

[19] 中华人民共和国国家知识产权局



[12] 发明专利说明书

[51] Int. Cl.

G06F 9/46 (2006.01)

G06F 9/455 (2006.01)

专利号 ZL 200380106627.5

[45] 授权公告日 2009 年 6 月 24 日

[11] 授权公告号 CN 100504789C

[22] 申请日 2003.11.12

CN1248742A 2000.3.29

[21] 申请号 200380106627.5

WO9914671A1 1999.3.25

[30] 优先权

US4814975A 1989.3.21

[32] 2002.12.17 [33] US [31] 10/322,003

US4674038A 1987.6.16

[86] 国际申请 PCT/US2003/036306 2003.11.12

US6205467B1 2001.3.20

[87] 国际公布 WO2004/061645 英 2004.7.22

审查员 孟宪超

[85] 进入国家阶段日期 2005.6.17

[74] 专利代理机构 中国专利代理(香港)有限公司

[73] 专利权人 英特尔公司

代理人 李亚非 王 勇

地址 美国加利福尼亚州

[72] 发明人 G·奈格尔 E·科塔-罗布尔斯
S·耶亚辛 A·卡吉 M·科祖赫
R·乌利希 E·博利恩
S·罗杰斯 L·史密斯三世
S·贝内特 A·格卢

[56] 参考文献

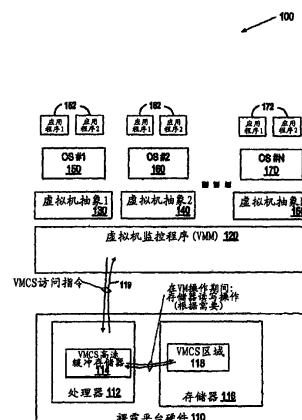
权利要求书 1 页 说明书 15 页 附图 5 页

[54] 发明名称

控制虚拟机的方法

[57] 摘要

提供用于控制虚拟机(VM)的执行的方法和系统。VM监视器(VMM)通过访问传递至处理器的指令间接地访问VM控制结构(VMCS)。在一个实施例中，所述访问指令包括VMCS组件标识符，所述VMCS组件标识符由所述处理器使用以确定VMCS组件的适当存储位置。所述处理器在处理器存储空间内或者在存储器内识别VMCS组件的适当的存储位置。



1. 一种用于激活虚拟机控制结构的方法：

由虚拟机监视器执行访问含有虚拟机控制结构所需的存储空间大小的存储位置的至少一个指令和提供虚拟机控制结构所需的存储空间大小的指令来确定虚拟机控制结构所需的存储空间大小；并且

向处理器提供指向存储器区域的指针，所述存储器区域至少与虚拟机控制结构所需的存储空间大小一样大。

2. 如权利要求 1 所述的方法，其中在向处理器提供指针时，通过执行虚拟机控制结构指针载入指令来提供该指针。

3. 如权利要求 2 所述的方法，其中在执行虚拟机控制结构指针载入指令时，所述虚拟机控制结构指针载入指令包括至少与虚拟机控制结构所需存储空间大小一样大的存储器区域的指针地址，并且将该指针地址作为操作数，所述操作数是虚拟机控制结构指针载入指令的显式操作数和虚拟机控制结构指针载入指令的隐式操作数中的至少一个。

4. 如权利要求 1 所述的方法，还包括：在向处理器提供指向存储器区域的指针之前，分配所述存储器区域。

控制虚拟机的方法

技术领域

本发明的实施例总体上涉及计算机系统，具体来讲，涉及计算机系统内虚拟机的操作控制。

背景技术

虚拟机体系结构从逻辑上划分物理机，以致机器的底层硬件是分时的，并且表现为一个或多个独立操作的虚拟机(VM)。虚拟机监视器(VMM)在计算机上运行，并且便于为其他软件抽象一个或多个VM。每个VM可以起独立平台的作用，运行其自身的操作系统(OS)和应用软件。此处把在VM中运行的软件统称为客户软件。

客户软件期待这样的操作，就好像它正在专用计算机上运行而不是在VM上运行。也就是说，客户软件期待控制各种事件并且有权访问计算机(例如物理机)上的硬件资源。所述物理机的硬件资源可以包括一个或多个处理器、驻留在所述处理器上的资源(例如控制寄存器、高速缓冲存储器及其他)、存储器(以及驻留在存储器中的结构、例如描述符表)以及驻留在所述物理机中的其他资源(例如，输入输出设备)。所述事件可以包括中断、异常、平台事件(例如初始化(INIT)或者系统管理中断(SMI)等等)。

所述VMM可以根据需要将客户软件状态交换(swap)进出物理机的设备、存储器以及寄存器。所述VMM可以通过允许直接访问底层的物理机来增强VM的性能。当在客户软件中正在以非特权模式执行操作时，所述方式限制软件访问所述物理机，或者当操作不利用物理机中VMM希望保持控制的硬件资源时，这是尤其适用的。

每当客户操作可能影响VMM或者任何未执行VM的正确执行时，所述VMM恢复控制。通常，所述VMM审查这种操作，在允许所述操作进行到底层物理机或者模拟对客户利益的操作之前，确定是否存在问题。例如，当客户访问I/O设备时、当其试图(例如通过改变控制寄存器值)改变机器配置时、当其试图访问存储器的某区域等等时，所述VMM可能需要恢复控制。

支持VM操作的现有系统使用固定格式结构来控制VM的执行环境，此处将所述固定格式结构称为虚拟机控制结构(VMCS)。所述VMCS被存储在存储器区域中，并且例如包含客户状态、VMM的状态以及控制信息，所述

控制信息表明在客户执行期间，在什么条件下所述 VMM 希望恢复控制。物理机中的处理器读取来自于所述 VMCS 的信息以便确定 VM 和 VMM 的执行环境，并且约束客户软件在所述 VMM 控制之下的行为。

常规的体系结构把 VMCS 定位在物理机的存储器中，并且允许 VMM 使用普通存储器读写指令来访问它。为此，必须在处理器指令集架构中从体系结构上定义所述 VMCS 的格式(并且以类似于其他系统结构和指令编码的方式在规范和手册中记载)。所述 VMM 被直接编码为这些规范。这种结构限制了支持 VMM 的处理器在执行方面的灵活性。由于 VMCS 的形式是从体系结构上来定义的，所以出于性能、扩展性、兼容性、安全性等等的原因，特定处理器实现方式的微体系结构不可以在 VMCS 数据的格式、内容、组织或者存储需求方面做出改变，此外不需要对已安装的 VMM 执行基础做出相应修改。

因此，需要更加灵活的虚拟机体系结构的实现方式，该实现方式不必严格耦合至物理机的底层实现方式。

发明内容

根据本发明的一个方面，提供了一种用于激活虚拟机控制结构的方法，该方法包括：由虚拟机监视器执行访问含有虚拟机控制结构所需的存储空间大小的存储位置的至少一个指令和提供虚拟机控制结构所需的存储空间大小的指令来确定虚拟机控制结构所需的存储空间大小；并且向处理器提供指向存储器区域的指针，所述存储器区域至少与虚拟机控制结构所需的存储空间大小一样大。

