

(19) 中华人民共和国国家知识产权局



(12) 发明专利申请

(10) 申请公布号 CN 103959233 A

(43) 申请公布日 2014. 07. 30

(21) 申请号 201280045166. 4

(51) Int. Cl.

(22) 申请日 2012. 07. 23

G06F 7/60 (2006. 01)

(30) 优先权数据

61/535, 131 2011. 09. 15 US

(85) PCT国际申请进入国家阶段日

2014. 03. 17

(86) PCT国际申请的申请数据

PCT/US2012/047860 2012. 07. 23

(87) PCT国际申请的公布数据

W02013/039606 EN 2013. 03. 21

(71) 申请人 埃克森美孚上游研究公司

地址 美国德克萨斯州

(72) 发明人 K·豪根

(74) 专利代理机构 北京纪凯知识产权代理有限

公司 11245

代理人 赵蓉民

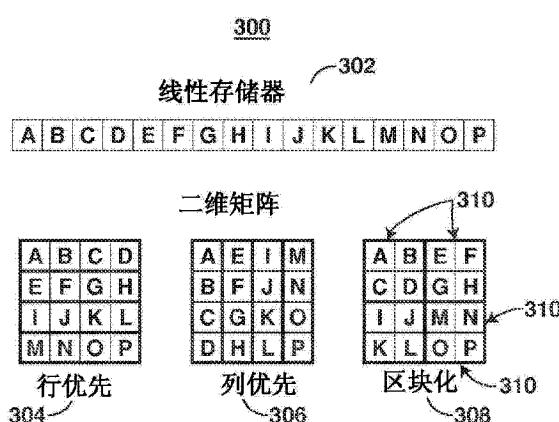
权利要求书2页 说明书15页 附图6页

(54) 发明名称

在执行 EOS 计算的指令受限算法中最优化矩阵和向量运算

(57) 摘要

本发明提供一种用于在执行 EOS 计算的指令受限算法中最优化矩阵和向量运算的系统和方法。该方法包括将与 EOS 稳定性方程和 EOS 分相方程相关的每个矩阵分成数个区块，其中该区块大小是非均匀的或均匀的。与 EOS 稳定性方程和 EOS 分相方程相关的每个向量可以被分成数个条带。区块和条带可以被存储在主存储器、高速缓存或寄存器中，并且与逐次代换和牛顿迭代相关的矩阵和向量运算可以使用区块和条带并行执行。



1. 一种在执行 EOS 计算的指令受限算法中最优化矩阵和向量运算的方法,其包括:
 - 将与 EOS 稳定性方程或 EOS 分相方程相关的每个矩阵分成数个区块,其中所述区块的大小是非均匀的或均匀的;
 - 将与所述 EOS 稳定性方程或所述 EOS 分相方程相关的每个向量分成数个条带;
 - 在主存储器、高速缓存或寄存器中存储所述区块和条带;以及
 - 使用所述区块和条带并行执行与逐次代换和牛顿迭代相关的所述矩阵和向量运算。
2. 根据权利要求 1 所述的方法,包括在主存储器、高速缓存或寄存器中顺序或交错存储区块或条带。
3. 根据权利要求 1 所述的方法,其中为了最小化单个硬件体系结构内的不同等级的存储器之间的数据传递而选择所述区块的大小。
4. 根据权利要求 1 所述的方法,包括基于与所述 EOS 计算相关的区块或条带大小展开循环。
5. 根据权利要求 1 所述的方法,包括将每个矩阵分成数个区块,其中所述矩阵被填充为允许使用均匀区块的大小。
6. 根据权利要求 1 所述的方法,包括通过特化展开循环,其中数个不同 EOS 算法针对具有不同的数个组分的混合物被生成。
7. 根据权利要求 1 所述的方法,包括 EOS 计算或与 EOS 计算相关的所述区块和所述条带内数据元素的单精度、双精度或混合精度向量化使用单指令多数据或单指令多线程向量指令。
8. 一种被适配用于在执行 EOS 计算的指令受限算法中的最优化矩阵和向量运算的计算机系统,所述计算机系统包括:
 - 处理器;以及
 - 有形机器可读存储介质,所述有形机器可读存储介质存储用于由所述处理器执行的机器可读指令,所述机器可读指令包括:
 - 代码,所述代码在由所述处理器执行时被适配为使所述处理器将与 EOS 稳定性方程和 EOS 分相方程相关的每个矩阵分成数个区块,其中所述区块大小是非均匀的或均匀的;
 - 代码,所述代码在由所述处理器执行时被适配为使所述处理器将与所述 EOS 稳定性方程和所述 EOS 分相方程相关的每个向量分成数个条带;
 - 代码,所述代码在由所述处理器执行时被适配为使所述处理器在主存储器、高速缓存或寄存器中存储所述区块和条带;以及
 - 代码,所述代码在由所述处理器执行时被适配为使所述处理器使用所述区块和条带并行执行与逐次代换和牛顿迭代相关的所述矩阵和向量运算。
9. 根据权利要求 8 所述的系统,其中所述处理器形成标量硬件体系结构或矢量硬件体系结构。
10. 根据权利要求 8 所述的系统,包括顺序或交错存储区块或条带的主存储器、高速缓存或寄存器。
11. 根据权利要求 8 所述的系统,包括代码,所述代码在由所述处理器执行时被适配为使所述处理器基于与所述 EOS 计算相关的区块或条带大小展开循环。
12. 根据权利要求 8 所述的系统,包括代码,所述代码在由所述处理器执行时被适配

为使所述处理器将每个矩阵分成数个区块，其中所述矩阵被填充为允许使用均匀区块的大小。

13. 根据权利要求 8 所述的系统，包括代码，所述代码在由所述处理器执行时被适配为使所述处理器为具有不同的数个组分的混合物生成数个不同 EOS 算法。

14. 根据权利要求 8 所述的系统，包括代码，所述代码在由所述处理器执行时被适配为使所述处理器使用单指令多数据或单指令多线程向量指令，执行单精度、双精度或混合精度 EOS 计算或与 EOS 计算相关的所述区块和所述条带内的数据元素的向量化。

15. 一种包括代码的非暂时性计算机可读介质，所述代码被配置为引导处理器：

将与 EOS 稳定性方程和 EOS 分相方程相关的每个矩阵分成数个区块，其中所述区块大小是非均匀的或均匀的；

将与所述 EOS 稳定性方程和所述 EOS 分相方程相关的每个向量分成数个条带；

在主存储器、高速缓存或寄存器中存储所述区块和条带；以及

使用所述区块和条带并行执行与逐次代换和牛顿迭代相关的所述矩阵和向量运算。

16. 根据权利要求 15 所述的非暂时性计算机可读介质，包括在主存储器、高速缓存或寄存器中顺序或交错存储区块或条带。

17. 根据权利要求 15 所述的非暂时性计算机可读介质，包括基于与所述 EOS 计算相关的区块或条带大小展开循环。

18. 根据权利要求 15 所述的非暂时性计算机可读介质，包括将每个矩阵分成数个区块，其中所述矩阵被填充为允许使用均匀区块的大小。

19. 根据权利要求 15 所述的非暂时性计算机可读介质，包括通过特化展开循环，其中数个不同 EOS 算法针对具有不同的数个组分的混合物被生成。

20. 根据权利要求 15 所述的非暂时性计算机可读介质，包括 EOS 计算或与 EOS 计算相关的所述区块和所述条带内的数据元素的单精度、双精度或混合精度向量化使用单指令多数据或单指令多线程向量指令。

在执行 EOS 计算的指令受限算法中最优化矩阵和向量运算

[0001] 相关申请的交叉参考

[0002] 本申请要求美国临时专利申请 61/535,131 的利益,该申请提交于 2011 年 9 月 15 日,标题为 OPTIMIZED MATRIX AND VECTOR OPERATIONS IN INSTRUCTION LIMITED ALGORITHMS THAT PERFORM EOS CALCULATIONS,在此通过引用将其全文合并于此。

技术领域

[0003] 本技术涉及具有最优化矩阵和向量运算的指令受限算法。具体地,本技术的示范实施例涉及用于执行状态方程(EOS)计算的系统和方法。

背景技术

[0004] 本节意图介绍可以与所披露技术的实施例相关的现有技术的各种方面。本讨论相信可以有助于提供框架从而促进更优理解所披露技术的具体方面。因此应理解本节从这点阅读,而不必认可现有技术。

[0005] 指令受限算法可以在各种工业中发生。例如指令受限算法可以在领域例如 3D 图形分析、加密、数据挖掘、压缩、信号处理、图像处理、链式法则评估、数值法例如有限元和有限体积分析、地震模式识别和状态方程(EOS)计算中发生。

[0006] EOS 计算可以用来模拟可以对储层性能具有显著效果的相行为。石油混合物可以流过多孔介质的速率受相态数、每个相态的粘度和每个相态的密度影响。一般地,相行为和相性质是温度、压力和成分的函数。在一些情况下,与相行为相关的成分效果是微弱的并可以忽略。这可以伴随称为黑油的石油流体发生。模拟含黑油的储层可以称为黑油仿真。

[0007] 在其他情况下,成分效果可以被解释。为解释成分效果,石油工业通常使用状态方程(EOS)。将具有成分效果的储层建模可以称为成分仿真。数个先进开采机制可以依靠相行为的成分效果。

