基于 Fork/Join 框架的等值面快速生成并行算法

鲍婷婷,焦圣明,殷笑茹,陈景丽,牛霭琛
(江苏省气象信息中心,江苏 南京 210008)

摘 要:针对传统串行等值面提取算法在处理离散点数量多、网格点密度大的数据时生成效率差的问题,提出一种新的基于 Fork/Join 框架下的等值面快速生成并行算法。通过对传统串行算法中的关键步骤进行并行计算可行性分析,提出可以实施并行计算的四个单独步骤:离散点数据网格化处理、等值点计算、等值线追踪与光滑、等值面标记识别。通过将并行计算作用于等值面生成的这四个步骤中,减少了等值面计算的执行时间,加快了等值面的生成速度。实验结果表明,在数据计算量较大时,与传统串行算法相比,并行算法能在 2 秒内快速生成等值面,最大加速比高于 5.0,提高了等值面的生成效率并取得了良好的绘制效果,满足了高实时性的业务需求。

关键词:并行计算;Fork/Join 框架;等值面;提取算法;图形绘制
中图分类号:TP319 **文献标识码**:A **文章编号**:1673-629X(2020)03-0187-07
doi:10.3969/j.issn.1673-629X.2020.03.036

A Parallel Algorithm of Rapid Isosurface Generation Based on Fork/Join Framework

BAO Ting-ting, JIAO Sheng-ming, YIN Xiao-ru, CHEN Jing-li, NIU Ai-chen
(Jiangsu Meteorological Information Center, Nanjing 210008, China)

Abstract: In order to solve the problem of poor generating efficiency of traditional serial isosurface extraction algorithm in processing data with large number of discrete points and large density of grid points, a new parallel algorithm of rapid isosurface generation based on Fork/Join framework is proposed. By analyzing the feasibility of parallel computing for the key steps in the traditional serial algorithm, four separate steps for parallel computing are proposed: discrete point data grid processing, equivalent point calculation, contour tracing and smoothing, and isosurface marking and identification. By applying parallel computation to these four steps, the execution time of isosurface calculation is reduced and the generation speed of isosurface is accelerated. The experiment shows that compared with the traditional serial algorithm, the proposed algorithm can quickly generate the isosurface within 2 seconds with a maximum acceleration ratio higher than 5.0 when the data calculation is large, which improves the isosurface generation efficiency and achieves the great rendering effect, satisfying the business demand of high real-time.

Key words: parallel computing; Fork/Join-framework; isosurface; extraction algorithm; graphics drawing

0 引 言

等值线图是一种重要图形,在地质、气象、水文、设计等领域中得到了广泛应用。在数值分析中,绘制等值线图是应用最多的基础技术手段之一。随着网络带宽的日益完善,传统的等值线图绘制应用已明显不能满足用户需求,探索基于 Web 或云计算下的绘制算法成为目前的研究主流。在软硬件环境及算法相同的情况下,传统应用程序在执行效率上要比基于 Web 的应用高,然而在离散点数量多、网格点密度大的情况下,按传统的串行方法实现等值面生成的这两类应用均存在响应速度慢、效率低的问题,这一问题同时也广泛存在于专业商业软件中,使得业务系统在实时性方面无法得到保证。随着多核处理器设计技术的快速发展,针对 Web 应用研究基于多核平台下的等值面快速生成并行算法显得尤为迫切。

等值面的绘制过程一般分为离散数据网格化、等值点计算、等值线追踪及光滑、等值面提取、地图裁剪等几个步骤。当前,国内外就如何提高等值线图生成

收稿日期:2019-03-07 **修回日期**:2019-07-09 **网络出版时间**:2019-11-07
基金项目:江苏省气象事业发展“十二五”重点工程项目(123030990403);江苏省气象局北极阁基金重点项目(BJG201301);江苏省气象局科技项目(KM201804)
作者简介:鲍婷婷(1986-),女,硕士,工程师,从事气象数据处理与信息系统运行保障研究;焦圣明,高工,正研,研究方向为气象信息技术。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20191107.0908.002.html

