一种面向复杂地理空间栅格数据处理算法并行化的任务调度方法*

程 果, 陈 荦, 吴秋云, 景 宁
(国防科技大学 电子科学与工程学院, 湖南 长沙 410073)

摘 要:随着并行计算技术的成熟,地理空间栅格数据处理算法的并行化研究成为新的热点。聚焦于处理流程包含多个计算步骤的复杂地理空间栅格数据处理算法,基于空间计算域理论,提出了一个随着算法处理流程而动态变化的任务调度方法。实验证明,该方法在算法流程的每一个计算步都会调整任务分组方案,因此相比于传统任务调度方法,任务调度的负载均衡效果更好,并行算法程序的运行时间更短。

关键词:地理空间栅格数据处理;并行化;任务调度;负载均衡;空间计算域
中图分类号:TP391 **文献标志码**:A **文章编号**:1001-2486(2012)06-0061-05

A task scheduling method for parallelization of complicated geospatial raster data processing algorithms

CHENG Guo, CHEN Luo, WU Qiuyun, JING Ning
(College of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, China)

Abstract: As the parallel computing technologies are becoming mature, the research on parallelization of geospatial raster data processing algorithms has been a hot spot issue. Focusing on the complicated algorithms whose processing procedures consist of multiple computing steps, this research proposes a task scheduling method based on the theory of spatial computational domain with which the task scheduling solution is not static, but adjusts itself as the algorithmic procedure proceeding. Experiments have verified the effectiveness of our method. Because the method keeps adjusting the task scheduling solution at every computing step, the load-balancing effect is better, and thus the parallel running time is shorter compared with the conventional task scheduling method.

Key words: geospatial raster data processing; parallelization; task scheduling; load-balancing; spatial computational domain

随着新一代传感器技术的成熟应用,地理空间数据的空间分辨率、时间分辨率和波段分辨率又有了新的飞跃,更为精细和丰富的地理空间数据可以促使地学领域专家研发更为复杂的处理算法,从而诞生了越来越多的计算密集和/或数据密集型地理空间栅格数据处理算法。这类算法计算强度高,运行时间长,需要依赖更多的新型硬件所提供的计算能力[1]。

并行计算技术是当前解决计算密集型和数据密集型问题的有效手段[2]。并行和分布式GIS的研究始于上世纪90年代中旬,经过这15年左右的发展,产生了很多相关的学术界成果和工业界应用[3]。地理空间栅格数据处理是一类重要GIS算法,本文的研究对象是地理空间栅格数据处理算法并行化时一个重要环节——任务调度。

栅格数据处理的并行化一般基于数据并行思想,将大规模栅格数据集划分为多个可并行处理的子数据集,利用多个计算单元同时地处理多个不同的子数据集,从而实现算法运行时间的缩减[4]。对任一子数据集的处理即为一个计算任务,对所有子数据集的处理即为需要计算的任务集合。任务调度的本质包括如何划分任务,如何分组任务,以及如何将每个分组指派给计算单元执行计算。任务调度的目标是尽可能地实现计算单元间的负载均衡[5]。当前,任务调度相关的研究有基于空间范围的域划分方法[6]和一些新颖的非均匀划分方法[7-8];有静态分配策略和动态分配策略的研究。然而,当面对复杂地理空间栅格数据处理算法时,现有方法都无法达到理想的负载均衡效果。

数据划分一般采用基于空间范围的域划分将大规模空间数据集切割成多个数据块。在域划分理论的基础上,Shaowen Wang提出了一个空间计算域理论,通过构建计算强度估计方程,预测域划

* **收稿日期**:2012-07-09
基金项目:国家863计划资助项目(2011AA120306);国家自然科学基金资助项目(41271403, 61070035);湖南省自然科学基金资助项目(12jj4033)
作者简介:程果(1984—),男,河北邯郸人,博士研究生,E-mail:guocheng@nudt.edu.cn;
景宁(通信作者),男,教授,博士,博士生导师,E-mail:ningjing@nudt.edu.cn

