分布式并行地形分析中数据划分机制研究*

宋效东[1],窦万峰[2],汤国安[1],江 岭[1],赵 菁[2],赵明伟[1]
(1. 南京师范大学 虚拟地理环境教育部重点实验室,江苏 南京 210046;
2. 南京师范大学 计算机科学与技术学院,江苏 南京 210046)

摘 要:数据粒度是海量空间数据并行计算的重要问题之一。通过对不同性质的并行算法的对比分析,提出空间数据粒度模型,量化地反映并行地形分析中数据划分的规模,建立并行数据粒度评价模型。通过研究集群环境下不同算法的数据并行数据粒度问题,提出基于并行数据粒度评价模型的优化数据粒度调度算法。通过计算每一次并行计算的时间与数据粒度效率,从而实现对计算数据粒度动态更新以追求更高的加速比。经过实验验证,该算法较之传统算法,可提供更高的任务执行效率并具有更好的可移植性。

关键词:并行计算;数字地形分析;数据划分;数据粒度
中图分类号: TP311 **文献标志码**: A **文章编号**:1001-2486(2013)01-0130-06

Research on data partitioning of distributed parallel terrain analysis

SONG Xiaodong[1], *DOU Wanfeng*[2], *TANG Guoan*[1], *JIANG Ling*[1], *ZHAO Mingwei*[1]
(1. Key Laboratory of Virtual Geographic Environment of Ministry of Education, Nanjing Normal University, Nanjing 210046, China;
2. School of Computer Science and Technology, Nanjing Normal University, Nanjing 210046, China)

Abstract: Data granularity is one of the most important issues of parallel computing based on large volume of spatial data. After a comparison with the different types of terrain analysis algorithms, a Geo-Data Granularity Model (GDGM in short) was proposed, which can be used for the quantitative description of data partition granularity in parallel computing process of massive spatial data. In this algorithm, according to the parallel evaluation method, the parallel data granularity was adaptively adjusted and finally an optimized data grain was obtained. The execution time of each parallel computing was recorded for the comparison of computing efficiency values from different data granularities. Furthermore, by means of the comparison, a dynamic algorithm was designed for the dynamic scheduling of different data granularity so that the optimal performance of specific algorithm was achieved. The preliminary experiments show that the algorithm has much better efficiency and portability than the traditional ones so far.

Key words: parallel computing; digital terrain analysis; data partition; data granularity

从目前的发展趋势来看,DEM已成为GIS的核心数据库和地学分析应用的基础数据。各应用领域要求能在本地、远程层次上对DEM数据进行实时访问,希望DEM能与遥感影像数据、矢量特征数据、多媒体数据等数据进行融合和信息的复合处理。空间地形分析在国民经济中的应用以及高分辨率、高精度DEM的大量获取,使得数字地形分析并行技术的研究成为当今高性能地学计算发展的重要内容[1-4]。

分布式并行计算是当前计算机科学研究的热点之一。并行和分布式计算是求解各种计算密集型、数据密集型负载调度问题的有效手段,而并行数据粒度(用于计算的划分和负载均衡控制)是影响并行与分布式计算性能的关键因素。

数据并行是指把数据划分成不同部分并分别映象到不同的处理器上,处理器运行同样的处理程序对所分派的数据进行计算。不同的处理器在计算过程中需要进行一定量的通信。因此,在这种数据并行处理过程中需要根据串行算法的特点进行合理的并行化设计,以减小不同处理器间的通信对并行程序性能的影响。如果从理论上的角度分析数据并行的性能,则整个问题的算法的计算速度与数据划分策略及大小无关[5-6]。然而,在实际的并行计算中,经常由于问题划分情况的不同而影响整个算法的计算速度[7]。

