(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 105528243 B
(45)授权公告日 2019.01.11

(21)申请号 201510382438.5

(22)申请日 2015.07.02

(65)同一申请的已公布的文献号
申请公布号 CN 105528243 A

(43)申请公布日 2016.04.27

(73)专利权人 中国科学院计算技术研究所
地址 100190 北京市海淀区中关村科学院南路6号

(72)发明人 陈莉 韩冬妮 侯雄辉

(74)专利代理机构 北京律诚同业知识产权代理有限公司 11006
代理人 祁建国 梁挥

(51)Int.Cl.
G06F 9/48(2006.01)

(56)对比文件
CN 101571814 A,2009.11.04,
CN 102193826 A,2011.09.21,
US 2005/0081200 A1,2005.04.14,
CN 102681901 A,2012.09.19,
CN 102831011 A,2012.12.19,
李静梅,等.“基于异构多核处理器的高效任务调度算法”.《高技术通讯》.2012,第22卷(第3期),全文.

审查员 崔茜

权利要求书2页 说明书12页 附图7页

(54)发明名称

一种利用数据拓扑信息的优先级分组调度方法及系统

(57)摘要

本发明公开一种利用数据拓扑信息的优先级分组调度方法及系统,该方法包括:获取数据拓扑信息的原始网格空间,设置原始网格空间的网格片的尺寸与浮点精度,生成新网格空间;根据新网格空间和并行区的stencil格式,构建紧缩任务图(四维时空域),计算其中每个任务的优先级分组编号;获取当前任务所访问的数据片,并通过格式抽象或函数指针与实参的形式来决定是否涉及邻居数据依赖,并生成相应标记,根据标记识别出涉及邻居数据依赖的循环,这种循环为有效时间步,根据当前有效时间步把循环上的任务映射到紧缩任务图的某个任务,根据后者的优先级编号计算当前任务的优先级编号,支持任务的优先级分组调度。

CN 105528243 B

1.一种利用数据拓扑信息的优先级分组调度方法,其特征在于,包括:

步骤1,获取所述数据拓扑信息的原始网格空间,设置所述原始网格空间的网格片的尺寸与浮点精度,生成新网格空间;

步骤2,根据新网格空间和并行区的stencil格式,构建紧缩任务图,计算所述紧缩任务图中每个任务的优先级分组编号,其中优先级计算规则为:按照行优先的顺序给各个数据片分配初始优先级编号;所述紧缩任务图上拓扑第一层任务的优先级编号为其访问的各个数据片所对应优先级编号的最大值;其他任务的优先级编号是各个依赖边的依赖源任务的优先级编号的最大值;

步骤3,获取当前任务所访问的数据片,并通过获取其所在循环的stencil计算的差分格式抽象的形式来决定是否涉及邻居数据依赖,或通过一个或者多个枚举所有邻居依赖的函数指针结合特定取值的实参的形式来决定是否涉及邻居数据依赖,并生成相应标记;根据所述标记,识别出涉及邻居数据依赖的循环,所述循环为有效时间步;根据所述有效时间步和任务的空间坐标,把当前任务映射到紧缩任务图上的某个任务,根据后者的优先级编号计算当前任务的优先级编号。

2.如权利要求1所述的利用数据拓扑信息的优先级分组调度方法,其特征在于,所述步骤1中每个数据片具有唯一的网格空间坐标。

3.如权利要求1所述的利用数据拓扑信息的优先级分组调度方法,其特征在于,所述步骤2与所述步骤3之间还包括根据每个任务所访问的数据片的空间位置,获取所述任务在所述新网格空间中的坐标。

4.如权利要求1所述的利用数据拓扑信息的优先级分组调度方法,其特征在于,所述步骤2还包括分组的约束包括数据足迹要小于最后一级cache的尺寸的一半;每个分组内的数据重用度大于预设阈值。

5.如权利要求1所述的利用数据拓扑信息的优先级分组调度方法,其特征在于,还包括把每个任务亲和到固定线程,从而把每个分组的任务均匀地分散到各个线程,把任务调度有效约束在当前分组内执行。

6.一种利用数据拓扑信息的优先级分组调度系统,其特征在于,包括:

生成新网格空间模块,用于获取所述数据拓扑信息的原始网格空间,设置所述原始网格空间的网格片的尺寸与浮点精度,生成新网格空间;

紧缩任务图的优先级分组模块,根据新网格空间和并行区的stencil格式,构建紧缩任务图,确定紧缩任务图的分组宽度;所述紧缩任务图上任务的优先级计算规则为:按照行优先的顺序给各个数据片分配初始优先级;所述紧缩任务图的拓扑第一层任务的优先级为该任务所访问的各个数据片所对应优先级编号的最大值;紧缩任务图上其他任务的优先级是这个任务的各个依赖边的依赖源任务的优先级编号的最大值;

动态分组模块,用于获取当前任务所访问的数据片,并通过获取其所在循环的stencil计算的差分格式抽象的形式来决定是否涉及邻居数据依赖,或通过一个或者多个枚举所有邻居依赖的函数指针与特定取值的实参的形式来决定是否涉及邻居数据依赖,并生成相应标记;根据所述标记,识别出涉及邻居数据依赖的循环,所述循环为有效时间步;根据所述有效时间步把所述循环上的任务映射到紧缩任务图上的某个任务,根据后者的的优先级编号计算当前任务的优先级编号。

7.如权利要求6所述的利用数据拓扑信息的优先级分组调度系统，其特征在于，所述生成新网格空间模块中每个网格数据片具有唯一的网格空间坐标。

8.如权利要求6所述的利用数据拓扑信息的优先级分组调度系统，其特征在于，还包括获取坐标模块，用于根据当前任务所访问的数据片的空间位置，获取任务在所述新网格空间中的坐标。

9.如权利要求6所述的利用数据拓扑信息的优先级分组调度系统，其特征在于，分组的约束包括数据足迹要小于最后一级cache的尺寸的一半；每个分组内的数据重用度大于预设阈值。

10.如权利要求6所述的利用数据拓扑信息的优先级分组调度系统，其特征在于，还包括亲和性分配模块，用于把每个任务亲和到固定线程，从而把每个分组的任务均匀地分散到各个线程，把任务调度有效约束在当前分组内执行。

一种利用数据拓扑信息的优先级分组调度方法及系统

技术领域

[0001] 本发明涉及计算机及用户级的任务调度领域，特别涉及一种利用数据拓扑信息的优先级分组调度方法及系统。

背景技术

[0002] 现代的微处理器结构中存在难以逾越的内存墙，如何优化程序中的数据局部性，提高其在处理器上的缓存性能，是应用优化的重要课题。

[0003] 结构网格方法的背景如下：在热扩散、电磁场和流体力学等领域中存在着大量的非线性方程求解，仅有极少数问题可以得到解析解或者摄动解，结构网格方法是解决这类问题的最重要的数值方法之一，其典型步骤为：离散化求解域，将连续的求解域化为有限的离散点集，比如固定的等距网格或自适应的结构网格，然后用差商代替微商，在求解域内求出网格节点的离散解，通过一定迭代步数使得差分方程的解收敛于微分方法的解。

