



(12) 发明专利

(10) 授权公告号 CN 103176863 B

(45) 授权公告日 2016.06.01

(21) 申请号 201310067346.9

审查员 李艳丽

(22) 申请日 2007.05.16

(30) 优先权数据

11/439485 2006.05.22 US

(62) 分案原申请数据

200780018483.6 2007.05.16

(73) 专利权人 英特尔公司

地址 美国加利福尼亚州

(72) 发明人 S. 赖因哈德特 S. 穆克赫吉

(74) 专利代理机构 中国专利代理(香港)有限公司

司 72001

代理人 汤春龙 朱海煜

(51) Int. Cl.

G06F 11/14(2006.01)

(56) 对比文件

US 5802266 A, 1998.09.01,

CN 1350675 A, 2002.05.22,

CN 1575453 A, 2005.02.02,

CN 1755636 A, 2006.04.05,

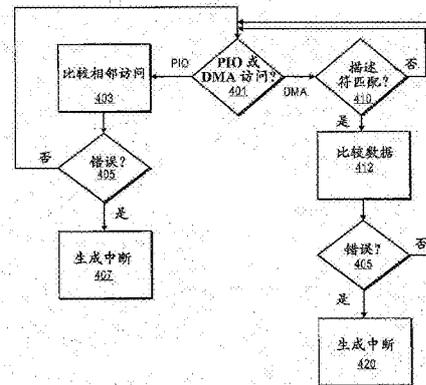
权利要求书2页 说明书6页 附图4页

(54) 发明名称

使用冗余虚拟机的错误检测

(57) 摘要

本发明的发明名称为使用冗余虚拟机的错误检测,其提供了一种在计算机系统中检测错误的技术。更具体地,本发明的至少一个实施例涉及使用冗余虚拟机和比较逻辑以在计算机系统中检测发生在输入/输出(I/O)操作中的错误。



1. 一种使用冗余虚拟机进行错误检测的装置,包括:  
用于比较对应于输入 / 输出 (I/O) 设备的至少两个冗余访问的数据以便确定与所述至少两个冗余访问中的任一个相关的错误是否发生的逻辑,  
其中所述至少两个冗余访问将由两个或更多的冗余虚拟机 (RVM) 生成,并且  
其中所述逻辑还适合复制由所述 I/O 设备提供给所述两个或更多的冗余虚拟机的输入。
2. 如权利要求 1 所述的装置,还包括两个或更多冗余访问接口存储区域以存储对应所述至少两个冗余访问的信息。
3. 如权利要求 2 所述的装置,其中所述两个或更多冗余访问接口存储区域位于输入 / 输出控制器设备内,而且将存储对应所述输入 / 输出控制器设备的控制信息。
4. 如权利要求 2 所述的装置,其中所述两个或更多冗余访问接口存储区域位于存储器设备内,而且将存储对应所述至少两个冗余访问的数据。
5. 如权利要求 1 所述的装置,其中,如果在所述至少两个冗余访问中的任一个中检测到错误,则将生成中断。
6. 如权利要求 5 所述的装置,其中所述中断要由对应所述两个或更多冗余虚拟机的虚拟机管理器 (VMM) 接收。
7. 一种使用冗余虚拟机进行错误检测的系统,包括:  
处理器,所述处理器的至少一些处理资源要由两个或更多冗余虚拟机 (RVM) 表示;  
输入 / 输出 (I/O) 控制器,包括:接口逻辑以与输入 / 输出设备接口,比较逻辑以比较与来自所述两个或更多冗余虚拟机的两个或更多访问对应的数据;以及  
所述输入 / 输出设备,接收来自所述两个或更多冗余虚拟机的两个或更多访问,  
其中所述比较逻辑还适合复制由所述 I/O 设备提供给所述两个或更多冗余虚拟机的输入。
8. 如权利要求 7 所述的系统,其中所述输入 / 输出控制器还包括输入复制逻辑以生成对应所述两个或更多冗余虚拟机的两组或更多组输入 / 输出控制器接口信息。
9. 如权利要求 8 所述的系统,其中所述两组或更多组输入 / 输出控制器接口信息要在两个或更多寄存器组中存储。
10. 如权利要求 7 所述的系统,还包括存储器以存储来自所述两个或更多访问的数据。
11. 如权利要求 10 所述的系统,其中来自所述两个或更多访问的数据要在所述存储器内的两个或更多缓冲区内存储,所述两个或更多缓冲区对应所述两个或更多访问。
12. 如权利要求 8 的所述的系统,其中所述两个或更多访问对应编程的输入 / 输出 (PIO) 访问。
13. 如权利要求 8 所述的系统,其中所述两个或更多访问对应直接存储器访问 (DMA)。
14. 如权利要求 8 所述的系统,其中,如果由所述比较逻辑检测到错误,则将生成中断。
15. 如权利要求 14 所述的系统,其中所述中断要由对应所述两个或更多冗余虚拟机的虚拟机管理器 (VMM) 接收。
16. 一种使用冗余虚拟机进行错误检测的方法,包括:  
比较对应于输入 / 输出 (I/O) 设备的至少两个冗余访问的数据;  
基于比较来确定与所述至少两个冗余访问中的任一个相关的错误是否发生,其中所述

