[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 12/08 (2006.01)

[12] 发明专利说明书

专利号 ZL 200610101532.X

[45] 授权公告日 2009 年 3 月 18 日

[11] 授权公告号 CN 100470509C

[22] 申请日 2000.11.3

[21] 申请号 200610101532.X

分案原申请号 00819240.5

[30] 优先权

[32] 1999.12.29 [33] US [31] 09/474527

[73] 专利权人 英特尔公司

地址 美国加利福尼亚州

[72] 发明人 D·T·马尔

[74] 专利代理机构 中国专利代理(香港)有限公司

代理人 程天正 王忠忠

[56] 参考文献

US5404483A 1995.4.4

US5887146A 1999.3.23

US5751986A 1998.5.12

US5987549A 1999.11.16

审查员 郑宗玉

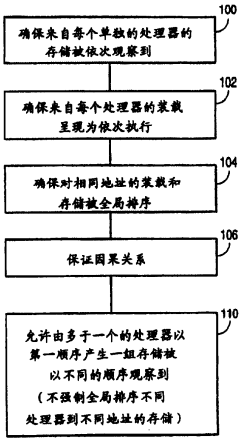
权利要求书 3 页 说明书 12 页 附图 8 页

[54] 发明名称

多重处理环境中基于因果性的存储器访问排序的方法和设备

[57] 摘要

多重处理环境中基于因果性的排序。 所公开的实施方案包括多个处理器和耦合到该多个处理器的仲裁逻辑。 处理器和仲裁逻辑保持处理器一致性，但允许由任意两个或多个处理器以第一顺序产生的存储被其它处理器中的至少一个其它处理器以不同的顺序观察。



1. 一种方法，包括:

从多个总线代理接收以第一顺序产生的多个存储内容;

透明地监视所述多个总线代理相对于所述多个存储内容的因果关系;

在所述多个总线代理与所述多个存储内容之间存在因果关系的情况下，允许多个存储内容被所述多个总线代理中的至少一个其它总线代理观察，所述观察是与存储内容的第一顺序相反的；以及

维持一个处理器一致性存储器排序模型。

2. 权利要求 1 的方法，还包括:

在所述多个总线代理与所述多个存储内容之间存在因果关系的情况下，确保来自该多个总线代理中的任意一个总线代理的存储内容被所有的所述多个总线代理以与所述存储内容的第一顺序一致地进行观察。

3. 权利要求 2 的方法，其中允许包括:

通过允许多个存储内容以与该存储内容的第一顺序相反地被观察，从而确定相对于所观察的存储内容的因果性是否被违反;

如果重新排序违反相对于所观察的存储内容的因果性，则阻止多个存储内容中的任一个存储内容的重新排序；以及

对不违反相对于所观察的存储内容的因果性的多个存储内容的一个子组进行排序，以便由所述多个总线代理中的至少一个总线代理与该存储内容的第一顺序相反地进行观察。

4. 权利要求 3 的方法，其中所述对是否违反因果性的确定包括确定一个存储内容是否依赖于一个先有的非全局观察的存储内容。

5. 权利要求 1 的方法，其中一个存储内容在处理器装载了由该存储内容指示的存储器单元时被观察。

6. 权利要求 1 的方法，其中一个存储内容在处理器装载和实际上使用由该存储内容指示的存储器单元时被观察。

7. 权利要求 3 的方法，其中如果第一存储内容在第二存储内容被第二处理器执行之前先被第一处理器执行和如果第二处理器在执行第二存储内容之前装载由第一存储内容指示的存储器单元，则所述阻止包括阻止第二存储内容在第一存储内容之前被全局观察。

8. 权利要求 1 的方法，其中所述允许包括对存储内容事务重新排序，以便更有效地访问存储器。

9. 权利要求 3 的方法，其中所述阻止包括在存储内容缓存器中设置一个或多个排序比特，以便指示排序限制。

10. 一种设备，包括：

多个缓存器；

耦合到所述多个缓存器的因果关系监视逻辑电路，该因果关系监视逻辑电路监视相对于缓存器的存储内容的因果关系；

存储内容转发逻辑电路，用于如果相对于第一存储内容和多个处理器中的第二处理器不存在因果关系，则把来自从多个处理器中的第一处理器到第一存储单元的第一存储内容的数据转发到来自所述多个处理器中的第二处理器的第一存储单元的负荷。

11. 权利要求 10 的设备，其中如果只是在第一存储内容被排序在一个访问负荷之后的第一存储单元的下一个较早存储内容之前时才存在因果关系的话，则所述存储内容转发逻辑电路把来自所述多个处理器中的第一处理器的第一存储内容的数据转发给来自所述多个处理器中的第二处理器的负荷。

12. 权利要求 11 的设备，其中存储内容转发逻辑电路还把来自所述多个处理器中的第一处理器的第二存储内容的数据转发给也来自所述多个处理器的第二负荷。

13. 权利要求 12 的设备，其中当在不存在先有的因果关系时数据从所述多个处理器中的第一处理器到第一存储单元的第一存储内容被转发到来自所述多个处理器中的第二处理器的第一存储单元的负荷时，该存储内容转发逻辑电路还通知建立因果关系。

14. 一种多处理器系统的存储器排序方法，包括：

对从第一代理到第一存储单元的第一存储内容进行缓存；

相对于所述第一存储内容透明地监视负荷；

确定对于该第一存储内容和来自第二代理的第一负荷是否存在因果关系；

如果不存在因果关系，则转发来自所述第一存储内容的数据以满足第一负荷。

15. 权利要求 14 的方法，还包括：

如果只是在第一存储被排序在一个访问负荷之后的第一存储单元的下一个较早存储内容之前时才存在因果关系，则转发来自所述第一存储内容的数据以满足第一负荷。

16. 权利要求 14 的方法，还包括:

把来自多个处理器中的第一处理器的第二存储内容的数据转发给也来自所述多个处理器的第二负荷。

17. 权利要求 15 的方法，还包括:

当不存在先有的因果关系时在数据从第一存储内容转发到多个处理器的第一处理器的第一存储单元到来自所述多个处理器的第二处理器的第一存储单元的负荷时，则建立因果关系。

多重处理环境中基于因果性的存储器访问排序的方法和设备

本申请是申请号为 00819240.5 的专利申请的分案申请。

技术领域

本公开内容涉及处理系统领域，更具体地，本公开内容涉及多重处理系统的存储器排序技术。

背景技术

改善计算机或其它处理系统的性能通常改善了整体吞吐量和/或提供了更好的用户体验。一种改善系统中处理的指令总量的技术是增加系统中处理器的数量。然而，实现多重处理（MP）系统通常所需要的不仅仅是并行互连处理器而已。例如，需要把任务或程序进行划分以使它们能够跨越并行的处理资源执行。

MP 系统中另一个主要困难是保持存储器一致性（也称为相关性）。存储器一致性是存储器保持足够的更新以向请求处理器或其它设备提供存储器内容的当前拷贝的一般要求。使用为了获得比从通常其它（例如，外部）存储器电路所能得到的更有效地访问而存储数据的内部高速缓冲存储器和其它数据结构使得保持存储器一致性变得复杂了。

