(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 102652309 B
(45)授权公告日 2015.11.25

(21)申请号 201080055635.1

(22)申请日 2010.12.07

(30)优先权数据
12/635,544 2009.12.10 US

(85)PCT国际申请进入国家阶段日
2012.06.08

(86)PCT国际申请的申请数据
PCT/IB2010/055626 2010.12.07

(87)PCT国际申请的公布数据
WO2011/070506 EN 2011.06.16

(73)专利权人 国际商业机器公司
地址 美国纽约

(72)发明人 H·C·亨特 R·P·路易顿 P·斯坦利-玛贝尔

(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 11038
代理人 鲍进

(51)Int.Cl.
G06F 9/50(2006.01)
G06F 9/48(2006.01)
G06F 12/12(2006.01)
G06F 12/08(2006.01)

(56)对比文件
EP 0848330 A2,1998.06.17,
US 2007156963 A1,2007.07.05,
US 2008133416 A1,2008.06.05,
US 7093147 B2,2006.08.15,
Pierre Michaud.Exploiting the Cache Capacity of a Single-Chip Multi-Core Processor with Execution Migration.《IEEE Proceedings 10th International Symposium on High Performance Computer Architecture》.2004,
Theofanis Constantinou* et al.Performance Implications of Single Thread Migration on a Chip Multi-Core.《SIGARCH Computer Architecture News》.2005,

审查员 王洋

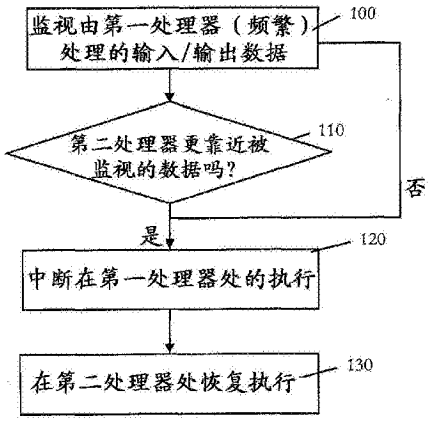
权利要求书2页 说明书8页 附图4页

(54)发明名称

处理资源管理的计算机实现的方法

(57)摘要

本发明公开涉及一种用于管理计算机化系统的处理资源的计算机实现的方法，所述计算机化系统至少具有第一处理器和第二处理器，每个所述处理器都操作性地互连到存储要由处理器处理的一组数据的存储器，所述方法包括：在执行的同时监视被所述第一处理器访问的数据；以及如果所述第二处理器比所述第一处理器离所监视的数据的距离更短，就指示中断在所述第一处理器处的执行并在所述第二处理器处恢复执行。



CN 102652309 B

1. 一种用于管理计算机化系统的处理资源的计算机实现的方法，所述计算机化系统至少具有第一处理器和第二处理器，每个所述处理器都操作性地互连到适于存储一组数据的主存储器，所述计算机实现的方法包括：

在执行的同时监视被所述第一处理器作为输入数据或输出数据中的至少一个进行处理的所述一组数据的数据；以及

如果基于所述监视，发现所述第二处理器比所述第一处理器离所监视的数据的距离更短，就指示中断在所述第一处理器处的执行并且在所述第二处理器处恢复执行。

2. 如权利要求 1 所述的计算机实现的方法，其中在所述指示步骤，进一步指示中断在所述第一处理器处的执行，所述第一处理器处于给定处理器状态，并且从至少部分地由所述给定处理器状态确定的处理器状态开始，在所述第二处理器处恢复执行。

3. 如权利要求 2 所述的计算机实现的方法，其中在所述指示步骤，进一步指示：

在所述第二处理器中加载捕捉部分或者全部所述给定处理器状态的状态数据，以及

根据所加载的状态数据在所述第二处理器处恢复执行。

4. 如权利要求 3 所述的计算机实现的方法，其中在所述指示步骤，在指示中断在所述第一处理器处的执行之前，进一步指示开始在所述第二处理器中加载所述状态数据。

5. 如权利要求 4 所述的计算机实现的方法，其中在所述指示步骤，进一步指示基于来自所述第一处理器的至少一个处理器寄存器的数据，开始加载所述状态数据。

6. 如权利要求 4 所述的计算机实现的方法，其中所述状态数据捕捉所述给定处理器状态的一子部分，并且其中在所述指示步骤，指示：

开始在所述第二处理器中加载组装后的状态数据，由此所述第二处理器的至少一个处理器寄存器中的部分数据保持不变；以及

根据所加载的状态数据和保持不变的数据二者，在所述第二处理器处恢复执行。

7. 如权利要求 1 所述的计算机实现的方法，其中所述指示恢复的步骤还包括：在线程在所述第一处理器处被挂起之后，指示在所述第二处理器处继续执行所述线程。

8. 如权利要求 1 所述的计算机实现的方法，还包括：在指示中断在所述第一处理器处的执行之后，指示维持所述第一处理器的处理器状态直到指示将捕捉第三处理器状态的至少一部分的状态数据加载到所述第一处理器中为止的步骤。

9. 如权利要求 8 所述的计算机实现的方法，还包括：在指示中断在所述第一处理器处的执行之后，在暂时性延迟之后指示关闭所述第一处理器的步骤。

10. 如权利要求 1 所述的计算机实现的方法，还包括：指示将第一存储器高速缓存的数据写入由所述第二处理器使用的第二存储器高速缓存的步骤，所述第一存储器高速缓存存储所述存储器中的由所述第一处理器最频繁访问主存储器中的数据拷贝。

11. 如权利要求 1 所述的计算机实现的方法，所述方法在计算机化系统中实现，其中处理器经互连操作性地互连到所述主存储器，其中监视数据还包括监视存储在所述互连上的处理后的数据的地址。

12. 如权利要求 1 所述的计算机实现的方法，其中从所述一组数据的给定数据到所述第一处理器或所述第二处理器中的一个的距离是从存储在所述主存储器上的给定数据到所述第一处理器或所述第二处理器中的一个的等待时间的函数。

