(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 109522108 B
(45)授权公告日 2020.10.27

(21)申请号 201811279163.2

(22)申请日 2018.10.30

(65)同一申请的已公布的文献号
申请公布号 CN 109522108 A

(43)申请公布日 2019.03.26

(73)专利权人 西安交通大学
地址 710049 陕西省西安市碑林区咸宁西路28号

(72)发明人 朱正东 田靖轩 郭辉 李少辉 王鹏博 韩靖雯 李小轩 张小雨

(74)专利代理机构 西安通大专利代理有限责任公司 61200
代理人 王艾华

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

(56)对比文件
US 8108844 B2,2012.01.31
CN 108241530 A,2018.07.03
CN 108415761 A,2018.08.17
CN 108681773 A,2018.10.19
易树平、谭明智、郭宗林、温沛涵、周佳.云制造服务平台中的制造任务分解模式优化.《计算机集成制造系统》.2015,第21卷(第8期),
李珊珊.面向异构平台的Kernel合并优化及编译技术研究.《中国优秀硕士学位论文全文数据库 信息科技辑》.2016,(第9期),

审查员 李金蔓

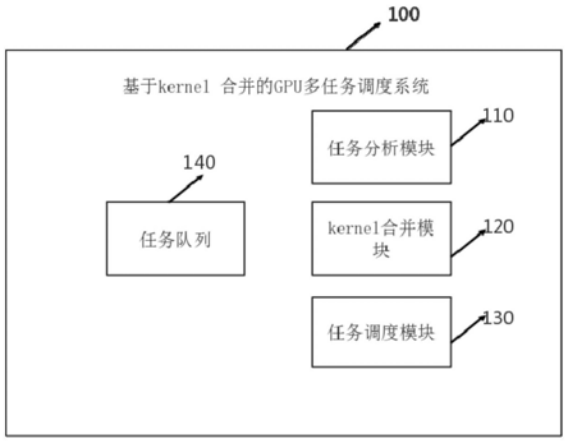
权利要求书3页 说明书7页 附图3页

(54)发明名称

一种基于Kernel合并的GPU任务调度系统及方法

(57)摘要

本发明公开了一种基于Kernel合并的GPU多任务调度系统及方法。包括任务分析模块、Kernel合并模块、任务调度模块和任务存储队列。用户提交GPU任务至任务存储队列,任务分析模块计算任务间的合并加速比,任务调度模块计算最优的合并调度顺序,并交至Kernel合并模块合并执行,获得最大任务吞吐率;本发明采用一般图的最大匹配计算最优合并调度顺序。通过任务分析模块计算任务的合并加速比;其次根据任务合并加速比构造无向图;通过二分图匹配算法计算最优的任务合并调度顺序;根据顺序调度任务组完成任务计算。本发明完成了上述功能的设计细节、实现算法和编码工作,提升了多任务下GPU的资源利用率。



CN 109522108 B

1.一种基于Kernel合并的GPU任务调度系统,其特征在于,所述任务调度系统部署于单个异构计算节点,异构计算节点至少包括一块CPU和GPU;所述任务调度系统包括任务存储队列、任务分析模块、Kernel合并模块、任务调度模块四个组件,按顺序关系完成工作;具体包括:

任务分析模块,用于接收用户提交的GPU任务,分析任务类型及输入数据规模,计算任务间的合并加速比,并构造所述任务的合并加速比矩阵Speed[i][j],存储两个GPU任务的合并执行加速比;

Kernel合并模块,用于接收两个GPU计算任务,对两个GPU计算任务的Kernel函数进行合并,形成一个大Kernel函数,等效于在GPU上实现Kernel函数的并发执行,同时接收所述任务的输入数据,向用户返回所述任务的计算结果,并给出执行时间等调试信息;

任务调度模块,用于根据所述任务分析模块构造的任务合并加速比矩阵Speed[i][j],构造任务加速关系无向图G(E,V,w),其中E表示无向图的边集,V表示无向图的点集,w表示无向图的权值集,计算最优的计算任务合并对,将任务对传送给所述Kernel合并模块进行合并计算,获得最大任务吞吐率;

任务存储队列,用于接收用户提交的GPU任务请求,将所述任务的Kernel函数发送至所述任务调度模块;

所述任务调度模块根据所述任务分析模块构造的任务合并加速比矩阵Speed[i][j],构造任务加速关系无向图G(E,V,w),计算最优的计算任务合并对;

所述任务加速关系无向图G(E,V,w)描述了任务队列中任务的合并加速关系,所述无向图顶点表示每个待执行的GPU任务,所述无向图的边描述两个任务之间的加速关系,无向图边权描述两个任务合并执行后可获得的加速比,无边表示计算任务合并后无法获得加速,无向图通过邻接表形式存储;

所述任务调度模块通过计算无向图G(E,V,w)的最大带权匹配,计算任务的最优合并对,其过程如下:

1)将所有计算任务节点标记为尚未访问;

2)若计算任务节点尚未访问,执行(3),否则结束;

3)若当前计算节点还未匹配,则拓展增广路径,转至(3);否则执行(4);

4)当前计算节点已经匹配,则找到一条增广路径,转至(2);

5)若发现图中奇环,则将奇环内所有任务节点视为一个大点,转至(2);

