



(12) 发明专利

(10) 授权公告号 CN 102981808 B

(45) 授权公告日 2015. 07. 29

(21) 申请号 201210472991. 4

G06F 9/30(2006. 01)

(22) 申请日 2006. 09. 29

G06F 9/46(2006. 01)

(30) 优先权数据

11/239475 2005. 09. 30 US

(56) 对比文件

CN 101038543 B, 2013. 01. 02, 权利要求 1-14.

(62) 分案原申请数据

200610144490. 8 2006. 09. 29

US 5872963 A, 1999. 02. 16, 全文.

CN 1402844 A, 2003. 03. 12, 全文.

US 6651163 B1, 2003. 11. 18, 全文.

(73) 专利权人 英特尔公司

地址 美国加利福尼亚州

审查员 白桦

(72) 发明人 G. 钦亚 H. 王 P. 塞蒂 S. 考施克

B. 比格比 J. 沈 R. 汉金斯 X. 邹

B. 帕特尔 J. 布兰德特

A. 阿加沃尔 J. 赖德

(74) 专利代理机构 中国专利代理(香港)有限公司
72001

代理人 朱海煜

(51) Int. Cl.

G06F 9/38(2006. 01)

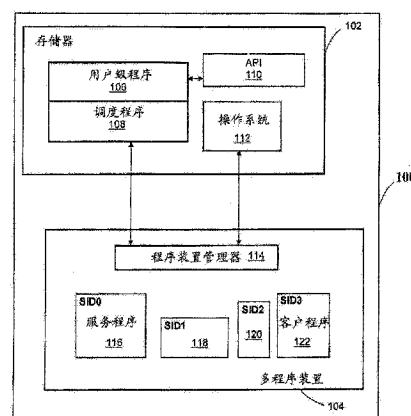
权利要求书2页 说明书11页 附图6页

(54) 发明名称

用于持续性用户级线程的设备、系统和方法

(57) 摘要

本发明的名称是“用于持续性用户级线程的设备、系统和方法”。本发明的实施例提供一种方法,其基于在操作系统可见的程序装置上运行并使用指令集扩展的操作系统调度的线程,来创建持续性用户级线程,以独立于在操作系统调度的线程上的上下文切换活动而在该操作系统隔离的程序装置上运行。所述操作系统调度的线程和所述持续性用户级线程共享一个公用虚拟地址空间。本发明的实施例还提供一种方法,用来使在附加的操作系统可见的程序装置上运行的服务线程向所述持续性用户级线程提供操作系统服务。本发明的实施例还提供其的设备、系统和计算机可读媒体。



1. 一种用于运行持续性用户级线程的方法,包括:

创建持续性用户级线程,以独立于在操作系统调度的线程上的上下文切换活动而在操作系统隔离的程序装置上运行,所述操作系统调度的线程在操作系统可见的程序装置上运行并使用指令集扩展,其中,所述持续性用户级线程与创建所述持续性用户级线程的操作系统线程并发地运行;

当所述持续性用户级线程遇到将要由操作系统处理的事件时,挂起所述持续性用户级线程的运行;以及

使在附加的操作系统可见的程序装置上运行的服务线程向被挂起的持续性用户级线程提供操作系统服务。

2. 如权利要求1所述的方法,其中所述操作系统调度的线程和所述持续性用户级线程共享公用虚拟地址空间。

3. 如权利要求1所述的方法,其中所述服务线程和所述持续性用户级线程在同一个应用程序的上下文中运行。

4. 如权利要求1所述的方法,其中向持续性用户级线程提供操作系统服务包括:

向所述附加的操作系统可见的程序装置传递控制传输指令;并且

在一种特权级上执行所述控制传输指令。

5. 如权利要求4所述的方法,其中所述特权级包括 ring-0 级。

6. 如权利要求4所述的方法,包括:

使所述服务线程获得持续性用户级线程的状态;

向所述持续性用户级线程提供代理执行;和

返回后执行状态到所述持续性用户级线程。

7. 如权利要求6所述的方法,还包括从所述后执行状态重新开始所述持续性用户级线程的运行。

8. 一种用于运行持续性用户级线程的装置,包括:

创建持续性用户级线程,以独立于在操作系统调度的线程上的上下文切换活动而在操作系统隔离的程序装置上运行的部件,所述操作系统调度的线程在操作系统可见的程序装置上运行并使用指令集扩展,其中,所述持续性用户级线程与创建所述持续性用户级线程的操作系统线程并发地运行;

当所述持续性用户级线程遇到需要操作系统处理的事件时,挂起所述持续性用户级线程的运行的部件;

使在附加的操作系统可见的程序装置上运行的服务线程向被挂起的持续性用户级线程提供操作系统服务的部件。

9. 如权利要求8所述的装置,其中所述操作系统调度的线程和所述持续性用户级线程共享公用虚拟地址空间。

10. 如权利要求8所述的装置,其中所述服务线程和所述持续性用户级线程在同一个应用程序的上下文中运行。

11. 如权利要求8所述的装置,其中向持续性用户级线程提供操作系统服务的部件包括:

向所述附加的操作系统可见的程序装置传递控制传输指令的部件;以及

在一种特权级上执行所述控制传输指令的部件。

12. 如权利要求 11 所述的装置,其中所述特权级包括 ring-0 级。

13. 如权利要求 11 所述的装置,包括:

使所述服务线程获得持续性用户级线程的状态的部件;

向所述持续性用户级线程提供代理执行的部件;和

返回后执行状态到所述持续性用户级线程的部件。

14. 如权利要求 13 所述的装置,还包括从所述后执行状态重新开始所述持续性用户级线程的运行的部件。

用于持续性用户级线程的设备、系统和方法

[0001] 本申请是申请日为 2006 年 9 月 29 日、申请号为 200610144490.8、发明名称为“用于持续性用户级线程的设备、系统和方法”的专利申请的分案申请。

技术领域

[0002] 各实施例针对使得服务线程向在多程序装置硬件的上下文中执行的被挂起的持续性用户级线程提供操作系统服务。

背景技术

[0003] 在多程序装置 (multi-sequencer) 的计算机系统或计算机平台中,技术和经济约束证明一种非对称结构的专用计算机资源或处理器或处理单元是合理的。在本申请中,处理器或处理单元在下文也被称为程序装置或处理核。通常,操作系统不能很好地衡量已经实现有很多程序装置的计算机平台。特别是,该操作系统不能高效地处理非对称结构的多个程序装置。

[0004] 对于计算机平台实现的程序装置的类型来说,该计算机平台的结构可以是非对称的。例如,有一种程序装置对操作系统 (OS) 来说是可见的,在这里被称为 OS 可见的程序装置,其由 OS 内核管理。OS 内核能够控制 OS 可见的程序装置的特权状态并提供异常处理以在 OS 可见的程序装置执行线程期间处理中断和 / 或异常。另一方面,还有一种程序装置对 OS 来说是“不可见的”,在这里被称为 OS 隔离 (OS-sequestered) 的程序装置。OS 隔离的程序装置不能执行操作系统代码并由程序装置管理器管理。

[0005] 近来,提出了多程序装置多指令流多数据流 (MIMD) ISA (指令集架构) 扩展,它定义一组指令,能够允许 OS 隔离的程序装置被当作为 ring-3 用户级结构资源,以用于在单个 OS 线程的上下文中并发执行多个用户级线程。因此,OS 隔离的程序装置对用户级程序来说变得可用。通过使用用户级多程序装置 MIMD ISA 扩展,在 OS 可见的程序装置上运行的程序线程能够使用与隔离该 OS 的附加程序装置。通过把工作提交给该 OS 可见或不可见的程序装置,该多程序装置 MIMD ISA 扩展能够独立于该 OS 来衡量很多程序装置。同时,操作系统可以继续在该 OS 可见的程序装置的一个子集上运行。用户级 MIMD ISA 扩展允许执行用户级线程以在 OS 隔离的程序装置上运行用户级线程,以及该用户级线程与创建该用户级线程并且在 OS 可见的程序装置上运行的 OS 线程并发地运行。