一个系统可以使用硬件或硬件和软件技术的结合来保持存储器一致性。硬件提供特定的存储器排序保证，保证该硬件将在系统层次中某些选中的点上保持程序存储器访问的顺序本性（至少保持在一些选定的程度上）。在一些系统中可以用软件通过在期望的时间强制附加排序限制来补充硬件提供的存储器排序。所实现的存储器排序方案是涉及硬件复杂性、软件复杂性以及高速缓存和缓冲数据的期望能力之间的折衷的设计选择。

一种现有技术“处理器一致性”代表了在弱有序存储器一致性模型和非常严格的一致性模型之间的折衷。处理器一致性模型是已知的现有技术中的一种模型，它允许有限的重排。一种实现方式被用在一些现有的当前处理器中（例如见美国专利 5，420，991）。图 1a 中显示了现有技术中的处理器一致性存储器模型系统的一种实施方案的存储器排序约束。

根据图 1 的框 100，该现有技术系统确保来自系统中每个单独的处

理器的存储能够依次被所有其它处理器观察到。换句话说，不对来自特定处理器的单个存储相对于彼此重新排序。如框 102 所示，该系统确保来自每个处理器的装载呈现为依次执行。在一些系统中可以进行优化；然而，装载数据表现为被返回到计算-执行单元以便避免更改系统装载和存储之间的排序关系。另一方面，如果正在返回的装载数据没有被非全局观察的存储更改过，装载数据被返回的顺序可以是变化的，而且数据仍然呈现为被依次返回。

另外，如框 104 所示，该系统确保对到相同地址的装载和存储全局排序。因而，系统中的所有代理以相同的顺序观察对相同地址的装载和存储。将作为也包括这些约束的本发明的一些实施方案更详细地论述框 100-104 的约束的结果（见图 4a-4b）。

最后，如框 105 所示，除了每个处理器在观察来自其它处理器的存储之前观察到它自己的存储之外，由不同处理器到不同地址的存储都被全局排序。下面这个现有技术约束还与本发明相对比（见暗示这种现有技术约束的图 4c-4e）。一些系统（例如，基于 Santa Clara 的 Intel 公司的 ProfusionChipset 的系统）可以要求基本的硬件以确保对由不同处理器到不同存储器位置的不同存储的相当有效的有序的全局观察。

此外，存储器排序费用继续随着实现传统存储器排序模型的系统扩大到满足附加的处理问题而大幅度增加。因此，存在能在维持预先确定的存储器排序协议例如处理器一致性的同时提高效率的存储器排序技术的连续需要。

发明内容

一种公开的实施方案包括多个处理器和耦合到该多个处理器的仲裁逻辑。这些处理器和仲裁逻辑保持处理器一致性，还允许由任意两个或多个处理器以第一顺序产生的存储被其它处理器中的至少一个其它处理器以不同的顺序观察到。

附图说明

通过实例和附图中的图来说明本发明，但本发明并不局限于附图中的图。

图 1 描绘了存储器排序技术的一种实施方案。

图 2a 描绘了能够根据所公开的存储器排序技术操作的系统的一种

实施方案。

图 2b 描绘了可以由图 2a 中的系统利用的存储转发技术的一种实施方案。

图 3a 是一个流程图，描绘在存储之间存在因果关系时依照一种实施方案的处理器 A、处理器 B 和仲裁逻辑的操作。

图 3b 是一个流程图，描绘在存储之间不存在因果关系时依照一种实施方案的处理器 A、处理器 B 和仲裁逻辑的操作。

图 4a-4h 描绘了使用所公开的存储器排序技术的系统的一种实施方案中发生的示例性存储器访问序列。

图 5 描绘使用基于开关的多重处理体系结构的一种实施方案。

图 6 描绘了使用了分级的基于分组（cluster-based）的体系结构的一种实施方案。

具体实施方式

下列描述提供了多重处理环境中基于因果性的存储器排序。在下面的描述中，阐明了多个特定的细节以提供对本发明更透彻的理解，这些细节例如系统布置和层次、总线代理的类型和逻辑划分/集成选择。然而，本领域的技术人员将会认识到没有这些特定细节也可以实践本发明。在其它实例中，没有详细显示出控制结构和门级电路，以免模糊本发明。本领域的普通技术人员用所包括的描述将能够在没有不适当的实验的情况下实现必要的逻辑电路。

所公开的存储器排序技术可以有利地改善一些系统中的整体处理吞吐量。可以通过以一种允许系统更有效地对装载和/或存储访问排序的方式放松传统的存储器排序规则来实现改进的吞吐量。可以通过或者积极地重排存储器访问或者通过允许存储器访问以放松的排序规则所允许的一种新的顺序发生来改善效率。可以实现这样的存储器排序规则以保持存储器一致性，由此允许与假定提供了传统的处理器一致性的软件向后兼容。“处理器”可以是总线代理或处理数据和/或指令并因此需要访问存储器的任意类型的设备。

图 1b 描绘了所公开的存储器排序技术的一种实施方案。在剩下的图中包括了不同的细节和实施方案，而且所附描述进一步解释了图 1b 中所提到的存储器排序技术。图 1b 的框 100-104 相对于图 1a 出现并对其进行了论述。

目前公开的技术偏离了图 1a 的框 105 中所详细说明的限制性排序约束。如图 1b 的框 106 所示，因果关系得以维持。维持因果关系需要保持充分连续的访问以在一个访问影响或可能影响另一个访问的情况下获得正确的结果。换句话说，如果由第二处理器装载由来自第一处理器的第一存储存储的值，这两个存储器访问的次序对于来自第二处理器的随后的存储就很重要。在装载和第一存储之后应该对由装载处理器的随后存储进行排序，因为第一存储可能会影响第二存储所存储的值。

当来自第二处理器的第二存储直接依赖于由来自第一处理器的第一存储产生的值时会发生由不同的处理器执行的两个存储之间真实的因果连接。因为推测执行和积极的预取技术，其中，可能难以精确地确定两个存储何时被真实地因果相连。因此，为简便起见，当第二处理器只装载由第一处理器在第一个存储中存储的值时可以假定因果性。

因而，当代理装载特定存储器位置的全部或部分内容时对该存储器位置的观察就发生了。当数据已经通过系统充分地传播，如果它们装载受到影响的存储器位置所有潜在的观察者都将观察到新的值时，就实现了对新存储的值的“全局观察”。换句话说，如果所有的代理在存储的全局观察之后执行装载操作，则它们都将看到新的值。如果潜在的观察者的一个子集执行装载当它们将看到新的值时，局部观察就发生了。