[0004] 在结构(离散)网格中每个点进行的差分计算，常常需要邻近的一些点的值，这一类的计算被称为Stencil计算，Stencil计算是结构网格应用的核心，也是世界公认的7类高性能计算模式之一。

[0005] 相关技术1：用Time skewing方法优化stencil计算中的数据重用，如图1所示：

[0006] 结构网格应用的计算访存比很低，如何优化cache的利用率成为提高这类程序在多核系统上性能的关键之一，学术界提出了各种Time skewing的优化算法，以同时优化并行性和数据局部性，其主要思想是，对stencil计算的迭代步和空间网格组成的多维空间进行倾斜的分块——把原来很长的数据重用降低到分块内部，然后确定这些分块之间的依赖关系，实现一种准静态的任务调度，Time skewing优化的本质是把外层的时间迭代循环与内部的空间网格的遍历计算进行统一的考虑，实施复杂的循环变换。

[0007] Time skewing方法需要把最外层的时间循环变换到内层才能实施，对于复杂应用而言实施的难度很高，甚至是不可实施的。适应性网格方法就是这样一种应用类型，该方法是求解复杂物理问题的一种有效方法，它根据误差动态地决定在每个时间点或者空间点是否需要适应性加密，以提高求解精度。首先，这类方法在实现上非常复杂——必须基于成熟的领域编程框架来进行应用开发，数据结构变得复杂而不再是简单的数组，计算步骤也出现C/Fortran混杂的情况，这导致编译很难对整个网格空间和各个计算步骤之间进行静态的依赖分析，从而无法实现time skewing。第二，由于网格划分的动态性，网格点的邻居是不能静态确定的，这也限制了time skewing优化的实施。

[0008] 相关技术2：任务并行和任务调度中的调度优化，如图2(图2中的英文描述对应的是choleskey分解中的四个基本操作，在代码实现中是调用的高性能数学库的库函数)、图3所示：

[0009] 任务并行编程模型是近年多核平台上广泛研究和使用的并行编程模型，旨在简化并行编程和提高多核利用率，产业界和学术界研发了很多这类的并行编程接口，比如有Cilk/Cilk++、OpenMP3.0、X10、Habanero-Java、TBB、TPL等，用这种接口写的程序，程序中的

任务会形成一棵派生树，也即一个有向无环图，其运行时系统负责任务调度，每个核对应一个物理线程，每个物理线程会执行许多逻辑任务，这种在用户态空间进行的任务调度大大降低调度的开销，从而提高多线程程序的执行效率，运行时系统采用任务窃取调度算法，获得负载平衡，提高多核的使用效率。

[0010] 在提高任务调度中的数据局部性方面，Umut A.Acar,Guy E.Blelloch,Robert D.Blumofe.The Data Locality of Work Stealing.ACM Symposium on Parallel Algorithms and Architectures.Proceedings of the twelfth annual ACM symposium of Parallel algorithms and architectures.Jul.2000,Bar Harbor,Maine,United States.pp.1-12提出了计算和任务亲和的机制，并被一些实际的任务调度系统所采纳，这种方法对网格应用的优化更多地体现在集群的节点间存储优化上，对于cache的优化效果并不显著。

[0011] 优先级调度也是一种常见的调度优化，但是它一般被用于那些关键路径明显比其他路径更长的任务图，改善任务的负载平衡。比如图2展示的是基于分块结构的choleskey分解的任务图，其中蓝黑色的dpotrf是关键任务，需要给予更高的优先级。

[0012] 但是很多应用的任务图中并不存在明显的关键任务，比如结构网格应用的任务图一般呈现出图3的样子，各个任务之间耦合紧密，各个有向路径几乎都是一样长的，并不存在某个特别长的关键路径。(这里，任务的颜色表示其在哪个线程上执行。)目前没有人发现，优先级分组调度能用于改善结构网格应用的任务调度性能。

发明内容

[0013] 本发明针对的是结构网格应用，目的是在传统的任务调度机制之上，通过语用户接口和调度系统的协同，实现一种利用数据拓扑信息的优先级分组调度方法及系统。

[0014] 本发明提出一种利用数据拓扑信息的优先级分组调度方法，包括：

[0015] 步骤1，获取所述数据拓扑信息的原始网格空间，设置所述原始网格空间的网格片的尺寸与浮点精度，生成新网格空间；

[0016] 步骤2，根据新网格空间和并行区的stencil格式，构建紧缩任务图，计算所述紧缩任务图中每个任务的优先级分组编号，其中优先级计算规则为：按照行优先的顺序给各个数据片分配初始优先级编号；所述紧缩任务图上拓扑第一层任务的优先级编号为其访问的各个数据片所对应优先级编号的最大值；其他任务的优先级编号是各个依赖边的依赖源任务的优先级编号的最大值；

[0017] 步骤3，获取当前任务所访问的数据片，并通过格式抽象或函数指针与实参的形式来决定是否涉及邻居数据依赖，并生成相应标记；根据所述标记，识别出涉及邻居数据依赖的循环，所述循环为有效时间步；根据所述有效时间步和任务的空间坐标，把当前任务映射到紧缩任务图上的某个任务，根据后者的优先级编号计算当前任务的优先级编号。

[0018] 所述的利用数据拓扑信息的优先级分组调度方法，所述步骤1中每个数据片具有唯一的网格空间坐标。

[0019] 所述的利用数据拓扑信息的优先级分组调度方法，所述步骤2与所述步骤3之间还包括根据每个任务所访问的数据片的空间位置，获取所述任务在所述新网格空间中的坐标。

[0020] 所述的利用数据拓扑信息的优先级分组调度方法，所述步骤2还包括分组的约束包括数据足迹要小于最后一级cache的尺寸的一半；每个分组内的数据重用度大于预设阈值。

[0021] 所述的利用数据拓扑信息的优先级分组调度方法，还包括把每个任务亲和到固定线程，从而把每个分组的任务均匀地分散到各个线程，把任务调度有效约束在当前分组内执行。

[0022] 本发明还提出一种利用数据拓扑信息的优先级分组调度系统，包括：

[0023] 生成新网格空间模块，用于获取所述数据拓扑信息的原始网格空间，设置所述原始网格空间的网格片的尺寸与浮点精度，生成新网格空间；

[0024] 紧缩任务图的优先级分组模块，根据新网格空间和并行区的stencil格式，构建紧缩任务图，确定紧缩任务图的分组宽度；所述紧缩任务图上任务的优先级计算规则为：按照行优先的顺序给各个数据片分配初始优先级；所述紧缩任务图的拓扑第一层任务的优先级为该任务所访问的各个数据片所对应优先级编号的最大值；紧缩任务图上其他任务的优先级是这个任务的各个依赖边的依赖源任务的优先级编号的最大值；

[0025] 动态分组模块，用于获取当前任务所访问的数据片，并通过格式抽象或函数指针与实参的形式来决定是否涉及邻居数据依赖，并生成相应标记；根据所述标记，识别出涉及邻居数据依赖的循环，所述循环为有效时间步；根据所述有效时间步把所述循环上的任务映射到紧缩任务图上的某个任务，根据后者的的优先级编号计算当前任务的优先级编号。

