络数据处理系统的框图；

[0029] 图2是可操作用于本公开的各种实施例的示例性数据处理系统的框图；

[0030] 图3是可操作用于本公开的各种实施例的安全有条件操作系统的框图；

[0031] 图4是包括用在本公开的各种实施例中的有条件操作的事务的代码片段表示；

[0032] 图5是用于用在本公开的各种实施例中的有条件操作的优化范围的表示；

[0033] 图6是生成用在本公开的各种实施例中的安全有条件操作代码部分的过程的流程图；及

[0034] 图7是使用用在本公开的各种实施例中的安全有条件操作代码部分的过程的流程图。

具体实施方式

[0035] 虽然以下提供了一种或多种实施例的说明性实现，但是所公开的系统和/或方法可以利用任何数量的技术来实现。本公开不应当以任何方式局限于以下说明的说明性实现、附图和技术，包括本文所说明和描述的示例性设计和实现，而是可以在所附权利要求连同其等价物的完全范围内被修改。

[0036] 如本领域技术人员将认识到的，本公开的各方面可以体现为系统、方法或计算机程序产品。因此，本公开的各方面可以采取完全硬件实施例、完全软件实施例(包括固件、驻留软件、微代码等)，或者组合软件和硬件方面的实施例的形式，这些在本文中一般都可以被称为“电路”、“模块”或“系统”。此外，本发明的各方面还可以采取在一个或多个计算机可读介质中体现的计算机程序产品的形式，该计算机可读介质上包含计算机可读的程序代码。

[0037] 可以利用一个或多个计算机可读数据存储设备的任意组合。计算机可读数据存储设备可以是例如——但不限于——电、磁、光或半导体系统、装置或设备，或者以上的任意组合，但是不涵盖传播介质。计算机可读数据存储设备的更具体的例子(非穷举列表)包括以下：便携式计算机盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦除可编程只读存储器(EPROM或闪存)、便携式紧凑磁盘只读存储器(CDROM)、光存储设备、或磁存储设备、或者上述的任意合适组合，但是不涵盖传播介质。在本文档的上下文中，计算机可读数据存

储设备可以是能够存储程序的任何有形设备，该程序可以被指令执行系统、装置或者设备使用或者与其结合使用。

[0038] 用于执行本公开各方面的操作的计算机程序代码可以用一种或多种编程语言的任意组合来编写，所述编程语言包括面向对象的编程语言—诸如Java®、Smalltalk、C++等，还包括常规的过程式编程语言—诸如"C"编程语言或类似的编程语言。Java和所有基于Java的商标和徽标都是位于美国、其它国家或者同时位于这两个地方的Oracle公司和/或其分支机构的商标。程序代码可以完全地在用户计算机上执行、部分地在用户计算机上执行、作为一个独立的软件包执行、部分在用户计算 机上并且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。在涉及远程计算机的情形中，远程计算机可以通过任意种类的网络——包括局域网(LAN)或广域网(WAN)—连接到用户计算机，或者，可以连接到外部计算机(例如利用因特网服务提供商来通过因特网连接)。

[0039] 下面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说明和/或框图来描述本公开的各方面。应当理解，流程图说明和/或框图的每个方框以及流程图说明和/或框图中各方框的组合都可以由计算机程序指令实现。

[0040] 这些计算机程序指令可以提供给通用计算机、专用计算机或其它可编程数据处理装置的处理器，从而生产出一种机器，使得经计算机或其它可编程数据处理装置的处理器执行的这些指令产生实现流程图和/或框图的方框中规定的功能/动作的装置。

[0041] 也可以把这些计算机程序指令存储在能使得计算机或其它可编程数据处理装置以特定方式工作的计算机可读数据存储设备中，使得存储在计算机可读数据存储设备中的指令产生出一个包括实现流程图和/或框图中的方框中规定的功能/动作的指令的制造品。

[0042] 也可以把计算机程序指令加载到计算机或其它可编程数据处理装置上，使得在计算机或其它可编程数据处理装置上执行一系列操作步骤，以产生计算机实现的过程，从而使得在计算机或其它可编程装置上执行的指令能够提供实现流程图和/或框图中的方框中规定的功能/动作的过程。

[0043] 现在参照附图并且尤其是参考图1-2，提供了其中可以实现说明性实施例的数据处理环境的示例性图。应当认识到，图1-2仅仅是示例性的并且不是要断言或暗示关于其中可以实现不同实施例的环境的任何限制。可以对所绘出的实施例进行许多修改。

[0044] 图1绘出了其中可以实现说明性实施例的数据处理系统的网络的图示表示。网络数据处理系统100是其中可以实现说明性实施例的计算机的网络。网络数据处理系统100包含网络102，这是用来在网络数据处 理系统100中连接到一起的各种设备和计算机之间提供通信链路的介质。网络102可以包括连接，诸如有线、无线通信链路或者光纤电缆。

[0045] 在所绘出的例子中，服务器104和服务器106连同存储装置108一起连接到网络102。此外，客户端110连接到网络102。客户端110可以是例如个人计算机或网络计算机。在所绘出的例子中，服务器104向包括安全有条件操作系统116的客户端110提供数据，诸如引导文件、操作系统图像和应用。在这个例子中，客户端110是服务器104的客户端。网络数据处理系统100可以包括未示出的附加服务器、客户端和其它设备。安全有条件操作系统116是本公开的代表性实施例，提供了在网络数据处理系统100中存储访问不能被证明是安全的时利用服务器104执行安全有条件加载和有条件存储的能力。

[0046] 在所绘出的例子中，网络数据处理系统100是互联网，其中网络102代表使用传输

控制协议/互联网协议(TCP/IP)协议套件彼此通信的网络和网关的全球集合。在互联网的中心，是主要节点或主机之间的高速数据通信链路的主干，包括数以千计的路由数据和消息的商业、政府、教育和其它计算机系统。当然，网络数据处理系统100还可以实现为多种不同类型的网络，诸如像内联网、局域网(LAN)或者广域网(WAN)。图1是要作为例子，而不是作为对不同说明性实施例的体系架构限制。

[0047] 参考图2，给出了可操作用于本公开的各种实施例的示例性数据处理系统的框图。在这个说明性例子中，数据处理系统200包括通信架构202，该架构202提供了处理器单元204、存储器206、持久性存储装置208、通信单元210、输入/输出(I/O)单元212以及显示器214之间的通信。