框 110 进一步依照目前公开的技术指出对传统存储器排序规则的放宽。框 110 声明系统对由多于一个的处理器以第一顺序产生的一组存储以便它们以不同的顺序被观察到。换句话说，在一定环境下，相对于来自另一个处理器的存储来自第一处理器的存储可进行重新排序。否定地声明这一点，该系统不强制全局排序由不同处理器到不同地址的存储。在一些实施方案中，不同的处理器可以不同的顺序观察到这样的存储的集合。值得注意地，因为一些多重处理系统中的多个处理器可以同时产生存储，所以该组存储所处的“第一顺序”可以包括同时发生的存储。

然而，根据一种实施方案框 110 中所允许的重排受几种条件支配。在这个实施方案中，框 110 中执行的重排受框 100、102、104 和 106

中的约束支配。即，只要来自每个单独处理器的存储保持有序、来自每个处理器的装载表现为按序执行、到相同地址的装载和存储被进行了全局排序并维持了因果性，就可以更改内部处理器存储排序。其它实施方案可以只采用实现具有基于因果性的存储器排序的多重处理系统中这些限制的子集。

图 2a 描绘了实现所公开的基于因果性的存储器排序技术的系统的一种实施方案。图 2a 的系统有第一多个处理器，包括处理器 A 205 和处理器 B 210，它们与分组 1 仲裁逻辑 230、因果监控逻辑 232、访问优化逻辑 234 和缓冲逻辑 236 相耦合。类似地，该系统还包括处理器 C 215 和处理器 D 220，它们与分组 2 仲裁逻辑 240、因果监控逻辑 242、访问优化逻辑 244 和缓冲逻辑 246 相耦合。

分组仲裁逻辑 230 和 240 与中央仲裁逻辑 250、因果监控逻辑 260、访问优化逻辑 270 和缓冲逻辑 280 相耦合。这些组仲裁逻辑、因果监控逻辑、访问优化逻辑和缓冲逻辑中的每一组都可以同样地操作。在一种实施方案中，不同的逻辑元件合作以实现图 1b 中所示的存储器排序协议。尤其，访问优化逻辑模块可以改善由缓冲逻辑缓冲的装载和/或存储的完成顺序的效率，除非因果监控逻辑指示这样的重排是有问题的。中央仲裁逻辑 250 可以协调这些活动，执行它自己的优化，并最终把存储器访问分派到存储器（未显示）或其它仲裁逻辑（未显示）。

将就分组 1 讨论这些不同框的细节；然而，可以将同样的实现方式用于其它同样标记的框。就分组 1 而论，缓冲逻辑 236 缓冲来自每个代理的装载和存储。仲裁逻辑 230 在访问优化逻辑 234 的帮助下以一种有效的方式仲裁并调度来自分组 1 中所有代理的访问。仲裁逻辑 230 还为系统层次中的下一级（例如，来自中央仲裁逻辑 250）上的资源仲裁。

访问优化逻辑 234 执行诸如访问重排、写结合（WC）和数据转发（FWD）等操作以改善由仲裁逻辑产生的访问的整体效率。然而，可以通过关于图 1b 论述的约束来限制这样的优化。因此，访问优化逻辑与因果监控逻辑 234 合作以确定容许的优化程度。例如，按照图 1b 中的框 100，相对于来自每个代理的其它存储依次调度来自该代理的存储。

可以由访问优化逻辑 234 执行的一种优化是存储转发。一种实施

方案执行图 2b 的流程图里所详细描述的存储转发。通常，存储转发（也称为数据转发）涉及到核对接收到的对于未决（缓冲）的存储的装载以检测允许带有待转发给装载存储的数据缓冲的转发条件。当装载从一个存储器位置请求数据，对该存储器位置有一个缓冲在缓冲逻辑 236 中的从相同代理到相同存储器位置的先前的存储时，转发条件就出现了。这种情况下，如图 2b 的框 281 和 282 所示，可以通过转发被缓冲的数据到请求代理来满足装载。

和框 284 所检测的一样，如果装载从一个存储器位置请求数据，对该存储器位置有来自不同代理到相同存储器位置的存储并且（如框 288 中所检测的）在这两个代理的存储之间没有因果关系，就可以像框 290 中所指示的那样把存储数据转发到该装载。这种情况下，就像框 292 中所示那样在两个代理之间建立因果关系。因此，在这个实施方案中在提供了转发数据的存储之前的存储将先于来自接收到新数据的代理的随后存储。

如果在框 288 中检测到了因果关系，就如框 294 所示检查装载和存储的顺序。当装载从一个存储器位置请求数据，对该存储器位置存在来自不同代理的（到相同存储器位置的）存储，并且在两个代理的存储之间存在因果关系时，这就会发生。如框 298 所示，仅在该存储被排在装载之后的下一个较早的存储之前时才可以把存储数据转发到该装载。否则像框 296 中所示那样不转发数据。因而，可以比传统系统中更积极地执行存储转发，并且可以方便地产生更有效的整体存储器访问。

另外，可以比传统处理器一致性多重处理系统中更积极地完成普通的装载和存储重排，因为由图 1b 的基于因果性的存储器排序系统所利用的约束要比通常所用的少些麻烦。因而，在一些情况下可以更早地对更关键的存储器访问排序，和/或改进整体次序。例如，可以组合或重新调整写周期以优化特定的脉冲顺序，交叉方案或为所涉及的系统特有的其它与顺序有关的约束。

分组仲裁逻辑 230 和 240、中央仲裁逻辑 250 以及因果监控逻辑、访问优化逻辑和各自的缓冲逻辑可以驻留在一个单个的集成电路、部件、模块或类似单元中，如它们包含在仲裁块 200 中所示。或者，可以将这些逻辑块分发到不同的部件中。一种有利的实施方案是包括多

个处理器（例如，A、B 等）和该组处理器的分组仲裁逻辑的集成（例如，单个集成电路）多重处理系统。另外，虽然把这些逻辑单元分到两个不同的模块中有助于理解目前所公开的技术，但并不需要对由这里讨论的这些和其它模块实现的不同的逻辑功能也进行这样清晰的分离，而且这样的分离实际上在很多实现方式中也不能存在。

图 3a 描绘了在存储之间存在因果关系的一种情况下由图 2a 的系统的一种实施方案所执行的操作。如框 310 所示，处理器 A 存储值到存储器位置 a 和 b。根据框 100（图 1），保持来自处理器 A 的所有存储相互有序。另一方面，可以用已知或另外的可用方式对于一些存储对装载进行重新排序。

另外，根据目前所公开的技术，可以在仲裁块 200 一级上执行存储优化。如框 322 所示，处理器 B 存储一个值在存储器位置 x 中。因为由处理器 B 执行的框 322 中的操作与由处理器 A 产生的对位置 a 和 b 的存储没有因果连接，仲裁块 200 可以按照框 315 所示把处理器 A 和 B 产生的存储重新排序成一种方便的顺序。例如，可以组合或流水一些或全部写操作并且/或者按照连续顺序排序写操作。

