PC集群环境下的并行地图切片负载均衡生成算法

**摘要**

webgis中随着切片层级的增加，切片的数量呈指数增长，针对单机切片时效低的问题，提出了PC集群环境下多级比例尺大规模切片数据生成的负载均衡并行算法，利用分布式数据存储和节点的本地计算资源，算法通过对切片的任务量的预估，实现了面向任务量的负载均衡；同时建立动态反馈机制根据节点的实时计算能力的量化反馈，完成切片剩余任务的再分配，实现了切片过程中的实时动态负载均衡，有效提高了集群中各节点的利用效率。

1. 并行地图切片生成算法

地图切片是将预先配置好的一定范围内的地图，按一定的比例尺级别（瓦片级别），切片成若干个以行列进行编排的图片数据，并按一定的命名规则和组织方式存储在目录系统或者数据库系统中，形成金字塔结构模型的静态地图缓存。在并行的地图切片算法中，以单幅配置地图为数据源，以矢量数据量均分到各集群节点为原则对切片进行分块，并以整块图片字节流的形式紧凑存储在各节点上，最后进行归并。

* 1. 基本思路

本文提出一种可扩展的地图切片服务PC集群负载均衡模型（图一），模型通过多次分配实现了如何将地图切片根据矢量数据内容量公平分配到各节点上，使用arcgis10.0推出的紧凑型瓦片存储格式进行128\*128个瓦片的字节流整体存储。在模型中，存在主控节点，即负载均衡器进行切片参数的初始化，通过对矢量数据最小外包络范围的矢量数据量统计，得到总的切片任务量，建立全局的分比例尺级别的矢量内容表，根据节点数目进行矢量数据的均分，并反算出每个节点获取的实际地理范围，实现首次任务分配。切片过程中，各节点主动向对向主节点发送节点负载情况，主节点将根据第一次切片的节点资源计算情况，对剩下的任务进行分配。

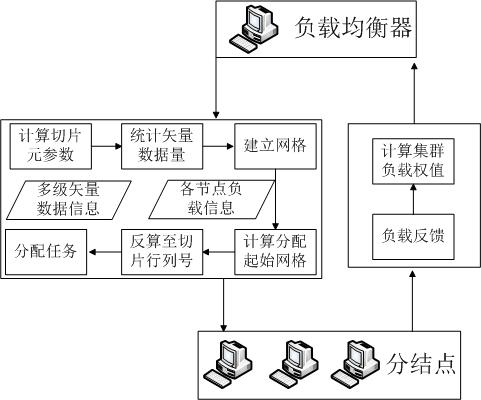


图1.面向矢量内容的切片负载均衡模型

Fig1.Load balancing model for generating map tile faced vector.

* 1. 负载均衡算法

1. 计算切片方案最优参数

由原始配置好的地图转换为切片时，对切片的原点、范围进行最优设置，否则容易出现多出一行一列空切片的情况。，，切片原点为(orgX,orgY)，

定义1 矢量数据范围。设待切片的以地图要素的最小外包络矩形范围为[Xmin,Ymin,Xmax,Ymax]，地图的宽为Width= Xmax-Xmin，高为Height= Ymax-Ymin

定义2 切片图片大小。切片图片像素宽高为imgW、imgH，每像素代表的地理长度为reslution。

TileWidth=imgW \* reslution

TileHeight=imgH \* reslution

定义3 切片的总行列数及总个数。

Rows =[Width / imgW / reslution]

Cols =[ Height / imgH / reslution]

