(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 103473181 B
(45)授权公告日 2017.06.13

(21)申请号 201310403763.6

(22)申请日 2008.01.24

(65)同一申请的已公布的文献号
申请公布号 CN 103473181 A

(43)申请公布日 2013.12.25

(30)优先权数据
60/897773 2007.01.26 US

(62)分案原申请数据
200880010154.1 2008.01.24

(73)专利权人 英特尔公司
地址 美国加利福尼亚州

(72)发明人 大卫·R·谢里登

(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 张懿 刘春元

(51)Int.Cl.
G06F 12/02(2006.01)
G06F 12/10(2016.01)
G11C 15/00(2006.01)

(56)对比文件
US 2002/0133668 A1,2002.09.19,说明书第[0007],[0010]-[0013],[0025]-[0027],[0030]-[0035],[0039],[0042]- [0044],[0047],[0051]-[0054],[0064],[0067]-[0069]段、附图8-9,11.
US 2002/0133668 A1,2002.09.19,说明书第[0007],[0010]-[0013],[0025]-[0027],[0030]-[0035],[0039],[0042]- [0044],[0047],[0051]-[0054],[0064],[0067]-[0069]段、附图8-9,11.
US 2003/0093616 A1,2003.05.15,说明书第[0011],[0021],[0028],[0057],[0060],[0062],[0078]段、附图1.
CN 1202653 A,1998.12.23,全文.

审查员 唐进岭

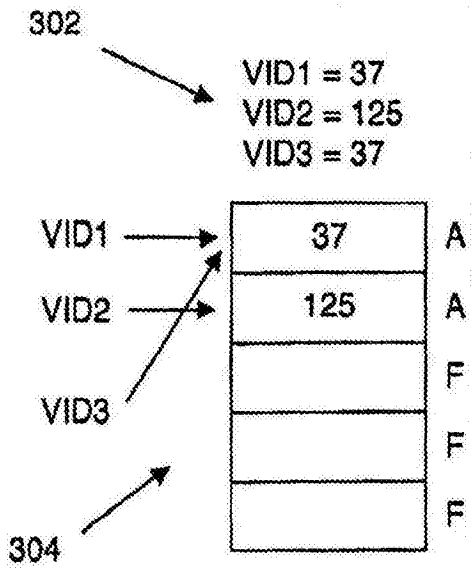
权利要求书2页 说明书12页 附图4页

(54)发明名称

分级式不可变内容可寻址存储器处理器

(57)摘要

改进的存储器管理根据分级式不可变内容可寻址存储器处理器(HICAMP)体系结构来提供。在HICAMP中,物理存储器被组织为两个或更多物理存储器块,每个物理存储器块具有固定的存储容量。对在任何时间点哪个物理存储器块中是现用的指示被提供。存储器控制器提供无重复写能力,其中将即将写至物理存储器的数据与写时刻的所有现用物理存储器块的内容相比较来确保在完成无重复写之后没有两个现用存储器块具有相同的数据。



CN 103473181 B

1.一种计算机系统,包括:

包括多个物理存储块的内容可寻址物理存储器,其中当所述存储块中的一些通过将其设置为现用状态而被创建时,其内容被认为是不可变的;以及

存储器控制器,其中将即将写至所述物理存储器的数据与写时刻被指示为处于现用状态的所述存储块的集合的内容相比以避免将数据写至被认为是不可变的存储块;

其中,所述存储器控制器提供指定需要两个或多个所述物理存储块来存储的数据项的表示的多块数据约定,其中所述存储器控制器避免写入被认为是不可变的多个所述存储块。

2.如权利要求1所述的系统,其特征在于,每个所述物理存储块具有相同的存储容量。

3.如权利要求1所述的系统,其特征在于,还包括与所述存储器控制器通信的处理器,其中所述处理器能够仅通过避免将数据写至被认为是不可变的存储块来向所述物理存储器写入。

4.如权利要求1所述的系统,其特征在于,所述物理存储器是易失性的。

5.如权利要求1所述的系统,其特征在于,被比较的所述存储块集合包括写时刻被指示为处于现用状态的所有所述存储块。

6.一种计算机系统存储器管理的方法,包括:

在包括多个物理存储块的内容可寻址物理存储器中通过将所述存储块中的一些设置为现用状态使得其内容被认为是不可变的来创建所述存储块中的所述一些;以及

使用存储器控制器来比较写时刻被指示为处于现用状态的所述存储块的集合的内容以避免将数据写至被认为是不可变的存储块;

所述方法还包括:

根据多块数据约定指定需要两个或多个所述物理存储块来存储的数据项的表示,其中所述存储器控制器避免写入被认为是不可变的多个所述存储块。

7.如权利要求6所述的方法,其特征在于,指示所述存储块中的一些处于现用状态包括:如果物理存储块已经被初始化或分配,则将其视为现用。

8.如权利要求6所述的方法,其特征在于,指示所述存储块中的一些处于现用状态包括:为每个所述物理存储块维持标记位,该标记位指示所述物理存储块是否是现用。

9.如权利要求6所述的方法,其特征在于,指示所述存储块中的一些处于现用状态包括:为每个所述物理存储块维持引用计数,该引用计数指示对每个所述物理存储块的现用引用次数,以及

将具有大于0的对应的引用计数的物理存储块标识为现用物理存储块。

10.如权利要求6所述的方法,其特征在于,还包括为一些或所有所述现用物理存储块将虚拟块ID映射到物理块ID。

11.如权利要求10所述的方法,其特征在于,所述存储器控制器使用按内容块读取BFBC指令来写入数据,该按内容块读取指令将块数据作为输入并且将块地址作为输出,

其中,如果所述块数据是执行所述BFBC指令之前所述集合的一个存储块中的数据的重复,则所述块地址是现用物理存储块的地址,

其中,如果所述块数据不是执行所述BFBC指令之前所述集合的一个存储块中的数据的重复,则所述块地址是新分配的物理存储块的地址。

12.如权利要求11所述的方法，其特征在于，所述块地址是物理块ID或对应的虚拟块ID。

13.如权利要求6所述的方法，其特征在于，所述现用物理存储块被组织为多个有向非循环图(DAG)。

14.如权利要求13所述的方法，其特征在于，所述多个DAG被限制为不包括多DAG循环引用环。

15.如权利要求6所述的方法，其特征在于，所述物理存储器的一些或所有内容的顺序读和比较能力提供内容可寻址性。

16.如权利要求6所述的方法，其特征在于，所述物理存储器的一些或所有内容的并行读和比较能力提供内容可寻址性。

17.如权利要求6所述的方法，其特征在于，还包括：

将所述物理存储器分区成N个组，其中N是大于1的整数；以及

根据具有N个可能输出的散列功能将块数据散列，以便提供所述块数据的散列值。

18.如权利要求6所述的方法，其特征在于，被比较的所述存储块集合包括写时刻被指示为处于现用状态的所有所述存储块。

分级式不可变内容可寻址存储器处理器

[0001] 本申请是申请号为200880010154.1(PCT/US2008/000983)、发明名称为“分级式不可变内容可寻址存储器处理器”的母案的分案申请，该母案的申请日为2008年1月24日。