[0008] EOS 计算可以使成分仿真比黑油仿真可观地更低。EOS 计算通常涉及确定相态数和每个相态的成分。尽管单个 EOS 计算执行的消耗低廉,但其可以在储层仿真的过程中重复数十亿次。因此 EOS 计算可以消耗总仿真时间的多于 50%。此外 EOS 计算是计算密集的并且其成本可以随着混合物中组分数目增加而迅速提高。

[0009] 为改进计算速度,并行储层仿真器可以使用。当使用并行化时,巨大问题被分成较小子问题,并然后在数个处理核心之间分配。子问题可以不是独立的,并且核心可以通信从而将其工作同步。核心可以通过共享存储器或通过高速网络同步。在并行计算环境中,存储器带宽和网络通信通常是速度限制因素。

[0010] D. Voskov 和 H. Tchelepi,“Tie-Simplex Based Mathematical Framework for Thermodynamical Equilibrium Computation of Mixtures with an Arbitrary Number of Phases”, Fluid Phase Equilibria, 第 283 卷, 2009, pp. 1-11 陈述基于连结线的并行化方法在成分流仿真中改进相行为表示的准确度和状态方程(EOS)计算的效率。对于不能融合的成分仿真,陈述该技术试验成分空间自适应列表以避免多余 EOS 计算中的大部分。然而,

矩阵和向量运算没有最优化。

[0011] C. Rasmussen 等人,“Increasing the Computational Speed of Flash Calculations with Applications for Compositional, Transient Simulations”, SPE Reservoir Evaluation and Engineering, 第 9 卷第 1 册, 2009, pp. 32-38 陈述在常规闪存计算中, 仿真时间的大部分花费在稳定性分析上。陈述该技术判定在其调整时绕过稳定性分析, 并且不在执行 EOS 计算的指令受限算法中最优化矩阵和向量运算。

[0012] E. Hendriks 和 A. Van Bergen, “Application of a Reduction Method to Phase-Equilibria Calculations”, Fluid Phase Equilibria, 第 74 卷, 1992, pp. 17-34 陈述对于特定热动力模型, 求解涉及混合物相平衡的一组非线性方程的方程数可以减少。问题大小和计算工作可以通过变量变换来减小。另外, 适当描述混合物相行为的减少变量的最小数目可以确定。然而在执行 EOS 计算的指令受限算法中的矩阵和向量运算没有被最优化。

发明内容

[0013] 本技术的示范实施例提供在执行 EOS 计算的指令受限算法中最优化矩阵和向量运算的方法。与 EOS 稳定性方程和 EOS 分相方程相关的每个矩阵可以被分成数个非均匀或均匀大小区块。与 EOS 稳定性方程和 EOS 分相方程相关的每个向量可以被分成数个条带。区块和条带可以在主存储器、高速缓存或寄存器中组织, 并且与逐次代换和牛顿迭代相关的矩阵和向量运算可以使用区块和条带并行执行。

[0014] 本技术的示范实施例提供系统, 其包括处理器和存储用于由处理器执行的机器可读指令的有形机器可读存储介质, 该机器可读指令包括在由处理器执行时被适配为使处理器将与 EOS 稳定性方程和 EOS 分相方程相关的每个矩阵分成数个区块的指令, 其中该区块大小是均匀的或非均匀的。代码在由处理器执行时可以被适配为使处理器将与 EOS 稳定性方程和 EOS 分相方程相关的每个向量分成数个条带, 并且在主存储器、高速缓存或寄存器中存储区块和条带。另外, 代码在由处理器执行时可以被适配为使处理器使用区块和条带并行执行与逐次代换和牛顿迭代相关的矩阵和向量运算。

[0015] 本技术的示范实施例提供非暂时性计算机可读介质, 其包括经配置引导处理器将与 EOS 稳定性方程和 EOS 分相方程相关的每个矩阵分成数个区块的代码, 其中该区块大小是非均匀的或均匀的。与 EOS 稳定性方程和 EOS 分相方程相关的每个向量分成数个条带。区块和条带可以在主存储器、高速缓存或寄存器中存储, 并且与逐次代换和牛顿迭代相关的矩阵和向量运算可以使用区块和条带并行执行。

附图说明

[0016] 本技术的优点可以紧接着浏览实施例的非限制示例的以下详细描述和附图变得更明显, 在该附图中:

[0017] 图 1 是过程流程图, 其概述根据本技术的实施例在执行 EOS 计算的指令受限算法中最优化矩阵和向量运算的方法;

[0018] 图 2 是示出根据本技术的实施例的存储器层次的图示;

[0019] 图 3 是图解根据本技术的实施例的如何使用三个不同存储格式将二维矩阵映射

到线性存储器的图示；

[0020] 图 4 是示出根据本技术的实施例的均匀区块和非均匀区块的图示；

[0021] 图 5 是图解根据本技术的实施例的指令的数目如何可以通过 SIMD/SIMT 向量化被减少的图示；

[0022] 图 6 是概述根据本技术的实施例的执行 EOS 计算的方法的过程流程图；

[0023] 图 7 是概述根据本技术的实施例的用于 EOS 计算的稳定性和分相算法的迭代性质的过程流程图；

[0024] 图 8 是概述根据本技术的实施例的执行逐次代换和牛顿算法的方法的过程流程图；

[0025] 图 9 是概述根据本技术的实施例的成分如何可以在牛顿迭代内被更新的方法的过程流程图；

[0026] 图 10 是根据本技术的实施例的计算机系统的框图，该计算机系统可以用来在指令受限算法中最优化矩阵和向量计算。

具体实施方式

[0027] 在以下详细描述节中，具体实施例作为示例描述。然而，就以下描述对具体实施例或特别使用特定来说，其意图仅用于示范目的并简单提供示范实施例的描述。因此本技术不限于在下面描述的实施例，但相反这样的技术包括落入附加权利要求的精神和保护范围内的全部替换、修改和等效。

[0028] 起初，并为容易参考，阐述用于本申请的某些术语及其用于该上下文的意义。就本文使用的术语不在下面定义来说，应给予其如在至少一个印刷出版物或已提交专利中反映的本领域技术人员给予的最广泛定义。

[0029] 术语“单元”指代面的集合，或暗示地定义面的节点的集合，其中该面一起形成封闭体积。另外术语“面”指代形成表面的点的任意集合。

[0030] 术语“通信受限”指代算法执行速度由速度限制的时候，处理核心能够通过共享处理器或高速网络在该速度通信并将处理核心的工作同步。

[0031] 术语“存储器受限”指代算法执行速度由速率限制的时候，数据在存储器和 CPU 之间在该速率移动。

[0032] 术语“指令受限”指代算法执行速度由速率限制的时候，指令由 CPU 在该速率处理。

[0033] 术语“成分储层仿真”指代用来仿真需要获知储层的至少一部分中成分改变的开采过程的仿真。例如，成分仿真可以有助于研究(1)易挥发油或气体冷凝物储层的耗尽，其中相成分和性质随着压力低于气泡压力或露点压力显著变化，(2)非平衡气体(干燥或富集的)喷射到黑油储层，从而由蒸发成更可移动的气相或由通过彻底的(单接触)或动态的(多接触)混溶性将油移动，以及(3)将 CO₂ 喷射到油储层，从而由混溶驱动(miscible displacement)并由油粘度减小和油溶胀(oil swelling)将油移动。

[0034] 由于大量迭代相平衡计算和大量流动方程求解，因此使用 EOS 描述多组分流体混合物的相行为的成分储层仿真可以是耗时的。在 EOS 计算中求解的方程的数目与流体中组分数目成比例。由于储层流体可以含有数百种纯组分，因此执行其中所有储层组分在计算

中使用的成分仿真可以是不实际的。因此希望将在描述流体混合物中使用的组分的数目保持最小。

[0035] 术语“计算机部件”指代计算机相关实体、硬件、固件、软件、其组合，或执行中的软件。例如，计算机部件可以是但不限于在处理器上运行的过程、处理器、对象、可执行过程、执行的线程、程序和计算机。一个或多个计算机部件可以位于执行的过程或线程内，并且计算机部件可以定位在一台计算机上或在两台或更多台计算机之间分布。

[0036] 术语“状态方程”或“EOS”指代方程，该方程代表包括烃的任何流体的相行为。在储层仿真中，状态方程可以仅用于烃相，并且经验关系式可以用来描述液相。EOS 可以在基于计算机的建模和仿真技术中用来创造模型，以便估计所关注储层中烃流体的性质和 / 或行为。一旦 EOS 模型定义，则其可以用来计算储层的石油流体的广泛的一系列性质，例如气 - 油比(GOR)或冷凝物 - 气体比(CGR)、每个相态的密度、体积因数和压缩率、热容量和饱和压力(泡点或露点)。因此 EOS 模型可以经求解获得在给定温度的饱和压力。此外 GOR、CGR、相密度和体积因数是 EOS 模型的副产物。输送性质例如传导率、扩散率或粘度可以从性质例如流体成分推导，该性质从 EOS 模型获得。

[0037] 术语“非暂时性计算机可读介质”、“有形机器可读介质”等指代参与向处理器提供指令以便执行的任何有形存储。这样的介质可以采取许多形式，包括但不限于非易失性介质和易失性介质。非易失性介质包括例如 NVRAM 或磁或光盘。易失性介质包括动态存储器例如主存储器。计算机可读介质可以包括例如软盘、软磁盘、硬盘、磁带，或任何其他磁介质、CD-ROM、任何其他光介质、RAM、PROM 和 EPROM、FLASH-EPROM、固态介质如全息存储器、存储器卡或任何其他存储器芯片或盒式磁带，或计算机可以从其读取的任何其他介质。在计算机可读介质配置为数据库时，理解数据库可以是任何类型的数据，例如关系的、分层的、面向对象的，等等。因此本技术的示范实施例可以认为包括有形存储介质或有形分布介质和现有技术认可的等效与后继介质，实施本技术的软件实施在这些介质中存储。

