[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/46 (2006.01)
G06F 12/10 (2006.01)

[12] 发明专利说明书

专利号 ZL 200580008007.7

[45] 授权公告日 2008 年 12 月 10 日

[11] 授权公告号 CN 100442237C

[22] 申请日 2005.5.18
[21] 申请号 200580008007.7
[30] 优先权
[32] 2004.5.27 [33] US [31] 10/855,726
[86] 国际申请 PCT/EP2005/052279 2005.5.18
[87] 国际公布 WO2005/119444 英 2005.12.15
[85] 进入国家阶段日期 2006.9.12
[73] 专利权人 国际商业机器公司
地址 美国纽约
[72] 发明人 维沙·奇特兰简恩·阿斯罗特
布鲁斯·米利
[56] 参考文献
US20030204648A1 2003.10.30
CN1292529A 2001.4.25
US20030105914A1 2003.6.5

审查员 段雪莲
[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
代理人 付建军

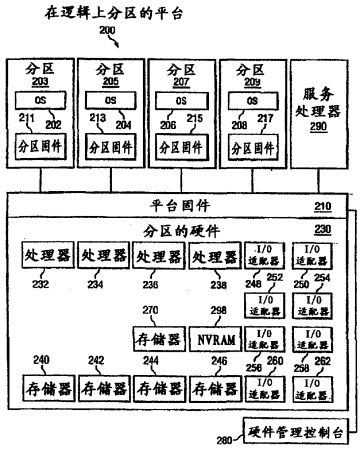
权利要求书 2 页 说明书 20 页 附图 5 页

[54] 发明名称

访问逻辑分区中的存储器的方法和系统

[57] 摘要

提供了用于在服务器分区环境中使用扩展跨存储器描述符的系统和方法，以便它可以用于描述另一个分区的存储器，例如，客户端分区的存储器(下面被称为“远程”存储器)。利用该系统和方法，当初始化在逻辑上分区的计算系统中的操作系统时，调用操作系统内核服务，这些操作系统内核服务检查计算系统的设备树并生成扩展的跨存储器描述符，该描述符描述了另一个逻辑分区的本地存储器，对于当前逻辑分区，该存储器是远程存储器。当执行某一操作需要对远程存储器进行访问时，服务器分区的操作系统使用存储的扩展的跨存储器描述符，以执行远程存储器访问。



1. 数据处理系统中用于第一逻辑分区中的第一进程访问第二逻辑分区中的远程存储器的方法，包括:

检索用于远程存储器的扩展的跨存储器描述符，其中，扩展的跨存储器描述符提供远程存储器的描述，以及其中，扩展的跨存储器描述符包括将扩展的跨存储器描述符指定为描述不同逻辑分区中的远程存储器的第一字段；以及

第一逻辑分区中的第一进程基于扩展的跨存储器描述符来访问第二逻辑分区中的远程存储器。

2. 根据权利要求 1 所述的方法，其中，扩展的跨存储器描述符进一步包括标识第二逻辑分区中的远程存储器的第二字段，指定远程存储器的大小的第三字段以及指定远程存储器中的起始地址的第四字段。

3. 根据权利要求 1 所述的方法，其中，当初始化第一逻辑分区中的操作系统时生成扩展的跨存储器描述符。

4. 根据权利要求 1 所述的方法，其中，由将其他逻辑分区的远程存储器附加到第一逻辑分区的操作系统的操作系统内核服务来生成扩展的跨存储器描述符。

5. 根据权利要求 4 所述的方法，其中，当初始化第一逻辑分区中的操作系统时，操作系统内核服务基于由操作系统内核服务分析的第二逻辑分区的设备树来生成扩展的跨存储器描述符。

6. 根据权利要求 5 所述的方法，其中，由系统管理程序维护设备树。

7. 根据权利要求 1 所述的方法，其中，第一逻辑分区位于服务器计算设备中，而第二逻辑分区位于客户端计算设备中。

8. 根据权利要求 1 所述的方法，其中，访问第二逻辑分区中的远程存储器包括使用扩展的跨存储器描述符来执行直接存储器访问操作。

9. 根据权利要求 1 所述的方法，其中，基于扩展的跨存储器描述符来访问第二逻辑分区中的远程存储器包括：

沿着输入/输出 (I/O) 堆栈将扩展的跨存储器描述符传递到物理设备驱动程序；

基于扩展的跨存储器描述符来生成直接存储器访问操作；以及

将直接存储器访问操作提交到 I/O 适配器，其中，I/O 适配器将直接存储器访问操作传输到第二逻辑分区。

10. 用于第一逻辑分区中的第一进程访问第二逻辑分区中的远程存储器的系统，包括：

用于检索用于远程存储器的扩展的跨存储器描述符的装置，其中，扩展的跨存储器描述符提供远程存储器的描述，以及其中，扩展的跨存储器描述符包括将扩展的跨存储器描述符指定为描述不同逻辑分区中的远程存储器的第一字段；以及

用于供第一逻辑分区中的第一进程基于扩展的跨存储器描述符来访问第二逻辑分区中的远程存储器的装置。

访问逻辑分区中的存储器的方法和系统

技术领域

一般来说，本发明涉及一种改进的数据处理系统。具体来说，本发明涉及用于扩展跨存储器描述符以便它可以用于描述来自在逻辑上分区的计算环境中的另一个分区的存储器的系统和方法。

背景技术

高级交互执行 (ADC) 操作系统在 International Business Machine (IBM) 公司的 pSeries 机器上支持逻辑分区 (LPAR)。LPAR 允许在单个处理器复合体上运行多个系统映像。每一个这样的映像都具有 CPU（专用或共享的）、中央存储器、扩充存储器和通道的完全补充。利用 LPAR，在各个分区之间资源有清晰的隔离，以便一个分区不会使任何其他分区的系统环境不稳定。在 IBM 公司的机器中，实现各个分区的资源的这种清晰的分离的功能是系统管理程序。

由于每一个分区都具有其自己的隔离的资源集，因此，每一个分区都具有其自己的“本地”存储器，该“本地”存储器对周围的分区不可见。直到创建虚拟输入/输出 (VIO)，一个分区没有任何合法理由直接对另一个分区的存储器进行访问。

利用 VIO，引入了各个分区之间的点对点直接存储器访问 (DMA) 操作和存储器复制操作的概念。DMA 操作允许将数据从一个存储器直接传输到另一个存储器，无需使用中央处理单元 (CPU) 来执行数据传输。DMA 操作允许进行比在必须使用 CPU 来传输数据的情况下更快的数据传输。存储器复制操作允许存储器的某些部分直接从一个存储器复制到另一个存储器。

如此，VIO 允许在各个分区之间共享物理资源，例如，I/O 适配器等等。由于某一个资源可以只属于一个分区（叫做“服务器”分区），其他分区必须经过服务器分区才能使用该资源（其他分区被称为“客户

端”分区）。

在共享资源是 I/O 适配器的情况下，为了使用服务器分区的 I/O 适配器，服务器分区必须建立从 I/O 适配器到客户端的存储器的 DMA 操作。然而，为了建立 DMA 操作，服务器分区必须知道有关客户端分区的存储器的结构的信息。此外，服务器分区需要能够将有关客户端分区的存储器结构的信息以可使用的格式沿着其自己的 I/O 堆栈即，软件层的规定的层次结构传递，通过该层次结构，数据必须经过处理才能执行 I/O 操作（例如，逻辑卷管理器 (LVM)、磁头驱动程序、适配器驱动程序、总线驱动程序、内核等等）。如此，最好有用于以这样的方式在服务器分区的环境中描述客户端分区的存储器，以便该描述可以轻松地沿着服务器分区的 I/O 堆栈传递的系统和方法。

