(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 105183562 B
(45)授权公告日 2018.09.11

(21)申请号 201510566712.4

(22)申请日 2015.09.09

(65)同一申请的已公布的文献号
申请公布号 CN 105183562 A

(43)申请公布日 2015.12.23

(73)专利权人 合肥芯碁微电子装备有限公司
地址 230088 安徽省合肥市高新区创新大道2800号创新产业园二期H2楼533室

(72)发明人 陆敏婷

(74)专利代理机构 合肥天明专利事务所(普通合伙) 34115
代理人 张祥骞 奚华保

(51)Int.Cl.
G06F 9/50(2006.01)
G06T 1/20(2006.01)

(56)对比文件
CN 103208103 A,2013.07.17,
CN 104657219 A,2015.05.27,
CN 103559018 A,2014.02.05,
US 2007296725 A1,2007.12.27,

审查员 王艳臣

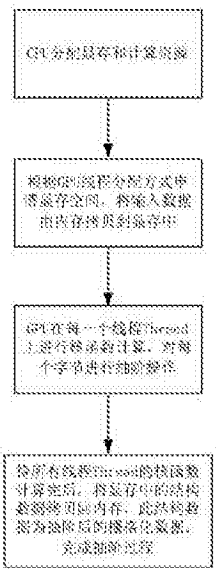
权利要求书2页 说明书5页 附图2页

(54)发明名称

一种基于CUDA技术对栅格化数据进行抽阶的方法

(57)摘要

本发明涉及一种基于CUDA技术对栅格化数据进行抽阶的方法，与现有技术相比解决了栅格化数据抽阶效率较低的缺陷。本发明包括以下步骤：CPU分配显存和计算资源，CPU根据进行抽阶数据的规模，结合当前GPU可供使用的硬件资源，计算出最优的GPU线程分配方式；根据GPU线程分配方式申请显存空间，将输入数据由内存拷贝到显存中；GPU在每一个线程Thread上进行核函数计算，对每个字节进行抽阶操作；完成抽阶过程。本发明通过并行化提高了计算效率，增加了直写式光刻机的产能，同时降低了数据规模，减少了对计算能力及传输带宽的依赖，降低了成本。



CN 105183562 B

1.一种基于CUDA技术对栅格化数据进行抽阶的方法,其特征在于,包括以下步骤:

11)CPU分配显存和计算资源,CPU根据进行抽阶数据的规模,结合当前GPU可供使用的硬件资源,计算出最优的GPU线程分配方式;

12)根据GPU线程分配方式申请显存空间,将输入数据由内存拷贝到显存中;

13)GPU在每一个线程Thread上进行核函数计算,对每个字节进行抽阶操作;所述的GPU在每一个线程Thread上进行核函数计算包括以下步骤:

131)计算当前线程偏移量,根据当前线程的栅格坐标计算全局线程编号,通过全局线程编号计算出当前线程在缓冲区中的偏移量;

132)根据线程的偏移量从显存中取出数据存入线程块内共享显存;

133)针对满足tid.x=0的所有线程来执行其所在二维线程块Block输入数据的抽阶操作;根据栅格化处理规则从每一个字节取出特定的位,将抽阶的结果数据暂存到结果缓存mask中,其中设缓存mask为4个字节;

134)设int型为4字节,使用tid.x<4的线程Thread将结果数据从结果缓存mask中拷贝至对应的显存中;

14)待所有线程Thread的核函数计算完后,将显存中的结构数据拷贝回内存,此结构数据为抽阶后的栅格化数据,完成抽阶过程。

2.根据权利要求1所述的一种基于CUDA技术对栅格化数据进行抽阶的方法,其特征在于,所述的CPU分配显存和计算资源包括以下步骤:

21)输入栅格化处理后的二维位图像素阵列,其宽度定义为width,高度定义为height;

22)将每个二维线程块Block的宽度定义为blockDim.x、高度定义为blockDim.y;

23)计算出线程栅格Grid的宽度gridDim.x,其计算公式如下:

gridDim.x = width/blockDim.x;

计算出线程栅格Grid的高度gridDim.y,其计算公式如下:

gridDim.y = height/blockDim.y;