至少两个冗余访问将由两个或更多的冗余虚拟机 (RVM) 生成。

17. 如权利要求 16 所述的方法,还包括将对应所述至少两个冗余访问的信息存储到两个或更多冗余访问接口存储区域中。

18. 如权利要求 17 所述的方法,其中所述两个或更多冗余访问接口存储区域位于输入 / 输出控制器设备内,所述方法还包括将对应所述输入 / 输出控制器设备的控制信息存储在所述输入 / 输出控制器设备内的所述两个或更多冗余访问接口存储区域中。

19. 如权利要求 17 所述的方法,其中所述两个或更多冗余访问接口存储区域位于存储器设备内,所述方法还包括将对应所述至少两个冗余访问的数据存储在所述存储器设备内的所述两个或更多冗余访问接口存储区域中。

20. 如权利要求 16 所述的方法,还包括:如果在所述至少两个冗余访问中的任一个中检测到错误,则将生成中断,并且发送所述中断到对应所述两个或更多冗余虚拟机的虚拟机管理器 (VMM)。

## 使用冗余虚拟机的错误检测

### 技术领域

[0001] 本公开涉及计算和计算机系统的领域,而且更详细地涉及使用虚拟机监视器的计算机系统中的错误检测的领域。

### 背景技术

[0002] 一些计算机系统对于在操作期间处理错误可能是易受影响的。例如,由于计算机系统暴露于辐射或其它电磁场引起的瞬时错误(“软错误”)可能损坏正在贯穿计算机系统传输的数据,从而引起不正确或非预期的计算结果。例如,软错误可引起计算机系统内不正确的数据在运行于处理器上的软件应用和由该软件应用生成的输入/输出(I/O)数据流之间传递。在该示例中,软错误可存在于应用软件、操作系统、系统软件或I/O数据自身中。

[0003] 计算机系统中软错误的问题已经通过技术定位,例如冗余软件执行,其中将软件的段处理两次或更多次,有时候在不同的处理硬件上,以便产生能相互比较的多个结果以在结果中检测错误。冗余软件处理,虽然在计算机系统中对于检测软错误有些效果,但会需要额外的计算资源,例如冗余的硬件,以冗余地处理软件。

[0004] 用在一些计算机系统中的另一种技术是在软件中虚拟化硬件并且在硬件的冗余虚拟版本内冗余地处理不同的代码段,以便检测软错误。冗余的虚拟硬件,或冗余的“虚拟机”(RVM),能提供底层处理硬件的软件表示,以使得软件代码能冗余地在RVM上并行处理。

[0005] 图1示出了冗余的虚拟机环境,其中软件段,例如软件线程,能冗余地处理以便检测软件中的软错误。具体地,图1示出表示相同处理硬件的两个虚拟机(VM),其中软件线程能冗余地和并行地处理。来自软件线程中的一个或多个操作的冗余拷贝的结果能相互比较,以便在软件线程实际上提交到硬件上下文状态之前或之后检测软错误。

[0006] 然而,为了确保软件在两个VM上等同地处理,通过这些VM的代码的执行路径必须由例如复制管理层(RML)的软件模块控制(或管理)成为相同的。此外,RML可能需要比较两个VM的输出。不幸地是,RML,或等同软件模块,会引入额外的处理开销,这些额外的处理开销会引起计算机系统性能下降。此外,RML自身可能包含软错误,并且因此是不可信任的。

### 附图说明

[0007] 本发明以示例而非限制的方式在附图中示出。

[0008] 图1示出现有技术的冗余虚拟机(RVM)环境。

[0009] 图2示出可与本发明的一个或多个实施例协作使用的计算机系统的组件。

[0010] 图3示出可与本发明的一个或多个实施例协作使用的处理器和输入/输出(I/O)控制器。

[0011] 图4是流程图,示出可在本发明的一个或多个实施例中使用的多个操作。

[0012] 图5是共享总线型计算机系统,其中可执行本发明的一个或多个实施例。

[0013] 图6是点对点计算机系统,其中可执行本发明的一个或多个实施例。

## 具体实施方式

[0014] 本发明的实施例涉及计算机系统。更具体地,本发明的至少一个实施例涉及计算机系统内检测和响应对应输入/输出(I/O)操作的错误的技术。

[0015] 本发明的至少一个实施例使用硬件逻辑执行与利用冗余虚拟机(RVM)检测软错误相关的功能的一部分。更具体地,本发明的一个或多个实施例使用一对指定的存储区域以及对应的输入复制和输出比较逻辑,以检测与在一个或多个处理器和一个或多个 I/O 设备之间的 I/O 数据的传输相关的软错误。