发明内容

[0006] 根据本发明的实施例,提供了一种用于运行持续性用户级线程的方法,包括:

[0007] 创建持续性用户级线程,以独立于在操作系统调度的线程上的上下文切换活动而在操作系统隔离的程序装置上运行,所述操作系统调度的线程在操作系统可见的程序装置上运行;

[0008] 当所述持续性用户级线程遇到将要由操作系统处理的事件时,挂起所述持续性用户级线程的运行;以及

[0009] 使在附加的操作系统可见的程序装置上运行的服务线程向被挂起的持续性用户级线程提供操作系统服务。

[0010] 根据本发明的实施例,提供了一种用于运行持续性用户级线程的装置,包括:

[0011] 创建持续性用户级线程,以独立于在操作系统调度的线程上的上下文切换活动而在操作系统隔离的程序装置上运行的部件,所述操作系统调度的线程在操作系统可见的程序装置上运行;

[0012] 当所述持续性用户级线程遇到需要操作系统处理的事件时,挂起所述持续性用户级线程的运行的部件;

[0013] 使在附加的操作系统可见的程序装置上运行的服务线程向被挂起的持续性用户级线程提供操作系统服务的部件。

[0014] 按照本发明的一种用于运行持续性用户级线程的方法,包括:

[0015] 基于在操作系统可见的程序装置上运行并使用指令集扩展的操作系统调度的线程,来创建持续性用户级线程,以独立于在该操作系统调度的线程上的上下文切换活动而在操作系统隔离的程序装置上运行;

[0016] 当所述持续性用户级线程遇到需要操作系统处理的事件时,挂起该持续性用户级线程的运行;

[0017] 使在附加的操作系统可见的程序装置上运行的服务线程向被挂起的持续性用户级线程提供操作系统服务。

[0018] 按照本发明的第一方面的一种用于运行持续性用户级线程的设备,包括:

[0019] 程序装置管理器,用来基于在操作系统可见的程序装置上运行并使用指令集扩展的操作系统调度的线程,来创建持续性用户级线程,以独立于在该操作系统调度的线程上的上下文切换活动而在操作系统隔离的程序装置上运行;

[0020] 其中所述程序装置管理器用来当所述持续性用户级线程遇到需要操作系统处理的事件时,挂起该持续性用户级线程的运行;

[0021] 其中所述程序装置管理器适于使在附加的操作系统可见的程序装置上运行的服务线程向被挂起的持续性用户级线程提供操作系统服务。

[0022] 按照本发明的第二方面的一种用于运行持续性用户级线程的设备,包括:

[0023] 用于基于在操作系统可见的程序装置上运行并使用指令集扩展的操作系统调度的线程,来创建持续性用户级线程,以独立于在该操作系统调度的线程上的上下文切换活动而在操作系统隔离的程序装置上运行的装置;

[0024] 用于当所述持续性用户级线程遇到需要操作系统处理的事件时,挂起该持续性用户级线程的运行的装置;

[0025] 用于使在附加的操作系统可见的程序装置上运行的服务线程向被挂起的持续性用户级线程提供操作系统服务的装置。

附图说明

[0026] 通过在下面结合附图对本发明的实施例进行详细的说明,可以更全面地了解和理解本发明,其中:

[0027] 图 1 是一个根据本发明的一个示意性实施例的多程序装置系统的框图说明;

[0028] 图 2 是一个根据本发明的一个示意性实施例的硬件的简化逻辑图,它构成多程序装置系统的一部分;

[0029] 图 3 是一个根据本发明的一个示意性实施例的多程序装置的简化说明,其能够提供持续性用户级线程服务;

[0030] 图 4 是一个根据本发明的一个示意性实施例的多程序装置的概念说明,其能够提供持续性用户级线程服务;

[0031] 图 5 是一个根据本发明的一个示意性实施例的多程序装置系统的指令集架构的示意性图;

[0032] 图 6 是一个根据本发明的一个示意性实施例的控制传输指令的简单指令格式的简化说明;

[0033] 图 7 是一个根据本发明的一个示意性实施例的监控指令的简单指令格式的简化说明;

[0034] 图 8 和图 9 是一个根据本发明的一个示意性实施例执行持续性用户级线程的简化流程图说明。

[0035] 应当理解,为了简化和清楚地说明,没有必要对图中示出的单元画出比例来。例如,为了清楚起见某些单元的尺寸可以相对于其他单元进行放大。

具体实施方式

[0036] 在下面的详细说明中,为了提供对本发明的实施例的一个全面的了解,陈述了很多特定细节。然而,本领域普通的技术人员应当理解,无需这些特定细节也能实现本发明的实施例。在其他情况下,为了不混淆本发明的实施例,没有详细描述已知的方法和过程。

[0037] 在下面,按照在计算机存储器中的数据或二进制数字符号上进行操作的算法和符号表示的形式来提出该详细说明的某些部分。这些算法说明或表示可以是本领域技术人员用来在数据处理领域向其他本领域技术人员传达要点的技术。

[0038] 一种算法在这里通常被认为是将引起所要结果的自相一致的动作或操作序列。这些包括对物理量的物理操作。通常,但又不是必须地,这些物理量采用能够被存储、传输、合并、比较和其他操作的电或磁信号的形式。有时为了方便,主要为了公用的原因,这些符号已经称为比特、值、元素、符号、字符、术语、数字或者类似表示。然而,应当知道所有这些和类似术语是与适当的物理量有关的并且仅仅是应用于这些物理量的方便标记。

[0039] 除非特别说明,否则如下面说明中清楚地示出的那样,应当知道整个说明书的论述利用了诸如“处理”、“运算”、“计算”、“确定”和类似的术语,它指的是计算机或计算系统,或类似的电子计算设备的动作和 / 或处理,它管理和 / 或把在计算机系统的寄存器和 / 或存储器中表示物理量,诸如电形式的数据转换为在计算机系统的存储器、寄存器或其他这样的信息存储、传输或显示装置中的同样表示物理量的其他数据。

[0040] 例如使用那些存储有一条指令或一组指令的机器可读介质或物品可以实现本发明的某些实施例,如果机器执行这些指令,可使机器执行根据本发明的实施例的方法和操作。这些机器例如可以包括任何适当的处理平台、计算机平台、计算机设备、处理设备、计算机系统、处理系统、计算机、处理器、或类似装置,并且可以使用硬件和 / 或软件的任何适当组合来实现。机器可读介质或物品例如可以包括任何适当类型的存储器单元、存储器设备、

存储器物品、存储器介质、存储设备、存储物品、存储介质和 / 或存储单元,例如存储器、可移除或不可移除介质、可擦除或不可擦除介质、可写或可重写介质、数字或模拟介质、硬盘、软盘、光盘只读存储器 (CD-ROM)、可记录光盘驱动器 (CD-R)、可重写光盘驱动器 (CD-RW)、光盘、磁介质、各种类型的数字化通用盘 (DVD)、磁带、盒式磁带、或类似介质。指令可以包括任何适当类型的代码,例如源代码、目标代码、编译代码、解释代码、可执行代码、静态代码、动态代码、或类似代码,并且可以使用任何适当的高级、低级、面向对象、可视化、编译和 / 或解释变成语言来实现,诸如 C、C++、Java、BASIC、Pascal、Fortran、Cobol、汇编语言、机器代码或类似语言。

[0041] 本发明的实施例可以包括用来执行这里的操作的装置。可以为所要求的目的构建这些装置,或者它们可以包括一个通用计算机,它由存储在该计算机中的计算机程序选择性地激活或重新配置。这些计算机程序可以存储在计算机可读介质中,诸如但不限于任何类型的盘,包括软盘、光盘、CD-ROM,磁光盘、只读存储器 (ROM)、随机访问存储器 (RAM)、电可编程只读存储器 (EPROM)、电可擦除和可编程只读存储器 (EEPROM)、磁卡或光卡,或适用于存储电指令并能够连接到计算机系统总线上的任何其他类型的介质。

