[19] 中华人民共和国国家知识产权局

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

[12] 发明专利说明书

专利号 ZL 200510077634.8

[45] 授权公告日 2008 年 7 月 23 日

[11] 授权公告号 CN 100405302C

[22] 申请日 2005.6.17
[21] 申请号 200510077634.8
[30] 优先权
[32] 2004.12.7 [33] US [31] 11/006,083
[73] 专利权人 国际商业机器公司
地址 美国纽约阿芒克
[72] 发明人 拉里·伯特·布伦纳
[56] 参考文献
US2003/0195920A1 2003.10.16
US5924097A 1999.7.13
CN1469246A 2004.1.21
US2002/0099759A1 2002.7.25
CN1202971A 1998.12.23
审查员 毛习文

[74] 专利代理机构 北京市金杜律师事务所
代理人 朱海波

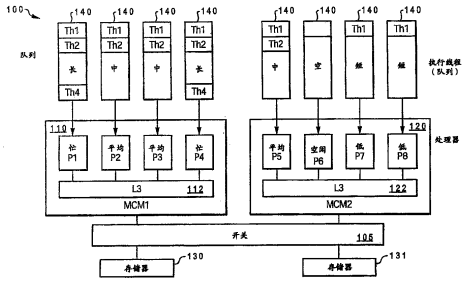
权利要求书 3 页 说明书 12 页 附图 4 页

[54] 发明名称

在多处理器数据处理系统中借入线程作为负载平衡的形式

[57] 摘要

多处理器数据处理系统(MDPS)中的一种方法和系统，该方法和系统支持第一 MCM(多片组件)中的带有空闲处理器周期的第一处理器和第二 MCM 中的第二忙处理器之间的有效负载均衡，而不会在分配给该空闲处理器周期时引起该线程的执行效率的显著降低。 提供同时支持跨 MCM 的线程窃取和借入的负载均衡算法。 允许空闲处理器从另一个存储域(即，跨 MCM)中的忙处理器中“借入”一个线程。 每次借入该线程一个调度周期。 当完成该调度周期时，释放该线程回到它的父处理器。 在该调度周期内，不改变被借入的线程的存储器分配。



1. 一种多处理器数据处理系统，包括:

具有第一处理器的第一多片组件，该第一处理器具有包含多个线程的第一处理器队列;

具有第二处理器的第二多片组件，该第二处理器具有空的第二处理器队列;

用于把所述第一多片组件连接到所述第二多片组件的装置;以及

负载均衡逻辑装置，用于估算所述第一多片组件和所述第二多片组件之间的负载均衡，并且使所述第二多片组件的所述第二处理器能够从所述第一多片组件的所述第一处理器队列中借入线程并执行一个调度周期的线程。

2. 如权利要求 1 的多处理器数据处理系统，其中所述负载均衡逻辑装置在该调度周期的末尾将该线程返回到所述第一处理器队列中。

3. 如权利要求 1 的多处理器数据处理系统，进一步包括:

与所述第一多片组件关联的第一存储部件，该部件存储与在所述第一多片组件中执行的线程相关联的存储数据;

与所述第二多片组件关联的第二存储部件，该部件存储与在所述第二多片组件中执行的线程相关联的存储数据; 以及

其中所述负载均衡逻辑装置进一步防止被借入的线程的存储对象在所述调度周期期间从所述第一存储部件移动到所述第二存储部件。

4. 如权利要求 1 的多处理器数据处理系统，其中所述负载均衡逻辑装置包括:

线程窃取算法，该算法使所述第二处理器能够从所述第一处理器的队列中或从所述第二多片组件的本地的第三存储器的队列中窃取线程; 以及

线程借入算法，当该线程窃取算法确定当前的负载不均衡小于开

始窃取该线程所需的阈值时，该算法开始一个调度周期的该线程的借入。

5. 如权利要求 4 的多处理器数据处理系统，其中该线程借入算法促使向所述第一处理器的多片组件的存储器分配存储对象。

6. 如权利要求 1 的多处理器数据处理系统，其中所述负载均衡逻辑装置包括软件算法。

7. 在第一多片组件和第二多片组件相连的多处理器数据处理系统中的一种方法，该方法包括:

分析指派给所述第一多片组件和所述第二多片组件内的多个处理器队列的每个处理器队列的线程数;

在所述第二多片组件的第二处理器忙时，确定所述第一多片组件的至少第一处理器何时空闲; 以及

通过从与所述第二处理器关联的处理器队列中借入一个线程并指派该线程在下一个调度周期内由所述第一处理器执行，进行该多处理器数据处理系统的负载均衡。

8. 如权利要求 7 的方法，其中所述确定进一步包括，当与所述第一处理器关联的处理器队列中没有可执行的线程时，把所述第一处理器标记为空闲，并且当与所述第二处理器关联的处理器队列中有多个线程时，把所述第二处理器标记为忙。

9. 如权利要求 8 的方法，进一步包括，仅当预计正被借入的线程不会在下一个调度周期内由所述第二处理器执行时，才能够借入该线程。

10. 如权利要求 7 的方法，进一步包括:

确定何时把所述第二处理器的线程完全重新指派给另一个处理器;