13. 一种计算机系统，具有：

第一处理器；

第二处理器；

一个或多个控制器；以及

存储器，适于存储一组数据以使得所述一组数据中的给定数据到每个所述处理器的距离不相等，

其中，

所述控制器、所述处理器和所述存储器操作性地彼此互连，并且所述一个或多个控制器被配置成实现如权利要求 1 所述的计算机实现的方法。

14. 一种用于管理计算机化系统的处理资源的计算机实现的方法，所述计算机化系统包括多个处理器，其中有第一处理器和第二处理器，每个所述处理器都操作性地互连到适于存储要由所述处理器处理的一组数据的主存储器，所述方法包括：

在执行的同时监视被所述第一处理器作为输入数据或输出数据中的至少一个进行处理的所述所述一组数据中的数据；以及

如果所述第二处理器比所述第一处理器离所监视的数据更近，就指示：

中断在所述第一处理器处的执行，所述第一处理器处于给定处理器状态；以及

从至少部分地由所述给定处理器状态确定的处理器状态开始，在所述第二处理器处恢复执行。

15. 如权利要求 14 所述的计算机实现的方法，其中在所述指示步骤，进一步指示：在所述第二处理器中加载捕捉所述给定处理器状态的至少一部分的状态数据，以及根据所加载的状态数据在所述第二处理器处恢复执行。

16. 如权利要求 15 所述的计算机实现的方法，其中在所述指示步骤，在中断在所述第一处理器处的执行之前，进一步指示开始在所述第二处理器中加载所述状态数据。

17. 如权利要求 14 所述的计算机实现的方法，其中在所述指示步骤，在线程在所述第一处理器处被挂起之后，进一步指示在所述第二处理器处继续执行所述线程。

18. 如权利要求 14 所述的计算机实现的方法，其中所述指示步骤还包括：指示把第一存储器高速缓存的至少一部分传输到由所述第二处理器使用的第二存储器高速缓存，其中所述第一存储器高速缓存存储由所述第一处理器最频繁访问的数据拷贝。

19. 如权利要求 18 所述的计算机实现的方法，其中所述第一和第二存储器高速缓存分别是由所述第一处理器和所述第二处理器使用的最小高速缓存。

20. 如权利要求 18 所述的计算机实现的方法，其中所述第一和第二存储器高速缓存分别是由所述第一处理器和所述第二处理器使用的唯一存储器高速缓存。

21. 如权利要求 18 所述的计算机实现的方法，其中所述第一处理器和所述第二处理器都不使用存储器高速缓存。

22. 如权利要求 18 所述的计算机实现的方法，所述方法在计算机化系统中实现，其中处理器经互连操作性地互连到所述存储器，其中监视被所述第一处理器处理的数据还包括：监视存储在所述互连上的处理后的数据的地址，由此能够确定从所述处理后的数据到所述第一处理器或所述第二处理器的距离。

23. 如权利要求 22 所述的计算机实现的方法，其中所述距离对应于对于处理器离所监视的数据在空间上有多远的测量。

处理资源管理的计算机实现的方法

背景技术

[0001] 计算机是根据指令处理数据的机器。如今，它们大部分被配置成诸如将其工作分布到数个 CPU 上，从而提供多处理能力。现在，个人和膝上型计算机也可以使用多处理器和多内核系统，而不再仅限于超级计算机、大型计算机或者服务器。然而，最大的计算机仍然受益于与普通计算机显著不同的独特体系结构。例如，它们常常以数以千计的处理器、高速互连和专用硬件为特征。

[0002] 不管是在多处理器背景下还是在非多处理器背景下，对计算机系统来说，一个挑战是要提高其整体性能，同时降低合计的功耗。除此之外，如今大多数 CPU 都倾向于花时间等待内存、I/O、图形等等，使得提高单个 CPU 指令执行的性能不再是主要可能的发展方向。

[0003] 例 如，Brown, J.A. 与 Tullsen, D.M. 的 论 文(在 Proceedings of the 22nd Annual international Conference on Supercomputing(Island of Kos, Greece, June 07-12, 2008) 学报上标题为“The shared-thread multiprocessor”，ICS'08. ACM，纽约，NY，73-82. DOI=http://doi.acm.org/1 0.1 145/1375527.1375541）描述了共享线程多处理器(STMP)的体系结构的结果。STMP 结合了多线程处理器与芯片多处理器的特征。具体而言，它使得芯片多处理器上不同的内核可以共享线程状态。这种共享的线程状态允许系统把线程从共享池调度到单个内核上，从而允许线程在内核之间的快速移动。该论文证明和评估了这种体系结构的好处。

[0004] 其它方法集中在：

[0005] 集成到存储器阵列结构中的多个处理器，参见，例如，Duncan G. Elliott, W. Martin Snelgrove, 和 Michael Stumm 所写的 Computational RAM: A Memory-SIMD Hybrid and its Application toDSP. 在 Custom Integrated Circuits Conference 上，pages30.6.1-30.6.4, Boston, MA, May 1992；

[0006] 集成到芯片上的多个处理器与存储器宏(PIM)，参见，例如，Maya Gokhale, Bill Holmes, 和 Ken Iobst 所写的 Processing in Memory: the Terasys Massively Parallel PEVI Array. Computer, 28(3): 23-31, April 1995；

[0007] 集成到芯片上的多个处理器与存储器宏(Execube)，参见，例如，Peter M. Kogge. EXECUBE-A New Architecture for Scalable MPPs. 在 1994 International Conference on Parallel Processing 上，pages177-184, August 1994；以及

[0008] IRAM，参见，例如，David Patterson, Thomas Anderson, Neal Cardwell, Richard Fromm, Kimberly Keeton, Christoforos Kozyrakis, Randi Thomas, 和 Katherine Yelick 所写的 Intelligent RAM(IRAM): Chips that Remember and Compute″，发表在 1997 IEEE International Solid-State Circuits Conference(ISSCC)6-8 February1997, San Francisco, CA.。