[0016] 在一个实施例中,指定的存储区域包括两个或更多寄存器组,这些寄存器组位于 I/O 控制器内或否则与 I/O 控制器相关,以存储在两个或更多虚拟机和 I/O 设备之间通信的数据。在一个实施例中,指定存储区域也可包括存储器的两个或更多段(例如,VM 缓冲区),以存储与在存储器和 I/O 设备之间的直接存储器访问(DMA)操作相关的数据。

[0017] 本发明的实施例可合并 I/O 控制器设备内的逻辑或否则与 I/O 控制器设备相关的逻辑,以执行由现有技术的 RML 执行的多个功能。例如,在一个实施例中,与表示处理硬件资源的两个或更多 RVM 相关的 I/O 控制器内的逻辑,可用来复制由 I/O 设备提供给 RVM 的输入并且比较由 RVM 生成的输出,以便确定软错误是否已经发生。有利的是,在硬件逻辑中包括输入复制和/或输出比较功能性的实施例能提高处理吞吐量,减少软件开销,并且减少软错误影响软错误检测处理的机会。

[0018] 图 2 示出计算机系统的组件,其中可实现本发明的一个实施例。具体地,图 2 示出 CPU201,CPU201 包括两个 RVM205、210 以表示 CPU 的多种处理资源。此外,图 2 包括含有 I/O 控制器 215 的 I/O 控制器以在 CPU(与 RVM)和一个或多个 I/O 设备 220 之间接口数据。在图 2 中还包括与 I/O 控制器相关的控制寄存器中的至少一些寄存器的两个表示 225、227。在一个实施例中,这两个表示分别对应不同的 RVM 并且用来存储控制信息,控制信息被 RVM 用来将数据发送到 I/O 控制器或从 I/O 控制器接收数据。在一个实施例中,这两个表示是 I/O 控制器内的或否则与 I/O 控制器相关的寄存器,然而在其它实施例中,这些表示是在例如 DRAM 的存储器结构中的位置。

[0019] 位于图 2 的 I/O 控制器内的还有输入复制和输出比较逻辑 230,以生成对应 I/O 控制器的控制接口信息以及比较 RVM 的输出和响应 RVM 执行与输入相关的任务而产生的 RVM 的输出。例如,在一个实施例中,对于将由 RVM 执行的给定的软件操作,对应 I/O 控制器的控制接口信息可存储在 I/O 控制器内的寄存器组中,或存储在否则与 I/O 控制器相关的寄存器组中,而且 RVM 的输出数据可由比较逻辑相互比较以确保没有发生损坏输出的软错误。而且,从 I/O 设备返回且将被发送至 RVM 的信息也可使用比较逻辑 230 复制,以便确保两个 RVM 接收到相同的数据,从而维护 RVM 之间的一致性。相似地,可比较 RVM 上正在执行的操作所产生的结果以确保在这些操作的执行或结果数据自身中没有发生软错误。

[0020] 在一个实施例中,如果比较的结果指示输出数据不一致,错误纠正逻辑或软件或两者能被调用以处置错误并从错误恢复。例如,在一个实施例中,响应正在检测的错误而调用软件处理程序(handler),它能防止错误将处理硬件置于不正确的状态中,或如果硬件已经被置于不正确的状态中,则将硬件置于正确的或已知的状态中。在处理程序从软错误恢复后,在一个实施例中,其中发生软错误的操作可能被再次执行。

[0021] 在一个实施例中,图 2 的 I/O 控制器通过在 I/O 设备上执行 I/O 操作前等待对复制寄存器组的相同的访问,便于对 PIO 访问的 RVM 的输出进行比较。在一个实施例中,PIO 操作可包括 PIO 写和 / 或与 PIO 读操作相关的副作用操作 (side effect operations) (如果有的话)。

[0022] 在未缓存的 I/O 读和写的情况下,它们可以非前瞻地 (non-speculatively) 和以程序顺序执行,来自一个 RVM 的设备寄存器访问可对照来自另一个 RVM 的程序顺序中的紧接的下一个设备访问而得到验证。为了防止一个 RVM 在每个访问可得到验证前发出几个 I/O 设备访问,在一个实施例中,I/O 设备可响应直到另一个 RVM 的访问已经发生的一个 RVM 的访问 (例如,使用总线级重试响应) 而推迟。如果随后的 RVM 的访问没有在特定时间限制 (在一个实施例中是可编程的时间限制) 内到达, I/O 设备可以总线错误来响应,该总线错误能由与 RVM 相关的 VMM 截取并相应地处理 (即,通过再重试或者将此情况处置为错误)。

[0023] 在一个实施例中,如果随后的 RVM 对 I/O 设备的访问不匹配第一个 RVM 对 I/O 设备的访问,例如因为该访问具有不同的类型,定向到不同的寄存器,或 (在写的情况下) 具有不同的数据值, I/O 控制器也可通过总线错误响应和 / 或中断将错误发信号给 VMM。

