(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 105677488 A
(43)申请公布日 2016.06.15

(21)申请号 201610018294.X

(22)申请日 2016.01.12

(71)申请人 中国人民解放军国防科学技术大学
地址 410073 湖南省长沙市开福区德雅路109号

(72)发明人 陈荦 吴秋云 刘世永 钟志农 熊伟 吴烨 陈浩 伍江江 李军 景宁

(74)专利代理机构 北京中济纬天专利代理有限公司 11429
代理人 胡伟华

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

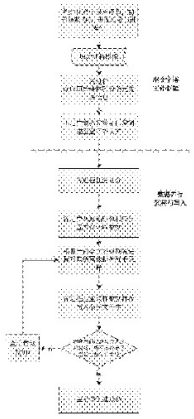
权利要求书1页 说明书7页 附图7页

(54)发明名称

一种混合并行方式栅格影像金字塔构建方法

(57)摘要

本发明属于地理空间信息处理技术领域，涉及一种混合并行方式栅格影像金字塔构建方法，具体包括步骤：(1)初始化金子塔总级数，进程总数与进程号，初始级数为0；(2)获取原始栅格影像，各进程读取原始栅格影像的元数据信息；(3)指定一个进程为主进程，创建空金字塔框架文件，并对创建的空金字塔框架文件采用多线程技术进行结构组织；(4)按照行划分的方式把栅格影像的数据量平均划分给各个进程，并计算在原始影像上的逻辑偏移位置；(5)根据逻辑偏移位置，采用GDAL类库函数将栅格影像的各波段数据依次读取到内存；(6)各个进程根据当前金字塔级数采用最邻近内插算法依次对数据进行对应粒度为2^{lev}的重采样；(S7)并行写出结果，构建金子塔文件。



CN 105677488 A

1.一种混合并行方式栅格影像金字塔构建方法，其特征在于，包括以下步骤：

(S1)初始化设定金字塔级数level、进程总个数n和金字塔级数迭代初始值lev为0级；消息传递接口(MPI)为每个进程分配一个进程号i，$0 \leq i<n$，作为该进程的唯一标识，其中i，level，n为整数；

(S2)获取原始栅格影像，各进程读取原始栅格影像的元数据信息；所述栅格影像的元数据信息包括原始栅格影像的长，宽，波段数，数据类型；

(S3)指定一个进程为主进程，创建空金字塔框架文件，并对创建的空金字塔框架文件采用多线程技术进行结构组织；具有过程为：主进程根据金字塔级数和栅格影像的元数据信息计算待创建空的金字塔框架文件的大小；若金字塔框架文件不超过4GB，则创建GeoTIFF格式的金字塔文件；如果金字塔文件大小超过4GB，则创建BigTIFF格式的金字塔文件；其中，GeoTIFF表示地理参考标签影像文件格式，BigTIFF表示大规模标签影像文件格式；主进程采用多线程的方式并行设置相应标记位把数据区和标签区的位置划分出来；将每层金字塔的元数据信息写入金字塔文件中标签区相应标签位，将数据区预留；

(S4)数据任务划分：按照行划分的方式把栅格影像的数据量平均划分给各个进程，每个进程根据进程号计算其所分配得到的数据块在原始影像上的逻辑偏移位置；

(S5)读取数据：各个进程根据步骤(S4)中的数据划分的逻辑偏移位置，采用GDAL类库的RasterIO函数将栅格影像的各波段数据依次读取到内存；

(S6)各个进程根据当前金字塔级数采用最邻近内插算法依次对步骤(S5)中读取的数据进行对应粒度为2^{lev}的重采样，lev为当前金字塔级数；将各波段重采样后的结果存入内存中；

(S7)并行写出结果，各个进程根据进程号以及波段号解算各波段在结果文件中的写入偏移位置；根据写入偏移位置采用MPI的MPI_File_write_at函数将内存中各波段重采样后的结果依次并行写入金字塔文件；

(S8)将当前金字塔级数加1，如果当前金字塔级数小于等于设定金字塔级数，则返回步骤(S6)；如果当前金字塔级数大于设定金字塔级数，则金字塔构建完毕。

2.如权利要求1所述的一种混合并行方式栅格影像金字塔构建方法，其特征在于，所述步骤(S4)中的行划分方式具体如下：以像素为单位，假设栅格影像的长为Xsize，宽为Ysize，波段个数为nBands，进程总数为n，则每个进程分配得到数据量DataSize等于(Ysize/n)×Xsize×nBands。

3.如权利要求1所述的一种混合并行方式栅格影像金字塔构建方法，其特征在于，所述步骤(S4)逻辑偏移位置具体计算方式如下：假设当前进程的进程号为i($0 \leq i<n$)，则该进程所读取数据的逻辑偏移位置DataOffset为i×(Ysize/n)×Xsize，其中，Xsize，Ysize分别表示栅格影像的长和宽。

4.如权利要求1所述的一种混合并行方式栅格影像金字塔构建方法，其特征在于，所述步骤(S7)中的具体解算方法如下：利用libtiff类库的TIFFGetField函数获取到该层金字塔数据区起始偏移位置ovrOffset，设当前进程的进程号为i，波段号为iband，$0 \leq iband<nBands$，iband为整数；以像素为单位当前层级金字塔大小为ovrXSize×ovrYSize，则各进程写入偏移位置为ovrOffset+ovrXSize×ovrYSize×iband+resBuffsize，resBuffsize为前i个进程重采样数据大小之和。

一种混合并行方式栅格影像金字塔构建方法

技术领域

[0001] 本发明属于地理空间信息处理技术领域，涉及一种混合并行方式栅格影像金字塔构建方法。

背景技术

[0002] 随着卫星传感器技术以及无人机航拍技术的快速发展，遥感影像的空间和时间分辨率都大幅度地提高，单幅遥感影像文件的数据量也急剧增加。对于大规模栅格影像进行快速显示和服务发布时，通过预先构建金字塔是提高可视化和服务性能的有效手段，但大数据量为金字塔的构建效率带来了巨大挑战。已有串行算法以及商业遥感软件构建金字塔的效率已远远跟不上数据的获取速度，高效快速创建金字塔已经成为栅格数据高效管理和可视化必须要解决的重要问题。

[0003] 金字塔是一种栅格数据的多分辨率组织结构。简单来说，金字塔结构就是由原始栅格影像开始，建立起一系列不同分辨率的栅格影像，不同分辨率的栅格影像对应不同的金字塔级。同时，金字塔也是栅格影像的一种有损压缩方式。构建金字塔以后，可以改善栅格影像显示性能，当用户需要对栅格影像进行不同分辨率地放大、缩小或平移时，通过选择一个与用户视图相近分辨率的数据进行可视化，从而系统只需进行少量的计算和查询就可以返回结果，不需要进行逐级采样计算，大大减少数据显示时间。