分生成的子数据集的计算强度,来有效地指导任务调度,实现计算单元间的负载均衡[9]。然而,目前空间计算域理论在任务调度方面的应用还仅限于空间矢量数据分析算法的并行化。对于本文研究的地理空间栅格数据处理算法,在构建计算强度估计方程时,由于栅格数据的矩阵结构和栅格处理类算法的遍历型计算,一旦方程选择不够科学,基于方程来估计计算强度所消耗的时间反而会成为一大负担,严重影响算法程序的并行性能。另外,对于本文所关注的复杂地理空间栅格数据处理算法,算法流程中可能会有多个计算特征千差万别的处理操作。计算强度估计方程的构建难度更大,估计精度也会大打折扣。

因此,空间计算域理论不能很好地应用于复杂地理栅格数据处理算法并行化。本文将以空间计算域理论为基础,提出一个适用于复杂地理栅格数据处理算法并行化的任务划分调度方法。

1 任务调度方法

1.1 流程建模

元胞自动机(Cellular Automata,CA)指的是对一个由有限个元胞(Cell)组成的元胞空间(Cellspace)按照一定的规则进行离散的时间变换(Transition)。每次变换都是对元胞空间内的所有元胞遍历地执行同一规则(Rule)的演化。对某指定元胞的演化过程可能不仅依赖于该元胞的前一状态,同时依赖该元胞某一指定邻域内的所有元胞的前一状态[10]。元胞自动机可形式化表示为

$$CA = (Cellspace, list < Transition, Rule >) \quad (1)$$

而对于大多数地理空间栅格数据处理算法,算法流程的开始是输入一个逻辑结构为二维矩阵的地理空间数据集(Geospatial Dataset,GeoDs),数据集中的任一点(Pixel)表示的是某实际地理点的属性值,例如高程值。算法流程的计算部分将包括一个或多个处理步(Processing Step,PStep),其中每一个处理步其本质即为对输入数据集的每一个点,按照某一算子(Processing Operator,POperator)统一地执行一次遍历型计算。算子的输入可能是该点当前属性值,也可能是该点以及该点附近若干点的当前属性值,算子的输出是该点的下一属性值。地理空间栅格数据处理算法(Geospatial Raster Data Processing,GRDP)的流程可形式化表示为

$$GRDP = (GeoDs, list < PStep, POperator >) \quad (2)$$

对比上述分析和式(1)、(2),不难看出,CA 可以用于地理空间栅格数据处理算法建模。通过式(3)所示的映射模型,CA 即可描述大多数地理空间栅格数据处理算法。数量繁多的地理空间栅格数据处理算法被统一地抽象为元胞自动机模型。复杂的算法流程被划分成多个具有前后关系的 transition。

$$GRDP \rightarrow CA, via \begin{cases} GeoDs \rightarrow Cellspace \\ PStep \rightarrow Transition \\ POperator \rightarrow Rule \end{cases} \quad (3)$$

1.2 域划分

接下来,本文采用均匀格网域划分方法将大规模地理空间栅格数据集进行数据划分。由图 1 可见,将 Cellspace 投影到一个均匀空间格网 $Grid(n_x, n_y)$,其中 n_x 和 n_y 分别表示格网的横向和纵

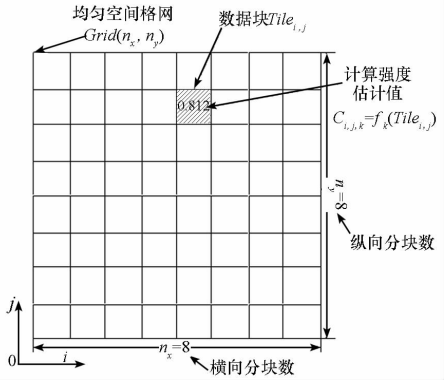


图 1 基于均匀格网的域划分方法及计算强度矩阵

Fig. 1 Domain decomposition based on a regular grid and computational intensity matrix