[0026] 所述的利用数据拓扑信息的优先级分组调度系统，所述生成新网格空间模块中每个网格数据片具有唯一的网格空间坐标。

[0027] 所述的利用数据拓扑信息的优先级分组调度系统，还包括获取坐标模块，用于根据当前任务所访问的数据片的空间位置，获取任务在所述新网格空间中的坐标。

[0028] 所述的利用数据拓扑信息的优先级分组调度系统，分组的约束包括数据足迹要小于最后一级cache的尺寸的一半；每个分组内的数据重用度大于预设阈值。

[0029] 所述的利用数据拓扑信息的优先级分组调度系统，还包括亲和性分配模块，用于把每个任务亲和到固定线程，从而把每个分组的任务均匀地分散到各个线程，把任务调度有效约束在当前分组内执行。

[0030] 由以上方案可知，本发明的优点在于：

[0031] 技术效果包括两个部分，其一是支持任务到四维时空域(紧缩任务图)的自动映射；其二是支持优先级分组调度，且分组内部、相邻分组之间实现对共享缓存的重用。

[0032] 1.提供一组用户接口，支持任务图到四维时空域(紧缩任务图)的自动映射，且不受程序复杂程度的限制。这组用户接口如表1所述：

[0033] 表1.新的一组用户接口

[0034]

支持紧缩任务图映射和优先级分组的用户接口	
任务并行区的初始化阶段（紧缩任务图的构造）	描述网格空间的全局分片信息 描述每个分片的尺寸信息，以及浮点数据的精度信息 描述该并行区域内的任务并行在哪些轴向有邻居依赖 描述并行区域内涉及的网格变量的个数。 用户选择优先级分组的方式
任务本身的描述	描述任务的数据访问信息（含邻居依赖） 描述任务在网格空间的坐标信息

[0035] 本技术点带来有益效果的原因：1)根据用户提供的信息，调度系统能实现任务到四维时空域(紧缩任务图)的映射。用户负责划分任务、描述任务的数据访问信息。任务到四维时空域的映射中的最大难点是——邻居依赖的表达，由于我们提供了最通用的函数指针的描述方式，这个难点得以克服；2)本工作属于调度系统的优化，可以适应任意复杂的结构网格应用。

[0036] 2.利用用户的指导信息，调度系统能自动选择一个好的分组形状、数据亲和方式、优先级分组方式，并优化任务之间的缓存重用、提高应用性能，可以抽象地用表2描述：

[0037] 表2.优先级分组的两个算法(自然剖分算法、平面剖分算法)

[0038]

算法阶段	优先级分组的算法	
	自然剖分	平面剖分
初始化阶段	计算出优先级分组的优选尺寸	
	对网格空间剖分，得到初始映射函数	对时空域进行剖分，得到： 相对映射函数、亲

[0039]

		和性函数
任务构造阶段	根据前驱任务的情况，计算优先级 累计分配亲和性；	根据映射函数，计算优先级 根据亲和性函数，得到亲和性

[0040] 本技术点带来有益效果的原因：1)本发明可以选择出好的分组形状和尺寸，分组的数据足迹可以用分割面的倾斜投影进行估算，其中要结合网格片的尺寸，对于满足共享缓存容量的分组尺寸，本发明优选缓存重用度更好的分组尺寸；2)本发明能把调度约束在一个分组内，本技术点负责确定数据亲和的映射，把同一分组内的任务尽量分散到各个线程，从而能获得共享缓存上的数据重用；3)分组的时间轴高度与组内的数据重用度相关，本发明会选重用度更高的分块形状，相邻分组之间的缓存重用则通过分组序号的精心分配来实现。

附图说明

[0041] 图1为一维问题的多种不同的time skewing方法图；

[0042] 图2为Cholesky分解(5x5)的任务依赖图；

[0043] 图3为网格应用的任务依赖图；

[0044] 图4为一维网格空间下采用自然剖分方式得到的优先级分组图；

[0045] 图5为二维网格空间下，采用平面剖分方式获得的分组图；

[0046] 图6为平面剖分和自然剖分的差别。三角部分在自然剖分中从属于左邻的分组图；

[0047] 图7为本发明整体流程图。

[0048] 其中附图标记为：

[0049] 步骤100/101/102/103/104/105。

具体实施方式

[0050] 为解决以上技术问题，本发明提出一种利用数据拓扑信息的优先级分组调度方法及系统。

[0051] 以下为本发明的整体步骤，如图7所示：

[0052] 获取所述数据拓扑信息的原始网格空间，设置所述原始网格空间的网格片的尺寸与浮点精度，生成新网格空间。分片方法对应全局的任务划分，而网格片的尺寸和浮点精度则用于指导数据足迹的计算，每个分片具有唯一的网格空间坐标；

[0053] 根据新网格空间和并行区的stencil格式，构建紧缩任务图，时间轴向上它只有一个分组的高度；计算其中每个任务的优先级分组编号，计算规则为：按照行优先的顺序给各个数据片分配初始优先级；所述紧缩任务图的拓扑第一层任务的优先级为该任务所访问的各个数据片所对应优先级的最大值；其他任务的优先级编号是该任务的各个依赖源任务的优先级编号的最大值；

[0054] 获取当前任务所访问的数据片，并通过格式抽象或函数指针与实参的形式来决定

是否涉及邻居数据依赖，并生成相应标记；根据所述标记，识别出涉及邻居数据依赖的循环，所述循环为有效时间步；根据所述有效时间步把所述循环上的任务映射到所述紧缩任务图，根据后者的优先级编号计算当前任务的优先级编号；

[0055] 用户(根据数据片的空间位置)给出任务在所述新网格空间中的网格空间坐标；

[0056] 获取邻居数据依赖。用户指定任务所访问的数据片，并用stencil格式抽象或(函数指针,实参)的形式表示邻居数据依赖。

[0057] 网格应用中涉及邻居数据依赖的有两类计算，一类是stencil计算，一类是近邻数据交换(可能是同层网格的相邻数据片之间、或者相邻层的对应数据片之间)，stencil计算的格式是固定的，可以静态描述，但是近邻数据交换就不一定了，对于一些复杂的应用，比如适应性网格应用，每个数据片在相应网格层上存在几个邻居是不确定的，而且数据片是分离存储的，这时候调度系统无法计算出其近邻数据片，但用户可以提供一个函数指针以及相应的实参来统计其邻居数据片。

[0058] 调度系统根据数据依赖的标记，识别出涉及邻居依赖的循环，称其为有效时间步；

[0059] 在由有效时间步和网格空间组成的四维空间上，根据数据依赖的倾斜情况，进行基于超平面划分的分组，分组的约束有两个：i)数据足迹要小于最后一级cache的尺寸的一半；ii)组内的数据重用度要尽量大(大于预设阈值)；