发明内容

本发明提供了用于在服务器分区环境中扩展跨存储器描述符的使用，以便它可以用于描述另一个分区的存储器，例如，客户端分区的存储器（下面被称为“远程”存储器）的系统和方法。利用该系统和方法，当初始化在逻辑上分区的计算系统中的操作系统时，如在逻辑分区的引导过程中，调用操作系统内核服务，以便执行诸如加载设备驱动程序之类的启动操作。作为这些操作系统内核服务的一部分，调用内核服务，该内核服务将其他逻辑分区的远程存储器附加到当前操作系统，以便可以利用其他逻辑分区的远程存储器执行远程存储器复制和 DMA 操作。

附加内核服务审查计算系统的可能由公开固件等等创建的并维护在服务器分区的存储器中的设备树，以便它可被服务器分区中的操作系统访问。每个逻辑分区有一个设备树实例。设备树包括该分区中资源的节点。这些资源包括每一个分区的本地存储器、PCI/ISA I/O 插槽、处理器，受支持的固件调用等等。

当 VIO 在计算环境中得到支持时，在典型的实施例中，服务器分区的设备树中的 /vdevice 节点或其子节点包含客户端分区的本地

存储器的描述。该描述包括客户端的本地存储器的逻辑标识符、本地存储器的起始地址以及本地存储器的长度。根据该信息，可以生成每一个逻辑分区的每一个本地存储器的扩展的跨存储器描述符。然后，由当前分区的操作系统维护这些扩展的跨存储器描述符，供在执行远程存储器复制和 DMA 操作时使用。

当执行某一操作需要对远程存储器进行访问时，服务器分区的操作系统使用扩展的跨存储器描述符，该描述符描述了作为远程存储器操作的主体的客户端分区的“远程”存储器。本发明的扩展的跨存储器描述符是对现有的跨存储器描述符的扩展，现有的跨存储器描述符是通常用于只描述其中使用了跨存储器描述符的某一个分区的本地存储器的数据结构。

在本发明中，跨存储器描述符的字段被扩展，以便它们被用来存储有关另一个分区的本地存储器（即，“远程”存储器）的信息。 具体来说，扩展的跨存储器描述符的字段包括地址空间标识符、远程存储器的大小、标识哪一个远程存储器的标识符（在有多个客户端分区的情况下）以及在远程存储器内的起始地址。当使用跨存储器描述符来代表“远程”存储器时，地址空间标识符将跨存储器描述符标识为远程存储器描述。如此，I/O 堆栈或存储器管理服务中的检查跨存储器描述符的任何接口现在可以查看地址空间标识符，并识别这是远程存储器描述。结果，接口可以采取合适的操作，以便设置 DMA 操作来访问“远程”存储器。

本发明的这些及其他特征和优点将在下面描述，或者考虑下面的对优选实施例的详细描述，这些及其他特征和优点对于本领域普通技术人员将变得显而易见。

附图说明

现在将参考下面的附图，只作为示例，对本发明的优选实施例进行描述：

图 1 是可以在其中实现本发明的实施例的数据处理系统的框图；

图 2 是其中可以实现本发明的实施例的示范性逻辑分区平台的框图；

图 3 是说明了根据本发明的一个示范性实施例的用于生成扩展的跨存储器描述符的机制的示例的示范图；

图 4 是说明了根据本发明的一个示范性实施例的使用扩展跨存储器描述符来执行需要进行远程存储器访问的操作的机制的示例的示范图；

图 5 是概述了根据本发明的一个示范性实施例的用于生成扩展的跨存储器描述符的示范性过程的流程图；以及

图 6 是概述了根据本发明的一个示范性实施例的使用扩展的跨存储器描述符的示范性过程的流程图。

具体实施方式

提供了一种机制，用于扩展第一分区环境中的跨存储器描述符，以便它描述另一个分区的本地存储器，以便促进需要直接存储器访问或远程存储器复制操作的操作。如此，优选情况下，本发明的实施例在其中资源是使用逻辑分区机制进行分区的计算设备中实现。

现在参考附图，具体来说，参考图 1，该图描述了其中可以实现本发明的实施例的数据处理系统的框图。数据处理系统 100 可以是包括连接到系统总线 106 的多个处理器 101、102、103 和 104 的对称多处理器 (SMP) 系统。例如，数据处理系统 100 可以是 IBM eServer，该系统是位于纽约 Armonk 的 IBM 公司的产品，被实现为网络内的服务器。或者，也可以使用单个处理器系统。连接到系统总线 106 的还有：存储器控制器/高速缓存 108，它提供了到多个本地存储器 160-163 的接口。I/O 总线桥路 110 连接到系统总线 106，并提供到 I/O 总线 112 的接口。如本文所述，存储器控制器/高速缓存 108 和 I/O 总线桥路 110 也可以集成在一起。

数据处理系统 100 是逻辑分区 (LPAR) 数据处理系统。如此，数据处理系统 100 可以具有同时运行的多个异构操作系统(或单个操作系统的多个实例）。这些多个操作系统中的每一个操作系统都可以

具有任意数量的软件程序在其内执行。数据处理系统 100 是在逻辑上分区的，以便不同的 PCI I/O 适配器 120-121、128-129 和 136、图形适配器 148 和硬盘适配器 149 可以分配给不同的逻辑分区。在此情况下，图形适配器 148 提供了到显示设备（未显示）的连接，而硬盘适配器 149 提供了连接，以控制硬盘 150。

如此，例如，假设数据处理系统 100 被分成三个逻辑分区，P1、P2 和 P3。每一个 PCI I/O 适配器 120-121、128-129、136，图形适配器 148、硬盘适配器 149、每一个主机处理器 101-104 以及本地存储器 160-163 中的存储器，被分配给三个分区中的每一个分区。在这些示例中，存储器 160-163 可以采用双列直插式存储器模块 (DIMM) 的形式。DIMM 通常不每个 DIMM 地分配到分区。相反，分区将获得由平台所看到的总存储器的一部分。例如，处理器 101、本地存储器 160-163 中的部分存储器以及 I/O 适配器 120、128 和 129 可以分配给逻辑分区 P1；处理器 102-103，本地存储器 160-163 中的部分存储器以及 PCI I/O 适配器 121 和 136 可以分配给分区 P2；处理器 104，本地存储器 160-163 中的部分存储器，图形适配器 148 和硬盘适配器 149 可以分配给逻辑分区 P3。

在数据处理系统 100 内执行的每一个操作系统都分配给不同的逻辑分区。如此，在数据处理系统 100 内执行的每一个操作系统只能访问其逻辑分区内的那些 I/O 单元。如此，例如，高级交互执行 (AIX) 操作系统的一个实例可以在分区 P1 内执行，AIX 操作系统的第二个实例（映像）可以在分区 P2 内执行，Linux 或 OS/400 操作系统可以在逻辑分区 P3 内操作。

连接到 I/O 总线 112 的外围组件互连 (PCI) 主机桥路 114 提供了到 PCI 本地总线 115 的接口。许多 PCI 输入/输出适配器 120-121 可以通过 PCI 到 PCI 桥路 116、PCI 总线 118、PCI 总线 119、I/O 插槽 170 和 I/O 插槽 171 连接到 PCI 总线 115。PCI 到 PCI 桥路 116 提供了到 PCI 总线 118 和 PCI 总线 119 的接口。PCI I/O 适配器 120 和 121 分别被插入 I/O 插槽 170 和 171。

典型的 PCI 总线实现方式将支持四个到八个 I/O 适配器（即，插入式连接器的扩展槽）。每一个 PCI I/O 适配器 120-121 提供数据处理系统 100 和输入/输出设备（如，作为数据处理系统 100 的客户端的其他网络计算机）之间的接口。

其他 PCI 主机桥路 122 提供其他 PCI 总线 123 的接口。PCI 总线 123 连接到多个 PCI I/O 适配器 128-129。PCI I/O 适配器 128-129 可以通过 PCI 到 PCI 桥路 124、PCI 总线 126、PCI 总线 127、I/O 插槽 172 和 I/O 插槽 173 连接到 PCI 总线 123。PCI 到 PCI 桥路 124 提供到 PCI 总线 126 和 PCI 总线 127 的接口。PCI I/O 适配器 128 和 129 分别被插入 I/O 插槽 172 和 173。如此，诸如调制解调器或网络适配器之类的其他 I/O 设备可以通过每一个 PCI I/O 适配器 128-129 中的每一个得到支持。如此，数据处理系统 100 允许连接到多个网络计算机。