效率展开了大量的研究,其可以从两方面着手:一方面把 GPU 等硬件资源应用到普通的数值并行计算中;另一方面通过软件方法提高计算速度,主要是通过并行算法改进程序提升效率[1-2]。有学者利用 GPU 提供大量的并行数据,应用其并行性来进行通用计算,如使用 GPU 进行等值面的提取与绘制,但其绘制过程无法完全在图形硬件上实现[3-4];文献[5]在 GPU 并行架构和 CUDA 可编程性基础上提出了一种基于区间块搜索的等值线并行提取算法,但提取方法并未给出;文献[6]将基于非规则四面体的等值面提取和绘制过程完全转移至 GPU 上执行,从而降低了 CPU 计算负荷,提高了等值面提取和绘制速度;文献[7]提出了一种等值线二值边界追踪的算法,以栅格边界为基础追踪等值线;文献[8]针对等高线绘制中排序耗时问题提出了一种改进的算法,可应用于 2D 与 3D 对象;文献[9]采用分区法和 OPENMP 的并行算法处理形成等值填色场,但文献中实现的等值面只是根据并行计算插值给图片像素赋予颜色值,生成的等值面是图像产品而不是矢量数据,在业务使用上存在一定的局限性;文献[10]是基于 ArcGIS Server 的模型来实现等值面生成,其成本高、算法可扩展性小,在等值面图形复杂度较高的情况下响应性能会受到一定的影响,致使效率不高。

针对以上问题,文中提出了一种基于 Fork/Join 框架的等值面快速生成并行算法。首先描述算法的基本原理、并行任务及划分策略,其次以自动气象站的累积降水量和不同网格下数值预报温度场的等值面提取为例实现该并行算法,并通过与传统串行算法比较,从计算性能、等值面绘制效果验证该算法的效率及优势。

1 Fork/Join 框架

Fork/Join 框架是一个分治(divide and conquer)策略的并行编程框架,实现了任务定义与任务处理的分离,大大简化了并行程序设计的复杂度,开发人员能够方便地利用多核平台的计算能力。尽管还没有做到对开发人员完全透明,但已经极大地简化了编写并行程序的琐碎工作。

工作窃取算法(work-stealing algorithm)是该框架用来提高性能、保证负载均衡的一种有效方法,其本质上是一种工作任务的调度方法。基本思想是利用已完成任务的线程去查看其他线程是否有未处理完的工作,如果有则窃取一部分工作来执行,从而提高资源的利用率,减少程序的处理时间。

该框架主要由 Fork 和 Join 两个操作构成,Fork 操作是对任务的划分,把大任务划分成若干小问题,能方便对简化后的问题进行处理;Join 操作主要是完成对各个部分的运行结果进行合并的任务。其工作原理如图 1 所示。

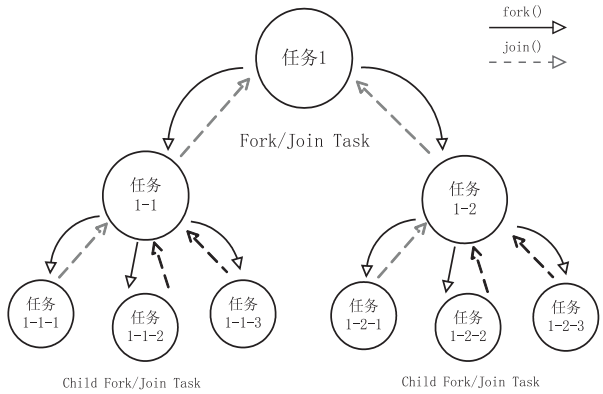


图 1 Fork/Join 框架工作原理

2 等值面并行算法

2.1 串行处理算法

目前,规则矩形网格法和不规则三角网格法是进行等值线图绘制比较常见并且较成熟的算法[11],由于气象业务中大部分数据是已经插值好的网格数据,故多采用规则矩形网格法进行等值线图绘制。基于规则矩形网格法的等值面提取算法描述如下:

算法:规则矩形网格法等值面提取算法。

输入:某种气象要素的离散数据集 U,等值线间隔数组 K;

输出:等值面矢量数据集。

Step1:将离散点气象数据集利用一定的插值方法进行网格化处理;

Step2:计算网格上的等值点,并将等值点连接成线段存贮到线段组队列;

Step3:利用队列中的线段组根据等值间隔追踪出所有等值线,并进行光滑处理;