当确定应完全重新指派该线程时，作为响应，使另一个处理器能够窃取该线程; 以及

仅当不完全重新指派所述线程时，才允许所述借入。

11. 如权利要求 10 的方法，其中所述允许包括，确定当前的负

载不均衡低于开始窃取该线程所需的阈值。

12. 如权利要求 7 的方法，进一步包括，在下一个调度周期末尾将该线程返回到与所述第二处理器关联的处理器队列。

13. 如权利要求 7 的方法，进一步包括：

在所述下一个调度周期内，把被借入的线程的存储对象保持在与所述第二多片组件关联的第二存储器内；以及

在所述调度周期内，把存储对象分配给所述第二多片组件的所述第二存储器。

在多处理器数据处理系统中借入线程作为负载平衡的形式

技术领域

本发明一般涉及数据处理系统，特别地，涉及多处理器数据处理系统。更准确地说，本发明涉及多处理器数据处理系统的处理器之间的负载平衡。

背景技术

为了更有效地完成软件代码的执行，大部分常规数据处理系统的处理器以指令的线程的方式处理代码。就多处理器数据处理系统(MDPS)来说，利用线程以便在处理代码时能够在不同处理器之间定义工作的划分。一个处理器可以处理多个线程并且每个处理器可以同时处理不同线程。本领域技术人员熟悉线程的使用以及要在处理器上执行的指令的线程的调度。

MDPS 中的处理器彼此协作运行以完成该数据处理系统执行的各种任务。这些任务被指派给特定处理器或平均分配给这些处理器。由于各种因素，应平均分配给这些处理器的处理负载经常被不均衡地分发。事实上，在某些情况中，MDPS 中的一个处理器可能空闲（即，当前不处理任何线程），而该 MDPS 中的另一个处理器很忙（即，被指派用来处理几个线程）。

AIX 中的现有负载均衡算法允许空闲（第二）处理器从相当忙的第一处理器中“窃取”线程。当完成该窃取的线程时，改变该线程的运行队列指派(即，将该线程指派到的执行该线程的处理器队列)，从而将被窃取的线程半永久地指派给窃取处理器。然后被窃取的线程将具有强大趋势：未来由该处理器提供服务。对于用来窃取线程的常规算法/协议，该线程的指令在窃取处理器上的初始调度通常遇

到额外的超高速缓冲存储器遗漏，尽管随后的重新调度最终变成有效的。

因为该线程窃取算法引起初始调度期间的额外的高速缓冲存储器遗漏，所以常规算法引入窃取“障碍”，以防止从不过载（或不接近过载）的处理器中窃取线程。对照因过分进取的线程窃取而产生的处理器周期的低效利用，窃取障碍的使用在或许留下处于空闲状态的空闲处理器的情况下，折衷选择浪费的处理器周期。

当窃取线程时，更新的POWER™处理器型号可能有附加代价。造成附加代价的原因是，在设计该 POWER 处理器型号时利用基于多片组件(MCM)的体系结构。在 POWER 处理器设计中，MCM 是共享 L3 高速缓冲存储器和物理存储器的一个小的处理器组（如，4 个处理器）。可以把 MCM 连接到提供增强处理能力的一个更大系统中的其它 MCM。

因为用于 MCM 的处理器的共享高速缓冲存储器和存储器配置，所以更希望在一个 MCM 内窃取线程（即，同一本地 MCM 的第二处理器从第一个 MCM 的第一处理器中窃取线程），而不是从第二个非本地 MCM 的处理器中窃取线程。随着用于 AIX 5.3 中的进程的新的存储关系控制的出现，例如，执行进程可能使它的存储页面后退到该 MCM 的本地存储器中，所以特别希望把窃取限制在该 MCM 内。

另外，众所周知，允许更自由地窃取将严重影响被窃取的线程的存储位置，并引起被窃取的线程的性能的显著降低。当从另一个 MCM 中窃取线程时，因窃取线程引起的性能降级（以及窃取线程的其它负面影响）更显著。因此，尽管限制跨 MCM 的线程窃取可能导致空闲处理器上的更多的浪费周期，但是允许跨 MCM 的线程窃取导致有关线程的相当大的降级。该降级部分是因为该线程的长期远程执行和不一致的性能。因此，跨 MCM 窃取线程特别不令人满意。

某些开发人员提出称为“远程执行”的方法。在某些情况中，在扩展时段内，把在本地节点(MCM)创建的整个进程卸载到远程节点

(MCM)，并最终移回到该本地节点(MCM)。通常，稍后将该进程的所有存储对象移动到新节点（随后成为该本地节点）。虽然用该方法可以对移动存储对像的时间范围进行延迟，但是该方法引入了与该线程的存储对象位于不同的本地MCM中时上面所述的跨MCM窃取线程或在扩展周期内在远程MCM上运行线程相同的代价。

因此，本发明认识到需要一种新的机制，该机制允许在不引起指派给空闲处理器周期的线程的永久降级时使用这些空闲周期。防止有关线程长期降级的用于MCM到MCM平衡的新的负载平衡算法将是一项受欢迎的改进。本文描述的发明提供各种优点。

发明内容

这里公开了一种方法和系统，该方法和系统支持第一个MCM(多片组件）中的带有空闲处理器周期的第一处理器和第二个MCM中的第二忙处理器之间的有效负载均衡，而不会在分配给该空闲处理器周期时引起该线程的执行效率的显著降低。本发明适用于包括两个或更多多片组件(MCM)的多处理器数据处理系统(MDPS)以及同时支持跨MCM的线程窃取和借入的负载平衡算法。