Num=Rows\*Cols

其中[]为向上取整操作。

定义4 切片图层的高宽。由于行列号进行了向上的取整操作，计算的切片图层比地图图层地理范围稍大且刚好覆盖所有的切片。

TileWidthCount=Cols \* TileWidth=Cols \* imgW \* reslution

TileHeightCount=Rows \* TileHeight=Rows \*imgH \* reslution

定义5计算最优切片原点。以整个切片图层覆盖范围区域中心，与地图区域中心重合为原则计算切片原点(orgX, orgY)。按照该原点及切片图层所计算得出的切片恰好覆盖所有地图图层，且避免了空白行列，最终形成如图2的切片方案。

orgX = Xmin - (TileWidth - Width) / 2

orgY = Ymax + (TileHeight - Height) / 2

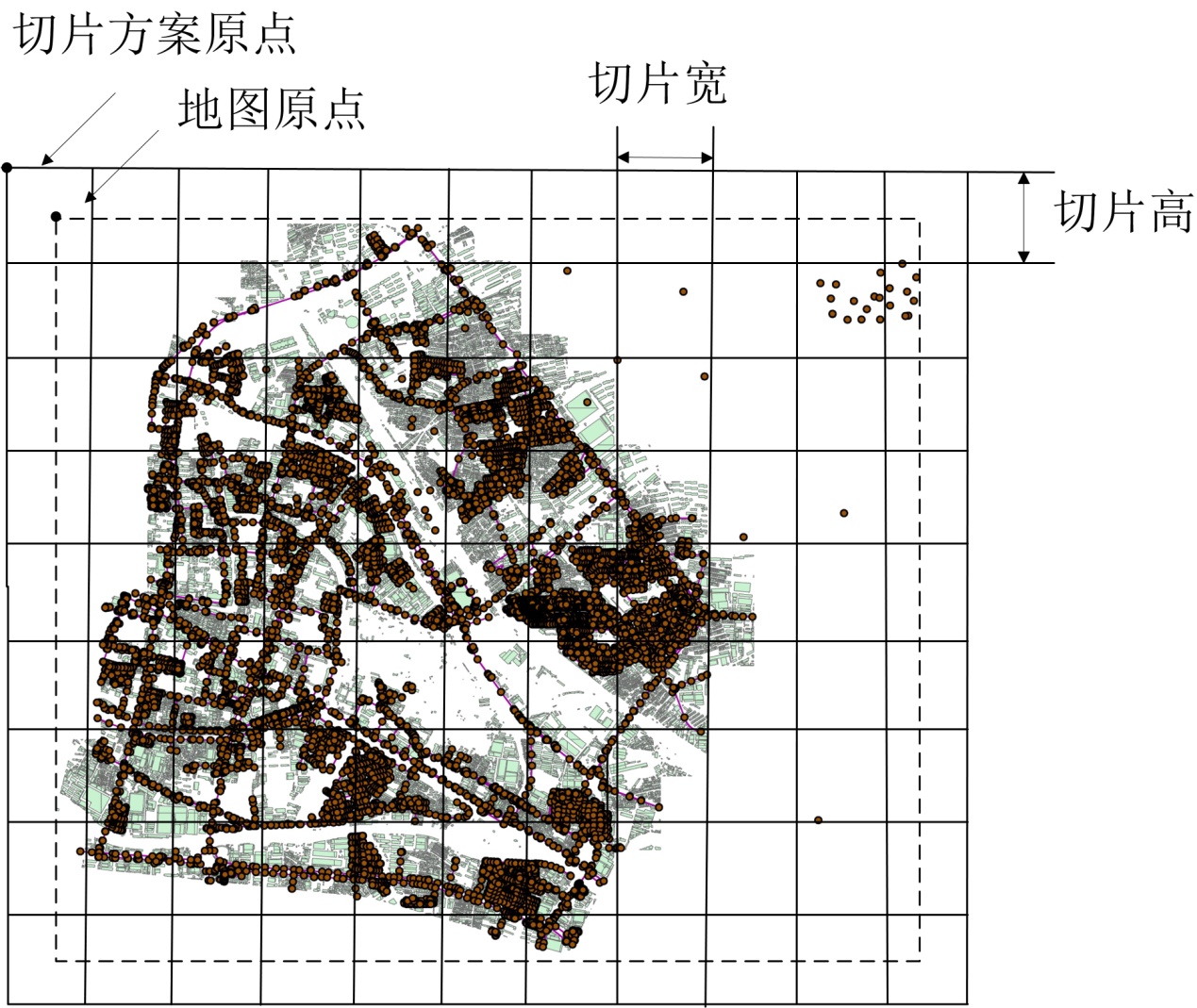


图2. 最优切片方案

Fig.2 The optimum tile scheme

1. 生成矢量统计网格。

在地图渲染中，要素种类、要素量的多少都是影响渲染速度的关键因素，因此在切片任务的划分中，不能简单将切片数量平均分配到各个节点上进行出图，这明显是一种不均衡的任务分配。我们以区域要素的个数来表达区域内部的任务负载，分别统计各区域内要素的个数及其在整个任务量中所占比例，预估子区域的不同负载。