[0004] 并行构建金字塔主要有三种思路。一种是基于GPU(Graphic Processing Unit，图形处理单元)进行并行加速，利用GPU的计算能力加快金字塔的构建速度，这种方法与GPU硬件能力相关，会提高系统架构的成本；另一种是利用分布式集群系统，将大规模栅格影像文件的金字塔构建任务划分为多个子任务，在多个分布式节点上同时进行。这种方法扩展性较好，可以充分利用分布式并行环境来处理大规模数据，但需要将栅格影像分布存储在多个节点，数据的分布存储以及结果的合并都是比较耗时的。还一种是基于MPI(Message Passing Interface，消息传递接口)的多进程方式进行影像金字塔的并行构建，这种方法利用共享外存的高性能集群，实现对栅格影像并行重采样，然后将重采样结果并行写入到文件系统，这种方法现阶段存在一些问题，由于任务划分的不同容易产生黑边现象以及类型溢出。目前基于MPI以及OpenMP(Open Muti-Processing，共享存储并行编程)利用共享外存的高性能集群系统采用多进程与多线程混合并行方式的大规模影像金字塔构建方式研究较少。

发明内容

[0005] 针对上述技术问题，本发明提供一种混合并行方式栅格影像金字塔构建方法。本发明利用多进程技术对任务进行划分，当单个进程任务过重时，每个进程内部可采用多线程技术对任务进行细分进行二级并行，不仅解决了传统并行构建金子塔常见的黑边现象，而且实现了金字塔框架文件的高效创建，金字塔并行重采样，以及重采样数据并行写入。具体技术方案包括步骤如下：

[0006] (S1)初始化设定金字塔级数level、进程总个数n和金字塔级数迭代初始值lev为0级;消息传递接口(记为:MPI)为每个进程分配一个进程号i,$0 \le i < n$,作为该进程的唯一标识,其中i,level,n为整数;后续可以通过对该进程号描述相应进程的操作;

[0007] (S2)获取原始栅格影像,各进程读取原始栅格影像的元数据信息;所述栅格影像的元数据信息包括原始栅格影像的长,宽,波段数,数据类型;

[0008] (S3)指定一个进程为主进程,创建空金字塔框架文件,并对创建的空金字塔框架文件采用多线程技术进行结构组织;具有过程为:主进程根据金字塔级数和栅格影像的元数据信息计算待创建空的金字塔框架文件的大小;若金字塔框架文件不超过4GB,则创建GeoTIFF格式的金字塔文件;如果金字塔文件大小超过4GB,则创建BigTIFF格式的金字塔文件;其中,GeoTIFF表示地理参考标签影像文件格式,BigTIFF表示大规模标签影像文件格式;主进程采用多线程的方式并行设置相应标记位把数据区和标签区的位置划分出来;而后将每层金字塔的元数据信息写入金字塔文件中标签区相应标记位,将数据区预留;所述金字塔的元数据信息主要包括该层金字塔的长、宽、波段数、数据类型和该层金字塔各条带数据在数据区内的物理偏移位置。

[0009] (S4)数据任务划分:按照行划分的方式把栅格影像的数据量平均划分给各个进程,所述行划分方式具体如下:以像素为单位,栅格影像的长为Xsize,宽为Ysize,波段个数为nBands,进程总数为n,则每个进程分配得到数据量DataSize等于(Ysize/n)×Xsize×nBands;每个进程根据进程号计算其所分配得到的数据块在原始影像上的逻辑偏移位置;

[0010] 具体计算方式如下:假设当前进程的进程号为i($0 \le i < n$),则该进程所读取数据的逻辑偏移位置DataOffset为i×(Ysize/n)×Xsize;

[0011] (S5)读取数据:如果每个进程分配的数据量DataSize≤2147483647各个进程根据步骤(S4)中的数据划分的逻辑偏移位置,采用GDAL类库的RasterIO函数将栅格影像的各波段数据依次读取到内存;

[0012] 如果每个进程分配的数据量DataSize>2147483647,即超过整型上限出现类型溢出的情况,进一步将每个进程下再细分多个线程,细分后的多个线程再对该进程的数据块进行细分,具体细分过程如下:将DataSize不断除以2进行二分,直到满足以下条件$2147483647 \times 2^{k-1} \le DataSize \le 2147483647 \times 2^{k}$为止,k为二分次数,取整数;而后采用OpenMP技术在该进程下细分2^k个子线程,将进程的数据量DataSize平均分配给其2^k个子线程,则每个子线程读取的数据量为$DataSize/2^k$。当子线程对父进程的数据细分后,各线程分别计算其相对父进程内存空间起始位置的偏移值,具体计算方法如下:假设当前线程的线程号为it($0 \le it < 2^k$),it取整数,则其偏移值为$it \times (DataSize/2^k)$。各线程根据相对父进程的偏移值以及父进程的逻辑偏移位置,计算得到各线程从栅格影像上读取数据的逻辑偏移位置tDataOffset等于$DataOffset+it \times (DataSize/2^k)$。由于各子线程是共享父进程的内存空间,因此各线程根据其偏移值tDataOffset利用GDAL类库的RasterIO函数将$DataSize/2^k$个数据并行独立读取到父进程的内存空间中。

[0013] (S6)各个进程根据当前金字塔级数采用最邻近内插算法依次对步骤(S5)中读取的数据进行对应粒度为2^{lev}的重采样,lev为当前金字塔级数;将各波段重采样后的结果存入内存中;

[0014] (S7)并行写出结果,各个进程根据进程号以及波段号解算各波段在结果文件中的

写入偏移位置;根据写入偏移位置采用MPI的MPI_File_write_at函数将内存中各波段重采样后的结果依次并行写入金字塔文件;

[0015] 具体方法如下:首先利用libtiff类库的TIFFGetField函数获取到该层金字塔数据区起始偏移位置ovrOffset,设当前进程的进程号为i,波段号为iband,0≤iband<nBands,iband为整数;以像素为单位当前层级金字塔大小为ovrXSize×ovrYSize,则各进程写入偏移位置为ovrOffset+ovrXSize×ovrYSize×iband+resBuffsize,resBuffsize为前i个进程重采样数据大小之和。

[0016] (S8)将当前金字塔级数加1,如果当前金字塔级数小于等于设定金字塔级数,则返回步骤(S6);如果当前金字塔级数大于设定金字塔级数,则金字塔构建完毕。