允许空闲处理器从另一个存储域（即，跨MCM）中的忙处理器中“借入”线程。每次为单个调度周期借入线程。当完成该调度周期时，释放该线程回到它的父处理器。如果确定该借入处理器在该调度周期后将变为空闲，则该借入处理器重新扫描全部MDPS以查找要借入的另一个线程。

下一个被借入的线程可能来自同一借出处理器或来自另一个忙处理器。同样，该借出处理器可以借给该借入处理器一个不同线程。因此，该分配算法不向另一个MCM“指派”线程。相反，每次在其它MCM上为单个调度周期运行该线程，并且把该线程的执行立即返回到位于其它MCM的本地（借出）处理器。

通过使该借入处理器释放该线程并然后重新扫描全部MDPS，该算法显著减少了任何单一线程在特定借入处理器上连续运行的可能

性。因此，该算法也显著减少了对于因存储位置遗失而造成的被借入的线程的任何性能恶化将积累的可能性，因为将相对于它的本地MCM而言本地地分配被借入的线程创建的任何新存储对象。

通过阅读下面的详细书面描述，本发明的上述目的和附加目的、特征及其优点将更加显而易见。

附图说明

在连同附图一起阅读时，通过参照示例实施方式的下述详细描述，将更好地理解本发明、其优选使用方式、其进一步的目的及其优点，其中：

图 1 是根据本发明之一个实施方式的其中可以有利实现本发明之特征的带有两个多片组件(MCM)的多处理器数据处理系统(MDPS)的方框图；

图 2 是一个流程图，说明根据本发明之一个实施方式跨两个MCM借入线程的处理；

图 3 是一个流程图，说明根据本发明之一个实施方式，负载均衡算法确定带有空闲周期的处理器是否应该从忙处理器中窃取或借入线程的处理；以及

图 4 是一个略图，说明根据本发明之一个实施方式，在每个调度周期跨 MCM 的线程的借入。

具体实施方式

本发明提供一种方法和系统，该方法和系统支持第一个MCM（多片组件）中的带有空闲处理器周期的第一处理器和第二个 MCM 中的第二忙处理器之间的有效负载均衡，而不会在分配给该空闲处理器周期时引起该线程的执行效率的显著（长期）降低。本发明适用于包括两个或更多多片组件(MCM)的多处理器数据处理系统(MDPS)以及同时支持跨 MCM 的线程窃取和借入的负载平衡算法。

正如本文使用的那样，术语“空闲”系指当前不处理任何线程或

尚未给它的线程队列指派任何线程的处理器。相反，“忙”系指其处理器的线程队列中安排有几个需要执行的线程的处理器。在负载均衡算法内，可以将该参数定义为该处理器线程队列内的具体线程数(如，4个线程)。作为选择，可以基于处理期间计算的跨该MDPS的平均值，定义该忙参数，其中相对于其它处理器，把明显高于平均值的处理器标记为忙。该负载均衡算法保持(或试图保持)一个平滑的平均负载值，后者是通过对每个处理器的队列长度进行重复取样确定的。

允许空闲处理器从另一个存储域(即，跨MCM)中的忙处理器中“借入”线程。每次为单个调度周期借入该线程。当完成该调度周期时，释放该线程回到它的父处理器。如果确定该借入处理器在该调度周期后空闲，则该借入处理器重新扫描全部MDPS以查找要借入的另一个线程。

下一个被借入的线程可能来自同一借出处理器或来自另一个忙处理器。同样，该借出处理器可以借给该借入处理器一个不同线程。因此，该分配算法不向另一个MCM“指派”线程。相反，每次在其它MCM上运行该线程单个调度周期，并且把该线程的执行立即返回到位于其它MCM的本地(借出)处理器。

通过使该借入处理器释放该线程并然后重新扫描全部MDPS，该算法显著减少了任何单一线程在特定借入处理器上连续运行的可能性。最后，利用该借出MCM的本地存储器而不是实际执行被借入的线程的MCM的本地存储器，解决被借入的线程对存储对象的所有引用。被借入的线程仍然是优选的，以便将来在它的“本地”MCM上执行。因此，该算法也显著减少了对于因存储位置遗失而造成的被借入的线程的任何性能恶化将积累的可能性，因为当它在它的本地MCM上运行时，该进程不需要存储对象的跨MCM移植。

现在参照附图，并特别地，参照图1，该图说明用来描述本发明之特征的带有两个4处理器多片组件(MCM)的示例性多处理器数据处理系统(MDPS)。MDPS 100包括两个MCM，MCM1 110和MCM2

120。每个 MCM 包括四个处理器，即用于 MCM1 110 的 P1-P4 以及用于 MCM2 120 的 P5-P8。处理器 P1-P4 共享公用 L3 高速缓冲存储器 112 和存储器 130，而处理器 P5-P8 共享公用 L3 高速缓冲存储器 122 和存储器 131。存储器 130 是 MCM1 110 的本地存储器，而存储器 131 是 MCM2 120 的本地存储器。每个存储器 130 和 131 分别具有用于非本地 MCM，即 MCM2 120 和 MCM1 110 的远程访问代价。

经由开关 105，把 MCM1 110 连接到 MCM2 120。开关 105 是一组连接线，在一个实施方式中，连接线使 MCM1 110 的每个处理器能够直接连接到 MCM2 120 的每个处理器。开关 105 也把存储器 130、131 连接到各自的本地 MCM（和非本地 MCM）。