附图说明

图 1 是依照本发明一个实施例的 VM 体系结构的图表。

图 2 是依照本发明一个实施例的控制 VM 的方法流程图。

图 3 是依照本发明一个实施例的 VMCS 访问指令的图表。

图 4 是依照本发明一个实施例的读取来自于 VMCS 的数据的方法流程图。

图 5 是依照本发明一个实施例的写数据至 VMCS 的方法流程图。

具体实施方式

描述新颖的 VM 控制体系结构。在随后对实施例的详细描述中，参照了附图，该附图作为本发明的一部分，并且其中通过举例说明而非限制的方式示出了可以实施本发明的特定实施例。这些实施例是以足够多的细节来描述的，以便使本领域普通技术人员可以理解和实现它们，并且应该理解的是，也可以利用其他实施例，并且可以在不脱离本公开内容的精神和范围的情况下，做出结构上、逻辑上和电气上的改变。因此，不应该将以下的详细说明认为是限制意义上的，并且此处所公开的本发明的实施例的范围只由所附权利要求定义。

VMM 给予其他软件（“客户软件”、“多个客户”或者仅仅“一个客户”）一个或多个 VM 的抽象。所述 VMM 可以向各种客户提供相同或者不同的抽象。每个客户期待出现于 VM 中的所有硬件平台设施都可供其使用。例如，依照处理器的体系结构和出现于所述 VM 中的平台，客户期待有权访问所有寄存器、高速缓冲存储器、结构、I/O 设备、存储器等等。此外，每个客户期待处理各种事件，诸如处理异常、中断和平台事件（例如初始化（INIT）或者系统管理中断（SMI））。

一些资源和事件被“给予特权”，因为它们必须由 VMM 管理，以便确保 VM 的正确操作并且保护 VMM 以及其他 VM。对于被给予特权的资源和事件来说，所述 VMM 简化（facilitate）客户软件所要求的功能，同时经由这些资源和事件保持最终控制。简化客户软件功能的动作就所述 VMM 而言可以包括各式各样的活动。所述 VMM 的活动及其特征不限制本发明各种实施例的范围。

当客户软件访问被给予特权的资源或者被给予特权的事件发生时，可以把控制传递给 VMM。把控制从客户软件传递到所述 VMM 称为 VM 退出。在简化资源访问或者适当地处理事件之后，所述 VMM 可以把控制返回至客户软件。把控制从 VMM 传递到客户软件称为 VM 进入。

所述虚拟机控制结构（VMCS）是从体系结构上定义的结构，该结构例如包含客户软件的状态、VMM 的状态、表明在哪些条件下 VMM 希望防止客户执行的控制信息以及涉及最新 VM 退出的信息。在当前系统中，正确地匹配从体系结构上定义的结构的 VMCS 的表示位于存储器中。物理机中的所述处理器读取来自 VMCS 的信息以便确定 VM 的执行环境并且约束其行为。

在客户执行期间，所述处理器查阅 VMCS 中的控制信息来确定哪些客户动作（例如某指令的执行、某异常的出现等）和事件（例如外部中断）将引

起 VM 退出。当发生 VM 退出时，由客户软件使用的处理机状态组件 (component) 被保存到 VMCS，并且把 VMM 所要求的处理机状态组件从所述 VMCS 中载入。当发生 VM 退出时，使用本领域普通技术人员所公知的任意机制把控制传递至 VMM 120。

当发生 VM 进入时，在 VM 退出时保存的处理机状态(并且其也许已由 VMM 修改)被恢复，并且把控制返回到客户软件。为了便于第一 VM 进入至客户，所述 VMM 把适当的客户状态写入 VMCS。当处理 VM 退出时，所述 VMM 可以改变 VMCS 中的客户状态。在一些实施例中，由单个物理机上的单个 VMM 管理支持多个 VM 的多个 VMCS 结构。所述 VMCS 不需要包括如上所述的全部信息，并且可以包括有助于 VM 的控制的附加信息。在一些实施例中，VMCS 可以包含很大量量的附加信息。

图 1 举例说明了依照本发明一个实施例的 VM 体系结构 100 的图表。所述 VM 体系结构 100 包括基础硬件平台 110 (例如物理机)。所述基础硬件平台 110 包括均有权访问易失性和/或非易失性存储器 116 的一个或多个处理器 112。另外，在基础硬件平台中还存在其他元件，在图 1 中没有示出这些元件(例如，输入输出设备)。所述 VM 体系结构 100 还包括 VMM 120，其管理一个或多个 VM (例如，130、140 和 150)，其中每个 VM (例如 130、140 和 150) 支持一个或多个 OS (例如 150、160 和 170) 和应用程序 (例如 152、162 和 172)。所述处理器 112 可以是能够执行软件的任何类型的处理器，诸如微处理器、数字信号处理器、微控制器等等。所述处理器 112 可以包括微代码、可编程逻辑或者硬编码逻辑，用于实现本发明的方法实施例的执行。

存储器 116 可以是硬盘、软盘、随机存取存储器 (RAM)、只读存储器 (ROM)、闪存、上述设备的任意组合，或者是可由处理器 112 读取的任何其他类型的机器介质。存储器 116 可以存储用于完成本发明的方法实施例的执行的指令或者数据。所述存储器 116 包括 VMCS 区域 118，由处理器 112 用于维护所述 VMCS 的状态，以下将更详细地描述。

所述处理器 112 可以是能够执行软件的任何类型的处理器，诸如微处理器、数字信号处理器、微控制器等等。每个处理器 112 可以包括 VMCS 高速缓冲存储器 114，以下将更详细地描述该 VMCS 高速缓冲存储器 114。所述处理器 112 可以包括微代码、可编程逻辑或者硬编码逻辑，用于完成依照本发明的方法实施例的执行。

如果存在，那么 VMCS 高速缓冲存储器 114 可用来临时或者在其整个使用期限内存储某些或者所有 VMCS 状态。所述 VMCS 高速缓冲存储器 114 可以包括寄存器、高速缓冲存储器或者任意其他存储器。在图 1 中，所述 VMCS 高速缓冲存储器 114 是作为处理器 112 的一部分示出的，但是它可以驻留在裸露平台硬件 110 的任意组件内的处理器 112 的外部。在随后的论述中，也把所述 VMCS 高速缓冲存储器 114 称为“处理器上的存储空间”或者“处理器上的资源”，但是应该理解的是，此存储器可以驻留在不同于处理器 112 的平台组件上。所述 VMCS 高速缓冲存储器 114 不是实现本发明方法所严格必需的。

通常，VMM 往往使用普通读写指令在存储器中访问 VMCS。然而在图 1 的 VM 体系结构 100 中，所述 VMM 120 通过一组处理器提供的 VMCS 访问指令 119 间接地访问 VMCS 区域 118。所述 VMCS 访问指令 119 利用 VMCS 区域 118 以及任意可利用的 VMCS 高速缓冲存储器 114。在一个实施例中，VMCS 访问指令 119 包括正被访问的 VMCS 组件的标识符的操作数。所述标识符此处被称为“组件标识符”。所述 VMM 120 不需要知道把任意特定的 VMCS 组件是存储在 VMCS 区域 118 中还是存储在 VMCS 高速缓冲存储器 114 中。所述 VM 体系结构 100 还规定：如果把普通读写指令用于访问 VMCS 区域 118，那么可能发生不可预测的结果。VMCS 访问指令 119 的使用给予处理器 112 自由使用可利用的处理器上存储器以及基于存储器（例如 VMCS 区域 118）的存储空间，并且也允许各种性能优化。