向分块数。Cellspace 被分解为 $n_x n_y$ 个数据块 $Tile_{i,j}$,其中 i/j 分别表示数据块的横/纵向索引号。假设该算法通过流程建模后被划分成 n_t 个 transition,那么对于其中每一个 transition,都根据算子特点构造一个可估计给定空间范围的数据块的计算强度的方程 $c_{i,j,k} = f_k(Tile_{i,j})$,其中 $c_{i,j,k}$ 表示数据块 $Tile_{i,j}$ 在第 k 个 transition 的计算强度估计值。那么,格网中的 $n_x n_y$ 个数据块将对应生成一个 $n_x \times n_y$ 二维计算强度矩阵

$$\{c_{i,j,k} \mid i \in [1, n_x], j \in [1, n_y]\} \quad (4)$$

1.3 任务分组

任务分组是将域划分所得的数据块集合分成指定数目个集合。考虑到不同计算单元的计算可能会有差别,任务分组的过程还要考虑任务分配的情况。由于同一集合的数据块将被分配至同一计算单元执行处理,本着负载均衡原则,任务分组的标准应该是尽可能实现每一组的计算强度之和与负责计算该集合的计算单元的计算能力成比例。

定义 $C_{n,k}$ 表示第 k 个 transition 的第 n 组的计算强度之和,定义 P_n 为相对应的将要处理该组数据的计算单元的计算能力,上述任务分组操作可以用数学中的任务规划描述,其中,n_g 表示分组的数目。

Minimize:

$$\sigma(C_n/P_n), n \in [1, n_g] \tag{5}$$

每一个分组的 C_n 都是由集合成员决定的,而每一个集合成员的计算强度都来自式(4)所示的计算强度矩阵。这样一个任务规划问题具有很大的时间复杂度,求解最优分组方案很可能需要消耗非常多的时间,这对我们试图降低并行程序的运行时间是得不偿失的。另外,考虑到对于很多地理空间栅格处理算法,在计算过程中相邻的数据块之间需要通过通信传递一些数据点和/或中间结果值。为了减少处理器之间的通信代价,应尽可能将空间上相邻的数据块分至同一集合,由同一计算单元执行计算。综合这两方面的考虑,我们利用空间填充曲线将二维格网串联成一个一维链表。相应地,二维计算强度矩阵也转换为一维计算强度串:

$$\{c_{i,k} \mid i \in [1, n_x n_y]\} \tag{6}$$

空间填充曲线一般用于将多维空间映射成一维空间,通过将多维空间中的每一单元用一条曲线串联,保证空间中的任一单元都只会被访问一次[11]。空间填充曲线最大的优点就是在多维向一维映射的过程中保持了单元之间的空间相邻属性。一维链表中两个相邻的元素在多维空间内必然也是空间相邻的[12]。因此,式(6)所示的计算强度串中,相邻的元素即为空间相邻分块对应的计算强度值。只要将一维计算强度串按顺序分割为几个子串,每一子串集合中的元素对应的数据块必然在二维格网中是空间相邻的。那么,上述任务规划问题相应简化成为

Minimize:

$$\sigma(C_n/P_n), n \in [1, n_g] \tag{7}$$

Subject to:

$$\begin{cases} 1 \leqslant i \leqslant n_g \\ 1 \leqslant p_i < p_{i+1} \leqslant n_x n_y \\ C_n = \sum c_j, j \subset S_n \\ S_n = \{Tile_m \mid m \in [p_n, p_{n+1}]\} \end{cases} \tag{8}$$

其中 p_i 表示第 i 个分割点的索引号,显然,在分组过程中,需要 $n_g - 1$ 个分割点将一维计算强度串分成 n_g 段。假设分割点按照索引号升序排列,则 $1 \leqslant p_i \leqslant p_{i+1} \leqslant n_x n_y$。$S_n$ 表示分割而成的第 n 段,包含的分块表示为 $\{Tile_m \mid m \in [p_n, p_{n+1}]\}$。

二维规划问题被简化为一维规划问题后,时间复杂度大幅度缩减。即便采用最耗时的遍历规划法,遍历法的复杂度为 $O((n_g - 1)^2) = O(N^2)$。对于每一个 transition,都执行上述规划方法实现任务分组。下面,需要将每一个任务分组方案串联生成完整的任务调度方案。

1.4　数据更新和结果传递

在将每一个 transition 的任务分组结果串联的过程中,由于不同的 transition 很可能拥有不同的计算强度估计方程 f_k 和任务划分调度结果,产生以下两个问题。

一是数据更新问题。对于 Transition-k,基于 f_k 计算所得的最优分组结果 $\{S_{n,k}\}$ 可能不同于 $\{S_{n,k+1}\}$,即某一计算单元需要处理的分块集合由 $S_{n,k}$ 变化成了 $S_{n,k+1}$。为此,需要设计一个数据更新方法,当任务分组结果变化时,对每一个计算单元进行代价最小的数据更新。在当前的研究工作中,基于一种主从结构下的并行数据 I/O 方法实现了数据动态更新。