在 MDPS 100 运行时，为每个处理器（或中央处理单元(CPU)）指派一个执行队列（或线程队列）140，在队列内安排各线程（标记为 Thl...THn）以便特定处理器执行。在处理期间的任何给定时刻，任意一个处理器正在处理（顺序执行）的线程（负载）数可以与另一个处理器正在处理的线程（负载）数不同。同样，一个 MCM（如，MCM1 110）的总负载可以与另一个 MCM (MCM2 120)的总负载非常不同。在图 1 中，以具体处理器内的“事务”标记（忙、平均、低和空闲）的方式，提供各处理器的相对负载的指示，并利用“长度”标记（长、中、短和空）指示相应队列中的线程数。假定该负载参数直接与安排在特定处理器执行的线程数（即，该队列的长度）相互关联。

因此，如图所示，MCM1 110 的处理器 P1 和 P4 具有安排有 4 个（或更多）线程的长队列，P1 和 P4 标记为“忙”。MCM1 110 的处理器 P2 和 P3 以及 MCM2 120 的处理器 P5 具有中长度队列（安排有两个线程），P2、P3 和 P5 标记为“平均”。把 MCM2 120 的处理器 P7 和 P8 标记为“低”，因为它们分别具有仅仅安排了一个线程的短队列。最后，MCM2 120 的处理器 P6 具有空队列（即，没有安排线程），P6 标记为“空闲”。

本文提供的具体线程计数仅仅用于说明，并不意味着对本发明的

任何限制。具体地，尽管把空闲处理器描述为不给它指派线程，但是应该理解，用于确定哪个处理器具有空闲周期并且作为借入（或窃取）线程的候选者的阈值，是由在特定 MDPS 内实现的负载均衡算法设置的。该阈值可以是安排有 2 个、3 个或 10 个线程的处理器，在某种程度上，这取决于该线程队列的深度和该处理器的操作参数。然而，所示实施方式假定只有借入/窃取处理器的“运行队列”为空时，该借入/窃取处理器才借入（或窃取）线程。此时，使用该负载平均值确定是否允许该处理器从另一个处理器中借入（或窃取）线程。

特别地，MCM2 120 的总负载（即，其上执行的线程数）明显低于 MCM1 110 的总负载。利用该不均衡性来描述本发明的负载均衡处理，目的是减轻负载不均衡性，特别是减轻忙处理器 P1 的负载不均衡性，而不会引起线程执行效率的任何明显的长期降低。因此，提供本发明的描述的目的是，基于考虑经由窃取算法可获得的负载减轻的负载均衡分析，通过实现适当的借入算法，解决跨 MCM 的负载不均衡性。

因此，使用两个 MCM 之间的显著负载平均值差值确定何时允许窃取。当缺乏此类显著不均衡性时，如果借入节点有显著的空闲时间（即，每个处理器的负载平均值相对较小）并且借出节点没有显著的空闲时间，则允许借入。如果节点有显著的空闲时间，则在本地窃取线程而不执行跨 MCM 的借入。

参照图 4 概括描述本发明的特征，该图表示第二个 MCM 的一个处理器在每个调度周期从第一 MCM 的一个处理器中借入线程。更具体地说，把 MCM2 120 的空闲处理器 P6 表示成从 MCM1 110 的忙处理器 P1 中借入线程。在下面的说明中使用具体处理器完全是为了便于描述该过程，并不意味着对本发明的限制。此外，请注意，图 4 最初假定 MCM2 中有一个空闲处理器并且 MCM 1 中没有空闲处理器。因此，图 4 说明的初始线程借入是跨 MCM 的，而不是发生在本地 MCM 内。

在图 4 中，利用下标“b”标识被借入的线程，而利用下标“s”标识来自同一个（本地）MCM 中的另一个处理器的被窃取的线程。当在线程的本地处理器上执行该线程时，不提供下标（“空”）。在第一调度周期 402 内，P6 是空闲的，而 P1 非常忙（必须安排 4 个线程）。在第二调度周期404内，P6已经从P1中借入一个线程(Th1)，并且 P6 在该调度周期内执行该线程(Th1)。一旦第二调度周期完成，P6 就释放该线程(Th 1)回到 P1。

接着，在第三调度周期 406 内，P6 再次从 P1 中借入一个线程。然而，本次借入的线程(Th3)不同于最初借入的线程(Th1)。同样，当该调度周期结束时，P6 释放该线程(Th3)回到 P1。在第四调度周期 408 内，P6 接收它自己的线程以便执行或者接收来自本地 MCM 的线程。P1 继续执行它的 4 个线程，而 P6 开始执行它的本地线程或它的 MCM 的本地线程。

图 3 是一个流程图，说明通过当适当时使用同时支持线程窃取和借入的负载均衡算法处理 MDPS 中的负载不均衡性的两种不同方式的通路。把处理器称之为忙处理器、窃取处理器、空闲处理器和借入处理器，以指示各个处理器的负载均衡状态。该处理在块 302 开始，在块 302 中，计算该 MDPS 的负载的加权平均值。然后在块 304 中，判断检测的不均衡是否超过允许的授权窃取（仅与借入相对）线程的最小不均衡阈值。当超过最小阈值时，在块 306 中，把忙处理器的完整的线程重新指派给其它先前空闲的（不太忙的）处理器。在块 308 中，也将该线程的存储位置变成与窃取处理器有关连的存储器。特别地，MCM 之间的/跨 MCM 的窃取要求两个 MCM 有显著的负载不均衡性，而 MCM 内的窃取没有上述严格要求。