[0060] 按照行优先的顺序给各个分组分配优先级，以发掘分组之间的数据重用；

[0061] 调度系统把每个任务亲和到固定线程，从而把每个分组的任务均匀地分散到各个线程，把任务调度有效约束在当前分组内执行。

[0062] 本发明还提出一种利用数据拓扑信息的优先级分组调度系统，包括：

[0063] 生成新网格空间模块，用于获取所述数据拓扑信息的原始网格空间，设置所述原始网格空间的网格片的尺寸与浮点精度，生成新网格空间；

[0064] 紧缩任务图的优先级分组模块，根据新网格空间和并行区的stencil格式，构建紧缩任务图，确定紧缩任务图的分组宽度；所述紧缩任务图上任务的优先级计算规则为：按照行优先的顺序给各个数据片分配初始优先级；所述紧缩任务图的拓扑第一层任务的优先级为该任务所访问的各个数据片所对应优先级编号的最大值；紧缩任务图上其他任务的优先级是这个任务的各个依赖边的依赖源任务的优先级编号的最大值；

[0065] 动态分组模块，用于获取当前任务所访问的数据片，并通过格式抽象或函数指针与实参的形式来决定是否涉及邻居数据依赖，并生成相应标记；根据所述标记，识别出涉及邻居数据依赖的循环，所述循环为有效时间步；根据所述有效时间步把所述循环上的任务映射到紧缩任务图上的某个任务，根据后者的的优先级编号计算当前任务的优先级编号。

[0066] 所述生成新网格空间模块中每个网格数据片具有唯一的网格空间坐标。

[0067] 获取坐标模块，用于根据当前任务所访问的数据片的空间位置，获取任务在所述新网格空间中的坐标。

[0068] 分组的约束包括数据足迹要小于最后一级cache的尺寸的一半；每个分组内的数据重用度大于预设阈值。

[0069] 亲和性分配模块，用于把每个任务亲和到固定线程，从而把每个分组的任务均匀地分散到各个线程，把任务调度有效约束在当前分组内执行。

[0070] 以下为本发明具体步骤，场景：本实施案例中网格空间上的数据是连续存储的，固

定网格;软件:在调度系统的用户接口和内部实现机制上;改进点:提出新的用户接口,提出优先级分组的自动算法。因为空间网格最多有3维,为方便描述,下面的实施例就只考虑3维的情况,为方便描述,本实施例假设优先级序号越小、级别越高,如下所示:

[0071] S1在任务并行区开始前增加初始化阶段

[0072] S1.1描述网格空间的全局分片信息。比如3维网格空间在3个轴向都做划分,划分可以是均匀的,也可以是不均匀的,均匀的划分给出划分的间距即可,不均匀的划分则要给出每个分片的形状,调度系统根据以上信息计算出分片后的网格空间。

[0073] S1.2描述每个分片的尺寸信息,以及浮点数据的精度信息。对于均匀的分片,尺寸是明确的,而对于不均匀的分片,用户可以给出一个平均的分片尺寸,浮点数据的精度则表示计算采用单精度、双精度或者128位精度。

[0074] S1.3描述该并行区域内的任务并行在哪些轴向有邻居依赖,并行区域内涉及的网格变量的个数。

[0075] S1.4用户选择优先级分组的方式:自然剖分、平面剖分。

[0076] S1.5调度系统在此阶段计算出优先级分组的尺寸<wx,wy,wz,wt>,此时也就知道了“紧缩任务图”的时间轴高度。

[0077] 比如分片的尺寸是16^3,采用双精度计算,参与计算的有Narray个网格变量,当前机器的L3缓存是20M。若stencil格式在三个空间轴向的6个方向都有邻居依赖,则<wx,wy,wz,wt>分组的数据足迹是sizeof(double)*Narray*(wx+wt-1)*(wy+wt-1)*(wz+wt-1)<=20M*0.5*1.1,这里1.1是一个放松系数以弥补投影估算方法的激进性。由此可计算出符合条件且数据足迹最大的多个分组形状,并选择其中重用度最高的形状。

[0078] 图4展示的是1维网格空间情况下,用自然剖分方式获得的优先级分组情况。因为时间轴的分块大小是8,X轴的分块大小是4,此处的任务图被拆分为两个窗口,可以看出X轴的划分平面向t轴倾斜。

[0079] S1.6根据用户选择的剖分方法,构建映射表

[0080] S1.6.1若采用自然剖分:

[0081] 对网格空间按照<wx,wy,wz>进行分块剖分,按照线性化顺序为每个数据分片分配初始分组号,映射函数initial_priority:<z,y,x>->priority;

[0082] S1.6.2若采用平面剖分:

[0083] S1.6.2.1需预先对由wt个有效时间步和网格空间组成的多维空间(对应紧缩任务图),进行仿射划分。

[0084] 根据1.3提供的邻居依赖信息,把X轴上wx间距的划分平面向t轴倾斜,然后依次把Y轴、Z轴的划分平面也进行相应的倾斜。这样分割出的凸形区域就是一个优先级分组,每个分组都拥有一个三维坐标。

[0085] S1.6.2.2各个分组按照<z,y,x>的字典序排序,优先级递减。

[0086] S1.6.2.3得到优先级映射函数relative_priority:<z,y,x,t>->priority_offset

[0087] S1.6.2.4同时得到亲和性映射函数affinity:<z,y,x,t>->thread_id

[0088] S1.6.2.5计算出每个时间窗口内的最大任务数total;

[0089] S2对并行区内的每个任务,分配优先级分组和线程亲和性:

[0090] S2.1用户描述任务的数据访问信息

[0091] S2.2用户描述本任务在网格空间的坐标信息

[0092] S2.3在任务的构造阶段，调度系统把该任务映射到对应的优先级分组

[0093] S2.3.1设当前窗口内的最小分组号from;若选自然剖分，则total是当前分组的累计任务数；

[0094] S2.3.2根据本任务是否有邻居依赖，累计当前窗口内的有效时间步ti;若当前窗口已满，则from＝from+total。设本任务在窗口内的时空坐标是<ti,zi,yi,xi>。

[0095] S2.3.3若采用自然剖分：

[0096] S2.3.3.1若本任务没有前驱或者前驱属于前一个窗口，则分组序号是initial_priority(zi,yi,xi)+total;否则，本任务的分组序号是其所有前驱的序号的最大值。

[0097] S2.3.3.2给本任务分配亲和性。

[0098] S2.3.4若采用平面剖分:

[0099] 本任务的分组序号是relative_priority(ti,zi,yi,xi)+from;其亲和性是affinity(ti,zi,yi,xi)；

[0100] 以下为本发明的实施例1，如下所示：

[0101] 这是采用jacobi迭代进行3d7p的stencil计算。在三维的网格空间进行stencil计算，对2个空间轴进行了全局任务划分，离散空间采用固定网格，数据采用单一数组的存储方式。

[0102]

```
//这是任务并行区的初始化阶段
#pragma acemesh dag init
//给出网格空间的原始形状。Mesh 是本系统提供的接口。
//虽然网格本身是 3 维的，但是任务划分只在 2 维进行，只需要描述 2 维。
```