发明领域

[0002] 本发明涉及计算机存储器管理。

[0003] 背景

[0004] 在常规的冯·诺伊曼计算机体系结构中，存储器被构造成大小固定的单元的线性阵列，按顺序地址进行索引。图1示出此常规体系结构的一个示例。指令102执行后的结果使存储器104的内容如图所示。虽然此方法易于实现，而且对于大小固定的应用数据单元应用起来也比较方便，但是对软件结构和技术提出的要求是处理大小可变且结构化的数据。

[0005] 对于大小可变的数据，软件通常实现一动态存储器分配器，该动态存储器分配器在存储器中定位一个至少与所需区域一样大的连续区域。然而，在长时间执行后，存储空间可能被分割成了较小的区域，以致于即使可用的存储器总量是充足的，但存储器分配请求仍会失败。诸如生成无用信息收集程序(generational garbage collector)等机制可以通过复制区域使之连续[方式来周期性地重新压缩存储器，但当这样的无用信息收集被调用时会影响应用程序正在进行中的执行，特别是在实时系统中或者在总体上要求可预测响应的系统中，这是一个不可接受的方面。而且，如果大小可变的数据项长度增加，则软件必须分配所需大小的新的连续区域并将数据复制到该新位置，而且使指向旧位置的所有引用变为现在指向该新位置。为了方便后一动作，一些软件通过大小固定的特定区域引入一个额外的间接指定级(level of indirection)，该特定区域提供指向大小可变的数据的实际指针，因此只有单个特定区域需要更新，但这是以每次访问时的额外间接指定为代价的。

[0006] 针对大小可变的数据的一种替代方法是使用指针从非连续的存储器单元来构造大小可变的数据类型(即，使用结构化的数据)。处理结构化的数据极具挑战性，因为在存在作为结构化数据的特征的复杂指针引用的情况下是难以确定何时存储器区域可得释放的。访问结构化数据表示下的数据还需要开销来通过指针间接指定以固定大小可变的数据项中的下一条目。

[0007] 对于以多个分离的进程运行的应用程序，因为一般必须将结构化的数据串行化并将该结果复制到分离的地址空间，然后将其串并转换以便共享该结构化的数据，所以结构化的数据引入进一步的开销。这之所以发生是因为，对于作为用来提供进程之间隔离的虚拟地址变换的结果的每个地址空间而言，用来将数据进行结构化的地址都是唯一的。共享还受到用于地址变换的存储页的与典型的应用数据单元(例如，32-128字节)相比较大的粒度(例如，8千字节或更大)的阻碍。结果，应用程序或者被组织成一个进程内的多个线程，放弃对单独地址的保护，或者在将结构化的数据于地址空间之间串行化、复制以及串并转换中付出重大代价。

[0008] 近来的和预期的技术发展使此标准冯·诺伊曼模型的缺点愈加突显问题。首先，存储器系统性能未跟上增强的处理器性能的步伐，使存储器性能成为对计算机性能而言越

来越具限制性的因素。因此，诸如复制和无用信息收集等存储器密集型操作变得相应地代价高昂。高速缓冲已是对付处理器/存储器速度失配的首要方法。然而，随着存储器大小增大、应用对象更大且更复杂、以及应用程序更数据密集化，高速缓存使用此常规存储器模型就变得显著地捉襟见肘了。

[0009] 作为另一个方向，计算机硬件越来越依赖于并行执行以实现性能收益。特别地，在单个微片上实现多个“核”是可行的，这在允许分享存储器控制器与高速缓存的同时改善成本效率。然而，由结构化数据引起的额外复制操作由于数据重复问题导致对高速缓存的使用效率低下。而且，诸如引用计数更新等额外更新和高速缓存线中与更新的假共享，导致存储器和高速缓存效能性进一步降低。

[0010] 作为最终趋势，应用程序变得越来越大且越来越复杂，这得益于存储器的大小和处理器的性能都在提高，但却增加了维持软件正确性的难度，特别是还要求持续的改进和特征。与此同时，应用软件采用了越来越时间攸关、任务攸关、甚至寿命攸关的功能，使应用软件的可靠性变得重要得多了。

[0011] 针对这些和其它理由，已考虑了替代的存储器体系结构。例如，在US4,989,137中，用户处理器只经由作为存储器管理系统的一个组成部分的绑定寄存器单元访问存储器。这样一来，存储器管理系统可对用户处理器隐藏物理存储器组织的低级别细节，改为将逻辑存储器抽象呈现给用户处理器。在US4,695,949中描述了块结构存储器，其中为每个块维持引用计数，从而减轻对频繁的无用信息收集的需要。

[0012] 在US5,784,699中，考虑了具有若干种标准块大小的块结构存储器系统。存储器分配请求被上舍入到最接近的标准块大小。在US5,950,231中，考虑了使用指针栈来控制的块结构存储器管理系统。在US5,247,634和US5,864,867中，考虑了基于树的使用的存储器管理。

[0013] 然而，以上这些方法和标准冯·诺伊曼高速缓存方法都有其困难，特别是针对结构化的数据而言。因此，提供改进的存储器管理，尤其是提供针对大小可变且结构化的数据的改进的存储器管理，在本技术领域是一种进步。

发明内容

[0014] 根据分级式不可变内容可寻址存储器处理器(HICAMP)体系结构提供一种改进的存储器管理。在HICAMP中，物理存储器被组织为两个或更多个物理存储器块，每个物理存储器块具有固定的存储容量。

[0015] 提供一种在任何时间点上哪一个物理存储器块是现用块的指示。存储器控制器提供非重复写能力，其中将待写至物理存储器的数据与写之时刻所有现用物理存储器块的内容相比较来确保在完成非重复写之后没有两个现用存储器块具有相同的数据。在有重复的情况下，使用现有块而非用相同的值来创建另一个块。

[0016] 附图简要说明

[0017] 图1示出常规计算机存储器使用的示例。

[0018] 图2是根据本发明实施例的计算机系统的示意性框图。

[0019] 图3a-d示出根据本发明实施例的计算机存储器使用的示例。

[0020] 图4示出适于结合本发明实施例使用的虚拟-物理块ID映射(VPBIM)的逻辑结构的

示例。

[0021] 图5a-d示出对应于图3a-d的示例的VPBIM示例。

[0022] 图6a-b示出其中同一多块对象可用两种不同方式在存储器中表示的示例。

具体实施方式

[0023] 为有助于更好地理解本发明，将首先以相对抽象、不依赖于实现的方式来考虑本发明实施例的关键方面，然后借助说明性示例提供一些进一步的实现细节。

[0024] 1)关键方面

[0025] 为简洁计，宜将本发明的各实施例作为分级式不可变内容可寻址存储器处理器(HICAMP)体系结构的示例。HICAMP体系结构在几个基本方面有别于标准冯·诺伊曼结构体系。

[0026] 第一，在HICAMP中，物理存储器被组织为两个或更多个物理存储器块，每个物理存储器块具有固定的存储容量。第二，提供对于在任何时间点上哪个物理存储器块是现用块的指示。第三，存储器控制器提供无重复写能力，其中将待写至物理存储器的数据与写之时刻所有现用物理存储器块的内容相比较来确保在完成无重复写之后没有两个现用存储器块具有相同的数据。