回到判定块 304，当上述不均衡没有超过开始窃取处理要求的阈值时，在块 310 中进行下一个判断，判断检测的不均衡是否处于跨 MCM 的借入阈值。当没有超过借入阈值时，在块 312 中结束负载均衡处理。然而，当超过该阈值时，激活跨 MCM 的借入算法，并且以调度周期为间隔开始线程的 MCM 到 MCM 借入，如块 314 所示。

与该线程窃取算法不同，被借入的线程的存储位置等继续保持在借出处理器的 MCM 中，如块 316 所示。

图 2 是在图 1 的 MDPS 100 内提供跨 MCM 的负载均衡的处理的流程图。该处理的假设包括：(1) MCM2 120 中要求减轻负载的所有忙处理器促使从本地 MCM 中窃取线程（即，正如前面所述的,参照图 3 进行窃取线程）；(2) MCM1 110 中有一个忙处理器并且 MCM2 120 中有一个带有空闲周期的处理器；以及(3)借入处理器最初是空闲的。该流程图显示的顺序并不意味着对本发明的任何限制，应该理解，在该处理中可以相对于彼此重新排列不同的块。

图 2 的处理在块 202 开始，块 202 说明负载均衡（或借入）算法开始扫描 MDPS 100，查找供 MCM2 120 的空闲处理器 P6（可交替称为空闲处理器或借入处理器或处理器 P6，以标识该处理器的当前状态）借入的线程。在搜索要借入或窃取的线程之前，处理器 P6 必须首先判断它自己的运行队列上是否有任务（安排的线程），如果有的话，则完成安排的线程的执行。只有它自己的队列中没有安排的线程时，处理器 P6 才可以开始扫描以从另一个处理器中窃取或借入线程。

回到图 2，在块 204 中判断本地 MCM2 120 中是否有可用线程。如果有空闲处理器 P6 的 MCM2 120 的本地线程，则使空闲处理器 P6 从 MCM2 120 的忙本地处理器的一个处理器中窃取线程，如块 210 所示。

当没有可以供空闲处理器 P6 从中窃取线程的忙本地处理器时，在块 206 中进行下一个判断，即判断 MCM1 110 内是否有带有可用线程的忙处理器。该算法使得空闲处理器 P6 继续扫描该 MDPS，直至空闲处理器 P6 找到要借入或要窃取的线程，或者直至给空闲处理器 P6 指派了线程并且其不再空闲。

当 MCM1 110 的处理器中有可用线程时，空闲处理器 P6 接收被借入的线程，并且在块 212 中，P6 在该调度周期内执行被借入的线程。在该调度周期内，借入处理器 P6 准备被借入的线程的所有未来

数据参考，分配存储器如同为该借出处理器分配本地存储器一样，在一个实施方式中，但是不移动/改变 MCM1 110 的远程存储器内的任何先前分配。因此，借入处理器 P6 处理被借入的线程，就像它实际上是由该借出处理器运行的一样。

正好在完成该调度周期前，在块 214 中进行检查，判断该借入处理器是否将再次变成空闲（即，具有可分配给某一线程的空闲处理周期）。如果处理器 P6 将变成空闲，则该借入算法再次扫描该 MDPS 以查找可以借入或窃取的线程。特别地，空闲处理器 P6 可以依据确定的负载值窃取、借入或忽略正在另一个处理器的运行队列中等待的线程。然而，本发明仅仅描述线程的借入。

如果给处理器指派正常线程，则处理器 P6 在该调度周期后不会变成空闲。该处理器指派的/安排的线程（即，本地线程（为被窃取的线程隐含变成））保持指派在同一处理器上运行，从而在它的每个调度周期后，期望该线程在它的本地处理器上运行（除非,例如该处理器变得太忙并且迫使它将该线程借给例如另一个空闲处理器），如块 216 所示。在标准线程完成时，该处理器再次进入空闲状态，这在块 218 中进行判断。一旦处理器 P6 变成空闲的，就触发借入/窃取算法以自动搜索处理器 P6 可以从中借入/窃取线程的忙处理器。

在一个实施方式中，如果需要分页输入/输出(I/O)，则把遇到页面出错作为被借入的调度周期的终止条件。假定在解决页面出错后该线程很可能转到重新开始在自己的/借出处理器上执行。因此，无论该线程下次在什么地方运行，将使该页面驻留在该线程的本地 MCM 的本地存储器中（除非该线程被另一个 MCM 中的处理器窃取）。

正如下面更加详细描述的那样，有两个借入负载平均值要求：(1)该借入处理器和它的 MCM 都必须有“足够的”预期的余暇（周期）进行分发，以及(2)该借出处理器和它的 MCM 不能有“足够的”预期的余暇（周期）立刻到达该线程。

该实现的若干附加重要细节包括：

（1）尽管 AIX 中的新的存储关系管理代码通常将从包含执行该线程的处理器（即，该借入处理器）的 MCM 的本地存储器中分配页面，但是该线程借入算法促使分配到“自己的”（借出）处理器的 MCM。这样，因为该线程并不打算在该借入处理器上长期运行，所以设置支持参数以优化该线程将来很可能在其中运行的该线程的存储位置；