本文首先分析了高性能计算系统中数据粒度

* **收稿日期**:2012-07-09
基金项目:国家863计划资助项目(2011AA120303);国家自然科学基金资助项目(41171298);资源与环境信息系统国家重点实验室开放基金资助项目(2010KF0002SA);江苏省普通高校研究生科研创新计划项目(CXZZ12_0393)
作者简介:宋效东(1986—),男,河南商丘人,博士研究生,E-mail:xiaodongfly@yahoo.com.cn;
汤国安(通信作者),男,教授,博士,博士生导师,E-mail:tangguoan@njnu.edu.cn

问题,通过给出的数据粒度优化模型进行了改进。由于不同算法的执行过程、算法复杂度以及对数据依赖程度的不同,本文提出基于优化数据粒度的调度算法。针对高性能地学分析中数据并行类的算法,通过两个不同复杂度的算法对比了最优数据粒度的计算。结果表明,不同性质的算法具有不同的最优计算数据粒度,动态数据划分算法可以很好地解决此类问题。

1　数据粒度优化模型

1.1　粒度问题

粒度是人们对问题的不同角度、不对层次进行细化的一种度量。高性能计算中将粒度作为调度的基本单位,主要包括任务粒度和数据粒度。任务粒度一般是用来描述分配到并行计算体系的各计算节点计算任务的大小。评价任务粒度的合理性主要是判断其任务划分是否具有灵活性、是否避免了冗余计算和存储、任务尺寸是否大致相当等方面。数据粒度是并行计算中数据划分的基本单位,可以通过计算环境初步估算。通常情况下经过数据划分后其大小不会发生变化,属于静态数据粒度(如图 1 所示)。

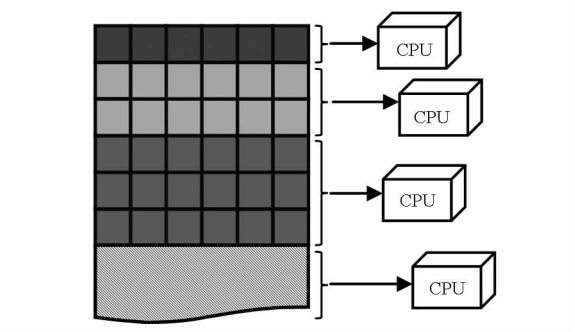

图 1　多粒度并行算法示意图

Fig. 1　Parallel computing with multi-granularity

静态数据粒度具有先天缺陷:(1)在很大程度上依赖于初始数据粒度的选择,若初始分类严重地偏离全局最优计算数据粒度时,可能会导致花费过多的系统开销;(2)过多的信息交换可能会导致潜在的安全性问题;(3)不适合异构计算环境的高吞吐量计算。众多学者根据特定的算法设计了不同数据粒度的算法,并取得较好的计算效率[8-9]。

高性能地学计算包含众多的分析算法,需要处理不同形式的空间数据。这些算法间存在不同程度的耦合度,是并行化设计的难点之一,如邻域型地形分析算法需要一定的缓冲区来解决边界问题。另外,空间数据的全局性以及拓扑关系给并行化设计带来了一定的难度。

依据分析窗口及计算特性,地形分析算法基本上分为局部地形算法和非局部地形算法;根据地形要素的关系特征,地形属性又可归纳为地形曲面参数、地形形态特征、地形统计特征和复合地形属性等四种[10]。典型的地形形态特征算法如洼地填平算法、平地流向和可视性分析算法等需要动态地从全局 DEM 上进行提取。该类算法的并行化只能依靠通信机制来实现全局信息的交换,然而通讯成本与数据并行的粒度又有密切关系。相关的实验分析证明,不同特征的地形分析算法只有在合理的数据粒度下才能取得最优的加速比[11-12]。对于局部型地形算法可以提高重复计算或增加分析窗口缓冲区减少通讯量。