[0027] 从该HICAMP体系结构的这些方面得出几个主要优点。

[0028] 1)因为对存储器的分配按照大小固定的物理存储块进行，所以消除了与存储器碎片化有关的常规问题。重复抑制使这些块可被高效地定位和管理。

[0029] 2)由于对重复的抑制，给定量的应用数据所需的存储器的量可相对于应用数据的大小来分析和限制。

[0030] 3)因为数据块为现用时不被修改，所以在地址空间内和地址空间之间数据都可被安全地共享，从而存储器复制得以减少。

[0031] 4)因为当且仅当存储器中两个对象由同一物理块表示时它们是等同的，所以可以高效地实现其等同/非等同比较。

[0032] 5)对重复的抑制使可用物理存储器得到高效使用。

[0033] 6)因为可以对许多公共数据结构执行原子性非阻塞更新，所以并行编程被简化并变得更高效。

[0034] 在优选的HICAMP实施例中，无重复写的概念被扩展到也包括需要多于一个物理存储块来存储的数据项(简称多块数据)。这一点可以通过为需要两个或多个物理块来存储的任何数据项提供一个指定唯一表示的多块数据约定来实现。当多块数据被写至存储器时就可实施多块数据约定，从而使无重复写确保在该组现用物理存储器块中不存在多块数据的任何实例的重复。

[0035] 作为多块无重复写的效果的一简单示例，考虑字符串“abcde”在其中物理块存储容量为三个字符的系统中的表示。这样的字符串可以总共由三个块来表示，其中块1包含该字符串的头部，块2包含该字符串的尾部，而块3包含指向块1和块2的指针。没有规定多块唯一性的约定，该示例字符串可以表示为(“abc”，“de”)或者表示为(“ab”，“cde”)，其中第一项是块1的内容，第二项是块2的内容。通过实施(或支持)这样的多块唯一性约定(例如，先填头部，或先填尾部)，排除了此类多种表示的可能性。结果，其指针指向块1和块2的块3是

存储器中字符串“abcde”的唯一表示，从而对多块重复做出抑制。

[0036] 较佳地，物理存储块的每一个都具有相同的存储容量，以便简化存储器管理。而且，与诸如通过在磁或光介质上记录来实现的文件系统等持久性存储器相反，这里的物理存储器最好是易失性存储器(例如，诸如动态RAM等计算机主存)。

[0037] 图2示出一优选实施例，其中HICAMP通过提供与物理存储器202通信的存储器控制器220来实现。一个或多个处理器230可以经由存储器控制器220来访问物理存储器202，该存储器控制器220将HICAMP型组织的存储器模型提供给处理器。在此示例中，有三个处理器，210、212以及214。因此，HICAMP可应用于单处理器或多处理器环境中。关于此示例中的存储器控制器220的其它细节描述如下。

[0038] 可以用硬件和/或软件的组合来提供对哪个物理存储块是现用块的指示。此外，在实践本发明时可采用对“现用块”的各种定义。例如，如果物理块在整体系统初始化后已经由HICAMP系统初始化或分配，那么可将其视为现用。以此方法，无法收回先前被使用过但现在不在使用中的物理存储块。由于一些计算机系统被设计成当存储器短缺时可以再次初始化，对于这样的系统而言这样的对“现用”的相对最小指示就足够了。

[0039] 用于提供“现用”指示的另一种方法是为每个存储块维持一个标志，该标志指示存储块是否为现用。这样的标志可根据先前描述的由HICAMP初始化/分配的方法来设置。或者，针对块现用性的标志可以在确定哪些块为现用的单独操作中设置(例如，如在标记和清扫式无用信息收集中所完成)。

[0040] 用于提供“现用”指示的又一方法是为每个物理存储块维持一引用计数，其中现用块对应的引用计数大于0，而非现用块的引用计数等于0。在某些情况下，这些引用计数可区分具有各种特定区域的引用，诸如处理器寄存器中的引用、物理存储块中的引用、和/或虚拟-物理映射(VPBIM)中的引用。

[0041] 在一优选实施例中，每个块提供一组标记，该组标记指示块中哪些字段包含物理和虚拟块引用或普通数据，使这样的块引用可以作为引用计数、标记和清扫式无用信息收集或类似的“主动”确定方案的一部分来处理，并且防止应用级处理没有HICAMP实施例的知识就产生块引用。其它标记可以支持检测循环引用以及提供特殊化的更新语义，诸如用于高争用数据块的更新时合并。

[0042] 在HICAMP的优选实施例中，提供VPBIM来针对物理存储器的一些或所有内容将虚拟块ID映射至物理块ID。通过提供这样的映射，就可以在限制开销的同时原子性地更新结构化的数据的表示。例如，对象中的字符串描述字段可被实现为存储虚拟块ID的一个块中的存储单元。该描述通过使用根物理块创建一个修正的字符串数据项和将此物理块的标识符存储在对应于虚拟块ID的映射条目中得到更新。

[0043] 图3a-d示出结合虚拟和物理块ID(分别是图上缩写的VID和PID)的HICAMP存储器使用的一些示例。为了简化说明，此示例为每个块存储单个值。在图3a上，指令302的执行致使存储器304的内容如图中所示。更具体而言，指令302中的第一赋值使值37被存储在新的物理块(即，PID1)中，并将VID1与PID1相关联，因为该赋值是到VID而不是到PID。同样地，指令302中的第二赋值使值125被存储在新的物理块(即，PID2)中，并将VID2关联于PID2。由于对应于VID3的值是已经存储的值的重复，所以指令302中的第三赋值只是将VID3关联于PID1。这种对重复的抑制与图1的常规存储器模型形成鲜明对比，在图1的常规存储器模型

中存储器访问是经由PID进行，而且重复的值被存储在存储器中。

[0044] 在图3b上，执行指令306致使存储器308的内容如图中所示。指令306唯一有别于指令302之处在于增加了最后的使VID1具有值25的赋值。此第四指令使值25被存储在新的物理块(即，PID3)中，并将VID1与PID3相关联。

[0045] 在图3c上，指令310的执行致使存储器312的内容如图中所示。指令310唯一有别于指令306之处在于增加了最后的使VID3具有值125的赋值。因为值125已经在存储器中，所以此第五指令只是将VID3与PID2相关联。

[0046] 在图3d上，指令314的执行致使存储器316的内容如图中所示。指令314唯一有别于指令310之处在于增加了最后的使VID4指向VID3的赋值。此第六指令使地址VID3被存储在新的物理块PID4中，并将VID4关联于PID4。

[0047] 此示例说明可以通过用VID指向存储器内容在消除所有重复的物理存储的同时提供赋值语句和指针的普通逻辑。一旦物理块被创建，其内容保持不可变。在图3a-d的示例中，字母“A”出现在现用的物理存储块之后，而字母“F”出现在空闲的(即，可供分配的)存储块之后。在图3c-d上，物理块PID1被标识为空闲，因为不存在对它的引用。