Step4:通过边界线把非闭合等值线处理成闭合等值线,对等值面进行标记识别;

Step5:按目标区域范围进行等值面裁剪;

Step6:算法结束。

2.2 可并行过程

并行算法是一些可同时执行的诸进程的集合,这些进程互相作用和协调动作从而达到给定问题的求解[12]。并行基本思想就是在一段时间内同时处理两件或两件以上的作业,这两件或两件以上的作业可以是同一性质也可以是不同性质的,如果要并行计算就必须保证过程之间的独立性,即不包括数据依赖、控制依赖、数据交换等关系。从整体上分析程序的并行化工作,综合考虑整个计算任务在所有计算资源上的分配,尽量做到负载平衡,可以取得较大的性能提升[13]。就等值面提取算法而言,其处理流程的各个步骤是有一定依赖关系的,它们不能同时并行处理串行算法中的各个步骤,而是要按一定的先后顺序执行。因此,并行算法只能作用在可以实施并行计算的单独步骤上,

从而减少这些步骤的作业时间,达到提高程序执行效率的目的。串行算法里有四个步骤可以实施并行计算,分别为:

(1)离散点数据网格化处理。离散数据量的多少、网格设定的大小以及不同的插值方法都会影响到算法的处理速度。在传统方法下,网格密度越大,插值精度越高,算法花费在格点值的计算时间就越长,反之网格密度越小,计算速度越快,但输出的等值面失真度就越高。精度和效率都是不可缺少的,为了同时保持高精度和高效率,一个有效的解决办法就是使用并行算法来加快网格点的计算速度。

(2)等值点计算。根据步骤 1 处理好的格点资料计算出有效的等值线间隔的最大值和最小值,确定出等值点判别循坏的起始值,在并行计算等值点时可能会减少当前等值线段组属于哪条等值线的标识时间,它在最差状态与上面不进行预处理时的循环次数相同。

(3)等值线追踪与光滑。从步骤 2 线段组中提取出所有不同间隔的等值线值,分别建立新的队列,将原先的线段组摆放到各自相对应的新队列中去,接着对这些新队列进行并行等值线追踪,最后按文献[14]中的方法对每条等值线进行穿过原点的光滑处理。

(4)等值面标记识别。仅根据等值线的标记值是无法判断由其构成的等值面等级,需要利用格点上的值对等值面进行判别,标记识别的目的是确定使用什么颜色对该等值面进行颜色填充。文中对标记识别采用一种简单实用的办法,就是先对所有等值面进行面积大小排序,排序完成后从等值面边界线任取一点计算出相对应的格点,利用该格点及周边四个格点进行判别,如果该格点在等值面内,就以此格点的值换算出等值面的标记值。通过边界线把非闭合等值线处理成闭合等值线,把所有排好序的等值面按顺序存放到数组队列,然后把这一大数组队列划分成数各小数组队列并行标记识别。

当用户对实时性要求比较高时,若操作响应时间超过 3 秒,则其交互体验效果将会变差。因此,对上述四个步骤进行并行化处理后,计算速度明显提升,有效地保证了客户端用户交互响应的时效性。

2.3　任务划分策略

通常算法中循环处理操作是程序执行中蕴含并行性最丰富的一种结构,上述四个步骤是串行程序的并行化,就是将可以并行化的循环移植到并行环境中进行计算的过程。一个大任务操作在划分成许多小任务时,也不是并行小任务越多处理速度就越快,需要从硬件资源和计算任务性质统筹考虑,并行任务线程池的线程数主要依据硬件资源设定,计算性质对并行阈值(threshold value)取值也十分重要,在任务规模不大的情况下采用串行的解决方法可能会更好。上述步骤 1、2 计算量级相似,其划分小任务的阈值基本一致但不能太小,可以设定在 3 000 ~ 4 000;步骤 3 队列数组不会太大但是一个较耗时的操作,分解的并行小任务数量和数组长度一样;步骤 4 需要几何空间,计算阈值不宜太大,可以设定在 10 ~ 30。这四个并行计算都使用了工作窃取算法。