在分布式并行计算环境下,算法的算法复杂性不同,如迭代与递归等,导致不同数据粒度的最优计算效率有所区别。其次,由于最大内存限制的要求,不同算法在处理超过最大内存限制的数据时必须要将超额的数据暂时存放在硬盘中,这就导致了过多的时间消耗在数据的频繁读写中。本文以并行地形分析为例,重点讨论栅格数据并行化设计中的数据粒度问题。

1.2　模型构建

格网 DEM 是存储了包含高程数值的二维数组,格网点的平面坐标隐含在行列号中。因为其数据结构简单、良好的表面分析功能得到广泛的应用。DEM 所包含的信息量越大(如区域性 DEM、高分辨率 DEM),其数据量往往也越大,远远超过了一般计算机的处理能力。为了缓解海量 DEM 数据的低效率操作与处理,本节针对规则格网模型提出一种量化的粒度控制模型。

面向分布式并行计算环境,并行程序应有良好的伸缩性。数字地形分析并行算法最重要的目的是处理海量的空间数据,因此面向分布式并行计算的地形分析算法需要重点考虑数据的管理机制。

定义 1　空间数据粒度模型 GDGM(Geo-Data Granularity Model)。定义一个四元组,GDGM = (DG, R, D, P),DG 是最小数据粒度,R 是粒度遵循的规则,D 是最优粒度的论域,P 是该任务计算的属性。各元素详细说明如下:

(1)最小数据粒度,又称原子粒,是并行计算中数据分配中不可拆分最小的基本单位。假设 S 是一幅 DEM 数据所占的存储空间(以 KB 为基本单位),D_{min}是划分为原子数据粒度所占的存储空

间，D_i 是第 i 次并行调度的数据粒度大小。假设系统的内存替换及调度的最小单位均为4KB，则原子数据粒度的大小用式(1)表示：

$$D_{\min} = f \times 4\text{KB} \tag{1}$$

在式(1)中，f 是原子数据粒度的计算系数，根据一幅数据大小取值，取值范围为[1, N]，为正整数。N 的计算方法为：

$$N = \left\lfloor \left(\frac{S}{4} \right) \div 4\text{KB} \right\rfloor \tag{2}$$

(2)粒度遵循的规则是指衡量粒度的适用性指标，可以表现为算法效率、加速比等。该规则也包含其他的影响因素，如任务依赖、异构计算环境、通信环境等因素。传统的并行计算模型是先进行数据分解然后再进行任务分解，计算集和数据集互不交互。这就要求数据粒度必须静态分解，很难适应复杂多变异构计算环境以及不同种类的非局部地形算法。这里采用一种简易的粒度效率计算方法，详见下一节。

(3)论域：一次并行计算过程中所有的数据粒度，各数据粒度不等。根据通信环境的动态更新或不同种类的算法，最优计算数据粒度可能也不是常数。最优计算数据粒度的定义如下：

$$D_{op} = 4^k \times D_{\min} \tag{3}$$

在式(3)中，k 的取值范围为[1, H]，且为正整数，H 为常规四叉树的深度。

(4)任务属性是指空间数据划分后的各种附加信息，以格网 DEM 为例，其数据属性包含分辨率、高程点数据类型、行列数、冗余行列、最小数据粒度以及最大内存限制等。

1.3 并行数据粒度评价模型

在分布式并行系统中处理海量地学数据时，当计算资源有限时，即所有的计算不能一次性地全部执行时，需要按照最大内存限制对各节点的数据分发进行控制。同时，由于通信及算法的全局性问题各节点处理相同大小数据集需执行的算法迭代次数也各不相同，这就导致同样大小数据集在不同的处理单元、不同地形分析算法的计算时间各不相同。对各节点的数据分配量进行优化可以优化提高系统的总执行效率，各节点按照最优的执行效率并行执行计算任务。系统对整个数据分配的要求主要是各节点计算的总数据量不能超过各节点的执行能力、最大完成时间，破坏各节点的负载平衡等。

