第37卷第12期 中国科学技术大学学报 Vol. 37, No. 12
2007年12月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Dec. 2007

文章编号:0253-2778(2007)12-1478-05

一种基于并行计算熵迁移策略的多分辨 DOM 数据生成算法

孙宏元[1,2],谢维信[2],杨 勋[1,2],陆克中[2]

(1. 西安电子科技大学电子工程学院,陕西西安 710071;2. 深圳大学,广东深圳 518060)

摘要:数字正射影像图(digital orthophoto map,DOM)数据通常以多分辨数据形式组织,并以切片的方式存储,而海量多分辨 DOM 数据的生成需要大量计算和大容量存储.针对此问题,提出一种基于并行计算熵迁移策略的并行多分辨 DOM 数据生成算法,以减少海量多分辨 DOM 数据的生成时间.该算法采用并行计算熵来衡量并行计算机系统的负载平衡程度,并以此判断何时需要进行负载迁移以及如何迁移.仿真实验表明,与串行算法相比,该算法能有效减少程序执行时间,并且能获得较高的加速比和并行效率.

关键词:数据正射影像;多分辨;并行计算熵;负载迁移

中图分类号:TP391.41 **文献标识码**:A

An algorithm based on emigrating strategy of parallel computing entropy for DOM

SUN Hong-yuan[1,2], XIE Wei-xin[2], YANG Xun[1,2], LU Ke-zhong[2]

(1. *School of Electronic Engineering, Xidian University, Xi'an* 710071, *China*;
2. *Shenzhen University, Shenzhen* 518060, *China*)

Abstract: Digital orthoimage map (DOM) data are usually organized in the multi-resolution data format and stored in slices. However, massive multi-resolution DOM data generation requires heavy computation and large-capacity storage. A parallel and multi-resolving DOM data generation algorithm was proposed to reduce the generation time of massive multi-resolution DOM data based on the parallel computing strategy of entropy migration. The algorithm employed the entropy not only to measure the load balance level among the nodes in a parallel computing system, but also to determine when and how to emigrate a load. Simulation results show that, compared with sequential algorithms, the parallel algorithm can effectively reduce execution time and achieve higher speedup ratio and parallel computing efficiency.

Key words: digital orthophoto map; multi-resolution; parallel computing entropy; load emigrating

收稿日期:2006-12-17;**修回日期**:2007-01-30

基金项目:国家自然科学基金(60572102),深圳市科技计划项目(200506)资助.

作者简介:孙宏元(通讯作者),男,1962年生,博士/教授. E-mail: hysun@hzu.edu.cn

0　引言

数字正射影像(digital orthophoto map,DOM)是空间信息应用的基础数据,也是实现空间信息应用的重要前提[1]. DOM 数据的主要应用包括作为图层叠加的背景、作为地物提取源数据和三维可视化的纹理数据等. 例如,城市景观的三维重建就可以将 DOM 数据以纹理映射技术贴到数字高程模型网格曲面上,以获得极具真实感的地形渲染.

由于 DOM 数据通常是海量的,通常利用多分辨数据组织方法[2~4],也就是根据人和设备接收或显示信息带宽的实际限制,将海量数据按照信息的分辨率进行组织,达到减少数据传输量和计算量的目的. DOM 数据以多分辨数据结构进行组织,各分辨率的数据层叠加起来,呈现像金字塔一样的四棱锥形状,形象地称其为金字塔形 DOM 多分辨数据结构或塔形 DOM 多分辨数据结构.

海量多分辨 DOM 数据的生成需要大量计算和大容量存储,普通计算机远远不能满足要求. 由于塔形 DOM 多分辨数据结构具有很强的域分解性,因此可以将塔形多分辨 DOM 数据的生成算法架构在并行计算机上. 如同一般并行程序一样,需要考虑如何调度任务及分配资源,以取得良好的性能[5].

1　多分辨 DOM 数据的生成

如图 1 所示,塔形多分辨 DOM 数据结构是由若干层从上到下尺度逐渐变大、逐渐清晰的图像构成,这些图层渐次接近于原始图像. 在传输数据给用户时,可以从顶层(低分辨率)到底层(高分辨率)逐步地发送. 因而,这种结构特别适用于图像的渐进传输.