[0048] 图4示出适于实现VPBIM的逻辑结构的示例。在此示例中，VPBIM中的每个条目包括VID、对应的PID以及对该VID的引用数。图5a-d示出具有此结构且分别对应于图3a-d中示例的VPBIM示例。由于如这些示例中所示的那样，同一PID可能对应于几个VID，VPBIM往往是多对一的VID-PID映射。因此，VID的使用可被视为管理对同一物理数据块的多个不同引用的系统性方式，这是一种由于作为HICAMP的特性消除了存储器中的物理重复后所自然引起的情况。在一可选实施例中，VID不需要是VPBIM中的显式条目。相反，映射可以是隐含的。例如，(PID，引用计数)对的阵列可以起到VPBIM的作用，其中使阵列索引充当VID。可以用与任何其它多块数据项相同的方式将VPBIM存储在HICAMP型组织的存储器中。

[0049] 在一优选实施例中，HICAMP无重复写能力是根据按内容块读取(BFBC)指令提供的，其中将块数据作为输入并提供块标识符作为输出。有两种情况要考虑。如果输入块数据是存在于现用物理存储块中的数据的重复，那么由BFBC指令返回的标识符是此现存的现用存储块的地址。

[0050] 如果输入块数据不是存在于任何现用物理存储块中的数据的重复，那么由存储器控制器分配新的物理数据块，其内容被设置为输入块数据，而且这个新的现用物理存储块的块地址由BFBC指令返回。

[0051] 通过按照BFBC指令对存储器访问进行结构化，管理重复抑制的过程可以由存储器控制器来执行，而且无须在应用级分别考虑以上两种情况。更具体而言，由于以BFBC存储器访问表达的应用程序算法通过实现BFBC抽象来处理，所以这种算法不需要去关心存储器重复抑制的细节。在采用VPBIM映射的情况下，由BFBC指令返回的块地址可以是物理块ID或虚拟块ID。通常，应用级BFBC指令返回虚拟块ID会更有用。

[0052] 在一优选实施例中，现用物理存储器块被组织成两个或更多个有向非循环图(DAG)。在这样的情况下，每个多块数据项都有其自己的DAG，以DAG是非循环性的这一条件排除了DAG内部的引用闭循环，而且DAG的方向性提供用于从整体上指向多块数据项的明确无误的根块。为了排除全局性循环引用环，更倾向于不在DAG的组中包括多DAG循环引用环(例如，任何包括这些DAG中的两个或更多个的指针引用闭环)。

[0053] 实现HICAMP可以依靠提供物理存储器内容可寻址性的各种方法。一个方法是提供对物理存储器中一些或所有内容的顺序读比能力。另一个方法是提供对物理存储器中一些或所有内容的并行读比能力。例如，可以实现顺序法，其中单个比较器被安排成顺序地读取现用物理存储块的内容并将其与输入数据相比较。类似地，并行法可以通过提供(例如，用硬件)分离的对应于每个物理存储块的比较器来实现，从而可同时执行对现用块内容与输入数据的所有比较。因此，在确定是采用顺序比较还是并行比较(或某种混合方法)来提供内容可寻址性时，要考虑速度与成本之间的折衷。

[0054] 在一优选实施例中，通过将物理存储器分为N个组(bank)来提供对HICAMP的内容可寻址性，其中N是大于1的正整数。当数据被写至该存储器时，向块数据应用一个具有N个可能的输出的散列功能于产生一散列值。散列值被用来选择对应的存储器组，该组将根据通常的HICAMP约定来存储数据(即，在相关组中创建新的块——当且仅当不会由此在该组中产生重复时)。这样一来，只要求针对内容寻址的读比能力一次处理一个组，而非整个物理存储器。如果每个组有M个块并且配置M个比较器，则可以通过这样的方案提供快速并行比较，这可以比实现全块级的并行性(即NM个比较器)大大地降低成本。

[0055] 例如，假设要根据以上方案将字符串“abc”存储至存储器。假定“abc”散列到3，那么在物理存储器中“abc”可以被存储于现用物理存储块的唯一所在在组3中。从而，为了防止重复，只检查组3中的块就足够了。

[0056] 图6a-b示出多块数据对象的简单示例。图6a示出在本发明的一个实施例中通过指令602将三元素列[1,2,3]赋值到VID1的结果。所产生的存储器604的内容可以理解如下。因为在此示例中没有元素是重复的，所以每个列元素得到其自己的物理块，诸虚拟ID VID3、VID4和VID5对应于阵列元素。该列的根在由VID1引用的块中，其内容是指向该列第一元素的指针(即，VID3)和指向该列其它部分(即，指向列[2,3])的指针。在VID2中，有指向列[2,3]第一元素的指针(即，VID4)和指向列[2,3]第二元素的指针(即，VID5)。

[0057] 图6b的示例与图6a的示例相似。图6b中存储器内容606与图6a中存储器内容604之间的差别仅存在于VID1与VID2的块引用的内容。更具体而言，在图6a的示例中，列是从其末尾逐步建立，而在图6b的示例中是从其开头逐步建立。为了允许对多块重复的消除，规定多块唯一性的约定(如上所述)为任何给定系统所支持的所有多块数据结构指定唯一的组织是重要的。有了这样的约定，同一对象将不可能具有两个不同的存储器表示，如图6a-b所示。

[0058] 2)实现细节

[0059] 块组织

[0060] 在一示例性实施例中，HICAMP物理存储块可以被结构化如下：

[0061] refCount|tags|inLevel|exLevel|data。

[0062] refCount(引用计数)字段包含对此块的完全引用的次数。refCount为零指示该块是空闲的且可供再使用。

[0063] Tags(标记)字段对应于相关联数据字段的每个子字段分别指示：

[0064] i)00-data(数据)。

[0065] ii)01-intraRef(内部引用)-包含指向多块树结构中在此块内部并从属于此块的块的blockId(块标识)。即，子树引用，使此块引用计数加一。如果任何子字段是intraRef，

那么要求块中所有的子字段都是intraRef。

[0066] iii)11-extraRef(外引用)-包含指向另一个块的blockId,extraRef使该其它块引用计数加一。iv)10-backRef(反向引用)-包含指向另一对象的blockId,但不表示该其它对象引用计数加一。extraRef值和backRef值是虚拟blockId。tags字段可以包括单个“合并一更新”标志,该“合并一更新”标志指示该块应该于更新时与当前块内容合并,而非替代这些内容。inLevel(内级)字段指示从此节点经由intraRef到不包含intraRef的节点的最大距离。例如,在大小可变的对象的典型分级(树或DAG)表示中,这个级是树或DAG中此节点的高度。exLevel(外级)字段的要求是至少比如下的任何节点的exLevel大1:它拥有到该节点的intraRef1,不论是直接地引用还是间接地通过它可以经由intraRef能够到达的节点而引用。例如,如果此节点是表示多块对象的树的根,那么其exLevel大于此对象对其拥有extraRef(相当于常规编程中的智能指针)的所有节点的exLevel。将此限定推行于exLevel是一种确保在HICAMP存储器组织中没有有害的循环引用环的方式。