2.4　并行算法实现

Java 从 JDK1.7 版本开始引入 Fork/Join 框架编程模式,采用这个并行框架最大的优势是能够高效快捷地实现程序跨平台功能。Fork/Join 框架里的 java.util.concurrent 包提供 ForkJoinTask 和 ForkJoinPool 两个类来实现算法的并行,通常情况下不需要直接继承 ForkJoinTask 类,而只需要继承它的子类 RecursiveAction 和 RecursiveTask。RecursiveAction 类用于没有返回结果的任务,RecursiveTask 用于有返回结果的任务。上述步骤 1 ~ 步骤 3 的并行算法都是继承有返回任务的 RecursiveTask 类实现。ForkJoinTask 需要通过 ForkJoinPool 来执行,任务分割出的子任务会添加到当前工作线程所维护的双端队列中,进入队列的头部,当一个工作线程的队列里暂时没有任务时,它会随机从其他工作线程的队列的尾部获取一个任务,原理实现如图 2 所示。

通过对上述等值面提取算法及其并行性的分析,得出算法核心过程的任务可分割性,提取了各个任务中的可并行部分,得到基于 Fork/Join 框架编程模式下实现的并行算法流程示意图(见图 3)。文中基于 Fork/Join 框架编程模式实现的并行算法最核心的伪代码如下:

```
public class TaskName extends RecursiveTask<V> {//V 是返回结果的变量
//变量定义
……
public TaskName(//传入参数){
//类初始化
}
@ Override
protected V compute( ){
//定义返回的变量
V sum;
if(问题规模<阈值){
//使用串行模式解决或选择其他更优算法解决
……
}
else{
//将任务 Task 进行分解,分解成 Task1、Task2…
TaskName Task1 = new TaskName( );
```