[0017] 采用本发明获得的有益效果:本发明的有益效果是:(1)本发明所生成的栅格影像金字塔文件与地理空间数据抽象库GDAL(Geospatial Data Abstract Library,地理数据抽象库)生成的金字塔文件格式一致,GDAL是一种被广泛应用的金字塔构建工具,其创建的金字塔文件能够被当前绝大多数地理信息系统软件直接使用,可以无缝集成到各种地理信息应用中。(2)本发明构建金字塔效率高。创建空金字塔文件时,选择主进程为0号进程,采用多线程技术加快了创建速度,而且创建时预先规划好金子塔标签区和数据区的位置,使得标签区和数据区内部可以连续紧凑排列,而标签区和数据区之间则相互独立互不影响,为后面重采样数据能够快速并行写入创造了条件。(3)本发明算法并行程度高。各个进程在并行从原始影像读取相应数据,并行对其所属数据重采样,并行将采样数据写入金字塔文件。(4)本发明算法稳定性高,支持各种数据类型的栅格数据,当单个进程任务过重时采用多线程对任务再进行细分,不仅加快了速度,而且避免类型溢出的产生。

附图说明

[0018] 图1是本发明的整体流程示意图;

[0019] 图2是本发明创建空金字塔文件的流程示意图;

[0020] 图3是本发明的各级重采样数据写入金字塔文件的流程示意图;

[0021] 图4是本发明金字塔文件数据的组织结构示意图;

[0022] 图5是本发明各进程任务划分的示意图;

[0023] 图6是本发明实施例各进程并行重采样的示意图;

[0024] 图7是本发明多波段数据组织的示意图;

[0025] 图8中(a)、(b)图是GDAL算法、ArcGISR软件、赫高进等人发明与本发明方法对不同类型栅格影像构建金字塔的性能对比;

[0026] 图9是本发明方法针对各种规模影像构建金字塔时执行时间随处理进程数目变化情况。

具体实施方式

[0027] 结合附图和具体实施例对本发明作进一步说明。

[0028] 图1为本发明的整体流程示意图。本发明方法分为两个部分,第一是生成空金字塔文件,第二是各进程并行从原始栅格影像文件中读取相应数据进行重采样,最后把各级重采样数据高效并行地写入空金字塔文件中。算法原理清晰,图2和图3针对流程图中的空金

字塔文件快速创建与数据并行重采样与写入这两方面再做进一步详细说明。

[0029] 图2为本发明创建空金字塔文件的流程示意图。将0号进程选为主进程，假设以像素为单位，原始栅格影像的长为Xsize，宽为Ysize，波段总数为nBands，每个像素所占字节数为nPixelSize，需要创建金字塔的级数为level，则该金字塔文件数据区的大小ODsize为$\sum_{l=1}^{level}\left(\frac{Xsize}{2^l}\times\frac{Ysize}{2^l}\times nPixelSize\times nBands\right)$，如果ODsize大于4GB，那么将利用libtiff库的VSI_TIFFOpen函数创建格式为BigTIFF的空的金字塔框架文件，否则创建格式为GeoTIFF的金字塔框架文件，此时的空金字塔框架文件中除了必要的头文件信息外，没有其他任何元数据信息以及影像数据。空金字塔框架文件创建完成后，主进程采用多线程的方式并行设置相应标记位把数据区和标签区的位置划分出来。所述金字塔的元数据信息主要包括该层金字塔的长、宽、波段数、数据类型和该层金字塔各条带数据在数据区内的物理偏移位置，这些信息通过libtiff库中的TIFFSetField和TIFFWriteDirectory等函数写入金字塔文件标签区内相应的标记位上，其中波段数、数据类型等信息与原始栅格影像的元数据信息一致，每层金字塔长$ovrXSize=\frac{Xsize}{2^{lev+1}}$、宽$ovrYSize=\frac{Ysize}{2^{lev+1}}$，各条带数据偏移位置计算方法参照图3说明中的StripOffsets计算过程。而后利用libtiff库的TIFFSetField等函数将每层金字塔的元数据信息例如长宽、波段数、存储方式等写入金字塔文件中标签区，预留出来的数据区用于后期各进程重采样数据并行写入。其余进程不进行任何操作一直阻塞直到主进程完成所有相应操作；具体的金字塔文件内部组织结构参考图4。

[0030] 图3为本发明的各级重采样数据写入金字塔文件的流程示意图。在空金字塔文件创建完成后，根据进程总数，采用行划分的方式对原始栅格影像做平均划分，建立进程号与数据逻辑偏移位置的映射关系。每个进程根据逻辑偏移位置读取相应数据，如果总进程数过小导致每个进程平均分配得到的数据量过大产生了类型溢出，那么利用OpenMP采用多线程技术，使每个进程下再细分多个线程，多个线程再对该进程的任务进行上述方法的细分，线程间并行将数据读入父进程的缓冲区中。各进程将原始数据读入其缓冲区后，根据当前金字塔的级别，采用最邻近的方法进行重采样，而后建立进程号与写入偏移位置的关系，各进程将其重采样结果并行写入该层金字塔文件中，直到所有层级金字塔都成功写入为止。

[0031] 图4为本发明金字塔文件数据的组织结构示意图，金字塔文件一共分为三个部分，分别为IFH(imge file head文件头)，DATA(data数据区)，IFD(image file directory图像文件目录)。IFH共占8个字节，记录了文件的标记、版本号以及第一个IFD的偏移量。DATA是金字塔中存放重采样数据的区域。IFD是金字塔文件的标签区，其记录了以下三个部分：该层金字塔的标签数(简称DEC)、每个标签的数据结构(简称DE)、下一个IFD的偏移量(简称NIFD)。金字塔重采样数据以条带形式组织，每个条带的偏移位置通过IFD中的StripOffsets标签进行定义。以第一层金字塔为例，假设该层金字塔数据的长为OvrXsize，宽为OvrYsize，则该层重采样数据大小OvrSize为OvrXsize×OvrYsize×nBands，写入的偏移位置范围为8～(8+OvrSize)。IFD写入的偏移范围为(8+OvrSize)～(8+OvrSize+(6+DEC×12))，StripOffsets标签中的第m个条带的值StripOffset(m)为8+m×stripSize，其中stripSize为条带大小。其余层金字塔组织方式与第一层类似，只不过其重采样数据存放的

初始偏移位置紧贴上一层金字塔IFD的后面，设该层金字塔初始偏移位置为DFOffset，则该层金字塔第m个条带的偏移值StripOffset(m)等于DFOffset+m×stripSize。金字塔文件经过上述组织后，之后步骤中各进程将重采样并行写入时，可以快速准确地获取金子塔数据区的位置，并将写入位置与进程号做一一映射。