[0048] 处理器单元204用来执行可加载到存储器206中的软件的指令。依赖于特定的实现，处理器单元204可以是一个或多个处理器的集合或者可以是多处理器内核。另外，处理器单元204可以利用一个或多个异质处理器系统来实现，其中主处理器与二级处理器存在于单个芯片上。作 为另一个说明性例子，处理器单元204可以是包含相同类型的多个处理器的对称多处理器系统。

[0049] 存储器206和持久性存储装置208是存储设备216的例子。存储设备是能够存储信息的任何硬件，其中信息诸如像，但不限于，暂时性和/或持久性的数据、函数形式的程序代码和/或其它合适的信息。在这些例子中，存储器206可以是例如随机存取存储器或者任何其它合适的易失性或非易失性存储设备。依赖于特定的实现，持久性存储装置208可以采取各种形式。例如，持久性存储装置208可以包括一个或多个部件或设备。例如，持久性存储装置208可以是硬驱、闪存存储器、可重写光盘、可重写磁带，或者以上的某种组合。由持久性存储装置208使用的介质也可以是可移除的。例如，移动硬驱可以用于持久性存储装置208。

[0050] 在这些例子中，通信单元210提供与其它数据处理系统或设备的通信。在这些例子中，通信单元210是网络接口卡。通信单元210可以通过物理和无线通信链路当中任意一种或者这二者来提供通信。

[0051] 输入/输出单元212允许利用可以连接到数据处理系统200的其它设备进行数据的输入和输出。例如，输入/输出单元212可以通过键盘、鼠标和/或某种其它合适的输入设备提供用于用户输入的连接。另外，输入/输出单元212可以把输出发送到打印机。显示器214提供了向用户显示信息的机制。

[0052] 用于操作系统、应用和/或程序的指令可以位于存储设备216中，其中存储设备216通过通信架构202与处理器单元204通信。在这些说明性例子中，指令是以持久性存储装置208上函数的形式。这些指令可以加载到存储器206中，由处理器单元204执行。不同实施例的过程可以由处理器单元204利用计算机实现的指令来执行，这些指令可以位于存储器，诸如存储器206，当中。计算机实现的指令还包含包括要加载到存储器206中以便由处理器单元204执行的安全有条件操作系统116的指令。

[0053] 这些指令被称为可以被处理器单元204中的处理器读取并执行的程 序代码、计算机可用程序代码或者计算机可读程序代码。在不同实施例中，程序代码可以体现在不同的物理或有形计算机可读存储介质，诸如存储设备216的存储器206或者持久性存储装置208，上。

[0054] 程序代码218以函数形式位于计算机可读介质220上，其中程序代码218是以计算

机可读介质的形式选择性可移除的并且可以加载到或传送到数据处理系统200上以供处理器单元204执行。在这些例子中，程序代码218和计算机可读介质220构成计算机程序产品222。在一个例子中，计算机可读介质220可以是非暂时性和有形的形式，诸如像插入或放入作为持久性存储装置208一部分的驱动器或其它设备中的光盘或磁盘，用于传送到诸如作为持久性存储装置208一部分的硬驱的存储设备上。以有形的形式，计算机可读介质220还可以采取持久性存储装置的形式，诸如连接到数据处理系统200的硬驱、拇指驱动器或者闪存存储器。计算机可读介质220的有形形式也被称为计算机可读存储介质或计算机可读存储设备224，其中数据是非暂时性的并且因此不构成包含暂时信号的传播介质。在有些情况下，计算机可读介质220不能是可移除的。

[0055] 或者，程序代码218可以作为计算机可读信号介质226——这是暂时形式——通过到通信单元210的通信链路和/或通过到输入/输出单元212的连接从计算机可读介质220传送到数据处理系统200。在说明性例子中，通信链路和/或连接可以是物理的或者无线的。

[0056] 在有些说明性实施例中，程序代码218可以经网络从在数据处理系统200中使用的另一设备或数据处理系统下载到持久性存储装置208。例如，存储在服务器数据处理系统系统中的计算机可读数据存储设备中的程序代码可以经网络从服务器下载到数据处理系统200。提供程序代码218的数据处理系统可以是服务器计算机、客户端计算机，或者能够存储和发送程序代码218的某种其它设备。在当前的例子中，程序代码218包含图1的安全有条件操作系统116。

[0057] 利用图2的数据处理系统200作为例子，给出了在存储访问不能被证明是安全的时进行安全有条件操作的计算机实现过程。处理器单 元204接收用于事务的部分源代码，并使用增强编译器分析接收到的所述部分源代码以确定由所述增强编译器分析的所述部分源代码是否是用于变换的候选。响应于确定所分析的所述部分源代码是用于变换的候选，由处理器单元204使用增强编译器变换被分析的所述部分源代码以在变换后代码的第一部分中使用有条件操作，其中相应的有条件操作使用硬件事务存储器在硬件中调用重试操作，及在变换后代码的第二部分中添加指向原始代码部分的分支，其中所述分支的代码是包含所述原始代码部分的恢复部分。

[0058] 因此，本公开的实施例提供了除去禁止器以生成较少分支代码的能力。当使用本公开的实施例时，LOAD_ON_COND或STORE_ON_COND操作当中任何一个的使用都不再被有条件加载或有条件存储操作可能造成无效访问的可能性禁止。与确定这些有条件操作的使用相关联的开销被消除了，这是因为本公开的实施例提供正确的结果。本公开的实施例使用在失败时重试的硬件事务存储器能力来提供所需的正确性并且适用于静态编译器。

[0059] 其中区域中null检查的推测性消除只在存在检测null引用并且适用于动态编译器的JVM虚拟存储器子系统时才成功的之前尝试过的解决方案没有解决程序很大程度上被数据驱动的分支代码选通(gated)的问题。

[0060] 其它尝试过的解决办法集中在具有更小占用面积以降低成本并且产生更少热量的硬件但是没有解决关于正确性问题的推测性。相反，本公开的实施例通常是作为软件程序的程序代码提供的，以便通过在其中分支操作的消除将危及程序的数据完整性的代码位置避免潜在分支预测失误的情况来提高运行时性能。

[0061] 本公开的实施例提供了放松检查条件并确保有条件加载/存储操作安全的需求的

能力。因此，实施例使用硬件事务存储器的启用编译后的代码的更积极的代码优化的能力。因此，本公开的实施例启用让编译器不生成具有高预测失误概率的依赖数据的分支的更积极方式。在例子中，本公开实施例的使用提供了启用推测性代码运动的附加能力。使用硬件事务存储器的成本与从允许推测得到的节约进行比较，以确定使用硬件事务存储器的净结果是否带来了益处。因此，本公开的实施例可以在支持硬件事务存储器的任何体系架构上使用。