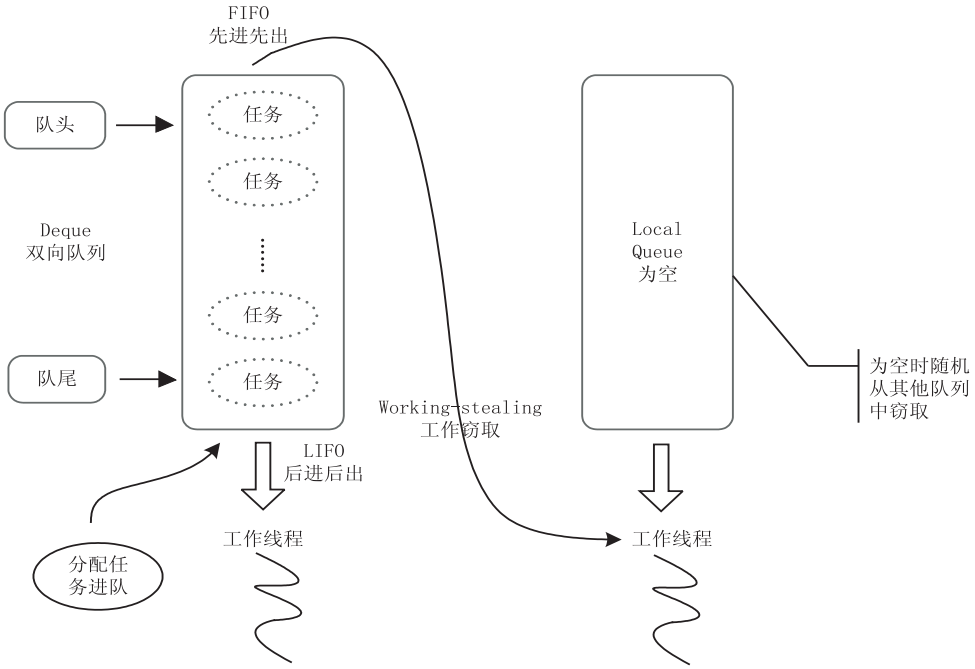


图 2 Work-stealing(工作窃取)实现

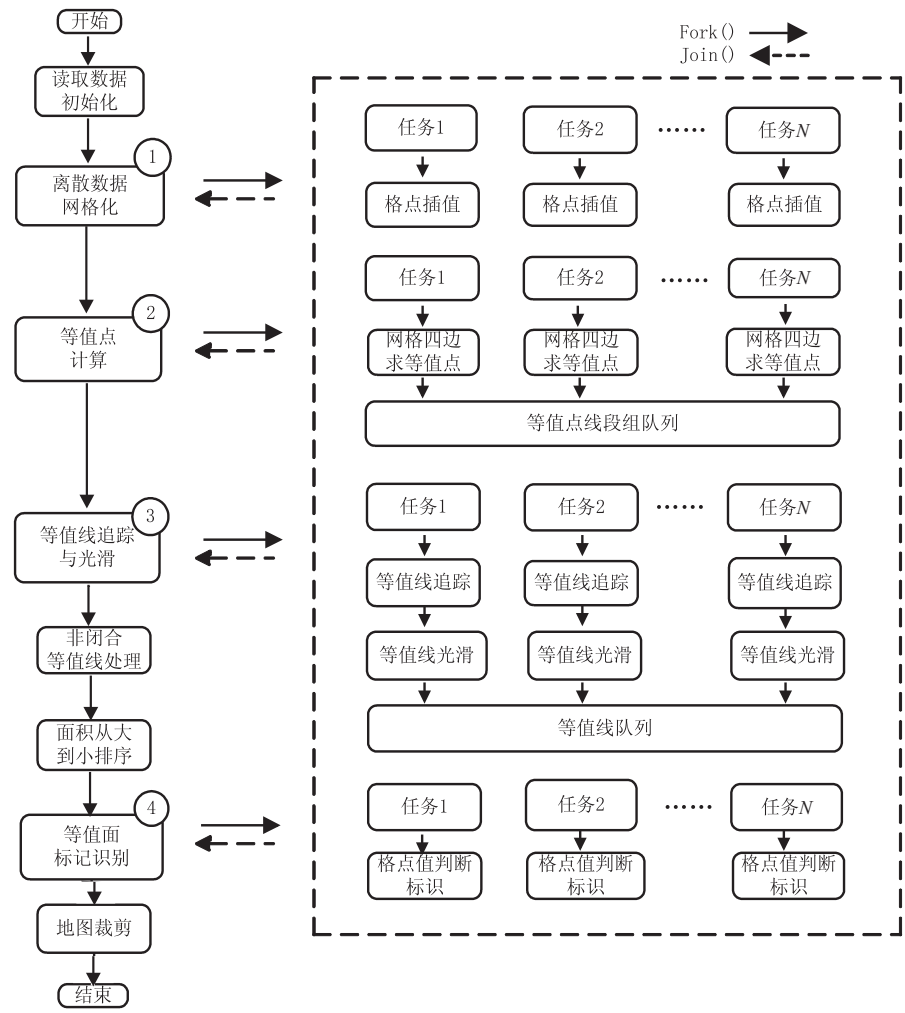


图 3 等值面提取并行算法流程示意图

```
TaskName Task2 = new TaskName( );                    //将任务 Task1、Task2…提交给线程池执行
……                                                  Task1. fork( );
```

```
Task2. fork( );
……
//如果任务有返回结果,收集结果合并
V v1 = Task1. join( );
V v2 = Task2. join( );
sum = v1 + v2;
}
return sum;
}
}
```

等值面提取算法是通过编写 com. jandy. isolines 类包来实现的,其中 IsoLines 是主类,定义了一系列属性和方法供程序调用,最重要的方法是 parallelMakeIsolines( ),它包含了四个并行过程。在 com. jandy. isolines. parallel 子包下,这四个并行过程的类就是利用上述伪代码思想编码完成,只不过各自类的输入任务、计算方法和输出结果不一样而已。这个包单独编译后形成的 jar 文件还可以提供其他 Java 开发人员当作组件使用,提高了代码的复用性。

3　实　验

3.1　算法处理时间

等值面提取算法的计算时间是可以进行估算的。

设算法处理时间为 T,串行算法处理时间为:

$$T = T_i + T_g + T_c + T_s + T_p + T_w + T_m \tag{1}$$

其中,T 为处理总的运行时间,T_i 为读取数据处理时间,T_g 为离散数据网格化处理时间,T_c 为等值点计算时间,T_s 为等值线追踪和光滑时间,T_p 为非闭合等值线处理成闭合等值线和面积排序时间,T_w 为等值面标记识别时间,T_m 为地图裁剪时间。