在一些实施例中，所述 VMCS 访问指令 119 是通过处理器微体体系结构在 VMCS 区域 118 中的读写存储器来实现的。在其他实施例中，所述 VMCS 访问指令 119 可以读和/或写处理器上的资源。所述 VMM 120 不必知道底层的物理机微体体系结构如何支持所述 VMCS。以这种方式，所述底层处理器实现方式可以被改变，以便适应性能、安全性、可靠性或者其他需要考虑的事项，而不会使所述 VMM 120 与底层处理器实现方式不兼容，并且可以为每个处理器实现方式开发自定义的 VMCS 实现方式。

以下例子举例说明了不采用 VMCS 访问指令 119 的 VM 体系结构的缺点。如果处理器实现方式可以在处理器上的存储空间中临时高速缓存 VMCS 数据，那么当发生特定事件时，只能把数据写入存储器内的 VMCS 区域 118 中。在高速缓存所述 VMCS 数据期间，对所述 VMCS 区域 118 的普通读取将返回失效值（不正确的值）。对 VMCS 区域 118 的普通写入不会更新处理器

上的存储空间中的数据，除非所述处理器实现方式采取特殊的保护措施，以便把写入内容正确映射至存储空间。在没有使用 VMCS 访问指令 119 的情况下，所述处理器实现方式必须保持 VMCS 区域与存储在处理器上的资源中的任意临时或者高速缓存的状态一致；这样会阻碍一定的性能优化、VM 体系结构的扩展等，如下面进一步讨论的那样。与普通存储器操作相比，如果适当的话，VMCS 读指令返回存储在处理器上的值，并且 VMCS 写入指令正确地更新 VMCS 状态，而无论其位于什么地方。下面将详细说明这些指令。

为了简化所述 VMCS 访问指令 119，VMCS 的每个元件由从体系结构上定义的常数来标识，该常数识别 VMCS 的组件；此处将这些恒定值称为“组件标识符”。在一个实施例中，所述组件标识符是 16 位值，但是在其他实施例中，该值可大可小。

在随后的论述中，使用 VMCS 中的多个字段来描述本发明的各种实施例。例如，GUEST_EIP 是包含客户软件的指令指针的字段；VM_CONTROLS 是包含控制 VM 执行环境的位的字段等。对各种字段的语法和语义的理解不是理解此处所描述的本发明所必需的。另外，所述字段被给予特定的从体系结构上定义的组件标识符。所使用的特定字段以及在这些例子中的组件标识符值不是以任何方式来限制本发明的应用。

如下是使用 VMCS 访问指令 119 的优点的示例说明。认为处理器实现方式可以根据处理器实现方式的不同来改变存储在 VMCS 区域 118 中的数据的布局。例如，第一处理器实现方式可以存储起始于 VMCS 区域的第 28 个字节处的 VMCS 区域的 GUEST_EIP 字段。第二处理器实现方式可以存储在 VMCS 区域中的第 16 个字节处的相同的 VMCS 字段。尽管 VMCS 区域 118 的布局已经改变，但是为初始实现方式写入并且使用所述 VMCS 访问指令 119 的 VMM 120 仍然对处理器 112 的后者实现方式起作用。因为在两个处理器实现方式中所述 VMCS 访问指令的实现方式包括所使用的 VMCS 区域 118 的布局并且适当地访问必要的数据，所以实现了兼容性。

在一些实施例中，为了简化所述 VMCS 访问指令 119，可以需要所述 VMM 120 留出存储器区域（例如 VMCS 区域 118）来收纳所有或者部分存储空间，该存储空间是处理器 112 为 VMCS 所要求的。在本发明的这些实施例中，处理器提供的指令将允许 VMM 120 把 VMCS 区域 118 的指针或者地址提供给处理器 112。在一个实施例中，所述地址是物理地址。其他实施例

可以使用虚拟或者线性地址。此后，把使用此指令提供给处理器 112 的所述地址称为 VMCS 指针。此指令通知处理器 112 VMCS 区域 224 的位置。此新的指令使 VMCS 有效，所述 VMCS 可以通过处理器 112 全部或部分存储在由 VMCS 指针指出 VMCS 区域 118 中。

另一实施例可以提供这样的机制，其允许 VMM 120 发现必须保留在 VMCS 区域 118 的存储器 116 的数量。例如，在英特尔奔腾 IV 的处理器指令基架构 (ISA) 中 (此处称为 IA - 32 ISA)，能够提供包括所需要的 VMCS 区域大小的模式特定寄存器 (MSR)。另一实施例提供了返回所需要的 VMCS 区域大小的指令。可利用的任意其他机制可用来把此信息传送至 VMM。以这种方式，所需要的存储器区域 118 的大小可以根据处理器实现方式的不同而改变，而不会强迫重新设计或者再编译所述 VMM 120。

例如，假如第一处理器 112 需要 64 字节用于存储器 116 中的 VMCS 区域 118。把此要求报告给 VMM 120，如上所述。使用如上所述的机制来确定把多少存储器 116 分配给 VMCS 区域 118，写入 VMM 120 以便在此处理器实现方式上运行。创建第二处理器实现方式，其现在需要 128 字节用于存储器 116 中的 VMCS 存储空间。为第一处理器实现方式写入的 VMM 120 将在第二实现方式上正确地操作，这是因为它将分配正确数量的存储空间给 VMCS 区域 118。

当所述 VMCS 指针有效时，所述处理器 112 可以把所有或者部分 VMCS 存储在 VMCS 区域 118 中，或者可替换地，存储在驻留在处理器 112 上的资源或者任意其他可利用的位置中 (即，存储在 VMCS 高速缓冲存储器 114 中)。另外，所述处理器 112 可以把非体系结构的状态信息存储在 VMCS 区域 118 中或者存储在处理器上的资源中。例如，所述处理器 112 可以存储临时变量、与 VM (例如 130、140 或者 150) 相关联的微体系结构状态、所述 VM (例如 130、140 或者 150) 的状态指示符等等。当所述 VMCS 指针是有效的时，软件不使用普通的存储器读和写来访问 VMCS 区域 118。此限制确保与所述 VMCS 相关联的 VMM 120 和 VM (例如 130、140 或者 150) 的操作的正确性。

另一处理器提供的指令可以把在处理器上的资源中高速缓存的任意 VMCS 数据写入 VMCS 区域 118，使处理器上的存储空间中的适当数据无效，并且此外使所述 VMCS 指针失效。一个实施例可以例如通过把状态指示符写入 VMCS 区域来把所述 VMCS 标记为无效，所述状态指示符是从体系结构

上或者非体系结构上定义的。另一实施例可以仅仅转储清除处理器上的存储空间并且使 VMCS 的处理器上的状态失效(无效)。在 VMM 120 试图移动 VMCS 区域 118 的内容之前执行此指令。当 VMM120 试图移动 VMCS 区域 118 时，它可以利用单个数据块移动来执行，这是因为所述 VMCS 格式被维护并且只有该处理器实现方式知道。另外，因为所述 VMM 知道(具有基于这里描述的机制分配的存储器的)VMCS 区域的大小，所以这种块移动是可能的。所述 VMCS 的个别元件不必由 VMM 120 识别出。