[0103]

```
Mesh<int> regular_mesh(2, D3, D2);
//给出网格空间的全局划分，也即分片的方法。同时给出分片的尺寸。
regular_mesh.set_uniform_block(2, s3, s2);
regular_mesh.set_typical_blocksize(s3*s2*D1, DOUBLE);
//并行区域内在哪些空间轴向有邻居依赖。
regular_mesh.set_skewed_info(2, 1,1,1,1);    //只在两个空间轴向有邻居依赖

//任务并行区的开始
#pragma acemesh dag start
for(t = 0; t<256; t++) {  //总共计算256次才结束
    for (int kk = 1; kk<D3- 1; kk+=s3)
for (int jj = 1; jj<D2- 1; jj+=s2)
//下面的制导，描述本任务的数据访问情况。是否涉及邻居依赖。网格空间的坐标。
//编译器处理后，这些制导将变成对调度系统的功能调用
#pragma acemesh task \
/*本任务会读该地址对应的数据片，以及邻居的数据片*/  \
data(&u[t%2][kk][jj][0]:<R,R>,\
&u[(t+1)%2][kk][jj][0]:<R,N>)  \
  layout(<kk, jj>)  /*本任务在网格空间的坐标*/
{
//这是一个并发任务的入口
for (int k = kk; k <min(kk+s3,D3 - 1); k++)
for (int j = jj; j <min(jj+s2, D2 - 1); j++)
for (int i = 1; i < D1- 1; i++)
u[(t+1)%2][k][j][i] = alpha * (u[t%2][k][j][i]) +
                              beta    *    (u[t%2][k+1][j][i]    +
u[t%2][k][j+1][i]
+ u[t%2][k][j][i+1]+ u[t%2][k-1][j][i]
```

[0104]

```
                    + u[t%2][k][j-1][i]+ u[t%2][k][j][i-1]);
    } //finishing kk-jj-ii loop nest
    } //finishing t loop
    #pragma acemesh dag finish              //这是任务并行区的结束
    }
```

[0105] 这里分片的尺寸是12^3,采用双精度计算,参与计算的有2个网格变量,当前机器的L3缓存是20M。因为本计算在三个空间轴向的6个方向都有邻居依赖,则<wx,wy,wz,wt>分组的数据足迹是8*2*(wx+wt-1)*(wy+wt-1)*(wz+wt-1)*12*12*12<=20M*0.5*1.1,这里1.1是一个放松系数以弥补投影估算方法的激进性。由此得到符合条件且数据足迹最大的两个分组形状是<2,2,2,6>和<4,4,4,4>,但是后者的组内重用度更高。所以调度系统选择的分组形状是<4,4,4,4>,即时间轴的窗口高度和空间轴的分块宽度都是4,组内的最大数据重用度是4。

[0106] 由于3维问题的图形展示效果不佳,下面的图示中只给出1维问题的分组结果。图4给出的是两个时间窗口内的自然剖分结果。这里空间网格划分后的宽度是16,空间分组的宽度是4,时间窗口的高度是8,根据用户标记,每个循环都对应一个有效时间步。每个椭圆上标记了一个三元组<loop_id,task_id,prio_id>,依次表示该任务属于哪个循环、第几个任务、其分组编号是多少。

[0107] 自然剖分分组的具体算法如下。根据本发明的步骤S1.6.1,计算出初始的分组映射表,第一个行的任务被分配了初始的分组编号,分别是0,1,2,3。从第二行起,每个任务的分组号取前驱的最大值。当有效时间步超出窗口高度8,则这一行的任务要根据“初始的分组映射表”进行调整,因为窗口内的分组数是4,所以这一行的分组号为4,5,6,7。以下继续进行分组。

[0108] 如果采用平面剖分的方法,则会把图4中的第一个窗口最右侧的分组,用倾斜的线进一步拆分为3个分组。图6给的例子则是把这个分组拆分为4个分组。平面剖分在其他细节上与自然剖分基本类似。

[0109] 本发明的实施例2,如下所示:

[0110] 本实施例是poisson方程求解的一个应用热点,本发明采用适应性细化网格,网格分片具有动态性和不规则性,数据则采用分片存储的格式而不是单一数组的形式。

[0111] 在调度系统的优先级分组上,并没有与实施例1明显不同的地方。

[0112] 这里主要展示的是,用户如何描述:网格空间的分片划分、任务图到网格空间的映射、以及不规则的邻居数据依赖的表达。

[0113]

```
Mesh<DataIndex> *current_level; //全局量
void AMRPoissonOp::relax(LevelData<FArrayBox>& a_e,
                         const LevelData<FArrayBox>& a_residual,
                         int a_iterations)
{
//任务并行区的初始化
#pragma acemesh init
//描述原始的网格空间
  Mesh<DataIndex> levelmesh(3, dd.loVect(), dd.hiVect());
//由于网格的动态性和不规则性，需要逐个遍历所有的网格分片，传递给调度系统
  For(所有的网格分片)
    regular_mesh.addPatch(网格片的左下坐标, 网格片的右上坐标);
  //调度系统重建分片后的网格空间
regular_mesh.building();
//任务并行区内在哪些空间轴向有邻居依赖。
regular_mesh.set_skewed_info(3, 1,1,1,1,1,1);    //在网格空间的三个轴向都有邻居依赖
current_level = & levelmesh;  //把网格空间信息存储到全局变量中

//给出网格空间的分片尺寸，浮点精度
regular_mesh.set_typical_blocksize(some_size, DOUBLE);
//任务并行区的开始
#pragma acemesh start
  for (int i = 0; i < a_iterations; i++){
      switch (s_relaxMode){
```

```
                case 0:
                  looseGSRB(a_e, a_residual);
                  break;
                default:
                  MayDay::Abort("unrecognized relaxation mode");
              }//end switch
            }//end time iteration
          #pragma acemesh finish
          } //end relax
```

[0114]

[0115] ●下面展示的是looseGSRB()内部的一个子程序,它的主要功能是实现相邻数据片的阴影区数据交换。但是在适应性网格方法中,邻居的个数和所在位置都是不能静态确定的。

[0116]

```
#pragma acemesh for
//遍历所有的网格分片
for (dit.begin(); dit.ok(); ++dit)
{
  DataIndex datIndx = dit();
   //这是一个制导，会被编译器下降为调度系统的功能调用。
#pragma acemesh task data(a_phi[datIndx]:(n,w))  \
/*这里给出函数指针和实参,让调度系统可以知道它需要哪些邻居数据*/
neighborop(_get_neighbors, args)  \
/*这里提取本任务的网格坐标*/
layout((*current_level)[datIndx])
{
//这里省略了复杂的阴影区交换的任务并行代码
}
} //
```