二是结果传递问题。还是由于任务分组结果的变化,不仅需要更新数据,还需要将计算结果及时地在计算单元之间进行传递,以满足下一个 transition 的计算需求。在当前的研究工作中,基于主从结构安排一个控制节点统一地掌控所有数据块的流向,并按照新的调度结果,指挥各个计算节点相互间传递计算结果。

上述这两个问题是本文的任务划分调度方法所带来的负面影响。不过后续实验将会证明,在当前主流的高性能计算集群环境下,负面影响很小,但是相比于负载均衡所带来的优化效果,本文所提的方法依然可以提升算法程序的并行性能。总之,本文所提的任务调度方法包括三个部分:流程建模、域划分和任务分组,具体流程参见图 2。

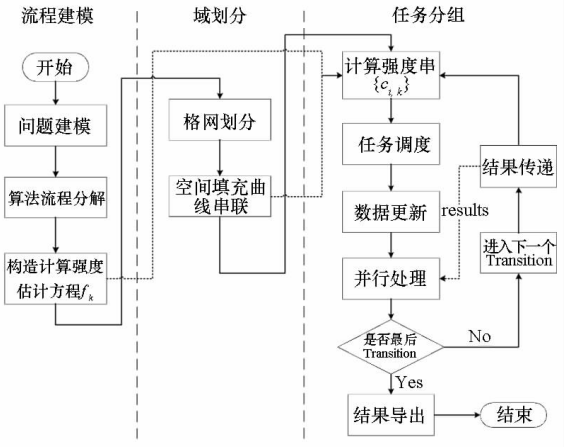


图 2　任务调度方法流程图

Fig. 2　The workflow of the task scheduling method

2　实验验证

实验案例是一个数字地形分析领域的应用，其目的是考察一片地域，评估其中山峰的某一潜在危险程度。算法的流程涉及山峰点的提取、险峰过滤以及危险程度的评估。因此，该算法分解成 3 个 transition，分别对应上述 3 个步骤。

(1) Transition-1、山峰提取

通过遍历输入 DEM 数据集的每一个点，提取出所有山峰点的坐标。山峰点的判定规则就是：指定一个邻域，如果中心点的海拔高于邻域内所有点的海拔，那么该中心点即为山峰点；只要存在一个邻域点的海拔大于等于中心点，则判定为否。

Transition-1 的计算强度依赖于判定次数。对于不同起伏的地形，有些情况下可能只需要一次比值操作即可完成判定；而有些情况下却需要遍历邻域内所有点才能完成判定。本文构造的 f_1 选择了正比例函数，其根据在于当点数量足够大时，每一个点的平均判定时间趋于均值，那么任务的计算强度就应该和数据集包含的点数成正比：

$$f_1 = a_1 \cdot sizeof(ds) \tag{9}$$

其中，$a_1 > 0$ 表示平均判定时间；*sizeof* 表示数据集点数量。可以推测，Transition-1 的计算强度估计精度有限。实验结果也验证了实测计算时间和估计计算强度的二维分布相差较大，这里不再给出展示。

(2) Transition-2、险峰过滤

如果对所有山峰点都进行危险评估，时间复杂度过高。因此这一步将山峰点过滤，留下所谓的“险峰”作为后续危险评估的对象。险峰的评判标准是山峰点的剖面曲率(Profile Curvature)与某一给定阈值的大小关系。剖面曲率越大，意味着该山峰越陡。

剖面曲率的计算属于邻域型计算，因此在并行计算过程中会引入通信问题。不过和演化计算代价相比，通信代价所占的比重很小，因此计算强度主要还是由剖面曲率的计算代价决定的。在实验过程中 f_2 是基于数据集的山峰点密度构造的，其根据在于剖面曲率的计算比较耗时，数据集内山峰点越多，任务的计算强度越大：

$$f_2 = a_2 \cdot \frac{psizeof(ds)}{sizeof(ds)} \tag{10}$$

其中，$a_2 > 0$ 表示剖面曲率计算时间；*psizeof* 表示数据集内山峰点数量。实验结果证明了实测计算时间和估计计算强度的二维分布非常相似，如图 3 所示。