在一些实施例中，专用于特定实施例的机制可以允许 VMM 120 识别建立特定的 VMCS 的处理器实现方式。所述 VMM 120 可以使用此处理器识别结果来确定所述 VMCS 是否与另一处理器实现方式兼容。在一个实施例中，此标识符位于 VMCS 区域 118 的最初 4 个字节中，并且例如依照 MSR 或者通过任意其他方法被报告给 VMM 120。在第一次使用 VMCS 之前，需要所述 VMM 120 把此标识符写到 VMCS 中的适当位置处，并且因此可以在体系结构上定义 VMCS 区域 118 的部分的格式。在一些实施例中，所有实现方式定义 VMCS 区域 118 的相同体系结构上定义的部分是合乎需要的。

如果先前的实施例留下新定义的区域没有定义，其他实施例可以定义 VMCS 区域 118 的新的在体系结构上定义的部分，并且最新定义的部分不妨碍没有明确使用它们的 VMM 120 的操作。以这种方式，对这些实施例的任意 VMM 120 写入将在更新的实现方式上操作。

在一些实施例中，所述处理器提供的指令 119 包括多个不同的指令：

- VMCS 载入指针指令具有作为操作数的存储器地址。此地址是处理器 112 可以存储 VMCS 的位置(例如，VMCS 区域 118 的开始)。通过指令使与此地址相关联的所述 VMCS 有效。此指令还允许激活与 VMCS 相关联并且由其控制的 VM(例如 130、140 或者 150)并且允许使用如下所述的各种其他指令。在一个实施例中，所述地址是物理地址。其他实施例可以使用虚拟或者线性地址。

- VMCS 存储指针指令把指向有效 VMCS 区域 118 的指针存储到(例如处理器 112 上的)寄存器或者(例如存储器 116 中的)存储器单元中；这种存储单元的位置可以作为操作数提供给指令。

- VMCS 读指令把所述 VMCS 的组件读取到(例如处理器 112 上的)寄存器或者(例如存储器 116 中的)存储器单元中。所述指令可以包括多个参数，该参数包括表明从 VMCS 被读取的 VMCS 组件的组件标识符以及将要存

储的从所述 VMCS 中读取的数据的位置(例如寄存器或者存储器单元)。

- VMCS 写指令载入来自于(例如处理器 112 上的)寄存器或者(例如存储器 116 中的)存储器单元中的所述 VMCS 的组件。就像所述 VMCS 读指令那样, VMCS 写指令可以包括针对 VMCS 组件标识符的操作数。另外, 它可以包括描述要被写入所述 VMCS 的数据的位置(例如寄存器或者存储器单元)的操作数。

- VMCS 清除指令确保与所述 VMCS 相关联的所有处理器上的存储空间的内容被存回 VMCS 区域 118, 使处理器上的资源中的数据无效并且此外使 VMCS 指针失效。此指令可以在没有操作数的情况下操作, 转储清除并且使当前有效的 VMCS 指针失效, 或者在可替代的实施例中, 它可以把用于转储清除和去活所需的 VMCS 指针作为操作数。在一个实施例中, VMCS 清除指令可以通过把状态指示符写入 VMCS 区域来把所述 VMCS 标记为无效, 所述状态指示符是从体系结构上或者非体系结构上定义的。

- 最后, VMCS 输入指令通过依照 VM 体系结构 100 的语义和所述 VMCS 的内容载入处理机状态, 来把控制转递至(即 VM 进入)由有效 VMCS 定义的 VM (例如 130、140 或者 150) 中。可替换地, 所述指令可以包括具有待输入的 VMCS 指针(或者所述指针的位置)的自变量。

由于可以提供各种不同的或者替代的指令 119, 所以如上所述的处理器提供的指令 119 只表示一个实施例。例如, 在一些实施例中, 所述 VMCS 清除指令可以在不使 VMCS 指针失效的情况下把来自于处理器上的存储空间的 VMCS 数据复制到所述 VMCS 区域 118。操作数可以是显式的或者隐式的。可以提供其他的指令 119 (或者上面提出的指令 119 的变形)等等, 所述指令作用于处理器 112 中所有有效的 VMCS 指针。

如本领域普通技术人员所理解的那样, 对存储器中的 VMCS 而言, 常规的 VM 体系结构具有静态的、体系结构上定义的形式。这限制了处理器实现方式扩充或者改变 VMCS 的大小、组织或者内容的能力, 这是因为变成所述 VMCS 往往需要相应地变成所述 VMM 实现方式。另外, 由于常规的 VMM 使用普通存储器读和写操作来访问常规的 VMCS, 所以所述处理器实现方式在把 VMCS 数据的存储空间分配给处理器上的资源方面具有较小的灵活性。例如, 在 VM 退出至 VMM 期间, 因为所述处理器难以检测到所述 VMM 是否对存储器中 VMCS 的对应区域做出修改, 所以所述处理器不能在处理器上的资源中高速缓存某部分 VMCS。这样不能确定处理器上的资源之间

的一致，并且存储器内的 VMCS 图像使错误和一致性检验变复杂，其中所述一致性检验在跟随 VM 进入执行客户软件之前需要执行。

然而，依照本发明的各种实施例，VMM 120 不直接使用普通存储器读和写操作来访问 VMCS 存储器图像，不使用存储器 116 中 VMCS 的预先定义的格式，并且确定在运行时所述 VMCS 在存储器 116 中所需存储空间的大小。

图 2 中举例说明了如上所述的 VMCS 存储空间存储需求的运行时绑定和处理器提供的指令 119 的使用。图 2 通过 VMM 中的各种活动举例说明了 VMCS 指针的状态。处理在块 210 开始。这时，没有有效的 VMCS 指针。在块 210，所述 VMM 确定处理器用于支持所述 VMCS 所需的存储器区域的大小。如上所述，这样可以例如通过读取所指定的 MSR 来实现。

在块 220，所述 VMM 分配所需要的存储器；所述 VMCS 指针仍是无效的。虽然最佳的是，此存储器区域在物理存储器内是连续的，但是对于本领域普通技术人员显而易见的是，依照此实施例这种要求不是必须的。所述 VMM 使用 VMCS 载入指针指令，通过把 VMCS 区域的地址提供给处理器来激活所述 VMCS，如上所述(进入块 230)。在此刻，把所述 VMCS 指针称作工作中 VMCS 指针。在所述 VMCS 指针有效之后，所述 VMM 可以使用适当的 VMCS 访问指令读和写 VMCS 的组件(图 1 中作为 119 示出并且在图 2 中作为 232 和 234 示出)。这些读取或者写操作之后，所述 VMCS 指针保持有效。

当所述 VMM 希望允许客户执行时，它使用 VM 进入指令来把客户加载到机器中(进入块 240)。在 VM 进入之后，所述 VMCS 指针仍是有效的，但是现在起控制 VMCS 指针的作用，所述指针由处理器使用以便确定客户执行环境和行为。

当 VM 退出发生时，控制返回到所述 VMM(从块 240 返回到块 230)。所述 VMCS 指针保持有效，再次作为工作中 VMCS 指针；如适当，所述 VMM 可以读和写 VMCS 字段(箭头 232 和 234)，并且可以使用 VM 输入指令再次输入客户(返回到块 240)。可替换地，所述 VMM 可以使用所述 VM 清除指令来使 VMCS 指针失效，在那时，所述 VMCS 指针成为无效的(返回到状态 220)。