[0038] 术语“等待时间”指代在系统中经历的时延的测量值。

[0039] 术语“相行为”指代流体混合物如何将作为压力、温度和成分的函数分成两个或更多个相态。混合物的相态可以是固体、蒸汽、液体或超临界。

[0040] 术语“相态”指代可以从非均匀混合物机械地分离的物质的化学或物理统一量。其可以由单种物质或多种物质的混合物构成。一般地，物质的四相是固体、液体、气体和等离子体。然而术语“相态”也用来描述物质的其他性质或状态，例如非混溶液体的分离层、胶体物质或混合物，以及无定形固体。在烃生产中，含水的(水)、液体(油)和蒸汽(气体)相经常存在。

[0041] 术语“性质”指代数据，该数据表示与基于每一元素的不同拓扑元素相关的特性。一般地，性质可以是包括整数和浮点数型等的任何计算值类型。此外，性质可以包括值向量类型。性质可以仅对几何对象的元素的子组有效。性质可以用来将对象的几何形状着色。术语“性质”也可以指代涉及对象的特性或已存储信息。适当定义的应用对计算机科学领域技术人员是直观的。

[0042] 术语“特化”指代为一组特定输入参数计算机程序或算法的版本。

[0043] 术语“线程”一般指代使用特别输入数据执行特别程序的实例。并行线程可以使用不同处理引擎同时执行，允许处理工作在给定量的时间内完成。

[0044] 总述

[0045] 尽管为解释简便,图解方法学示出并描述为一系列方框,但认识到由于一些方框可以在不同顺序中发生和 / 或与源自示出并描述的其他方框同时发生,因此方法学不受方框的顺序限制。此外,少于全部的图解方框可以需要实施例子方法学。方框可以组合或分离为多个组成。此外,另外和 / 或可替换的方法学可以采用另外的未图解方框。尽管附图图解各种串行发生的行为,但认识到各种行为可以连续发生、基本并行发生和 / 或在时间中基本不同点发生。

[0046] 实施例提供用于在任意硬件体系结构上的指令受限算法中最优化矩阵和向量运算的方法。计算机算法可以含有各种矩阵和向量运算。任何计算机算法的速度由三个可能瓶颈限制:指令吞吐量、存储器吞吐量和在集群情况下的通信吞吐量。最科学的算法由在其上处理它们的硬件来存储器或通信限制。然而 EOS 计算通常是指令受限的,意味着 EOS 算法的最终速度可以由硬件能够在其执行单个运算的速率确定。结果,最优化可以涉及在其中计算发生的高速缓存和硬件寄存器之间有效移动数据。该数据可以包括矩阵和向量数据。

[0047] 图 1 是过程流程图,其概述根据本技术实施例在执行 EOS 计算的指令受限算法中优化矩阵和向量运算的方法 100。在方框 102 处,每个矩阵被划分成多个区块(tile)。区块通常可以被描述为从原始的较大矩阵形成的较小子矩阵。图 3 进一步描述将矩阵分成区块。此外,每个矩阵可以与具有非均匀或均匀区块大小的 EOS 稳定性方程或 EOS 分相方程相关。在方框 104 处,每个向量都被分成多个条带(strip)。与区块类似,条带通常可以被描述为从原始的较大向量形成的较小子向量。每个向量可以与 EOS 稳定性方程或 EOS 分相方程相关。

[0048] 在方框 106 处,区块和条带可以被组织在计算机存储例如主存储器、高速缓存或寄存器中。在区块和条带中含有的数据可以被操作之前,运算数(operand)中的至少一个被传递到硬件寄存器。非必需的数据传递可能减慢 CPU 执行速度。为使数据传递更加有效,由于高速缓存存储器与主存储器相比可以更快并且具有更低的等待时间,因此现代硬件体系结构可以在主存储器和寄存器之间使用高速缓存存储器作为缓冲器。在方框 108 处,可以使用区块和条带并行执行矩阵和向量计算,其中所有矩阵和向量计算以区块运算和条带运算的术语来表述。进一步地,在 EOS 计算中,可以使用区块和条带来并行执行与逐次代换和牛顿迭代相关的矩阵和向量运算。图 6-9 进一步描述逐次代换和牛顿迭代。并行化可以用来加速单个区块单条带运算或同时执行多个区块多条带运算。

[0049] 图 2 是示出根据本技术的实施例的存储器层次的图示 200。如图示 200 所说明的,现代 CPU 通常具有多级高速缓存层次 202。图示 200 具有在 204 的等级 1、在 206 的等级 2 和在 208 的等级 3 这三个高速缓存的等级。在 204 的高速缓存等级 1 最接近寄存器 210,并且高速缓存等级 3 最接近主存储器 212。因此,越接近存储数据的寄存器 210,速度 214 将提高并且等待时间 216 降低。在主存储器 212 和多级高速缓存层次 202 之间、在多级高速缓存层次 202 自身内或者在多级高速缓存层次 202 和寄存器 210 之间的每次数据传递花费有限量的时间。非必需的数据传递可能导致硬件浪费若干计算机时钟循环来等待数据从而进行处理。在现代 CPU 上,与小问题例如 EOS 计算相关的数据可以完全容纳在高速缓存存储器级内。因此在现代 CPU 上的数据传递问题通常在高速缓存存储器级和寄存器之间发生。相反,现代 GPU 通常具有显著较少量的可用高速缓存存储器。因此在现代 GPU 上的数

据传递问题通常在高速缓存存储器级和主存储器之间发生。

[0050] **最优化技术**

[0051] 致力于指令吞吐量的最优化技术包括区块化、每区块最优化、特化、条带化和向量化。因为尽管矩阵是二维数据结构,但矩阵通常被存储在一维存储器中,所以有效实施这些最优化技术可能是挑战性的。矩阵在存储器中所依照存储的方式影响执行矩阵运算所需要的数据传递的数目。如上面讨论,非必需的数据传递可能导致硬件浪费若干计算机时钟循环来等待有待处理的数据。

[0052] 一般具有两类的可用计算机体系结构,标量和向量。标量体系结构可以每次对单个数据元素运算,而向量体系结构可以同时对若干数据元素运算。在各种计算机和移动装置中发现的中央处理单元(CPU)是标量硬件体系结构的示例。在计算机显卡上发现的图形处理单元(GPU)是向量硬件体系结构的示例。两类体系结构都由工作可以在其间划分的若干核心构成。工作的划分可以通过将工作分成称为线程的较小子任务来实现。

[0053] 图3是示出根据本技术的实施例的二维矩阵304-308如何使用三个不同存储格式映射到线性存储器302的图示300。存储矩阵的一个普遍方法是将数据行相互挨靠放置。这称为行优先存储格式并在行优先矩阵304中示出。可替换地,人们可以将数据列相互挨靠放置,从而获得在列优先矩阵304示出的列优先存储格式。这两种存储格式在应用到各种计算时可能在硬件寄存器和高速缓存存储器之间产生大量非必需的数据传递。数据传递问题可以通过将矩阵分成称为区块的较小子矩阵来解决,这些区块在区块化矩阵308示出。单个区块310的大小被选择用于确保每个算术运算中的至少一个运算数以及与区块运算相关的中间结果保留在寄存器中。这样,在寄存器和高速缓存之间的数据传递可以最小化,结果,CPU循环的较大部分可能花费在计算上。矩阵可以被再分成均匀或非均匀区块。使用非均匀区块可以允许任何大小的矩阵使用区块化存储格式。

[0054] 图4是示出根据本技术的实施例的均匀区块和非均匀区块的图示400。矩阵402含有每个是相同大小的在参考号404处的区块。因此矩阵402的区块是均匀区块。矩阵406含有在参考号408处的区块、在参考号410处的区块和在参考号412处的区块。在参考号408处的区块、在参考号410处的区块和在参考号412处的区块大小不同。因此矩阵402的区块是非均匀区块。具有均匀区块的矩阵和具有非均匀区块的矩阵可以被使用在本技术中。此外,区块可以按照顺序或交错的形式被组织或存储在主存储器、高速缓存或寄存器中。如本文所讨论的条带也可以按照顺序或交错形式被组织或存储在主存储器、高速缓存或寄存器中。进一步地,可以使用标量硬件体系结构或向量硬件体系结构来实现区块。

[0055] 无论均匀或非均匀,区块的大小都可以被选择以便最小化单个硬件体系结构内的存储器层次的不同等级之间的数据传递。当区块是均匀的时,矩阵大小可以被约束到区块大小的整数倍。通过用伪元素将矩阵填充至允许使用均匀区块的大小,可以为任何矩阵实现整数倍的矩阵大小。然而,除非矩阵大小显著大于区块大小,否则填充矩阵可能显著增加矩阵运算的计算时间。在此情况下,因为可以不使用填充将矩阵分成区块,所以使用非均匀区块可以消除这种计算时间上的增加。

[0056] 使用区块允许基于每区块的最优化。换言之,每个区块最优化促进使用分治途径,其中最优化矩阵运算可以被减少到运算少数区块运算。例如, EOS特定区块运算可以被用来计算逸度导数并用于在牛顿算法中构建雅可比矩阵。牛顿算法也使用线性求解器。在EOS