发明内容

[0009] 描述了用于管理计算机化系统的处理资源的计算机实现的方法的各实施方式。在

有些实施方式中，计算机化系统至少包括第一处理器和第二处理器，每个所述处理器都操作性地互连到适于存储一组数据的存储器。在一种实施方式中，该计算机实现的方法包括在执行的同时监视被所述第一处理器作为输入数据或输出数据中的至少一个进行处理的一组数据。该计算机实现的方法还包括：如果基于所述监视，发现所述第二处理器比所述第一处理器离所监视的数据的距离更短，就指示中断在所述第一处理器处的执行并且在所述第二处理器处恢复（resume）执行。

[0010] 描述了另一种用于管理计算机化系统的处理资源的计算机实现的方法的各实施方式。在有些实施方式中，计算机化系统包括多个处理器，其中有第一处理器和第二处理器，每个所述处理器都操作性地互连到适于存储要由处理器处理的一组数据的存储器。在一种实施方式中，该计算机实现的方法包括：在执行的同时监视被所述第一处理器作为输入数据或输出数据中的至少一个进行处理的所述组中的数据。此外，如果发现第二处理器比第一处理器离所监视的数据更近，该计算机实现的方法还包括：中断在所述第一处理器处的执行，所述第一处理器处于给定处理器状态，以及从至少部分地由所述给定处理器状态确定的处理器状态开始，在所述第二处理器处恢复执行。

[0011] 从以下的具体描述并联系附图，本发明实施方式的其它方面与优点将变得显而易见，附图是作为本发明原理的例子来说明的。

附图说明

[0012] 图 1 是用于管理计算机化系统的处理资源的方法的一种实施方式的流程图；

[0013] 图 2 是更具体的流程图，说明了用于管理计算机化系统的处理资源的方法的一种具体实施方式；

[0014] 图 3 示意性地说明了线程执行被挂起 / 继续的一种实施方式；

[0015] 图 4 是实现用于管理计算机化系统的处理资源的方法的一种实施方式的计算机系统的示意性表示；以及

[0016] 图 5 和 6 是图 4 的变体实施方式。

具体实施方式

[0017] 在以下描述中，提供了各种实施方式的具体细节。但是，有些实施方式可以用比所有这些具体细节少的细节被实践。在其它情况下，为了简洁与清晰，对某些方法、过程、部件、结构和 / 或功能的描述以使本发明各种实施方式能够运作为限，不作更具体的描述。

[0018] 在有些实施方式中，用于管理计算机化系统的处理资源的计算机实现的方法打破了传统模式，根据这种传统模式，处理器是世界的中心并且输入 / 输出数据都被带入处理器 / 从处理器带走。相反，所述计算机实现的方法的实施方式提出了把“计算”状态信息移动到数据所在的位置，例如，通过把内核状态移动到最靠近的内核，而不是把数据移动到 CPU。基本上，这导致在实践当中移动几千字节而不是几兆字节。

[0019] 在得出这种逆向模式之前，本发明人意识到以下：

[0020] - 大量的功率是用于移动数据；

[0021] - 消耗在例如双列直插内存模块（DIMM）上的功率的一半是输入 / 输出（I/O）；

[0022] - 把更多的内核放到单个大的管芯上不能持续很长时间，更不用说每个内核的带

宽问题、功耗和制冷的挑战；以及

[0023] - 高速缓存使用大量的面积和功率。

[0024] 相应地，移动数据的成本变成比移动计算更本质的问题，由此实现了本方法的好处。

[0025] 图 1 是用于管理计算机化系统的处理资源的方法的一种实施方式的流程图。该方法旨在提出一种管理计算机化系统中的处理资源的新途径，其中几个处理器操作性地互连到存储器。存储器适于存储一组数据，这些数据可以由所述处理器中的一个或多个使用。一般来说，所关心的数据是由内核读取（“输入数据”）和 / 或生成的并且写回（“输出数据”）。该方法包括两个主要的步骤，现在我们来讨论。

[0026] 为了简化，现在只考虑两个处理器。

[0027] 首先，在步骤 100，监视由处理器中的第一处理器处理的输入和 / 或输出数据。

[0028] 其次，如果发现第二处理器比第一处理器离所监视的数据的距离更短（测试 110），就在步骤 120 中断在第一处理器处的执行，并在步骤 130 在第二处理器处恢复所述执行。因而，数据是从更靠近的处理器访问 / 生成的，由此避免了从 / 到第一处理器（或者其 L1-L3 高速缓存）不必要的更长距离的数据移动。

[0029] 更一般地，同时监视由 N 个处理器访问的数据。为了理解起见，图 1 流程图的目的是提供关于本发明的高级视图。

[0030] 正如所说的那样，被监视的数据可以是输入数据和 / 或输出数据。但是，下文中的实施方式是基于只监视输入数据，即，处理器读取的数据，的例子，这仅仅是为了简化。但是，不管是输入数据还是输出数据，本发明的原理都保持相同。

[0031] 在离被监视的数据有较短距离的处理器处恢复指令。所讨论的距离可以是例如对于处理器离其所处理的数据有多远的测量。例如，在以下所讨论的一种实施方式中，CPU 内核是在存储器的 3D 堆栈上；在这里，从一个给定内核到其对应存储器的距离在物理上短，而到另一个 3D 堆栈的存储器的距离就大得多。更一般地说，所讨论的距离反映了成本，例如，能量或时间的成本。

[0032] 例如，数据越远，就需要越多的功耗来运输它们（这是成本的来源）。

[0033] 距离还可以反映时间成本。在这点上，所考虑的距离可以象数据到达给定处理器的等待时间（或者预期的等待时间）一样变化。等待时间（即，访问时间）是向处理器提供它所请求的数据时存储器延迟的时间。大概地说，离处理器越远，到达其所花的时间越长。等待降低了处理器的性能；如果处理器必须等待五个存储器时钟周期来接收其所请求的数据，那么它的性能将只有在它使用能够立即传输数据的存储器时所具有的性能的 1/5。因而，在更近的处理器处恢复执行可以导致更好的性能。