[0032] 图5是本发明各进程任务划分和重采样的示意图。本发明采用分波段按行划分的方式进行数据分配，各个进程按照划分好的数据大小，各自从原始栅格影像相应逻辑偏移位置处地读取条带数据。假设原始栅格影像长为ImgXsize，宽为ImgYsize，参与运算的进程总数为n，rank(i)表示第i个进程，i取值为大于等于0至小于等于n-1之间的整数，则当i≠n-1时各个进程rank(i)读取数据大小srcBuffsize(i)为ImgXsize×floor(ImgYsize/n)，当i=n-1时srcBuffsize(i)为$\mathrm{ImgXsize}\times\left(\mathrm{ImgYsize}-\left[(n-1)\times\mathrm{floor}\left(\frac{\mathrm{ImgYsize}}{n}\right)\right]\right)$，其在波段内的起始读取位置offset(i)为i×floor(ImgYsize/n)，floor表示向下取整。基于上述任务划分方法，如果影像过大，参与的进程数不足导致单个进程任务过重造成类型溢出，那么采用OpenMP对该进程进行多线程化，对该进程数据量进行再次细分，整型类型发生溢出的上限为TypeMAX，即整型最大值2147483647；则各进程申请的线程总数t满足以下关系$\frac{\mathrm{srcBuffsize}\ \ (i)}{2^t}\leq\mathrm{TypeMAX}\leq\frac{\mathrm{srcBuffsize}\ \ (i)}{2^{t-1}}$。

[0033] 图6是本发明各进程并行重采样的示意图，图中以进程数n=3为例对一个7×11的影像数据进行第一层金字塔的重采样操作，黑色块为采样结果。进一步推导出各lev层级相对原始数据的采样步骤，这种采样方法为最邻近法，即长和宽方面各每隔2^{lev}取一个像素，lev为当前金字塔层级。由于边界的问题导致相邻进程即使读取的数据量一样，但最终重采样的大小不一定相同，如图所示Rank(0)重采样大小为2，rank(1)重采样大小为1，rank(2)重采样大小为3；具体解释为：任务划分采用行划分，影像数据被平均划分给各个进程，但是由于取整的问题，所以除了最后一个进程外其余进程的数据量都是一样的，但是数据量一样，其重采样后大小不一定一样，这里主要是反驳以往常见思维误区。以图6为例，这个影像的大小是7×11，被三个进程划分，$\frac{11}{3}$取整等于3所以进程0和进程1分到三行数据，进程2分到(11-2×3=5)五行数据，当为其创建第一级金字塔时，按照最邻近采样法即影像的长宽方向各自隔两个像素点取一个像素，图中黑色部分框格就是相应采样结果，从图中可以看到，各个进程所属数据包含黑色框格行数进程0为2、进程1为1、进程2为3，各黑色窗格在各进程内行偏移起始位置RowOffset满足进程0为0、进程1为1、进程2为0。因此各进程对其数据重采样时不能简单的从数据位置处进行采样，应该从行偏移RowOffset处进行重采样才是正确的。

[0034] 这种问题主要原因是各个进程采样起始位置RowOffset(i)不一定相同，其计算方式如下：当offset(i)%2^{lev}=0时RowOffset(i)=0，当offset(i)%2^{lev}≠0时RowOffset(i)=2^{lev}-offset(i)%2^{lev}，offset(i)表示第i个进程在波段内的起始读取位置；各进程重采样结果数据大小resBuffsize(i)等于$\left(\mathrm{ImgXsize}/2^{lev}\right)\times\left[\left(\frac{\mathrm{ImgYsize}}{n}-\mathrm{RowOffset(i)}\right)/2^{lev}\right]$，%表示取余。基于这种重采样方法，各进程只需从磁盘中读取一次原始数据到内存，进行不

同层级重采样时，只需设置不同的采样间隔，从原始数据中抽稀采样即可，有效地减少了磁盘IO的耗时。

[0035] 图7是本发明多波段数据组织的示意图。相对赫高进等人采用基于BIP(波段按行交叉)方式来组织多波段数据(参考文献：赫高进，陈荦，熊伟等.基于MPI的大规模遥感影像金字塔并行构建方法[J].地球信息科学学报，2015，17(5)：515-522.)，本发明提出一种基于BSQ(波段顺序格式)的多波段重采样数据组织方式，这种方式克服了BIP方式读写不连续，创建文件视图复杂的缺点。其组织方式是每行数据后面紧接着存储同一波段的下一行数据，存完一个波段的所有数据后再紧接着存储下一个波段的数据。采用这种数据组织方式，各进程并行写入时不需要创建文件视图，也不需要对单波段和多波段分别考虑，只需计算出其在金字塔文件中的绝对偏移位置，然后根据偏移位置把重采样结果连续写入金字塔文件中即可，进程Rank(i)中所属第q波段数据绝对偏移位置AbsOff等于$\mathbf{ovrOffset}+\mathbf{q}\times\mathbf{OvrXsize}\times\mathbf{OvrYsize}+\sum_{j=0}^{i-1}\mathbf{resBuffsize(j)}$。在使用MPI的并行IO函数MPI_File_write_at将重采样数据写入时，只需将每个进程绝对偏移位置AbsOff，以及其重采样数据缓冲区大小resBuffsize作为实参代入函数中，即完成重采样数据的并行写入。

[0036] 图8中(a)、(b)图是GDAL算法、ArcGISR软件、赫高进等人发明与本发明方法对不同类型栅格影像构建金字塔的性能对比；在相同硬件条件下，所采用栅格影像数据为1.tif、2.tif、3.tif、4.tif、5.tif、6.tif，其中数据1.tif：长72001像素，宽48001像素，数据类型为32位浮点型，1波段，大小12.9GB；数据2.tif：长87040像素，宽58368像素，数据类型为8位字节型，3波段，大小14.2GB；数据3.tif：长220672像素，宽86272像素，数据类型为8位字节型，3波段，大小53.2GB；数据4.tif：长432001像素，宽144001像素，数据类型为16位整型，1波段，大小115.9GB；数据5.tif：长136448像素，宽90368像素，数据类型为8位字节型，3波段，大小137.8GB；数据6.tif：长428142像素，宽232572像素，数据类型为8位字节型，3波段，大小278.2GB；并行构建金字塔算法进程总数为32，图中纵坐标为算法耗时(单位为秒)，横坐标为测试影像名称，表格内数字表示相应算法对应不同数据的耗时(单位为秒)。可以看到本发明在面对各种数据类型的栅格影像都保持稳定高效的效率，当数据量达到200GB以上时GDAL和ArcGIS在运行十多个小时后会产生错误异常，赫高进等人发明当数据量达到100GB以上时在运行十分钟左右后会产生错误异常，都无法再继续执行下去。随着数据量的增大本发明的优势愈加明显，特别是在面对多波段数据时本发明较赫高进等人发明具有很大提升。

[0037] 图9是本发明方法构建金字塔时执行时间随处理进程数目变化情况。横坐标是进程数，纵坐标是算法执行时间(单位为秒)，表格内数字表示相应算法对应不同数据的耗时(单位为秒)。从图中可以看到：(1)一定范围内随着进程数的增加，本发明的效率逐步提升，但是随着进程数的持续增加，算法耗时逐步趋于某一稳定值，如果进程数继续加大，算法速度在某一程度反而会出现下降；(2)随着数据量的增大，算法的并行化效率愈加明显。其原因如下：随着进程数的增加，任务被划分给更多进程执行，每个进程对应任务规模相应更小，所以算法整体性能得到提升。当进程数目增大到一定程度后，受限于硬盘的读写速度，算法性能趋于稳定，但是如果继续增大那么可能出现各进程读写竞争的情况，导致性能下降。所以针对不同规模的影像，应采用相应合理的进程数才能获得最优的执行效率，通过测