[0042] 这里所提出的处理和显示并不必然与任何特定计算机或其他装置有关。根据这里的教导可以使用具有程序的多种类型的通用系统,或者可以方便地构建一个专门的装置来执行所要求的方法。根据下面的说明,用于各种这些系统的所要求结构可以从下文看出。另外,没有参照任何特定的编程语言来描述本发明的实施例。应当知道,可以使用多种编程语言来实现如下所述的本发明的教导。

[0043] 在下面的说明中,提供各种图像、图表、模块、和描述以作为有效传达和说明本申请中提出的本发明的不同实施例的不同方式。本领域技术人员应当理解,提出它们仅作为示例性的例子,不应被看作是限制。

[0044] 图 1 是一个根据本发明的一个示意性实施例的多程序装置系统 100 的框图说明。

[0045] 如图 1 所示,多程序装置系统 100 可包括存储器 102 和多程序装置硬件 104。存储器 102 例如可包含用户级程序 106、调度程序 108、应用程序接口 (API) 110、和操作系统 (OS) 112。多程序装置硬件 104 例如可包括程序装置管理器 114 和多个程序装置,例如程序装置 116, 118, 120 和 122,它们在图 1 中分别被表示为 SID0, SID1, SID2 和 SID3。尽管在图 1 中示出 4 个程序装置,但是本领域技术人员理解程序装置硬件 104 可以包括其他数量的程序装置。

[0046] 程序装置管理器 114 可被实现为驱动器、诸如 OS 112 扩展的操作系统扩展、任何其他硬件或者硬件和软件的组合。在程序装置 116, 118, 120 和 / 或 122 的类型方面,多程序装置硬件 104 可以是对称的或非对称的。程序装置 116, 118, 120 和 122 中的一个或者多个可以是一个具有属于自己的一组执行资源的物理处理器。可选择地,程序装置 116, 118, 120 和 122 中的一个或者多个可以是逻辑处理器 (例如超线程处理器),它可以与其他逻辑处理器共享物理资源。可以在单个硬件内核或多个分开的硬件内核中实现程序装置 116, 118, 120 和 122。

[0047] 多程序装置系统 100 的非穷尽例子列表可包括桌上型个人计算机、工作站、服务器计算机、膝上型计算机、笔记本电脑、手持式计算机、个人数字助理 (PDA)、移动电话、游戏控制台、和类似装置。

[0048] 存储器 102 的非穷尽例子列表可包括下面半导体设备中的一个或者任何组合,诸如同步动态随机访问存储器 (SDRAM)、RAMBUS 动态随机访问存储器 (RDRAM) 设备、双数据率 (DDR) 存储器设备、静态随机存储器 (SRAM)、闪存 (FM) 设备、电可擦除只读存储器 (EEPROM) 设备、非易失性随机访问存储器 (NVRAM)、通用串行总线 (USB) 可移动存储器设备、等等;光盘设备,诸如光盘只读存储器 (CD ROM) 等等;以及磁盘设备,诸如硬盘、软盘、磁带等等。存储器 102 可以固定在系统 100 中或者可以从系统 100 中移走。

[0049] 根据本发明一个示意性的实施例,用户级程序 106 可以通过应用程序接口 (API) 110 访问线程库。API 110 可以向用户级程序 106 提供线程的创建、控制和同步。调度程序 108 可以调度线程的指令以在多程序装置硬件 104 上执行。

[0050] 多程序装置硬件 104 的非穷尽例子列表可包括多个单线程或多线程中央处理单元 (CPU)、数字信号处理器 (DSP)、精简指令集计算机 (RISC)、复杂指令集计算机 (CISC) 等等。另外,多程序装置 104 可以是特定用途集成电路 (ASIC) 的一部分,或者是特定用途标准产品 (ASSP) 的一部分。

[0051] 根据本发明一个示意性的实施例,多程序装置硬件 104 可以是已实现程序装置 116, 118, 120 和 122 (SID0-SID3) 的单核处理器,这些程序装置例如可以是逻辑处理器。单核处理器 104 例如可以支持并发、并行、和 / 或基于事件切换 (switch-on-event) 的多线程。根据一个示意性的实施例,每个程序装置 SID0-SID3 可具有自己的下一个指令指针逻辑,而同一个单核处理器 104 可以执行所有的线程指令。尽管单核处理器 104 的执行资源可以在并发、并行、和 / 或基于事件切换的线程中共享,但是每个逻辑处理器 SID0-SID3 可以保持一个它自己的结构状态版本。

[0052] 尽管上述的多程序装置硬件 104 的示意性实施例对于每个程序装置针对的是单线程,然而本领域技术人员知道在这里本申请并不局限于单线程的处理器。这里所述的技术同样地可应用于任何芯片多处理 (CMP) 或并行多线程处理器 (SMP) 系统,包括具有 CMP 处理器核 SMP 处理器的混合系统,其中 CMP 处理器中的每个核是 SMP 处理器或基于事件切换的多处理器 (SoeMP)。例如,这里所述的技术可用于这样的一种系统,其在一个单芯片硬件保 104 中包括多线程的处理核。

[0053] 如上所述,程序装置 SID0-SID3 没有必要是统一的,而在影响计算质量的因素方面可以是非对称的,所述因素诸如是处理速度、处理能力、和功率消耗。例如,程序装置 SID0 可以是“重量级”程序装置,因为它可被设计为在给定的指令集架构 (例如与 32 位处理器相关的 IA-32 指令集架构) 中处理大多数指令。而 SID1 可以是“轻量级”程序装置,因为它处理在这些指令中所选择的子集。在本发明的另一个实施例中,重量级处理器可以是这样的处理器,其可以用比轻量级处理器更快的速度来处理指令。另外,某些程序装置对于操作系统来说是可见的,而其他程序装置对操作系统来说是不可见的。例如,程序装置 SID0 对 OS 112 是可见的,而程序装置 SID1 至 SID3 对 OS 112 是不可见的。然而,这并不意味着重量级程序装置是 OS 可见的和 / 或轻量级程序装置是 OS 隔离的。操作系统不在一个处于隔离状态下的程序装置 (诸如在这里被称为 OS 隔离的程序装置) 上调度指令。

[0054] 根据本发明某些示意性的实施例,如下面参照图 2-8 所详细描述的那样,多程序装置系统 100 能够提供持续性用户级线程 (PULT) 服务。在本申请中,常规的用户级线程,或者在下面被简称为用户级线程,可以是一序列指令,它仅包含在用户级或 ring-3 级运行

的非特权指令。通常, OS 线程在后台支持用户级线程, OS 线程也被称为本地 OS 线程或真正本地线程。在该 OS 线程上下文中, 至少一个用户级线程可以在 OS 可见的程序装置或 OS 隔离的程序装置上运行。当 OS 线程被 OS 换出(和换入)时, 属于该 OS 线程的所有程序装置的上下文将被保存(和还原)。由此, 在这些程序装置上运行的用户级线程将被挂起(和重新开始)。根据本发明的一个实施例, 持续性用户级线程(PULT)是在 OS 隔离的程序装置上运行的用户级线程。该 PULT 可以在例如创建该 PULT 的另一个用户级线程的虚拟地址空间(VAS)的环境下执行。该 PULT 的执行可以独立于创建该 PULT 的另一个用户级线程的执行, 并因此可以不与其同步。即使当该用户级线程所属的 OS 线程处于上下文换出状态时, 该 PULT 仍可以在 OS 隔离的程序装置上继续执行。在这里概念“持续性”是指用户级线程继续执行, 而与有关产生或创建 PULT 的线程上的 OS 线程上下文切换活动无关。

[0055] 图 2 是一个根据本发明的一个示意性实施例的硬件 200 的简化逻辑图, 它构成多程序装置系统的一部分。

[0056] 硬件 200 可对应于构成图 1 的多程序装置系统 100 的一部分的多程序装置硬件 104。如图 1 所示, 多程序装置 104 可包括程序装置管理器 114 和例如多个程序装置 116, 118, 120 和 122。根据本发明的一个实施例, 程序装置管理器 114 能够以这样一种方式虚拟化程序装置 116, 118, 120 和 122, 在该方式下, 程序装置 116, 118, 120 和 122 以统一和对称的形式出现在用户级程序 106 中。换句话说, 对应于图 1 中程序装置管理器 114 的程序装置管理器 201 可以掩饰程序装置 211, 212, 213 和 214 的非对称性, 使得从汇编语言程序员的逻辑视角来看, 如图 2 所示, 程序装置 211, 212, 213 和 214 看起来是统一的和对称的, 并由相同大小的正方形表示。程序装置 211, 212, 213 和 214 可以表示图 1 中的逻辑处理器 SID0, SID1, SID2 和 SID3。