[0034] 例如，到给定数据的距离可以基于它们的存储器地址与处理器位置来估计。

[0035] 此外，监视由第一处理器访问的输入数据可以基于存储在互连（例如，总线）上的输入数据的存储器地址来执行。例如，监视最近 100 个被访问的存储器地址。如果显示例如大部分被监视的地址都属于更靠近第二处理器的一个存储器或者一个存储器扇区，那么就决定在第二处理器处恢复执行。监视什么数据被处理器访问也可以经由处理器的寄存器或者其它装置来执行。

[0036] 例如，为了确定第二处理器是否比第一处理器更近，距离可以被确定为包括由第

一处理器读 / 写的数个数据(例如,最近 100 个读 / 写存储器地址)的向量。

[0037] 关于是否把执行移动到另一个处理器的决定可以在一个或多个控制器作出。所关心的控制器可以是软件或者硬件,例如,硬件电路。在后一种情况下,硬件电路适当地(例如,经一个或多个开关)既互连到处理器又互连到存储器。控制逻辑可以进一步是集中式的(一个控制器决定)或者是分布式的,即,数个控制器(例如,每个处理器一个控制器)可以竞争性地决定,如随后将要说明的那样。

[0038] 为了使这些想法清晰,考虑下面的例子。请求第一处理器 P1 执行一些代码。在某个时候,代码的执行需要两个(非常)大的数字——N1 和 N2 (即,输入数据) ——相加,应当指出,如果只是涉及充分小的数字,那么所关心的相加可以在一个周期内很好地实现。我们假定在 P1 执行的代码反映了被设计成使比可用的 RAM 大的数字相加的算法。在所给出的情况下,当开始相加时,P1 轮询其存储器高速缓存关于 N1 或者至少其一部分。出于现有目的,为了简化,我们考虑只存在一个高速缓存级,即 L1。但是,由于 N1 非常大,例如,它占用 8T (百万兆)字节的存储器,因此所述存储器高速缓存不能很容易地获得 N1 来应答 P1 的请求(高速缓存线的大小一般是 64 字节 -1K 字节)。因而,L1 随后将询问存储 N1 的不同部分的存储器位置(例如,在邻近的块中),以便在执行的同时正确地馈送给 P1。N1 各部分的地址相应地记录到例如总线中(或者某种其它互连中)。同时,监视所存储的地址(代替输入数据,对于输出数据也一样)。如果这种地址对应于远端存储器位置,这个位置离另一处理器 P2 更近,那么执行就移动到 P2,如上所述。

[0039] 顺便提一句,尽管以上的例子假定有高速缓存存储器 (cache memory),但是,即使完全没有高速缓存存储器,所述方法的实施方式也可以非常好地应用。事实上,如果不使用高速缓存 (cache),即,当处理器直接询问存储器中的输入数据的时候,所述方法的实施方式的原理也保持相同。例如,可以以基本上相同的方式监视所询问数据的地址。除此之外,可以预期处理器的寄存器成为下一级高速缓存,“更靠近”处理器,使得上述例子扩展到不依赖高速缓存存储器的情况(在一般的意义上)。

[0040] 在一种变体中,有些处理器可以有高速缓存存储器,而一个或多个其它处理器可以没有。以与主存储器针对最初冷丢失而为处理器提供服务的相同方式,具有高速缓存存储器的处理器可以用来覆盖在其它处理器处的计算的建立。因而,执行可以在 P1 开始(其中 P1 配备有高速缓存),而当发现了更合适的处理器 Pn 时,执行就移动到后者(最合适的处理器事先是不知道的)。

[0041] 接下来,可以例如通过从(至少部分地)由第一处理器的处理器状态(即,在执行被中断时的处理器状态)所确定的处理器状态开始,在第二处理器处恢复执行,来实现无缝过渡。

[0042] “处理器状态”通常指在某个时间点由(至少)处理器寄存器反映出的处理器的状态,包括,例如,状态标记。更广泛地说,它可以指处理器寄存器、锁存器和存储器(例如,高速缓存存储器)或者更多。一般来说,在处理器执行的代码和被处理器访问的输入数据是这个处理器的当前状态的一部分。

[0043] 相应地,在第二处理器处恢复执行可以通过把寄存器数据导入第二处理器来实现,假定第二处理器可以解释它们(就象具有两个相同处理器的情况)。

[0044] 在这点上,关于如何在处理器之间移动数据的令人感兴趣的细节(尽管是在不同

的背景下）可以例如在 P. Stanley-Marbell, K. Lahiri, A. Raghunathan, "Adaptive Data Placement in an Embedded Multiprocessor Thread Library," date, vol. 1, pp. 151, Proceedings of the Design Automation & Test in Europe Conference Vol. 1, 2006 找到。

[0045] 在一种实施方式中，在已知状态信息数据部分相同的那些情况下，寄存器数据的子集被导入到第二处理器中。但是，第二处理器在恢复时的状态仍然至少部分地是由第一（初始）处理器的状态确定的。在一种变体实施方式中，在导入到第二处理器之前，状态信息数据可以利用来自第一处理器的处理器寄存器的部分或全部数据来组装 (populated)，并且例如，被便利地格式化。

[0046] 更一般地说，适当地捕捉部分或全部处理器状态的任何数据都可以依赖（下文中称为状态信息数据，或者简称为状态数据）。因而，在第二处理器的执行是在第二处理器中加载一些便利的状态信息之后恢复的，关键之处仍然是把“计算”状态数据移动到（输入 / 输出）数据，而不是把数据移动到计算。正如所说，与在处理时移动被访问的（输入）数据所需的兆字节相比，这导致移动数打——有可能数百——千字节。