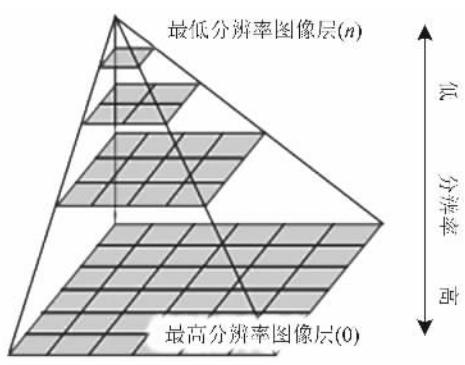

图 1　塔形多分辨 DOM 数据结构

Fig. 1　Data structure of multi-resolution

通常以切片的方式保存多分辨 DOM 中的图像层数据. 切片就是预定尺寸大小(如 128×128)的图像块. 各层图像都可划分为若干个大小相同的切片. 这样,切片就成为组成金字塔的基本单元. 多分辨 DOM 数据的生成也就是为多分辨塔中各图像层生成切片. 用户浏览图像时,一般只需要局部图像,因此可以仅传输浏览点周围的切片. 与传输整幅图像相比,这样的传输方式可大大减少响应时间.

由此,可以把多分辨率 DOM 数据的生成分为图像压缩和图像切割两部分. 其中,图像压缩是生成塔形 DOM 多分辨数据结构中的各图像层. 通常最高分辨率图像层的分辨率和原始图像分辨率保持一致,最低分辨率图像层为一个切片的大小. 相邻两层图像分辨率的比值叫压缩率,常用的压缩率为1/4. 图像切割则是为塔形多分辨 DOM 数据结构中的各图像层生成切片.

在图像压缩时,一般可以对图像进行逐层压缩,即由多分辨 DOM 数据塔的底层向上压缩,每一层图像都是由下一层图像经压缩得到,直至图像为切片大小. 压缩可以采用四邻域平均、小波等算法. 而图像切割也可每次将一个图像块分为四个子图像块,直到切割为切片大小.

图 2 即是一个 3 层塔形 DOM 多分辨数据结构的生成示意图. A 为原始图像,A 经切割后生成 D 中的 4 个图像块;这 4 个图像块再分别经切割后生成 F 中的 16 个切片,F 即为 3 层 DOM 多分辨数据塔的底层. A 经压缩后生成 B,B 的大小为 A 的 1/4,B 经图像切割后生成 E 中的 4 个切片,E 为塔形 DOM 多分辨数据结构的第 2 层. B 再经压缩后生成 C,C 的大小为 B 的 1/4,C 即为多分辨 DOM 数据结构的顶层.

2　并行化和任务运算量分析

对于多分辨 DOM 数据的生成,我们可以定义两种变换和两种类型的生成任务. 两种变换即切割变换 S 和压缩变换 c,切割变换 S 得到输入图像块经切割后的左上、右上、左下、右下 4 个图像块;压缩变换 c 得到输入图像块经压缩后的图像块. 两种类型的生成任务分别为生成切片任务 G 和生成多分辨 DOM 任务 F. 其中,前者为输入图像块生成其切片数据,后者为输入图像块生成多分辨 DOM 数据.