如框 325 所示，处理器 B 可以阻止或中断这样的重排。在框 325 中，处理器 B 观察（例如，通过装载）存储器位置 b。这个观察由仲裁块 200 识别，它可以连续检查对所缓冲的存储的地址的访问。如框 330 中所示，处理器 B 对存储器位置 b 的观察导致仲裁块 200 在观察处理器 A 到 b 的存储之后产生的存储进行排序以便在对位置 b 的存储之后实际完成这样的存储。因而，仲裁块 200 不能在处理器 A 到位置 b 的存储之前对框 327 中由处理器 B 产生的对存储器位置 y 的存储进行重排。实施这个收紧的存储排序，如框 335 所示，直到实现了对 b 的全局观察。其后，可以如框 340 所示恢复存储重排。

因而，在图 3a 的实例中，公开的存储器排序技术允许在由不同处理器产生的存储之间不存在因果连接的情况下具有改进的效率。换句话说，因为由处理器 A 到存储器位置 a 的存储和由处理器 B 到存储器位置 x 的存储之间没有因果关系，所以存储可以更有效的方式进行重排。另一方面，因为处理器 B 到位置 y 的存储与处理器 A 到位置 b 的存储有因果关系，因而不能对它们进行完全地重排。不违反因果关系的合法的或允许的顺序由参考编号 345 所示。

图 3a 的实施方案允许的排序

a, b, x, y

a, x, b, y

x, a, b, y

因为大量处理器可能有更多因果不相关的存储，通过执行跨越多于两个处理器的重排可以期待附加的效率。

图 3b 描绘了在存储间不存在因果关系的情况下由图 2a 的系统的一种实施方案执行的操作。如框 350 所示，处理器 A 执行对位置 a 和 b 的存储。同样，处理器 B 执行对位置 x 和 y 的存储，如框 355 所示。因为两个处理器都不装载由另一处理器修改过的另一值，在不同的存储间也没有因果关系。因而，如框 360 所示，仲裁块 200 没有检测到因果关系。因此，可以用任意方便的顺序来执行存储，只要观察到其它系统约束。

图 3b 的实施方案允许的排序

a, b, x, y

x, y, a, b

x, a, y, b

a, x, b, y

x, a, b, y

a, x, y, b

图 4a-4h 描绘在多重处理系统中使用所公开的存储器排序技术的不同后果。在一种实施方案中，实施了图 1b 中所示的约束。

1. 依次观察到来自每个处理器的存储（框 100）。

再次，当处理器或代理（或相同代理或不同的代理）装载一个存储器位置并检索由一个存储修改的该存储器位置的新值时，这个存储就被“观察”到。当来自各个处理器的存储被依次观察到时，没有代理能够观察到来自那个处理器的连续的后来存储的新值和来自那个处理器的连续的先前存储的旧值。

图 4a 描绘代理 A 执行两个存储。初始条件是所有存储器位置存储 0（在图 4a-4h 的所有图中）。首先，由代理 A 在存储器位置 a 中存储值 1。然后，代理 A 还存储值 1 到存储器位置 b。代理 B、C 和 D 以所有合法的情况（即，依照所实现的存储器排序方案的允许的次序）观

察这两个存储。代理 B 观察到 a 的新值，而不是 b 的旧值，这是可以接受的，因为对 a 的存储在对 b 的存储之前发生。代理 C 观察到 a 和 b 的旧值。代理 D 观察到 a 和 b 的新值。没有代理能够观察到 b 的新值和 a 的旧值，因为那将违反框 100 的约束。

2. 来自每个处理器的装载呈现为依次执行（框 102）

在这个实施方案中，装载被限制以呈现为依次执行。通过呈现为依次执行，意味着如果非全局观察到的存储可能影响装载数据，装载数据就被依次返回到请求计算单元（虽然数据传输信号可能会由于总线协议、传输技术等而被重排）。如果非全局观察到的存储没有更改正在返回的装载数据，那么可以对它的返回顺序进行重排，而且数据仍然呈现为依次被返回，因为重排不影响计算单元。如果对由非全局观察到的存储更改过的存储器位置允许进行装载重排，就会影响装载和存储的相对次序，并且可能观察到存储不按次序执行。在图 4a 中，代理 B、C 和 D 都依次执行它们的装载，而且返回给代理的是正确的值。作为一个实例，如果允许其中一个代理不按次序退回装载，那么可能观察到 b 在 a 之前改变。

3. 对到相同地址的装载和存储进行全局排序（框 104）

对到相同地址的装载和存储被全局排序，使得如果两个代理都向相同地址存储值，所有观察代理观察到其中一个存储在另一个之前发生。在图 4B 中，代理 A 和代理 B 都向位置 a 存储，代理 A 存储 1 而代理 B 存储 2。代理 C、D 和 E 以所有可能的顺序观察这两个存储。代理 D 观察把 a 的值改为 1 的代理 A 的存储，然后观察把 a 的值从 1 改成 2 的代理 B 的存储。因此，在这个实施方案中没有其它代理能够观察到相反的顺序。因此，代理 C 能够观察到 a 是 0 然后是 1，代理 E 能够观察到 a 是 0 然后是 2。根据这个实施方案没有代理能够观察到 a 是 2 然后 a 是 1。

另一方面，在图 4C 中，代理 D 观察到把位置 a 的值从 0 改为 2 的代理 B 的存储，随后观察到相同位置的代理 A 的存储，它把位置 a 的值从 2 改为 1。如果另一代理看到了相反的顺序，就违反了框 104 中的约束。因此，代理 C 首先观察到 a 是 0，然后观察到 a 是 1（当 a 是 2 时没有观察）。代理 E 观察到 a 是 0，然后观察到 a 是 2。

在现有技术的“处理器一致性”系统（例如，图 1a）中，由不同

处理器到不同地址的存储通常进行全局排序（框 105）。在这样的现有技术的系统中，对到不同地址的存储全局排序，除了每个代理能够在观察到来自其它处理器的存储之前观察到它自己的存储。在图 4D 中，代理 A 和 B 分别在存储器位置 a 和 b 中存储值 1。代理 D 在 b 是 1 之前观察到 a 是 1。因此，在这个实施方案中没有其它代理在 a 是 1 之前观察到 b 是 1。代理 C 观察到 a 和 b 都是 0，代理 E 观察到 a 和 b 都是 1。

在图 4E 中，代理 A 和 B 再次各自分别在存储器位置 a 和 b 中存储值 1。代理 D 在观察 a 是 1 之前观察到 b 是 1。因此，在这个实施方案中没有其它代理在 b 是 1 之前观察到 a 是 1。代理 C 观察到 b 和 a 都是 0，代理 E 观察到 b 和 a 是 1。

图 4F 中显示了这种例外，那里代理正在观察它们自己的存储。这种情况下，并没有利用需要对到不同地址的存储进行全局排序的约束。因而，如图 4F 所示，代理 A 在代理 B 到位置 b 的存储之前观察到它自己到位置 a 的存储。同样，代理 B 在代理 A 到位置 a 的存储之前观察到它自己到位置 b 的存储。因而，除了自我观察例外以外，该现有技术中的策略对于来自不同处理器的不同存储是非常严格的。