试表明该发明针对不同规模的影像具有良好的扩展性。

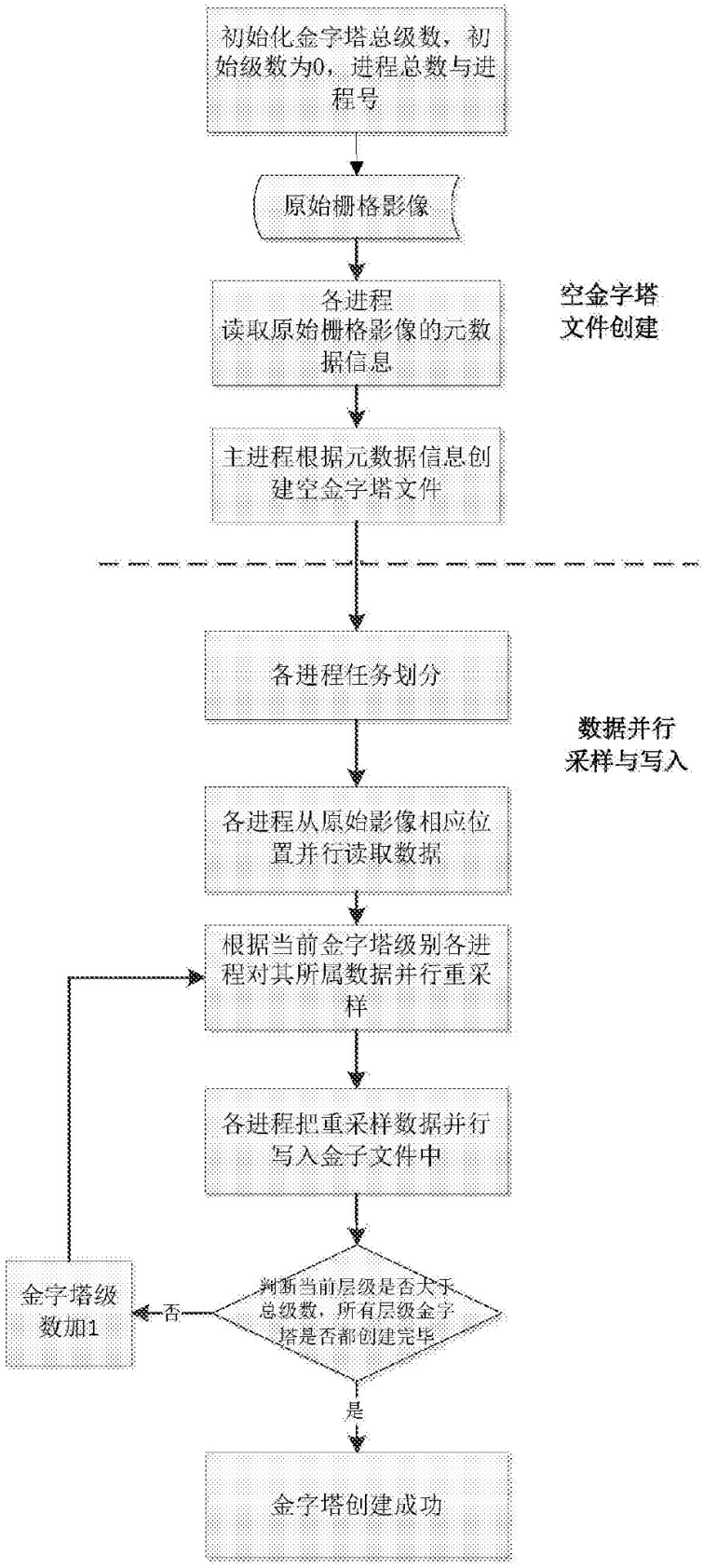

初始化金字塔总级数，初始级数为0，进程总数与进程号
原始栅格影像
各进程读取原始栅格影像的元数据信息
空金字塔文件创建
主进程根据元数据信息创建空金字塔文件
各进程任务划分
数据并行采样与写入
各进程从原始影像相应位置并行读取数据
根据当前金字塔级别各进程对其所属数据并行重采样
各进程把重采样数据并行写入金子文件中
判断当前层级是否大于总级数，所有层级金字塔是否都创建完毕
否
金字塔级数加1
是
金字塔创建成功

图1

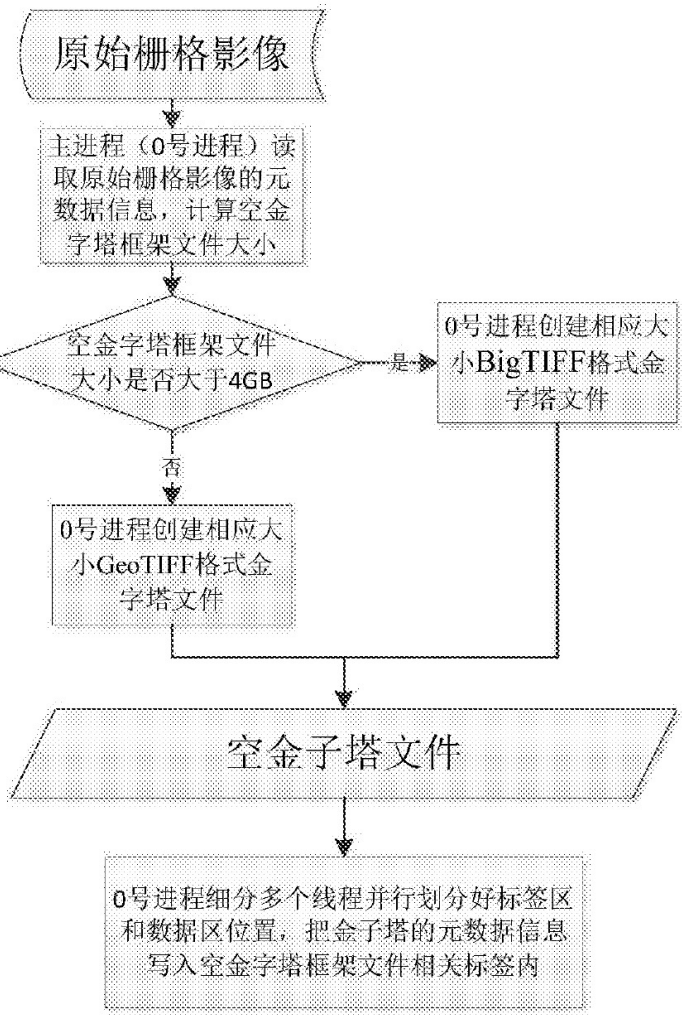

原始栅格影像
主进程（0号进程）读取原始栅格影像的元数据信息，计算空金字塔框架文件大小
空金字塔框架文件大小是否大于4GB
是
否
0号进程创建相应大小BigTIFF格式金字塔文件
0号进程创建相应大小GeoTIFF格式金字塔文件
空金子塔文件
0号进程细分多个线程并行划分好标签区和数据区位置，把金子塔的元数据信息写入空金字塔框架文件相关标签内