插入到 I/O 插槽 174 的存储器映射图形适配器 148 可以通过 PCI 总线 144、PCI 到 PCI 桥路 142、PCI 总线 141 和 PCI 主机桥路 140 连接到 I/O 总线 112。硬盘适配器 149 可以插入 I/O 插槽 175 中，该插槽 175 连接到 PCI 总线 145。而此总线又连接到 PCI 到 PCI 桥路 142，该桥路 142 通过 PCI 总线 141 连接到 PCI 主机桥路 140。

PCI 主机桥路 130 为 PCI 总线 131 提供了接口，以连接到 I/O 总线 112。PCI I/O 适配器 136 连接到 I/O 插槽 176，该 I/O 插槽 176 通过 PCI 总线 133 连接到 PCI 到 PCI 桥路 132。PCI 到 PCI 桥路 132 连接到 PCI 总线 131。此 PCI 总线还将 PCI 主机桥路 130 连接到服务处理器邮箱接口和 ISA 总线访问通路逻辑 194 和 PCI 到 PCI 桥路 132。服务处理器邮箱接口和 ISA 总线访问通路逻辑 194 将转发发往到 PCI/ISA 桥 193 的 PCI 访问。NVRAM 存储器 192 连接到 ISA 总线 196。服务处理器 135 通过其本地 PCI 总线 195 连接到服务处理器邮箱接口和 ISA 总线访问通路逻辑 194。服务处理器 135 还通过多个 JTAG/I^2C 总线 134 连

接到处理器 101-104。JTAG/I^2C 总线 134 是 JTAG /扫描总线（请参见 IEEE 1149.1）和 Phillips I^2C 总线的组合。然而，或者，JTAG/I2C 总线 134 只可以被 Phillips I^2C 总线或只可以被 JTAG/扫描总线替代。主机处理器 101、102、103 和 104 的所有 SP-ATTN 信号一起连接到服务处理器的中断输入信号。服务处理器 135 具有其自己的本地存储器 191，并可以访问硬件 OP 面板 190。

当数据处理系统 100 最初被通电时，服务处理器 135 使用 JTAG/I2C 总线 134 询问系统（主机）处理器 101-104、存储器控制器/高速缓存 108 以及 I/O 桥路 110。在完成此步骤时，服务处理器 135 具有数据处理系统 100 的库存和拓扑理解。服务处理器 135 还在通过查询主机处理器 101-104、存储器控制器/高速缓存 108 和 I/O 桥路 110 查找到的所有元素上执行内部自测（Built-In-Self-Test (BIST)）、基本保证测试 (BAT) 和存储器测试。在 BIST、BAT 和存储器测试期间检测到的故障的任何错误信息都由服务处理器 135 收集和报告。

如果有意义的/有效的系统资源配置在提取了在 BIST、BAT 和存储器测试期间发现有故障的元件之后仍是可能的，那么，数据处理系统 100 被允许继续将可执行代码加载到本地（主机）存储器 160-163 中。然后，服务处理器 135 释放主机处理器 101-104，以便执行加载到本地存储器 160-163 中的代码。当主机处理器 101-104 执行来自数据处理系统 100 内的相应的操作系统的代码时，服务处理器 135 进入监视和报告错误的模式。被服务处理器 135 监视的项目的类型包括，例如，冷风扇的速度和操作、热传感器、电源调节器、处理器 101-104 报告的可恢复的和不可恢复的错误、本地存储器 160-163 以及 I/O 桥路 110。

服务处理器 135 负责保存和报告与数据处理系统 100 中的所有被监视的项目相关的错误信息。服务处理器 135 还基于错误类型和定义的阈值执行操作。例如，服务处理器 135 可能会注意到处理器的高速缓存存储器上有过多的可恢复的错误，并判断这是硬件故障的前

兆。基于此判断，服务处理器 135 可以标记该资源，以便在当前运行会话和未来的初始程序加载 (IPL) 期间取消配置。IPL 有时也被称为“启动”或“自举”。

数据处理系统 100 可以使用各种市场上买得到的计算机系统来实现。例如，数据处理系统 100 可以使用 IBM 公司所提供的 IBM eServer iSeries Model 840 系统来实现。这样的系统可以支持使用 OS/400 操作系统（这也是 IBM 公司所提供的）的逻辑分区。

那些本领域普通技术人员将认识到，图 1 所描述的硬件可以改变。例如，除了所描述的硬件，也可以使用诸如光盘驱动器之类的其他外围设备，或者代替所描述的硬件。所描述的示例并不意味着对本发明的体系结构作出限制。

现在参考图 2，框图描述了在其中可以实现本发明的实施例的示范性逻辑分区平台的框图。逻辑分区平台 200 中的硬件可以作为例如图 1 中的数据处理系统 100 来实现。逻辑分区平台 200 包括分区硬件 230、操作系统 202、204、206、208 和分区管理固件 210。操作系统 202、204、206 和 208 可以是同时在逻辑分区平台 200 上运行的单个操作系统的多个副本或多个异构操作系统。这些操作系统可以使用 OS/400 来实现，这些操作系统是为与诸如系统管理程序之类的分区管理固件连接而设计的。OS/400 在这些说明性实施例中只起到示例的作用。当然，诸如 AIX 和 linux 之类的其他操作系统类型，也可以使用，具体情况视特定的实现方式而定。操作系统 202、204、206 和 208 位于分区 203、205、207 和 209 中。系统管理程序软件是可以用来实现分区管理固件 210 的软件的示例，是 IBM 公司所提供的。固件是存储在没有通电的情况下仍能保存其内容的存储器芯片（如只读存储器、可编程序只读存储器 (PROM)、可擦可编程序只读存储器 (EPROM)、电可擦可编程序只读存储器 (EEPROM) 以及非易失性随机存取存储器（非易失性 RAM））中的“软件”。

另外，这些分区也包括分区固件 211、213、215 和 217。分区固件 211、213、215 和 217 可以使用初始自举程序代码、IEEE-1275

标准公开固件以及运行时抽象软件 (RTAS)（这是 IBM 公司所提供的）来实现。当分区 203、205、207 和 209 被实例化时，自举程序代码的一个副本被平台固件 210 加载到分区 203、205、207 和 209 上。此后，控制被传输到自举程序代码，然后，自举程序代码加载公开固件和 RTAS。然后，将与分区关联的处理器或分配给分区的处理器派遣到分区的存储器，以执行分区固件。

分区硬件 230 包括多个处理器 232-238、多个系统存储器单元 240-246、多个输入/输出 (I/O）适配器 248-262 和存储单元 270。可以将处理器 232-238、存储器单元 240-246、NVRAM 298 和 I/O 适配器 248-262 中的每一个分配给逻辑分区平台 200 内的多个分区中的某一个，其中每一个都对应于操作系统 202、204、206 和 208 中的某一个。

分区管理固件 210 为分区 203、205、207 和 209 执行许多功能和服务，以创建和实施逻辑分区平台 200 的分区。分区管理固件 210 是与基础硬件相同的固件实现的虚拟机。如此，分区管理固件 210 通过虚拟化逻辑分区平台 200 的所有硬件资源，允许同时执行独立的 OS 映像 202、204、206 和 208。

可以使用服务处理器 290 来提供各种服务，如处理分区中的平台错误。这些服务还可以充当服务代理，来将错误报告回诸如 IBM 公司之类的供应商。不同的分区的操作可以通过诸如硬件管理控制台 280 之类的硬件管理控制台来进行控制。硬件管理控制台 280 是单独的数据处理系统，系统管理员可以从该系统执行各种功能，包括将资源重新分配到不同的分区。