4. 保证因果关系（框 106）

与强制对由不同处理器到不同地址的存储进行全局排序的现有技术的系统相比，一些采用了目前公开的技术的实施方案仅实施了需要用来维持因果关系的最小量的存储排序。图 4G 描绘了由对所存储的值的观察创建因果连接的情况。代理 A 存储值 1 到存储器位置 a，代理 B 观察到 a 的存储然后执行值 1 到存储器位置 b 的存储。代理 B 引入到存储器位置 a 和 b 的存储之间的因果关系，因此所有代理都被约束为在对 b 的存储之前观察到对 a 的存储。代理 C 只这样做，首先观察到对存储器位置 a 的存储。显示代理 D 已经观察到了这两个存储，但不在观察对 a 的存储之前观察到对 b 的存储。

如果处理器 B 没有观察到对存储器位置 a 的存储，该系统可能已经对到存储器位置 a 和 b 的这些存储的全局观察进行了重排。换句话说，如果处理器 B 没有询问关于位置 a 的值，处理器 C（和系统中的剩余处理器）关于是 a 或 b 首先变成 1 可能已经无关紧要了。因为由处理器 B 观察到了对 a 的存储，创建了因果连接，仲裁逻辑保持了 a 先 b 后的次序。

5. 由不同处理器对不同地址的存储不被全局排序（框 110）

图 4H 描绘了代理 A 和 C 存储到两个不同地址的情况。代理 A 存储值 1 在存储器位置 a 中，代理 C 存储值 1 在存储器位置 b 中。随后代理 B 和 D 以两种不同的顺序观察这两个存储。只要在代理 A 和 C 之间没有因果关系就允许这种次序（即，在对那个存储的全局观察之前没有观察到来自其它代理的存储）。因此，其它代理（非存储流出代理，在这个例子中是代理 B 和 D）能够以不同的顺序观察到这些存储。在传统处理器排序方案中不允许这种次序。

图 5 示出一种利用基于开关的多重处理体系结构的系统的一种实施方案。在该系统中，中央开关 500 耦合多个处理器 510，512，514，516，518，520，522 到多个存储器和包括 I/O 接口 530 和 534 的 I/O 设备和存储器系统 532 和 536。I/O 接口 530 和 534 可分别提供接口到一个或多个 I/O 设备。这种基于开关的多重处理布置的基础在现有技术中是已知的。

因为有别于先前基于开关的多重处理系统，图 5 中所示的实施方案用图 1b 中所描绘的基于因果性的存储器排序技术的实施方案实现了处理器一致存储器体系结构。因此，中央开关 500 确保来自每个单独的处理器的存储依次被所有其它处理器观察到（框 100）。只要不违反因果性，中央开关 500 还允许由多个处理器以第一顺序产生的一组存储被以不同的顺序观察。为此目的，在中央开关 500 中包括多个缓冲器 502 以在来自不同处理器的存储被提交给存储器（未显示）之前对它们进行缓冲。只要没有其它系统限制被暗示并且因果监控逻辑 504 没有检测到因果关系并因此限制可以由访问优化逻辑 506 完成的优化，访问优化逻辑 506 能够对来自多个缓冲器 502 的存储进行重排，执行存储转发，和/或进行其它优化。

图 6 描绘了一种分层的、基于分组的多重处理系统，该系统也实现所公开的存储器排序技术。图 6 的系统包括多个处理器，处理器 650、652、654、656、658 和 660。处理器 650 和 652 以及任意数量的附加处理器，形成了由控制器 672 控制的一个分组。类似地，处理器 654 和 656 形成了由控制器 674 控制的一个分组，处理器 658 和 660 形成了由控制器 676 控制的一个分组。控制器 672 和 674 被耦合到中间级控制器 680，控制器 680 又被耦合到顶级控制器 690，控制器 690

与存储器（未显示）接口。控制器 676 也被耦合到顶级控制器 690。也可以用很多其它分层布置，包括对每个分组使用不同数量的处理器，不同数量的分组和控制器的不同（或没有）划分。

在图 6 的系统中，可以在层次体系的不同级上执行因果监控和缓冲。例如，可以在控制器 672、674、676、680 和 690 中的每个中包括缓冲（BUF）和因果监控（CM）逻辑。存储随后被传递到层次体系中的更高级，这些层由因果监控逻辑所利用的任意已知的排序限制标记。结果，该系统中的不同处理器可以不同的顺序观察存储。例如，顶级控制器 690 能够比像控制器 672 这样的低级控制器执行更多的优化，因为顶级控制器 690 有更多或至少不同的存储来重新排列、组合或另外操作。

像对于其它实施方案所论述的，可以通过更有效的排列存储器访问来改善系统吞吐量。这样的优化可以包括或涉及写组合、分页、交叉、装载旁路或其它已知或可用的存储器访问优化技术。所公开的实施方案可以允许维持处理器一致性存储器排序模型，有利地提供与现有代码的向后兼容性，现有的代码假定与处理器一致性存储顺排序模型的兼容性。

因而，公开了多重处理环境中基于因果性的存储器排序。虽然已经在附图中描述并显示了一定的示范性实施方案，因为在研究这个公开内容时对本领域的普通技术人员来说可能发生多种其它修改，因而应当理解这样的实施方案仅仅是说明性的，对主要发明并无限制，而且本发明也不局限于所显示和描述的具体构造和布置。

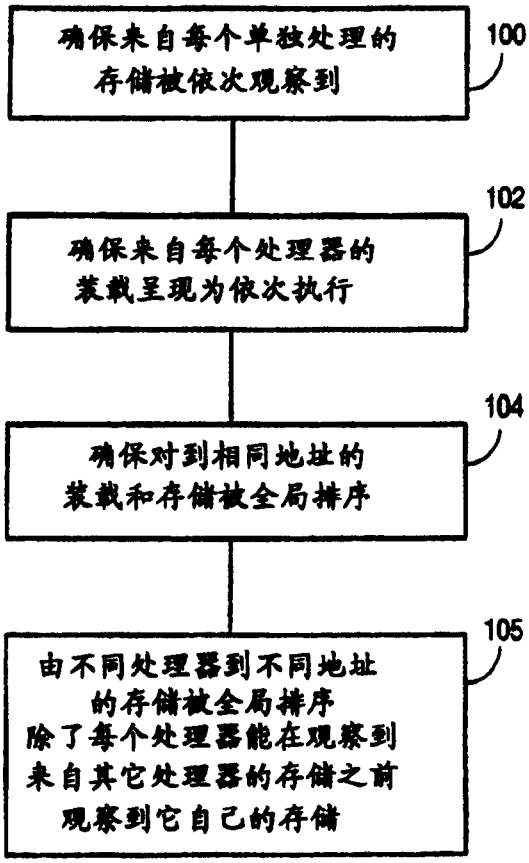

确保来自每个单独处理的
存储被依次观察到
100
确保来自每个处理器的
装载呈现为依次执行
102
确保对到相同地址的
装载和存储被全局排序
104
由不同处理器到不同地址
的存储被全局排序
除了每个处理器能在观察到
来自其它处理器的存储之前
观察到它自己的存储
105