定义如下参数：分布式并行计算中节点个数为 n(需执行的任务总数为 n)，任务 i 在处理单元 j 上的计算时间为 $t_{i,j}$，则该任务的执行时间可表示为：

$$T_i = \{t_{i,1}, t_{i,2}, t_{i,3}, \cdots, t_{i,n}\} \tag{4}$$

其中 $1 \leqslant i \leqslant n$。通常情况下任务总数与数据量有如下关系：

$$n = \left| \frac{S}{MEMLIMT} \right| \tag{5}$$

$MEMLIMT$ 是最大内存限制，也即单节点最大的处理能力。

定义2 数据粒度效率：指各处理器的计算能力在当前分配数据量和计算任务需求下计算效率，具体描述为：

$$E_{DG} = \frac{D_{temp}}{T_{cost}} \tag{6}$$

D_{temp} 是当前的数据粒度的数据量大小，T_{cost} 是计算当前数据粒度的时间消耗。如果 D_{temp} 相同，则时间消耗越小，E_{DG} 越大。

定义3 最大内存行限制：在最大内存限制条件下，节点最大处理当前 DEM 的行数。

$$MaxRow = \frac{MEMLIMT}{Column \times Nbyte} \tag{7}$$

$Column$ 是当前栅格的列数，也即是一行数据所有的栅格点数，$Nbyte$ 是每一个栅格点所占的字节数。

2 优化粒度调度算法

由于并行算法的复杂性，这里暂时不考虑地形分析算法的任务依赖关系。基于空间数据粒度模型的并行算法流程如图2所示，本文提出了如下的数据划分算法，并给出必要的任务调度方法：

(1)进行数据并行化，即将串行计算任务转换为并行的计算任务，并将数据划分为独立的个体分发给各计算节点。这里的数据粒度可能是一个分析窗口、多个行或列、也可以以文件为基本单位。

1)对串行任务 t_i 进行语义分析，将其中可并行计算的部分划分出来(如对一副 DEM 求解坡度，各进程对不同的数据进行相同操作、各次循环不存在依赖关系的迭代等)，然后尽可能多地提取出若干相互独立的并行子任务 $t_1 \cdots t_m$；

2)这里如果 t_i 存在子任务并且可以继续划分，则参照(1.1)对子任务 $t_1 \cdots t_m$ 进行逐个划分其并行化操作；如果 $t_i (0 < i < m)$ 不能进行并行化，该算法的并行化结束。否则，依次将 $t_1 \cdots t_m$ 进行并行化调度。

(2)各节点并行的执行分析算法，然后将计算结果返回给主节点。

(3)主节点按照数据粒度遵循的规则,计算并记录一次计算的数据粒度计算效率。

1)当一次子任务计算结束后,主节点计算 E_{DGi},并判断比较上一数据粒度效率,如果 $E_{DGi} > E_{DGi-1}$,则继续增大计算数据粒度,即:$D_{i+1} = 2 \times D_i$;否则 $D_{i+1} = D_i$;

2)如果 E_{DG} 停止增长,则 $D_{i+1} = D_i \times 1.5$,继续计算下一次子任务的粒度计算效率,并比较 E_{DGi};如果 E_{DG} 增长,则 $D_{i+1} = D_i \times 1.5$,否则 $D_{i+1} = D_i$。终止数据粒度计算效率的更新及计算;

3)主节点记录不同粒度计算后的时间以及粒度计算效率,并得出调度所需的不同数据粒度计算时间的估计模型:

$$T_i = Max(T_{Di}) + T_{com} + T_{wr} = \frac{D_{temp}}{E_{DG}} + T_{com} + T_{wr} \quad (8)$$