[0062] 因此，事务存储器提供了一种框架，通过这种框架，非法访问异常可以被安全地捕捉。照此，事务存储器可以用来放松对于让编译器证明LOAD_ON_COND或STORE_ON_COND将不造成非法异常的需求，由此启用LOAD_ON_COND和STORE_ON_COND操作的更积极使用，以避免涉及依赖数据的分支的潜在分支预测失误。

[0063] 参考图3，给出了可操作用于本公开的各种实施例的安全有条件操作系统的框图。这个例子在图2数据处理系统200的上下文中绘出了安全有条件操作系统116的可能实现结构的视图。

[0064] 在实现但不损失功能的情况下，所绘出的安全有条件操作系统116可以包含比所说明的更多或更少的部件。例如，部件可以组合，以减少独立部件的个数，但不损害部件的功能。所提供的部件支持操作的编译时和运行时模式，同时充分利用部件所依赖的数据处理的底层支持。部件在相互依赖的上下文中操作，以提供所公开的能力集合。

[0065] 源代码302代表对安全有条件操作系统中的处理的基本输入。源代码302是为了被增强编译器304编译而提供的编程指令。所提供的源代码包含识别出的分支模式，这意味着分支指令在所提供的源代码的段或部分中存在。在编译的一步或多步当中，所接收到的代码中的模式可以被处理利用增强编译器304检测到。源代码的段或部分代表程序代码，这可以是以高级语言、脚本语言或者其它形式的程序指令。

[0066] 增强编译器304提供了编译作为输入接收到的源代码302的能力。增强编译器304执行通常与包括代码分析器306和代码变换优化器308的功能的优化编译器关联的功能。作为增强编译器304的部件或功能，代码分析器306提供了检测分支的一个或多个预定模式，其表示在所提供的源代码的一个或多个段或部分中存在的分支指令，的能力。

[0067] 响应于代码分析器306检测到分支的一个或多个预定模式(或者有条件操作的常规使用)，包含检测到的分支操作的代码段被识别出来，供进一步处理。代码变换优化器308处理识别出的包含检测到的分支操作的代码段。代码变换优化器308执行变换，其中识别出的代码段被修改，以便启用让编译器不生成具有高预测失误概率的依赖数据的分支的更积极方式。变换通常在增强编译器304的编译器优化器引擎中发生。

[0068] 例如，变换使用事务存储器来放松对于让增强编译器304证明LOAD_ON_COND或STORE_ON_COND当中任何一个的源将不造成非法异常的需求，由此启用LOAD_ON_COND和STORE_ON_COND操作的更积极使用，以避免在识别出的代码段中涉及依赖数据的分支的潜在分支预测失误。在失败时重试的硬件事务存储器能力被用来提供结果操作的正确性。变换后代码包括当事务失败时离开失败事务的分支。该离开失败事务的分支是到结合原始分支代码的恢复代码段。

[0069] 因此，代码分析器306和代码变换优化器308的组合在解析后的源代码中识别出与利用原始分支代码的分支恢复段相结合地使用硬件事务存储器的机会。当有条件操作使用

变换后代码段的硬件事务存储器时，在识别出的代码段中涉及依赖数据的分支的潜在分支预测失误被避免了。

[0070] 编译器生成的代码312提供了确定所执行代码的执行流和输出是否如预期那样的能力。例如，确定包括变换后代码部分的事务是否将失败。

[0071] 指令集310提供了让诸如作为输入在源代码302中提供的程序之类的程序使用具体而言包括LOAD_ON_COND指令或STORE_ON_COND指令的有条件操作的指令。

[0072] 硬件事务存储器能力是利用事务存储器支持件314提供的。硬件事务存储器通常是作为系统服务或平台服务提供的。硬件事务存储器支持件包括无需编程干预的re-try on failure(在失败时重试)能力。

[0073] 参考图4，给出了包括在本公开的各种实施例中使用的有条件操作的事务的代码片段表示。代码片段400的例子绘出了与图3的安全有条件操作系统116一起使用的事务中的有条件操作以及恢复分支的可能实现的视图。

[0074] 语句402代表事务的代码段，包括一组指令，包含到特定恢复代码段和有条件操作的分支。语句404代表代码的原始分支实现的代码段。利用图3的代码分析器306和代码变换器308的组合，语句402和404代表代码的原始分支实现的代码变换结果。

[0075] 语句406识别定义事务的代码部分的开始并且被指示所定义事务结束的对应语句414界定。

[0076] 语句408代表离开事务例程到被识别为Label_Recovery的特定恢复代码段的定向分支。关联的注释指示该分支要在识别出的事务失败时采用。然后，恢复路径可以执行代码的原始分支实现。

[0077] 语句410规定要设置的特定条件代码，这在后续的有条件操作中使用。语句412是有条件操作。语句412指示，如果被测试的条件不为真，则加载操作使用由0识别的存储位置处的值(Rptr)加载到Rt的另一个存储位置。

[0078] LOAD_ON_COND指令或STORE_ON_COND指令中任何一个现在可以在所定义的事务中推测性地使用。恢复路径可以执行代码的原始分支实现。

[0079] 语句416规定Label_Recovery：与语句408中指示的离开事务例程的定向分支对应。语句418与语句410相同，它规定要设置的特定条件代码，在这个例子中这在后续的无条件操作中使用。

[0080] 语句420规定原始提供的分支指令。关联的注释指示这个分支实例预测得不好。

[0081] 语句422规定形式为LOAD指令的无条件操作。如在关联的注释中指示的，如果Rptr的值不等于零，则该无条件加载指令使得*ptr加载到位置Rt中。存储对另一个变量的引用的变量被称为指针并且指针用来“指向”其引用被存储的特定变量。使用指针使得能够对存储在被特定指针指向(引用)的变量中的值进行直接访问。为了实现这个目的，指针的标识符前面加充当解除引用操作符的星号(*)。*ptr按照字面被解释为“被…指向的值”。

[0082] 语句424规定在语句420中规定的原始分支指令的目标。

[0083] 参考图5，给出了用于在本公开的各种实施例中使用的有条件操作的优化范围的表示。优化范围500的例子绘出了与图3的安全有条件操 作系统116一起使用的事务中的有条件操作以及恢复分支的可能实现的范围。