[0067] 分析

[0068] 为了评估HICAMP对于各种情况的适用性,限制添加至存储器系统和控制器的字段的大小是有帮助的,讨论如下。

[0069] 不同于常规存储器系统,在本发明系统中对给定块的引用次数因每个内容至多存在单一拷贝而受到限制。特别地,最差情况是所有DAG具有一个块的单个共用前缀,因此此块具有最大引用次数。每个DAG必须具有一内部节点块来引用此共用块,另外它必须具有至少一个唯一的块来确保每个根块是不同的。而且,每个内部节点需要由另一内部节点引用,一直到某个根节点。那么,假设创建最小附加唯一引用DAG需要2个块且每个最小DAG引用这些根节点需要另一个块内节点,则使用64-字节的块和太字节(2^{40})的存储器(因此2^{33}块)时,最差情况下的引用次数被限定至2^{32}个这样的根节点,(即,32-比特的refCount是足够的)。在此结构中,作为选择,叶子是共用块和唯一块的序列,其中另一N用作整个DAG中的内节点。

[0070] 通过为每个块存储inLevel和exLevel避免了引用循环。inLevel字段需要足够大以容纳最大的intraRef结构化对象。比特数是loglogN/B,其中N是最大的单个对象,B是每个块的字节数。因此,6比特将容纳大至$B*2^{63}$的对象。exLevel需要容纳extraRef的深度,这个深度通常远小于100。例如,指向具有子对象的对象的目录基本上处于exLevel(外级)3。因此,8比特似对于这一字段就非常适合。

[0071] 在一实施例中,块是20字节的数据、4字节的refCount(引用计数)、2字节的level(级)、1字节的tag(标记),招致近25%的空间开销。较大的块允许每一块存储4个blockId,支持四叉树作为分级数据表示。另外的实施例可支持更大的块大小和多重块大小,以进一步减少开销。

[0072] 存储器控制器

[0073] 在一示例中,图2的存储器控制器220包括3个主要组件。blockId/偏移量读取器204响应于处理器请求来定位并返回一个存储器存储中对应于在分级块结构内的指定偏移量的其根位于blockId所标识的块的数据块。VPBIM206在虚拟块ID(vBlockId或VID)与物理块ID(pBlockId或PID)之间转换,而且通过为PID维持引用计数来管理这些映射的分配和释放。块数据目录208向块实现指定的数据和标记规范(例如,如上所述),如果这样的块尚不

存在则分配这样的块并将其初始化。已知有各种技术可实现此等映射。在一实施例中，二元内容可寻址存储器可用作数据存储。在此情况下，refCount和level(级)可被分别存储在较廉价的DRAM中。在另一实施例中，映射可使用可供选择的查找数据结构，诸如前面所述的基于树或哈希表的实现。

[0074] 在一示例性实施例中，存储器控制器管理被组织成B字节的块的存储器DRAM组，其中B的期望值在16至64字节的范围内，但就每个系统而言(至少在系统运行期间)是固定的。它还具有一个32比特引用计数字段的阵列，每个块一个。一种实现是一个DRAM阵列，DRAM阵列中对应于系统中的每个物理块都有一个条目。存储器控制器提供以下操作，在给定索引处使引用计数字段原子性地加1和减1，当引用计数减少至零时，释放这个块，如下所述。

[0075] 块分配可以通过使用空闲块的引用计数字段将其链入块空闲列表中来控制。即，空闲存储块的引用计数字段包含空闲列表中的下一空闲存储块的索引。系统一初始化，所有空闲块立即在此空闲列表上排队。当需要新的块时，存储器控制器使空闲块从空闲列表的头部出列，而且该空闲块的引用计数被置位到1，对应于新的引用。当块的引用计数变为零时，存储器控制器将该块添加在空闲列表的头部。相应地，可将坏的存储块从空闲列表中移除，因此不分配它们，这与用在页级的常规技术相似，但粒度更细。通过维持空闲列表尾指针，释放的块可被添加到该空闲列表的末端，从而它们在尽可能长的时间不被使用。当通过诸如在存储器芯片的整个使用寿命中支持有限次的写操作的闪存等技术来实现存储器时，此可选的改良提供了一种块间“磨损均匀化”的形式。

[0076] 一种可行的做法是将一额外比特与引用计数机制相关联来指示其以此形式用作空闲的“下一”字段，以便避免诸如对空闲块的错误引用等错误行为。

[0077] 以此方法，引用计数字段需要足够大以便存储blockId。或者，可以有K个空闲列表，从而使第i个空闲列表中的所有块在其低位比特中有i，这样就不需要存储blockId了。下面将对在K个空闲列表间的块进行分区的实用程序作为实现内容可寻址查找或按内容块读取(BFBC)的一部分来进行进一步描述。

[0078] 按物理块ID的块读取以与在常规存储器中一样的方式执行。存储器控制器将blockId解码到一DRAM组中，亦即此组中的行/列，发出对此块的读取并返回数据。更非同一般的方面是支持按内容块读取(BFBC)，如下所述。

[0079] 在理想的或逻辑的环境中，主存被实现为二元的内容可寻址存储器(CAM)，其宽度对应于块大小，不含引用计数字段。因此，按内容块读取(BFBC)存储器控制器操作将块数据传递给CAM并且如果块或“行”ID存在则将其接收回，否则接收回不存在指示。在后一情况下，按内容块读取(BFBC)存储器控制器操作如上所述分配块并向块写入相关数据。基于在整个存储器系统中的数据唯一性，即不会存在两个命中，存储器可被分成多个组，其中在每个组中并行执行查找。通过把比较逻辑嵌入各DRAM芯片中，可使此方法具有可行性。然而，当下的二元的CAM存储器相对于常规DRAM而言还是非常昂贵的且功耗大。

[0080] 允许使用常规DRAM的实现降低了比较器的数量，组中的每K行只有一个比较器，而非每行有一个比较器。块内容随后被散列为值0至K-1，例如，h，而且要求每个比较器将数据与其第h相关联行相比较。如果该块匹配，则比较器报告命中和块号码。为了使其正确工作，在K个空闲列表中维持空闲块。若散列为h的数据未命中，立即由第h空闲列表分配一个块来存储该数据。

[0081] 使用以上方法，存储器控制器可以包含C个比较器并访问C个独立组中的存储器，每个组都通过常规DRAM芯片来实现。存储器控制器接着执行并行的存储器读取和比较，以及未命中时块的分配。用适当的散列功能，耗尽一空闲列表应该不可能比耗尽所有空闲列表提前很多。反言之，空闲列表的分隔应该不显著减小存储器的有效大小。如前所述，K个空闲列表可以被用来减少当它用作树列表中的“下一”链接时需要存储在引用计数字段中的比特数。

[0082] K的大小和比较器的数量可以就HICAMP的特定实现具体而定，对于软件是透明的。它可以取决于存储器控制器上可行的I/O引脚的数量。而且，在某些情况下，每次BFBC请求可以为每个DRAM的组发出多个读取(read)的，通过每一次BFBC操作多次使用比较器而有效增加比较器的数量。