[0047] 图 2 示出了更具体的流程图，说明了管理计算机化系统的处理资源的方法的更具体实施方式。除了已经参考图 1 所讨论的步骤，在这里可以预期懒惰迁移，以便进一步提高关于图 1 所述的实施方式的策略的效率。具体而言，可以指示组装（步骤 112）来自第一处理器的寄存器的状态数据；所组装的数据的加载在实际中断第一处理器处的执行之前开始（步骤 112、114）。例如，当控制器识别出被（太）远的处理器访问的数据时，它可以扫描处理器的寄存器，识别出最持久的状态数据，并决定把这种数据移到更靠近的处理器。

[0048] 在一种变体实施方式中，第一处理器的状态的早先版本被发送到第二处理器。一旦中断在第一处理器处的执行，就发送编码所述早先状态与最新状态之间的差异的增量文件。因而，最终只需要移动非常少的数据（几千字节或者更少），从而确保了快速过渡。

[0049] 此外，当第二处理器有可能在随后被重用时，不是所有的寄存器数据都需要重新发送第二次。控制器将相应地指示迁移（步骤 112、114）捕捉第一处理器状态的一个子部分的状态数据。

[0050] 一旦迁移（步骤 114）完成，就可以在第二处理器处恢复执行（步骤 130）。

[0051] 本发明实施方式允许安全地摆脱数据高速缓存，因为高存储器带宽允许非常短的连接。然而，在实践当中不一定需要这样。

[0052] 例如，进一步的改进是通过除状态数据之外还迁移由第一处理器使用的高速缓存来获得的。因此，第一处理器的“环境”在第二处理器处被更完整地重新创建。更明确地说，存储被第一处理器最频繁访问的数据拷贝的存储器高速缓存 (memory cache) 的数据可以被写入（步骤 116）由第二处理器使用的存储器高速缓存中。为了一致性，同步将优选地与状态数据所使用的相同。

[0053] 所迁移的存储器高速缓存一般是最小的，即，第一处理器使用的 L1 存储器高速缓存。更有效地，可以依赖一个存储器高速缓存（L1），而不是通常的 L1-L3 三件套。实际上，在本发明实施方式中实现的高存储器带宽允许安全地除去数据高速缓存。此外，依赖于例如“硅导孔”或 TSV（Through Silicon Via, TSV）技术允许非常短的连接。顺便提一句，空出来的面积可以用于更多的内核。

[0054] 此外，可以象高速缓存中的行 (line) 一样来对待计算内核状态本身。计算内核具

有千字节的状态。由此，执行可以在第一处理器处被挂起，而内核寄存器与 L1 高速缓存可以被移动到更适当的内核。

[0055] 接下来，在中断在第一处理器处的执行之后，第一处理器的处理器状态可以一直维持到其它的状态数据被加载到其中(步骤 112')。这可以是例如缺省的行为。同样，控制器可以确保在适当的时间内而且如果必要的话经增量文件更新第一处理器处的状态。

[0056] 如果没有指示新状态要加载到其中的话，有可能第一处理器可以在暂时性延迟(temporization delay)之后被关闭(步骤 140)。由此，优化了功耗。

[0057] 令人感兴趣的是，要中断和恢复的计算任务的最适合规模依赖于例如监视算法的反应性。找到最有效的规模依赖于上下文、处理器的个数、体系结构、处理的类型、任务、输入数据的本质，等等。假定有效的监视策略是可以获得的，那么本发明有利地是以单个执行线程的规模实现的。将相应地指示在挂起线程在第一处理器处的执行(步骤 120)之后继续线程在第二处理器处的执行(步骤 130)。

[0058] 线程的执行可以看作运行任务的一部分。对于大部分操作系统，线程包含在过程中，而且在同一过程中可以出现数个线程，共享诸如存储器之类的资源，随过程有所不同。从多线程技术已知，如何使单个处理器在不同的线程之间切换(就象在多任务中一样)。类似地，在这里，本发明实施方式使一个线程在一个处理器处被停止。然而，代替在同一个处理器处恢复该线程，该线程在另一个处理器处继续。

[0059] 这一点在图 3 中说明。y 轴是时间线。圆圈内的两条平行线指示各自的处理器，P1 和 P2。如从图中暗示的那样，线程属于给定的过程 p。首先，线程 t 在 P1 开始并执行。在某个时候，控制器(未示出)检测到 P1 正在使用更靠近 P2 的资源并且决定中断线程 t，处理器 P1 处于状态 s1。如前面所讨论的，然后，线程 t 从状态 s1 开始，在 P2 恢复。执行继续，直到线程终止，或者甚至是在 P2 被中断(处于状态 s2)以被导入回到 P1，如图 3 中所示的，并且如果上下文使之有利的话就这样。关于线程管理的细节可以例如在以上所引用的出版物，即“Adaptive Data Placement in an Embedded Multiprocessor Thread Library”，中找到。

[0060] 因而，本发明实施方式允许把计算状态移动到存储器，而不是让分层的高速缓存集合从主存储器移动数据。正如所说的那样，该方法的有些实施方式是在硬件控制之下执行的(控制器被体现为硬件电路)，对软件是透明的。

[0061] 图 4 是实现用于管理计算机化系统的处理资源的方法的一种实施方式的计算机系统的示意性表示。在这里，处理器 P1-PN 经合适的互连 B 连接到存储器 M，其中互连 B 包括总线 B 或者交叉开关，还有可能包括开关。它们又以别的方式连接到存储器 D (例如，硬盘)，并在控制器 C 的控制之下。在一种变体实施方式中，开关是分布式的，例如，每个内核或者内核子集一个开关。在还有另一种变体实施方式中，控制器 C 是互连的一部分。在所有情况下，控制器 C 都适当地耦合到处理器和存储器，以便诸如能够指示把执行从一个处理器移动到另一个。

[0062] 在所描绘的例子中，特定的输入数据 d 被物理地存储成比 P1 访问其的位置更靠近 P2 访问其的位置。在操作中，如果 P1 变成过于频繁地访问 d，那么控制器可能想把 P1 的状态移动到 P2，从而使得较少的数据被移动。