所述调度器依据任务加速比无向图G(E,V,w),求解最优的任务合并调度序列;进一步包括:

1)所述调度器所述任务调度模块根据所述任务分析模块构造的任务合并加速比矩阵Speed[i][j],构造任务加速关系无向图G(E,V,w),通过计算二分图最大带权匹配求解最优的计算任务合并对;

2)所述调度器选取一个任务合并对,传输给位于GPU端的所述Kernel合并模块进行合并计算;

3)所述Kernel合并模块接收两个候选执行任务的计算数据,进行计算并返回计算结果和计算时间;

执行2),直到所有任务计算结束。

2.根据权利要求1所述的基于Kernel合并的GPU任务调度系统，其特征在于，所述GPU任务是指通过OpenCL编写的GPU内核任务，包括：

任务Kernel函数，是用户基于OpenCL编写的需要在GPU上执行的具体任务代码；

任务输入数据，是用户所提交的任务输入。

3.根据权利要求1所述的基于Kernel合并的GPU任务调度系统，其特征在于，所述任务分析模块通过以下步骤构造所述任务的合并加速比矩阵Speed[i][j]：

1)从所述任务存储队列接收两个GPU计算任务，记作Q_i,Q_j,任务分析模块调用计算源评估两个任务单独串行执行及合并执行时间；

2)计算任务加速比，计算公式为：

$$Speed\_up_{ij} = \frac{T_i + T_j}{MergeTime_{ij}}$$

其中，$Speed_up_{ij}$表示任务Q_i,Q_j串行执行于合并执行的加速比，T_i,T_j表示任务Q_i,Q_j单独执行时间，$MergeTime_{ij}$表示任务Q_i,Q_j合并执行时间；

3)将加速比$Speed_up_{ij}$写入合并加速比矩阵Speed[i][j]，其中若$Speed_up_{ij}>1$则直接写入，否则，向Speed[i][j]写入-1，表示无法获得加速比。

4.根据权利要求1所述的基于Kernel合并的GPU任务调度系统，其特征在于，所述Kernel合并模块包括：

GPU任务预处理模块，统计GPU任务的计算特征，所述计算特征包括工作组数、全局维度、工作组维度，以及生成KernelId；

GPU调度内核，是基于OpenCL的GPU端的持久化Kernel任务，用于接收两个等待合并Kernel任务，按照先进先出方法调度两个Kernel任务的工作组轮询执行，完成Kernel的合并，返回任务执行结果并给出执行时间等调试信息，具体步骤包括：

1)所述GPU调度内核从工作组队列中获取一个工作组及工作组的相关参数，装载至所述GPU调度内核的工作组中；

2)计算原始工作组到所述GPU调度内核的坐标偏移，建立原始工作组到所述GPU调度内核的映射；

3)调度所述GPU调度内核的工作组完成工作组数据计算；

4)转至1)，直到所有工作组调度完毕。

5.一种基于Kernel合并的GPU任务调度方法，其特征在于，所述GPU任务调度方法包括：

用户提交一批待处理GPU计算任务至所述任务队列，分别记作Q_1～Q_n，调用计算资源评估任意两个任务单独串行执行及合并执行时间，计算加速比，并构建加速比矩阵Speed[i][j]，等待计算调度顺序；

通过加速比矩阵Speed[i][j]，构造任务加速比无向图G(E,V,w)，通过计算所述无向图G(E,V,w)的带权最大匹配，求出待执行任务Q_1～Q_n最优合并调度顺序；

根据求得的任务调度顺序，选取任务对$\langle Q_i,Q_j\rangle$，对所述任务对的每个任务进行计算特征抽取，将所述任务的Kernel函数合并为大Kernel函数，传入输入数据，完成任务对$\langle Q_i,Q_j\rangle$的合并执行，返回任务计算结果以及计算时间信息。

6.根据权利要求5所述的基于Kernel合并的GPU任务调度方法，其特征在于，通过合并

加速比计算合并顺序,合并加速比计算公式为:

$$Speed\_up_{ij} = \frac{T_i + T_j}{MergeTime_{ij}}$$

其中,$Speed_up_{ij}$表示任务Q_i,Q_j串行执行于合并执行的加速比,T_i,T_j表示任务Q_i,Q_j单独执行时间,$MergeTime_{ij}$表示任务Q_i,Q_j合并执行时间。

7.根据权利要求5所述的基于Kernel合并的GPU任务调度方法,其特征在于,所述任务合并加速比矩阵Speed[i][j]存储了任务Q_i,Q_j的合并执行加速比$Speed_up_{ij}$,Speed[i][j]=-1表示任务Q_i,Q_j合并后不能获得加速。

8.根据权利要求5所述的基于Kernel合并的GPU任务调度方法,其特征在于,所述任务加速关系无向图G(E,V,w)描述了任务队列中任务的合并加速关系,所述无向图顶点表示每个待执行的GPU任务,所述无向图的边描述两个任务之间的加速关系,所述无向图边权描述两个任务合并执行后可获得的加速比,无边表示计算任务合并后无法获得加速,所述无向图通过邻接表形式存储。