计算出显存分配总大小length,其计算公式如下:

length=width *height * (1+N/8),

其中,N=1、2或4;

24)获得线程分配方式,

线程分配方式中二维线程块Block为Block(blockDim.x,blockDim.y);出线程栅格Grid为Grid(gridDim.x,gridDim.y)。

3.根据权利要求1所述的一种基于CUDA技术对栅格化数据进行抽阶的方法,其特征在于,所述的计算当前线程偏移量包括以下步骤:

31)计算当前线程所在块的编号bid,其计算公式如下:

bid=gridDim.x*blockIdx.y+blockIdx.x;

其中,blockIdx.y为当前线程所在线程块Block中的列号,blockIdx.x为当前线程所在线程块Block中的行号;

32)计算当前块内线程编号cur_tid,其计算公式如下:

cur_tid=blockDim.x*threadIdx.y+threadIdx.x;

其中,blockIdx.y为当前线程所在线程块Block中的列号,blockIdx.x为当前线程所在

线程块Block中的行号；

33)计算全局线程编号total_tid,其计算公式如下：

total_tid=bid*blockDim.x*blockDim.y+cur_tid;

34)根据全局线程编号确定当前线程输入输出数据在缓冲区中的偏移量offset,其计算公式如下：

offset=total_tid*(blockDim.x*blockDim.y)*(N/8)，

其中8是一个字节的位数,N为抽取的阶数,N=1、2或4。

一种基于CUDA技术对栅格化数据进行抽阶的方法

技术领域

[0001] 本发明涉及直写式光刻机数据处理技术，具体来说是一种基于CUDA技术对栅格化数据进行抽阶的方法。

背景技术

[0002] CUDA是NVIDIA公司2007年提出的支持GPU进行通用计算的编程模型和开发环境，CUDA编程的思想是用海量的线程来开发程序中的并行性，海量线程以层次化的方式组织，单个的线程被映射到标量核SP上执行，一组线程被组织成一个线程块(Block)被映射到一个流处理单位SM上执行，最后由线程块组成的线程栅格(Grid)映射到一个GPGPU(GPU)上执行。由于GPU具有远超CPU的计算核心数以及海量的并行计算资源，适合进行计算密集型、高度并行化的计算任务。同时，由于GPU的价格远远低于同等性能的并行计算系统，由CPU和GPGPU(GPU)组成的异构系统已经越来越广的应用到生物医学、流体力学等诸多工程应用领域。

[0003] 直写式光刻机的数据处理过程是将用户提供的矢量数据，转化为图形发生器能接受的图像数据，数据处理过程中涉及到数据的分析、计算和传输。目前实际应用处理得到的栅格化数据中，一个像素用一个字节来表示(8阶灰度)，而下位机只需要其中的1、2、4位即可满足显示的灰度要求，因此如果能够针对栅格化数据进行抽阶处理，去掉其中冗余的数据，提取出有效的灰度值，即可降低了数据规模，降低传输链路带宽。如针对同样一幅图，抽阶后的数据量变少了，传输这些数据要求的时间不变，所以需要的带宽(传输速率)降低了。实际应用中选择成本低，速度慢的传输链路也可满足传输时间要求，则相当于降低了生产成本。

[0004] 对栅格化数据进行抽阶处理是根据实际需要来处理，如针对4位的灰度要求，在对数据栅格化时便可以将需要的4位排列在8位字节(一个像素)的前4位，在做抽阶工作时，直接抽取0-3位即可；或针对2位的灰度要求，在数据栅格化时将需要的2位排在8位字节的第2位和第3位，在做抽阶工作时，直接抽取1-2位即可。但是目前栅格化数据的数据量过于庞大，导致抽阶工作较慢，抽阶工作的分析、计算和传输均比较耗时，难以满足产能要求，如何利用CUDA技术的特点，实现栅格化数据抽阶的多线程并行处理已经成为急需解决的技术问题。