T_{com} 为通信开销,T_{wr} 是数据读写耗时。

(4)更新计算数据粒度,各进程按照新的数据粒度进行计算,然后循环则对子任务进行步骤(3)的操作。

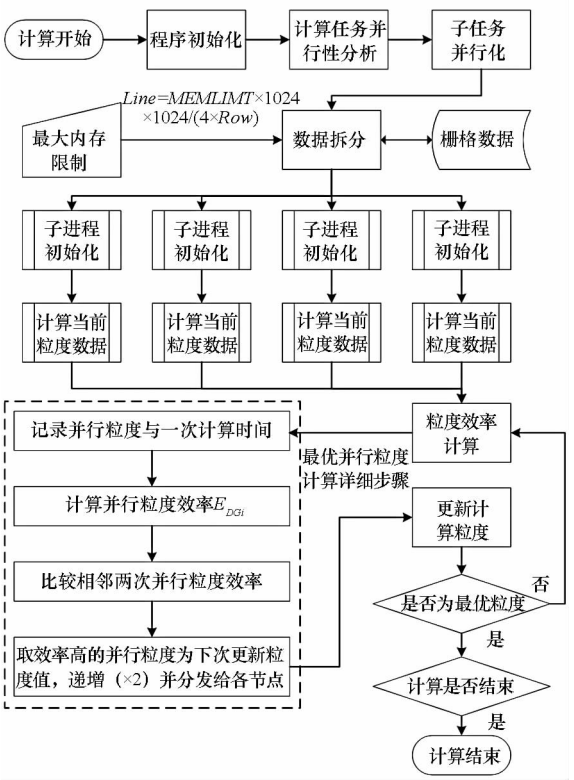

图 2　优化数据粒度算法执行流程

Fig. 2　Workflow of optimized-grain algorithm

3　实验分析

为对比多数据粒度的并行效率,本文设计了两个地形分析并行算法的实验——曲率和填洼算法,并在小规模的机群系统上实现并测试。实验数据为 SRTM(TIFF 格式),数据大小为:16300 × 17400,465MB。性能测试的环境为:9 台微机构成机群结构(1 个管理节点,8 个计算节点);节点配置如下:Intel Core 2CPUs,2.4 GHz,2048MB 的内存,通过 1000Mbps 的快速交换式以太网互连,采用 GDAL1.8 和 MPICH2 消息传递并行库。为了测试最大内存限制对并行计算的影响,这里设定最大读取行数为 800。

3.1　实验 1

地形曲率是表达地形曲面结构的主要参数之一,是地表过程模拟、土壤侵蚀模型、土地利用分类等环境模型的基本变量。该算法是基于邻域窗口分析的一类地形因子。地形曲率计算方法主要有不完全四次曲面方法和二次曲面方法[13],曲率的种类也较多,这里以二次曲面方法的剖面曲率为例。

初始数据粒度 $D_{\min}$ 设定为 100 行,然后每次计算结束后 $D_{i+1} = 2 \times D_i$。不同数据粒度计算时间如图 3(a)所示,随着处理器数目的增大,加速比明显增大。图 3(b)是并行计算数据粒度的动态更新对计算效率及加速比的影响。当数据粒度上升到 800 以后,并行加速比和效率基本保持不变,通过对比不同的计算效率,程序的并行数据粒度就采用 800 并不再发生改变。为了测试超过内存限制的计算数据粒度的计算效率,程序指定了 1600 行和 6400 行两个并行数据粒度,其计算加速比与最大并行数据粒度基本保持一致。这是因为,受到内存限制的原因,虽然指定了超过内存限制的计算数据,但是在计算过程中将超过内存限制的并行数据粒度以内存限制为单位进行划分,实际的并行数据粒度依然是最大内存限制的 800 行。

这里最优并行数据粒度与系统内存限制基本相等,传统的数据划分方法与本文提出的方法并无太大出入,其主要原因是曲率算法属于邻域分析算法,进程间通信较少,计算量容易度量且各进程上执行的计算量基本相似。因此,该算法最优数据粒度的选择与计算节点最大处理能力相关性较大。