图 1A

现有技术

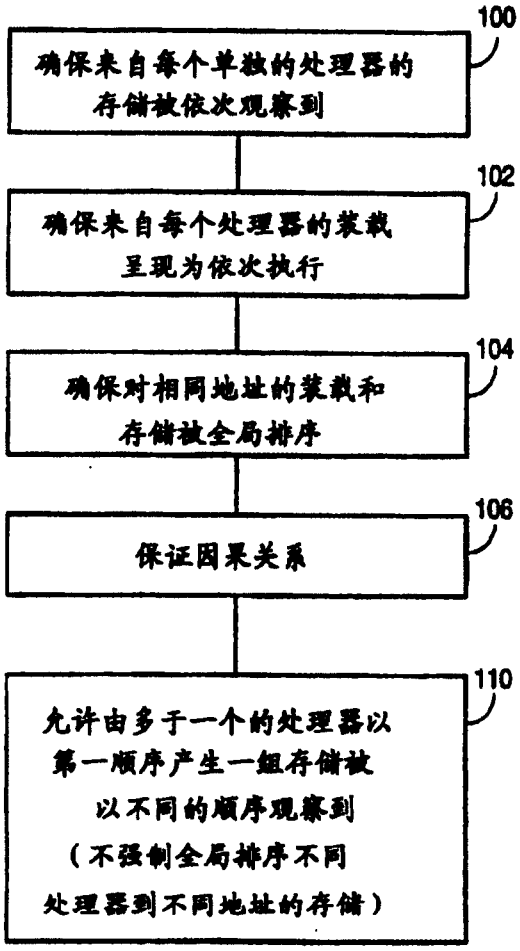

100
确保来自每个单独的处理器的
存储被依次观察到
102
确保来自每个处理器的装载
呈现为依次执行
104
确保对相同地址的装载和
存储被全局排序
106
保证因果关系
110
允许由多于一个的处理器以
第一顺序产生一组存储被
以不同的顺序观察到
（不强制全局排序不同
处理器到不同地址的存储）

图 1B

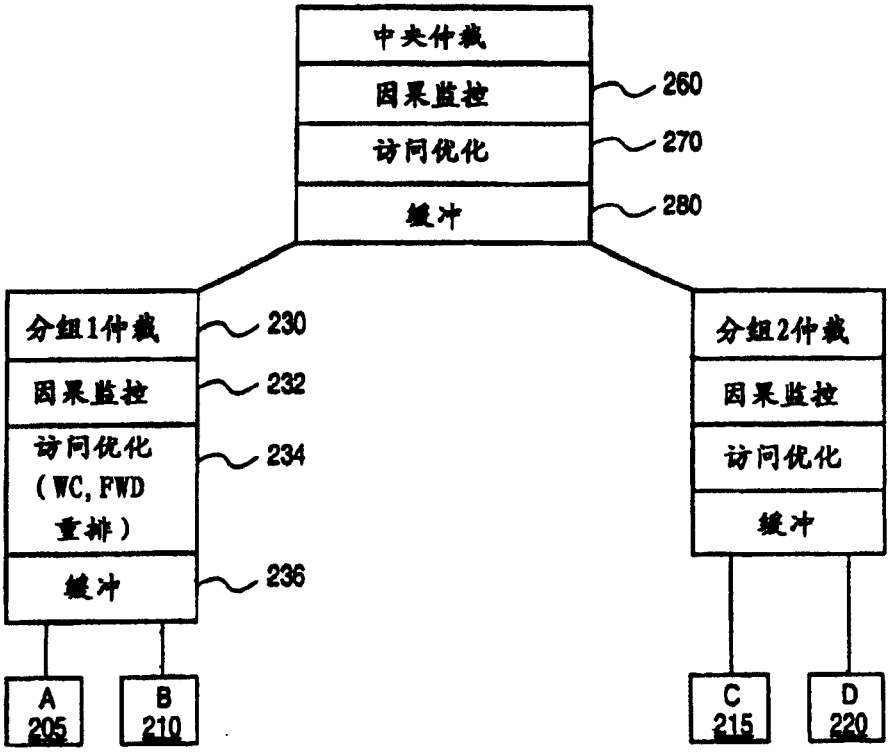

中央仲裁
因果监控
260
访问优化
270
缓冲
280
分组1仲裁
230
因果监控
232
访问优化
(WC, FWD
重排)
234
缓冲
236
A
205
B
210
分组2仲裁
因果监控
访问优化
缓冲
C
215
D
220

图 2A

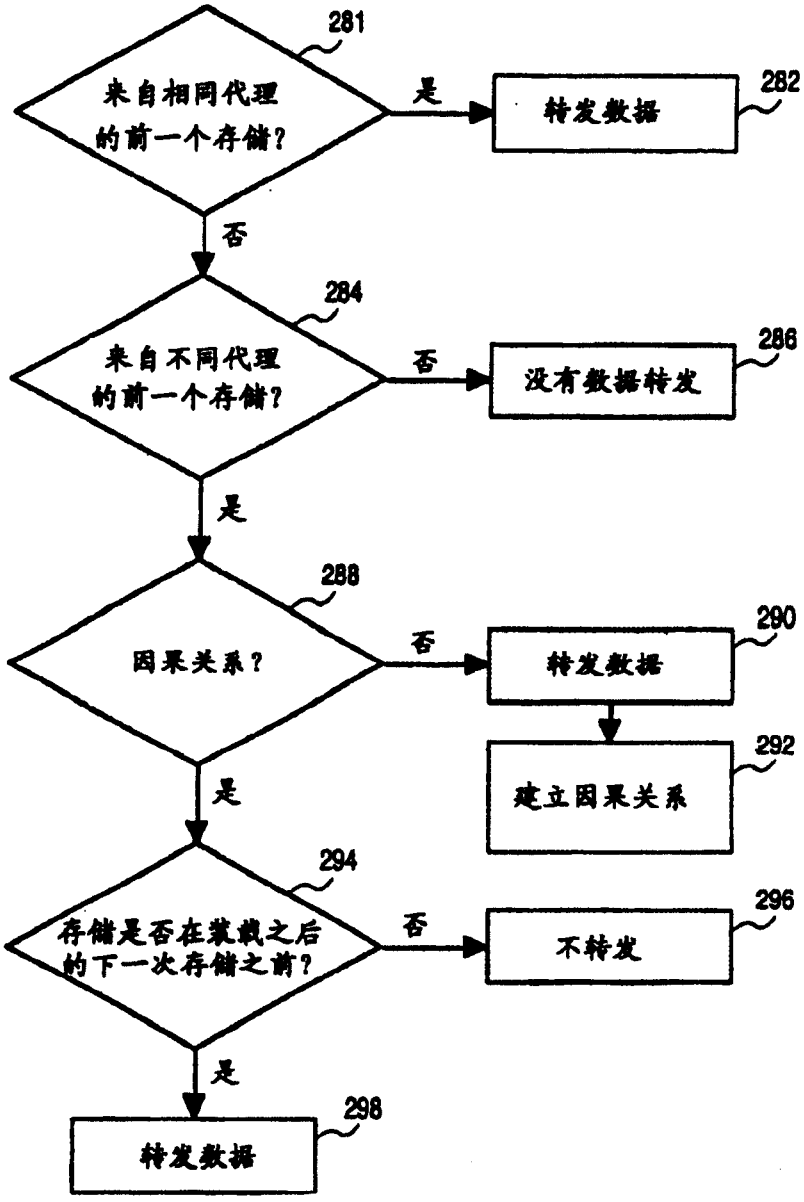

281
来自相同代理的前一个存储?
是
282
转发数据
否
284
来自不同代理的前一个存储?
否
286
没有数据转发
是
288
因果关系?
否
290
转发数据
292
建立因果关系
是
294
存储是否在装载之后的下一次存储之前?
否
296
不转发
是
298
转发数据