（2）该负载均衡协议包括用来防止不合需要的借入的新“障碍”。因此，另外的忙 MCM 中的空闲处理器不会把周期借给另一个 MCM，而是等待并且把该空闲周期给它自己的本地 MCM 中的空闲处理器。同样，一个 MCM 中的空闲处理器也不会把处理周期借给另一个负载较轻的 MCM 的忙处理器。因此，在一个实施方式中，该线程借入算法仅仅是指导性的，因为该负载均衡算法通常假定，最好使本地 MCM 内的很快成为空闲的空闲处理器执行正常的线程窃取，而不准许跨 MCM 的线程借入。如上所述，准许另一个 MCM 中的一个处理器借入时两个有关 MCM 必须达到的忙程度的准确值是一个设计参数，该参数在最小化因跨 MCM 的借入和窃取线程引起的低效率的情况下，最大化空闲处理器周期的使用；以及

（3）比借入具有更高障碍的窃取优先于借入。每当完成窃取和完成借入选项都是可行选项时，总是执行窃取。亦即，为了克服显著的长期负载不均衡，窃取是必需的，并且不使用大量的借入来隐藏此类不均衡。特别地，窃取障碍是有关 MCM 的负载平均值的函数。此外，在线程分配期间进行分析以防止该借入处理使这些负载平均值失真。

因此，通过对等待在处理器上执行的线程队列的长度进行取样，确定该处理器的负载平均值。在借入为可用选项的情况下，取样变为：队列长度 + 发送到其它处理器的线程数 −B，其中仅当该处理器正在运行被借入的线程时 B 才为 1，否则 B 为 0。

本发明的益处包括，实现了用于 MCM 到 MCM 的均衡的新的负载均衡算法，该算法能够防止有关线程的长期降级。换句话说，跨

MCM 的借入算法导致降低了任何一个线程的代价。所有线程必须在负载均衡期间分享临时再分配，并且因此系统性能保持一致。同样，在某些情况中，借入帮助处理器充分减少该处理器的积压。

最后，重要的是，尽管已经并将继续在安装有管理软件的全功能数据处理系统的上下文中描述本发明的示例实施方式，但是本领域技术人员应该理解，能够以各种形式的程序产品的方式体现本发明的示例实施方式的软件方面，并且不管实际执行分发使用的信号承载介质的特定类型，本发明的示例实施方式同样适用。信号承载介质的例子包括诸如软盘、硬盘驱动器、CD ROM 之类的可记录类型的介质，以及诸如数字和模拟通信链路之类的传输类型的介质。

尽管参照示例实施方式详细展示和描述了本发明，但是本领域技术人员懂得，其形式和细节可以作出各种变更而并不背离本发明的实质和范围。例如，尽管借助使用线程计数来计算和保持负载平均值的负载均衡算法具体描述了本发明，但是一种实现方法可以跟踪处理器的相对事务（通过使用不同于各自队列中的线程数的某些其它机制）并利用该负载均衡算法内的忙参数。同样，尽管本发明描述为 MCM 到 MCM 的操作，但是本发明并不限于此类体系结构，并且可以利用负责非均匀存储器存取(NUMA)体系结构的机制实现。