一种基于Kernel合并的GPU任务调度系统及方法

技术领域

[0001] 本发明属于计算机应用领域,具体涉及一种基于Kernel合并的 GPU任务调度系统及方法。

背景技术

[0002] 异构并行编程模型随着GPU的发展而兴起,由于GPU保留了许多流式处理器的特征,因此早期的异构并行编程模型秉承了流式编程思想.流(stream)是这类编程模型的核心,stream是一组数据的集合,计算在stream的每个数据元素上并行执行,符合单指令多数据(SIMD) 或多指令多数据(MIMD)的执行模式。在这种背景下,伴随着NVIDIA GPU在商业上获得的巨大成功,2007年出现的CUDA。然而,CUDA 仅能在以NVIDIA GPU为加速设备的异构系统中使用,于是,2008年底出现了多家公司共同制定的跨平台异构并行编程框架标准OpenCL, 它适用于任何并行系统。OpenCL将实际的不同硬件平台抽象为一个统一的平台模型。

[0003] 随着光刻技术不断发展,单个GPU存储体系日趋复杂,单片计算单元数量不断提升使得单GPU算力得到了极大的增强。对于一般地可加速并行任务,则必须针对硬件及任务特点,采用一定的编程策略才能最大化发挥硬件性能。另外,单个计算任务往往很难均衡地发挥单个芯片的计算资源,需要使用一定的策略使计算任务能并发地执行,提升任务吞吐率以及GPU资源利用率;在上述前提下,如何合理安排任务并发,以获得多任务下最大吞吐率也是亟待解决的问题。

发明内容

[0004] 本发明的目的在于提供一种基于Kernel合并GPU任务调度系统及方法。该系统主要包括任务分析模块、Kernel合并模块、任务调度模块和任务存储队列四个部分。本方法采用开放编程语言OpenCL,降低Kernel任务执行的时间,提高GPU资源利用率。本发明给出一种通过构建无向图描述任务加速比的方法,调度器通过求解最大匹配问题,得到最优的GPU任务合并调度方式,以获得最大的任务吞吐率。本方法为用户提供API接口,并完成了上述功能的设计细节、实现算法以及编码工作。本发明实现了基于Kernel合并的多GPU任务调度系统,屏蔽底层设备的差异性,并且该系统架构具有充分的灵活性和可扩展性。

[0005] 为了实现上述目的,本发明采用如下技术方案:

[0006] 一种基于Kernel合并的GPU任务调度系统,所述任务调度系统部署于单个异构计算节点,所述异构计算节点至少包括一块CPU和 GPU;所述任务调度系统包括任务存储队列、任务分析模块、Kernel 合并模块、任务调度模块四个组件,按顺序关系完成工作;具体包括:

[0007] 任务分析模块,用于接收用户提交的GPU任务,分析任务类型及输入数据规模,计算任务间的合并加速比,并构造所述任务的合并加速比矩阵Speed[i][j],存储两个GPU任务的合并执行加速比;

[0008] Kernel合并模块，用于接收两个GPU计算任务，对两个GPU计算任务的Kernel函数进行合并，形成一个大Kernel函数，等效于在GPU 上实现Kernel函数的并发执行，同时接收所述任务的输入数据，向用户返回所述任务的计算结果，并给出执行时间等调试信息；

[0009] 任务调度模块，用于根据所述任务分析模块构造的任务合并加速比矩阵Speed[i][j]，构造任务加速关系无向图G(E,V,w)，其中E表示无向图的边集，V表示无向图的点集，w表示无向图的权值集，计算最优的计算任务合并对，将任务对传送给所述Kernel合并模块进行合并计算，获得最大任务吞吐率；

[0010] 任务存储队列，用于接收用户提交的GPU任务请求，将所述任务的Kernel函数发送至所述任务调度模块；

[0011] 可选地，所述GPU任务是指通过OpenCL编写的GPU内核任务，包括：

[0012] 任务Kernel函数，是用户基于OpenCL编写的需要在GPU上执行的具体任务代码；

[0013] 任务输入数据，是用户所提交的任务输入。

[0014] 可选地，所述任务分析模块通过以下步骤构造所述任务的合并加速比矩阵Speed[i][j]：

[0015] 接收两个GPU计算任务，记作Q_i，Q_j，任务分析模块调用计算源评估两个任务单独串行执行及合并执行时间；

[0016] 计算任务加速比，计算公式为：

[0017] $$Speed\_up_{ij} = \frac{T_i + T_j}{MergeTime_{ij}}$$

[0018] 其中，$Speed_up_{ij}$表示任务Q_i，Q_j串行执行于合并执行的加速比，T_i，T_j表示任务Q_i，Q_j单独执行时间，$MergeTime_{ij}$表示任务Q_i，Q_j合并执行时间。

[0019] 将加速比$Speed_up_{ij}$写入合并加速比矩阵Speed[i][j]，其中若

[0020] $Speed_up_{ij}>1$则直接写入，否则，向Speed[i][j]写入-1，表示无法获得加速比。

[0021] 可选地，所述Kernel合并模块包括：

[0022] GPU任务预处理模块，将GPU任务的计算特征抽象封装。所述计算特征包括工作组数、全局维度、工作组维度，以及生成Kernel Id；

[0023] GPU调度内核，是基于OpenCL的GPU端的持久化Kernel任务，用于接收两个等待合并Kernel任务，按照先进先出方法调度两个 Kernel任务的工作组轮询执行，完成Kernel的合并，返回任务执行结果并给出执行时间等调试信息，具体步骤包括：