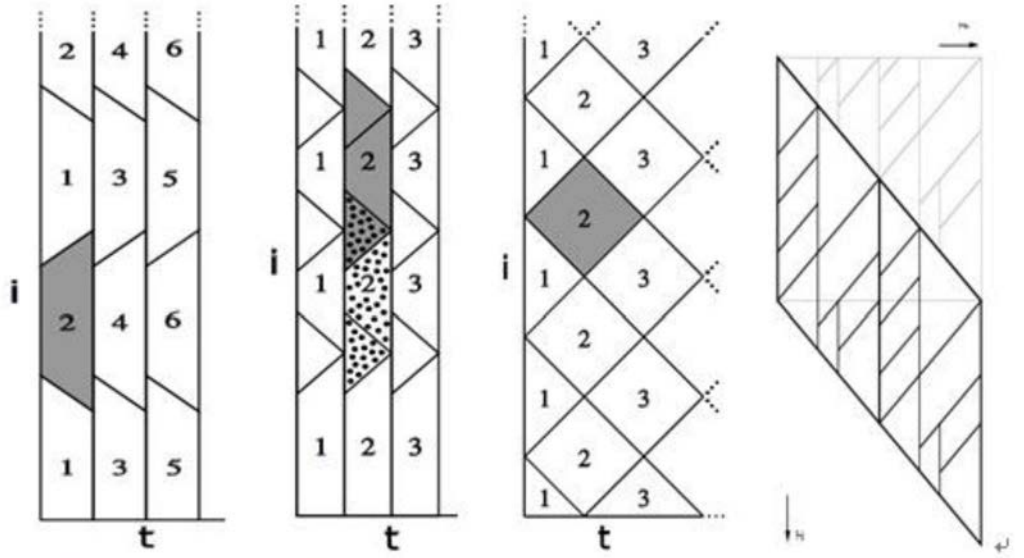

图1

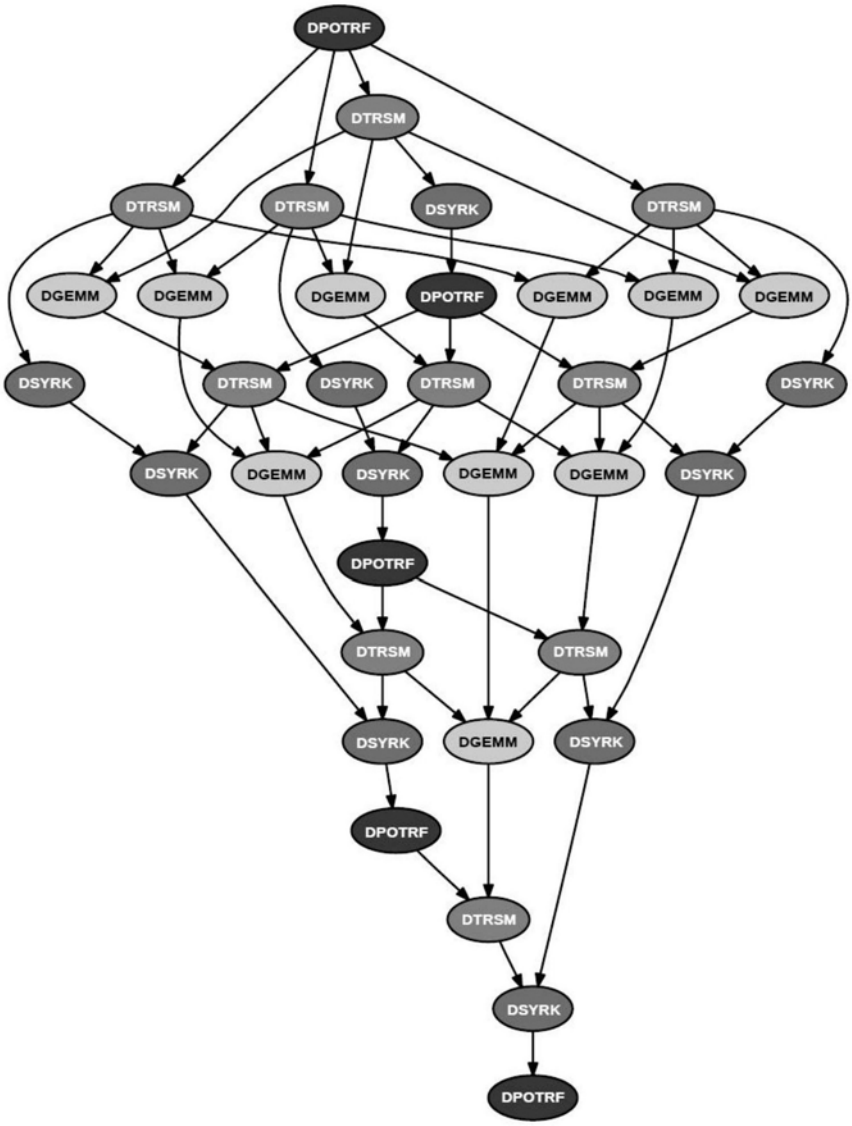
DPOTRF
DTRSM
DTRSM
DTRSM
DSYRK
DTRSM
DGEMM
DGEMM
DGEMM
DPOTRF
DGEMM
DGEMM
DGEMM
DSYRK
DTRSM
DSYRK
DTRSM
DTRSM
DSYRK
DSYRK
DGEMM
DSYRK
DGEMM
DGEMM
DSYRK
DPOTRF
DTRSM
DTRSM
DSYRK
DGEMM
DSYRK
DPOTRF
DTRSM
DSYRK
DPOTRF