应注意的是，由于所述 VMM 可以及时在任意特定时刻执行各种操作(例如 VM 读取、VM 写入、VM 输入等)，所以方法 200 不局限于任意特定的操作顺序。另外，在使用方法 200 的给定的处理器内的在任意特定时刻，

单个 VMCS 指针或者各种 VMCS 指针可以是有效或者无效的。在一些实施例中，在任意给定处理器内在任意特定时间点，多个 VMCS 可以是有效的。以这种方式，在不执行 VMCS 清除指令的情况下，VMM 可以在 VM 之间切换，由此改善 VM 的处理效率。

在一个实施例中，附加的处理器提供的指令可以向 VMM 提供查询和获得并发或并行 VMCS 的数量的能力，该 VMCS 在任意特定时间点都可以是有效的。其他实施例可以提供 MSR，所述 VMM 可以读取该 MSR 以获得此信息。在可替代的实施例中，同时有效的 VMCS 指针的数量（由此例如可以被高速缓存在处理器上的资源中）未必是 VMM 软件可直接看到的，当 VMM 激活比处理器可以存储在处理器上的资源中的更多的 VMCS 指针时，所述处理器实现方式自动地处理溢出条件。在该情况下，所述处理器可以自动地把适当的 VMCS 数据转储清除至所关联的存储器中的 VMCS 区域。在一些实施例中，在如上依照图 1 论述的多个处理器提供的指令内，可以要求显式的 VMCS 指针自变量。

本发明的实施例消除了让软件管理与管理 VMCS 数据的处理器上和存储器中的存储空间相关联的细节的要求。由此，所述 VMM 在高层抽象处在其控制下管理每个 VM。上述在图 1 和图 2 中所论述的那样，这种策略允许处理器提供的指令管理 VMCS 存储空间的细节，以致 VM 执行更加有效，并且可以在不改变现有 VMM 软件的情况下、通过添加或修改所述处理器提供的指令来被最佳地改变或者扩展。

图 3 是在本发明的一个实施例中描述 VMCS 读/写指令的使用的图表。示出了体系结构 VMCS 格式 305，不过它不是由 VMM 明确使用的；相反，所述 VMM 利用对 VMCS 组件标识符、字段大小和字段语义的认识来编程，但是不依照存储器中 VMCS 存储空间的形式的结构限定来编程。当 VMM 希望访问体系结构 VMCS 305 中的字段时，它执行 VMCS 读取（或者如适当则执行 VMCS 写入；为了举例说明的目的，图 3 中只示出了 VMCS 读取）。VMCS 读指令 310 的参数是组件标识符，该组件标识符是识别 VMCS 组件的体系结构上定义的常数。所述处理器使用所述组件标识符来如恰当的话使用映射函数 320 映射至处理器（例如 VMCS 高速缓冲存储器 322）上的资源或者存储器（例如 VMCS 区域 324）内的资源，以便访问代表所述 VMM 的 VMCS 数据。如果存储在 VMCS 区域或者 VMCS 高速缓冲存储器中的数据不是体系结构上定义的格式，那么使用重新格式化函数 323 把所述数据适当地重新格

式化以便匹配。

图 3 中示出了两个实例 VMCS 访问指令。虽然这两个例子用于 VMCS 读指令，不过如下面就图 5 的论述中详细说明的那样，类似的活动集也发生在 VMCS 写指令上。

参考图 3，第一实例 VMCS 读取指令 310 访问所述 GUEST_EIP 组件，其具有体系结构上定义的编码 0x4032。所述处理器实现方式把这种组件的数据保持在 VMCS 区域 324 中，如映射函数 320 所确定的那样。所述处理器使用存储器读操作 330 从 VMCS 区域 324 中读取适当的组件数据。所述读操作 330 了解组件在 VMCS 区域 324 中的位置(此外，如映射函数 320 所确定的那样；在此例子中，它位于 VMCS 区域中的偏移 12 处)。然后，所述处理器把数据值返回至所述 VMM (如箭头 331 所示)。

第二实例 VMCS 读指令 350 访问所述 VM_CONTROLS 组件，其具有体系结构上定义的编码 0x1076。这种处理器实现方式把这种组件保持在所述 VMCS 高速缓冲存储器 322 中(如映射函数 320 所确定的那样)。所述处理器访问所述 VMCS 高速缓冲存储器 322 以便检索数据。在该情况下，把所述数据以不与 VM_CONTROLS 组件的结构限定匹配的形式存储在处理器上(例如，把数据作为重新排序的 8 字节对象存储在处理器上，而 VM_CONTROLS 字段的结构限定是 4 字节对象)。在把数据返回至 VMM 之前，所述处理器使用重新格式化函数 323 重新格式化所述 VM_CONTROLS 数据以便匹配所述字段的结构限定。把此重新格式化的值(即，匹配所请求的组件的结构限定的值)返回到所述 VMM (显示为箭头 326)。

使用这些 VMCS 访问指令，当从存储器高速缓存至处理器上的资源时，不需要管理每个都由独立的 VMCS 标识的一个或多个 VM 的 VMM 来管理所述存储空间，该存储空间是与 VMCS 相关联的。另外，所述 VMM 不需要管理改变处理器实现方式的细节，诸如存储器中或者处理器上的资源中的数据的格式化。由此，处理器开发人员可以改变或者修改 VMCS 访问指令，以便改善 VM 性能并且扩展 VM 能力，而不会对现有 VMM 或者 VM 的操作带来负面影响。然而，现有 VMM 和 VM 可以受益于这种变化，所述变化诸如是性能、可靠性、可缩放性或者其他改善，这些给 VM 体系结构带来了新的实现方式。

图 4 是依照本发明一个实施例的用于执行 VMCS 读指令的方法 400 的流程图。在 410，从 VMM 接收 VMCS 读请求。在 420，从所述 VMCS 读请求

获取 VMCS 组件标识符。所述组件标识符是体系结构上定义的常数，该常数识别 VMCS 的所要求的组件。

在 430，处理器检查所述组件标识符，以便确定与所述组件标识符相关的 VMCS 组件是否在存储器中。如果所述 VMCS 组件是存储在存储器中，那么在 440 计算所述存储器位置的地址，并且在 450 获取与 VMCS 组件相关的 VMCS 数据。然而，如果所述 VMCS 组件没有存储在存储器中，那么在 460，所述处理器从处理器上的存储空间中获取与所述 VMCS 组件相关联 VMCS 数据。以这种方式，所述处理器使用所述组件标识符来把读请求映射至 VMCS 组件存储空间，而不管其是驻留在存储器中还是处理器上的存储空间中。

在 470，通过所述处理器做出检验以便确定在把 VMCS 数据返回至所述 VMM 以前是否需要重新格式化 VMCS 数据。如果所述 VMCS 数据以不同于 VMCS 组件的体系结构上定义的数据格式的格式来(要么在 VMCS 区域中或者在处理器上的资源中)存储(即，所述 VMCS 组件是以不同于体系结构上定义的格式的实现方式特定的数据格式存储的)，那么在 480，访问重新格式化函数以便把实现方式特定的 VMCS 数据翻译为体系结构上定义的数据格式。如果所述 VMCS 数据以 VMCS 组件的体系结构上定义的数据格式来(或者在 VMCS 区域中或者在处理器上的资源中)存储，那么不需要发生对 VMCS 数据的重新格式化函数或者翻译。最后在 490，把(现在为体系结构上定义的数据格式的)所述 VMCS 数据返回到所述 VMM。