[0083] VPBIM机制

[0084] 虚拟-物理blockId映射(VPBIM)可以被实现成由虚拟块ID(vBlockId)做索引的存储器阵列。每个条目具有字段：

[0085] [pBlockId|refCount)]

[0086] 使用pBlockId(物理块ID)字段将空闲条目链接在一起，与针对pBlock(物理块)所描述的方案相似。由于blockId对应40比特而且每个引用计数32比特，每个条目是9字节。存储器控制器被配制成支持相对于pBlock足够多的条目，因此VPBIM条目不是限制性的资源。假定比率是每4个字的存储器1个指针、每个块4个字的数据，则存储器控制器可以每个块提供一个VPBIM，因此pBlockId字段对于空闲列表而言足够大。理想的情况是，存储器控制器可被配置来支持不同大小的VPBIM。

[0087] 对于VPBIM，存储器控制器支持以下操作：

[0088] a)分配VPBIM条目并使用给定的pBlockId和引用计数1来将其初始化，使相关联的pBlockId引用计数加1。这只是包括使空闲VPBIM条目出列并将其初始化。

[0089] b)返回对应于给定vBlockId的pBlockId。这就是用vBlockId做VPBIM阵列的索引并返回该条目中的pBlockId。

[0090] c)使给定的vBlockId引用计数加1。这就是用vBlockId做VPBIM阵列的索引并使那个特定区域处的引用计数加一。

[0091] d)使给定的vBlockId引用计数减1，如果引用计数为零则释放该条目，使pBlockId减1并将此VPBIM条目添加至空闲列表。

[0092] 以上存储器控制器操作可以使用固件和用于诸如分配等复杂操作的内部微控制器来实现。简单的性能攸关性操作可以使用硬连线逻辑。存储器控制器性能受DRAM性能限制，如同常规体系结构一样。因此，HICAMP处理性能高度取决于在处理器单元级上对高速缓存的有效使用，如下所述。

[0093] 高速缓存

[0094] HICAMP处理器单元是基于实现常规的寄存器到寄存器算法的、逻辑的等指令和存储器寄存器负载操作的CPU块，但是被进一步扩充以用于与新颖的HICAMP存储器系统接口的特殊化操作。如常规体系结构所充分证明，此处理器单元包括以时间和空间局部性提供所访问的数据的高效访问。对于HICAMP的重大性能挑战是提供能对常规处理器单元有竞争力的存储器高速访问。对于后者而言，一个关键方面是有效的高速缓存。

[0095] HICAMP处理器高速缓存被结构化为其大小与存储块大小兼容，通常与块的大小相匹配的高速缓存线的集合，这与常规处理器高速缓存一样。同样地，HICAMP处理器高速缓存包括常规的高速缓存目录，将blockId映射于高速缓存线，可能赋予某些合理的集合结合性，比如4或8。此高速缓存机制可以用HICAMP数据-blockId映射，即内容可寻址性来扩充。此映射的实现与存储器控制器的BFBC操作相似，但是是应用于高速缓存中的数据。高速缓存中的块也组成对存储器控制器的引用。当把块从高速缓存中删除时，其引用计数在存储器控制器中减1(因此替换必须跳过寄存器所引用的任何块)。

[0096] 高速缓存支持如果给定的物理blockId存在则检索对应于它的数据块的操作。高速缓存还支持如果给定块的数据的pBlockId存在则将其返回，否则此映射即依赖于存储器控制器。高速缓存独立于要被比较的数据的块来存储其物理blockId。这是个对应于pBlockId的宽度的单独的组，C个各有R行的存储器组具有R*C个条目，即每个高速缓存线一个条目。只要块-pBlockId映射得到支持，这就可以与支持pBlockId查找的高速缓存目录相结合。

[0097] 在未命中的情况下，如通过高速缓存中的额外比特所指派，高速缓存还支持在称为“开”的瞬态下的块分配。这样可以使块被增量写入高速缓存中，然后在完成时“被提交”到存储器，只在那个时刻方确定对应于块数据的系统pBlockId。特别地，当块的pBlockId从其初始分配寄存器中被移走时或当块被存储进存储器时，或者当块正在从高速缓存中被移除时，该块被提交向存储器。该提交需要为数据确定系统指派的pBlockId，而且可能要将线移到该数据散列至其上的行。提交有可能确定具有此数据的块已经存在于该高速缓存中，导致原始的块被释放，在片段中由现存块和高速缓存线代替。

[0098] 高速缓存支持“打开”新的或现存的高速缓存线以供写入。如果是新的，则新的线被分配，标记为打开。如果是现存的而且该线具有另外的引用，则数据被复制到新的高速缓存线上而且此新的高速缓存线接着被标记为“打开”。提交高速缓存线将其对修改关闭，并如上基于内容确定其blockId。

[0099] 高速缓存在任何给定的时间永远只包含数据块的至多一个拷贝这一事实超越常规的处理器高速缓存改善了高速缓存的利用。有各种软件技术预计都会提高高速缓存中的共享，比如将数据和代码在块边界上对准和标准化的指令序列的使用。

[0100] 存储器操作

[0101] 处理器按照vBlockId和偏移量来访问数据。例如，程序的常规代码段可以原状引用，因此程序计数器是由blockId指示的一个代码段中的偏移量。硬件遍历对象的分级结构来定位指定偏移量处的数据并将该数据返回，或者如果不存在则指示存储器异常。当数据被返回时，即被加载到处理器寄存器中，该处理器寄存器保留指示数据是否是blockId的标记，而如果是blockId则保留其引用类型。因为硬件查找在intraRefblockIds中继续，所以响应于访问而返回的数据中的blockId是或者extraRef或者是backRef，即它实际上是应用级的指针或引用。

[0102] 数据读取可以在硬件中设置为索引加载操作，指定blockId和偏移量，返回指定的特定区域处的数据或blockId，或者抛出意外/中断。此操作沿树向下递归，如果指定的偏移量处的数据存在则将其返回，否则扔出指示访问失败的异常。

[0103] 在一实施例中，处理器支持一个或多个迭代器寄存器，每个迭代器寄存器保持通

过多块数据DAG向当前存储器特定区域有效地映射和向下一特定区域有效地加1所需的状态。读和写访问由通过迭代器寄存器的非直接读和写替代，这与常规计算机处理器中地址寄存器的使用相似。

[0104] 在一优选实施例中，硬件按内容块读取操作采用指定所需数据和块的标记的参数并将该块的虚拟blockId返回。如果BFBC参数指定blockId，则参数只可以从已经保存这些blockId的寄存器中产生。当块的blockId在使用BFBC操作的新的块中被指定时，该块的引用计数也加1。

[0105] 读取包含blockId的现存块以及BFBC是处理器可使blockId出现在寄存器中的唯一方式。BlockId不能以另外的方式通过计算产生。因此，硬件系统可以维持每个块的精确引用计数，而一个进程只可以访问它已经接收到了其blockId的对象，或者是通过有效地指定其内容或者是被(直接地或间接地)传递以blockId。