[0024] 1)所述GPU调度内核从工作组队列中获取一个工作组及工作组的相关参数，装载至所述GPU调度内核的工作组中；

[0025] 2)计算原始工作组到所述GPU调度内核的坐标偏移，建立原始工作组到所述GPU调度内核的映射；

[0026] 3)调度所述GPU调度内核的工作组完成工作组数据计算；

[0027] 4)转至1)，直到所有工作组调度完毕。

[0028] 可选地，所述任务调度模块根据所述任务分析模块构造的任务合并加速比矩阵Speed[i][j]，构造任务加速关系无向图G(E,V,w)，计算最优的计算任务合并对；

[0029] 所述任务加速关系无向图G(E,V,w)描述了任务队列中任务的合并加速关系，所述

无向图顶点表示每个待执行的GPU任务，所述无向图的边描述两个任务之间的加速关系，所述无向图边权描述两个任务合并执行后可获得的加速比，无边表示计算任务合并后无法获得加速，所述无向图通过邻接表形式存储；

[0030] 所述任务调度模块通过计算无向图G(E,V,w)的最大带权匹配，计算任务的最优合并对，其过程如下：

[0031] 1)将所有计算任务节点标记为尚未访问；

[0032] 2)若计算任务节点尚未访问，执行(3)，否则结束；

[0033] 3)若当前计算节点还未匹配，则拓展增广路径，转至(3)；否则执行(4)；

[0034] 4)当前计算节点已经匹配，则找到一条增广路径，转至(2)；

[0035] 5)若发现图中奇环，则将奇环内所有任务节点视为一个大点，转至(2)；

[0036] 所述调度器依据任务加速比无向图G(E,V,w)，求解最优的任务合并调度序列；进一步包括：

[0037] 1)所述调度器所述任务调度模块根据所述任务分析模块构造的任务合并加速比矩阵Speed[i][j]，构造任务加速关系无向图G(E,V,w)，通过计算二分图最大带权匹配求解最优的计算任务合并对；

[0038] 2)所述调度器选取一个任务合并对，传输给位于GPU端的所述Kernel 合并模块进行合并计算；

[0039] 3)所述Kernel合并模块接收两个候选执行任务的计算数据，进行计算并返回计算结果和计算时间；

[0040] 4)执行2)，直到所有任务计算结束；

[0041] 一种基于Kernel合并的GPU任务调度方法，所述GPU任务调度方法包括：

[0042] 用户提交一批待处理GPU计算任务至任务队列，记作Q_1～Q_n，调用计算源评估任意两个任务单独串行执行及合并执行时间，计算加速比，并构建加速比矩阵Speed[i][j]，等待计算调度顺序；

[0043] 通过加速比矩阵Speed[i][j]，构造任务加速比无向图G(E,V,w)，通过计算所述无向图G(E,V,w)的带权最大匹配，求出待执行任务 Q_1～Q_n最优合并调度顺序；

[0044] 根据求得的任务调度顺序，选取任务对$<Q_i,Q_j>$，对所述任务对的每个任务进行计算特征抽取，将所述任务的Kernel函数合并为大 Kernel函数，传入输入数据，完成任务对$<Q_i,Q_j>$的合并执行，返回任务计算结果以及计算时间信息。

[0045] 可选地，通过合并加速比计算合并顺序，合并加速比计算公式为：

[0046] $$Speed\_up_{ij} = \frac{T_i + T_j}{MergeTime_{ij}}$$

[0047] 其中，$Speed_up_{ij}$表示任务Q_i，Q_j串行执行于合并执行的加速比， T_i，T_j表示任务Q_i，Q_j单独执行时间，$MergeTime_{ij}$表示任务Q_i，Q_j合并执行时间。

[0048] 可选地，所述任务合并加速比矩阵Speed[i][j]存储了任务Q_i，Q_j的合并执行加速比$Speed_up_{ij}$，Speed[i][j]＝-1表示任务Q_i，Q_j合并后不能获得加速。

[0049] 可选地，所述任务加速关系无向图G(E,V,w)描述了任务队列中任务的合并加速关系，所述无向图顶点表示每个待执行的GPU任务，所述无向图的边描述两个任务之间的加速关系，所述无向图边权描述两个任务合并执行后可获得的加速比，无边表示计算任务合并

后无法获得加速，所述无向图通过邻接表形式存储。

[0050] 与现有技术相比，本发明具有以下有益效果：

[0051] 首先，本方法屏蔽底层了设备的差异性，并且该系统架构具有充分的灵活性和可扩展性。其次，本方法利用了CPU控制程序执行流程并处理基础数据和GPU处理密集的浮点运算任务相结合的模式提高了大量计算任务的计算速度；其次，通过GPU端的调度Kernel将两个候选Kernel进行合并，形成一个大的Kernel，等效实现了GPU 上多个Kernel函数的并发执行，提升了GPU的资源利用率；再者，调度器通过计算二分图最大匹配，求出最优的任务调度合并顺序，提升了整体任务的加速比。