图 2B

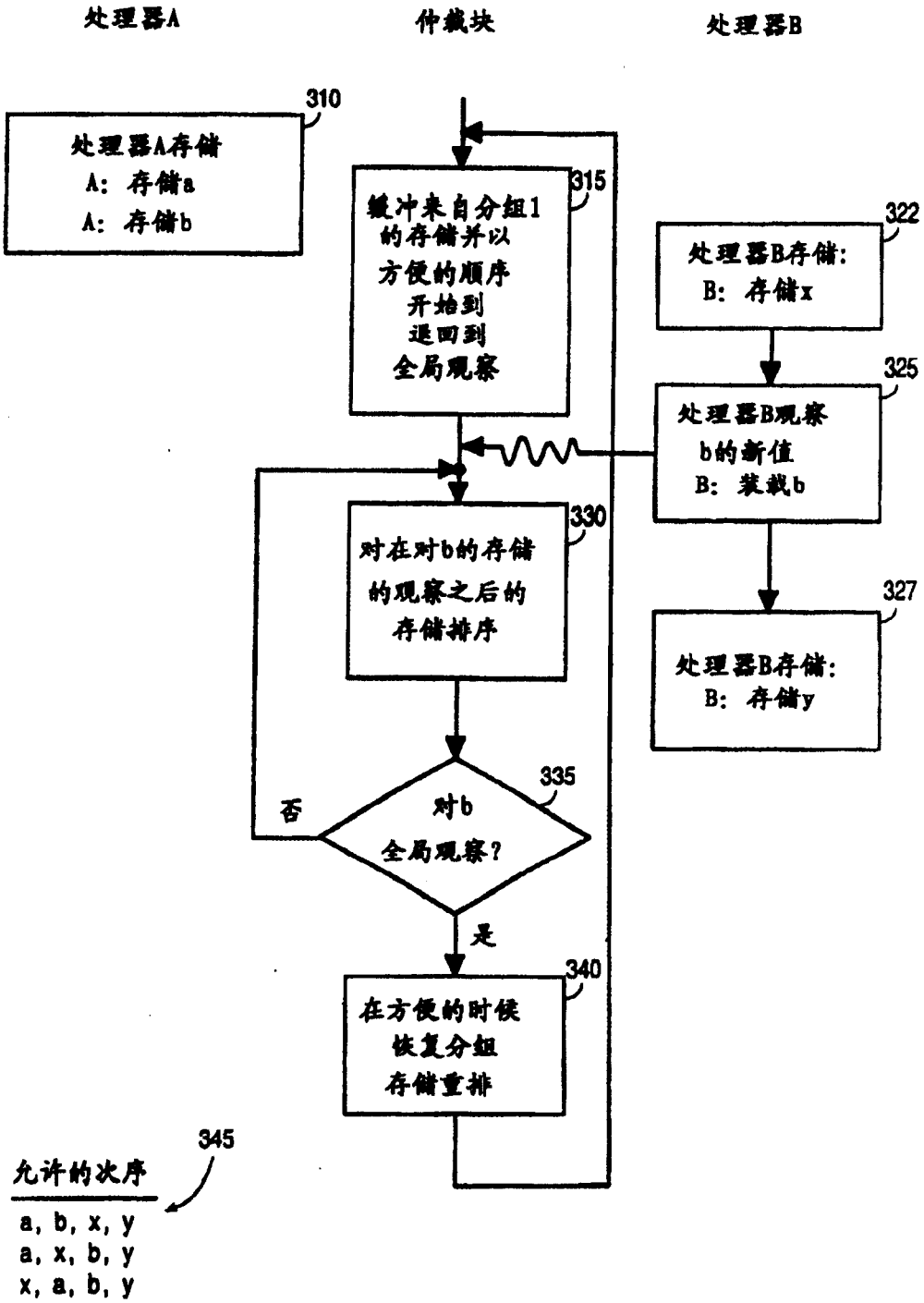

处理器A
仲裁块
处理器B
310
处理器A存储
A: 存储a
A: 存储b
315
缓冲来自分组1的存储并以方便的顺序开始到退回到全局观察
322
处理器B存储:
B: 存储x
325
处理器B观察b的新值
B: 装载b
327
处理器B存储:
B: 存储y
330
对在对b的存储的观察之后的存储排序
335
对b全局观察?
否
是
340
在方便的时候恢复分组存储重排
345
允许的次序
a, b, x, y
a, x, b, y
x, a, b, y