图2

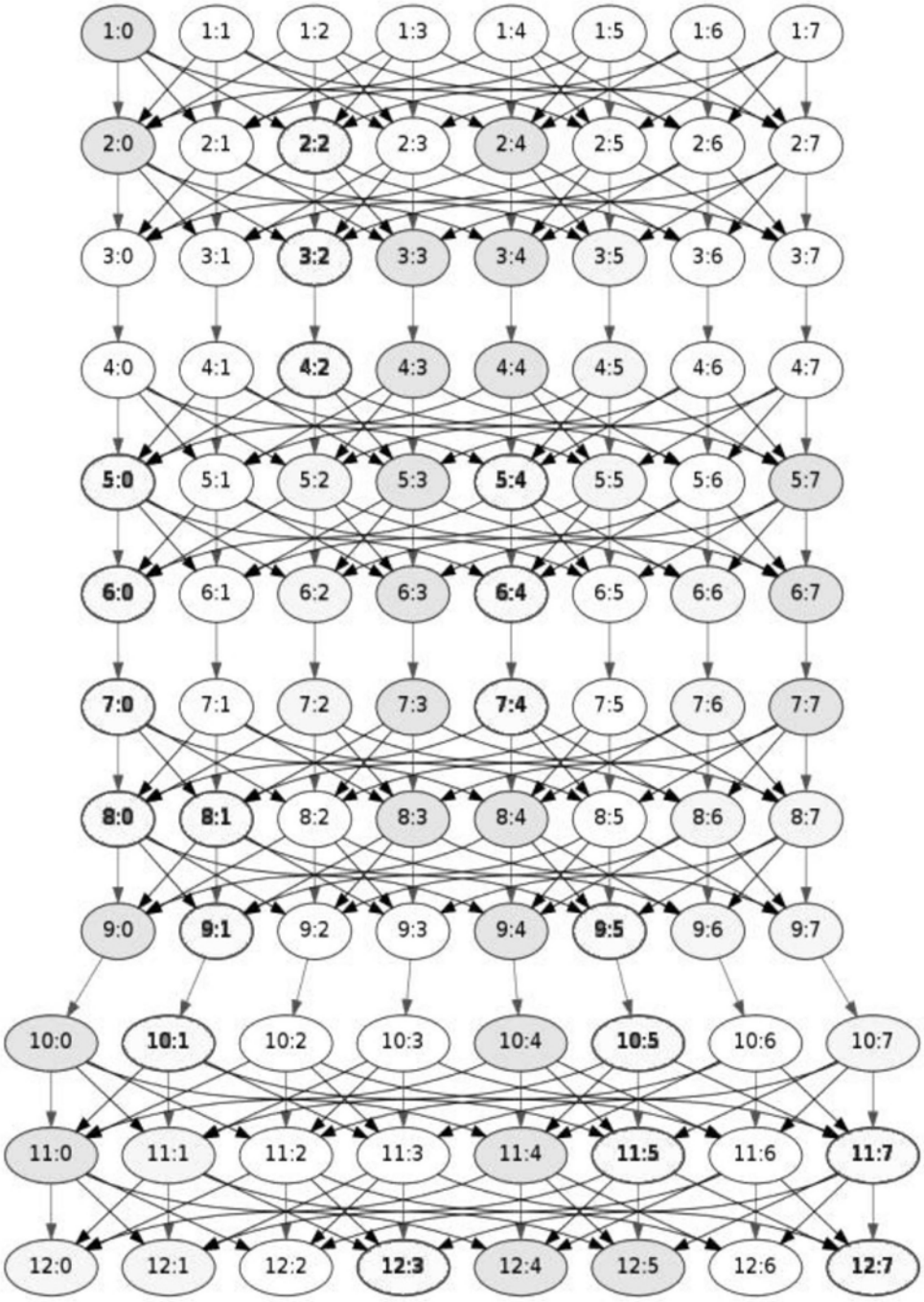
1:0 1:1 1:2 1:3 1:4 1:5 1:6 1:7
2:0 2:1 2:2 2:3 2:4 2:5 2:6 2:7
3:0 3:1 3:2 3:3 3:4 3:5 3:6 3:7
4:0 4:1 4:2 4:3 4:4 4:5 4:6 4:7
5:0 5:1 5:2 5:3 5:4 5:5 5:6 5:7
6:0 6:1 6:2 6:3 6:4 6:5 6:6 6:7
7:0 7:1 7:2 7:3 7:4 7:5 7:6 7:7
8:0 8:1 8:2 8:3 8:4 8:5 8:6 8:7
9:0 9:1 9:2 9:3 9:4 9:5 9:6 9:7
10:0 10:1 10:2 10:3 10:4 10:5 10:6 10:7
11:0 11:1 11:2 11:3 11:4 11:5 11:6 11:7
12:0 12:1 12:2 12:3 12:4 12:5 12:6 12:7