图 5 是依照本发明一个实施例的用于执行 VMCS 写指令的方法 500 的流程图。在 505，处理器从 VMM 接收 VMCS 写入请求。所述处理器获取作为写入请求的操作数之一的组件标识符，如在 510 中所述。此外，在 515，所述处理器获取希望被写入所述 VMCS 的写数据作为另一操作数。此写数据具有被写入的 VMCS 组件的体系结构上定义的数据格式。

在 520，做出检验以便确定所述 VMCS 组件是否通过所述处理器以体系结构上定义的数据格式来(在 VMCS 区域中或者在处理器上资源中)存储。据此，如果所述 VMCS 组件没有以体系结构上定义的数据格式来存储，那么在 525，执行数据的适当的重新格式化，以便使所述数据具有适当的实现方式特定的数据格式，该数据格式与被访问的 VMCS 组件相关联。存储数据的格式可以和体系结构上定义的格式一致，或者它可以在大小或组织方面不同。

接下来在 530，执行另一检验以便确定处理器存储所述 VMCS 组件的位置。使用所述组件标识符，所述处理器确定所述 VMCS 组件的存储位置。如果所述存储位置位于存储器中，那么在 535，计算所述存储器地址，并且在 540，把写数据写入所述存储位置。然而，如果所述存储位置不在存储器中，那么在 545，确定处理器上的存储空间中的适当位置，并且把写数据写入所述处理器上的存储空间。写入所述存储空间的数据具有实现方式特定的数据格式，该数据格式可以与体系结构上定义的格式相同或者不同，这取决于所查询的所述 VMCS 组件、所述处理器实现方式并且在一些实施例中，这取决于所查询的组件当前是存储在处理器上的资源中还是存储在所述 VMCS 区域中。

用于确定特定 VMCS 组件的存储位置并且用于确定所述 VMCS 组件是否是由处理器以体系结构上定义的数据格式来存储的过程 400 和 500 中使用的机制是实现方式特定的。在一个实施例中，所述处理器使用由所述组件标识符索引的查找表。另外，应注意的是，如果多个 VMCS 指针可以同时有效，那么这些确定可以取决于 VMCS 正被访问。

当阅读和领会此公开内容时，本领域普通技术人员将理解这样的方式，其中本发明可以在基于计算机的系统中实现，以便执行此处公开的方法。本发明可以使用（诸如在通用计算机系统上或者专用机上运行的）软件、硬件（例如电路、专用逻辑、可编程逻辑、微代码等）或者硬件和软件的组合来实现。

应该理解的是，上述描述是例证性的，而不是限制性的。当回顾上述描述时，许多其他的实施例对于本领域中普通技术人员是显而易见。因此，本发明的实施例的范围应该参照所附权利要求以及与被授予这种权利要求等效的全部范围来限定。

需要强调的是，遵照 37 C.F.R. § 1.72 (b) 的规定提供了摘要，该规定要求摘要允许读者快速确定技术公开的特性和要旨。提交所述摘要是基于这样的认识，即不会使用它来解释或者限制权利要求的范围或者意义。

在上述实施方式的描述中，为了简化公开内容，在单个实施例中集合了各种特征。所公开的这种方法不应该被解释为反映这样的意图，即所要求保护的本发明的实施例要求比在每个权利要求中清楚讲述的特征更多的特征。相反，如随后权利要求书反映的那样，发明的主题在于少于单个