图 3A

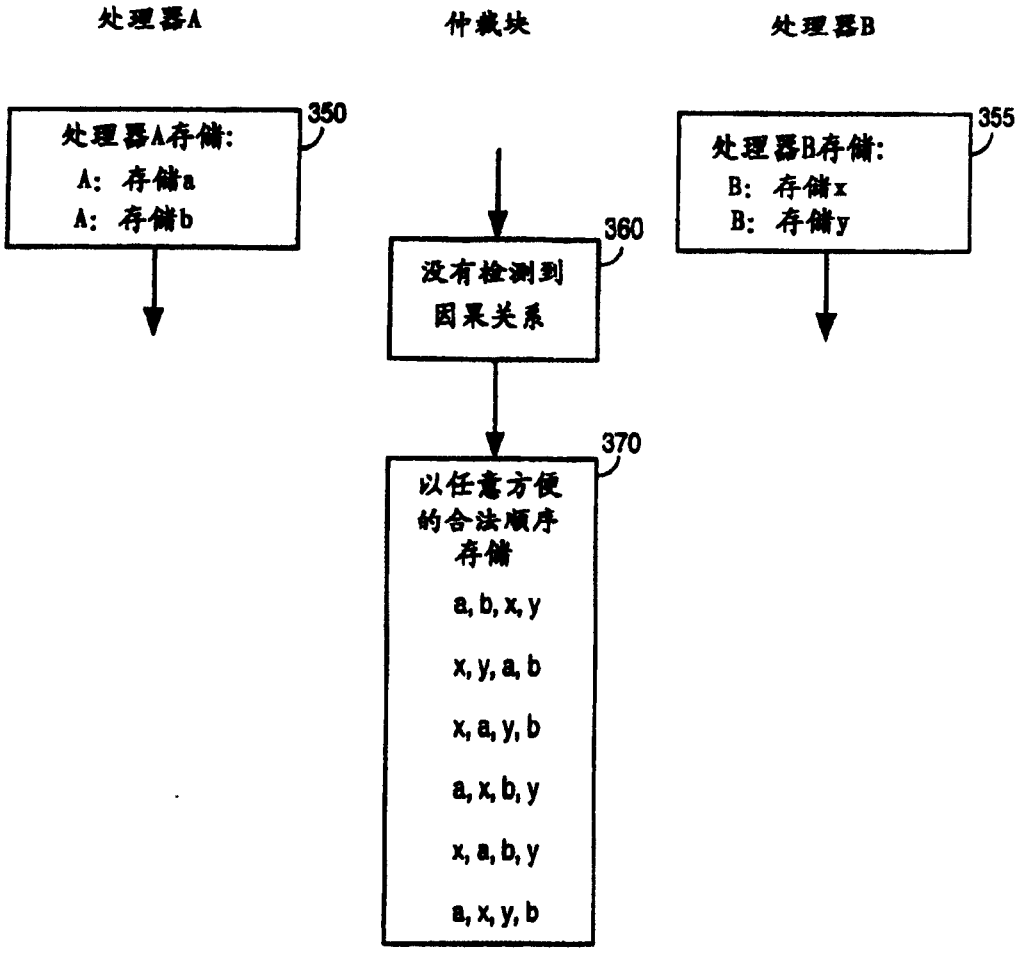

处理器A
仲裁块
处理器B
处理器A存储:
A: 存储a
A: 存储b
350
处理器B存储:
B: 存储x
B: 存储y
355
没有检测到
因果关系
360
以任意方便
的合法顺序
存储
a, b, x, y
x, y, a, b
x, a, y, b
a, x, b, y
x, a, b, y
a, x, y, b
370

图 3B

A	B	C	D
ST a+1	LD a=1	LD a=0	LD a=1
ST b+1	LD b=0	LD b=0	LD b=1

图 4A

A	B	C	D	E
ST a←1	ST a←2	LD a=0	LD a=1	LD a=0
		LD a=1	LD a=2	LD a=2

图 4B

A	B	C	D	E
ST a←1	ST a←2	LD a=0	LD a=2	LD a=0
		LD a=1	LD a=1	LD a=2

图 4C

A	B	C	D	E
ST a←1	ST b←1	LD a=0	LD a=1	LD a=1
		LD b=0	LD b=0	LD b=1

图 4D

A	B	C	D	E
ST a←1	ST b←1	LD b=0	LD b=1	LD b=1
		LD a=0	LD a=0	LD a=1

图 4E

A	B
ST a←1	ST b←1
LD a=1	LD b=1
LD b=0	LD a=0

图 4F

A	B	C	D
ST a←1	LD a=1	LD a=1	LD a=1
	ST b←1	LD b=0	LD b=1

图 4G

A	B	C	D
ST a←1	LD a=1	ST b←1	LD b=1
	LD b=0		LD a=0

图 4H

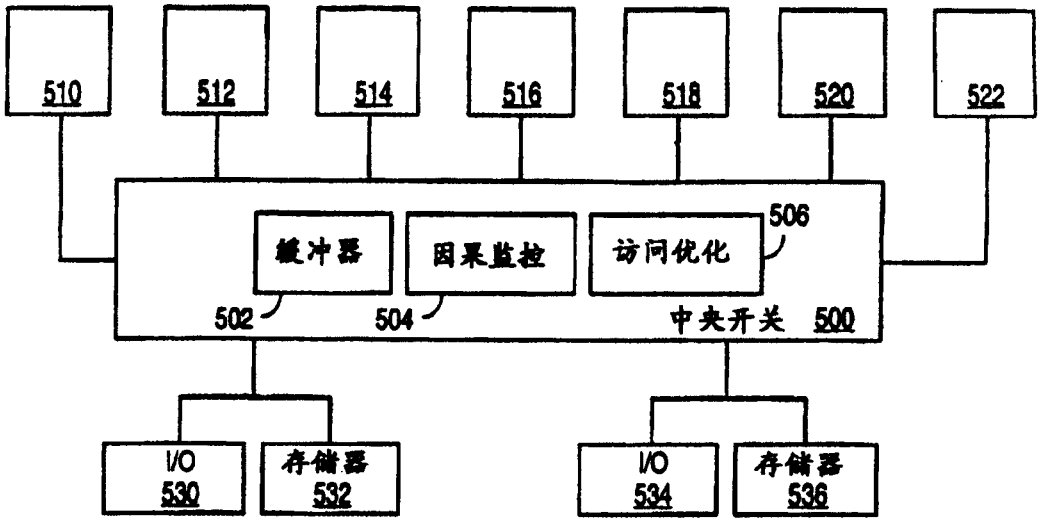

510
512
514
516
518
520
522
缓冲器
因果监控
访问优化
506
502
504
中央开关 500
I/O 530
存储器 532
I/O 534
存储器 536

图 5

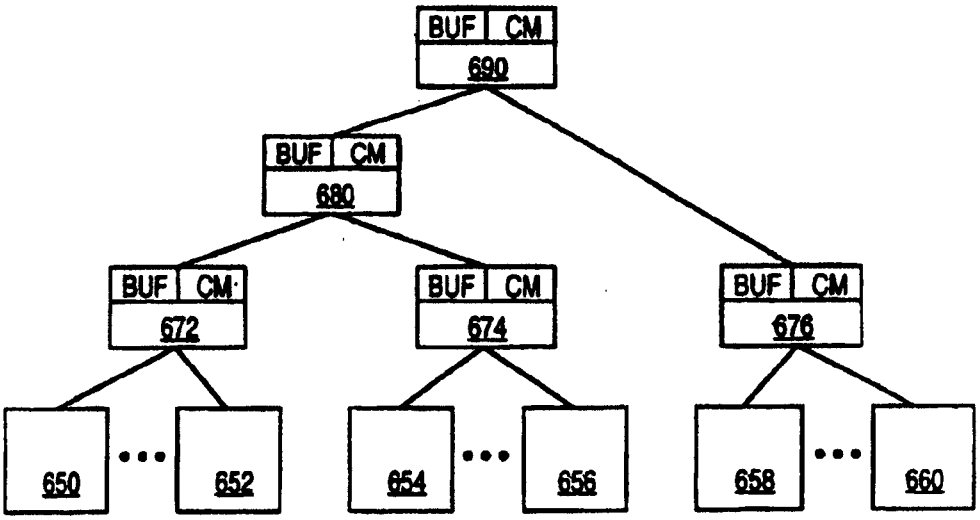

BUF CM 690
BUF CM 680
BUF CM 672
BUF CM 674
BUF CM 676
650
652
654
656
658
660

图 6