假设原始图像为 p,初始时只一个 F 型任务,设为 $F(p)$,即为原始图像块 p 生成多分辨 DOM 数

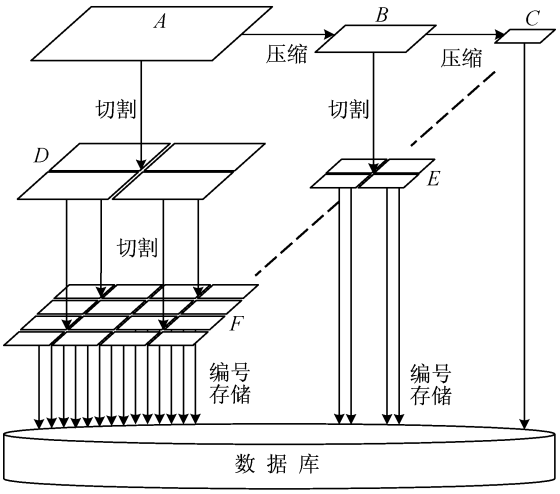

图 2 多分辨 DOM 数据生成示意图

Fig. 2 An example of generating multi-resolution DOM data

据. 原始图像块 p 经一次切割变换(对应图 2 中 $A\rightarrow D$)和一次压缩变换(对应图 2 中 $A\rightarrow B$)后生成 5 个子图像块,即 $S_1(p),S_2(p),S_3(p),S_4(p)$(图 2 中 D 的 4 个子块)和 $c(p)$(图 2 中图块 B). 对于 $S_1(p),S_2(p),S_3(p),S_4(p)$这 4 个子图像块,只要分别为其生成切片数据即可. 对于 $c(p)$这个子图像块,则要为其生成多分辨 DOM 数据,即 $F(p)$经切割变换和压缩变换后生成 5 个子任务:1 个 F 型子任务和 4 个 G 型子任务. 由于这 5 个任务互不相关,因此可并行执行. 对于任意输入图像块 q 的 G 型任务 $G(q)$,则只需对 q 进行切割变换,生成 $S_1(q),S_2(q),S_3(q),S_4(q)$这 4 个子图像块,再分别为其生成切片数据,即 G 型任务经切割变换后可生成 4 个 G 型子任务,并且由于这 4 个子任务互不相关,因此也可并行执行.

由于任务 $F(q)$和 $G(q)$的运算量只与图像 q 的大小相关,因此,当 q 的大小为 $2^n\times 2^n$时,以 $f(n)$表示 $F(q)$的运算量,以 $g(n)$表示 $G(q)$的运算量. 切割变换 $S_1(q),S_2(q),S_3(q),S_4(q)$的运算量与输入图像 q 的大小成正比,令正比系数为 c_s. 压缩变换 $c(q)$的运算量也与输入图像 q 的大小成正比,令正比系数为 c_c. 假设切片的大小为 $2^m\times 2^m$,存储一个切片的运算量为 c_w,由此可以得到 $f(n)$和 $g(n)$的递归求解公式

$$\left.\begin{aligned} f(n) &= f(n-1)+4g(n-1)+2^{2n+2}c_s+2^{2n}c_c\\ g(n) &= 4g(n-1)+2^{2n+2}c_s\\ f(m) &= c_w\\ g(m) &= c_w \end{aligned}\right\} \quad (1)$$

可以先通过迭代法求出 $g(n)$的表达式为

$$g(n)=4^{n-m}[c_w+(n-m)4^{m+1}c_s] \quad (2)$$

将 $g(n)$代入式(1),可以通过迭代法求出 $f(n)$的表达式为

$$f(n)=\frac{4^{n-m+1}-1}{3}c_w+\frac{4^{n+1}-4^{m+1}}{3}c_c+\frac{(3n-3m-1)4^{n+2}+4^{m+2}}{9}c_s \quad (3)$$

由上可知,只要已知 c_w,c_s,c_c的值,就可以求出任务 $F(q)$和 $G(q)$的运算量. 实际上可以根据仿真实验得到 c_w,c_s,c_c的值.

3 基于并行计算熵的并行生成算法

由前面的分析可知,在塔形 DOM 多分辨数据结构的生成过程中,会生成许多任务. 这些任务互不相干,可完全并行执行. 然而,由于其计算量不同,计算过程中,节点间必定会产生负载的不平均. 因此需要对各计算节点上的任务进行动态迁移,以使各计算节点上的负载达到平衡,减少程序执行时间.

本文采用当前各节点运行的所有任务的运算量之和作为节点的负载信息,采用并行计算熵来衡量并行计算机系统的负载平衡程度. 并行计算熵的定义如下:

定义 3.1 假设一个并行计算机系统共有 n 个计算节点,在 t 时刻,第 k 个计算节点的负载为 l_k,其相对负载率为 $q_k=l_k/\sum_i l_i$,则并行计算机系统在 t 时刻的并行计算熵 $H(t)$ 为

$$H(t)=\sum_{k=1}^{n}q_k\cdot\log_2(1/q_k) \quad (4)$$

并行计算熵有两个重要性质:

(Ⅰ) 各计算节点的负载相等时,并行计算熵达到最大值.

(Ⅱ) 并行计算熵增大时,计算节点间的负载更均衡.

并行计算熵的以上两点性质说明其可以很好地衡量节点间的负载平衡程度. 在程序执行过程中,生成的一个调度任务可以周期性地收集各节点的负载信息,并计算出并行计算熵. 如果并行计算熵低于某个阈值 H_c,则进行负载迁移,即将负载较重的节点上的任务迁移到负载较低的节点上. 在每次负载迁移判断中,调度任务进行如下工作:

(Ⅰ) 收集各节点上任务的量,计算各节点的负载;

(Ⅱ) 计算出当前的并行计算熵,如果高于 H_c,则此过程结束,否则进行第(Ⅲ)步;

(Ⅲ) 将负载最重的节点上的适当运算任务迁移到负载最轻的节点上(适宜迁移的任务的运算量应该小于该节点超出平均负载的那部分,否则会造成新的不均衡),如果没有满足此要求的任务则此过程结束;

(Ⅳ) 如果此时并行计算熵高于 H_c,则此过程结束,否则再转到步骤(Ⅲ).