计算中,雅可比矩阵是稠密的使得大多数元素非零。求解具有稠密雅可比矩阵的线性方程组的两个普遍算法是均可以按照区块运算表述的高斯消去和上 - 下(UL)分解。每区块最优化可以减小循环开销和跳跃开销。循环开销指代与每个循环相关的计算开销。该循环开销可以是在循环的每个迭代期间必须被估计从而确定循环的随后迭代是否发生的状况的结果。同样,跳跃开销指代在计算机程序遇到告诉其从代码中的一行跳跃到另一行的指令时的相关计算开销。跳跃开销通常作为函数调用、循环迭代的结果发生,或在执行路径取决于一些测试的输出结果(分支)时发生。这样的跳跃指令可以可观地减慢执行速度,除非跳跃开销小于执行跳跃之间的代码花费的时间。

[0057] 为了减小循环开销和跳跃开销,现代编译器如可能则可以“展开”循环。当展开循环时,编译器可以估计控制循环迭代的数目的状况,并将循环转变成相应的非循环指令。例如,在 EOS 运算中,循环迭代的数目可以由混合物中的组分的数目来控制。通常,混合物中的组分的数目对于编译器来说是不可用的,从而防止循环展开发生。相反地,与 EOS 计算相关的区块可以具有预定大小,并且已经与区块运算相关并且基于区块大小的每个循环可以被完全地展开。相似地,在 EOS 计算中,已经与条带相关并且基于条带大小的每个循环可以被完全地展开。

[0058] 另一最优化技术可以使用寄存器变量。在使用数据结构例如向量和矩阵时的普遍编程实践是使用指向结构中的第一数据元素的地址的指针。使用该地址作为参考,算法获知如何访问其他数据元素。按照这种方式使用指针,尤其是在编译器采用多个指针可以指向相同的数据地址时,可能导致硬件寄存器和高速缓存存储器之间的非必需的数据传递。为了避免数据不一致,在寄存器中修改的任何数据元素可以被拷贝到高速缓存存储器从而确保每个指针具有到最近更新的访问。相似地,数据元素每当有待运算时其可以从高速缓存存储器中被拷贝,从而确保寄存器变量更新。过量的数据传递可以通过为中间计算使用寄存器变量或者保存输入数据的拷贝来避免。因此寄存器变量中的数据不可由指针访问,并且数据不必再在寄存器和高速缓存存储器之间被拷贝。

[0059] 用特化来最优化涉及做出最优化算法的若干版本从而调节取决于未知区块数目的展开循环,其与取决于区块大小的循环相反。如本文所讨论的,由于区块大小是编译器已知的,因此基于区块大小的循环可以被展开。然而,区块的数目可以是未知的。因此,取决于区块的数目的循环不可以被展开。特化为所关注的每个数目的区块提供算法。每个算法版本具有已知数目的区块,该已知数目的区块允许循环基于区块的数目被展开。特化还应用到与向量运算相关的循环并可以是自动使用模板。例如,通过特化展开循环可以在为具有不同数目组分的混合物生成数个不同 EOS 算法时发生。

[0060] 由于向量是一维数据结构,因此区块化不应用到向量。为了减少与向量相关的循环和跳跃开销,向量可以被分成条带。使用条带促进循环展开并且促进按照与区块相同的方式使用寄存器变量。例如,循环可以基于条带大小被展开。如本文所讨论的,条带利用涉及向量运算、从向量化分离并且不同于向量化的各种算法。

[0061] 单指令多数据(SIMD)和单指令多线程(SIMT)向量化也可以最优化指令受限算法。SIMD 指令对多个数据元素同时运算。大多数现代 CPU 实施经常称为向量扩展的 SIMD 指令。然而关于 GPU,具有 SIMD 和 SIMT 两种体系结构。SIMD 指令可以由单处理核心执行,而 SIMT 指令由多处理核心执行。SIMD 和 SIMT 指令都可以用来加速指令受限算法。

[0062] 图 5 是示出根据本技术的实施例的如何可以通过 SIMD/SIMT 向量化来减少指令的数目的图示 500。每个阵列 502–508 代表可以对其运算的数据段。数目鉴别对单个数据元素运算的指令。双精度标量指令在参考号 502 处示出，并且双精度向量指令在参考号 504 处示出。单精度标量指令在参考号 506 处示出，并且单精度向量指令在参考号 508 处示出。例如，假设每个向量的大小是 128 位。相同运算可以被执行在阵列中的每个元素上。使用双精度向量，每个运算对两个数据元素运算，将指令数减少到原来的二分之一。单精度向量含有数据元素数目的两倍，并且指令数减少到原来的四分之一。因此数据元素的单精度、双精度或混合精度向量化可以与 SIMD 或 SIMT 向量指令一起使用。进一步地，单精度、双精度或混合精度向量化可以被用来并行执行 EOS 计算，其中每个 EOS 计算可以对应于不同混合物。

[0063] 在指令受限的算法中，从双精度切换到单精度不会加速计算。尽管数据元素的大小被减小，但指令的数目保持相同。利用单精度涉及使用向量指令。因为单精度向量在与双精度向量比较时保存两倍之多的元素，所以加速随着单精度向量指令发生。因此每个指令对两倍数目的数据元素运算。

[0064] SIMD/SIMT 向量化可以被用来加速单个区块和条带运算。在该情况下，区块和条带可以是 SIMD/SIMT 向量长度的整数倍。因此，该方法用于短 SIMD/SIMT 向量。可替换地，SIMD/SIMT 向量化可以被用于并行执行若干区块和条带运算。在该情况下，区块和条带可以被交错以使每个 SIMD/SIMT 向量指令访问源自每个区块或条带的单个元素。因为由每个 SIMD/SIMT 向量指令访问的数据元素属于分离的区块或条带，因此 SIMD/SIMT 向量化与并行的区块和条带运算不会对向量或条带大小添加任何约束，并可以与任意长度的 SIMD/SIMT 向量一起使用。这样，与 EOS 计算相关的区块和条带或者与 EOS 计算相关的区块和条带内的数据元素的单精度、双精度或混合精度向量化可以使用单指令多数据或单指令多线程向量指令。大多数现代 CPU 的向量寄存器可以保存两个双精度或四个单精度元素，或者甚至可以保存四个双精度或八个单精度元素。

[0065] 示例

[0066] 本文所描述的各种最优化可以用不同方式来组合，并可以针对算法可以在其上运行的具体硬件被制定。为了描述的目的，三个示例被说明。然而，这些示例用于描述目的并非意在限制本技术。同样，第一示例可以为没有 SIMD 指令的硬件而设计，第二示例可以为具有短 SIMD 向量的硬件而设计，并且第三示例可以为具有任意长度的 SIMD 向量的硬件而设计。

[0067] I. 在第一示例中，硬件可以不支持 SIMD 指令。所有矩阵可以被组织成区块，并且区块内的数据可以使用如本文所描述的行优先、列优先或区块化存储格式。矩阵内的区块可以被顺序地组织或存储在主存储器、高速缓存或寄存器中。区块大小可以用尽可能有效地使用寄存器的方式来选择。当前，通常的台式和工作站 CPU 可以具有 16 个寄存器，因此三乘三和四乘四是合理的区块大小。使用均匀区块大小在组分的数目中引入粒度。例如，使用每 n 个的区块大小可以将组分的数目驱动到 n 的倍数。对于其中组分数目不是 n 的倍数的混合物，添加痕量的伪组分可以允许组分总体是 n 的倍数。为避免过多填充，即使在使用具有大量寄存器的硬件时，区块大小仍可以保持较小。

[0068] 可替换地，通过将矩阵分成不同大小的区块，填充可以完全地被避免。执行的区块

运算可以包括线性代数运算如矩阵 - 矩阵乘法、矩阵 - 向量乘法和矩阵求逆。此外，基于最优化区块运算使用线性求解器可以求解各种线性方程组。

[0069] 此外，在没有 SIMD 指令的硬件上使用时，向量可以被组织成条带。条带长度可以匹配区块大小。例如，使用每 n 个的区块大小意味着使用 n 的条带长度。向量运算可以被表达为一系列条带运算。这允许如本文所描述的部分展开对应的循环。其他循环可以使用特化来展开。然而在线性求解器内，一些循环可以对于编译器来说过于复杂而难以展开。这些循环可以被人工展开或通过使用模板而展开。

[0070] II. 在第二示例中，硬件可以支持在短向量上的 SIMD 指令。因此 SIMD 指令可被用来自通过如本文所讨论的向量化来加速单个区块和条带运算。这意味着涉及若干数据元素的计算可以被并行执行，并且单精度 SIMD 向量和双精度 SIMD 向量都可以被使用。所有矩阵可以被组织成区块，并且区块内的数据可以使用行优先或列优先存储格式。可以按照尽可能有效地使用寄存器的方式来选择区块大小。由于向量化用来加速单个区块运算，因此即使区块大小是非均匀的，区块大小仍可以经选择是向量长度的整数倍。实施 SIMD 向量的通常台式和工作站 CPU 可以保持两个双精度元素或四个单精度元素，或者甚至是四个双精度元素或八个单精度元素。此外，条带长度也可以是向量长度的整数倍并可以被选择用于而匹配区块大小。