3.2　实验 2

从数字高程模型中提取河网水系需要经过一系列的预处理,其中最复杂的部分是对 DEM 中洼地的处理。洼地聚合和嵌套式洼地合并过程被认为是洼地填平算法中计算较为复杂的一类操作。洼地涉及全局,大小不固定。本文采用 Planchon 算法进行并行化设计[14],为了保证并行填洼中洼地的正确处理,各节点在判定边界线上是否存在

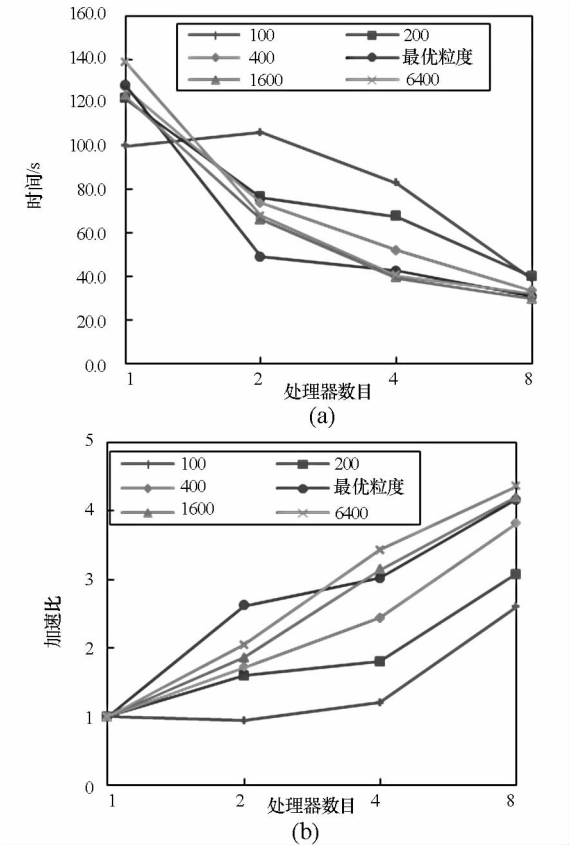

图3 不同数据粒度的曲率算法并行性能测试

Fig. 3 Cost of parallelism of profile curvature algorithm with different granularity

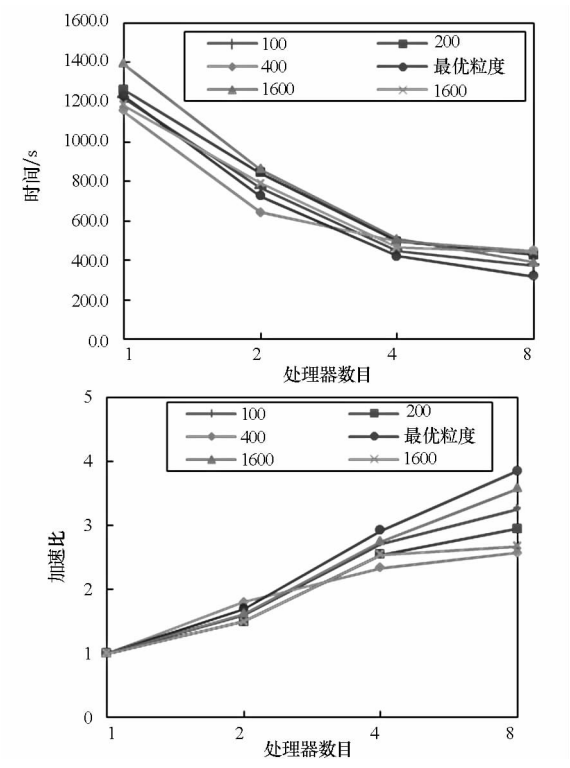

图4 不同数据粒度的填洼算法并行性能测试

Fig. 4 Cost of parallelism of pit remove algorithm with different granularity

洼地时可能需要访问更多的数据。当利用一定缓冲区无法判断边界线上是否有洼地时,该节点将把数据读取范围扩大一倍进行遍历,直到处理所有洼地。当计算数据粒度较大并超过系统的最大内存限制,程序将无法处理的数据转存在外存。