在这样的在逻辑上分区的环境中，每当将要在内核和当前进程地址空间之外的地址空间之间移动数据时，都使用跨存储器内核服务来执行此数据的移动。通过调用 xmattach 内核服务（该服务附加到跨存储器操作的用户缓冲器中），附加地址空间的一个区域内的数据区域。当调用 xmattach 内核服务时，xmattach 内核服务生成跨存储器描述符。此后，可以使用其他跨存储器内核服务来将数据从内核移动

或复制到当前进程地址之外的地址空间。例如，xmemin 内核服务通过将数据从指定的地址空间复制到内核全局存储器中来执行跨存储器移动。xmemout 内核服务通过将数据从内核全局存储器复制到指定的地址空间中来执行跨存储器移动。xmemdma 内核服务为 DMA I/O 准备存储器的一个页面或在 DMA I/O 完成之后处理一个页面。

在已知计算系统中，跨存储器描述符是用于描述本地存储器的数据结构。从操作系统的虚拟存储器管理 (VMM) 组件维护的信息来生成本地存储器跨存储器描述符。

由内核服务基于操作系统在引导时生成和分析的设备树来生成根据本发明的优选实施例的扩展的跨存储器描述符。设备树类似于公开固件设备树，它是描述系统硬件和用户配置选择的分级数据结构。公开固件设备树还包含硬件驱动程序和供这些驱动程序使用的支持例程。下面是公开固件设备树的示例：

root/

ff8885f8/rtas

ff866bf4 /rom@ff000000

ff8627c0 /flash@fff00000

ff8513d0/cpus

ff88alc8 /PowerPC,604ev@()

ff88a788 /12-cache

ff84d8e0/pci

ff89552c /ethernet@4

ff8952e0/display

ff88cf44/mac-io@2

ff893e68 /misc@0

ff894688 /iic

ff893d7c/via@ 16000

ff8939fc/escc@ 13000

ff893c2c/ch-b@13000

ff893af4/ch-a@ 13020

ff88fd08/scsi@ 10000

ff892elc/tape

ff8924d0/disk

ff88f944 /escc-legacy@ 12000

ff88fba8/ch-b@ 12000

ff88fa60/ch-a@ 12002

ff88d78c/adb@ 11000

ff88f364/mouse@3

ff88e6dc /keyboard@2

ff88d638 /open-pic@40000

ff88aad8/ide@1,1

ff88c504/disk

ff85a534/isa@1

ff864208 /sound@i534

ff8640e8/midi@i330

ff863ff8 /game@i200

ff863368/gpio@i800

ff863008 /nvram@me0000

ff862aa4 /rtc@i70

ff85f644/8042@i60

ff86l8b0 /mouse@aux

ff860260 /keyboard@

ff85d804 /floppy@i3f0

ff85d3b4 /parallel@i3bc

ff85c704/serial@i2f8

ff85b9f4/serial@i3f8

ff85b490 /timer@i40

ff85b01c /interrupt-controller@i20

ff85ae08 /dma-controller@i00

ff84a650 /mmu

ff83f2e4 /memory@0

对于本发明的实施例，使用类似的设备树表示来获取用于生成扩展的跨存储器描述符的信息。设备树是面向对象的，以便设备树中的每一个条目都具有可以被检查的关联的属性。下面说明了设备树中的有关远程存储器的信息可以呈现给操作系统的节点的示例：

/devicenode

…

remote-memory-info10000000 00200000 00040000

20000000 003D0000 00020000

…

…

其中，“remote-memory-info”是节点“/devicenode”的属性。在客户端-服务器虚拟 I/O 模式的典型的实现方式中，节点“/devicenode”将代表服务器设备的节点。如下面所讨论的，可以对此节点进行分析，通过分析此节点所获得的信息可以用来生成根据本发明的优选实施例的扩展的跨存储器描述符。如此，对于第一客户端，“10000000”条目是存储器的逻辑标识符，“00200000”条目是存储器的起始地址，而“00040000”条目是存储器的以字节为单位的大小。同样，对于第二客户端，“20000000”是存储器的逻辑标识符，“03D00000”是存储器内的起始地址，而“00020000”是存储器的以字节为单位的大小。对于服务器所服务的每一个客户端，可以重复此三元组（逻辑 id、起始地址、大小）。

根据该设备树信息，可以获取系统硬件和用户配置选择的详细描述。具体来说，如上所述，设备树的存储器节点包括指定了存储器的逻辑标识符、存储器的起始地址和存储器的长度的属性。此信息可以被内核服务用来生成计算系统的本地存储器资源的跨存储器描述符。

跨存储器描述符被用作从两个不相关的上下文中访问存储器以便不需要知道所涉及的存储器的所有者的方式。例如，当中断处理程序需要向一个进程传递数据时，它使用跨存储器描述符来获取有关进程的存储器空间信息，以将数据复制到进程的存储器空间。这是必需的，因为在拥有其数据的进程的上下文中中断处理程序不一定运行，因此不知道该数据属于哪一个进程。中断处理程序所知道的一切是应该在哪里复制数据的存储器的描述，即，跨存储器描述符。

磁盘 I/O 是使用本地存储器跨存储器描述符的另一个示例。例如，用户级别的应用程序可以调用 IOCTL 来读取有关磁盘的信息。信息将存储在应用程序提供的缓冲器中，而该缓冲器位于用户空间。当调用磁盘驱动程序时，磁盘驱动程序调用 xmattach 以附加应用程序的缓冲器。所有这一切都是由磁盘驱动程序的前半部执行的。

将来某个时间，当 I/O 请求完成，数据变得可用，磁盘驱动程序的位于磁盘驱动程序的后半部的“iodone”例程，调用 xmout 以使用跨存储器描述符将数据复制到用户空间。磁盘驱动程序存储跨存储器描述符，以便它能够被前半部和后半部访问。

在操作系统初始化时，由内核服务生成跨存储器描述符，并在需要时，将它在计算设备的 I/O 堆栈中从一个接口传递到另一个接口（或存储器管理服务）。I/O 堆栈的接口和存储器管理服务检查跨存储器描述符以获取执行跨存储器操作所需要的信息。

操作系统在另一个分区的远程存储器方面面临着类似的问题，因为远程存储器复制/DMA 操作对另一个分区中的所有者设备没有任何了解。如此，这里所描述的方法利用了跨存储器描述符的扩展版本来在执行远程存储器复制/DMA 操作时提供有关另一个分区的远程存储器的信息。

本地跨存储器描述符包括四个主要字段：地址空间标识符、段的数量、开始段标识符以及段内的起始地址。地址空间标识符提供了将向其中复制数据的存储器的唯一标识符。段的数量提供了由地址空间标识符标识的地址空间的段的总数。开始段标识符标识了所有段内的

被复制数据将开始在其中写入的段。段内的起始地址标识了由开始段标识符标识的被复制数据将开始在其中写入的段内的地址。

下面是其中说明了这些主要字段的跨存储器数据结构的示例：

```
enum asid={LOCAL=0, REMOTE=1};
struct cross_memory {
enum asid address_space_Id; /* LOCAL,
REMOTE,
_ etc. */
int number_of_segments; /* number of segments
if address_space_id=LOCAL*/
long segmen_Id; /* segment number if
address_space_id=LOCAL */
char *vaddr; /* starting address within
the segment if
address_space_id=LOCAL */
uint flags; /* any special attributes of the
segment or address range */
};
#define remote_size (number_of_segments) /* size of
remote memory if
address_space_id=REMOTE */
#define remote_logical_id (segment_id) /*
remote
memory's logical
identifier*/
#define remote_start_addr (vaddr) /*
starting
address within the remote
memory */
```