发明内容

[0005] 本发明的目的是为了解决现有技术中栅格化数据抽阶效率较低的缺陷，提供一种基于CUDA技术对栅格化数据进行抽阶的方法来解决上述问题。

[0006] 为了实现上述目的，本发明的技术方案如下：

[0007] 一种基于CUDA技术对栅格化数据进行抽阶的方法，包括以下步骤：

[0008] CPU分配显存和计算资源，CPU根据进行抽阶数据的规模，结合当前GPU可供使用的硬件资源，计算出最优的GPU线程分配方式；

[0009] 根据GPU线程分配方式申请显存空间，将输入数据由内存拷贝到显存中；

[0010] GPU在每一个线程Thread上进行核函数计算，对每个字节进行抽阶操作；

[0011] 待所有线程Thread的核函数计算完后，将显存中的结构数据拷贝回内存，此结构数据为抽阶后的栅格化数据，完成抽阶过程。

[0012] 所述的CPU分配显存和计算资源包括以下步骤：

[0013] 输入栅格化处理后的二维位图像素阵列，其宽度定义为width，高度定义为height；

[0014] 将每个二维线程块Block的宽度定义为blockDim.x、高度定义为blockDim.y；

[0015] 计算出线程栅格Grid的宽度gridDim.x，其计算公式如下：

[0016] gridDim.x = width/blockDim.x；

[0017] 计算出线程栅格Grid的高度gridDim.y，其计算公式如下：

[0018] gridDim.y = height/blockDim.y；

[0019] 计算出显存分配总大小length，其计算公式如下：

[0020] length=width *height * (1+N/8)，

[0021] 其中，N=1、2或4；

[0022] 获得线程分配方式，

[0023] 线程分配方式中二维线程块Block为Block(blockDim.x,blockDim.y)；出线程栅格Grid为Grid(gridDim.x,gridDim.y)。

[0024] 所述的GPU在每一个线程Thread上进行核函数计算包括以下步骤：

[0025] 计算当前线程偏移量，根据当前线程的栅格坐标计算全局线程编号，通过全局线程编号计算出当前线程在缓冲区中的偏移量；

[0026] 根据线程的偏移量从显存中取出数据存入线程块内共享显存；

[0027] 针对满足tid.x=0的所有线程来执行其所在二维线程块Block输入数据的抽阶操作；根据栅格化处理规则从每一个字节取出特定的位，将抽阶的结果数据暂存到结果缓存mask中，其中设缓存mask为4个字节；

[0028] 设int型为4字节，使用tid.x<4的线程Thread将结果数据从结果缓存mask中拷贝至对应的显存中。

[0029] 所述的计算当前线程偏移量包括以下步骤：

[0030] 计算当前线程所在块的编号bid，其计算公式如下：

[0031] bid=gridDim.x*blockIdx.y+blockIdx.x；

[0032] 其中，blockIdx.y为当前线程所在线程块Block中的列号，blockIdx.x为当前线程所在线程块Block中的行号；

[0033] 计算当前块内线程编号cur_tid，其计算公式如下：

[0034] cur_tid=blockDim.x*threadIdx.y+threadIdx.x；

[0035] 其中，blockIdx.y为当前线程所在线程块Block中的列号，blockIdx.x为当前线程所在线程块Block中的行号；

[0036] 计算全局线程编号total_tid，其计算公式如下：

[0037] total_tid=bid*blockDim.x*blockDim.y+cur_tid；

[0038] 根据全局线程编号确定当前线程输入输出数据在缓冲区中的偏移量offset，其计

算公式如下：

[0039] offset=total_tid*(blockDim.x*blockDim.y)*(N/8)，

[0040] 其中8是一个字节的位数，N为抽取的阶数，N=1、2或4。

[0041] 有益效果

[0042] 本发明的一种基于CUDA技术对栅格化数据进行抽阶的方法，与现有技术相比通过并行化提高了计算效率，增加了直写式光刻机的产能，同时降低了数据规模，减少了对计算能力及传输带宽的依赖，降低了成本。

[0043] 本发明在直写式光刻机的数据处理过程中，发掘出了其中的计算密集型、高度并行化过程，并将这个过程并行化，通过CUDA将其部署到GPU上并行执行，极大的提高了处理速度。并且，在实现并行化的过程中，充分地利用了GPU以及CUDA框架的特性，实现了最大化的加速比；在分配线程资源时，根据当前硬件的最大线程数、线程块中的最优线程数来确定Block和Grid的尺寸；核函数在处理时先将输入数据拷贝到共享显存，充分利用了共享显存的高带宽特性，提高了处理速度；读写全局显存时根据线程编号同步操作，有效的屏蔽了访存延时，进一步提高了处理效率。