图3

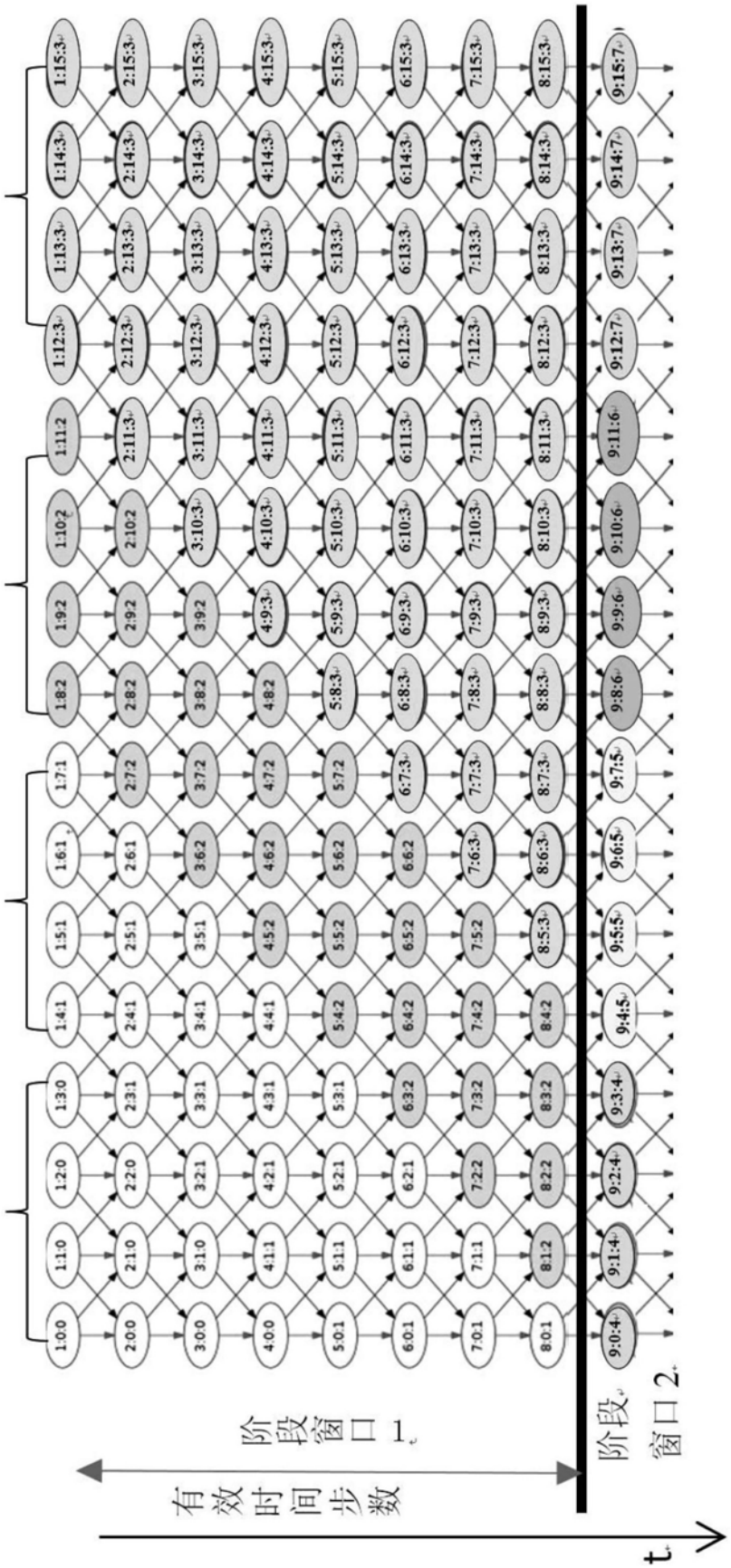

图4

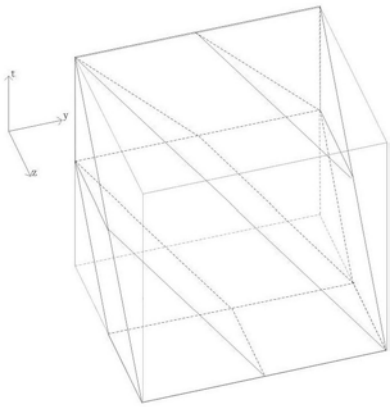

图5

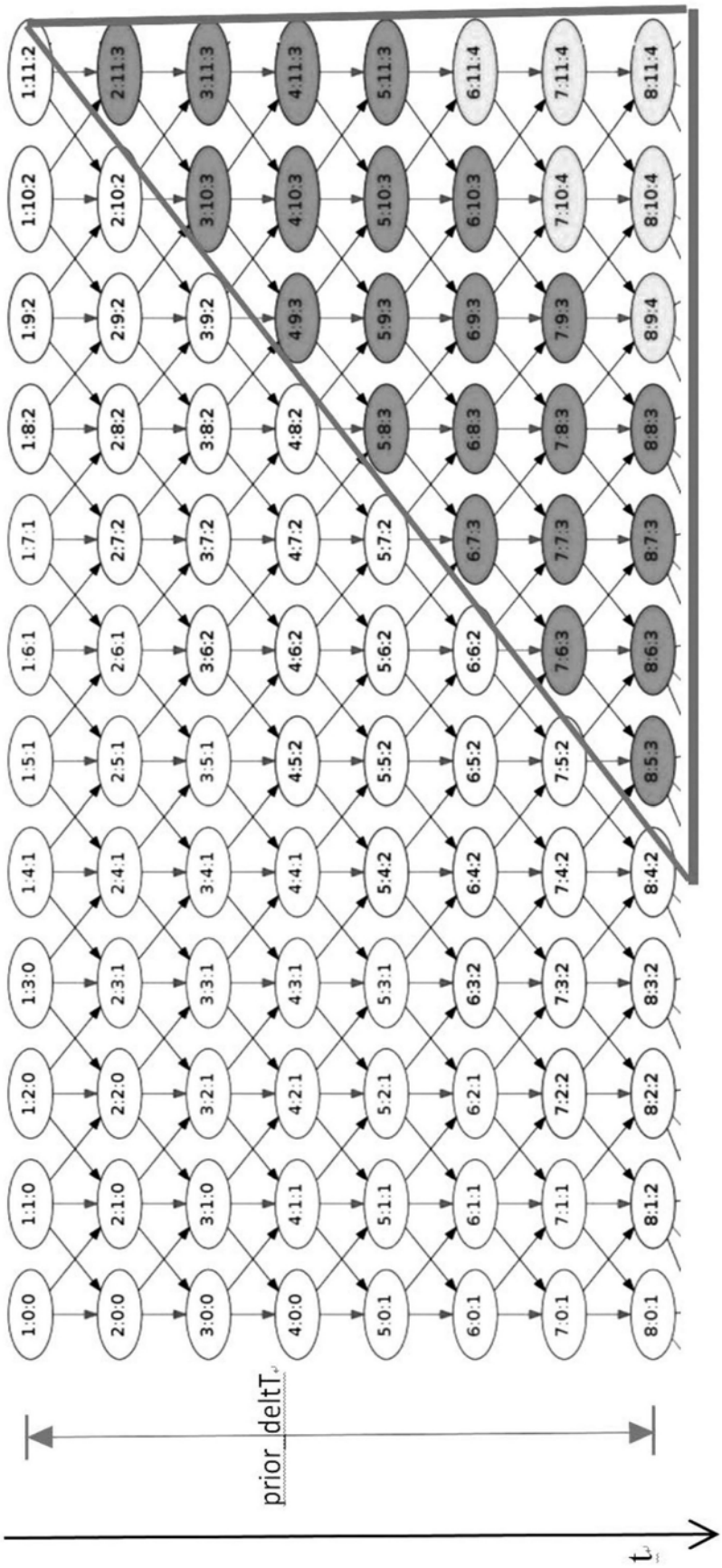
1:0:0
1:1:0
1:2:0
1:3:0
1:4:1
1:5:1
1:6:1
1:7:1
1:8:2
1:9:2
1:10:2
1:11:2
2:0:0
2:1:0
2:2:0
2:3:1
2:4:1
2:5:1
2:6:1
2:7:2
2:8:2
2:9:2
2:10:2
2:11:3
3:0:0
3:1:0
3:2:1
3:3:1
3:4:1
3:5:1
3:6:2
3:7:2
3:8:2
3:9:2
3:10:3
3:11:3
4:0:0
4:1:1
4:2:1
4:3:1
4:4:1
4:5:2
4:6:2
4:7:2
4:8:2
4:9:3
4:10:3
4:11:3
5:0:1
5:1:1
5:2:1
5:3:1
5:4:2
5:5:2
5:6:2
5:7:2
5:8:3
5:9:3
5:10:3
5:11:3
6:0:1
6:1:1
6:2:1
6:3:2
6:4:2
6:5:2
6:6:2
6:7:3
6:8:3
6:9:3
6:10:3
6:11:4
7:0:1
7:1:1
7:2:2
7:3:2
7:4:2
7:5:2
7:6:3
7:7:3
7:8:3
7:9:3
7:10:4
7:11:4
8:0:1
8:1:2
8:2:2
8:3:2
8:4:2
8:5:3
8:6:3
8:7:3
8:8:3
8:9:4
8:10:4
8:11:4
prior_deltT
t

图6

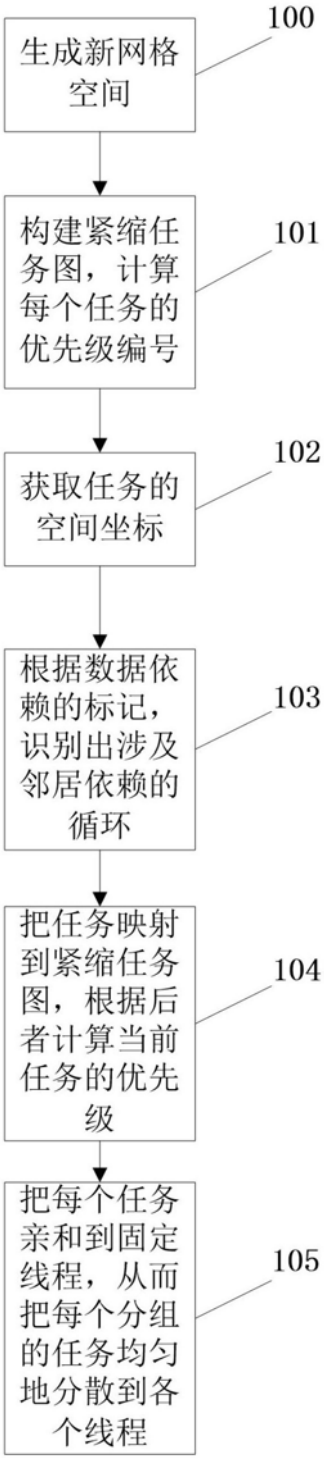
生成新网格空间
100
构建紧缩任务图，计算每个任务的优先级编号
101
获取任务的空间坐标
102
根据数据依赖的标记，识别出涉及邻居依赖的循环
103
把任务映射到紧缩任务图，根据后者计算当前任务的优先级
104
把每个任务亲和到固定线程，从而把每个分组的任务均匀地分散到各个线程
105

图7