[0071] 在第二示例中，区块运算可以被重设计为按照 SIMD 指令工作。除了矩阵 - 矩阵乘法，矩阵向量乘法和矩阵求逆运算以外，计算矩阵转置的算法也可以被实施。除了使用基于 SIMD 的区块运算以外，线性求解器可以与第一示例几乎相同。然而对于第二示例，一个主要区别是在每个区块内的 SIMD 向量对齐必须被考虑。水平 SIMD 向量对齐意味着 SIMD 向量中的所有数据元素属于区块中的相同行。相似地，垂直 SIMD 向量对齐意味着 SIMD 向量中的所有数据元素属于区块中的相同列。单个区块运算采用特定 SIMD 向量对齐，并且偶然地，区块内的 SIMD 向量对齐必须倒转。这可以通过使用矩阵转置运算来完成。可以按照 SIMD 指令来表达条带运算，并且特化可以使用人为展开的循环或者通过使用模板展开的循环来发生。

[0072] 在第二示例的部分实施中，最优化算法通过对二十组分混合物执行 EOS 计算并为各种子算法记录平均执行时间来测试。不包括涉及使用条带的最优化。表 1 示出针对稳定性和分相算法获得的执行时间。实施的最优化导致 2.2 倍的平均加速。

[0073] 表 1 : 用于稳定性和分相算法的执行时间

[0074]

算法	之前	之后	加速
稳定性	107 μ s	42.5 μ s	2.51x
分相	60.6 μ s	32.8 μ s	1.85x
总计	168 μ s	75.9 μ s	2.21x

[0075] 加速主要由在稳定性和分相算法中执行的牛顿算法中的显著性能改进而引起。表 2 示出在应用到稳定性和分相算法时关于逐次代换和牛顿算法两者所见的性能改进。

[0076] 表 2 : 用于逐次代换和牛顿算法的执行时间

[0077]

算法	之前	之后	加速
逐次代换(S)	24.7 μ s	19.7 μ s	1.25x
逐次代换(PS)	22.8 μ s	19.1 μ s	1.19x
牛顿(S)	82.0 μ s	22.8 μ s	3.60x
牛顿(PS)	37.8 μ s	13.7 μ s	2.76x

[0078] S= 稳定性, PS= 分相

[0079] 因为矩阵运算中的若干发生牛顿算法, 并且矩阵运算可以大幅受益于本文描述的最优化, 所以在牛顿算法中的显著加速是可预期的。牛顿算法在计算涉及矩阵 - 向量乘法的 EOS 参数和寻求三次方程的根的同时调用少数基础子算法例如雅可比和线性求解器。表 3 列出在这些基础算法中的性能改进。显著加速在线性求解器中发生, 其中性能被提高近六倍。

[0080] 表 3 :用于基础算法的执行时间

[0081]

算法	之前	之后	加速
EOS 参数	0.69 μ s	0.36 μ s	1.92x
逸度导数	1.27 μ s	0.59 μ s	2.15x
雅可比(S)	1.48 μ s	0.37 μ s	3.97x
雅可比(PS)	3.91 μ s	1.54 μ s	2.54x
线性求解器	9.40 μ s	1.65 μ s	5.70x

[0082] S= 稳定性, PS= 分相

[0083] III. 第三示例可应用于支持任意长度的 SIMD 向量的硬件。该实施方式通过将循环重排序以使最内循环取决于同时计算的数目, 来使用 SIMD 向量并行运行若干算法。如本文描述, 单精度和双精度 SIMD 向量都可以被使用在向量化中。数据组织或存储可以包括将区块交错从而与在存储器中顺序地存储区块相反。将区块交错可以改进 SIMD 向量化。交错区块的数目可以匹配 SIMD 向量的长度, 其还可以取决于硬件体系结构。可以用相似形式交错条带。表 4 示出与参考执行时间相比, 针对用于第二和第三途径的稳定性和分相算法获得的执行时间。

[0084] 表 4 :用于稳定性和分相算法的执行时间

[0085]

算法	参考	第二途径	第三途径
稳定性	107 μ s	42.2 μ s	14.0 μ s
分相	59.9 μ s	33.5 μ s	9.7 μ s
总计	167 μ s	75.7 μ s	23.7 μ s

[0086] 表 5 示出与参考执行时间相比,逐次代换和牛顿算法两者应用到用于第二和第三途径的稳定性和分相算法时所见的性能改进。

[0087] 表 5 :用于逐次代换和牛顿算法的执行时间

[0088]

算法	参考	第二途径	第三途径
逐次代换(S)	24.5 μ s	19.9 μ s	4.5 μ s
逐次代换(PS)	23.0 μ s	19.6 μ s	3.0 μ s
牛顿(S)	82.0 μ s	22.3 μ s	9.5 μ s
牛顿(PS)	37.9 μ s	13.9 μ s	6.7 μ s

[0089] S= 稳定性, PS= 分相

[0090] 表 6 列出与参考执行时间相比,在用于第二和第三途径的这些基础算法中的性能改进。注意表 4-6 分别包括稍不同于表 1-3 中的结果的用于第二途径的更新计算。

[0091] 表 6 :用于基础算法的执行时间

[0092]

算法	参考	第二途径	第三途径
EOS 参数	0.69 μ s	0.36 μ s	0.11 μ s/0.24 μ s
逸度导数	1.26 μ s	0.59 μ s	0.18 μ s
雅可比(S)	1.48 μ s	0.37 μ s	0.16 μ s
雅可比(PS)	3.91 μ s	1.53 μ s	0.43 μ s
线性求解器	9.17 μ s	1.48 μ s	0.47 μ s

[0093] S= 稳定性, PS= 分相

[0094] 在第三示例中,区块大小可以不再受向量长度影响,并可以被选择用于尽可能有效地利用寄存器。在其中少量高速缓存存储器可用的 GPU 上,区块大小可以被选择用于减少主存储器和高速缓存存储器之间的数据传递。同样地,条带长度不再受向量长度影响并可以被选择用于匹配区块大小。

[0095] 进一步地,区块运算可以被重设计为按照 SIMD 指令工作。由于不同的数据结构,

因此第三示例中的设计可以不同于第二示例中的设计。矩阵转置运算可以不再是必需的。相似地，除了使用基于 SIMD 的区块运算以外，第三示例中的线性求解器可以与第一示例中的线性求解器相同。可以按照 SIMD 指令来表达条带运算并且可以使用特化。线性求解器中的复杂循环可以被人为展开或通过使用模板来展开。

[0096] 工业示例

[0097] 本文描述的指令受限算法中的矩阵和向量最优化可以在各种情境中应用，例如 3D 图形分析、加密、数据挖掘、压缩、信号处理、图像处理、链式法则评估、数值法例如有限元和有限体积分析，以及模式识别例如地震模式。在 3D 图形分析的情况下，矩阵可以微小因此区块化可以不必需，但本文描述的其他最优化可以使用。在其他情况例如图像处理中，矩阵可以巨大因此运算是存储器约束的，并且最大速度在为存储器和高速缓存之间有效数据传递将矩阵区块化时获得。对于这样的巨大矩阵，嵌套区块化策略可以用来将不同高速缓存级之间与高速缓存和寄存器之间数据传递最优化。寻找最优区块大小可以依靠试错法。此外使用非均匀区块在区块大小的选择上提供自由。

[0098] 另外，将本技术与有限元法一起使用可以在对每个区块运算时确保最大效率而不在块大小上放置任何约束。在每个有限元数值单元内的变化可以由形状函数描述。这可以意味着从形状函数导致的线性系统的系数矩阵是块稀疏的，其中非零元素成群出现。稀疏矩阵存储格式可以通过在存储器中毗连存储每个区块来利用该结构。块稀疏矩阵也可以在其他数值法例如有限体积法中出现。

[0099] 本技术可以应用于地震模式识别，其中一组矩阵运算在每个数据点和每个数据点的邻近数据点上执行。包括的邻近数据点定义中心在所关注数据点上的 3D 窗口。3D 窗口的大小和形状由人们寻找的特征确定。通过区块化，矩阵运算可以更有效。

[0100] 本技术也可以应用于链式法则，其中链式法则用来计算当两组变量之间具有隐关系时一组变量相对于另一组变量的导数。区块和条带最优化可以应用，其中微小矩阵向量运算与链式法则相关。

[0101] 同样，区块和条带最优化可以在任意硬件体系结构上的 EOS 模型中应用于 EOS 计算。EOS 计算可以用来确定混合物的稳定热动力状态，并通常包括数个嵌套循环。与 EOS 计算中的循环相关的循环和跳跃开销可以与循环内的运算一样计算上昂贵。分析 EOS 模型的工作一般集中在减少与 EOS 计算相关的计算工作上，而不注意单个 EOS 计算如何可以实施以完全利用基础硬件。

[0102] 寻找混合物的稳定热动力状态对应于寻找热动力函数例如吉布斯自由能的全局最小值。尽管全局最小化技术存在，但它们一般过于计算昂贵从而不能在储层仿真中包括。结果石油工业主要集中在其中 EOS 计算分成稳定性测试和分相计算的方程求解途径上。

[0103] 稳定性测试可以确定给定相态是否稳定或被分成至少两个不同相态。如果测试相态不稳定，则所有相态的量和成分可以由分相计算确定。随后稳定性测试可以为新相态中的至少一个重复从而确定更多相态是否存在。稳定性测试可以重复直到确定相态的稳定配置。一般地，认为水不可与所有其他相态混溶，并且仅非水相态在 EOS 计算中包括。

[0104] 图 6 是概述根据本技术的实施例的执行 EOS 计算的方法的过程流程图。在方框 602 处，开始点用平衡比 K_i 的猜测初始化，该平衡比 K_i 定义为相态 B 和参考相态 A 之间的组分 i 的摩尔分数 n_i ，以使满足方程 1 中的状况。

[0105] $K_i = n_i^B / n_i^A$ (1)