4　仿真实验

在"深超-21C"上运行多分辨 DOM 数据的并行生成程序. 这是一台由 128 个节点,256 个处理器构成的集群式高性能计算机,峰值速度为 1.5 万亿次. 为了使实验结果更准确,对实验结果运行了 50 次后求平均值.

先确定 c_s,c_c,c_w的值,其中 c_s和 c_c分别是切割变换和压缩变换的运算量系数,c_w是写一个切片的运行量.

分别运行切割变换 $S(q)$和压缩变换 $c(q)$,图像 q 的大小为 $2^a \times 2^a$,每个像素用 3 个字节表示,让 a 的值从 8 增加到 13,处理器的频率 f 为 2.8 GHz. 表 1 为不同图像大小下切割变换 $S(q)$和压缩变换 $c(q)$的运算时间及运算量系数 c_s,c_c. 由表 1 可以看出,运算量系数与输入图像 q 的大小基本无关. 因此,我们取 $c_s=4.7$,$c_c=20.5$.

表 1　切割变换和压缩变换在不同输入图像大小下的运算量系数

Tab. 1　Computing coefficient of cutting and compressing under various graphs

a	切割变换时间/s	压缩变换时间/s	c_s	c_c
8	1.10×10^{-4}	4.80×10^{-4}	4.7	20.7
9	5.41×10^{-4}	1.94×10^{-3}	4.7	20.7
10	1.77×10^{-3}	7.65×10^{-3}	4.7	20.4
11	7.05×10^{-3}	3.06×10^{-2}	4.7	20.4
12	2.83×10^{-2}	1.23×10^{-1}	4.7	20.5
13	1.15×10^{-1}	4.89×10^{-1}	4.8	20.4

运行写切片任务,取切片的大小为 $2^7\times2^7$,平均写入时间为0.000 93 s,可以得到 $c_w=0.000\ 93\times2.8\times10^9=2.6\times10^6$.

根据文献[6],我们在实验中取 $H_c=0.9\times H_{\max}$,$H_{\max}$表示并行计算熵的最大值,其值为节点数的对数.

我们分别运行初始图像大小为 $2^{14}\times2^{14}$,$2^{15}\times2^{15}$和 $2^{16}\times2^{16}$的多分辨 DOM 生成程序,即 8 层、9 层和 10 层金字塔. 每个像素以 3 个字节表示. 对于每个初始图像,我们在不同节点数下运行,让节点数从 1 增加到 16(节点数为 1 时表示串行程序). 主要分析并行程序执行时间、并行加速比和并行效率三个参数随节点数变化的实验结果.

图 3 为不同节点数下多分辨 DOM 数据生成程序的执行时间. 从图中可以看出,随着并行计算机系统中节点数的增加,程序执行时间减少.

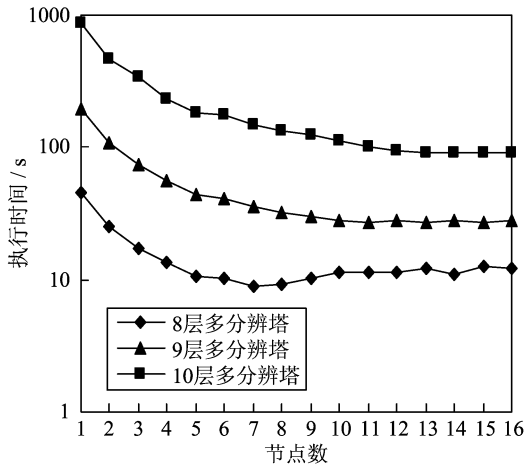

图 3　算法在不同节点数下的程序执行时间

Fig. 3　Executing time under various nodes

由图 3 可以看到,节点数多到一定程度时,节点数的增加并不能显著减少程序执行时间. 这是由于过多的任务迁移,使得负载均衡的代价偏高. 8 层多分辨塔的执行时间在节点数为 5 时已经趋向于最低,9 层多分辨塔的执行时间在节点数为 9 时趋向于最低,而 10 层多分辨塔的执行时间在节点数为 13 时趋向于最低. 这是因为 10 层多分辨的运算量更大,相比之下负载迁移的代价就较小,因此节点数增多时能获得更好的性能.