附图说明

[0044] 图1为本发明的方法顺序图；

[0045] 图2为本发明中CUDA线程栅格示意图。

具体实施方式

[0046] 为使对本发明的结构特征及所达成的功效有更进一步的了解与认识，用以较佳的实施例及附图配合详细的说明，说明如下：

[0047] 如图1所示，本发明所述的一种基于CUDA技术对栅格化数据进行抽阶的方法，抽阶的过程在GPU中并行进行，利用CUDA框架实现。其包括以下步骤：

[0048] 第一步，CPU分配显存和计算资源，CPU根据进行抽阶数据的规模，结合当前GPU可供使用的硬件资源，计算出最优的GPU线程分配方式，为后面的步骤分配显存和计算资源。其具体包括以下步骤：

[0049] (1)输入栅格化处理后的二维位图像素阵列，二维位图像素阵列即为栅格化数据，其宽度定义为width，高度定义为height。

[0050] (2)根据CUDA的技术要求，每个二维线程块Block的宽度为blockDim.x、高度为blockDim.y。

[0051] (3)计算出线程栅格Grid的宽度gridDim.x，其计算公式如下：

[0052] gridDim.x = width/blockDim.x。

[0053] 计算出线程栅格Grid的高度gridDim.y，其计算公式如下：

[0054] gridDim.y = height/blockDim.y。

[0055] 计算出显存分配总大小length，其计算公式如下：

[0056] length=width *height * (1+N/8)，

[0057] 其中，N=1、2或4。

[0058] 如图2所示，在此步骤后获得最优的GPU线程分配方式，在GPU(device端)共有

gridDim.x * gridDim.y 个二维线程块Block并行执行，每个二维线程块Block中有BlockDim.x * BlockDim.y个线程Thread并行执行，通过这种高度并行化极大地提高了执行效率。

[0059] (4)获得线程分配方式，

[0060] 线程分配方式中二维线程块Block为Block(blockDim.x,blockDim.y)；出线程栅格Grid为Grid(gridDim.x,gridDim.y)。

[0061] 第二步，根据GPU线程分配方式申请显存空间，将输入数据由内存拷贝到显存中，即将CPU工作移到GPU，在本发明中，在CPU端，利用多核处理器的多线程计算能力，使整个步骤构建成一个流水线结构。同时，根据GPU线程分配方式的计算结果，通过CUDA将此步骤部署到GPU的每一个grid、block以及thread，保证其高度并行化地执行。

[0062] 第三步，GPU在每一个线程Thread上进行核函数计算，核函数计算以多线程方式进行，在每个线程中独立运行着一个核函数，对每个字节进行抽阶操作。其具体步骤如下：

[0063] (1)计算当前线程偏移量。根据当前线程的栅格坐标计算全局线程编号，通过全局线程编号计算出当前线程在缓冲区中的偏移量。在流处理单元SM中并发的所有线程的执行顺序是按thread0、thread1...threadn的顺序依次执行的，在这个过程中，通过线程同步使得每个block中的线程并发进入访存语句，并按照线程号从小到大按顺序依次执行，使得thread0在等待访存结果时，thread1能够立即开始访存操作，依次类推，屏蔽了大部分线程的访存延时，节约了执行时间。而在此便需要对每个线程计算出其位于缓冲区中的偏移量，实现按照线程号从小到大按顺序依次执行。其具体步骤如下：