图4是不同计算数据粒度的并行性能测试。随着并行数据粒度的增加,在上升到650行时达到最优数据粒度。当并行数据粒度过大或过小时,会导致更多的计算耗时,计算节点将花费更多的时间用来处理边界上的洼地区域,这就导致部分区域的多次重复计算。对比发现,本文提出的算法能够快速地根据数据粒度效率得出优化的计算数据粒度。

4 相关工作比较

并行计算中增大计算数据粒度是否减少了通讯成本?是否减少了并行执行的机会?在当前地学数据呈几何级指数增长、地形分析算法复杂多样、计算资源与计算架构多变的背景下,这些问题一直是并行算法设计的关键问题。为了减少任务映射的目标,减少算法的执行时间,传统的并行化算法采取将并发的任务映射到不同的处理器,尽可能地将任务之间存在高速通讯的映射到同一处理器。但是,映射是一个NP完全问题,需要根据不同的计算要求与计算环境尽可能地对算法本身进行优化。

目前,已被提出的一些粒度控制算法主要集中在任务并行的粒度对并行效率的影响,部分学者提出了动态控制算法[9,15]来解决计算任务分割问题。这些方法主要是根据算法、数据与计算环境动态地对计算粒度进行控制,如果计算任务过大,就将计算任务进行分解,分配到其他的计算节点。该方法取得了较好的负载均衡与系统效率。

Christos[16]等人提出的多粒度并行约束Delaunay网格生成算法,算法针对数据划分的不同对象进行并行化设计,只适合面向多层次数据的复杂算法。通常情况下DEM只有一个数据层,并未涉及其他的划分对象,而且地形分析算法也复杂多样。尽管本文提出的方法面向单数据层的DEM数据与单算法的并行化,根据算法的复杂性,动态地对数据并行的粒度进行控制,期望在改进算法并行度的同时也提高算法的计算效率。

为了克服MPP计算环境下过多的同步通信消耗问题,McCombs[9]等提出将众处理器分组的多粒度并行算法,部分处理器组处理解决向量块问题,其余的处理器进行更细粒度的计算。该方

法为多任务并行问题提供了更好的延迟容忍能力,使系统达到了更高的并行度,提升了整个系统的效率。该多粒度模型能够根据不同的计算环境预测最佳的计算效率,但是它忽略了处理海量数据时的数据并行粒度问题。Lee 等虽然指出问题的规模不同,并行算法的性能随着并行粒度的不同而出现截然不同的效率[8],并证明当算法涉及大量的内部交互时,细粒度的并行程序具有较高的效率,但是未对具体并行粒度的量化进行深入讨论。本文模型中增加了量化的机制来深入讨论数据并行问题,以及同样数据量时不同并行粒度对计算效率的影响,避免处理海量数据时计算时间、效率预测不准确问题。

DAVID[15]等通过估计并行程序动态任务剖分时的消耗和效率,提出利用细粒度线程在分布式并行计算环境下的动态任务并行算法。该算法使通信与计算交替覆盖,对于任务并行算法具有指导意义,但是过细的线程将导致任务划分过多,可能出现局部的负载不均衡问题。在本文的算法中,通过最优粒度控制模型充分利用了计算资源的处理能力。同时,多粒度的程序设计方式也提高了并行程序的加速比。

5　结论

对于数据分块策略,为充分开拓地形分析算法的并发性和可扩展性,数据划分阶段常常忽略处理器数目和目标机器的体系结构,因此动态调整数据并行数据粒度以适应不同的计算环境显得愈加重要。数据粒度模型能够通过对数据并行的量化来提升高性能地学分析的计算效率。实验分析表明,本文方法具有很好的可行性和稳定性,动态调度策略取得较为理想的计算效率,可以有效缓解分布式并行计算效率低的问题。此方法有待于在更多的领域内应用,以检验其性能。本文提出并行模型的粒度控制策略具有较高的可移植性,为不同的并行环境提供了参考。但是,在未来复杂多变的异构计算环境下如何仿照数据粒度模型对任务粒度模型进行深入剖析尚有待于进一步的研究。

参考文献(References)

[1] Ortega L, Rueda A. Parallel drainage network computation on CUDA[J]. Computers & Geosciences, 2010, 36(2): 171 – 178.