并行算法处理时间为:

$$T = T_i + \frac{T_g + T_c + T_s}{N} + T_p + \frac{T_w}{N} + T_m \tag{2}$$

其中,N 是处理器核个数,算法通过把各个任务分配到 N 个核处理器上以相同方法同时进行计算,从而减少程序的计算时间,提高执行效率。实验数据分析主要针对可以并行的过程,不能并行的过程执行时间相似且极小,基本可以忽略不计。

根据上述串行算法处理时间(T_1)与并行算法处理时间(T_2)得到的两者所耗时间比(加速比)来评价该并行程序的性能,即,并行执行与串行执行时的计算性能的比值[15]:

$$S = \frac{T_1}{T_2} \tag{3}$$

3.2　实验平台

等值面提取算法的性能测试是在软硬件环境相同的实验平台上进行的,通过串、并行算法实例运行,比较两者的计算时间。采用的硬件平台是一台配备 Intel (R) Xeon (R) E7 - 4820 v2 主频 2.00 GHz 的 8 核 CPU 处理器和 128 G 内存的 X3860 服务器,操作系统为 64 位 Redhat Linux 6.0,Java 虚拟机为 64 位 JDK1.8。为了避免其他软件的干扰,对操作系统及测试必要的软件都进行了全新安装。

3.3　实验结果与分析

实验中分别使用了离散站点的累积降水量和不同网格大小的数值预报温度场这两类资料进行测试和分析,之所以选择这两种类型的资料是因为它们在实际的气象业务应用中极具代表性。为了评估并行计算对等值面处理效率的提高,使用加速比作为衡量标准,执行时间单位为毫秒。

(1)离散数据。

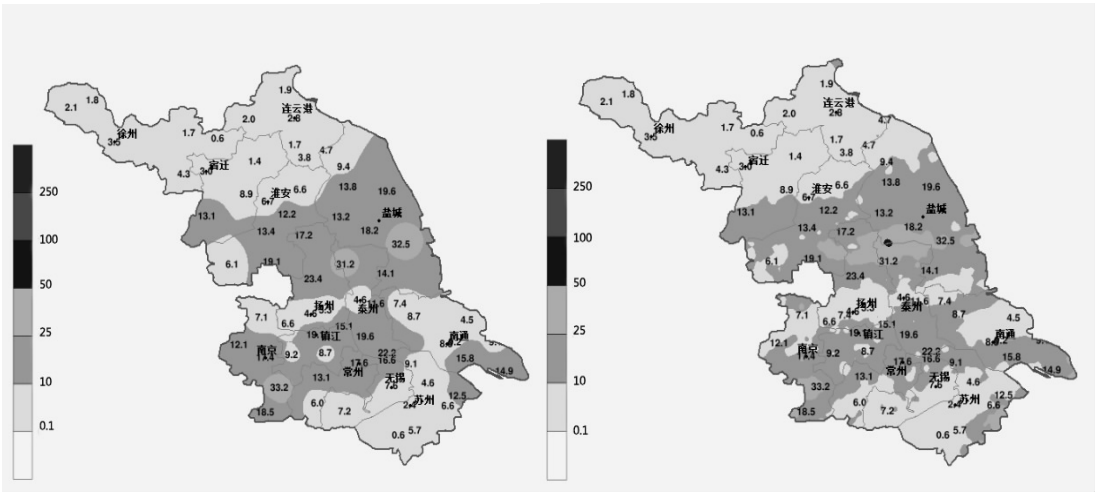
离散数据使用的是江苏省气象局全省地面自动站 2015 年 6 月某日的累积降水量数据,对 71 个基本站和含加密站的 1 441 个所有站的两组数据分别进行了实验。网格大小设定为 201×201,经度 116° ~ 122°,纬度 30° ~ 36°,空间分辨率远小于加密站分布密度能够满足插值精度,生成了 7 级降水分布图,测试不同站点数量对插值速度的影响,见表 1。

表 1　71 个基本站和含加密站的 1 441 个所有站的等值面提取串并行性能对比

时间	71 个基本站			1 441 个站(含加密站)		
	串行/ms	并行/ms	加速比	串行/ms	并行/ms	加速比
T_g	408.1	388.6	1.050 1	7 077.4	1 227.4	5.766 0
T_c	7.7	7.6	1.013 2	7.5	17.0	0.438 2
T_s	2.9	15.5	0.187 1	16.7	15.5	1.076 7
T_w	4.9	15.9	0.308 2	14.2	32.4	0.436 4
处理时间	423.6	427.6	0.990 6	7 115.8	1 292.3	5.506 3