将切片图层划分为M行，N列，其中Rows和Cols为M和N的整数K倍，K数值可根据实际情况进行调整。

定义6 设N(m,n)为第m行第n列的数据块。为了查询每个网格的矢量数据量，必须对矢量数据范围进行计算，设[Txmin,Tmax,Tymin,Tymax]为该数据块的范围极值。

Txmin = orgX + n\*k\*tileWidth

Tymin = orgY - (m+ 1) \*k\* tileHeigh

Txmax = orgX + (n + 1) \* k\*tileWidth

Tymax = orgY - m \*k\* tileHeigh

1. 统计矢量数据量

以M行N列的尺度对切片进行分块后，区域被分解成M\*N个子区域，以子区域大小进行矢量要素的查询。得到每块区域的不同负载N(i,j)。

定义7 整幅地图范围的矢量数据总量Ntotal。对每个子区域进行统计得到整幅地图中的矢量数据总量。

Ntotal=

1. 对各节点的任务量进行分配。

本文采取的面向切片任务量的动态平衡和静态负载相结合的方式。主要考虑2方面的内容：任务的计算复杂度和节点的处理及负载能力。在地图切片中，同样地理范围大小的不同区域，由于要素空间不均匀分布，要素图层数目及个数会影响地图绘制和渲染的时间，需要对任务计算进行预估；即使是在同构的集群环境中，不同节点对相同任务量的处理能力也会有所差异，不可能出现完全的负载均衡，因此我们将多级的切片任务量逐级多次分发，以实现最大可能的负载均衡。设共有w个节点，在首次分发中，此时各个节点的处理能力相同，他们的任务负载权值相同，需要将切片数据范围内的矢量数据分布信息网格分为w份，此时第r个节点应分得的矢量数据任务量为w(r)=Ntotal/w。

从切片原点开始对逐行逐列统计(M,N)的矢量数据量，设目前正在扫描的是第i行第j列子区域，正在为第r个节点进行任务分配。此时再地图全幅范围内已对i\*j个子区域的矢量数据进行统计求和，统计所有已扫描过的所有网格矢量数据量为N=，比较已统计的矢量数据量并与w(r)大小作比较，直到计算出的N与w(r)的绝对值差值小于整个矢量数据量的5%，通过调整(M,N)子区域的大小控制节点间的数据量误差，此时就将该区域分配给r节点，同时记下结束时的子区域行列号（i，j），那么r节点切片的范围为:

Xminr=xmaxr-1，r>0 xmax=xmin+(i+1)\*k\*TileWidth，r<R-1

xmin，r=0 xmax，r=R-1

ymin=ymaxr-1，R>0 ymax=ymin+(j+1)\*k\*TileHeight，r<R-1

ymin，r=0 ymax，r=R-1

逐行逐列扫描直至该比例尺层级的所有的子区域都被分配到各节点上。如图3所示。

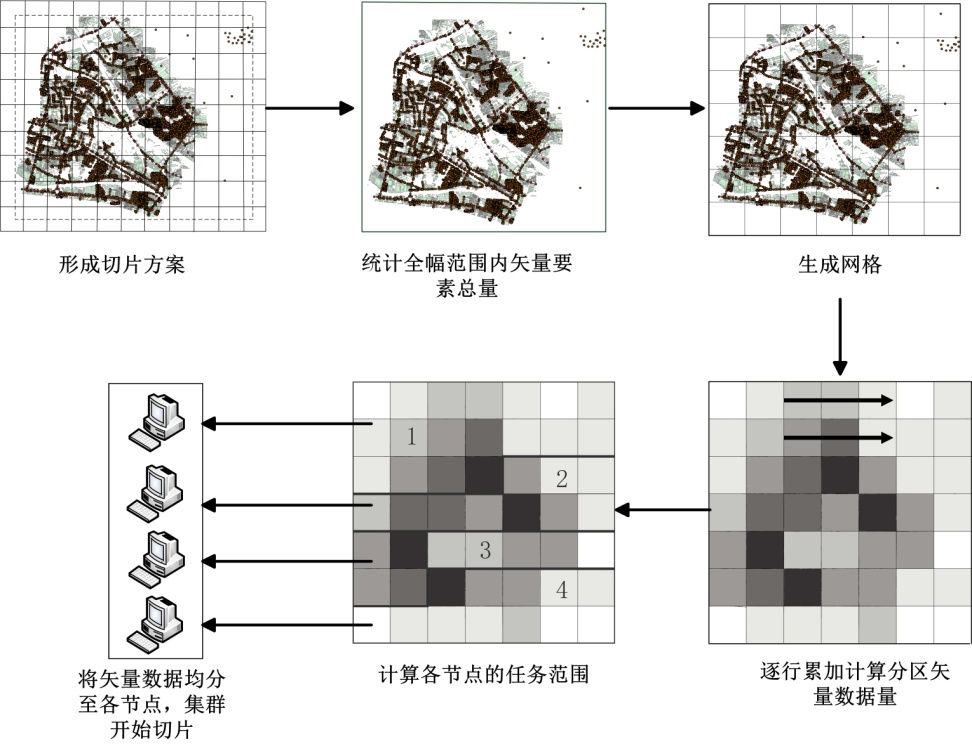


图3. 矢量数据量的任务均分过程

Fig.3 The process of average task division for vector data.

1. 节点负载动态反馈

服务节点的硬件条件不同，决定了地图服务节点的负载能力不同。如果计算负载可以在运行之前确定，可以事先将负载划分，这是静态负载的问题；若只能在运行是测量计算负载并动态确定负载划分，则是动态平衡问题[9]。在首次任务分配完毕后，各节点在动态切片的过程中其CPU占用率，内存使用率，系统 I/O 使用率，网络带宽占用率及剩余切片任务量都是不同的，即使实现了任务量的均分也不可能在同一时间节点同时完成任务，因此需要在切片过程中向负载均衡器反馈资源利用率并计算出在集群中负载率，作为对剩余比例尺层级任务再分配的参考，任务分配的原则是尽量做到分配到每个节点任务量比例与节点自身的负载权值比例一致。设现有w个节点进行切片，每个节点分配到的切片任务范围应该满足区域内部的矢量要素总量大小比例与该节点的负载能力比例Pr一致。因此我们需要按照节点反馈形成的负载信息表对任务范围进行比例划分。

设第r台节点在首次任务分配中得到了P(r)的矢量数据切片量，对应的行列号为N(i,j)，在节点开启线程池实现多个数据块的并行切片，同时添加监控线程，对切片任务进度进行监控，统计已完成切片的子区域空间数据量Loadr及切片开始的时间T，当仅剩两个线程执行任务时（其中一个包括监控线程本身），表明该节点中只剩一个数据块的任务量，此时记下时间T’，表明节点r对Loadr的数据量处理完成时间为Tr=T‘-T，那么节点r在时间段Tr内的处理能力为W(r)=。在集群中记录每个节点对N-1个子区域的矢量数据处理能力，计算负载权值。

定义8 再分配时各节点的负载权值:

P(r)

各节点将自身的负载信息发送至主节点(负载均衡器)，更新负载权值表后，主节点根据各节点根据自身的负载情况，在保证动态负载均衡的基础上实现剩余任务再分发。一直循环直至各级的切片都被分配到各个节点上。

定义9根据权值比例再切片区域w(r)。

w(r)=Pr\*Ntotal

循环过程（1）-（4），最终得到各节点再分配过程中的切片范围，实现了在切片过程中根据矢量数据量的任务分发和动态反馈的负载均衡模式。