[0084] 优化范围500绘出了有条件操作的可能实现的范围，其中使用硬件事务存储器的

成本与从允许推测得到的节约进行比较，以确定使用硬件事务存储器的净结果是否带来益处。代码优化502的成本代表使用包括固有开销的硬件事务存储器和使用在分支操作的预测失误中固有的推测的组合。

[0085] 代码优化502的成本被下限504和上限506界定。下限504反映硬件事务存储器(HTM)开销主要因素508代表代码优化成本的重要部分的点。例如，使用硬件事务存储器导致处理开销，这就任何接收到的好处而言太昂贵了。这会在分支太不可预测的时候或者在当使用有条件操作时已知分支操作要等待太长时间并且是代码重新设计的候选的时候发生。推测主要因素工作量510代表当推测由于分支代码的不确定本性而变得不太期望时的情形。来自识别出的代码段的代码轨迹的统计可以用来提供分支行为的量化分析。根据该分析，优化机会可以被进一步过滤，以确定硬件事务存储器或推测可能有或者可能没有益处的点并因此识别使用哪种方法。

[0086] 参考图6，给出了生成在本公开各种实施例中使用的安全有条件操作代码部分的过程的流程图。过程600是使用图3的安全有条件操作系统的例子。

[0087] 过程600开始(步骤602)并且接收源代码的一部分。源代码代表期望对其进行优化并且包含分支操作的程序。将检查分支操作是否有可能使用有条件操作和恢复代码段。

[0088] 过程600分析所接收到的源代码部分。分析通常是在解析阶段进行的，以识别分支的预定模式(或者有条件操作的常规使用)。分析可以由诸如图3的代码分析器306之类的部件或者适于解析和检查源代码以获得特定代码序列或操作指令或者其组合的其它部件执行。

[0089] 过程600确定被分析的代码部分是否是用于变换的候选(步骤608)。作为分析的结果，确定被分析的部分是否包含预定的分支模式或者有条 件操作的常规使用。确定可以利用源代码部分中的语句与包括BRANCH(分支)指令的特定已知模式的语句的简单比较来进行。

[0090] 响应于确定所分析的代码部分不是用于变换的候选，过程600确定是否存在更多为了变换而被分析的代码部分(步骤614)。响应于确定不存在更多为了变换而被分析的代码部分，过程600终止(步骤616)。

[0091] 返回步骤608，响应于确定所分析的代码部分是用于变换的候选，过程600把所分析的代码部分变换成使用有条件操作。代码变换过程保留原始代码部分的条件设置代码并且用对应的有条件操作代替无条件操作(诸如加载操作或存储操作)。对应的有条件操作使用硬件事务存储器在硬件中调用重试操作。

[0092] 过程600添加了在变换后代码失败时使用原始代码部分的分支(步骤612)。该分支是从包括变换后代码的有条件操作代码部分的段到包括原始分支代码的恢复段。原始分支代码是在变换后代码的有条件操作失败时的目标。

[0093] 就像前面一样，过程600确定是否存在更多为了变换而被分析的代码部分(步骤614)。响应于确定存在更多为了变换而被分析的代码部分，就像前面一样，过程600返回去执行步骤610。

[0094] 参考图7，给出了使用在本公开各种实施例中使用的安全有条件操作代码部分的过程的流程图。过程700是使用利用如图6中的过程生成的安全有条件操作代码部分的例子。过程700代表利用图6的过程600生成的变换后代码部分的执行。

[0095] 过程700开始(步骤702)并且执行变换后代码部分(步骤704)。变换后代码部分是包括变换后代码的有条件操作代码部分的那些段和包括与该有条件操作代码部分关联的原始分支代码的相应恢复段。

[0096] 在变换后代码部分的执行期间,过程700设置条件(步骤706)。虽然条件的设置也是在变换后代码中执行的,但是条件是原始源代码部分的条件。

[0097] 过程700处理变换后代码的有条件操作代码部分,以执行规定的有条件操作(步骤708)。所规定的有条件操作代表变换后代码的第一部分,而包括与该有条件操作代码部分关联的原始分支代码的相应恢复段代表变换后代码的第二部分。

[0098] 确定具有变换后代码部分的事务是否失败(步骤710)。响应于确定了具有变换后代码部分的事务没有失败,过程700终止(步骤720)。

[0099] 响应于确定了具有变换后代码部分的事务失败,过程700分支到恢复例程(步骤712)。恢复例程是在变换后代码部分的第一部分中规定的并且使得能够在变换后代码的有条件操作失败时通过使用原始分支代码作为目标而从第一部分的失败的有条件操作得体地离开。

[0100] 过程700确定分支是否预测得不好(步骤714)。响应于确定分支预测得不好,过程700使用*ptr作为有条件操作的自变量(步骤716)并且在这之后终止(步骤720)。响应于确定分支不是预测得不好,过程700离开恢复例程分支到在原始分支代码中规定的目标例程(步骤718)并且在这之后终止(步骤720)。

[0101] 这样就在说明性实施例中给出了当存储访问不能被证明是安全的时进行安全有条件操作的计算机实现方法。该计算机实现的方法由增强编译器接收用于事务的部分源代码并且由所述增强编译器分析接收到的所述部分源代码。确定由所述增强编译器分析的所述部分源代码是否是用于变换的候选。

[0102] 响应于确定由所述增强编译器分析的所述部分源代码是用于变换的候选,该计算机实现的方法的执行变换被分析的所述部分源代码以在变换后代码的第一部分中使用有条件操作及在变换后代码的第二部分中添加指向原始代码部分的分支,其中所述分支的代码是包含所述原始代码部分的恢复部分。该恢复部分提供向后兼容,同时使得能够使用事务处理器能力。

[0103] 附图中的流程图和框图说明了根据本发明各种实施例的系统、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上,流程图或框图中的每个方框可以代表一个模块、程序段或代码的一部分,所述模块、程序段或代码的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。还应当指出,在有些作为替换的实现中,方框中所 标注的功能也可以以不同于附图中所标注的顺序发生。例如,依赖于所涉及的功能,两个连续示出的方框实际上可以基本并行地执行,或者它们有时也可以按相反的顺序执行。还要指出的是,框图和/或流程图图示中的每个方框、以及框图和/或流程图图示中的方框的组合,可以用执行规定的功能或动作的专用的基于硬件的系统来实现,或者可以用专用硬件与计算机指令的组合来实现。

[0104] 以下权利要求中所有方式或步骤加功能元素的对应结构、材料、动作及等同都是要包括用于结合具体所述的其它所述元素执行所述功能的任何结构、材料或动作。已经为了说明和描述的目的而给出了本发明的描述,但这不是详尽的或者要把本发明限定到所公