附图说明

[0052] 图1为本发明的一种基于Kernel合并的GPU任务调度方法的流程示意图。

[0053] 图2为本发明的一种基于Kernel合并的GPU调度系统的原理框架图。

[0054] 图3为本发明的一种基于Kernel合并的GPU调度方法及系统的实施过程示意图。

[0055] 图4为本发明的一种基于Kernel合并的GPU调度系统任务分析模块程示意图。

[0056] 图5为本发明的一种基于Kernel合并的GPU调度系统任务调度模块程示意图。

具体实施方式

[0057] 以下通过特定的具体实例说明本发明的实施方式，本领域技术人员可有本说明书所揭露的内容轻易地了解本发明的其他优点与功效。本发明还可以通过另外不同的具体实施方式加以实施或应用，本说明书的各项细节也可以基于不同观点与应用，在没有背离 本发明的精神下进行各种修饰或改变。

[0058] 本实施例的目的在于提供一种基于Kernel合并的GPU任务调度方法及系统，用于解决现有技术中GPU调度资源利用率低，吞吐率低下的问题。以下将详细阐述本发明的一种基于Kernel合并的GPU 任务调度方法及系统的原理及实施方式，使本领域技术人员不需要创造性劳动即可理解本发明的一种基于Kernel合并的GPU任务调度方法及系统。

[0059] 具体地，本实施例旨在实现一个高性能的基于Kernel合并的GPU 调度系统，包括任务分析模块、Kernel合并模块、任务调度模块、任务存储队列，完成过GPU计算任务的提交、任务分析、计算调度合并顺序、合并执行等调度工作。便于开发人员在无需关心硬件系统以及调度细节的前提下获得高吞吐和高资源利用率。

[0060] 以下结合附图对本实施例的基于Kernel合并的GPU任务调度方法及系统做进一步详细描述。

[0061] 如图1所示，本实施例提供一种基于Kernel合并的GPU任务调度方法，所述GPU任务调度方法包括：

[0062] 步骤S101，用户提交一批待处理GPU计算任务Q_1～Q_n至任务队列。

[0063] 步骤S102，评估任意两个任务单独串行执行及合并执行时间，计算加速比，并构建加速比矩阵Speed[i][j]，等待计算调度顺序。

[0064] 步骤S103，通过加速比矩阵Speed[i][j]，构造任务加速比无向图 G(E,V,w)。

[0065] 步骤S104，计算所述无向图G(E,V,w)的带权最大匹配，求出待执行任务Q_1～Q_n最优合并调度顺序。

[0066] 步骤S105,根据求得的任务调度顺序,选取任务对<Q_i,Q_j>,对所述任务对的每个任务进行计算特征抽取,将所述两个任务的Kernel 函数合并为大Kernel函数。

[0067] 步骤S106,传入输入数据,完成任务对<Q_i,Q_j>的合并执行,返回任务计算结果以及计算时间信息。

[0068] 以下对本实施例的基于Kernel合并的GPU任务调度方法进行详细说明。

[0069] 由于Kernel任务在运行过程中对不同类型资源需求程度不同,将 Kernel合并为一个大Kernel函数后执行,可以最大程度提高资源的利用率,提升任务吞吐率。通过合并加速比计算合并顺序,合并加速比计算公式为:

[0070] $$Speed\_up_{ij} = \frac{T_i + T_j}{MergeTime_{ij}}$$

[0071] 其中,$Speed_up_{ij}$表示任务Q_i,Q_j串行执行于合并执行的加速比, T_i,T_j表示任务Q_i,Q_j单独执行时间,$MergeTime_{ij}$表示任务Q_i,Q_j合并执行时间。

[0072] 所述任务合并加速比矩阵Speed[i][j]存储了任务Q_i,Q_j的合并执行加速比$Speed_up_{ij}$,Speed[i][j]＝-1表示任务Q_i,Q_j合并后不能获得加速。

[0073] 所述任务加速关系无向图G(E,V,w)描述了任务队列中任务的合并加速关系,所述无向图顶点表示每个待执行的GPU任务,所述无向图的边描述两个任务之间的加速关系,所述无向图边权描述两个任务合并执行后可获得的加速比,无边表示计算任务合并后无法获得加速,所述无向图通过邻接表形式存储。

[0074] 所述步骤S104通过计算无向图G(E,V,w)的最大带权匹配,计算任务的最优合并对,最大化加速比和,从而的到最优的GPU任务合并调度执行顺序,其过程如下:

[0075] 1)将所有计算任务节点标记为尚未访问;

[0076] 2)若计算任务节点尚未访问,执行(3),否则执行(5);

[0077] 3)若当前计算节点还未匹配,则拓展增广路径,转至(3);否则执行4);

[0078] 4)当前计算节点已经匹配,则找到一条增广路径,转至(2);

[0079] 5)若发现图中奇环,则将奇环内所有任务节点视为一个大点。

[0080] 所述步骤S105从步骤S104计算得到的合并任务顺序中接收两个等待合并Kernel任务,按照先进先出方法调度两个Kernel任务的工作组轮询执行,完成Kernel的合并,返回任务执行结果并给出执行时间等调试信息,具体步骤包括:

[0081] 1)GPU调度内核从工作组队列中获取一个工作组及工作组的相关参数,装载至所述GPU调度内核的工作组中;

[0082] 2)计算原始工作组到GPU调度内核的坐标偏移,建立原始工作组到所述GPU调度内核的映射;

[0083] 3)调度所述GPU调度内核的工作组完成工作组数据计算;