[0063] 图 5 是图 4 的一种变体实施方式，其中控制器 C1-CN 分布在多个处理器上，例如，

每个处理器一个控制器 Cn。在这里，控制器可以同时决定把处理器状态从一个处理器移动/接收到另一个处理器。因而，在 C1 执行的、基于监视的输入数据的测试可以导致把 P1 状态移动到 P2。然后，基于 P2 当前的活动性(状态)，C2 可以决定是否接受。例如，如果 P2 是不活动的，那么 C2 就确认把 P1 状态移动到 P2。类似地而且如所提到的那样，开关可以是分布式的。

[0064] 图 6 是图 4 的另一种变体实施方式，其中存储器现在包括至少两个存储器芯片 M1、…、MN，第一和第二处理器 P1、P2 中的每一个都是操作性互连到诸如 DIMM 的存储器的芯片 M1、M2 的处理器内核，控制器经合适的互连互连到处理器。

[0065] 更具体地说，在有些实施方式中，处理器 P1-PN 是各自 3D 存储器堆栈 M1-MN(例如，DIMM 存储器芯片)之上的四内核布置。但是，总体的原理保持相同：在操作中，如果 P1 的第一个给定的内核(让我们称之为 P11)变成频繁地访问给定的数据 d，控制器就可以把 P11 状态移动到 P2 的一个内核。由此实现的目标是把合理的计算量放得非常靠近 DRAM 并且采用 3D 堆栈，例如利用“硅导孔”(TSV)技术，从而，与真正分布式的系统一起，导致更快、更小而且消耗更少功率的包。

[0066] 为了说明，考虑以下例子建立。在这里，处理器内核是在各自的 DIMM 存储器缓冲上。一个应用关心例如数据库扫描，其中处理器必须检查很大一部分的存储器。操作的顺序可以例如如下：

[0067] - 首先，存储器扫描开始并且计数器监视存储器访问；

[0068] - 其次，“懒惰”迁移代码在 DIMM 之间开始(即，在指示中断在第一内核处的执行之前，控制器指示开始在第二处理器内核处加载状态数据)；以及

[0069] - 第三，控制器触发线程状态的迁移并且暂停在第一内核上的执行。

[0070] 在有些实施方式中，该方法至少部分地解决了存储器带宽问题，及存储器容量问题。此外，所述方法有些实施方式的核心原理本质上是可缩放的。而且，由于所述方法的有些实施方式使得有更少的高速缓存级，因此它们本质上更简单。

[0071] 最后，尽管在有些情况下可能需要修改 OS 和存储器控制器设计，但是在此提出的方法对至少一些应用是透明的。然而，修改 OS 本质上不是必需的。此外，修改存储器控制器是要包括附加的硬件，用于例如监视每个内核的访问和用于把执行从一个处理器迁移到另一个，并且当控制器体现为硬件电路时，这样做。但是，在所述方法的纯“软件”实现中，不需要修改。

[0072] 在软件实现中，控制器执行的指示操作可以以包括至少两个可编程处理器的系统上可以执行的程序实现。每个计算机程序都可以以高级(例如，过程性或者面向对象的)编程语言实现，或者如果期望的话可以用汇编或者机器语言；而且在任何情况下，语言都可以是编译或解释语言。作为例子，合适的处理器包括通用和专用微处理器。在另一种变体实施方式中，控制器执行的指示操作可以存储在计算机程序产品上，该计算机程序产品有形地体现在机器可读存储设备中，用于让可编程处理器执行；而且，本发明的方法步骤可以由执行指令的可编程处理器执行，以便执行本发明的功能。在所有的情况下，本发明都包含结果产生的计算机系统。

[0073] 更一般地说，以上方法的实施方式可以在数字电子电路中，或者在计算机硬件、固件、软件或者其组合中实现。

[0074] 总的来说，处理器将从只读存储器和 / 或随机访问存储器接收指令和数据。适合有形地体现计算机程序指令与数据的存储设备包括所有形式的非易失性存储器，作为例子，包括半导体存储器设备，诸如 EPROM、EEPROM 和闪存存储器设备；磁盘，诸如内部硬盘和可拆卸磁盘；磁 - 光盘；及 CD-ROM 盘等。

[0075] 本发明有利地适用于大型计算机，拥有多达数以千计的处理器、高速互连和专用硬件。

[0076] 尽管本发明已经参考某些实施方式进行了描述，但是本领域技术人员将理解，在不背离本发明范围的情况下，可以进行各种变化而且可以替换等同物。此外，在不背离其范围的情况下，可以进行许多修改以使特定的情形或者材料适合本发明的教义。因此，本发明不限于所公开的特定实施方式，本发明将包括属于所附权利要求范围之内的所有实施方式。例如，在此所公开的中断 / 恢复原理可以在与线程级不同的另一个级别上实现。

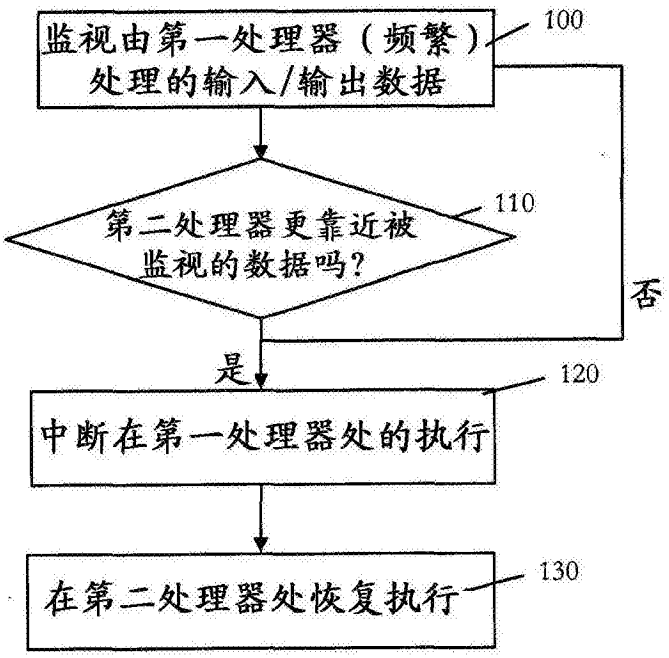

监视由第一处理器（频繁）处理的输入/输出数据
100
第二处理器更靠近被监视的数据吗?
110
否
是
中断在第一处理器处的执行
120
在第二处理器处恢复执行
130