[0024] 在一个实施例中,图 2 的 I/O 控制器通过在对应的访问上将相同的值返回给两个 RVM 而支持对于 PIO 访问的输入复制。例如,对于不具有副作用的设备寄存器读,或者对于返回值与副作用无关的读,如果缓存响应值,则设备可响应较早的 RVM 访问以使得响应随后的 RVM 访问而返回一致的值,即使设备的内部状态在此期间改变。同样,如果未缓存的 I/O 读和写以非前瞻地和以程序顺序执行,则在一个实施例中,对 PIO 读的响应可关于 RVM 内的程序流程同步。因此,在这样的实施例中,设备不需要参与响应的具体定时。

[0025] 图 3 示出与本发明的至少一个实施例相关的多个组件,其中通过 DMA 传输从 I/O 设备传输信息或将信息传输到 I/O 设备。具体地,图 3 示出 CPU301,对于 CPU301,两个或更多 RVM (未示出) 可用于表示多个资源。在图 3 中也示出存储器 305,它可用于存储通过存储器控制器 310 和 I/O 控制器 315 而在两个或更多 RVM 和 I/O 设备 320 之间通信的信息。具体地,存储器 305 可以是 DRAM,例如,其中缓冲区 325 可以指定为对应 RVM 其中之一而缓冲区 330 可以指定为对应另一个 RVM。

[0026] 正如图 2 中所示的范例,输入和 / 或输出比较逻辑可包括在 I/O 控制器 315 之内,或者否则与 I/O 控制器 315 相关,以比较对应由 RVM 正在执行的软件操作的输入和 / 或输出。此外, I/O 控制器控制信息可由对应两个或更多的 RVM 的两个或更多寄存器组 (未示出) 表示,如图 2 中所示的范例。然而,在 DMA 的情况下,与 PIO 访问相反,从 RVM 写到 I/O 设备的数据或从 I/O 设备写到 RVM 的数据首先存储在对应的 RVM 缓冲区 (325 或 330) 中。

[0027] 在一个实施例中,如果为虚拟化的 I/O 访问重映射 DMA 地址,RVM 缓冲区可对应相同的物理地址但带有不同的 I/O 重映射上下文。否则,在其它实施例中,缓冲区可位于不同的物理地址处。在一个实施例中,仅仅缓冲区的内容必须被验证或者被复制,因此缓冲区地址的不同可能不是那么重要。

[0028] 在一个实施例中, I/O 控制器内的逻辑通过等待直到它从 RVM 之一接收到描述符数据而在外出的 DMA 传输 (到 I/O 设备) 上执行输出比较。描述符数据可在其中支持 DMA 传输的系统中提供。 I/O 控制器然后可比较数据缓冲区长度和 / 或与第一对 RVM 描述符数据相关的其它参数 (例如,磁盘块偏移)。如果数据缓冲区长度和 / 或其它参数匹配,则 I/

0 控制器可从两个缓冲区取出数据内容并且在逐比特、逐字节、逐字（或一些其它的间隔尺寸）的基础上比较它们。如果两个缓冲区的内容匹配，则在一个实施例中，I/O 操作被验证且转发给设备。如果操作的参数或数据有任何不匹配，这可能是软错误的指示，并且 I/O 控制器可发起中断，该中断将由 VMM 处置。

[0029] 在一个实施例中，在进入的 DMA 传输（来自设备）上的输入复制可以如上描述的输出复制相似的方式处置。在数据传输完成后，在一个实施例中，数据可写入物理存储器两次，在每个对应 RVM 的位置处。

[0030] 在一个实施例中，输入复制可能需要从 I/O 控制器到 CPU 的完成通知。例如，如果 I/O 设备驱动正在轮询 DMA 缓冲区完成情况，则 DMA 传输的不同步特性会使一个 RVM 解释描述符数据以指示 DMA 完成而另一个 RVM 在其执行中的同一个逻辑点却不这样做，因此导致在它们的执行路径中可能的分歧。

[0031] 在一个实施例中，当 RVM 正在执行以及有中断服务例程 (ISR) 正在执行时，阻止 I/O 控制器写描述符完成标志，以便防止上面提到的 RVM 执行路径的分歧。在一个实施例中，在 ISR 执行的执行期间完成的 DMA 缓冲区传输可不写入它们对应的描述符直到 RVM 退出 ISR。在一个实施例中，设备驱动可在进入 ISR 的入口处或退出 ISR 的出口处访问特定的设备寄存器，以便推迟描述符更新。

[0032] 在一个实施例中，不是将描述符信息写到基于存储器的 DMA 描述符域，而是 I/O 控制器可通过增加与存储器中的对应的 DMA 缓冲区相关的计数器发出完成 DMA 请求的信号。在该实施例中，完成通知可然后通过对该寄存器的 PIO 读而发生，从而允许使用如上所述的 PIO 输入复制技术。

[0033] 图 4 是示出可用在本发明的至少一个实施例中的多个操作的流程图。在操作 401 处，确定到 I/O 设备的访问（例如，读或写）是 PIO 访问或 DMA 访问。如果访问是 PIO 访问，则相邻访问可认为是来自两个或更多的 RVM 的冗余访问。因此，来自 RVM 的相邻访问可相互比较以确定访问中错误是否已经发生（在操作 403 处）。在操作 405 处，如果错误发生，中断可被生成并由对应 RVM 的 VMM 处置且做出相应的处理（在操作 407 处）。