[0084] 4)转至1),直到所有工作组调度完毕。

[0085] 具体地,对于所有GPU计算任务Q_1～Q_n,本实施例提供的基于 Kernel合并的GPU调度方法将提高任务吞吐率,提升GPU资源利用率。

[0086] 为了实现上述基于Kernel合并的GPU调度方法,本实施例提供了一种基于Kernel合并的GPU调度系统100,如图2所示,所述GPU 调度系统100包括:任务分析模块110、Kernel

合并模块120、任务调度模块130、任务存储队列140。下面对本实施例进行详细说明。

[0087] 所述基于Kernel合并的GPU任务调度系统100，部署于单个异构计算节点，所述异构计算节点至少包括一块CPU和GPU，所述CPU 和GPU均需要支持OpenCL计算平台。还包括：

[0088] 任务分析模块110，用于接收用户提交的GPU任务，分析任务类型及输入数据规模，计算任务间的合并加速比，并构造所述任务的合并加速比矩阵Speed[i][j]；

[0089] Kernel合并模块120，用于接收两个GPU计算任务，对所述任务的Kernel函数进行合并，形成一个大Kernel函数，等效于在GPU上实现Kernel函数的并发执行，同时接收所述任务的输入数据，向用户返回所述任务的计算结果，并给出执行时间等调试信息；

[0090] 任务调度模块130，用于根据110构造的任务合并加速比矩阵 Speed[i][j]，构造任务加速关系无向图G(E,V,w)，计算最优的计算任务合并对，将任务对传送给120进行合并计算，获得最大任务吞吐率；

[0091] 任务存储队列140，用于接收用户提交的GPU任务请求，将所述任务的Kernel函数发送至130；

[0092] 图4描述了所述任务分析模块执行示意，具体地，所述任务分析模块110通过以下步骤构造所述任务的合并加速比矩阵Speed[i][j]：

[0093] 1)接收两个GPU计算任务Q_i，Q_j，任务分析模块调用计算源评估两个任务单独串行执行及合并执行时间；

[0094] 2)计算任务加速比，计算公式为：

[0095] $$Speed\_up_{ij} = \frac{T_i + T_j}{MergeTime_{ij}}$$

[0096] 其中，$Speed_up_{ij}$表示任务Q_i，Q_j串行执行于合并执行的加速比，T_i，T_j表示任务Q_i，Q_j单独执行时间，$MergeTime_{ij}$表示任务Q_i，Q_j合并执行时间。

[0097] 3)将加速比$Speed_up_{ij}$写入合并加速比矩阵Speed[i][j]，其中若 $Speed_up_{ij} > 1$则直接写入，否则，向Speed[i][j]写入-1，表示无法获得加速比。

[0098] 所述Kernel合并模块120包括GPU任务预处理模块，将GPU 任务的计算特征抽象封装。所述计算特征包括工作组数、全局维度 xDim*yDim、工作组维度xThreads*yThreads，以及生成Kernel Id；GPU调度内核接收两个等待合并Kernel任务，按照先进先出方法调度两个Kernel任务的工作组轮询执行，完成Kernel的合并，返回任务执行结果并给出执行时间等调试信息，具体步骤包括：

[0099] 1)所述GPU调度内核从工作组队列中获取一个工作组及工作组的相关参数，装载至所述GPU调度内核的工作组中；

[0100] 2)计算原始工作组到所述GPU调度内核的坐标偏移，建立原始工作组到所述GPU调度内核的映射；

[0101] 3)调度所述GPU调度内核的工作组完成工作组数据计算；

[0102] 4)转至1)，直到所有工作组调度完毕。

[0103] 本实施例中，所述任务调度模块130根据所述任务分析模块构造的任务合并加速比矩阵Speed[i][j]，构造任务加速关系无向图 G(E,V,w)，计算最优的计算任务合并对，如图5所示；

[0104] 所述任务加速关系无向图G(E,V,w)描述了任务队列中任务的合并加速关系，所述

无向图顶点表示每个待执行的GPU任务，所述无向图的边描述两个任务之间的加速关系，所述无向图边权描述两个任务合并执行后可获得的加速比，无边表示计算任务合并后无法获得加速，所述无向图通过邻接表形式存储；

[0105] 所述调度器依据任务加速比无向图G(E,V,w)，求解最优的任务合并调度序列；进一步包括：

[0106] 1)所述调度器所述任务调度模块根据所述任务分析模块构造的任务合并加速比矩阵Speed[i][j]，构造任务加速关系无向图G(E,V,w)，通过计算二分图最大带权匹配求解最优的计算任务合并对；

[0107] 2)所述调度器选取一个任务合并对，传输给位于GPU端的所述Kernel 合并模块进行合并计算；

[0108] 3)所述Kernel合并模块接收两个候选执行任务的计算数据，进行计算并返回计算结果和计算时间；

[0109] 4)执行2)，直到所有任务计算结束；

[0110] 为了使本领域技术人员进一步理解本实施例中的基于Kernel合并的任务调度系统100，如图3所示，以下以具体实例说明本实施例中GPU任务调度系统100和方法的实施过程。