[0106] 在一实施例中，为了说明基本操作，HICAMP程序以将字符串的字符生成到块寄存器中并随后进行BFBC操作获取包含那些字符的块的blockId的方式来创建该字符串，该字符串末端用零来填充。如果字符串的长度大于可以纳入块数据部分的字符个数B，那么程序为字符串中的每B个字符创建一个块，并且通过使用获取包含对应于该字符串的前缀和后缀的blockIds的块的BFBC操作来有效地递归级联这些块。可以以类似的方式来例示大于单个块的数据大小的任何连续数据对象，例如，阵列。

[0107] 在一实施例中，比块大的对象可以实现为具有根节点的二叉树，根节点包含用于前缀树的blockId和用于后缀树的blockId。可以保留blockId0来表示数据为0。在另一实施例中，树可以是四叉树，每个内节点具有多达4个子节点。我们提议参考将状态的逻辑连续部分表示为一个对象的块树，无论是单个块或是非平凡树。

[0108] 对象具有了针对另一对象的根节点的extraRef即可有效地包含对另一对象的指针，被存储为一个虚拟blockId，由VPBIM映射到物理blockId。因此，可以通过原子性地更新VPBIM使虚拟blockId映射于另一个对象的物理blockId，对象就变成指向该新的对象了。在一实施例中，比较并交换操作支持此原子更新，条件是当前映射包含指定的物理blockId。因此，基本更新可以通过以下操作来获取：创建即将被更新的对象的“虚拟”复制(实际上只是对原始对象的另一物理blockId引用)，修改此复制，从而向新的对象提供新的物理blockId，并且随后如果对象自虚拟复制以来没有改变则自动更新VPBIM映射以便映射到新的物理blockId，否则异常中断更新。接着软件可以按照使用无锁定编程的常见模式来重试该操作。

[0109] 当块被创建得具有到vBlockId的新引用时，对应的vBlockId上的引用计数加一。同样地，当包含vBlockId的块被释放时，对应的vBlockId上的引用计数减一，当该引用计数变为零时实际上将它释放并移除它对对应的物理blockId的引用。通俗而言，VPBIM为HICAMP中的“智能指针”等效物保持引用计数，即extraRefs。

[0110] 要求I/O经历以上机制。即，不存在到存储器的直接写。每个DMA I/O设备只请求具有给定数据的块，建立如诸如包缓冲器或磁盘存储块等对象，如上所述。这消除了预分配I/O缓冲器的需要。如果需要，只需限定接收的量。一般，与连网相关联的复制在很大程度上得以消除，尤其是在网络有效负载起始于块边界的情况下。

[0111] 已经根据几个示例性实施例描述了本发明，所述示例性实施例旨在在所有方面都

起说明性作用而非限制作用。因此，本发明可接受用硬件和/或软件的具体实现中的多种变化，本技术领域中具有普通技能的人员可以从这里所包含的描述中得出此结论。一变化涉及其中HICAMP被实现为通过单个逻辑存储器控制器操作的多个处理器的实施例，如图2所示。本发明还可以被实现得具有多个存储器控制器和处理器芯片，通过在分配块和更新到VPBIM之后立即在不同单元之间进行适当的同步来实现。在另一变化中，尽管本发明是针对通用计算机而描述的，但也可以应用于网络包交换，允许带有重复抑制的单元/块分配，提供无对多播的专门优化的内容共享。本发明还可应用于具有存储器系统的其它设备。在又一变化中，归于硬件的机制可以用微码或受保护的软件或者甚至是在可靠性上有某些损失的未受保护的机制来替代实现。所有这些变化和其它变化被认为在本发明的范围和要旨之内。