[0034] 另一方面，如果确定访问是 DMA 访问，则在操作 410 处，在与来自对应数量的 RVM 的两个或更多访问相关的描述符之间做出比较。在一个实施例中，对应访问的描述符可由例如数据缓冲区长度、偏移信息等的信息组成。如果描述符匹配，则在操作 412 处，随后存储在存储器中对应 RVM 的缓冲区中的数据可相互比较以确定错误是否发生。如果数据中或描述符中发生错误，则在操作 420 处中断被生成且由对应 RVM 的 VMM 以适当的方式处置。

[0035] 图 5 示出前端总线 (FSB) 计算机系统，其中可使用本发明的一个实施例。处理器 505 从一级 (L1) 缓存存储器 510 和主存储器 515 访问数据。在本发明的其它实施例中，缓存存储器可以是二级 (L2) 缓存或计算机系统存储器体系内的其它存储器。此外，在一些实施例中，图 5 的计算机系统可同时包含 L1 缓存和 L2 缓存。

[0036] 在图 5 的处理器内示出的是关于机器状态的存储区域 506。在一个实施例中，存储区域可以一组寄存器，然而在其它实施例中，存储区域可以是其它存储结构。处理器可具有任意数量的处理核。然而，本发明的其它实施例，可在系统内的其它设备内实现，例如单独的总线代理，或遍及系统地分布在硬件、软件或它们的一些组合中。

[0037] 主存储器可在多种存储器源 (memory source) 中实现，例如动态随机访问存储器

(DRAM)、硬盘驱动器 (HDD) 520 或通过网络接口 530 远离该计算机系统、包含多种存储设备和技术存储器源。缓存存储器可位于处理器内或接近处理器,例如在处理器的本地总线 507 上。

[0038] 此外,缓存存储器可包含较快的存储器单元,例如六晶体管 (6T) 单元,或具有大致相等或更快访问速度的其它存储器单元。图 5 的计算机系统可以是例如微处理器的总线代理的点对点 (PtP) 网络,它通过专用于 PtP 网络上的每个代理的总线信号通信。图 6 示出以点对点 (PtP) 配置设置的计算机系统。具体地,图 6 示出处理器、存储器和输入 / 输出设备通过多个点对点接口互连的系统。

[0039] 图 6 的系统也可包括几个处理器,为了清楚仅地示出其中两个:处理器 670、680。处理器 670、680 可分别包括本地存储器控制器中心 (MCH) 672、682 以连接存储器 22、24。处理器 670、680 可通过点对点 (PtP) 接口 650 使用 PtP 接口电路 678、688 交换数据。通过使用点对点接口电路 676、694、686、698,处理器 670、680 可通过单独的 PtP 接口 652、654 与芯片组 690 交换数据。芯片组 690 也可通过高性能图形接口 639 与高性能图形电路 638 交换数据。本发明的实施例可位于具有任意数量处理核的任何处理器内,或在图 6 的 PtP 总线代理的每一个内。然而,本发明的其它实施例,可存在于图 6 的系统内的其它电路、逻辑单元或设备中。此外,在本发明的其它实施例中,可分布遍及图 6 中示出的若干电路、逻辑单元或设备。

[0040] 本文所指的处理器、或根据本发明的实施例设计的任何其它组件,可以在不同阶段设计,从创意到仿真到制作。表示设计的数据可以多种方式表示设计。首先,如在仿真中有用的,可使用硬件描述语言或其它功能描述语言表示硬件。附加地或备选地,可在设计过程的某些阶段产生带有逻辑和 / 或晶体管门电路 (transistor gate) 的电路级模型。此外,大多数设计,在某个阶段,达到它们可用表示多个设备的物理布局的数据建模的程度。在使用常规半导体制作技术的场合中,表示设备布局模型的数据可以是在指明在不同掩模层上多个特征的出现或缺失的数据 (掩模用于生产集成电路)。

[0041] 在设计的任何表示中,数据可以存储在任何形式的机器可读介质中。为传送此类信息而调制或否则生成的光波或电波、存储器、或磁的或光的存储介质,例如磁盘,可以是所述机器可读介质。这些介质的任一个可“携带”或“指示”设计、或用于本发明的实施例中的其它信息,例如错误恢复例程中的指令。当传送指示或携带信息的电载波,就执行电信号的复制、缓存或重传来说,实现新的拷贝。因此,通信供应商或网络供应商的动作可以是实现例如载波的制品的拷贝,从而实施本发明的技术。

[0042] 因此,公开了用于操纵存储器访问 (例如加载或存储) 的技术。虽然描述了特定的实施例,并在附图中示出,应该理解这些实施例仅仅是本广义发明的示意而非限制,而且本发明不限于所示出的和所描述的具体的构造和设置,因为学习本公开的本领域技术人员可实现多种其它的修改。在例如这种进步很快且未来的改进不容易预见的技术领域,在不脱离本公开的原则或所附权利要求的范围的情况下,所公开的实施例在设置和细节方面在启用技术进步的帮助下可以是轻易可修改的。