开的形式。在不背离本发明范围与主旨的情况下，许多修改和变化对本领域普通技术人员都将是显而易见的。实施例的选择和描述是为了最好地解释本发明的原理和实践应用，并使本领域普通技术人员能够理解本发明具有适于预期特定使用的各种修改的各种实施例。

[0105] 本发明可以采取完全硬件实施例、完全软件实施例或者结合硬件与软件元素的实施例的形式。在优选实施例中，本发明用软件实现，包括但不限于固件、驻留软件、微代码以及本领域技术人员可以认识到的其它软件介质。

[0106] 值得注意的是，虽然本发明已经在全功能的数据处理系统的上下文中进行了描述，但是本领域普通技术人员将认识到，本发明的过程能够以其上存储计算机可执行指令的计算机可读数据存储设备的形式分布。计算机可读数据存储设备的例子包括可记录类型的介质，诸如软盘、硬盘驱动器、RAM、CD-ROM、DVD-ROM。计算机可执行指令可以采取编码格式的形式，这种编码格式为了在特定的数据处理系统中使用而被解码。

[0107] 适于存储和/或执行包括程序代码的计算机可执行指令的数据处理系统将包括通过系统总线直接或间接耦合到存储器元件的一个或多个处理器。存储器元件可以包括在程序代码的真正执行期间所采用的本地存储器、大容量存储装置和高速缓存存储器，其中高速缓存存储器提供至 少一些程序代码的临时存储，以便减少在执行过程中必须从大容量存储装置检索代码的次数。

[0108] 输入/输出或者I/O设备(包括但不限于键盘、显示器、定点设备等)可以直接地或者通过中间I/O控制器耦合到系统。

[0109] 网络适配器也可以耦合到系统，以便使数据处理系统能够通过中间的专用或公共网络变得耦合到其它数据处理系统或者远端打印机或存储设备。调制解调器、电缆调制解调器和以太网卡仅仅是当前可以获得的几种类型的网络适配器。