[0064] A、计算当前线程所在块的编号bid，其计算公式如下：

[0065] bid=gridDim.x*blockIdx.y+blockIdx.x；

[0066] 其中，blockIdx.y为当前线程所在线程块Block中的列号，blockIdx.x为当前线程所在线程块Block中的行号。

[0067] B、计算当前块内线程编号cur_tid，其计算公式如下：

[0068] cur_tid=blockDim.x*threadIdx.y+threadIdx.x；

[0069] 同样，blockIdx.y为当前线程所在线程块Block中的列号，blockIdx.x为当前线程所在线程块Block中的行号。

[0070] C、计算全局线程编号total_tid，其计算公式如下：

[0071] total_tid=bid*blockDim.x*blockDim.y+cur_tid。

[0072] D、根据全局线程编号total_tid确定当前线程输入输出数据在缓冲区中的偏移量offset，其计算公式如下：

[0073] offset=total_tid*(blockDim.x*blockDim.y)*(N/8)，

[0074] 其中8是一个字节的位数，N为抽取的阶数，N=1、2或4。

[0075] (2)根据线程的偏移量从显存中取出数据存入线程块内共享显存，显存和线程块内共享显存在物理上均是位于GPU上的存储设备，在逻辑上两者则不同，显存为从内存中拷贝数据，而线程块内共享显存则为开辟于线程块内的缓存，从显存获取数据，并在共享显存内进行线程的核函数计算。

[0076] (3)由于CUDA的技术特性，在流处理单元SM中的执行顺序决定了thread0会优先执行，因此针对满足tid.x=0的所有线程来执行其所在二维线程块Block输入数据的抽阶操

作。具体的抽阶操作根据栅格化数据处理来决定，从每一个字节取出特定的位，栅格化时数据的组织方式决定了该取哪一位或者哪几位(目前应用中高四位均为有效数据)，即根据栅格化处理规则从每一个字节取出特定的位，将抽阶的结果暂存到结果mask中。这样便可让所有线程分组执行既节约了运算资源，又避免了线程间数据冲突，保证了并行性。在此可以设缓存mask为4个字节，则处理完毕后，一个块的数据抽阶后只剩下了32位，被组织成一个int型(4字节)，所以只需要四个线程来拷贝结果数据。

[0077] (4)设int型为4字节，使用tid.x<4的线程Thread将结果数据从结果缓存mask中拷贝至对应的显存中，核函数执行完毕。

[0078] 第四步，待所有线程Thread的核函数计算完后，将显存中的结构数据拷贝回内存，此结构数据为抽阶后的栅格化数据，完成抽阶过程。

[0079] 本发明利用GPU高并行计算性能，以及CUDA框架的层次化并行特性，将直写式光刻机数据处理中的栅格化数据抽阶过程进行了并行优化，用大量的线程并行执行来加快执行速度，提高了直写式光刻机的产能。同时，在实现并行化的过程中，充分地利用了GPU的线程资源，实现了最大化的加速比，利用了共享显存的高带宽特性，提高了处理速度，读写全局显存时根据线程编号同步操作，有效的屏蔽了访存延时，进一步提高了处理效率。

[0080] 以上显示和描述了本发明的基本原理、主要特征和本发明的优点。本行业的技术人员应该了解，本发明不受上述实施例的限制，上述实施例和说明书中描述的只是本发明的原理，在不脱离本发明精神和范围的前提下本发明还会有各种变化和改进，这些变化和改进都落入要求保护的本发明的范围内。本发明要求的保护范围由所附的权利要求书及其等同物界定。

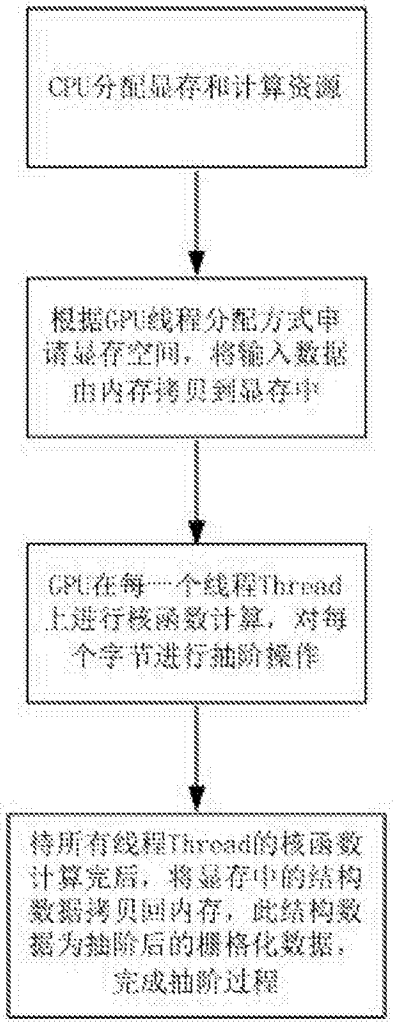

CPU分配显存和计算资源
根据GPU线程分配方式申请显存空间，将输入数据由内存拷贝到显存中
CPU在每一个线程Thread上进行核函数计算，对每个字节进行抽阶操作
待所有线程Thread的核函数计算完后，将显存中的结构数据拷贝回内存，此结构数据为抽阶后的栅格化数据，完成抽阶过程