如此，从上述字段可以看出，跨存储器描述符提供了具有这样的特征的存储器的描述，即，为了访问特定地址空间，该存储器描述可以用来执行跨存储器操作而无需知道所述特定地址空间的所有者并进而将数据传递到该所有者。本发明的实施例使用此已知数据结构来实现不同的操作，其中不需要有关资源的所有权的信息即可完成此操作。即，跨存储器描述符的扩展版本用于实现从一个分区到另一个分区的远程存储器复制/DMA 操作（其中，“远程”存储器是与当前执行的进程所在的分区不同的另一个分区中的存储器）。利用这样的操作，作为虚拟输入/输出 (VIO) 进程的一部分，一个分区中的进程可以访问另一个分区的存储器，而无需将数据传递到拥有该远程存储器的特定进程。结果，通过使用扩展的跨存储器描述符，可以实现跨多个分区的远程存储器复制和 DMA 操作。

跨存储器描述符字段用于存储有关远程存储器的信息。跨存储器描述符的四个原始字段被覆盖，以描述远程存储器，从而生成具有下列四个字段的扩展的跨存储器描述符：地址空间标识符、远程存储器的大小、标识远程存储器的标识符以及远程存储器内的起始地址。扩展的跨存储器描述符的地址空间标识符类似于原始跨存储器描述符的地址空间标识符，例外是，此字段中可以存储预先确定的值，以指出扩展的跨存储器描述符对应于客户端分区的远程存储器。

远程存储器的大小可以以字节为单位或存储器大小的任何其他单位来表示，并指定可以用来执行访问的远程存储器的总大小。远程存储器标识符标识特定客户端分区中的特定远程存储器。在有多个客户端分区并且必须指定正在访问哪一个远程存储器的情况下，这是重要的。远程存储器内的起始地址提供了远程存储器内的由远程存储器标识符指定的将要在哪里开始访问的位置。根据这四个字段，可以获取远程存储器的全貌，以便执行远程存储器复制/DMA 操作。

扩展的跨存储器描述符可以以与原始跨存储器描述符同样的方式通过 I/O 堆栈从一个接口传递到另一个接口。可以加强需要检查扩展的跨存储器描述符的那些接口和存储器管理服务，以包括用于处理

扩展的跨存储器描述符的代码。即，加强那些接口和存储器管理服务，以包括识别何时地址空间标识符包括预先确定的值的代码，所述预先确定的值在描述远程存储器访问请求的远程存储器时标识扩展的跨存储器描述符。然后，接口和存储器管理服务可以基于由扩展的跨存储器描述符提供的远程存储器的大小、远程存储器的身份以及远程存储器中的起始地址执行适当的操作。例如，d_map_page 和 d_map_list 接口使用远程跨存储器描述符来正确地建立从由服务器分区所拥有的物理设备到远程存储器的 DMA。如此，远程跨存储器描述符由物理设备的驱动程序传递到 d_map_page/list 接口以建立 DMA。按照服务器分区的定义，物理设备属于服务器分区。结果，可以跨多个分区地执行远程存储器复制/DMA 操作。

通过覆盖现有的跨存储器描述符以成为与原始跨存储器描述符基本上具有相同格式的扩展的跨存储器描述符，只是在跨存储器描述符的字段中存储不同数据，以及与此不同数据关联的不同内涵，本发明的实施例允许许多需要跨存储器描述符但不检查跨存储器描述符的内容的接口和存储器管理服务在不进行修改的情况下操作。即，扩展的跨存储器描述符允许操作系统维持与需要跨存储器描述符但不实际对它进行检查的接口和存储器管理服务的二进制兼容性。例如，大多数设备驱动程序需要跨存储器描述符，但不对它进行检查。它们只需将跨存储器描述符传递到内核或 PCI 总线驱动程序。由于对跨存储器描述符的更改，不需要对这样的设备驱动程序进行重新编译，它们可以如往常那样继续进行操作。

图 3 是说明了根据本发明的一个示范性实施例的用于生成扩展的跨存储器描述符的机制的示例的示范图。如图 3 所示，当初始化在逻辑上分区的计算系统中的逻辑分区 310 的操作系统 320 时，如在逻辑分区 310 的引导过程中，调用操作系统内核服务 325，以便执行诸如加载设备驱动程序之类的引导操作。作为这些操作系统内核服务 325 的一部分，调用内核服务，该内核服务将其他逻辑分区 312-316 的远程存储器附加到当前操作系统，以便可以利用其他逻辑分区的远

程存储器执行远程存储器复制和 DMA 操作。在本发明的一个示范性实施例中，附加远程存储器的内核服务被称为 xmattach_remio，是已知 xmattach 跨存储器内核服务的扩展版本。xmattach_remio 附加跨存储器操作的远程存储器，这些操作可以由已知 xmemout 和 xmemin 跨存储器内核服务和 d_map_page 和 d_map_list 内核服务的扩展版本执行。

扩展的附加内核服务检查计算系统的可以在系统管理程序 330 等等中维护的设备树 332-338。系统管理程序 330 是在逻辑上分区的计算系统中提供和管理多个虚拟机的程序。在系统管理程序 330 中维护了每一个逻辑分区的单个设备树 332-338。此外，逻辑分区的设备树 332-338 的副本可以存储在该逻辑分区的本地存储器中，以便它可被逻辑分区中的操作系统内核访问。操作系统提供了内核扩展（例如，设备驱动程序）和用户级别代码可以使用的功能库，以便对设备树 332-338 进行分析。

设备树 332-338 包括计算系统的每一个分区的节点和它们的资源。这些资源包括每一个分区的本地存储器 340-360。设备树 332-338 的代表分区的这些本地存储器的节点包括本地存储器的属性，这些属性包括本地存储器的逻辑标识符、本地存储器的起始地址和本地存储器的长度。根据该信息，可以获取每一个逻辑分区的本地存储器（相对于当前逻辑分区，是“远程”存储器）的大小、每一个逻辑分区的本地存储器的身份以及每一个逻辑分区的本地存储器内的起始地址。可以将此信息封装到每一个逻辑分区的每一个本地存储器的扩展的跨存储器描述符 370-390 中。然后，由当前逻辑分区 310 的操作系统 320 维护这些扩展的跨存储器描述符 370-390，供在对其他逻辑分区 312-316 执行远程存储器复制和 DMA 操作时使用。

图 4 是说明了根据本发明的一个示范性实施例的使用扩展跨存储器描述符来执行需要进行远程存储器访问的操作的机制的示例的示范图。如图 4 所示，当诸如客户端设备 400 的应用程序 410 之类的进程例如通过 read() 系统调用向操作系统内核空间 420 生成要

求执行远程存储器访问操作的输入/输出 (I/O) 请求时，通过 I/O 堆栈 422 处理 I/O 请求，直到客户端虚拟设备驱动程序 424。客户端虚拟设备驱动程序 424 将该请求转发到服务器 480 的对应物，即，服务器虚拟驱动程序 430。

服务器虚拟驱动程序 430 已经在引导时为存储器 415 的远程分区（例如，逻辑分区 2）创建了扩展的跨存储器描述符 440。例如，服务器虚拟驱动程序 430 可以在引导时调用 xmattach_remio 内核服务，以生成扩展的跨存储器描述符。服务器虚拟驱动程序 430 在服务器端将 I/O 请求与扩展的跨存储器描述符 440 一起沿着 I/O 堆栈 450 传递。

I/O 请求最后沿着 I/O 堆栈 450 传递到物理设备驱动程序 460，例如，SCSI 适配器驱动程序，该驱动程序利用其 I/O 适配器 470 安排 DMA 操作。当由 I/O 适配器 470 完成 DMA 时，将数据写入到远程存储器 415 中。如此，如上述示例中所说明的，除了数据本身之外，进行远程直接存储器访问所需的唯一的其他必需的元素是通过扩展的跨存储器描述符所提供的远程存储器的描述。