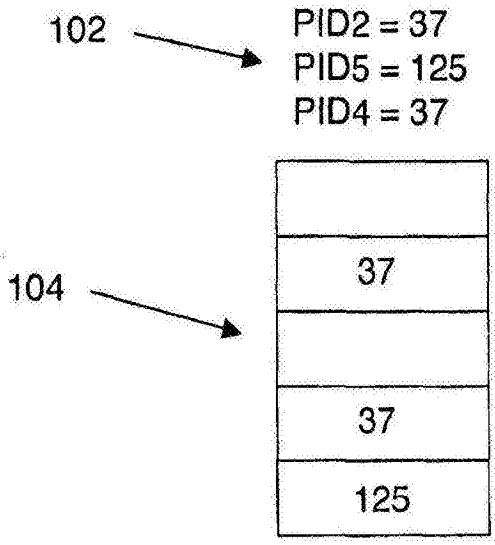

102
PID2 = 37
PID5 = 125
PID4 = 37
104
37
37
125

图1现有技术

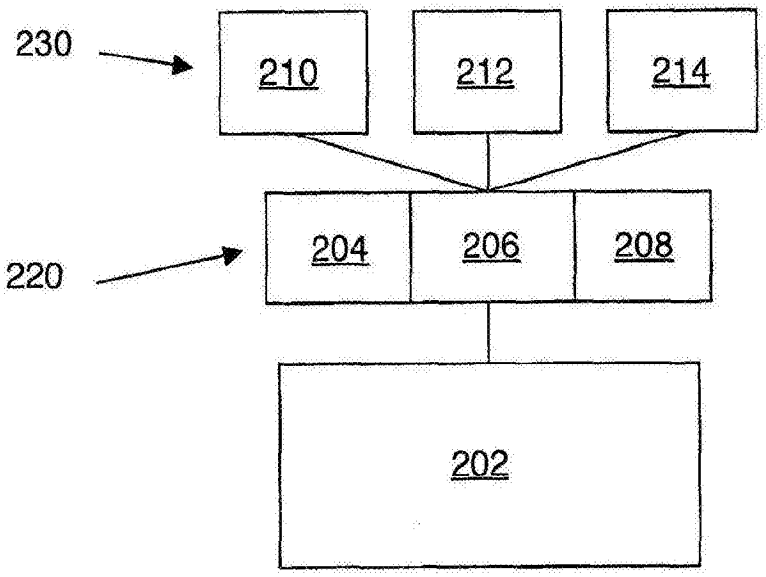

230
210
212
214
220
204
206
208
202

图2

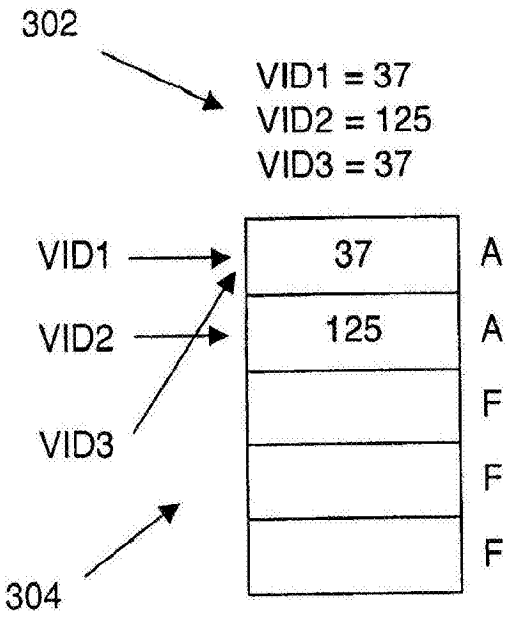


图3a

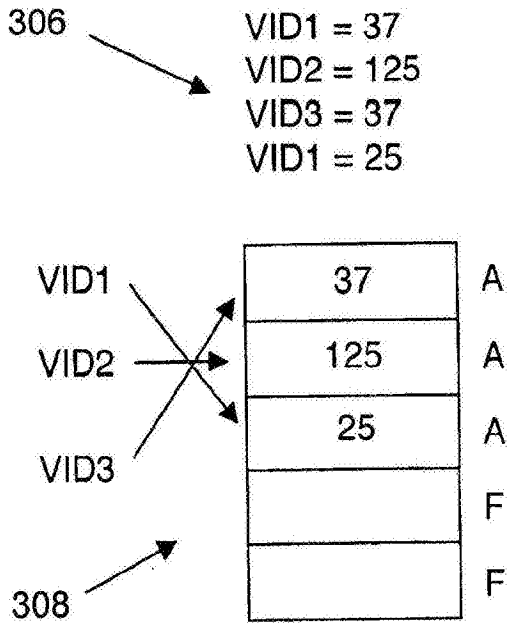


图3b

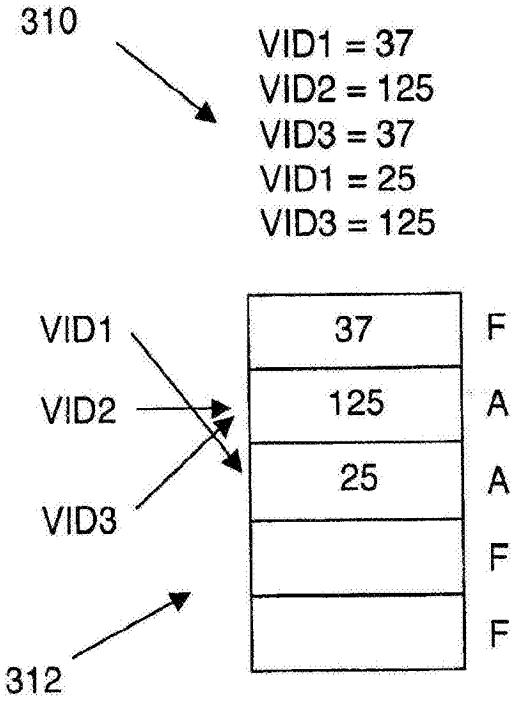


图3c

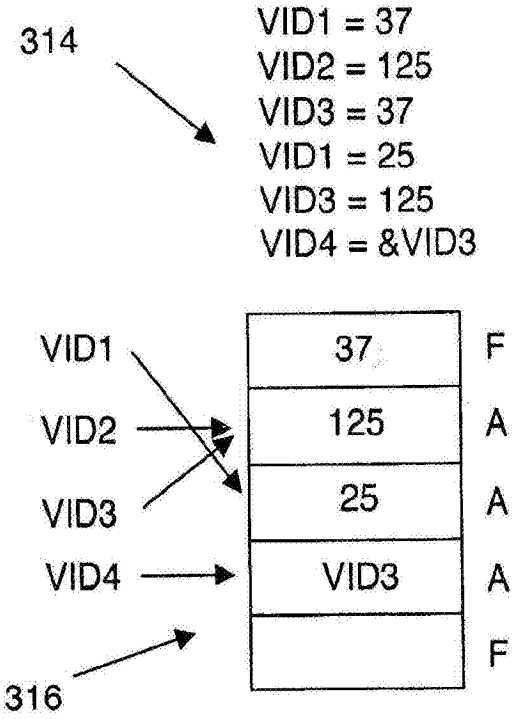


图3d

VID	PID	计数

图4

VID1	PID1	1
VID2	PID2	1
VID3	PID1	1

图5a

VID1	PID3	1
VID2	PID2	1
VID3	PID1	1

图5b

VID1	PID3	1
VID2	PID2	1
VID3	PID2	1

图5c

VID1	PID3	1
VID2	PID2	1
VID3	PID2	2
VID4	PID4	1

图5d

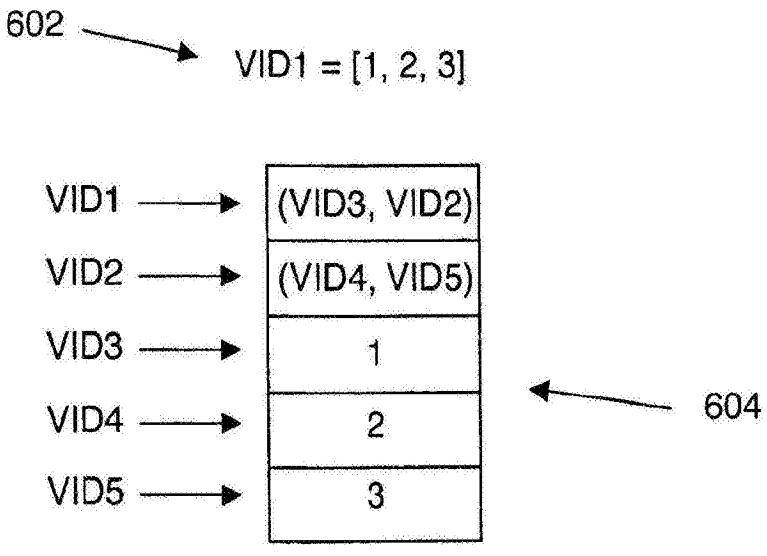


图6a

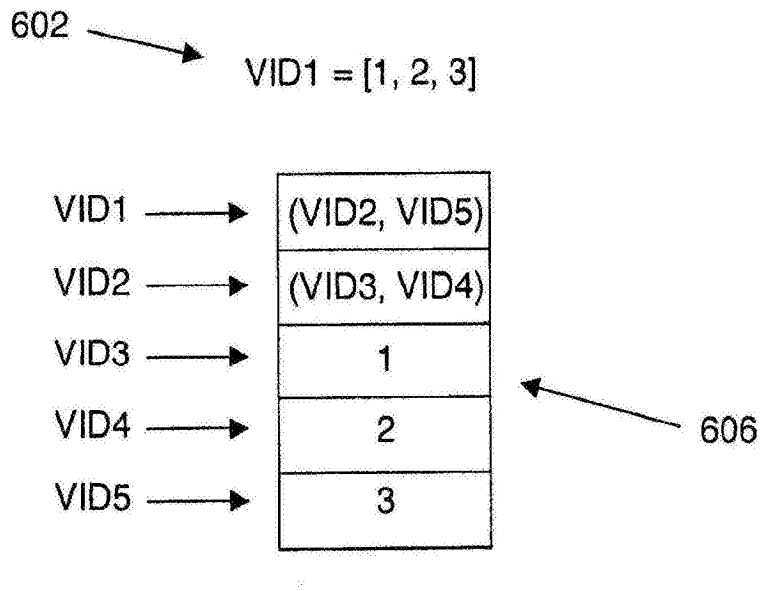

602
VID1 = [1, 2, 3]
VID1
VID2
VID3
VID4
VID5
(VID2, VID5)
(VID3, VID4)
1
2
3
606

图6b