[0057] 图 3 是一个多程序装置 300 的简化说明, 根据本发明的一个示意性实施例, 其能够提供持续性用户级线程服务。

[0058] 根据本发明的一个实施例, 应用程序 301 的执行可以由操作系统(OS) 302 管理, 而该程序例如可以是图 1 的用户级程序 106。OS302 可以通过创建并调度一个或多个本地线程来管理应用程序 301 的执行, 每个本地线程潜在地包含用户级线程, 诸如在图 3 中运行在一组程序装置之一中的本地线程 303, 这些程序装置对 OS302 是可见的, 例如是程序装置 311 和 312。OS302 可以通过一个或多个自身的内核来管理程序装置 311 和 312。OS302 还可以控制程序装置 311 和 312 的特权状态并响应在程序装置 311 和 312 上发生的中断和异常处理。

[0059] 根据本发明的一个实施例, 程序装置管理器 310 可以控制对 OS 可见的程序装置 311 和 312 以及另一组程序装置的访问, 另一组程序装置例如是与 OS302 隔离并不受 OS302 的内核管理的程序装置 313, 314, 315, 316, 317 和 318。程序装置 313, 314, 315, 316, 317 和 318 是 OS 隔离的程序装置, 或“OS 不可见”的程序装置。

[0060] 根据本发明一个实施例, OS 调度的线程(例如潜在地包含用户级线程的本地线程)可以通过一个接口与程序装置管理器 310 进行交互, 该接口通过硬件或软件或者它们的组合来提供。为了使用 OS302 不可见的程序装置, 诸如程序装置 313, 314, 315, 316, 317 和 318, 例如本地线程 303 的 OS 调度的线程可以向程序装置管理器 310 发送一个请求以创建例如线程 304 的用户线程, 来运行 OS 隔离的程序装置, 例如程序装置 313。根据本发明

一个实施例,用户级线程 304 可以独立于本地线程 303 的 OS 上下文切换活动而在程序装置 313 上运行,并因此可以是持续性用户级线程 (PULT)。

[0061] 在下面的说明中,本地线程 303 被用作示出创建和执行持续性用户级线程的例子。本领域技术人员将理解,下面的说明也适用于其他线程,例如属于本地线程的用户级线程。

[0062] 根据本发明的一个示例性实施例,OS 隔离的程序装置 313 可以在虚拟地址空间 (VAS) 320 中执行 PULT304,该空间由在 OS 可见的程序装置 311 上运行的用户级线程 303 共享。OS302 可以创建 VAS320,以执行应用程序 301。根据本发明的一个示例性实施例,通过共享通用 VAS320,程序装置管理器 310 能够捕获在 OS 可见的程序装置 311 上运行的用户级线程 303 的执行环境,例如一个虚拟地址映射,并接着把该执行环境应用到在 OS 隔离的程序装置 313 上运行的 PULT304。根据本发明的一个实施例,OS 隔离的程序装置 313 可以具有一个或者多个窥探 TLB(转换后备缓冲器),其能够自动跟踪虚拟地址空间的未来变化。TLB 可以是存在于处理器中的一个小高速缓冲器,其跟踪虚拟地址到物理地址转换以便快速查找。处理器使用该 TLB 来避免不必要的页面移动,以便把虚拟地址转换为物理地址。

[0063] 图 4 是一个根据本发明的一个示意性实施例的多程序装置系统 400 的概念说明,其能够提供持续性用户级线程服务。

[0064] 根据本发明的一个实施例,多程序装置系统 400 可以包括一个程序装置管理器 410,该管理器管理多个 OS 隔离的程序装置,例如程序装置 411,412,413 和 414 和其他 OS 可见的程序装置,例如程序装置 405 和 406。操作系统 (OS) 402 可管理一个或多个用户级程序,诸如应用程序 401A 和 401B 的执行。操作系统 (OS) 402 通过创建一个或者多个线程来管理应用程序 401A 和 401B 的执行,这些线程例如包括用户级线程 403 和 404,以及其他本地线程,其中用户级线程 403 和 404 在 OS 可见的程序装置 405 和 406 上运行。

[0065] 程序装置管理器 410 可以通过访问程序装置 411,412,413 和 414 来向应用程序 401A 和 401B 提供执行资源,而这些程序装置与 OS402 隔离且不由 OS402 管理。可以通过由在 OS 可见的程序装置 405 和 406 上运行的线程 403 和 404 或者由 OS 调度的其他本地线程做出的请求来提供这种访问。因此,应用程序 401A 和 401B 可以通过用户级线程 403 和 / 或 404 或其他本地线程来访问 OS 隔离的程序装置 411,412,413 和 414。

[0066] 根据本发明的一个实施例,程序装置管理器 410 可以在程序装置 411,412,413 和 414 上为应用程序 401A 和 401B 分配不同时隙来执行一个或多个由用户级线程 403 和 404 创建的持续性用户级线程。例如,图 4 中的符号 t_1, t_2, t_3, t_4, t_5 和 t_6 表示分别为执行 PULT T_1, T_2, T_3, T_4, T_5 和 T_6 而分配的时隙。

[0067] 根据本发明的一个实施例,程序装置管理器 410 可以为一个或多个 OS 隔离的程序装置,例如程序装置 411,412,413 和 414,建立一个或多个中断描述符表,例如表 421,422,423 和 424。中断描述符表,例如表 421,可以包括事件句柄 431 来管理在程序装置 411 上正在发生的事件。在例如 PULT T_3 在 OS 隔离的程序装置,例如程序装置 411 上的执行时间期间 t_3 ,PULT T_3 可能遇到一种需要 OS402 处理或注意的事件或情形,诸如页面错误或系统调用。那么在该事件发生时间,事件句柄 431 挂起 PULT T_3 的执行并保存 PULT T_3 的执行状态。

[0068] 根据本发明的另一个实施例,当遇到页面错误或系统调用时,OS 不可见的程序装

置,例如程序装置 411,可以直接中断 OS 可见的程序装置的执行,例如这个调用 PULT T₃的程序装置 405。接着,程序装置 411 可以向程序装置管理器 410 发送一个触发器来请求处理这个中断。根据本发明又一个实施例,OS 隔离的程序装置 411 可以设计 ring-0 服务信道或用来处理页面错误或系统调用的信道。信道可以是一个保存结构状态的寄存器,这些状态例如包括触发器-响应映射,其中触发器是在体系结构上定义的一个或多个处理器状态的集合,其中响应是服务例程。另外,例如可以由用户级指令来设计信道。当出现页面错误或系统调用时,OS 隔离的程序装置 411 可以调用一个句柄,该句柄的地址在信道被指定。

[0069] PULT T₃可以一直保持挂起直到当新的用户级线程,例如运行同一应用程序 401A 的线程 409,被调度来在 OS 可见的程序装置上运行时为止。线程 409,也被称为服务线程,可以向程序装置管理器 410 做出一个请求以确认或验证是否有一个 PULT 已经被挂起并等待例如通过如下面详细说明了代理执行所提供的 OS 服务。线程通过从程序装置管理器 410 中获得的 PULT T₃状态来模仿或初始化 PULT T₃,由此来实现代理执行。共享通用地址空间,例如通用共享虚拟地址空间 (VAS) (图 3) 的服务线程 409 重新开始 PULT T₃的执行状态。接着线程 409 服务已经使 PULT T₃挂起的执行状态。在该服务后,线程 409 保存过去执行状态并将该状态返回到程序装置管理器 410。接着,程序装置管理器 410 把该状态传递到 PULT T₃以重新开始它的执行。