[2] 程果,景宁,陈荦,等. 栅格数据处理中邻域型算法的并行优化方法[J]. 国防科技大学学报, 2012, 34(4): 114 – 119.
CHENG Guo, JING Ning, CHEN Luo, et al. Parallel optimization methods for raster data processing algorithms of neighborhood-scope [J]. Journal of National University of Defense Technology, 2012, 34(4): 114 – 119. (in Chinese)

[3] Schiele S, Möller M, Blaar H, et al. Parallelization strategies to deal with non-localities in the calculation of regional land-surface parameters[J]. Computers & Geosciences, 2012, 44: 1 – 9.

[4] Li T, Wang G, Chen J. A modified binary tree codification of drainage networks to support complex hydrological models[J]. Computers & Geosciences, 2010, 36(11): 1427 – 1435.

[5] Alain J M, Sophie L B, Zhen L. An analytical approach to the performance evaluation of master-slave computational model [J]. Parallel Computing, 1998, 24(5, 6): 841 – 862.

[6] Sun X H, Gustafson J L. Toward a better parallel performance metric[J]. Parallel Computing, 1991, 17(10 – 11): 1093 – 1109.

[7] 舒继武,郑纬民,沈美明,等. 大规模问题数据并行性能的分析[J]. 软件学报, 2000, 11(5): 628 – 633.
SHU Jiwu, ZHENG Weimin, SHEN Meiming, et al. Performance analysis for massive problem data parallel computing[J]. Journal of Software, 2000, 11(5): 628 – 633. (in Chinese)

[8] Lee J, Pillardy J, Czaplewski C, et al. Efficient parallel algorithms in global optimization of potential energy functions for peptides, proteins, and crystals [J]. Computer Physics Communications, 2000, 128(1 – 2): 399 – 411.

[9] McCombs J R, Stathopoulos A. Parallel, multigrain iterative solvers for hiding network latencies on MPPs and networks of clusters [J]. Parallel Computing, 2003, 29 (9): 1237 – 1259.

[10] 周启鸣, 刘学军. 数字地形分析[M]. 北京: 科学出版社, 2007.
ZHOU Qiming, LIU Xuejun. Digital terrain Analysis[M]. Science Press, 2007. (in Chinese)

[11] Gong J, Xie J. Extraction of drainage networks from large terrain datasets using high throughout computing [J]. Computers & Geosciences, 2009, 35(2): 337 – 346.

[12] Kidner D B, Railings P J, Ware J A. Parallel processing for terrain analysis in GIS: visibility as a case study [J]. Geoinformatica, 1997, 1(2): 183 – 207.

[13] Zevenbergen L W. Quantitative analysis of land surface topography [J]. Earth surface processes and landforms, 1987, 12(1): 47 – 56.

[14] Planchon O, Darboux F. A fast, simple and versatile algorithm to fill the depressions of digital elevation models [J]. Catena, 2001, 46(2 – 3): 159 – 176.

[15] Lowenthal D K, Freeh V W, Andrews G R. Using fine-grain threads and run-time decision making in parallel computing [J]. Journal of Parallel And Distributed Computing, 1996, 37(1): 41 – 54.

[16] Antonopoulos C D, Blagojevic F, Chernikov A N, et al. A multigrain delaunay mesh generation method for multicore SMT-based architectures [J]. Journal on Parallel and Distributed Computing, 2009, 69(7): 589 – 600.