图 1

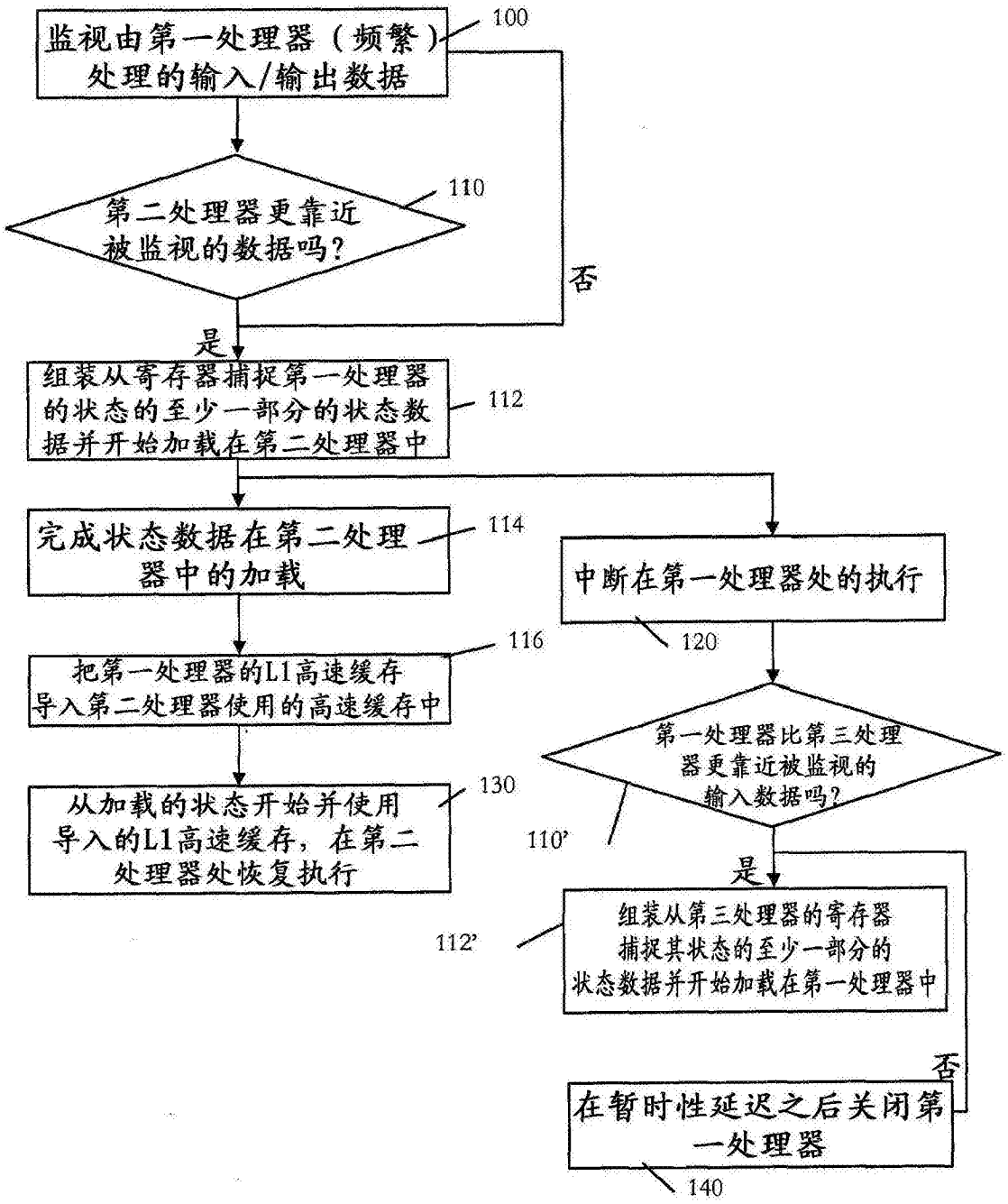

监视由第一处理器（频繁）处理的输入/输出数据
100
第二处理器更靠近被监视的数据吗？
110
否
是
组装从寄存器捕捉第一处理器的状态的至少一部分的状态数据并开始加载在第二处理器中
112
完成状态数据在第二处理器中的加载
114
把第一处理器的L1高速缓存导入第二处理器使用的高速缓存中
116
从加载的状态开始并使用导入的L1高速缓存，在第二处理器处恢复执行
130
中断在第一处理器处的执行
120
第一处理器比第三处理器更靠近被监视的输入数据吗？
110'
是
组装从第三处理器的寄存器捕捉其状态的至少一部分的状态数据并开始加载在第一处理器中
112'
否
在暂时性延迟之后关闭第一处理器
140