[0106] 在方框 604 处, 稳定性测试基于 K_i 的初始猜测, 检查极少量的第二相态的形成是否影响混合物的吉布斯自由能。即, 稳定性测试可以确定混合物是否被分成至少两个相态。使用一阶泰勒展开式证明原成分的微小扰动, 从而描述丰富相态的吉布斯自由能。所得的有待求解的非线性方程组在方程 2 中示出。

[0107]

$$\ln N_i + \ln \varphi_i(T, P, n_j) - \mu_i^0(T, P, n_j^0) = 0 \quad (2)$$

[0108] 在方程 2 中, μ 是化学势, φ 是逸度, 并且上标“0”表示原混合物的流体性质。变量 N_i 通过在方程 3 中示出的表达式涉及摩尔分数 n_i 。

[0109] $n_i = N_i / \sum N_j \quad (3)$

[0110] 在方框 606 处确定混合物的稳定性。如果稳定性测试断定混合物稳定, 则方法进展到方框 610。因为仅有一个相态存在并且吉布斯自由能不随着另一相态形成而降低, 所以稳定混合物不需要进一步分相。然而如果稳定性测试断定吉布斯自由能随着另一相态形成而降低, 则混合物不稳定并且方法进展到方框 608。在方框 608 处执行分相计算。分相确定在每个仿真单元中存在的每个相态的量和成分。有待求解的对应非线性方程组表达相态之间化学平衡, 如在方程 4 中示出。

[0111]

$$\ln K_i - \ln \varphi_i^A(T, P, n_j^A) + \ln \varphi_i^B(T, P, n_j^B) = 0 \quad (4)$$

[0112] 可从质量守恒示出摩尔分数 n_i 由方程 5 中的表达式涉及平衡比。

[0113] $n_i^A = n_i^0 / (1 + \beta(K_i - 1)), n_i^B = K_i n_i^A \quad (5)$

[0114] 在方程 5 中, β 是由方程 6 给出的 Rachford-Rice 方程的解。

[0115] $RR = \sum n_i^0 (K_i - 1) / (1 + \beta(K_i - 1)) = 0 \quad (6)$

[0116] 该途径容易延伸到多相态。

[0117] 在完成分相计算后, 方法进展到方框 610, 其中为每个相态计算所选择流体性质和导数。在将仿真中每个相态的流动建模时需要该信息。

[0118] 稳定性和分相方程的非线性性质需要迭代解规程。普遍途径是使用逐次代换(SS)直到符合切换准则并然后使用牛顿迭代直到收敛。

[0119] 图 7 是概述根据本技术的实施例的用于 EOS 计算的稳定性和分相算法的迭代性质的过程流程图 700。在方框 702 执行逐次代换。逐次代换的步骤在图 8 中进一步描述。在方框 704 执行对逐次代换收敛的检查。如果逐次代换收敛则方法进展到方框 706。如果逐次代换不收敛则方法返回到方框 702。这样, 逐次代换在循环中执行直到符合切换准则, 如下面描述。

[0120] 同样在方框 706 执行牛顿计算。牛顿计算的步骤在图 8 中进一步描述。在方框 708 执行对牛顿计算收敛的检查。如果牛顿计算收敛则方法进展到方框 710。如果牛顿计算不收敛则方法返回到方框 706。这样, 牛顿计算在循环中执行直到计算收敛。

[0121] 通常, 检查收敛可以意味着将方程 2 或方程 4 的残差的范数与预定准则比较。借

助逐次代换循环,准则可以相对宽松,而用于牛顿循环的准则可以严格得多。因为逐次代换循环的转换准则确定算法何时切换到牛顿循环,所以其频繁称为切换准则。

[0122] 图8是概述根据本技术的实施例的执行逐次代换(图7的方框702)和牛顿算法(图7的方框706)的方法的过程流程图800。逐次代换(图7的方框702)和牛顿算法(图7的方框706)都涉及相同基础计算。在方框802更新成分。逐次代换(图7的方框702)和牛顿算法(图7的方框706)的主要区别是如何更新成分。

[0123] 当执行逐次代换时,成分通过直接从方程2和方程4更新 $\ln N_i$ 和 $\ln K_i$ 来更新。随后对应摩尔分数分别从方程3和方程5计算。牛顿算法基于方程2和方程4的残差更新成分。方程2和方程4的残差可以描述为每个分别方程的左侧偏离零的量。残差如何精确转变为成分更新由线性系统的雅可比矩阵确定。线性系统的雅可比矩阵分别指代方程2和方程4的导数的矩阵。另外扭断算法涉及计算逸度导数并求解线性方程组,如在图9中进一步解释。

[0124] 尽管每当提供良好初始猜测时牛顿法迅速收敛,但其可以在初始猜测远离真实解时发散。因此更鲁棒的逐次代换可以用来获得良好初始猜测。牛顿算法可以然后应用直到收敛。尽管牛顿算法比逐次代换计算上更昂贵,但成本可以由提高的收敛速率证明是合理的。

[0125] 在方框804,EOS参数更新,并且在方框806逸度系数更新。在方框808残差更新。在方框810执行对收敛的检查。如果算法收敛则方法在方框812终止。如果算法不收敛则方法返回到方框802。

[0126] 图9是概述如何可以在牛顿迭代内更新成分的方法的过程流程图。在方框902计算关于摩尔分数的逸度导数。

[0127] 在方框904逸度导数用来构建有待求解的非线性方程组的雅可比矩阵。雅可比矩阵定义为关于主变量的残差导数。在稳定性和分相算法中,残差由方程2和方程4分别定义。

[0128] 在方框906,所得的线性方程组经求解更新主变量。一旦主变量确定则成分容易更新。通过特化,EOS算法可以为所选择数目的组分生成,允许编译器展开更多循环。同样,SIMD/SIMT向量化的可替换使用是通过将循环重排序以使最内循环取决于同时EOS计算的数目,为多种混合物并行运行EOS计算。因为与矩阵运算相关的循环的迭代数目取决于代替混合物中组分数目的区块数目,所以使用区块在EOS计算中减少循环和跳跃开销。EOS区块运算也可以用来计算逸度导数,并用于在牛顿算法中构建雅可比矩阵。牛顿算法也使用线性求解器。在EOS计算中,雅可比矩阵是稠密的以使大多数元素非零。求解具有稠密雅可比矩阵的线性方程组的两个普遍算法是都可以按照区块运算表述的高斯消去和LU分解。

[0129] 系统

[0130] 图10是根据本技术的实施例的计算机系统1000的框图,该计算机系统1000可以用来在指令受限算法中最优化矩阵和向量计算。中央处理单元(CPU)1002耦合到系统总线1004。CPU1002可以是任何通用CPU,尽管只要CPU1002(和示范系统1000的其他部件)支持如本文描述的运算,则其他类型体系结构的CPU1002(或示范系统1000的其他部件)可以使用。本领域技术人员认识到尽管仅单个CPU1002在图10中示出,但另外CPU可以存在。此外计算机系统1000可以包括图形处理单元(GPU)1014。系统可以包括联网多处理器计算

机系统，该联网多处理器计算机系统可以包括混合并行 CPU/GPU 系统。CPU1002 和 GPU1014 可以根据各种示范实施例执行逻辑指令。例如 CPU1002 可以与 GPU1014 并行执行指令以便根据上面连同图 1 和 6-9 描述的运算流程执行处理。所描述的处理可以并行执行。

[0131] 计算机系统 1000 也可以包括计算机部件例如非暂时性计算机可读介质。计算机可读介质的示例包括可以是 SRAM、DRAM、SDRAM 等的随机访问存储器(RAM)1006。计算机系统 1000 也可以包括另外非暂时性计算机可读介质例如只读存储器(ROM)1008，该 ROM1008 可以是 PROM、EPROM、EEPROM 等。RAM1006 和 ROM1008 保存用户和系统数据和程序，如在本领域中已知。计算机系统 1000 也可以包括输入 / 输出(I/O)适配器 1010、通信适配器 1022、用户接口适配器 1024、显示驱动器 1016 和显示适配器 1018。I/O 适配器 1010、用户接口适配器 1024 和 / 或通信适配器 1022 可以在某些实施例中使用户能够与计算机系统 1000 交互以便输入信息。

[0132] I/O 适配器 1010 可以将另外非暂时性机器可读介质例如存储装置 1012 连接到计算机系统 1000，该存储介质 1012 包括例如硬盘驱动器、紧凑光盘(CD)驱动器、软盘驱动器、磁带驱动器等。存储装置可以在 RAM1006 不满足相关为本技术的实施例的运算存储数据的存储器需求时利用。计算机系统 1000 的数据存储可以用于存储信息和 / 或如本文所讨论的使用或生成的其他数据。用户接口适配器 1024 将用户输入装置例如键盘 1028、指点装置 1026 和 / 或输出装置耦合到计算机系统 1000。根据某些示范实施例，显示适配器 1018 由 CPU1002 驱动以控制显示装置 1020 上的显示，从而例如显示关于从计算所得的仿真的显示信息或表示。

[0133] 系统 1000 的体系结构可以按希望变化。例如，任何合适的基于处理器的装置可以使用，无限制包括个人计算机、膝上计算机、计算机工作站和多处理器服务器。此外，实施例可以在专用集成电路(ASIC)或超大规模集成(VLSI)电路上实施。实际上，本领域技术人员可以使用能够根据实施例执行逻辑运算的任何数目的合适结构。