图 5 和 6 是说明了根据本发明的示范性实施例的生成和使用扩展的跨存储器描述符的示范性过程的流程图。可以理解，流程图中的每一个方框，以及流程图中的方框的组合可以通过计算机程序指令来实现。这些计算机程序指令可以提供给处理器或其他可编程数据处理设备，以产生一个机器，以便在处理器或其他可编程数据处理设备上执行的指令创建用于实现流程图方框中指定的功能的装置。这些计算机程序指令也可以存储在计算机可读的存储器或存储介质中，该存储器或存储介质可以指示处理器或其他可编程数据处理设备以特定方式运行，以便存储在计算机可读的存储器或存储介质中的指令产生一种包括可以实现流程图方框中指定的功能的指令装置的产品。

相应地，流程图的方框支持用于执行指定的功能的装置的组合、用于执行指定的功能的步骤以及用于执行指定的功能的程序指令装置的组合。还应该理解，流程图的每一个方框，以及流程图中的方框的

组合，可以通过执行指定的功能或步骤的基于专用硬件的计算机系统或专用硬件和计算机指令的组合来实现。

图 5 是概述了根据本发明的一个示范性实施例的用于生成扩展的跨存储器描述符的示范性过程的流程图。如图 5 所示，操作开始时是初始化操作系统（步骤 510）。然后，操作系统内核服务对计算机系统的设备树进行分析，寻找代表其他逻辑分区的远程存储器的节点（步骤 520）。然后，检索这些远程存储器节点的属性（步骤 530），并用这些属性判断其他分区的每一个远程存储器的标识符、大小和起始地址（步骤 540）。然后，利用远程标识符将标识符、大小和起始地址封装到每一个其他逻辑分区的每一个远程存储器的扩展的跨存储器描述符中（步骤 550）。然后，存储这些扩展的跨存储器描述符，供以后在执行远程存储器访问时使用（步骤 560）。

图 6 是概述了根据本发明的一个示范性实施例的使用扩展的跨存储器描述符的示范性过程的流程图。如图 6 所示，操作开始时，接收需要远程存储器访问的请求（步骤 610）。然后，检索该请求所指向的远程存储器的扩展的跨存储器描述符（步骤 620）。然后，将扩展的跨存储器描述符与该请求一起传递到 I/O 堆栈（步骤 630），该 I/O 堆栈相应地对扩展的跨存储器描述符和请求进行处理。然后，使用扩展的跨存储器描述符通过 I/O 适配器执行远程存储器访问（步骤 640）。

如此，提供了用于扩展已知的跨存储器描述符的机制，以便它可以描述不同于进程当前正在其中运行的分区的另一个分区的远程存储器。由于本发明的实施例所提供的扩展与已知的跨存储器描述符使用相同格式，但具有不同的数据和与数据关联的内涵，不必为处理扩展的跨存储器描述符而修改计算设备中存在的许多接口和存储器管理服务。可以加强检查跨存储器描述符的其他接口和存储器管理服务，以包括用于处理扩展的跨存储器描述符的代码，以便提供用于执行远程存储器访问的功能。

虽然是在典型的 DMA 操作的上下文中描述本发明的示范性实

施例的，但是，应该理解，本发明的实施例不仅限于这样的情况。例如，扩展的跨存储器描述符也可以用于远程复制操作。可以由系统管理程序或其他硬件和/或软件机制来实现复制操作，其中，服务器分区通过操作系统和系统管理程序之间的很好地描述的接口向该机制（例如，系统管理程序）提供信息。所提供的信息来自扩展的跨存储器描述符（例如，标识客户端分区的远程存储器的标识符）。然后，系统管理程序可以执行从服务器到客户端或从客户端到服务器的实际复制操作。

操作可以由服务器端启动，因此，服务器要么可以拉数据（如利用 **xmemin** 系统调用），要么可以向客户端分区推数据（如利用 **xmemout** 系统调用）。系统管理程序可以执行实际复制操作，因为它比在它上面运行的操作系统具有更高的监督权限。这就使得系统管理程序访问任何分区的任何存储器。这种特性的远程复制操作促成了服务器和客户端之间的更先进的数据交换协议的设计和实现。

值得注意的是，尽管是在具有完全功能的数据处理系统的上下文中描述本发明的实施例，本领域普通技术人员将认识到，进程能够以存储了指令的计算机可读的介质的形式和各种各样的形式进行分发，这里所描述的方法同样适用，不管实际用于进行分发的承载信号的介质的特定类型是什么。计算机可读的介质的示例包括可记录类型的介质，如软盘、硬盘驱动器、RAM、CD-ROM、DVD-ROM、传输类型的介质，例如使用诸如射频和光波传输的传输形式的数字和模拟通信链路、有线或无线通信链路。计算机可读的介质可以采取编码格式的形式，这些编码在特定数据处理系统中实际使用时被解码。

说明书只作说明，而不是穷尽性的或限于所说明的形式。那些本领域普通技术人员将认识到，可以进行许多修改。所选择和描述的实施例是为了最好地说明本发明的原理，实际应用，并使本领域普通技术人员懂得，带有各种修改的各种实施例也是可以接受的。

图1

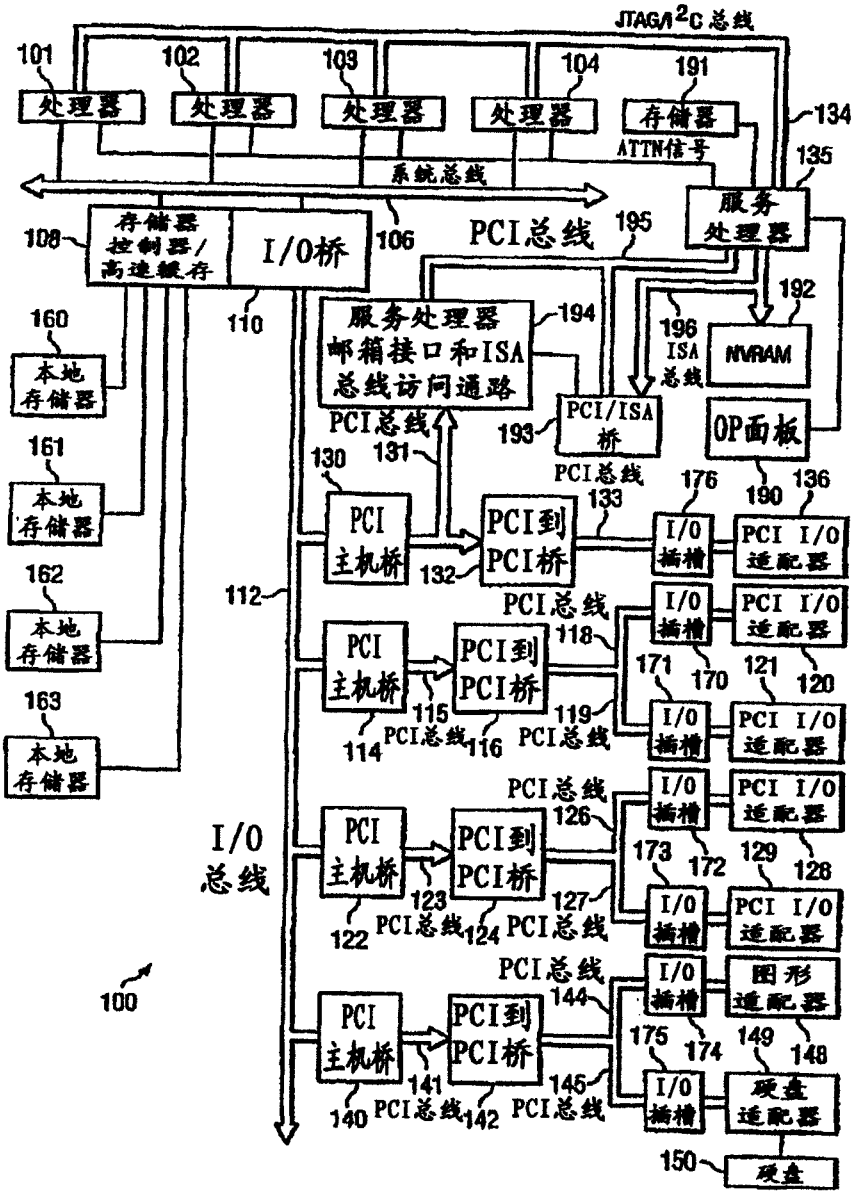

JTAG/I2C总线
101
处理器
102
处理器
103
处理器
104
处理器
191
存储器
134
ATTN信号
135
系统总线
服务处理器
108
存储器控制器/高速缓存
I/O桥
106
PCI总线
195
192
服务处理器邮箱接口和ISA总线访问通路
194
196
ISA总线
NVRAM
160
本地存储器
110
PCI总线
193
PCI/ISA桥
OP面板
161
本地存储器
130
131
PCI主机桥
PCI到PCI桥
133
176
190
136
I/O插槽
PCI I/O适配器
162
本地存储器
112
PCI总线
118
171
170
121
120
163
本地存储器
115
114
PCI总线
116
PCI总线
119
I/O总线
PCI总线
126
173
172
129
128
123
122
PCI总线
124
PCI总线
127
PCI总线
144
175
174
149
148
图形适配器
141
140
PCI总线
142
PCI总线
145
硬盘适配器
150
硬盘
100