公开的实施例的所有特征。由此如下的权利要求被并入所述实施方式的描述中，每个权利要求坚持其自身作为独立的示例性实施例。

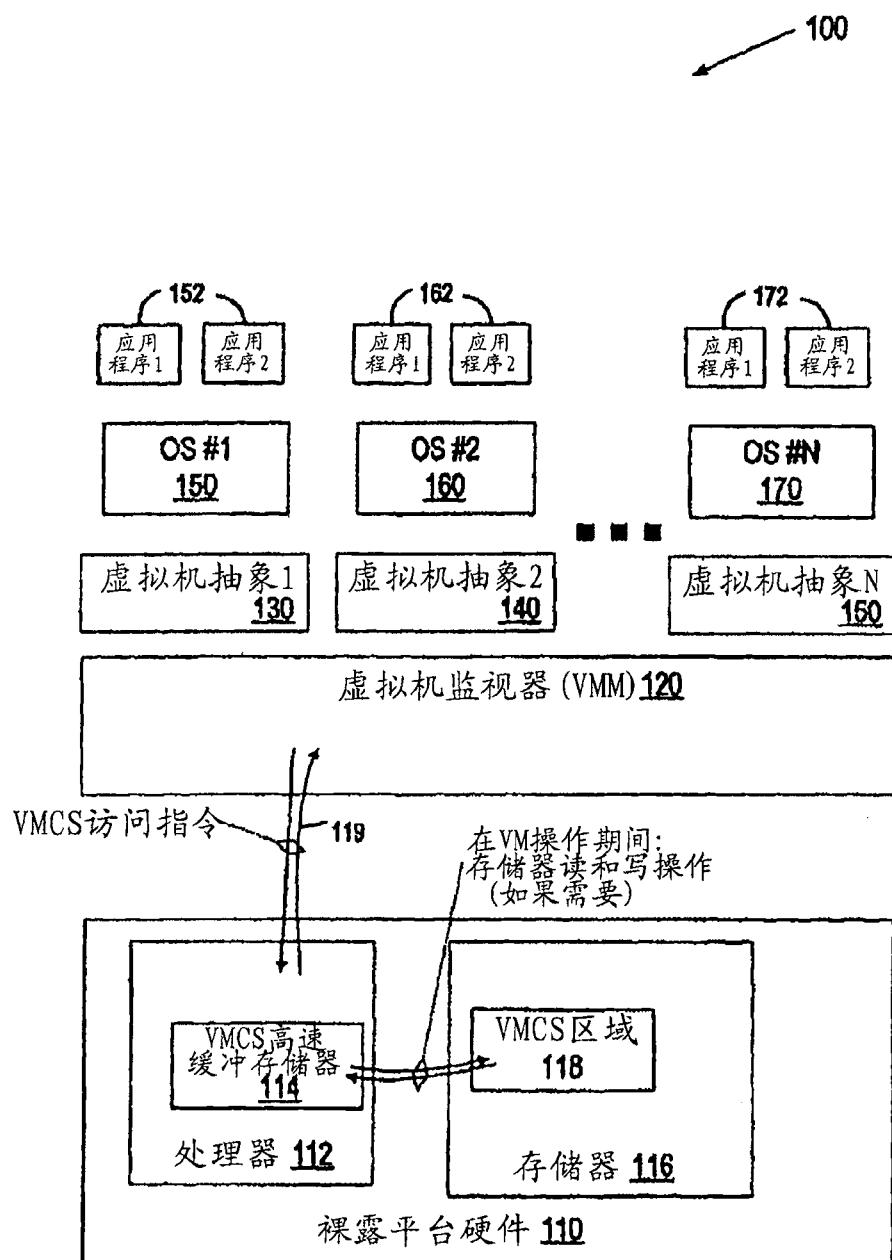


图 1

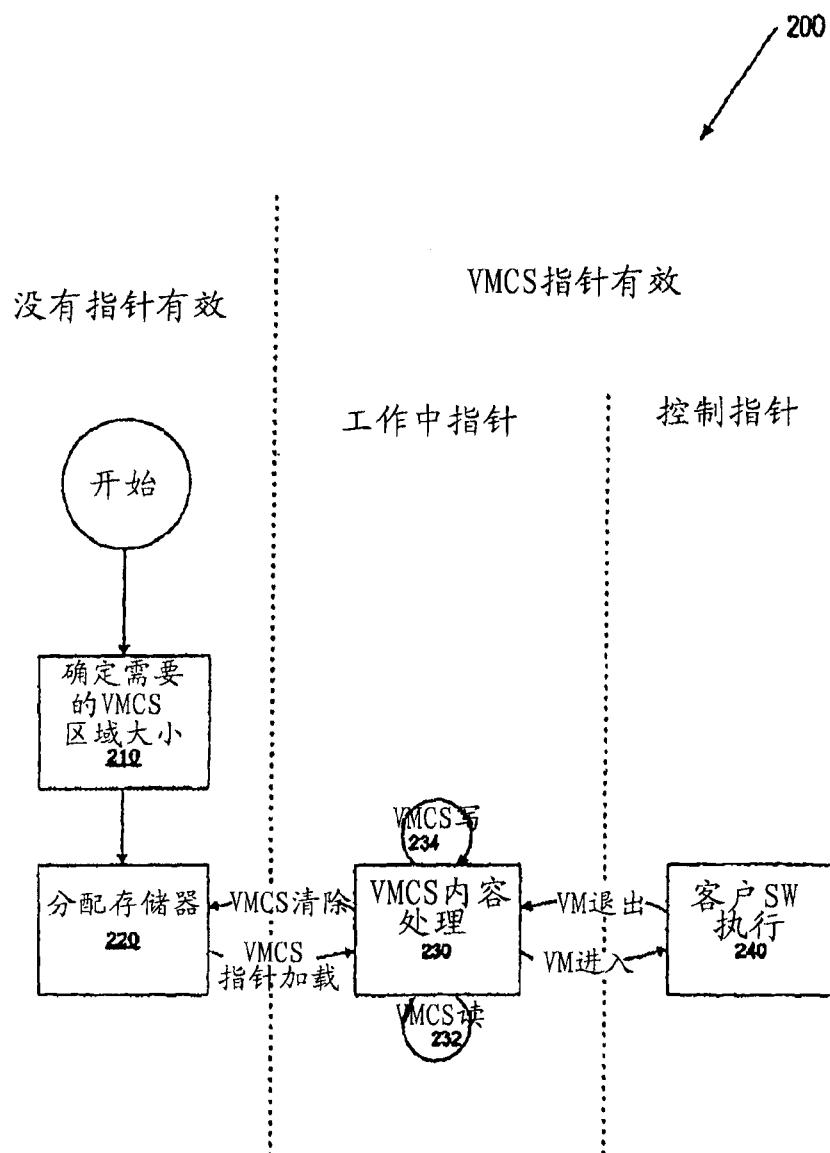