从表 1 可以看出,上述离散数据的等值面提取算法四个并行过程 T_g、T_c、T_s 和 T_w,无论是串行还是并行最耗时间的过程是离散数据网格化处理过程 T_g,在 71 个基本站这一组数据中 T_g 的串、并行执行时间分别占总处理时间的 96.34% 和 90.88%;在含加密站的这一组数据中分别为 99.46% 和 94.98%;随着参与插值

计算站点个数的增加，T_g 过程的计算时间呈增加趋势但加速比获得显著提高，但其他三个过程的加速比就不那么明显甚至会降低。因此对于小网格、等值线稀疏的场景下更适合使用串行算法，并行算法可能会适得其反，增加了处理时间。针对此类业务应用是采取串行还是并行需要具体对待，但离散数据网格化处理过程 T_g 采用并行算法处理是毫无疑问的。图 4(a)是以 71 个基本站数据进行插值计算的累积降水量分布图，图 4(b)是以含加密站的 1 441 个所有站数据进行插值计算的累积降水量分布图。图 4(b)更能反映出较小尺度下的降水差异性，提高了空间精度，与图 4(a)相比在处理时间上要长些，但采用并行算法处理后生成时间控制在 2 秒以下能够满足业务要求。



(a)71个基本站　　(b)1 441个站

图 4　降水分布对比

(2)网格数据。

网格数据是使用美国、欧洲、日本和中国的数值预报中心输出的 2 米高温度场格点预报产品作为实验数据，产品数据都是采用气象信息综合分析处理系统(meteorological information comprehensive analysis and process system，MICAPS)标准的第四类文件格式进行存储，十分方便业务人员进行读写操作。提取 2016 年 3 月 06 日 16 时四家数值预报中心全球模式不同分辨率下的温度场数据，等值线间隔设为 5 ℃，温度范围设定在-60 ~ 40 ℃，对比串行与并行算法的耗时，生成 20 级等值面，测试不同网格大小对加速比的影响，见表 2。

表 2　不同网格大小等值面提取串并行性能对比

预报中心	网格大小	等值线段组个数	等值面个数	串行处理总时间/ms	并行处理总时间/ms	所耗时间比(加速比)
美国	180×91	11 039	428	126.9	76.5	1.658 8
欧洲	360×181	27 448	1 045	310.2	168.6	1.839 9
日本	720×361	66 617	2 188	1 760.1	381.6	4.612 4
中国	1 440×721	153 887	3 962	7 369.8	1 388.4	5.308 1

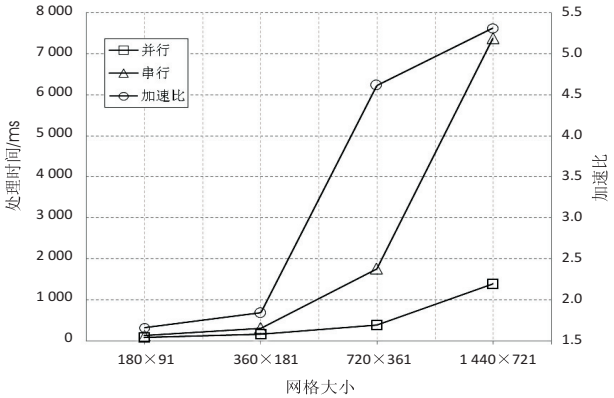


图 5　不同网格下的串、并行处理时间及加速比

根据表 2 结果绘制如图 5 所示的折线图。在网格规模较小的情况下，并行算法的性能突出不明显，但网格规模加大时，并行算法效率提升的效果十分显著。从表 2 中可以看出，在 1 440×721 网格下达到最大加速比 5.308 1，网格大小从 720×361 到 1 440×721，串行处理时间增加了 5.6 秒，而并行处理时间只增加了 1.0 秒，串行算法的处理时间增长幅度远高于并行算法。

图 6 为利用文中并行算法绘制的网格大小为 1 440×721 的温度要素等值面效果图，从读取数据到绘制完成用时不到 2 秒，而用专业气象业务软件 MICAPS 绘制耗时近 15 秒。