图2

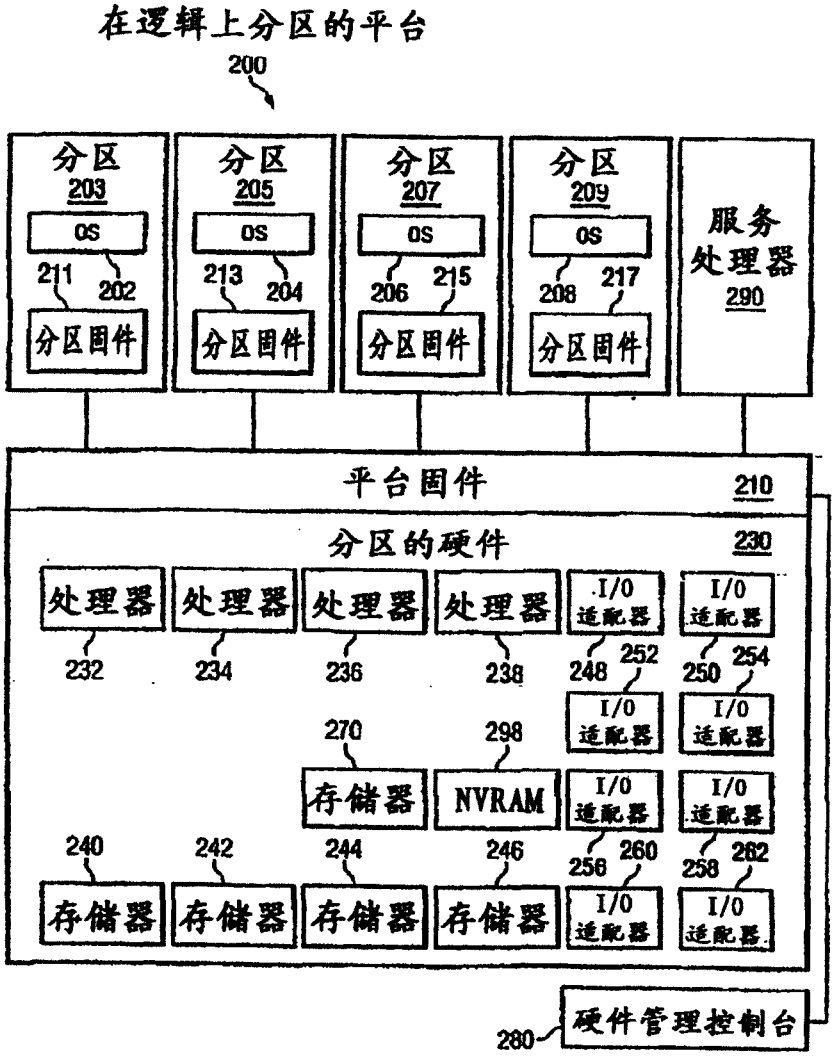

在逻辑上分区的平台
200
分区
203
OS
211
202
分区固件
分区
205
OS
213
204
分区固件
分区
207
OS
206
215
分区固件
分区
209
OS
208
217
分区固件
服务
处理器
290
平台固件
210
分区的硬件
230
处理器
处理器
处理器
处理器
I/O
适配器
I/O
适配器
232
234
236
238
248
252
250
254
I/O
适配器
I/O
适配器
270
298
存储器
NVRAM
I/O
适配器
I/O
适配器
240
242
244
246
256
260
258
262
存储器
存储器
存储器
存储器
I/O
适配器
I/O
适配器
280
硬件管理控制台

图3

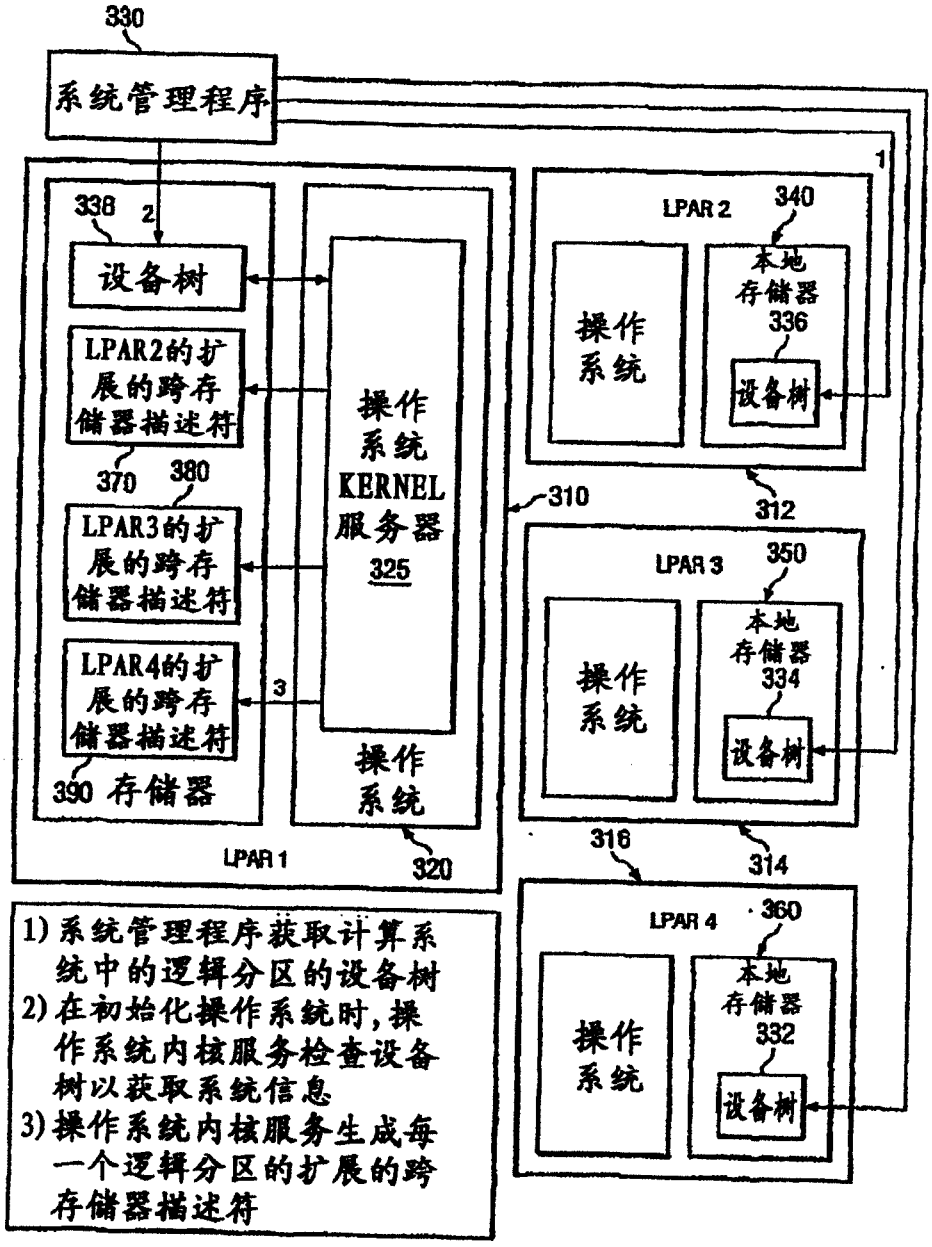
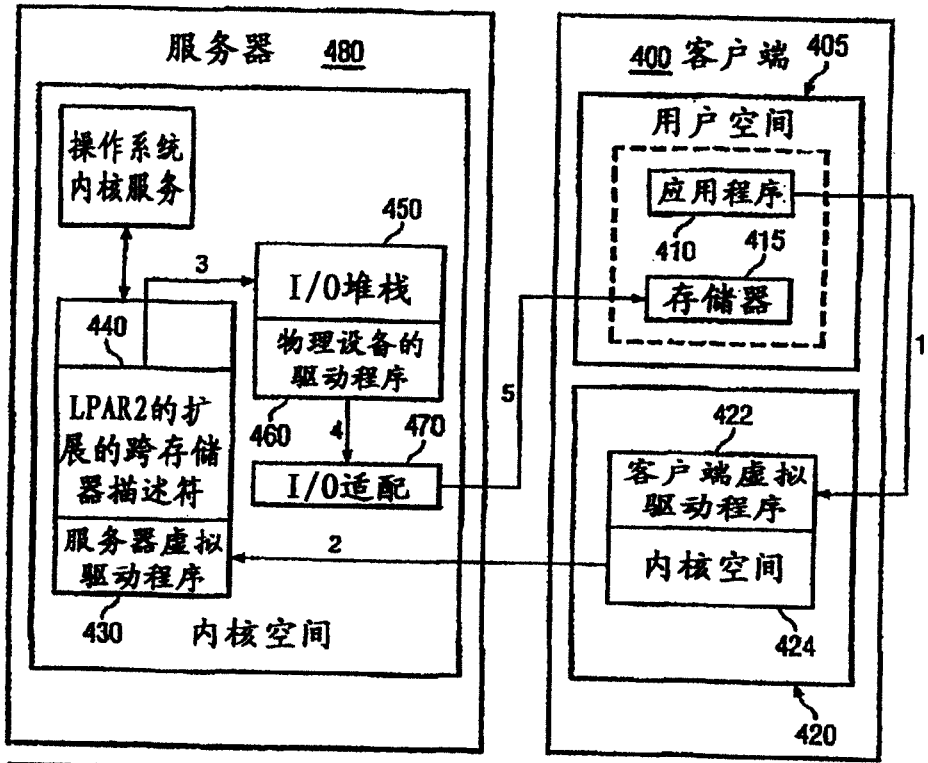