[0111] 1.用户编写GPU计算任务：用户按照GPU编程规范，编写能在 GPU上运行的Kernel程序。

[0112] 2.用户提交应用请求：用户将任务提交至任务队列140。

[0113] 3.任务分析模块110分析任务：任务分析模块110分析任务存储队列140中的任务，构造任务加速比矩阵。

[0114] 4.任务调度模块130构造任务加速无向图：任务调度模块130根据任务分析模块110求得的任务加速比矩阵，构造任务加速无向图，以邻接表存储。

[0115] 5.任务调度模块130计算任务合并调度顺序：任务调度模块130 根据任务加速无向图，计算任务合并调度顺序。

[0116] 6.Kernel合并模块120完成任务合并：Kernel合并模块120根据任务调度模块130计算任务合并调度顺序，取两个待合并 Kernel合并成一个大Kernel函数。

[0117] 7.Kernel合并模块120执行合并任务：Kernel合并模块120调度执行合并任务，返回计算结果和计算时间。

[0118] 以上实施例仅例示说明了本发明的原理及功效，而非用于闲置本发明。任何熟悉此技术的人士皆可在不违背本发明的精神及范畴下，对上述实施例进行修饰或者改变。因此，举凡所记述领域中具有通常知识者在未脱离本发明所揭示的精神与技术思想下所完成的一切等效修饰或改变，仍应由本发明的权利要求所涵盖。

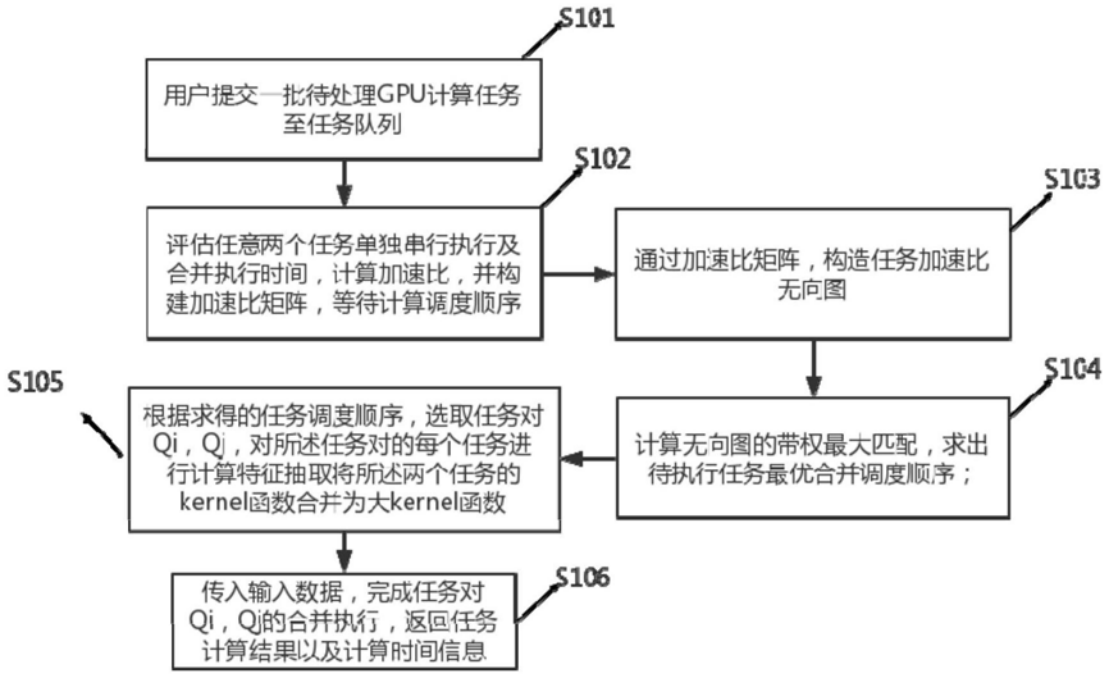

S101
用户提交一批待处理GPU计算任务至任务队列
S102
评估任意两个任务单独串行执行及合并执行时间，计算加速比，并构建加速比矩阵，等待计算调度顺序
S103
通过加速比矩阵，构造任务加速比无向图
S104
计算无向图的带权最大匹配，求出待执行任务最优合并调度顺序；
S105
根据求得的任务调度顺序，选取任务对Qi，Qj，对所述任务对的每个任务进行计算特征抽取将所述两个任务的kernel函数合并为大kernel函数
S106
传入输入数据，完成任务对Qi，Qj的合并执行，返回任务计算结果以及计算时间信息