图2

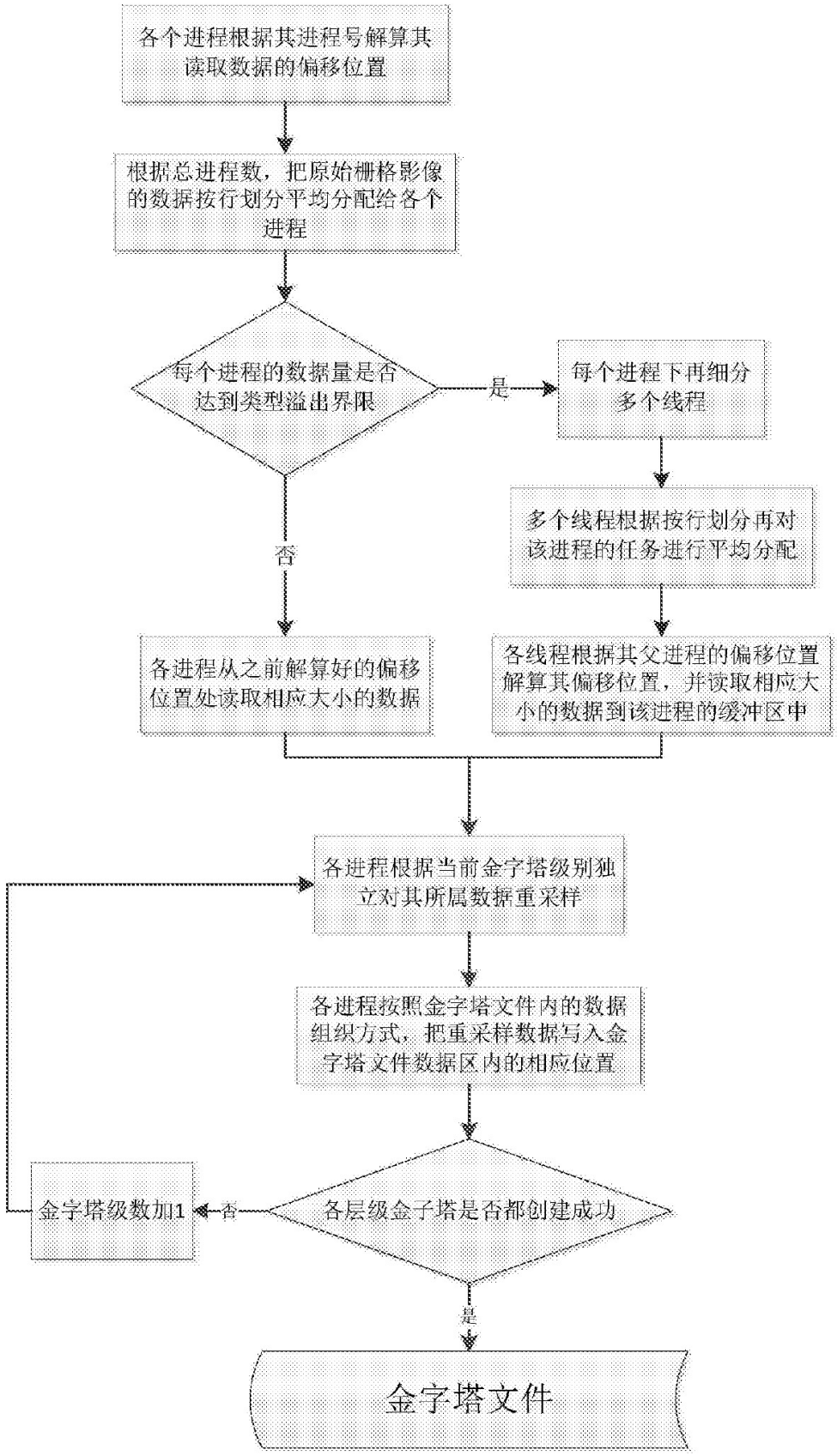

各个进程根据其进程号解算其读取数据的偏移位置
根据总进程数，把原始栅格影像的数据按行划分平均分配给各个进程
每个进程的数据量是否达到类型溢出界限
是
每个进程下再细分多个线程
多个线程根据按行划分再对该进程的任务进行平均分配
否
各进程从之前解算好的偏移位置处读取相应大小的数据
各线程根据其父进程的偏移位置解算其偏移位置，并读取相应大小的数据到该进程的缓冲区中
各进程根据当前金字塔级别独立对其所属数据重采样
各进程按照金字塔文件内的数据组织方式，把重采样数据写入金字塔文件数据区内的相应位置
各层级金子塔是否都创建成功
否
金字塔级数加1
是
金字塔文件