330
系统管理程序
1
LPAR 2
340
操作系统
本地存储器
336
设备树
338
2
设备树
LPAR2的扩展的跨存储器描述符
370
380
LPAR3的扩展的跨存储器描述符
LPAR4的扩展的跨存储器描述符
3
390 存储器
操作系统KERNEL服务器
325
操作系统
320
LPAR 1
310
312
LPAR 3
350
操作系统
本地存储器
334
设备树
316
314
LPAR 4
360
操作系统
本地存储器
332
设备树
1) 系统管理程序获取计算系统中的逻辑分区的设备树
2) 在初始化操作系统时,操作系统内核服务检查设备树以获取系统信息
3) 操作系统内核服务生成每一个逻辑分区的扩展的跨存储器描述符

图4



1) 接收到对I/O操作的请求，要求对LPAR2进行远程存储器操作
2) 将I/O请求传递到服务器虚拟驱动程序，并标识LPAR2的扩展的跨存储器描述符
3) 将LPAR2的扩展的跨存储器描述符和I/O请求传递到I/O堆栈
4) 通过I/O堆栈将LPAR2的扩展的跨存储器描述符和I/O请求从一个接口传递到另一个接口，并发送到I/O适配器使用跨存储器描述符建立远程DMA
5) LPAR2中的I/O适配器DMA到远程存储器

图5

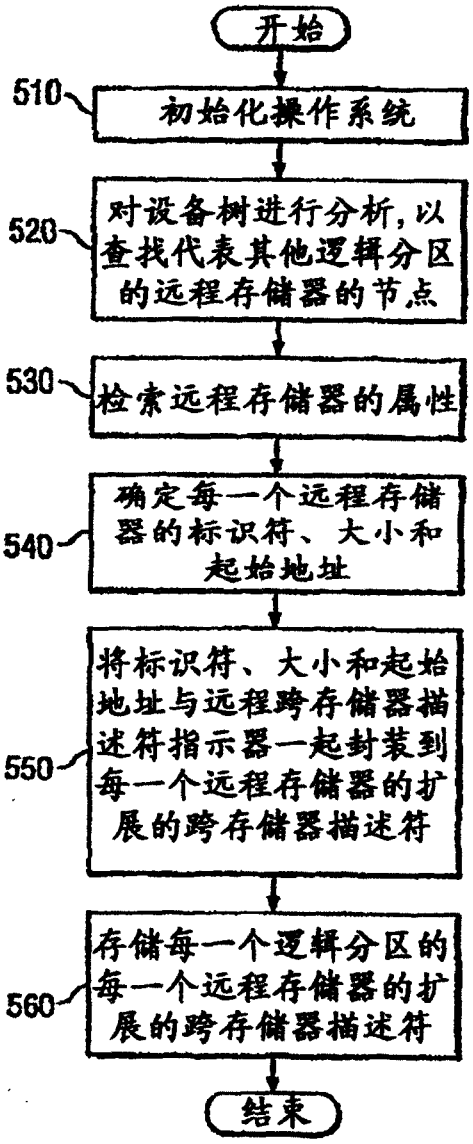
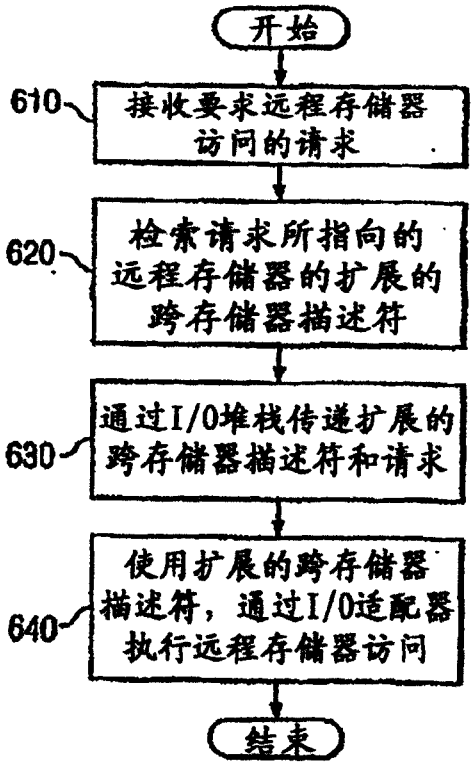

开始
510
初始化操作系统
520
对设备树进行分析，以查找代表其他逻辑分区的远程存储器的节点
530
检索远程存储器的属性
540
确定每一个远程存储器的标识符、大小和起始地址
550
将标识符、大小和起始地址与远程跨存储器描述符指示器一起封装到每一个远程存储器的扩展的跨存储器描述符
560
存储每一个逻辑分区的每一个远程存储器的扩展的跨存储器描述符
结束

图6


开始
610
接收要求远程存储器访问的请求
620
检索请求所指向的远程存储器的扩展的跨存储器描述符
630
通过I/O堆栈传递扩展的跨存储器描述符和请求
640
使用扩展的跨存储器描述符，通过I/O适配器执行远程存储器访问
结束