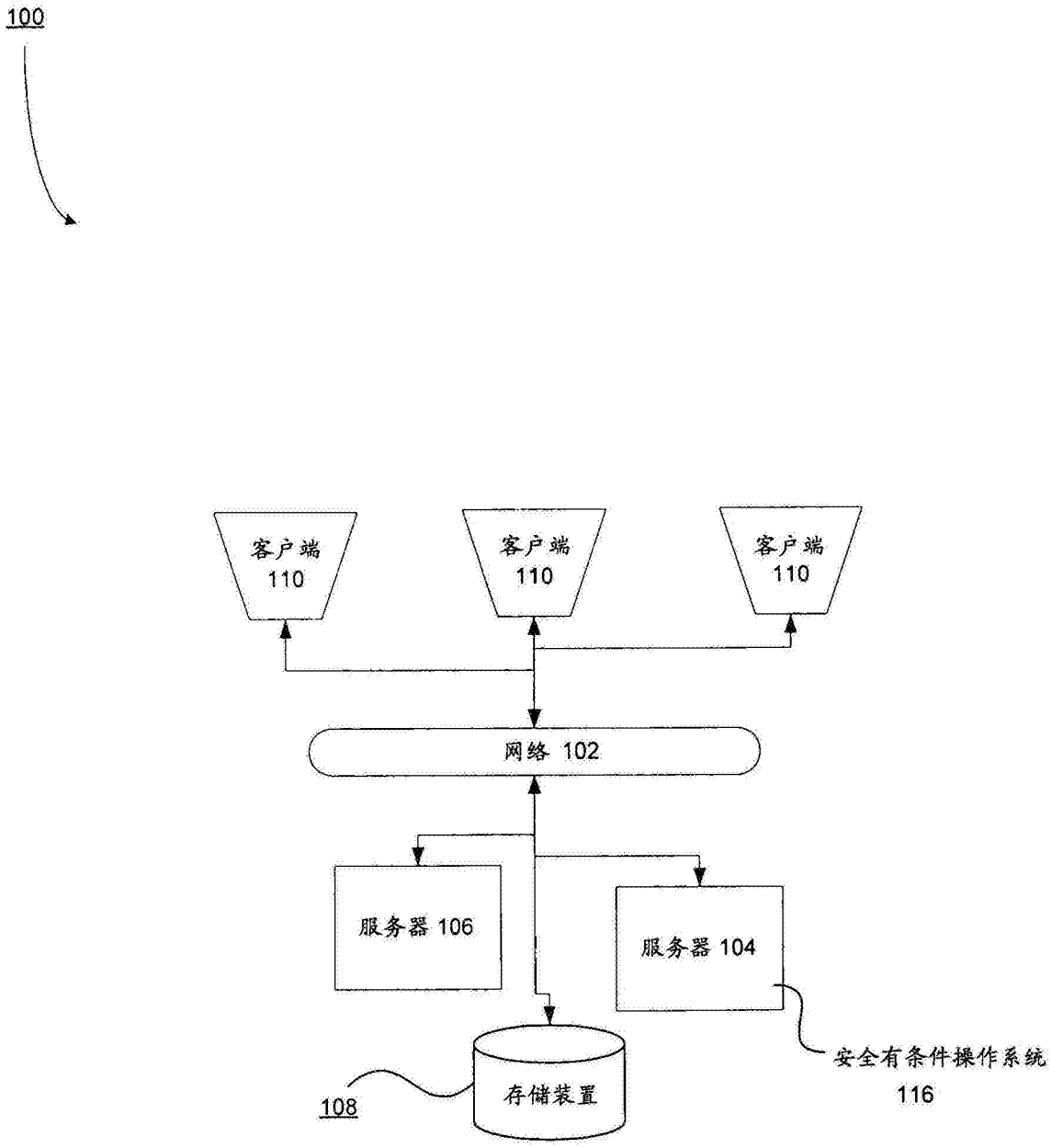

100
客户端
110
客户端
110
客户端
110
网络 102
服务器 106
服务器 104
安全有条件操作系统
116
108
存储装置

图1

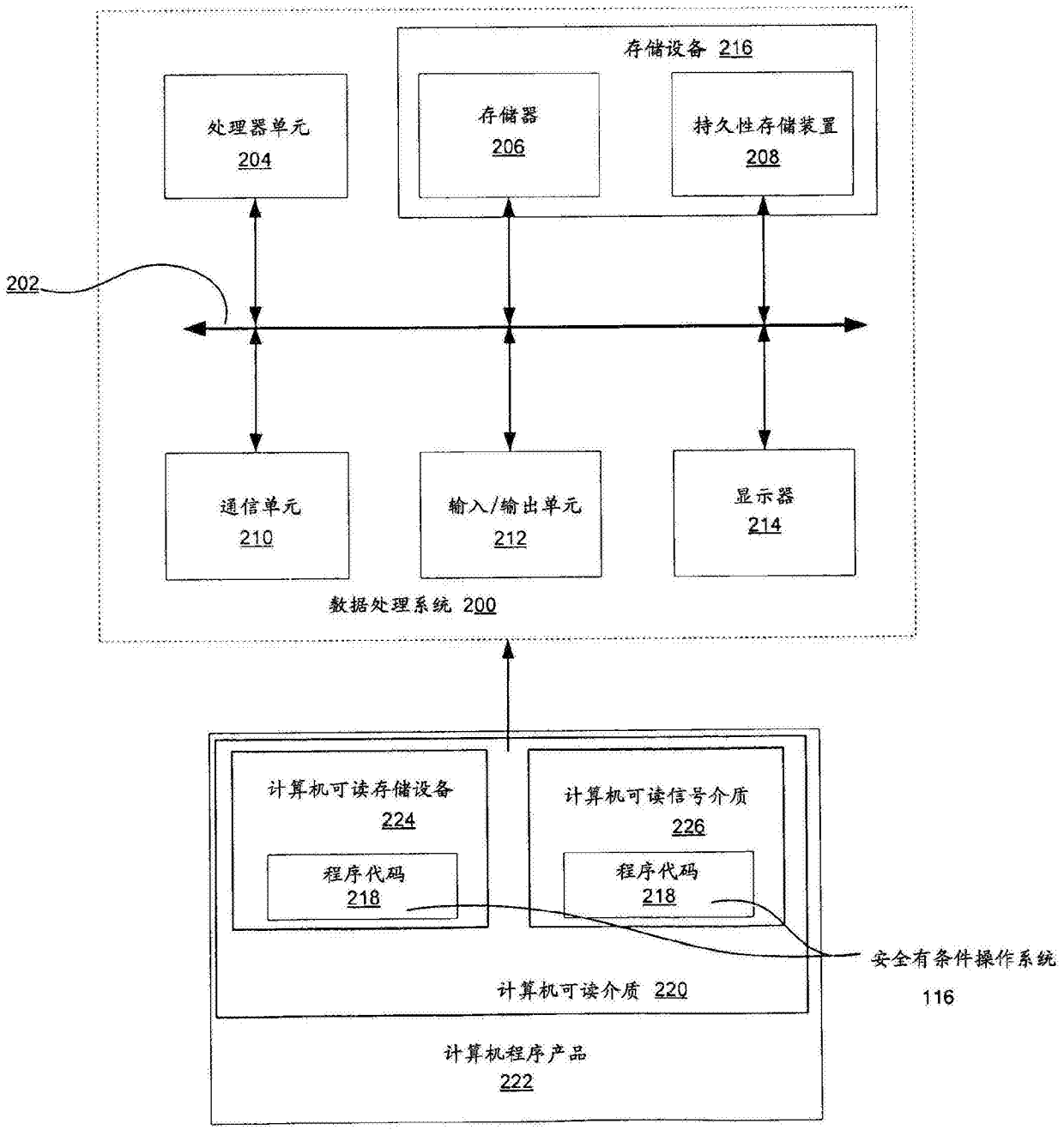

存储设备 216
处理器单元
204
存储器
206
持久性存储装置
208
202
通信单元
210
输入/输出单元
212
显示器
214
数据处理系统 200
计算机可读存储设备
224
程序代码
218
计算机可读信号介质
226
程序代码
218
安全有条件操作系统
116
计算机可读介质 220
计算机程序产品
222

图2

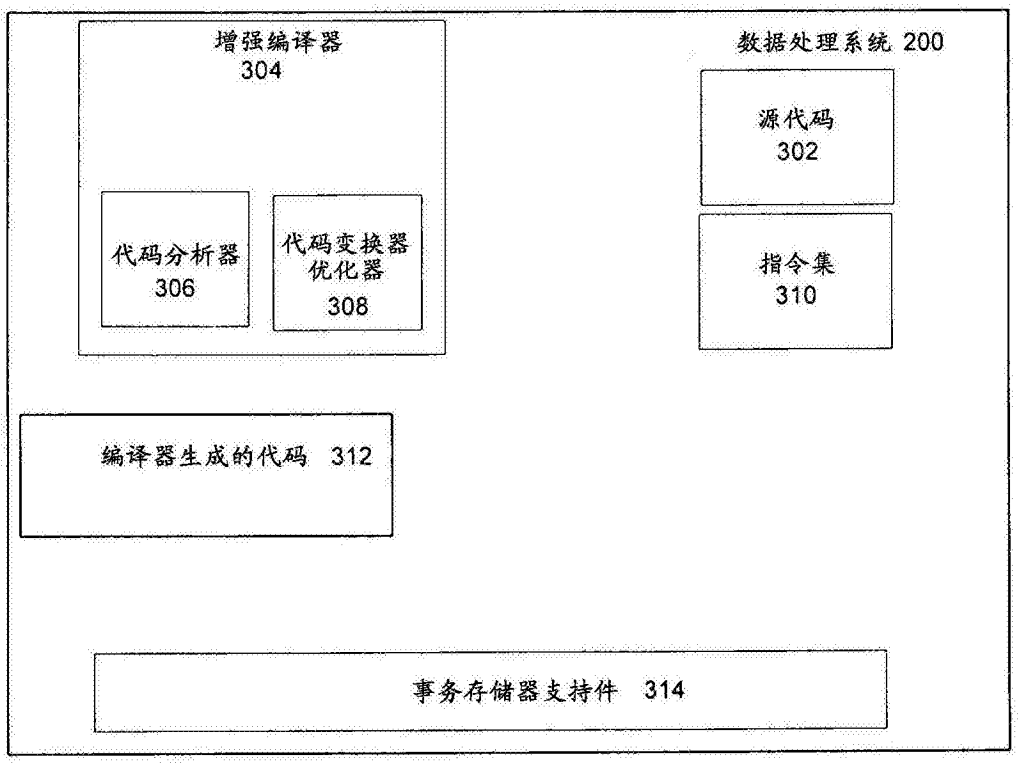

增强编译器
304
数据处理系统 200
源代码
302
代码分析器
306
代码变换器
优化器
308
指令集
310
编译器生成的代码 312
事务存储器支持件 314
安全有条件操作系统
116