图3

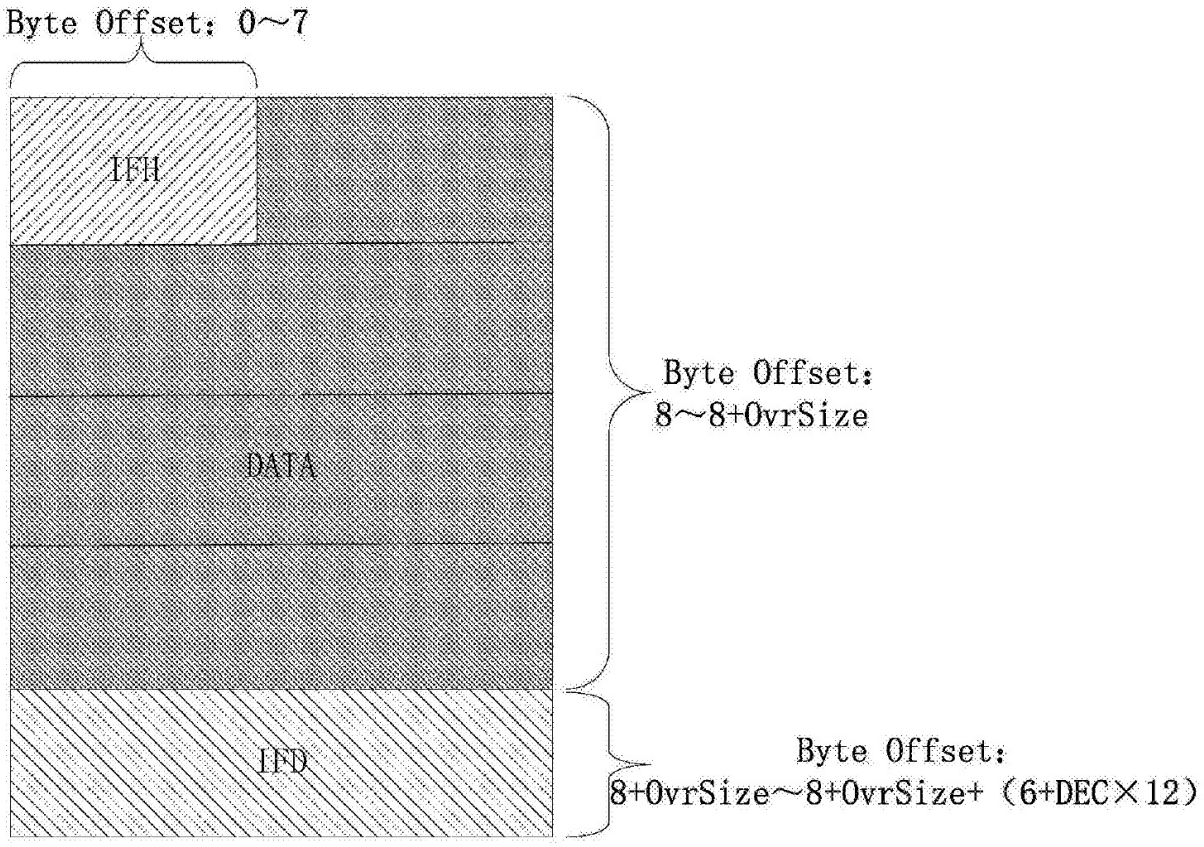

Byte Offset: 0～7
IFH
DATA
IFD
Byte Offset:
8～8+OvrSize
Byte Offset:
8+OvrSize～8+OvrSize+ (6+DEC×12)

图4

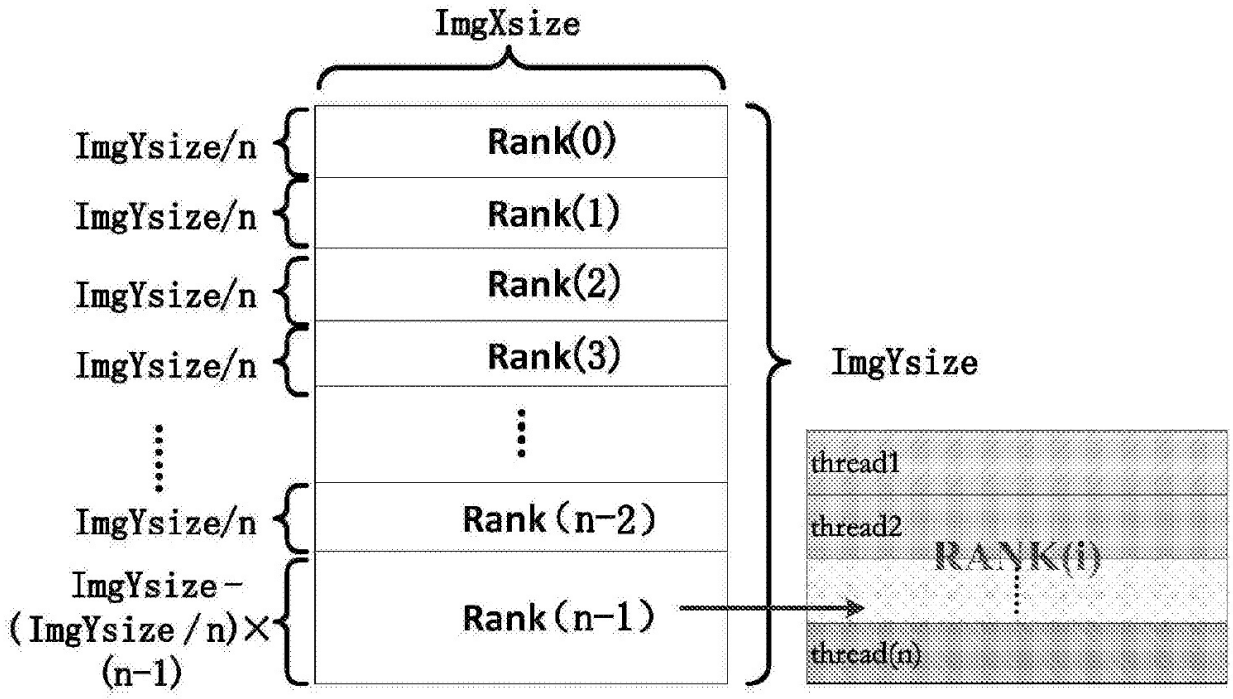

ImgXsize
ImgYsize/n
ImgYsize/n
ImgYsize/n
ImgYsize/n
ImgYsize/n
ImgYsize-
(ImgYsize/n)×
(n-1)
Rank(0)
Rank(1)
Rank(2)
Rank(3)
Rank(n-2)
Rank(n-1)
ImgYsize
thread1
thread2
RANK(i)
thread(n)

图5

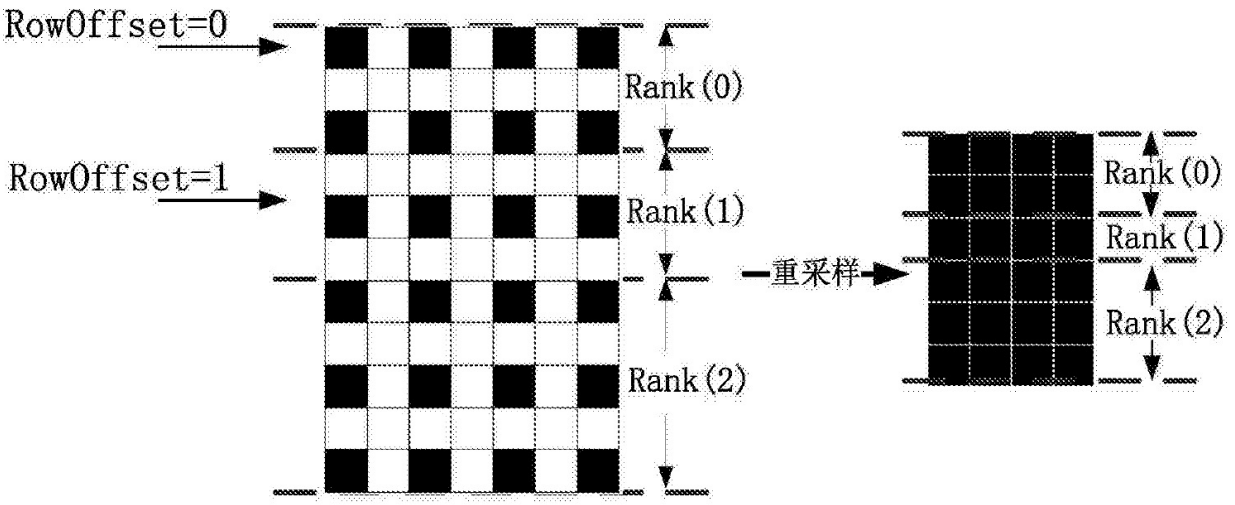

RowOffset=0
RowOffset=1
Rank (0)
Rank (1)
Rank (2)
重采样
Rank (0)
Rank (1)
Rank (2)