[0070] 上述代理执行给应用程序程序员创建了对称的错觉并因此掩盖了如何构建一个多程序装置系统,例如系统 100 的固有的非对称性属性。

[0071] 图 5 是一个根据本发明的一个示意性实施例的多程序装置系统的指令组结构 (ISA) 500 的示意性图。

[0072] ISA 定义一个系统的逻辑视图,如汇编语言程序员、二进制翻译器、汇编器等等所看到的那样。ISA500 可包括逻辑存储器 502 和指令集 504。逻辑存储器 502 可以为多程序装置系统,诸如图 1 中所示的多程序装置系统 100 定义可见的分级存储器体系、寻址方案、寄存器组等,而指令集 504 可以定义多程序装置系统 100 支持的指令和指令格式。

[0073] 根据本发明的示例性实施例,指令集 504 可包括称为 IA-32 指令集的指令集及其扩展,尽管本发明并不局限于这个方面,并且其他指令集也是可能的。根据本发明的一个实施例,指令集 504 例如可包括 E-SXFR 指令和 E-SEMONITOR 指令,其中 E-SXFR 指令包括 SXFR 控制传输指令及其扩展, E-SEMONITOR 指令包括 SEMONITOR 监控指令及其扩展。与可以在属于同一个 OS 本地线程中的两个程序装置之间执行的 SXFR 和 SEMONITOR 相比较,可以在分别属于不同的 OS 本地线程中的两个程序装置之间执行 E-SXFR 和 E-SEMONITOR,它们可以在不同的特权级上运行。例如,一个线程可以是受 OS 上下文切换影响的用户级应用程序,而另一个线程可以是以该特权级持续运行的 PULT,像一个设备驱动器,并不受 OS 上下文切换支配。

[0074] 根据本发明的一个实施例,E-SXFR 指令可以用来从第一程序装置向第二程序装置发送信号,而 E-SEMONITOR 指令可以用来配置所述第二程序装置以监控来自所述第一程序装置的信号。另外, SXFR 控制传输指令和 SEMONITOR 监控指令是程序装置可知道的,并可用来创建其他程序装置可知道的组合指令。

[0075] 根据本发明的一个实施例,E-SXFR 指令可以具有如图 6 所示的简单指令格式。在图 6 中示出该 E-SXFR 指令可以包括操作码 602,以及操作对象 604,606,608 和 612。根据本

发明的一个实施例,程序装置管理器可以提供具有操作对象的 E-SXFR 指令,该操作对象可包括程序装置的特权状态,诸如来创建一个执行环境以服务所挂起的持续性用户级线程。

[0076] 根据一个实施例,操作对象 604 可对应于目的 / 目标程序装置的程序装置 ID(SID),其中该 E-SXFR 指令信号将被发送到该目的 / 目标程序装置上。操作对象 606 可包括场景 (scenario) 或控制消息,其可以是用来表示条件或预期事件的在体系结构上定义的标识符代码。如下面参照表 1 详细说明的那样,场景可被用来影响异步控制传输。

[0077] 根据一个实施例,操作对象 608 可以包括一个参数,该参数设置在程序装置上的执行该 E-SXFR 指令的指令条件。例如,参数在图 6 中被示为条件参数,可包括一个“WAIT”或“NO-WAIT”参数。例如,当在代理执行场景下使用 E-SXFR 指令时,WAIT 条件参数可以在执行该 E-SXFR 指令的程序装置上的指令停止,直到另一个程序装置上代理执行的完成。NO-WAIT 条件参数可以规定在执行该 E-SXFR 指令的程序装置上的指令可以与在另一个指令程序装置上代理执行并行地进行。

[0078] 根据一个实施例,操作对象 610 可包括场景特定的有效载荷或数据消息。例如,在 FORK 场景的情况下,有效载荷可包括一个指令指针,其指向由操作对象 604 所标识的程序装置要开始执行的位置。根据另一个实施例,有效载荷操作对象 610 可包括指令指针、栈指针、一组控制寄存器等等。包含在有效载荷操作对象 610 中的地址可以被表示为多种寻址模式,诸如直接寻址、间接寄存器寻址,和 / 或基地址 / 偏移寻址。

[0079] 根据一个实施例,操作对象 612 可以规定包含在操作对象 604 中的 SID 上的路由方法或函数。路由函数控制由于执行 E-SXFR 指令而产生的信号是否作为广播、单播、或多播信号来发送。路由函数还可以编码信息,诸如可被用来在路由中帮助基本程序装置间互连传送信号的特定拓扑提示。

[0080] 根据一个实施例,E-SEMONITOR 指令可以具有如图 7 所示的简单指令格式。在图 6 中示出该 E-SEMONITOR 指令可以包括操作码 702,和操作对象 704,706 和 708。操作对象 704,706 和 708 可以包含程序装置的特权状态信息,以帮助持续性用户级线程创建一个执行环境。例如,操作对象 704 可以通过包括场景 ID 来规定场景。操作对象 706 可以包括与程序装置 ID(SID) 和指令指针 (EIP) 相关的信息。为了方便说明,如图 7 所示,包含在操作对象 706 中的信息可被称为“SIDEIP”。

[0081] E-SEMONITOR 指令可以把在操作对象 704 中规定的场景映射到在操作对象 706 中指定的 SIDEIP 上。从场景到 SIDEIP 的映射在这里被称为“服务信道”的映射。操作对象 708 允许程序员输入一个或多个控制参数来规定或控制如何服务这个特定服务信道。例如,程序员可以使用 E-SEMONITOR 指令来设计程序装置监控的服务信道。

[0082] 根据上述说明,应该理解 E-SXFR 和 E-SEMONITOR 指令都是“程序装置可识别的”指令,因为它们包含可以识别特定程序装置的操作对象。

[0083] 图 8 和图 9 是一个根据本发明的一个示意性实施例来执行持续性用户级线程的简化流程图说明。相同模块数字被用来表示相同的操作。

[0084] 根据本发明的一个实施例,如模块 801 所示,OS 隔离的程序装置 822 可以执行持续性用户级线程。为了方便说明,该持续性用户级线程在这里被称为 PULT-1。例如可以由第一用户级线程创建 PULT-1,其中第一用户级线程由第一 OS 可见的程序装置执行。在执行期间,PULT-1 可能遇到一个或多个需要 OS 服务或注意的事件。第二用户级线程可以与

创建 PULT-1 的第一用户级线程是同一个用户级线程,也可与其不同,并在例如 OS 可见的程序装置 821 的第二 OS 可见的程序装置上执行,如下面详细描述的那样,第二用户级线程向 PULT-1 提供所请求的 OS 服务。

[0085] 根据本发明的一个实施例,如在图 8 和 9 的模块 802 中所示的那样, PULT-1 在时间 t_1 遇到需要 OS 服务的事件。如模块 803 指出的那样,在检测到事件时,程序装置管理器 820 在时间 t_1 调用一个过程来保存 PULT-1 的状态,诸如例如一个逻辑指针。PULT-1 的状态如模块 803 所示那样已经被保存后,接着如模块 804 所示,从时间 t_2 挂起 PULT-1 的执行并向前进行。根据本发明的一个实施例,在接收到一个有关需要操作系统服务的事件的通知后,通过一个外部中断,程序装置管理器 820 也可以挂起 PULT-1 的执行。PULT-1 一直被挂起直到服务线程例如通过代理执行提供所要求的 OS 服务。

[0086] 根据本发明的一个示例性实施例,如模块 805 所示,诸如新用户级线程的 OS 调度的线程可以在 OS 可见的程序装置 821 上开始运行。该新的用户级线程可以为如 PULT-1 的同一个应用程序来执行。该线程能够通过例如代理执行向 PULT-1 提供 OS 服务并因此在这里被称为服务线程。如模块 806 所示,该服务线程可能检查或验证程序装置管理器 820,来判定是否存在任何等待 OS 服务的被挂起的 PULT。