图 2

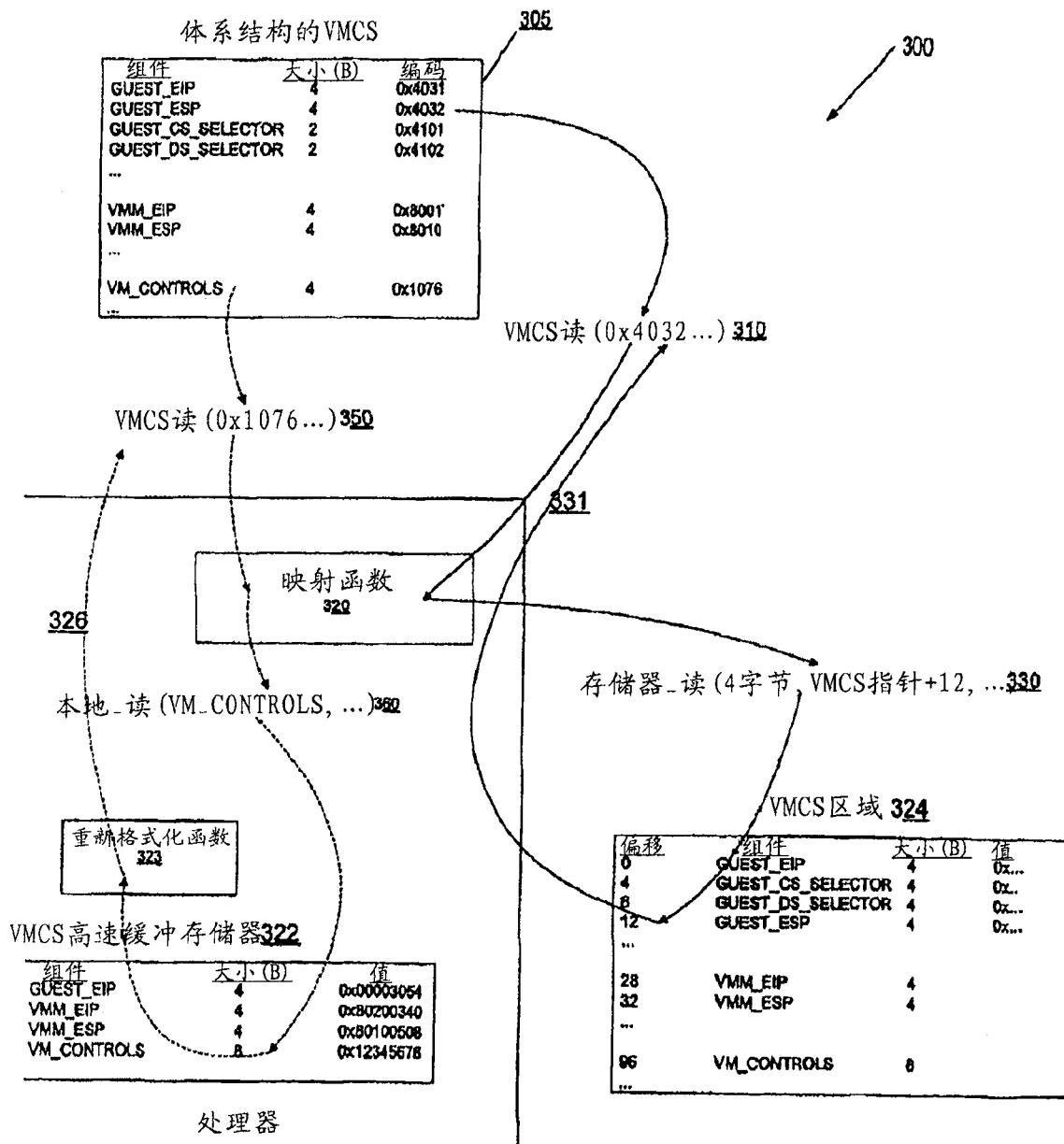


图 3

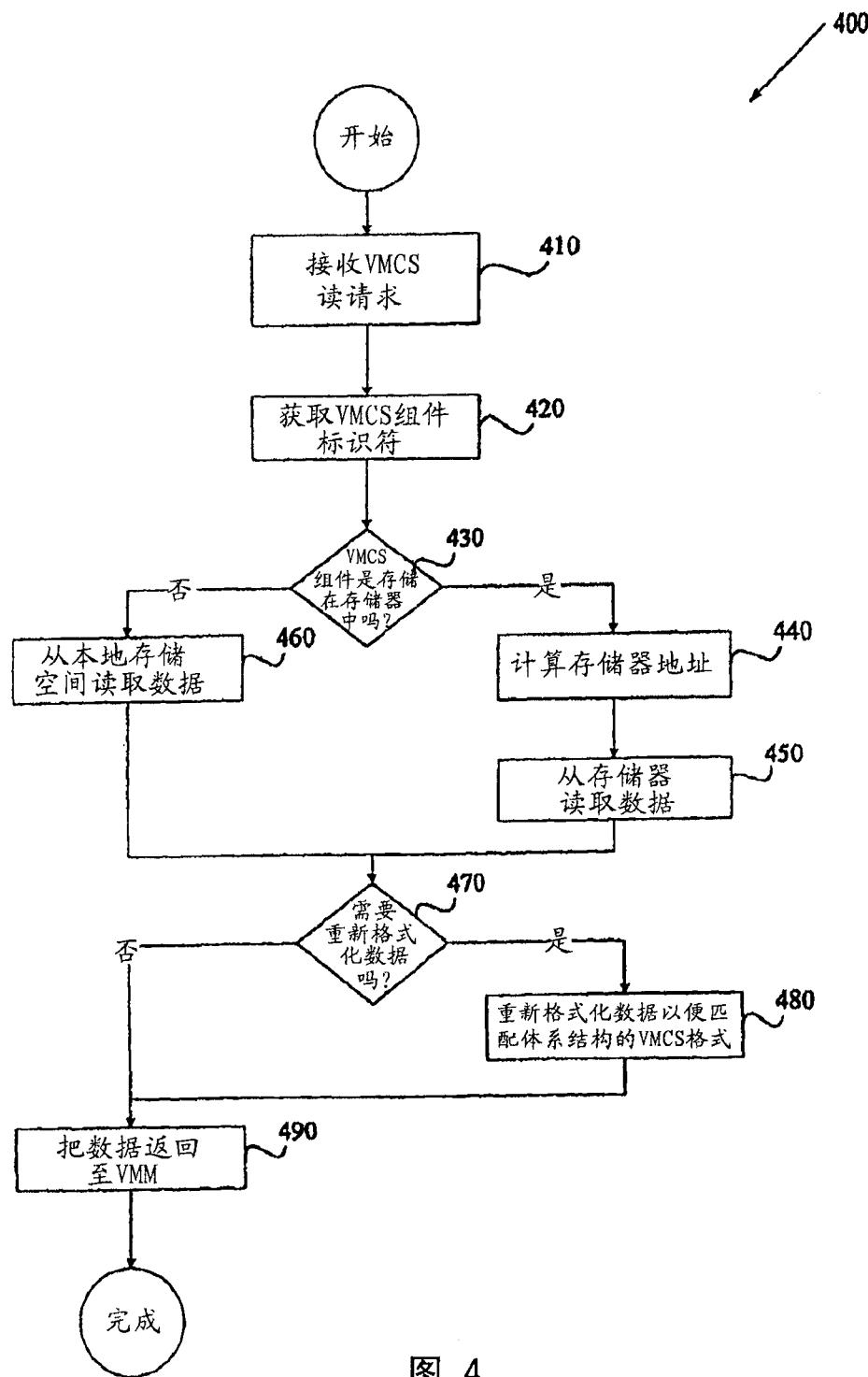


图 4

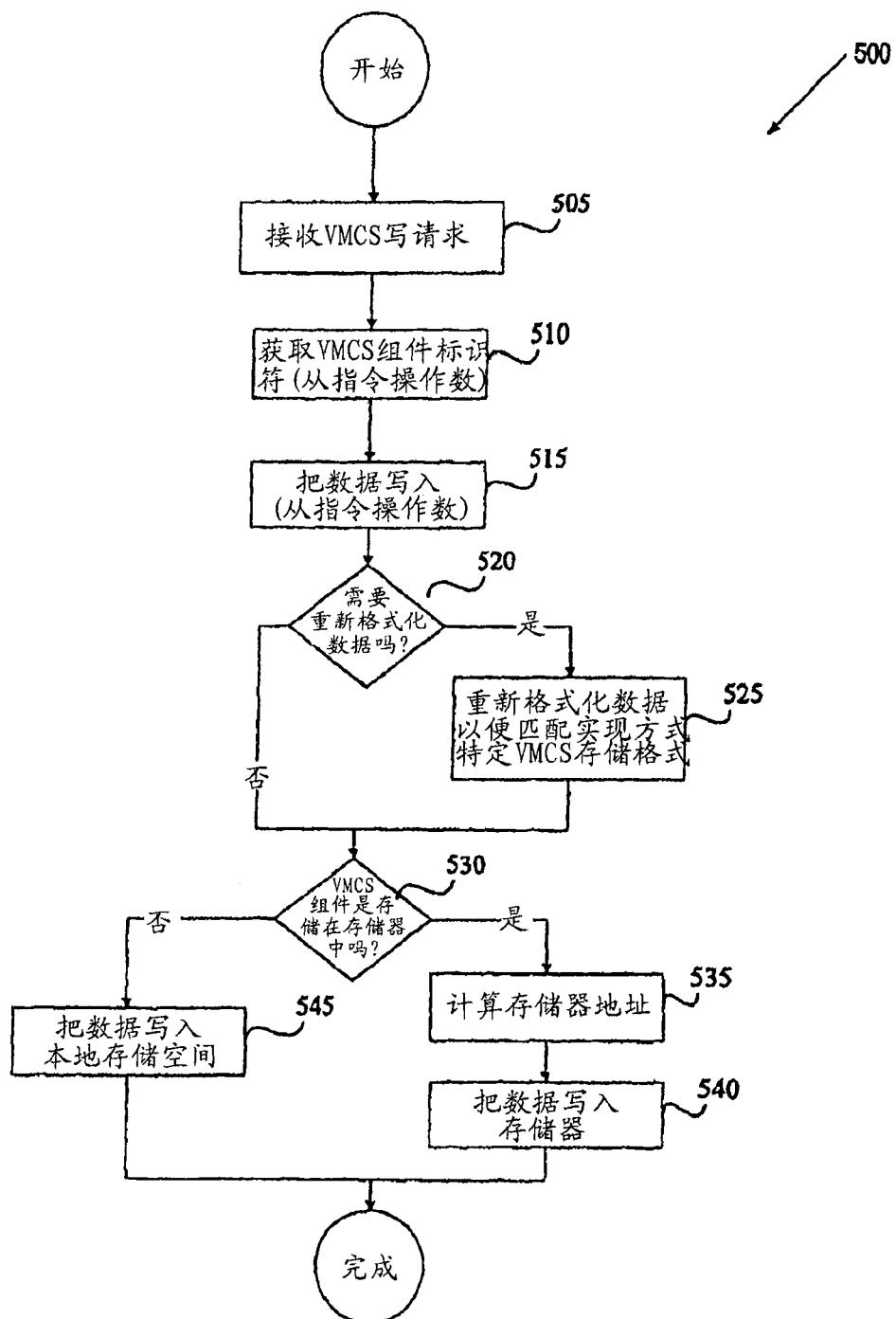


图 5