图1

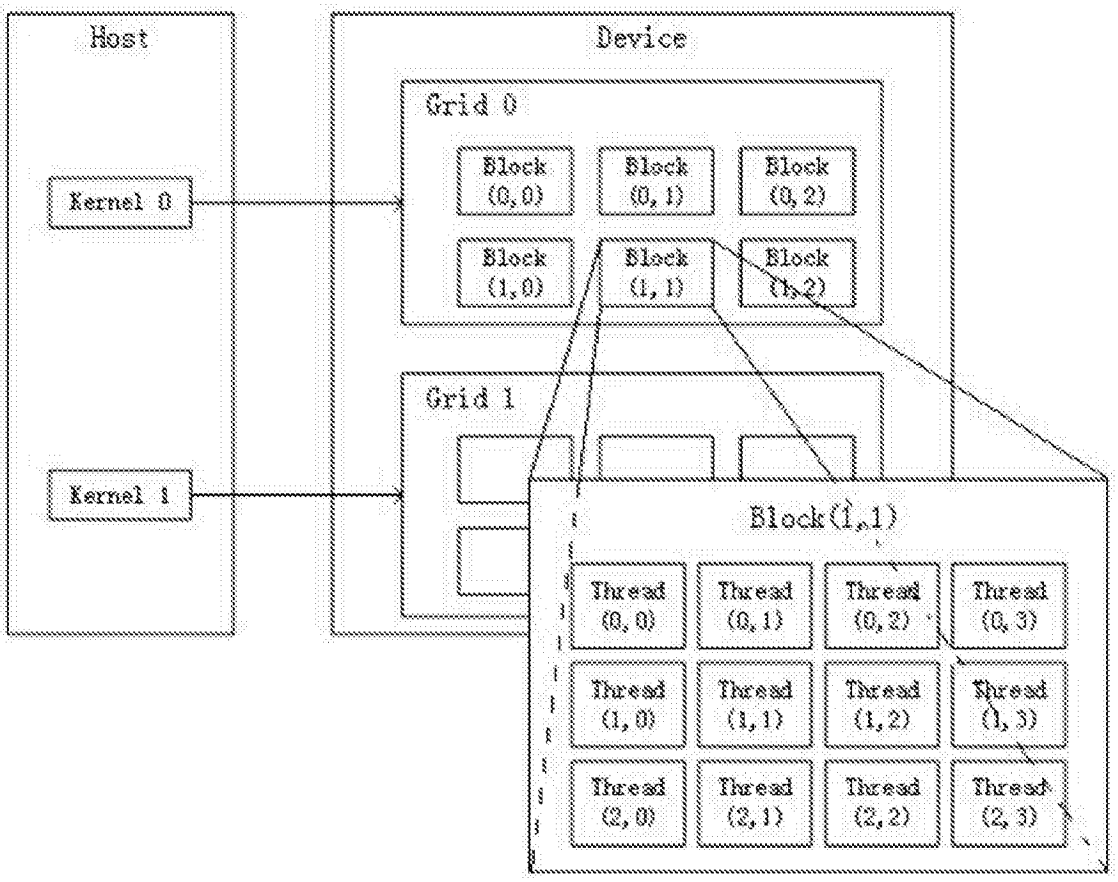

Host
Device
Grid 0
Kernel 0
Block (0,0)
Block (0,1)
Block (0,2)
Block (1,0)
Block (1,1)
Block (1,2)
Grid 1
Kernel 1
Block(1,1)
Thread (0,0)
Thread (0,1)
Thread (0,2)
Thread (0,3)
Thread (1,0)
Thread (1,1)
Thread (1,2)
Thread (1,3)
Thread (2,0)
Thread (2,1)
Thread (2,2)
Thread (2,3)

图2