[0087] 服务线程可以识别 PULT-1 为一个正等待 OS 服务的线程。根据本发明的一个实施例,如模块 807 所示,服务线程接着例如通过它们通用的共享地址空间 (VAS) 来获取 PULT-1 的执行状态,来模仿 PULT-1,换句话说模拟 PULT-1。由服务线程模仿的 PULT-1 可接着在执行服务线程的 OS 可见的程序装置上执行,以接收 OS 服务。因此,如模块 808 所示,通过服务线程的代理执行把 OS 服务提供给 PULT-1,并且 PULT-1 挂起的状态现在有操作系统来服务。根据本发明的一个实施例,运行在 OS 可见的程序装置,例如 OS 可见的程序装置 821 上的用户级线程,可以通过监控使用一个场景的地址位置来检测被挂起的 PULT。服务场景可用来监控高速缓冲器未命中的条件,并且当发生高速缓冲器未命中时,例如将导致阈值被越过,则在服务信道中的一个句柄被调用。

[0088] 根据本发明的一个实施例,如模块 809 所示,在操作系统完成服务所挂起的 PULT-1 的条件后, PULT-1 的后执行状态被保存,并接着通过程序装置管理器 820 被返回到 OS 隔离的程序装置 822。如模块 810 所示,从服务线程中接收到后执行状态后, OS 隔离的程序装置 822 可以重新开始其 PULT-1 的执行。

[0089] 表 1 包含根据本发明的某些实施例的,示例场景列表。

[0090] 表 1

[0091]

SPU 特征		出口服务场景	入口服务场景
OS 可见性	OS 不可见	需要 OS 处理并造成交叉环-级转变的操作, 例如: ☐ 页面错误处理 ☐ I/O 访问	进来的控制消息 ☐ Go-shred ☐ Halt-shred ☐ Report/self ☐ Save/Restore-shred ☐ Single-step-shred

[0092]

	OS 可见	出去的控制消息 ☐ Fork ☐ Jion ☐ Yield ☐ Wait ☐ Scatter/Set ☐ Gather/Get ☐ CSAVE/RSTOR ☐ Go/Halt/Single-step	需要 OS 处理并造成交叉环- 级切换的操作，例如： ☐ 页面错误处理 ☐ I/O 访问
核的复杂性	简单	SSE 13 指令	
	复杂		SSE 13 执行

[0093] 根据本发明的示例性实施例，表 1 中的场景可以被分为出口服务和入口服务。在每种服务之中，操作和控制消息可以由它们的 OS 可见性来特征化，OS 可见性可以是微处理单元 (SPU) 特征的一部分。在出口服务中，有些操作场景落入资源不可用 (RNA) 的类别，该类别是用于在程序装置上执行期间由于访问了在 OS 隔离的程序装置上不可用的资源所生成事件的类别。根据本发明的一个实施例，落入 RNA 类别的场景可包括例如在 OS 隔离的程序装置上的页面错误处理和 / 或 I/O 访问，这些程序装置不能直接启动 OS 服务。

[0094] 根据本发明的示例性实施例，这些上述的操作场景是一组出去的控制消息。在一个示例性实施例中，消息例如可包括表 1 中出口服务场景所列出的消息。然而，本发明并不局限于这些方面，可以使用其他消息。该控制消息转到一个或多个 OS 可见的程序装置。

[0095] 与上述出口服务场景相对，存在入口服务场景，它们落入 OS 可见的程序装置的类别，这些 OS 可见的程序装置向来自 OS 隔离的程序装置的服务请求提供资源。如表 1 所示在入口服务场景，这些服务场景直接对应于出口服务下的场景，但从 OS 可见的程序装置上是可用的。在一个示例性实施例中，附带有入口服务的进来的控制消息可以包括表 1 中入口服务场景中列出的消息。然而，本发明并不局限于这些方面，可以使用其他消息。进来的消息可以来自一个或多个 OS 隔离的程序装置。

[0096] 根据本发明的示例性实施例，那些服务出口服务的指令在内核的复杂性方面例如是相对简单的 SSE13 指令，而例如服务来自 OS 隔离的程序装置的入口服务或请求的指令通常是更复杂的。

[0097] 尽管本发明的某些特征已经在这里被说明和描述，然而对本领域普通技术人员来说可以出现很多的改变、替换、变化和等同。因此应当知道附后的权利要求书希望覆盖所有这些在本发明精神范围内的改变和变化。