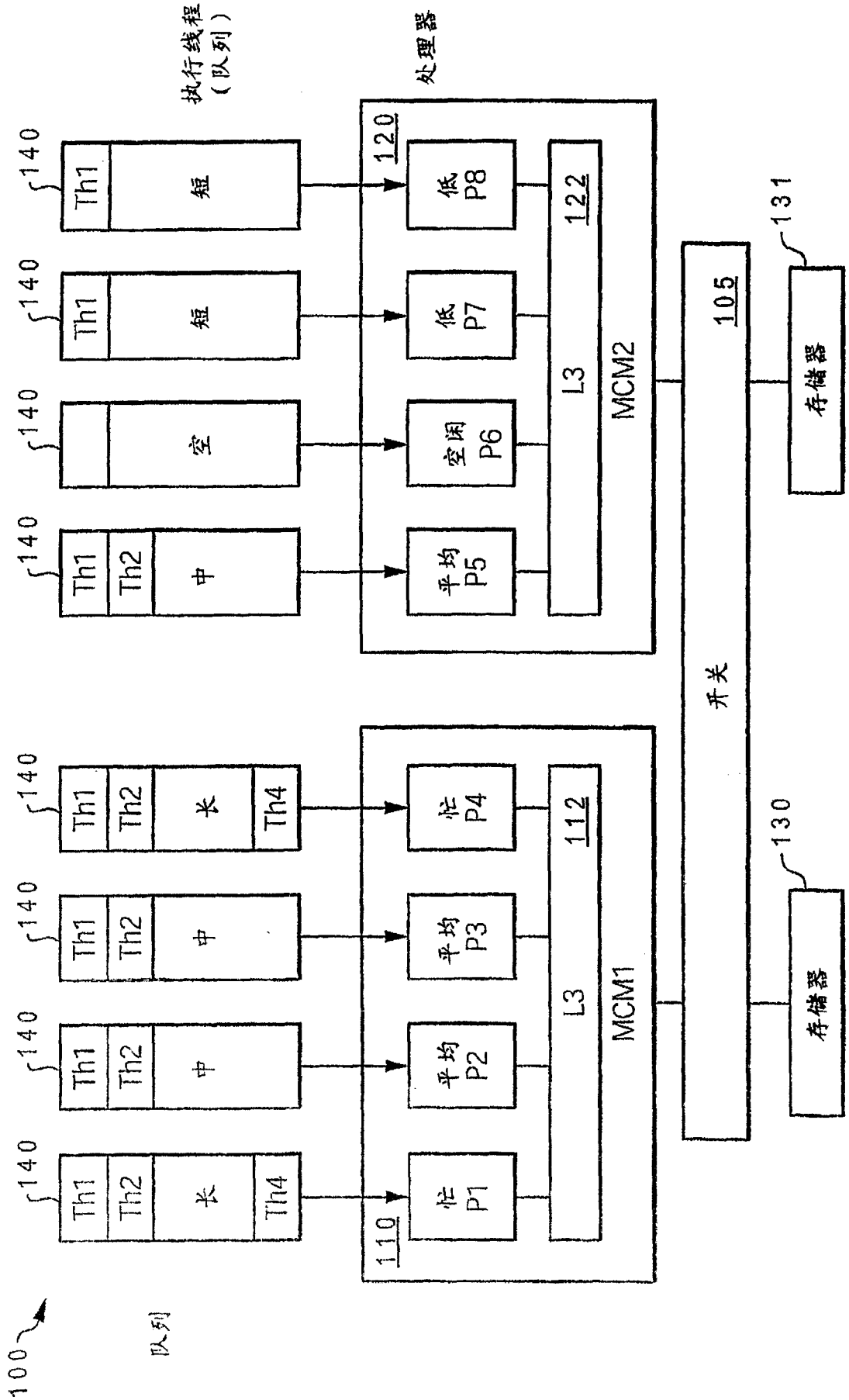

100
队列
执行线程（队列）
处理器
140
Th1
Th2
Th4
短
中
长
空
忙 P1
平均 P2
平均 P3
忙 P4
空闲 P6
低 P7
低 P8
110
112
L3
MCM1
120
122
L3
MCM2
105
开关
130
131
存储器
存储器