图 4 为不同节点数下多分辨 DOM 数据生成程序的并行加速比. 从图 4 可以看出,随着并行计算机系统中节点数的增加,加速比会提高,但节点数达到一定程度后,程序的加速比趋于稳定. 10 层多分辨塔的最大加速比要比 9 层多分辨塔的最大加速比大,而 9 层多分辨塔的最大加速比要比 8 层多分辨塔的最大加速比大. 这说明多分辨塔的层数越多,运算量就越大,多节点处理的优势就越大.

图 5 为不同节点数下多分辨 DOM 数据生成程

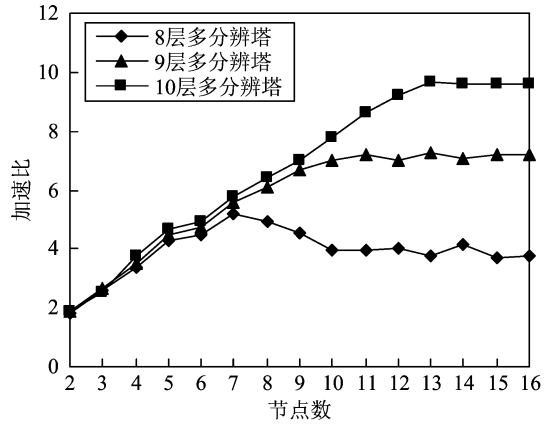

图 4 算法在不同节点数下的加速比

Fig. 4 Speed-up ratio under various nodes

序的并行效率. 从图 5 可以看出,随着并行计算机系统中节点数的增加,程序的并行效率提高. 同样地,节点数达到一定程度后,程序的并行效率下降. 在本文特定实验条件下, 当节点数为 5 时,并行效率达到最高. 这一点可以解释为最初生成的 5 个子任务的运算量差不多,正好分配在 5 个节点上,此时并行计算熵很大,因而负载较为均衡,迁移代价较小,所以并行效率最高.

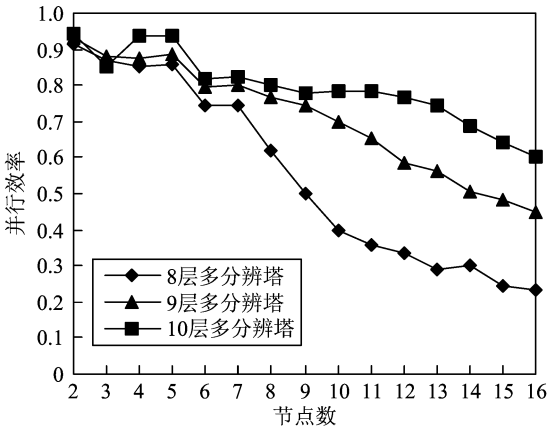

图 5 算法在不同节点数下的并行效率

Fig. 5 Parallel efficiency under various nodes

5 结论

本文通过对多分辨 DOM 数据并行化和任务运算量的分析,提出了一种基于并行计算熵迁移策略的并行多分辨 DOM 数据生成算法. 该算法采用并行计算熵来衡量并行计算机系统的负载平衡程度,并以此来判断何时需要进行负载迁移以及如何迁移. 仿真实验表明,相比串行程序,该算法能有效地减少程序执行时间,提高加速比和并行效率.

参考文献(References)

[1] Sun H, Luo Q, Pei J, et al. A study on spatial horizon overcastting based on GA[J]. Chinese Journal of Electronics, 2006, 15(4a): 933-936.

[2] Goodchild M F. Tiling large geographical database [C]// Design and implementation of large spatial database. New York: Springer-verlag, 1990: 137-146.

[3] Garland M. Multiresolution modeling: survey & future opportunities[R]. Eurographics'99: State of the Art Reports, 1999: 111-131.

[4] Feng Y M, Huang J J, PEI J H, et al. Compression method used to partly rebuild immense image with multi-resolutions[J]. Chinese Journal of Stereology and Image Analysis, 2003, 8(1): 1-5.
冯月明, 黄建军, 裴继红, 等. 一种用于海量图像多分辨率局部重建的压缩算法[J]. 中国体视学与图像分析, 2003, 8(1): 1-5.

[5] 陈国良. 并行算法的设计与分析[M]. 北京: 高等教育出版社, 2002.

[6] Sun H Y, Xie W X, Yang X, et al. An algorithm for the load balance based on parallel computing entropy in HPC[J]. Journal of Shenzhen University, 2007, 24(1): 89-93.
孙宏元, 谢维信, 杨勋, 等. 基于并行计算熵的同构集群负载均衡算法[J]. 深圳大学学报. 2007, 24(1): 89-93.