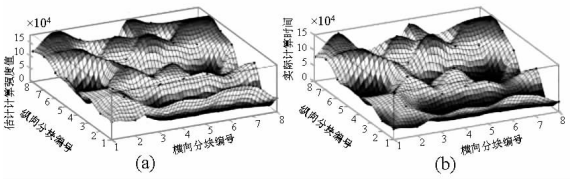


图 3　Transition-2 的计算强度估计值和实测计算时间的二维矩阵分布

Fig. 3　The estimated computational intensity and the practical computing time of Transition-2

(3) Transition-3、危险评估

最后评估险峰的危险系数，按照危险系数排序输出空间坐标。评估标准依然基于剖面曲率。评估方法是逐渐降低山峰点的高程，直到剖面曲率和阈值相同，需要降低的高程值越大，表示危险系数越高，反之亦然。

由于剖面曲率的计算不是一次线性计算，难以根据剖面曲率值反推高程值，那么只能通过迭代的方法，每一迭代步等值递减高程后计算剖面曲率，直至剖面曲率小于等于给定阈值，迭代结束。当险峰点数量比较多时，每一次迭代所用的时间趋于均值，因此计算强度估计方程的构造应该基于数据集的险峰点密度：

$$f_2 = a_3 \cdot \frac{dpsizeof(ds)}{sizeof(ds)} \tag{11}$$

其中，$a_3 > 0$ 表示迭代平均时间；*dpsizeof* 表示数据集内险峰点数量。实验结果证明了实测计算时间和估计计算强度的二维分布不完全一样，但是有着大致相似的起伏度，如图 4 所示。

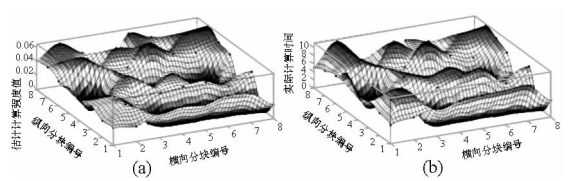


图 4　Transition-3 的计算强度估计值和实测计算时间的二维矩阵分布

Fig. 4　The estimated computational intensity and the practical computing time of Transition-3

接下来，利用 1.3 小节提出的任务规划方法，将所有任务分成 8 组，分配至 8 个计算能力相同的处理器，每个处理器的 Transition-2 和 Transition-3 计算时间分别记录，并和采用传统任务调度方法时各个处理器的计算时间对比。

如图 5 所示，由于传统方法在任务调度时没有估计计算强度，无法实现负载均衡，各个处理器负责计算的数据块虽然空间范围和数量相同，但是受到数据分布和算法特性的影响，各个处理器的计算时间相差很大。相对而言，当采用本文所

提的方法时,各个处理器计算时间分布线明显更为平坦,意味着彼此之间的时间差别不大。由于总计算量是确定常量,而本文方法可以更均匀地平摊负载,因此采用本文方法后,算法的并行运行时间有所降低。另外,对比图 5 的两个子图可见,Transition-2 的处理器时间分布相比于 Transition-3 更为平坦,这验证了上述推论——估计更为准确的计算强度方程将会带来更好的负载均衡效果。