图 2

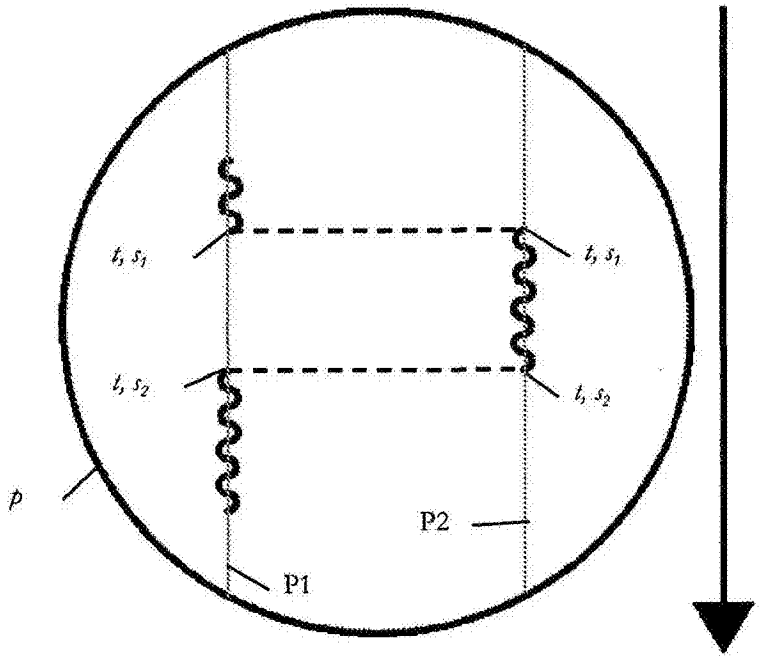


图 3

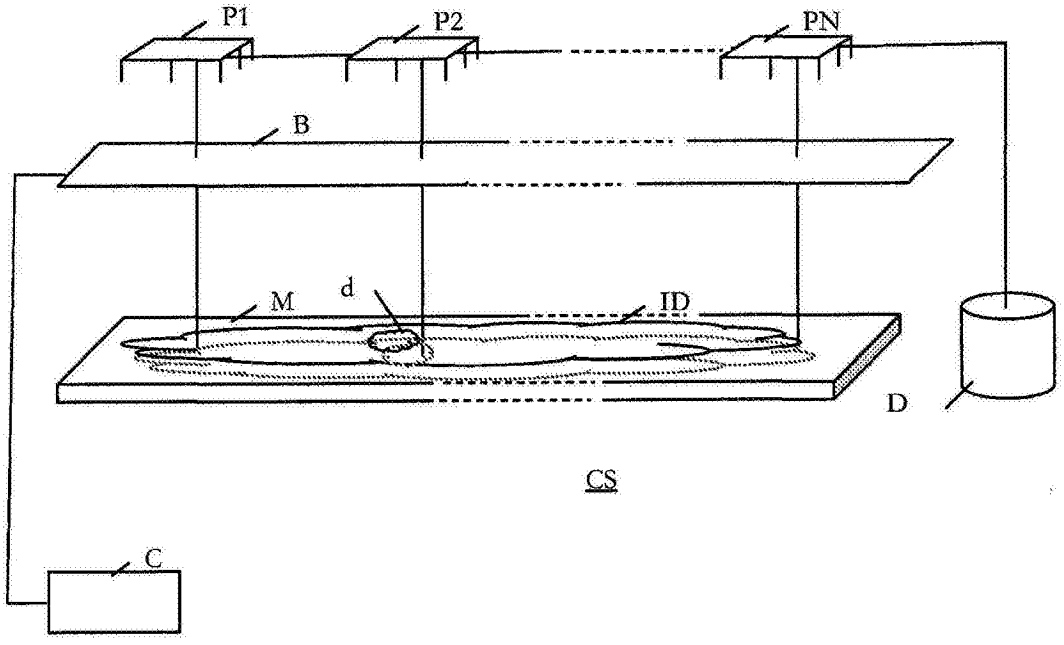


图 4

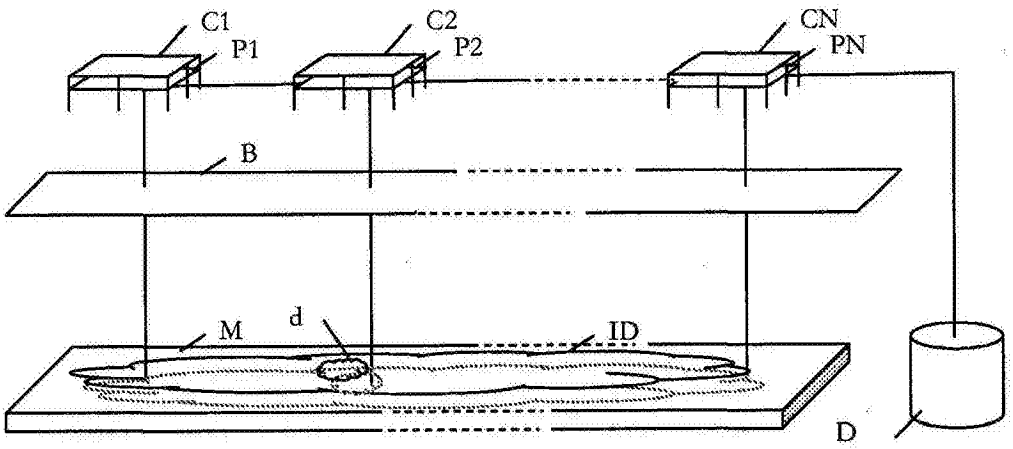


图 5

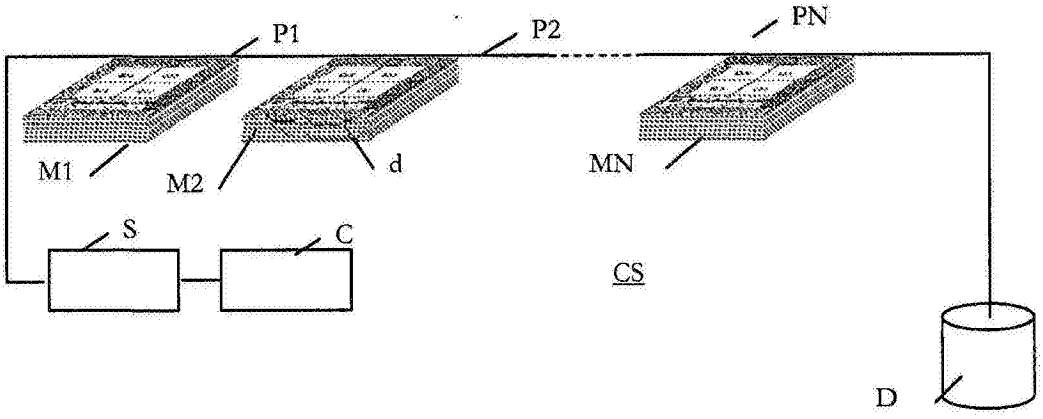


图 6