[0043] 本发明的一个或多个实施例的多个方面可在关于其中可使用本发明的一个或多个实施例的处理器或计算机系统的公布中描述、讨论或否则提到。这些公布可包括,但不限于新闻出版物、杂志、广告牌、或其它报纸或否则可触媒体。具体地,本发明的一个或多个实

施例的多个方面可在互联网上通过网站，“弹出”广告或其它基于网络的媒体公布，不管装有生成该网站或弹出窗口的程序的服务器是否位于美国或其属地内。

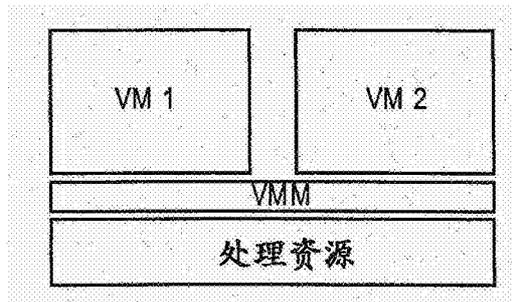


图 1(现有技术)

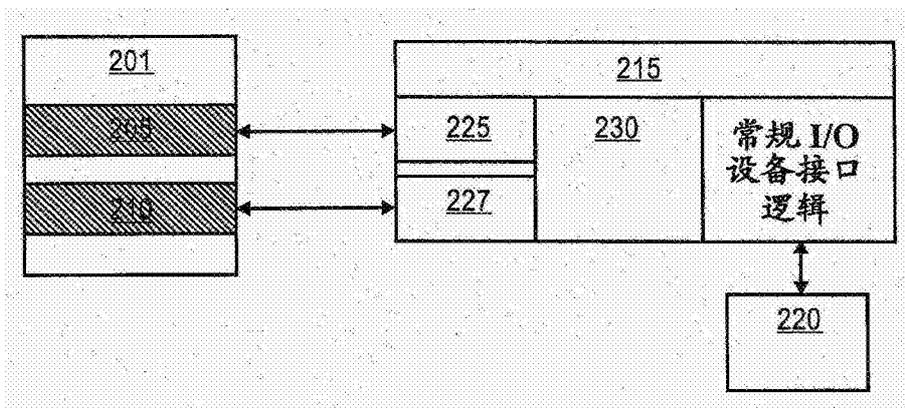


图 2

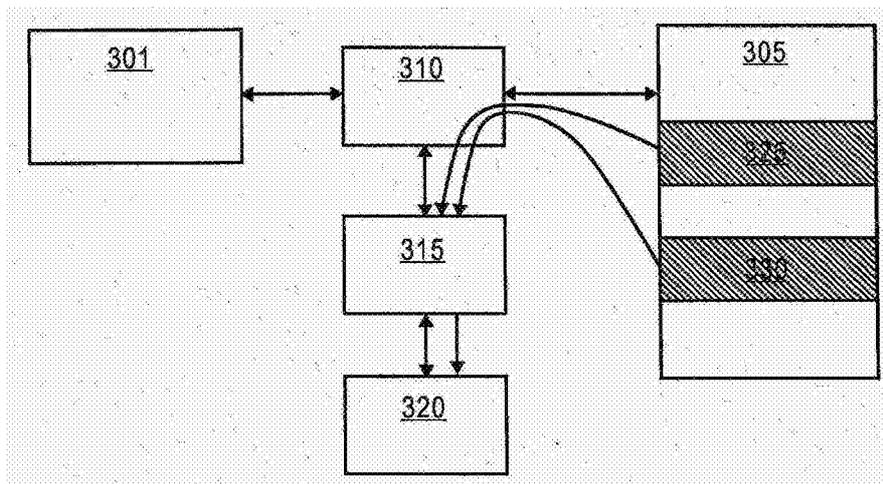