[0134] 在实施例中，到计算机系统 1000 的输入数据可以包括 3D 图形数据、图像数据、储层模型和 EOS 模型。输入数据可以另外包括各种矩阵和向量数据。

[0135] 本技术可以容易受到各种修改和可替换形式，并且在上面讨论的示范实施例仅作为示例示出。然而本技术不意图限于本文披露的具体实施例。当然，本技术包括落入附属权利要求的真实精神和保护范围内的全部替换、修改和等效。

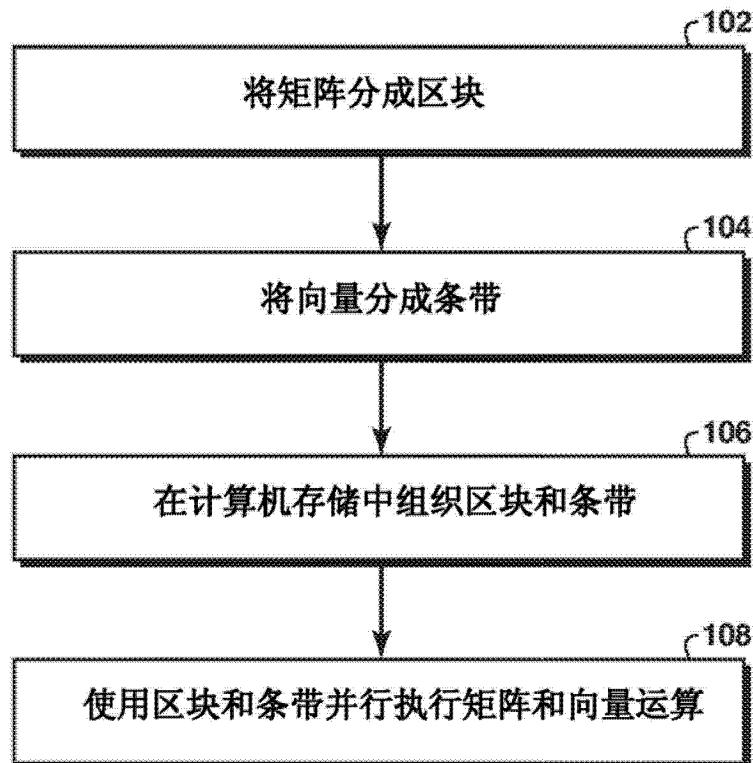
100

图 1

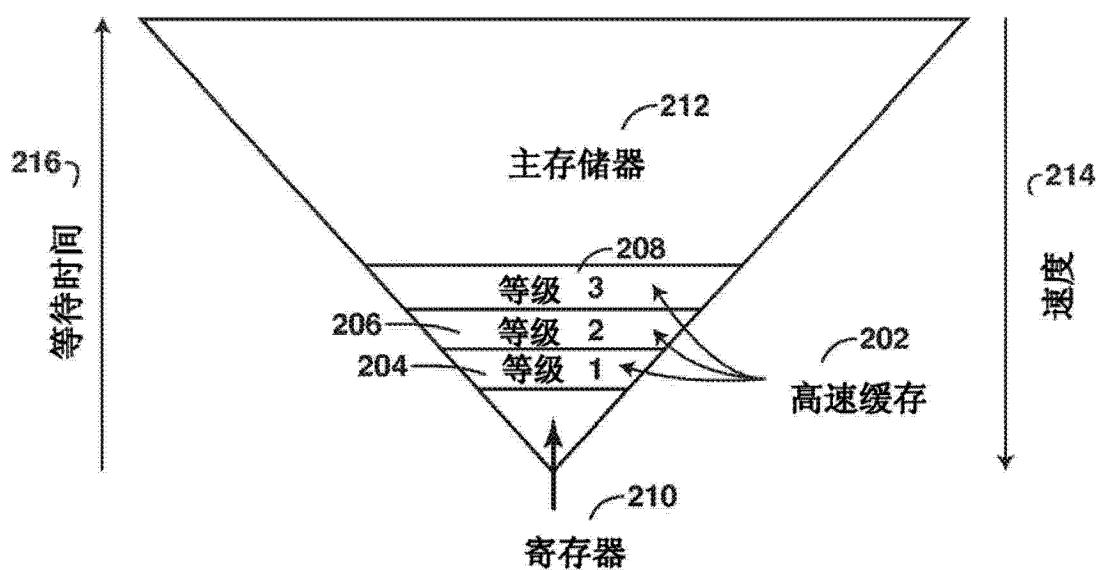
200

图 2