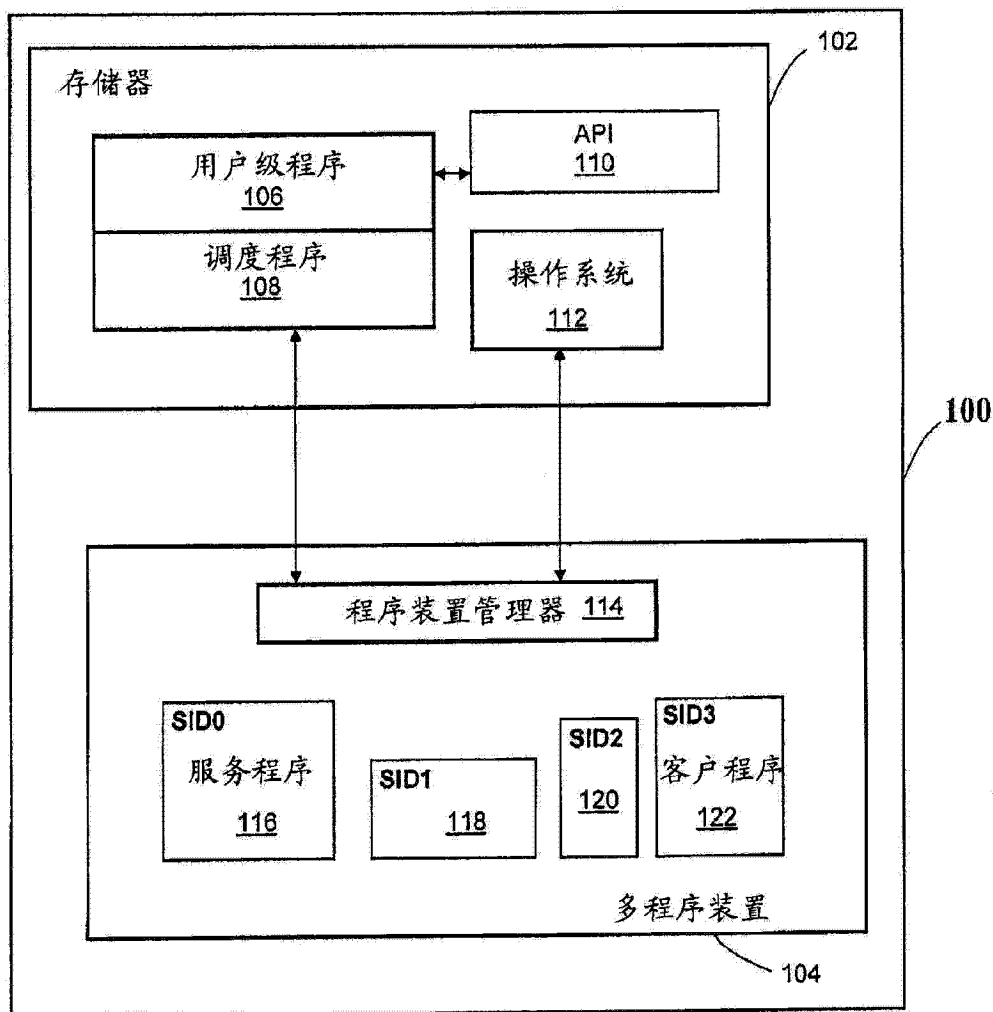


图 1

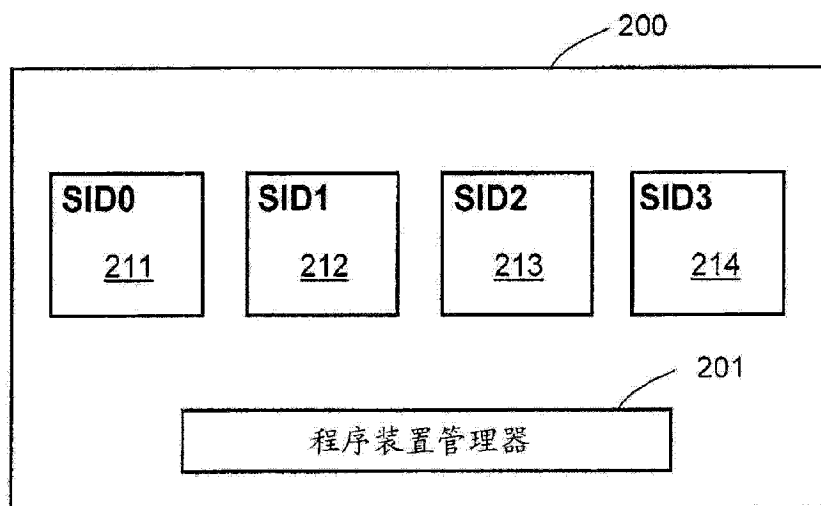


图 2

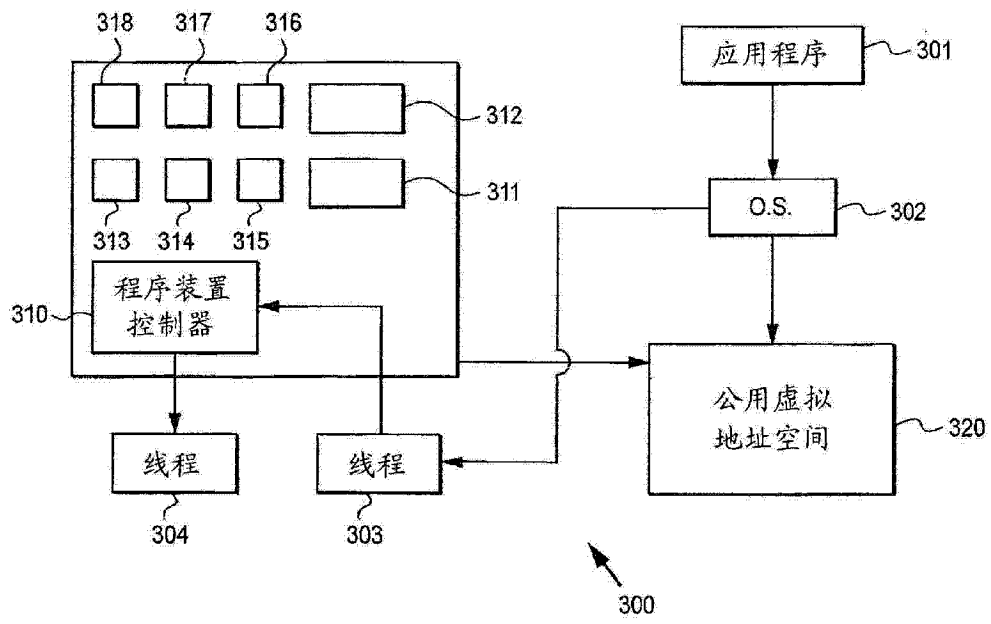


图 3

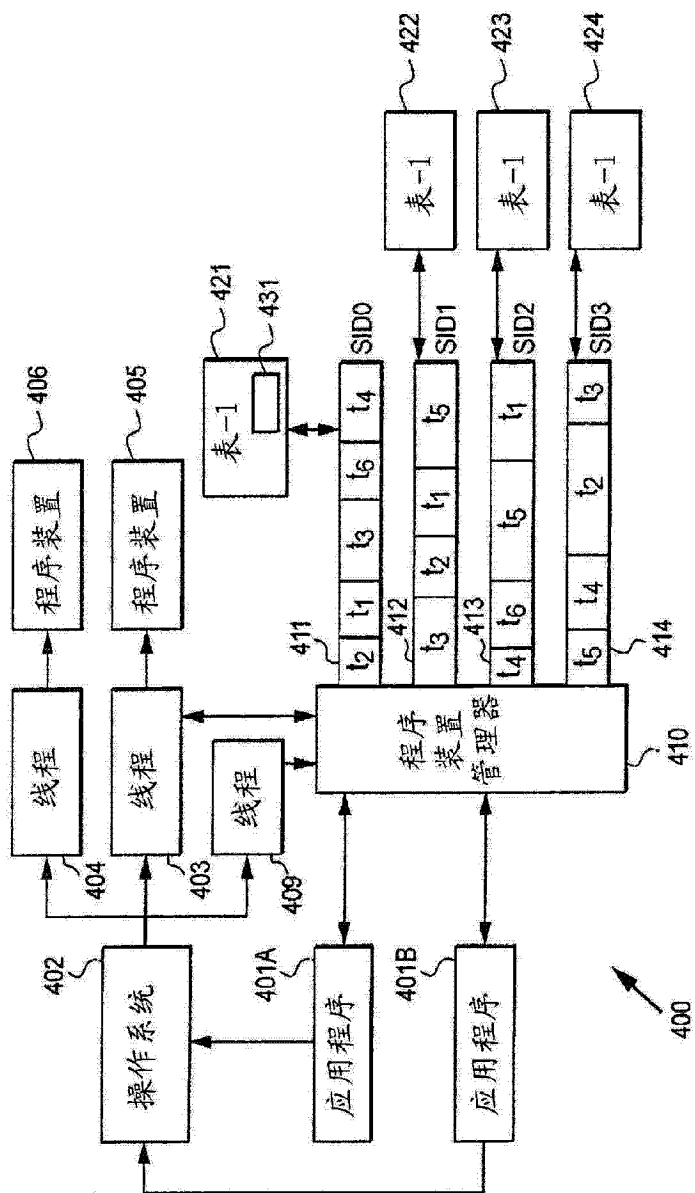


图 4