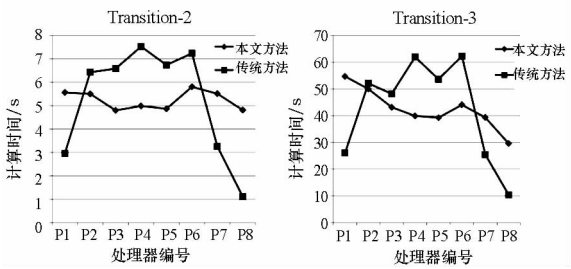


图 5　两种方法下各个处理器计算时间分布对比

Fig. 5　The comparison of computing times on all processors between two methods

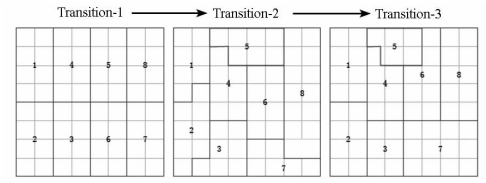


图 6　任务调度方案随流程推进而调整

Fig. 6　The task scheduling solution adjusts itself as the procedure proceeding

最后,将 3 个 transition 的分组结果串联,形成了最终任务调度方案。如图 6 所示,使用本方法生成的任务调度方案不是一成不变,而是随着算法流程的推进而不断调整的。实验程序分别测试记录了每个处理器消耗的时间。参见图 7 可知,虽然由于数据更新、结果传递以及计算强度的估计,本文方法会额外引入部分时间代价(在本例中大约 2s 多的时间消耗),但是相比于传统方法,这种动态变化的任务调度方案使得每个 transition 的负载更为均衡。算法程序的运行时间(即处理器消耗总时间中的最大值)降低比率达 14.8%,并行加速比由 5.564 提升至 6.388。

3　小结

本文针对复杂地理空间栅格数据处理算法的并行化问题,提出了一种任务调度方法。相比于已有研究方法,本方法的最大创新之处是一改一劳永逸式任务调度策略,而是将复杂算法进行流程建模分解,根据每一步的计算特征动态地改变任务分组结果,形成了一个随算法流程推进而动态调整任务调度方案。实验证明了本任务调度方法具有很好的负载均衡效果。

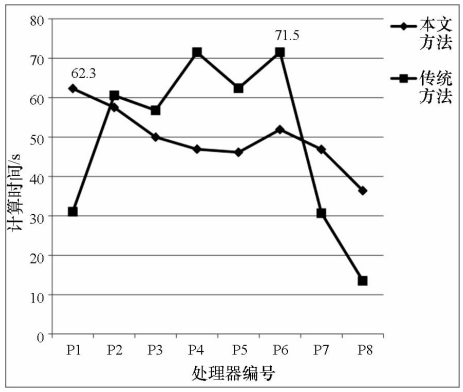


图 7　两种方法下各个处理器消耗总时间分布对比

Fig. 7　The comparison of total cost times on processors between two methods

下一步研究的内容有两点:一是着眼于降低 1.4 小节中提到的数据更新和结果传递带来的负面影响;二是通过更多更复杂的案例来完善本任务调度方法的细节,例如域划分时如何确定最优分块粒度,以及任务分组时如何选择空间填充曲线。

参考文献(References)

[1] Armstrong M P. Geography and computational science[J]. Annals of the Association of American Geographers, 2000, 90(1):146 - 168.

[2] Kuck D J. high performance computing: challenges for future systems[M]. New York, NY: Oxford University Press, 1996.

[3] Clematis A, Mineter M, Marciano R. High performance computing with geographical data[J]. Parallel Computing, 2003, 29(10):1275 - 1279.

[4] Wang S, Liu Y. TeraGrid GIScience gateway: bridging cyberinfrastructure and GIScience[J]. International Journal of Geographical Information Science, 2009, 23(5):631 - 656.

[5] Li X, Zhang X, Yeh A, et al. Parallel cellular automata for large-scale urban simulation using load-balancing techniques[J]. International Journal of Geographical Information Science, 2010, 24(6):803 - 820.

[6] Armstrong M P, Densham P J. Domain decomposition for parallel processing of spatial problems[J]. Computers, Environment and Urban Systems, 1992, 16(6):497 - 513.

[7] Guan Q, Clarke K C. A general-purpose parallel raster processing programming library test application using a geographic cellular automata model[J]. International Journal of Geographical Information Science, 2010, 24(5):695 - 722.

[8] Wang S, Armstrong M P. A quadtree approach to domain decomposition for spatial interpolation in Grid computing environments.[J] Parallel Computing, 2003, 29(10):1481 - 1504.

[9] Wang S, Armstrong M P. A theoretical approach to the use of cyberinfrastructure in geographical analysis[J]. International Journal of Geographical Information Science, 2009, 23(2):169 - 193.

[10] Bandini S, Mauri G, Serra R. Cellular automata: from a theoretical parallel computational model to its application to complex systems[J]. Parallel Computing, 2001, 27:539 - 553.

[11] Mokbel M F, Aref W G, Kamel I. Analysis of multi-dimensional space-filling curves[J]. Geoinformatica, 2003, 7(3):179 - 209.

[12] Sagan H. Space filling curves[M]. New York: Springer-Verlag, 1994.