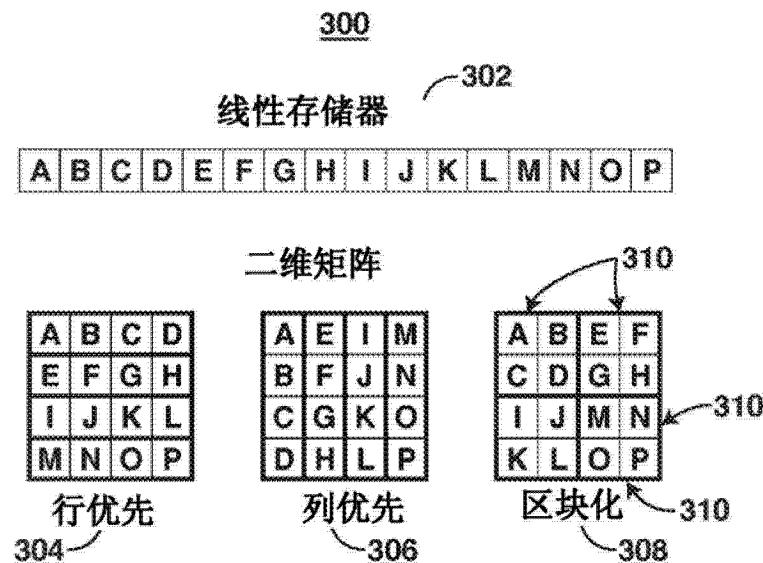


图 3

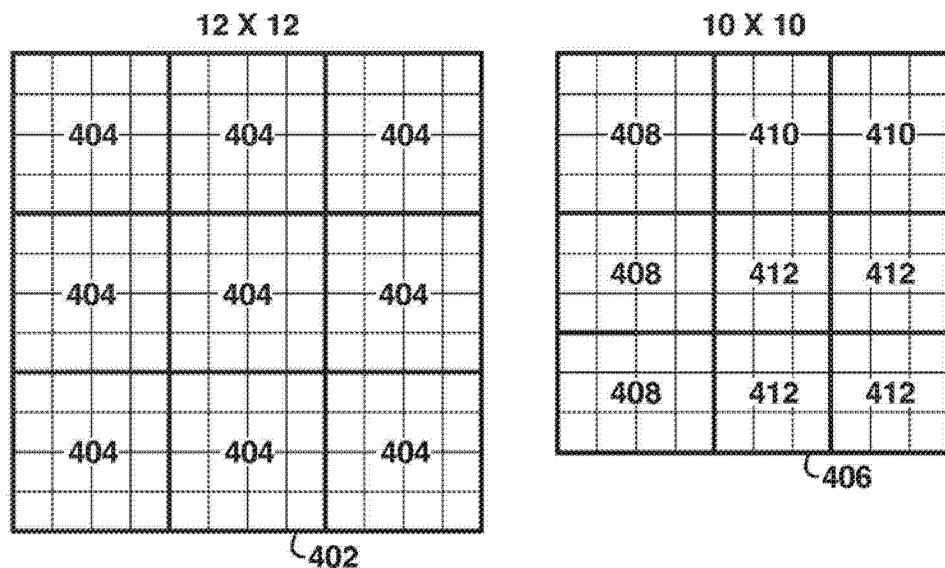
400

图 4



图 5

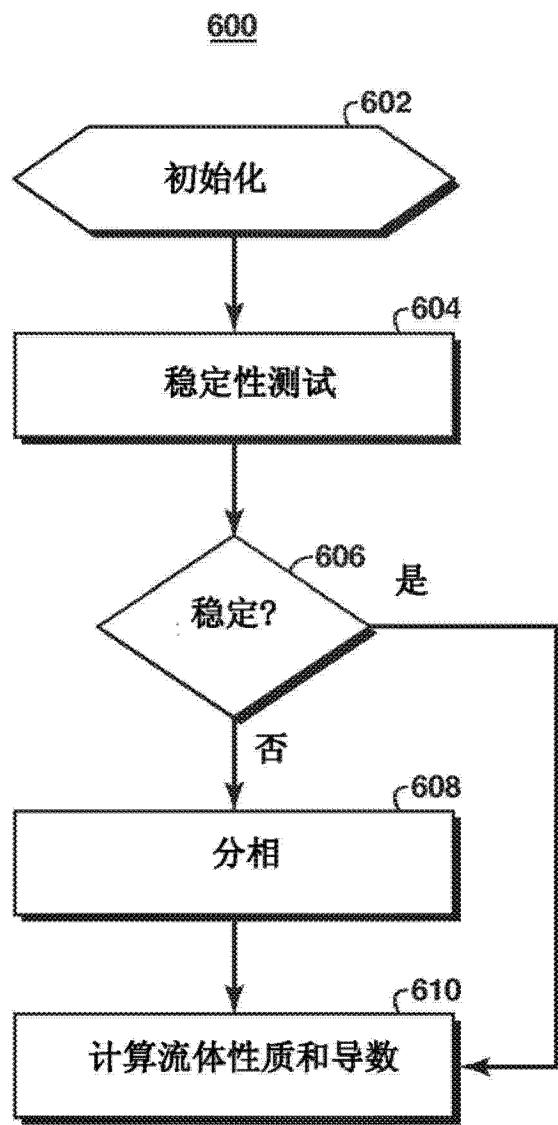


图 6

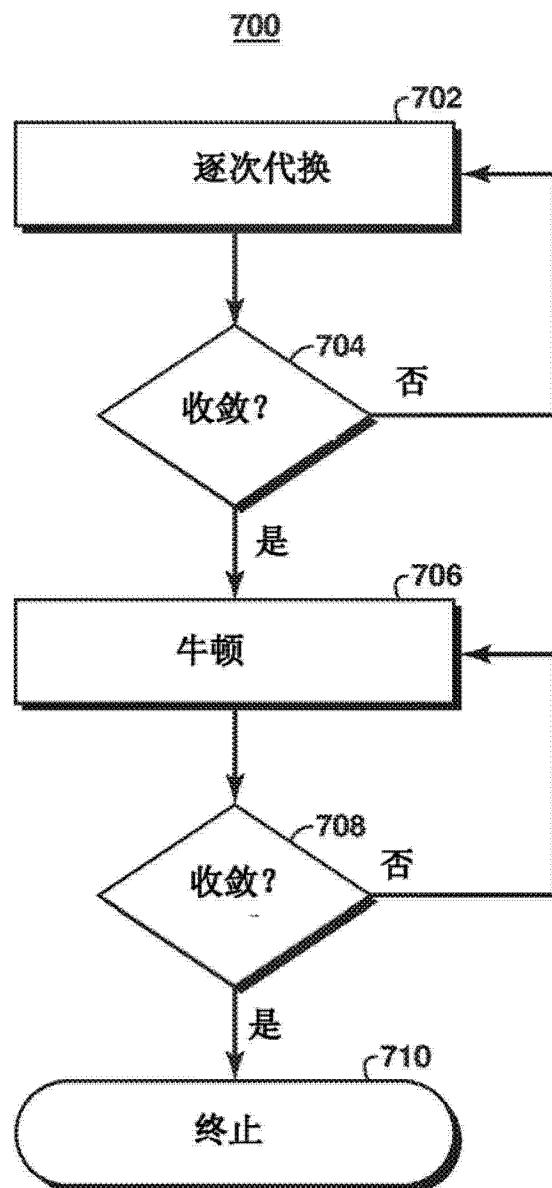


图 7

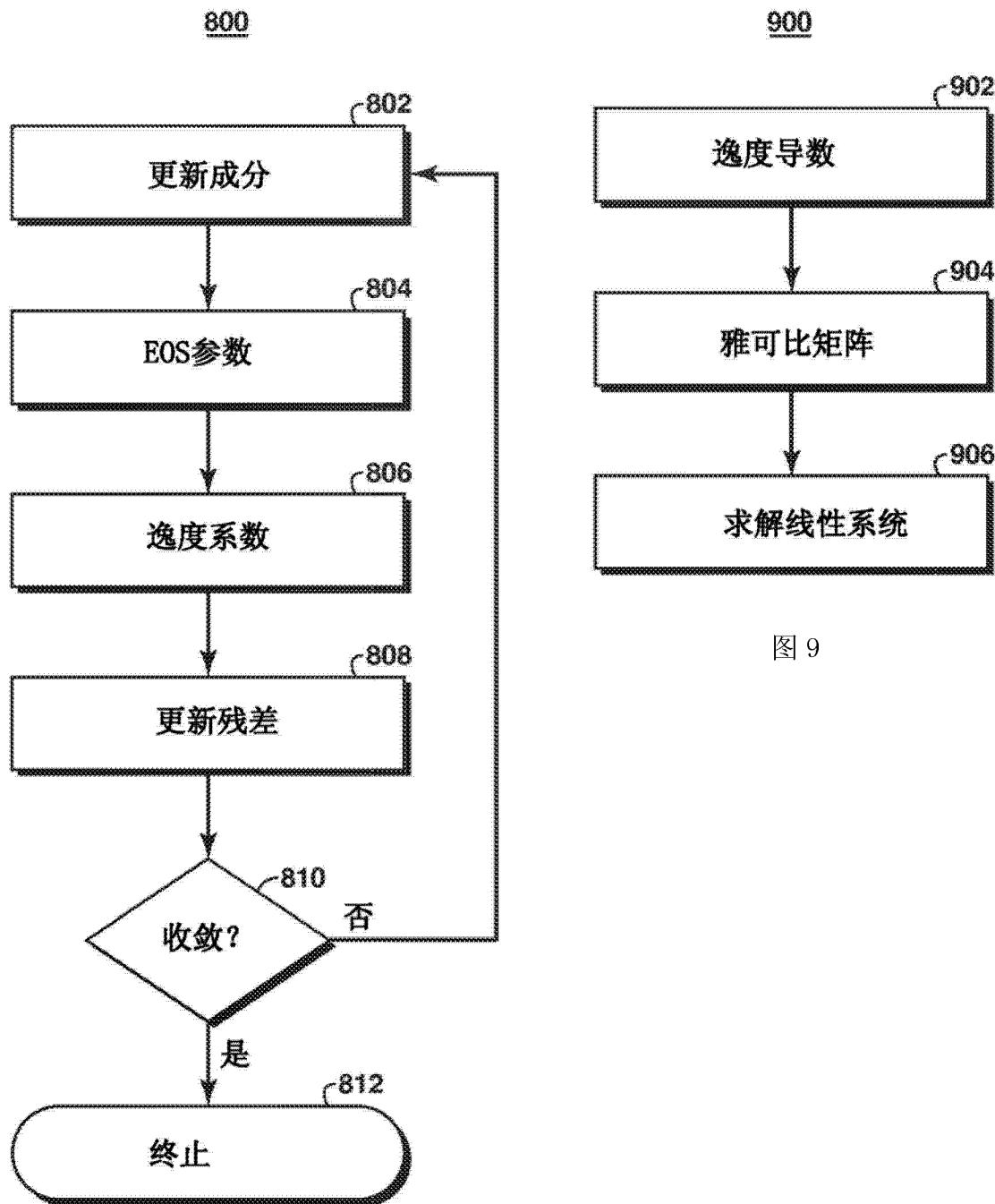


图 9

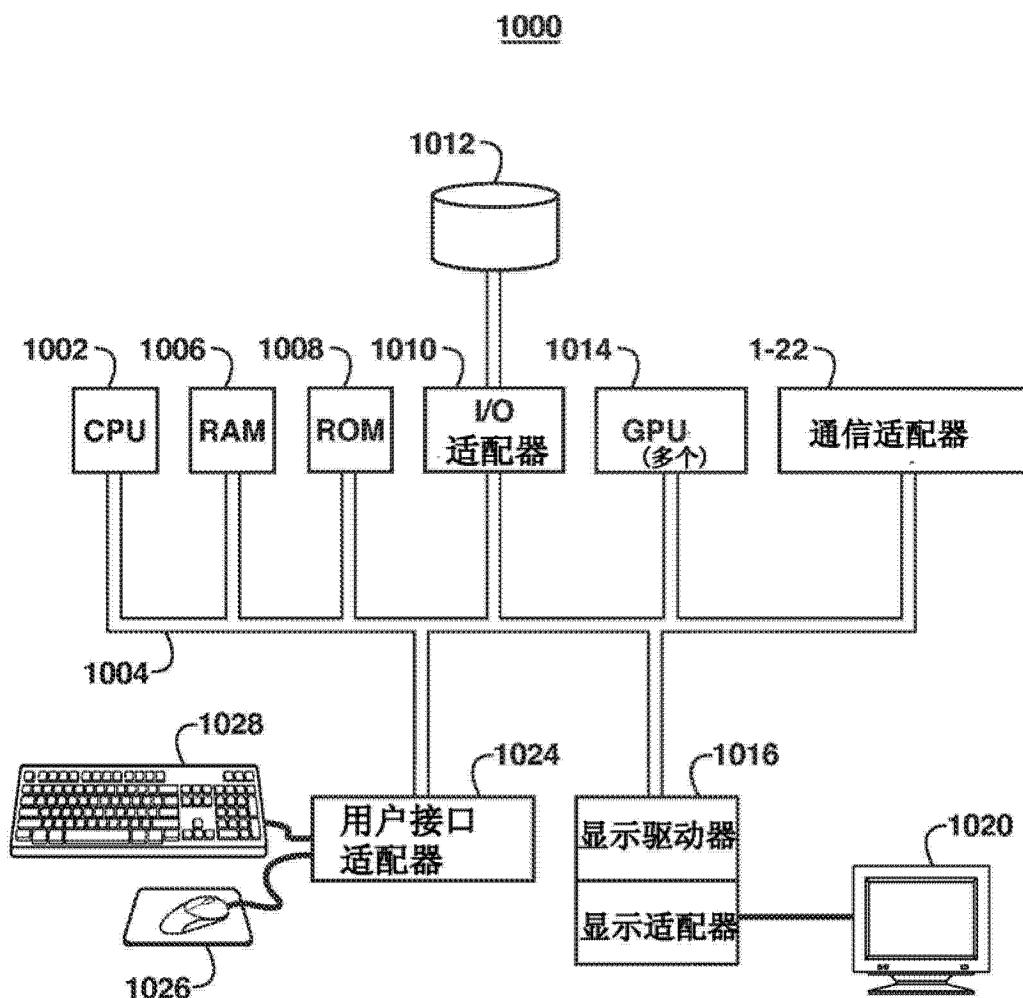


图 10