图 1

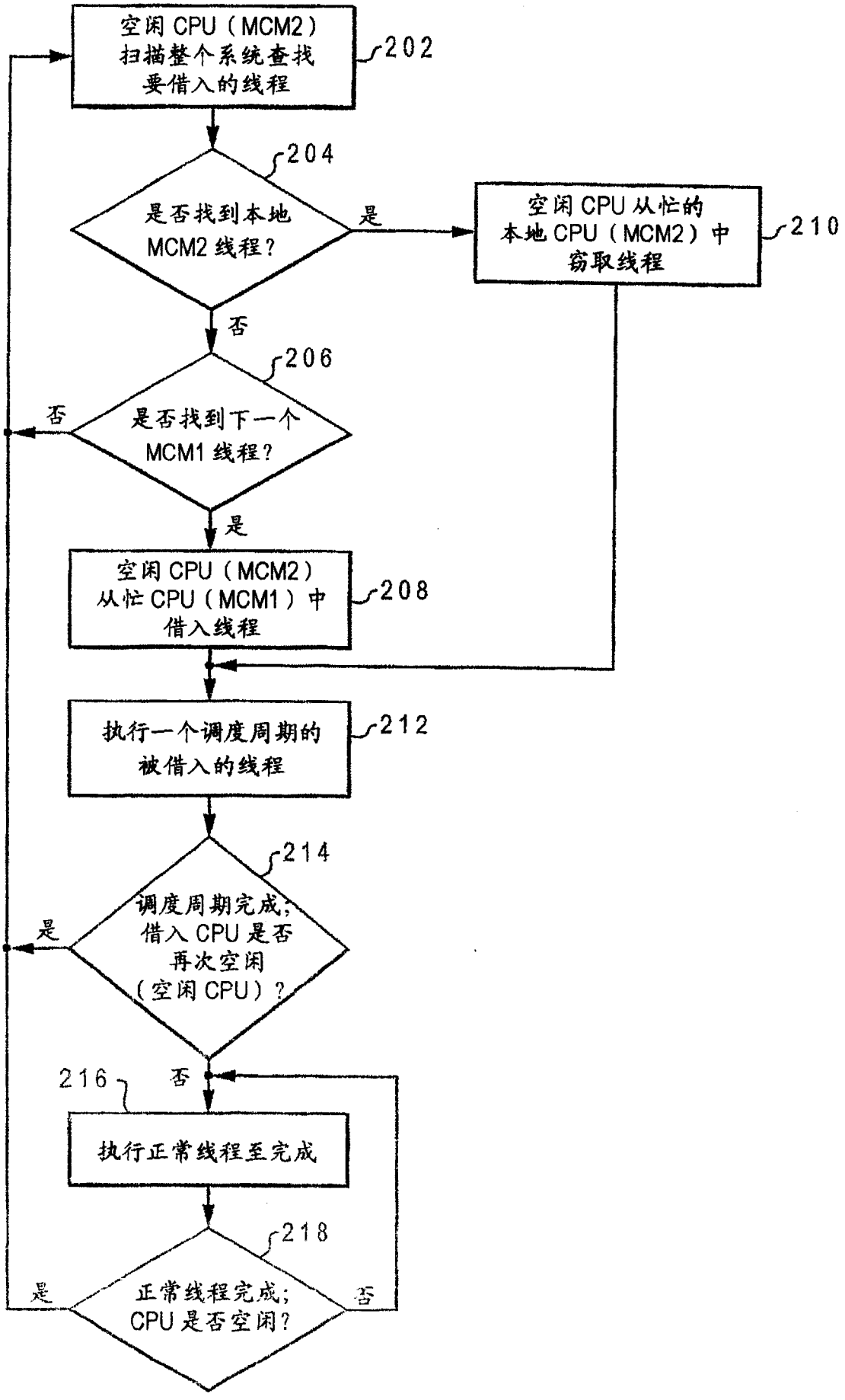

空闲 CPU（MCM2）扫描整个系统查找要借入的线程
202
204
是否找到本地 MCM2 线程？
是
空闲 CPU 从忙的本地 CPU（MCM2）中窃取线程
210
否
206
否
是否找到下一个 MCM1 线程？
是
空闲 CPU（MCM2）从忙 CPU（MCM1）中借入线程
208
执行一个调度周期的被借入的线程
212
214
是
调度周期完成；借入 CPU 是否再次空闲（空闲 CPU）？
216
否
执行正常线程至完成
218
是
正常线程完成；CPU 是否空闲？
否

图 2

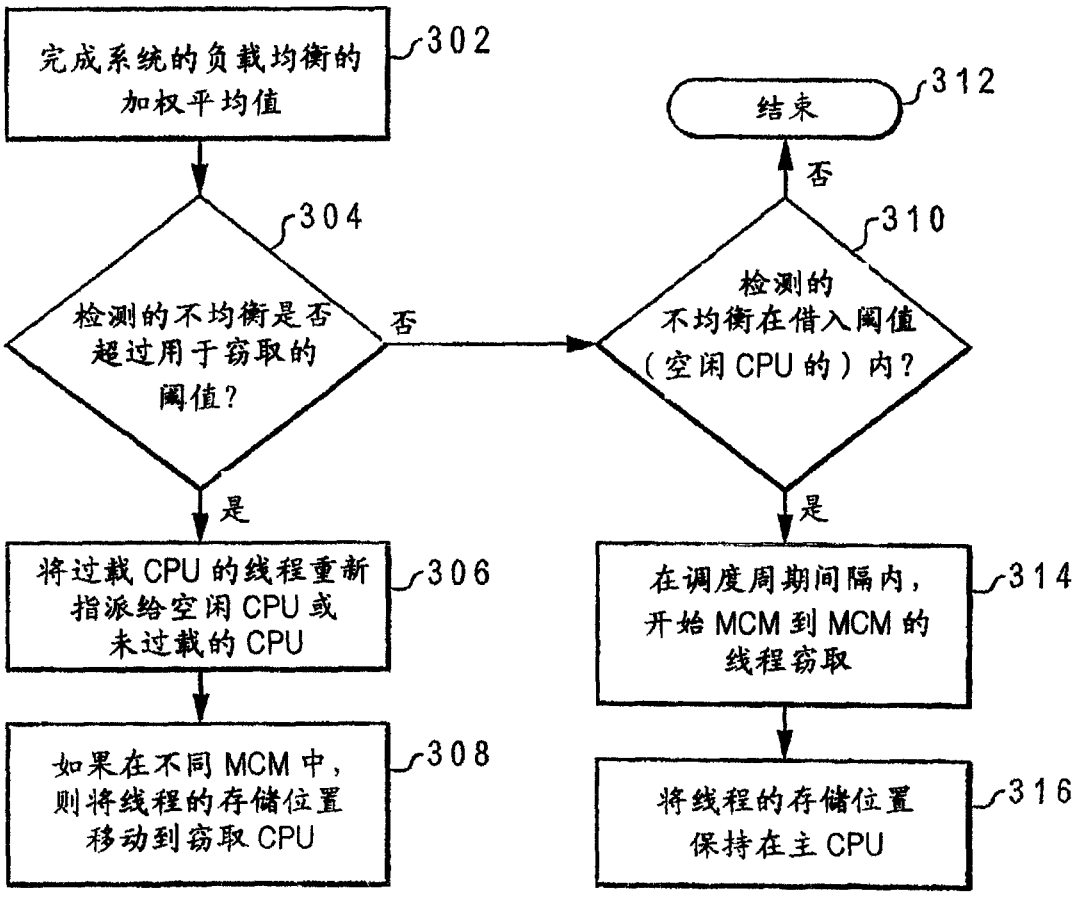

完成系统的负载均衡的加权平均值
302
检测的不均衡是否超过用于窃取的阈值?
304
否
是
将过载 CPU 的线程重新指派给空闲 CPU 或未过载的 CPU
306
如果在不同 MCM 中，则将线程的存储位置移动到窃取 CPU
308
检测的不均衡在借入阈值（空闲 CPU 的）内?
310
否
是
结束
312
在调度周期间隔内，开始 MCM 到 MCM 的线程窃取
314
将线程的存储位置保持在主 CPU
316

图 3

MCM1 CPU1 (忙)	MCM2 CPU2	
TH1, TH2, TH3, TH4 (忙)	空闲	调度 0 ~402
TH2, TH3, TH4	$TH1_b$	调度 1 ~404
TH1, TH2, TH4	$TH3_b$	调度 2 ~406
TH1, TH2, TH3, TH4	TH1 或 $TH4_{bl}$	调度 3 ~408

b= 被借入的远程 MCM

bl= 被借入的本地 MCM

"空"=自己的线程

图 4