图6

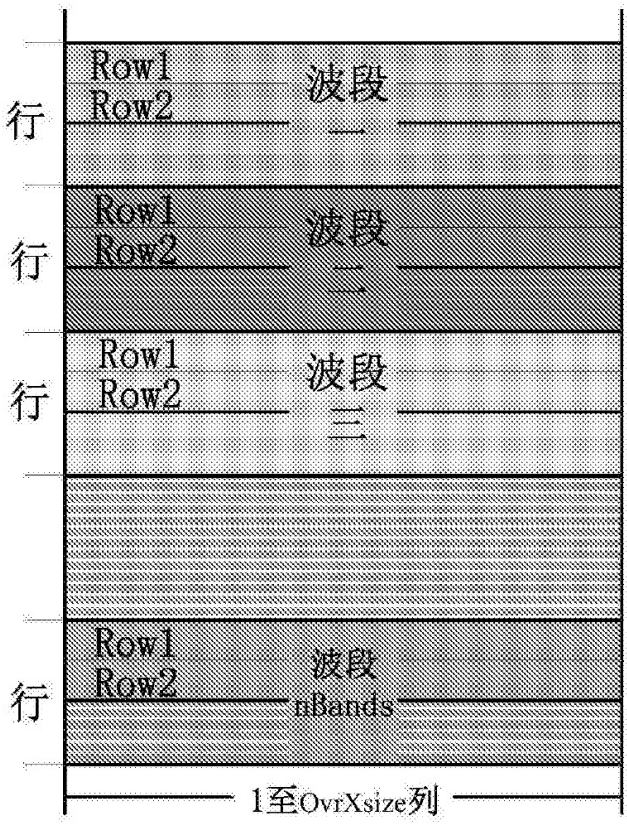
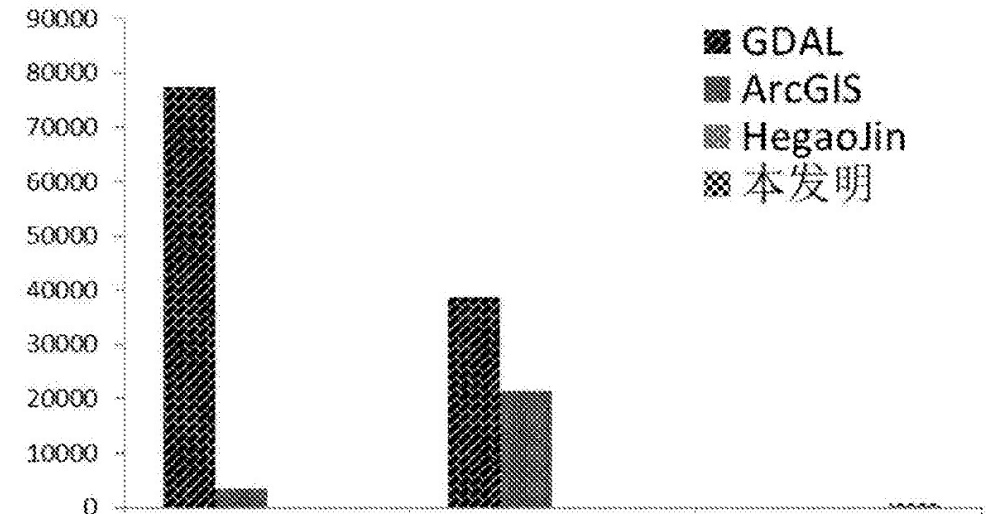

行
Row1
Row2
波段
一
行
Row1
Row2
波段
二
行
Row1
Row2
波段
三
行
Row1
Row2
波段
nBands
1至OvrXsize列

图7

	1.tif	2.tif	3.tif
GDAL	218.42	191.34	1909
ArcGIS	218.4	191.336	1909
HegaoJin	41.208	202.97	957.83
本发明	27.526	38.55	200.57

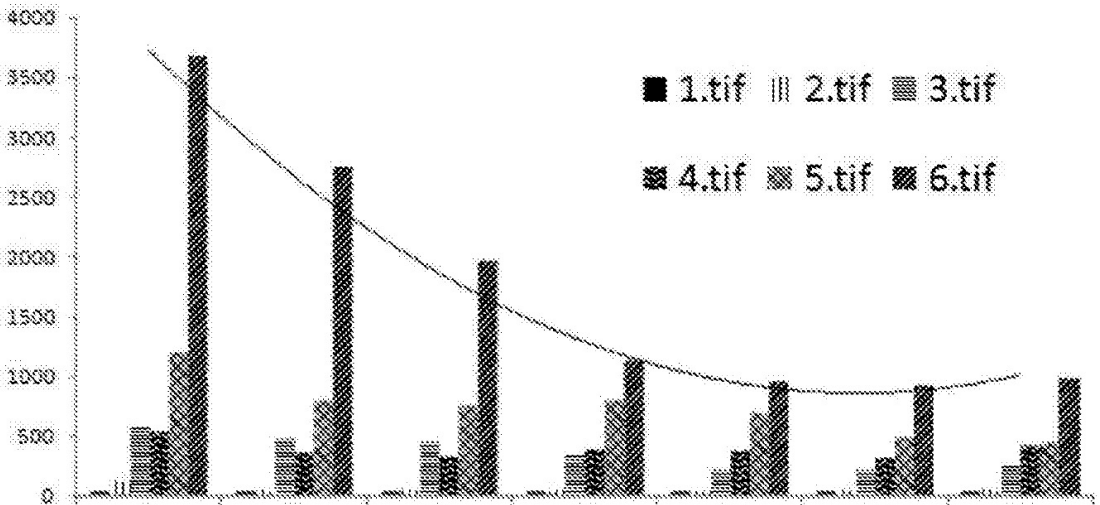
(a)



	4.tif	5.tif	6.tif
GDAL	77350.79	38711.2	
ArcGIS	3485	21600	
HegaoJin	420.13		
本发明	369.46	694.46	958.5

(b)

图8



	2	3	4	8	16	32	64
1.tif	34.836	33.89	30.48	28.95	27.52	27.24	27.35
2.tif	118.194	48.974	43.562	41.71	38.55	40.354	47.342
3.tif	572.53	472.705	441.665	344.685	200.57	206.78	259.76
4.tif	530.29	357.05	328.08	387.24	369.46	314.6	418.42
5.tif	1205.46	796.15	748.85	786.74	694.24	488.84	446.66
6.tif	3681.75	2750.68	1959.08	1127.21	958.5	930.25	989.14

图9