图1

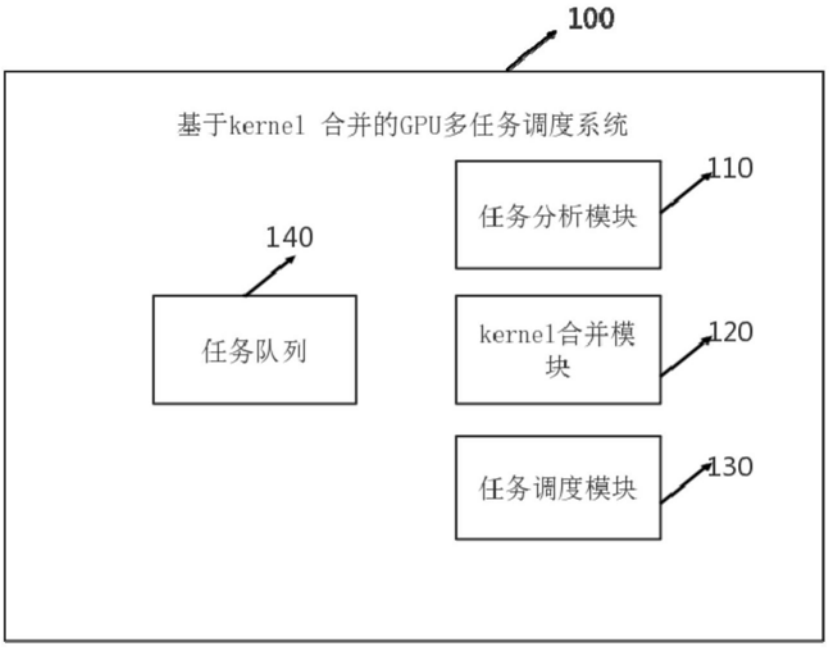

100
基于kernel 合并的GPU多任务调度系统
110
任务分析模块
140
任务队列
kernel合并模块
120
任务调度模块
130

图2

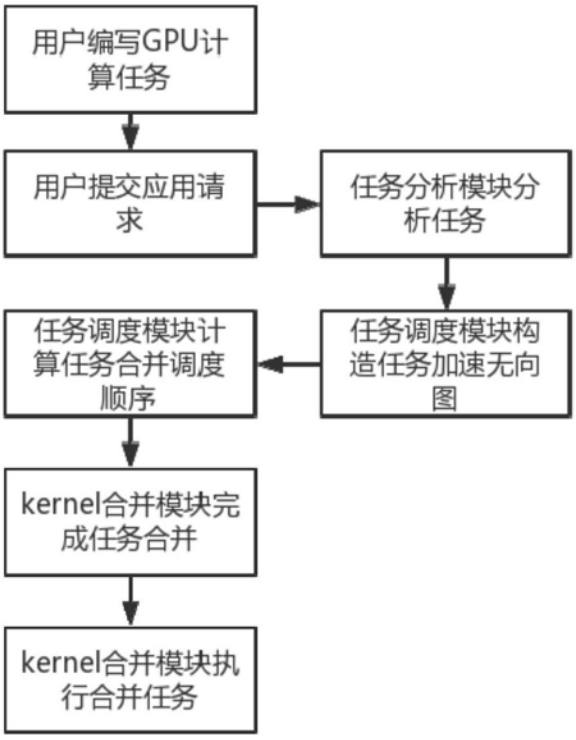

用户编写GPU计算任务
用户提交应用请求
任务分析模块分析任务
任务调度模块构造任务加速无向图
任务调度模块计算任务合并调度顺序
kernel合并模块完成任务合并
kernel合并模块执行合并任务

图3

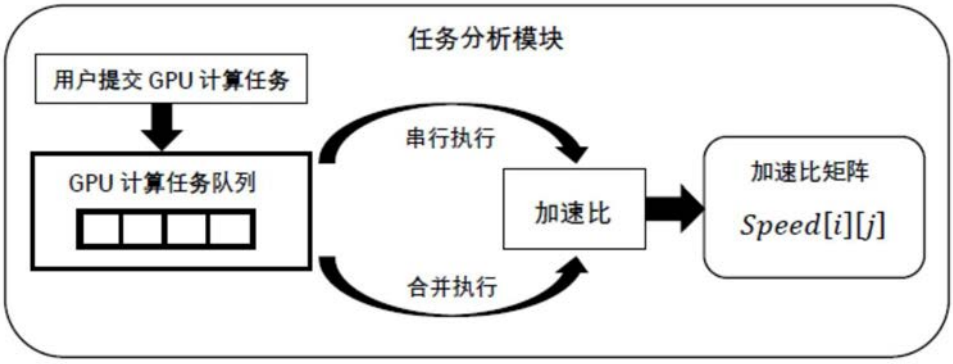

任务分析模块
用户提交 GPU 计算任务
GPU 计算任务队列
串行执行
合并执行
加速比
加速比矩阵
Speed[i][j]

图4

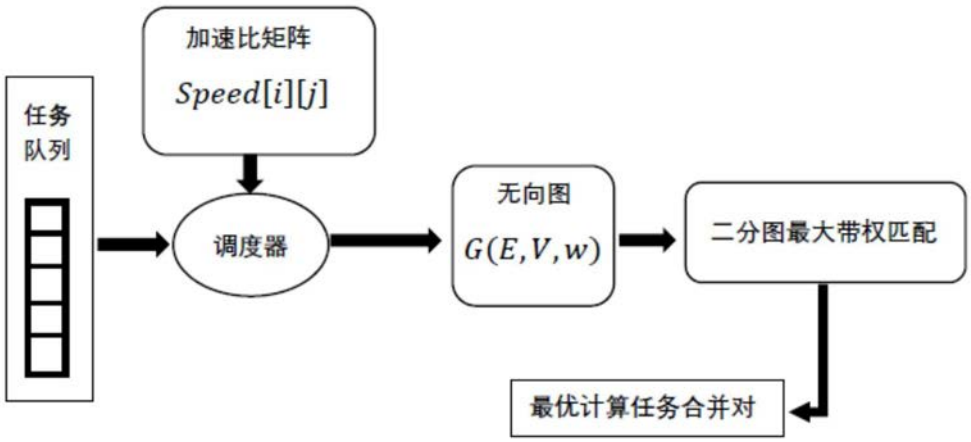


图5