图3

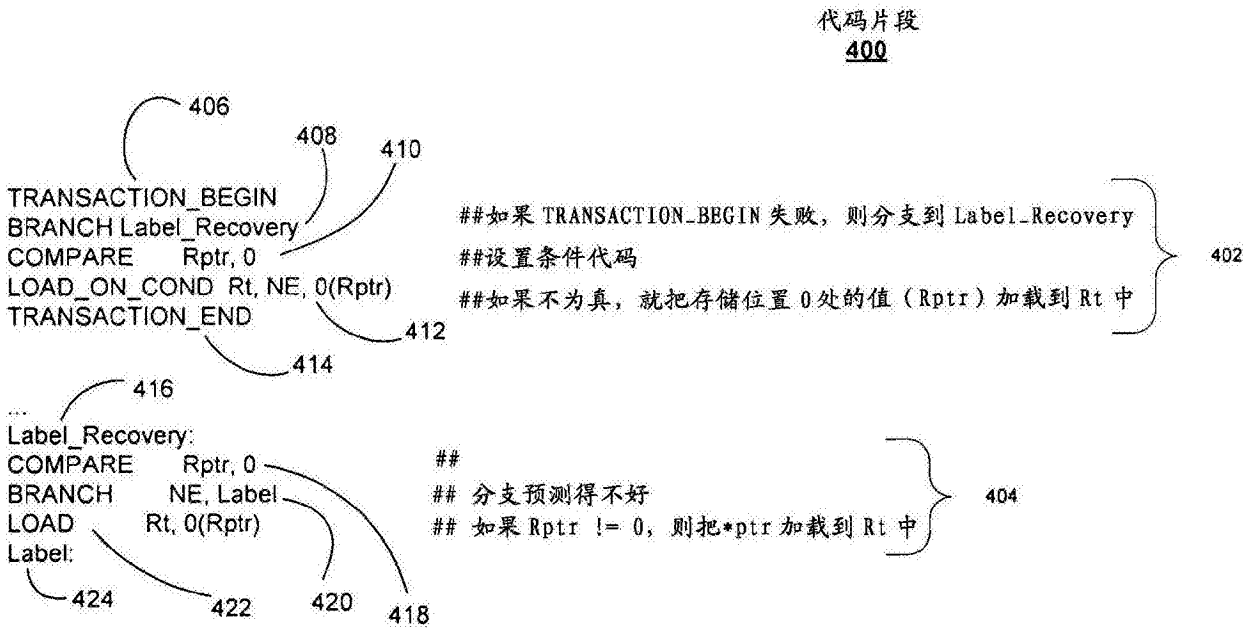

代码片段
400
406
408
410
TRANSACTION_BEGIN
BRANCH Label_Recovery
COMPARE Rptr, 0
LOAD_ON_COND Rt, NE, 0(Rptr)
TRANSACTION_END
412
414
##如果 TRANSACTION_BEGIN 失败，则分支到 Label_Recovery
##设置条件代码
##如果不为真，就把存储位置 0 处的值（Rptr）加载到 Rt 中
402
416
...
Label_Recovery:
COMPARE Rptr, 0
BRANCH NE, Label
LOAD Rt, 0(Rptr)
Label:
424
422
420
418
##
分支预测得不好
如果 Rptr != 0，则把*ptr 加载到 Rt 中
404