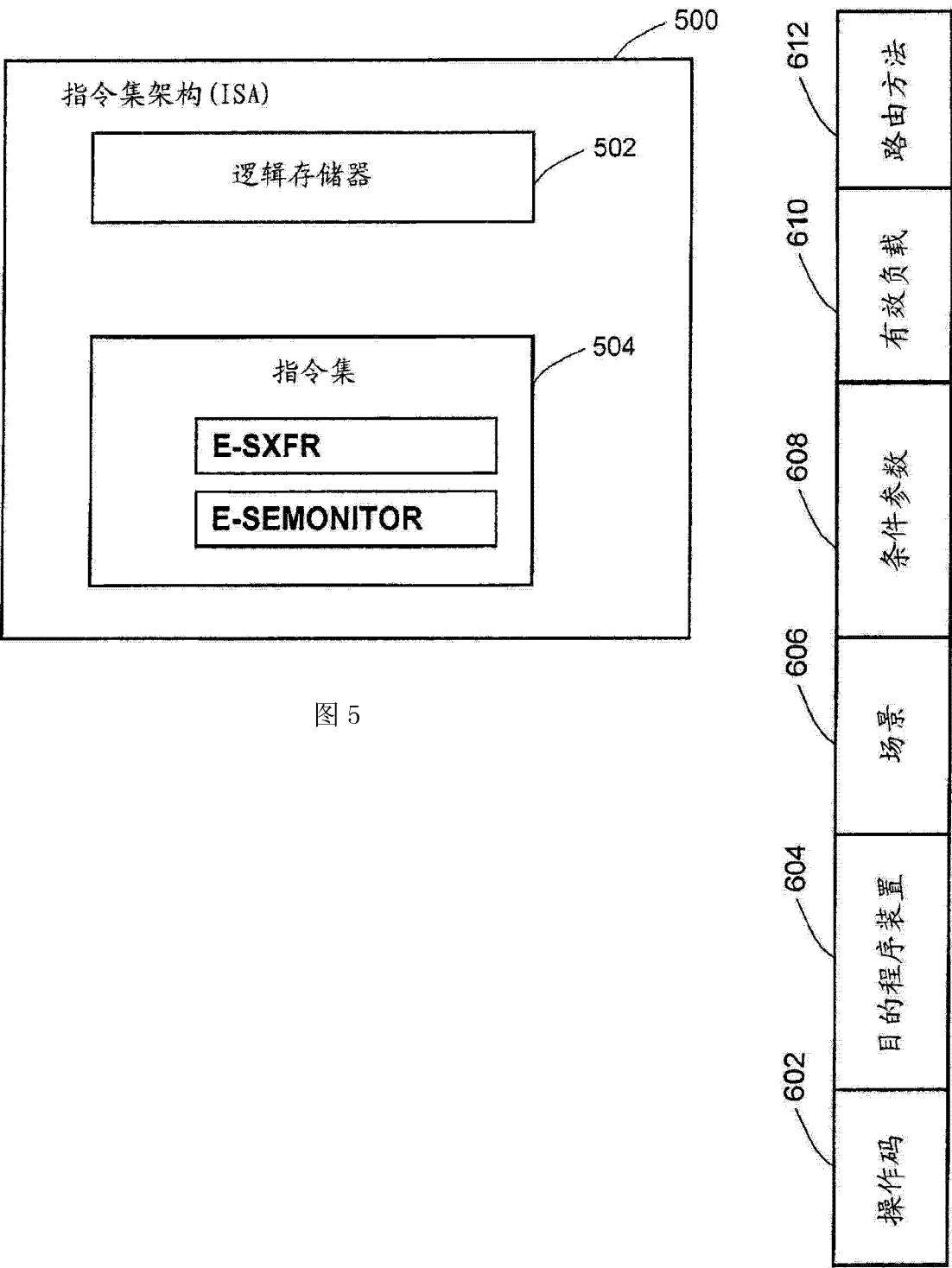


图 5

图 6

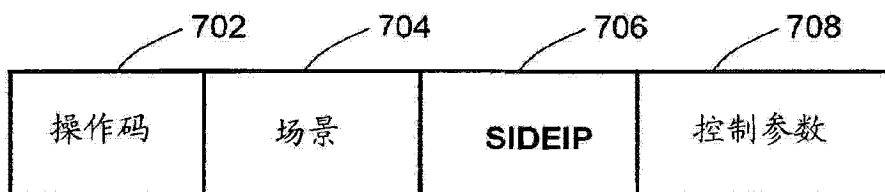


图 7

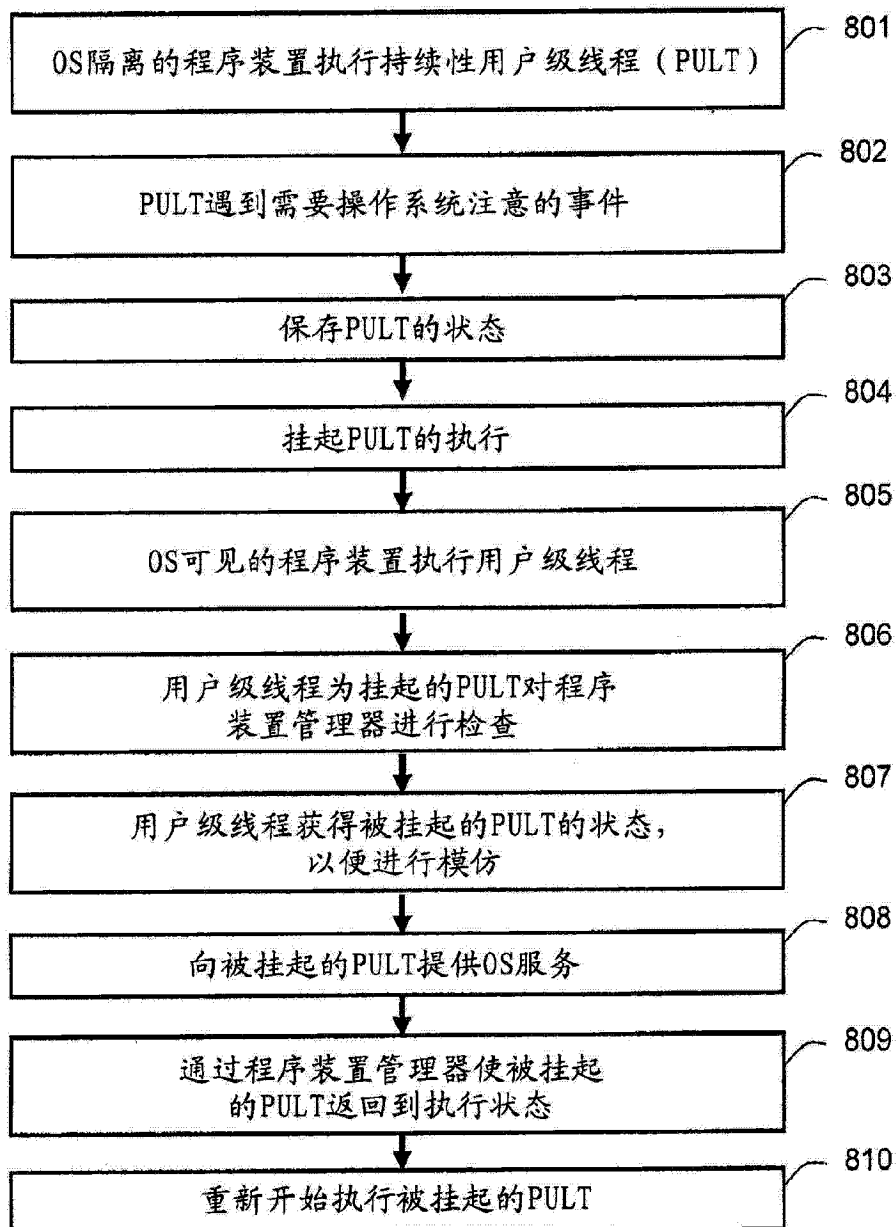


图 8

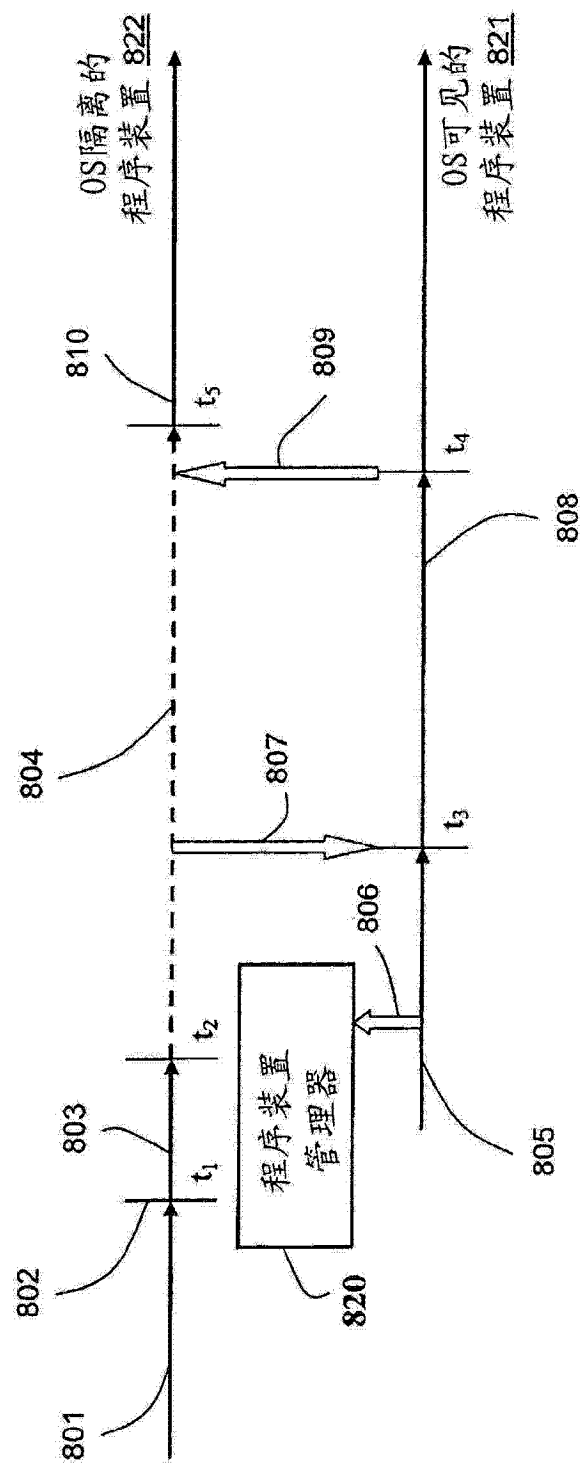


图 9