图 3

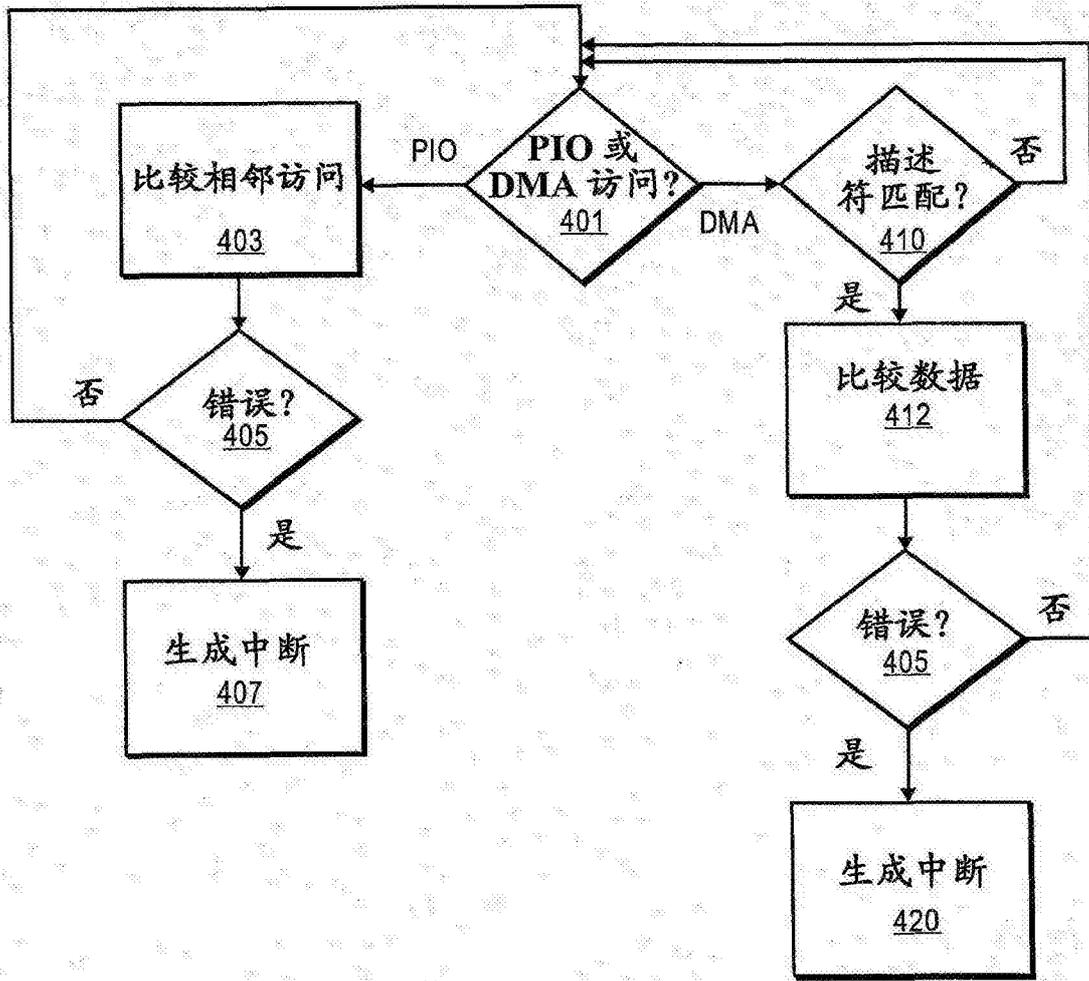


图 4

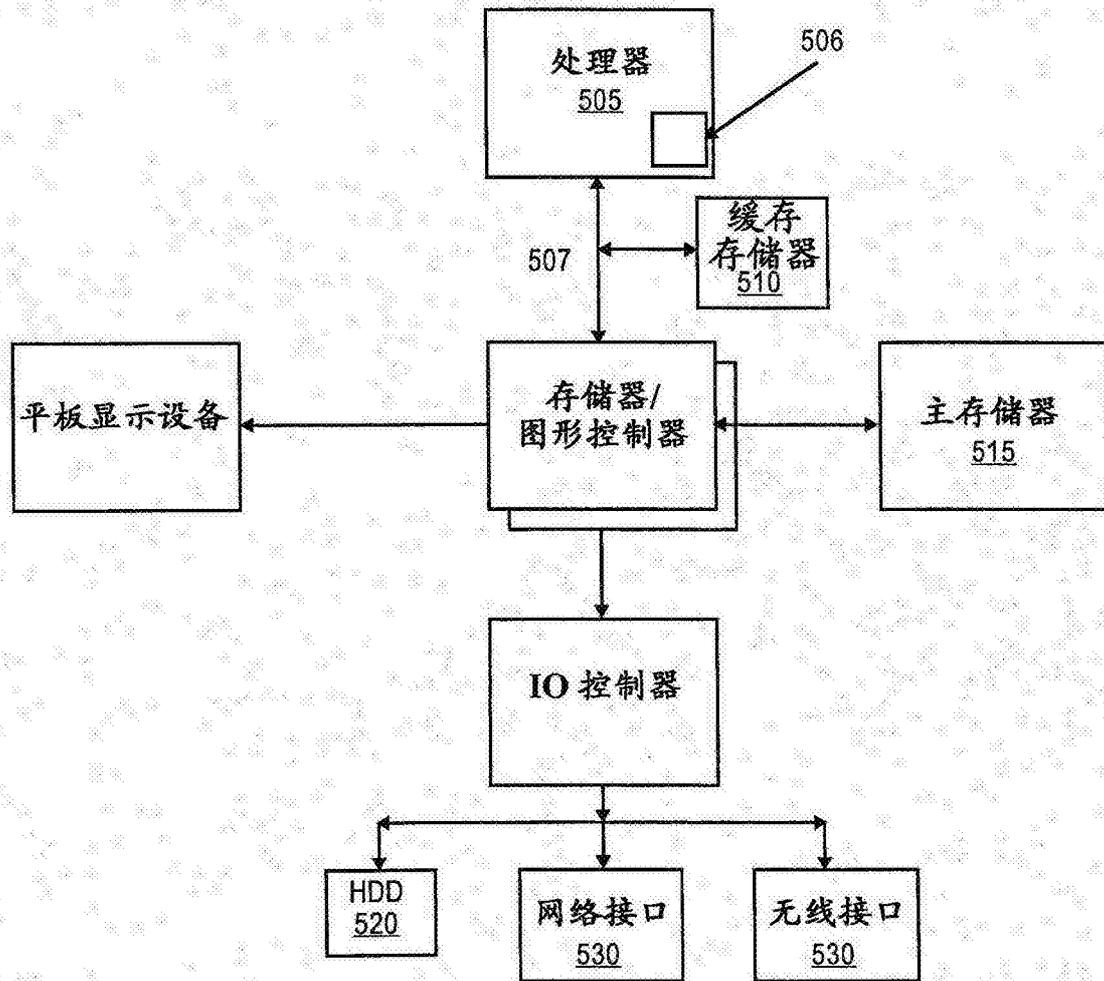


图 5

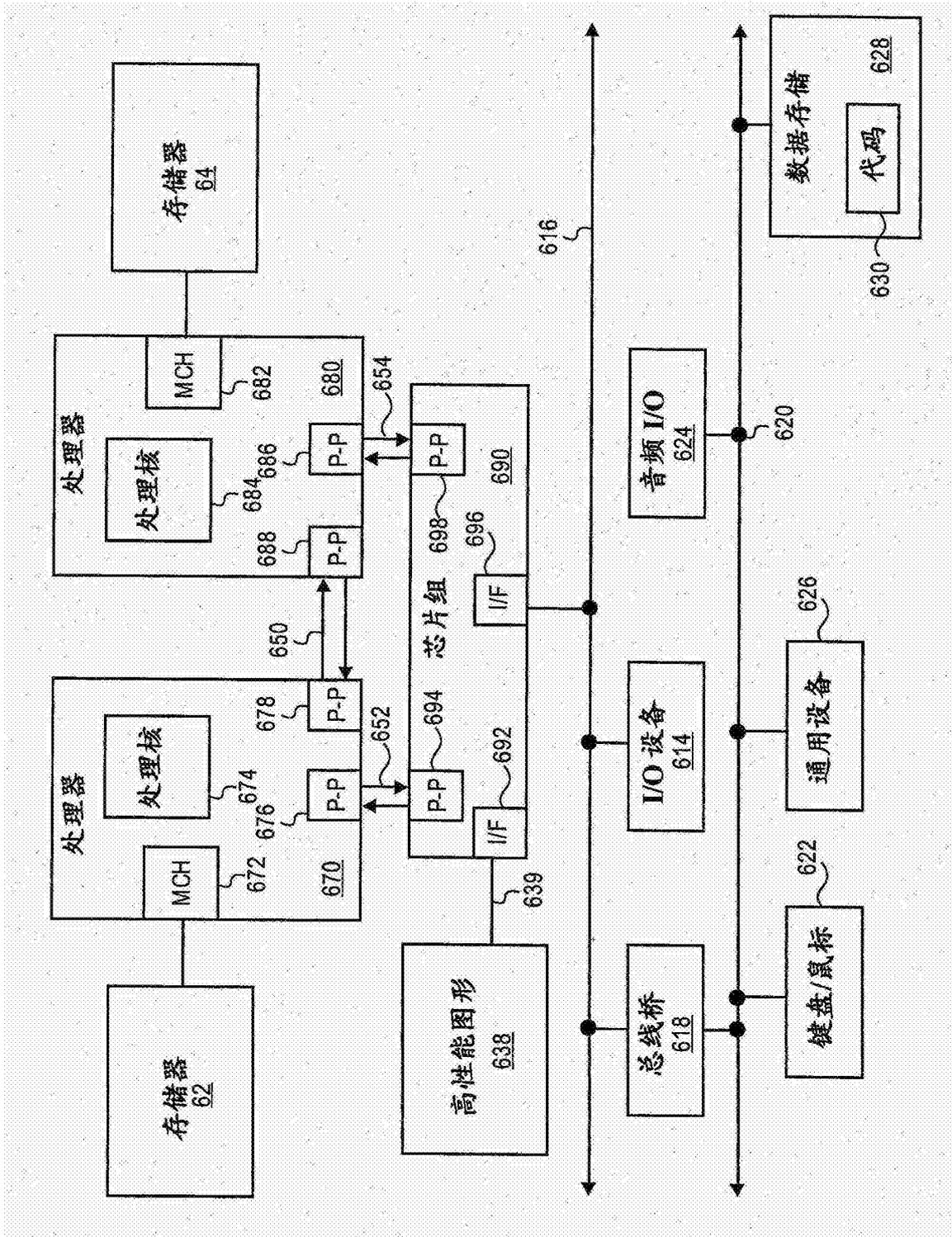


图 6