图4

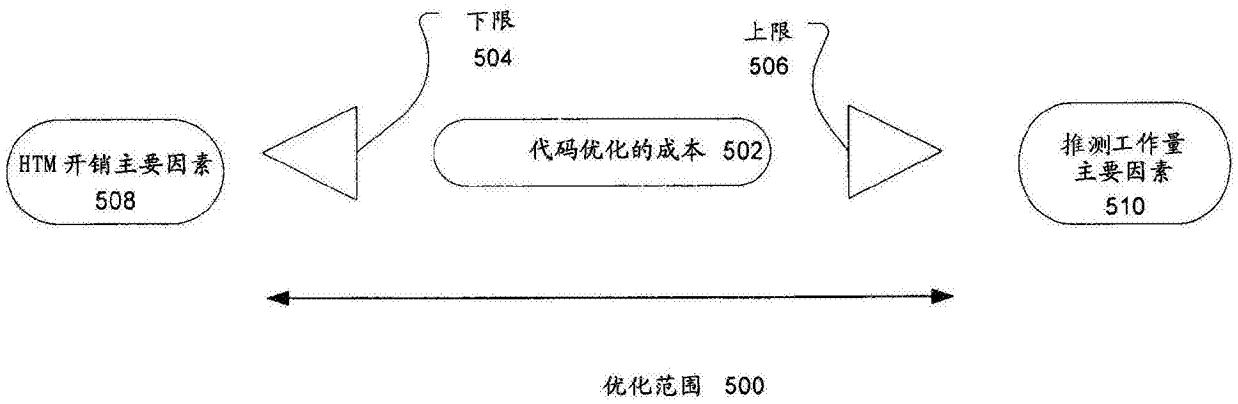

下限
504
上限
506
HTM 开销主要因素
508
代码优化的成本 502
推测工作量
主要因素
510
优化范围 500

图5

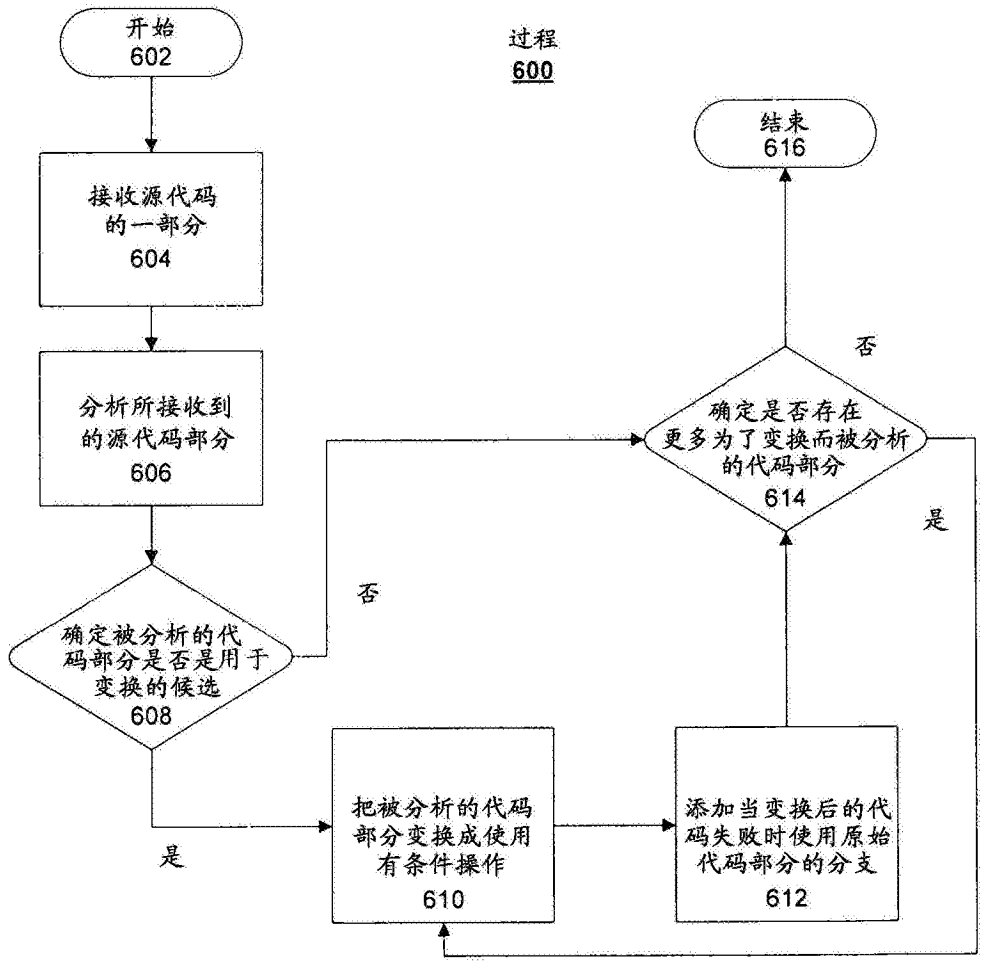

开始
602
过程
600
结束
616
接收源代码
的一部分
604
分析所接收到
的源代码部分
606
确定是否存在
更多为了变换而被分析
的代码部分
614
否
是
确定被分析的代
码部分是否是用于
变换的候选
608
否
是
把被分析的代码
部分变换成使用
有条件操作
610
添加当变换后的代
码失败时使用原始
代码部分的分支
612

图6

过程
700

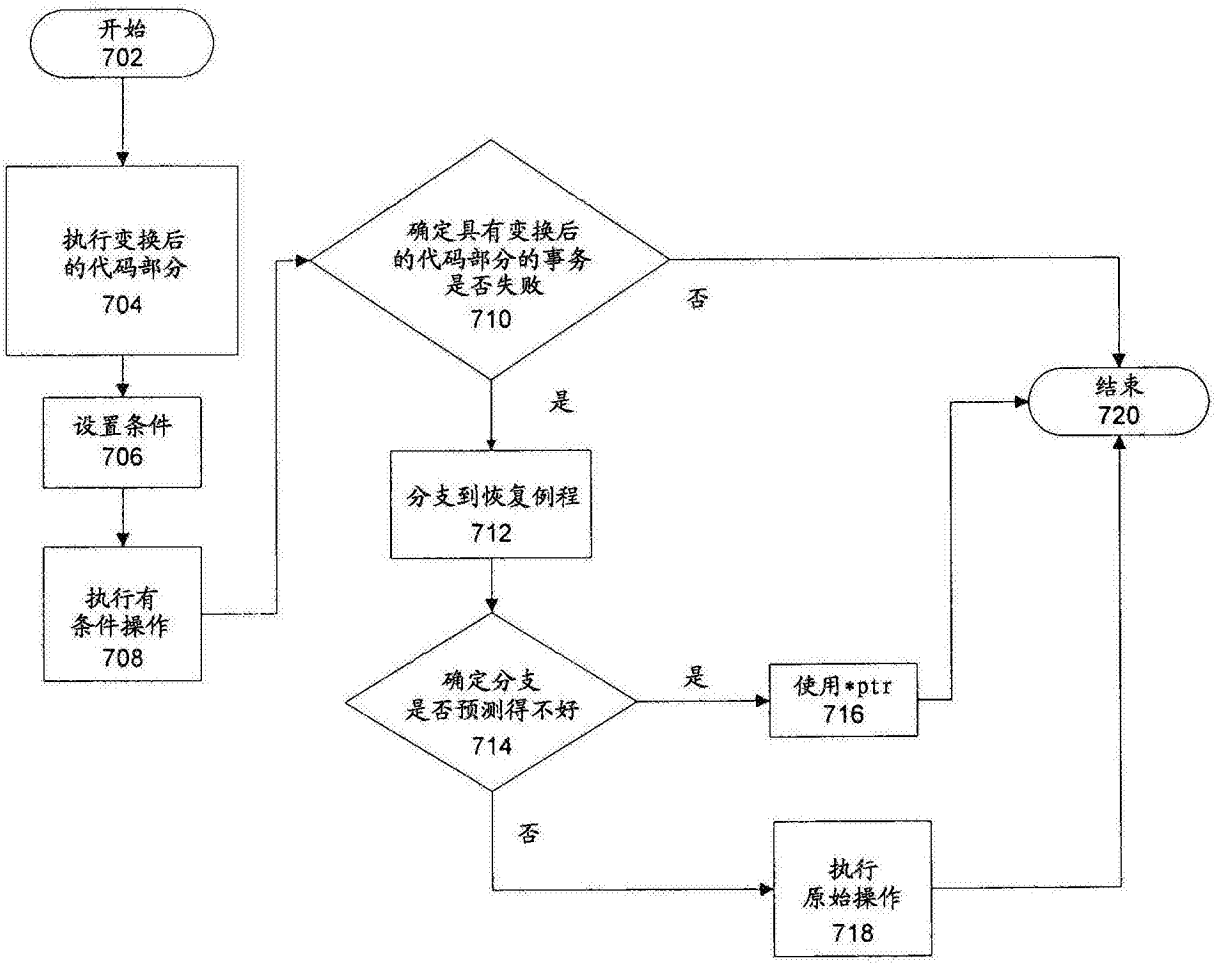

开始
702
执行变换后的代码部分
704
设置条件
706
执行有条件操作
708
确定具有变换后的代码部分的事务是否失败
710
否
是
结束
720
分支到恢复例程
712
确定分支是否预测得不好
714
是
使用*ptr
716
否
执行原始操作
718

图7