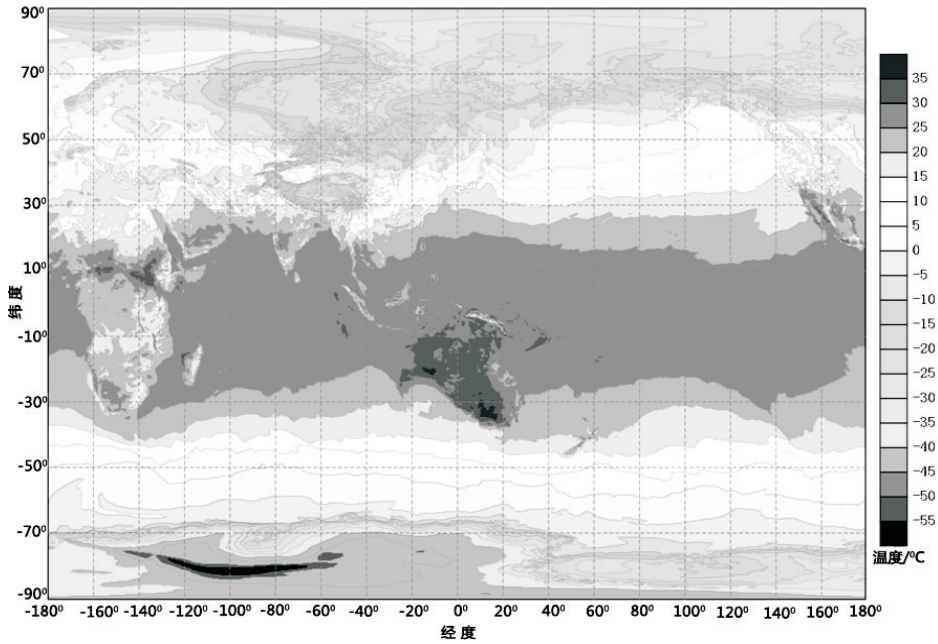


图 6　1 440×721 网格温度要素个例

4　结束语

通过对常用的两类气象资料等值面生成的特点分析,研究了基于 Fork/Join 框架下多核并行计算数据处理的方法。实验结果表明,该方法在实现复杂计算量下的等值面提取时较大地提高了数据处理速度,等值面的生成时间控制在 2 秒内,能够满足气象业务需求,在生成效率上具有优越性,为其他相关业务的等值面分析提供了一种借鉴思路。特别是针对实时性要求极高、基于 Web 应用的等值面分析时,在服务端使用并行计算是一种较好的解决方案。后续工作将利用此算法构建云服务平台,为注册用户开放 API 云接口,便于用户在自己的 Web 应用中接入使用。

参考文献:

[1] 孙桂茹,马　亮,路登平,等. 等值线生成与图形填充算法[J]. 天津大学学报,2000,33(6):816-818.

[2] CHE S,BOYER M,MENG J,et al. A performance study of general-purpose applications on graphics processors using CUDA[J]. Journal of Parallel and Distributed Computing,2008,68(10):1370-1380.

[3] MATSUMURA M,ANJYO K. Accelerated isosurface polygonization for dynamic volume data using programmable graphics hardware[C]//Proceedings of the SPIE-the international society for optical engineering. United States:Society of Photo-optical Instrumentation Engineers,2003:145-152.

[4] KLEIN T,STEGMAIER S,ERTL T. Hardware-accelerated reconstruction of polygonal isosurface representations on unstructured grids[C]//12th Pacific conference on computer graphics and applications. Seoul,South Korea:IEEE,2004:186-195.

[5] 钱　宸,杜震洪,曹润洲,等. 基于 CUDA 并行的全球海洋表面温度场等值线提取算法研究[J]. 浙江大学学报:理学版,2014,41(1):82-89.

[6] 陈　鹏,杨　超,吴玲达. 硬件加速的等值面提取与绘制[J]. 小型微型计算机系统,2008,29(8):1538-1541.

[7] REN Mingwu,YANG Jingyu,SUN Han. Tracing boundary contours in a binary image[J]. Image and Vision Computing,2002,20(2):125-131.

[8] JONES N,KENNARD M,ZUNDEL A. Fast algorithm for generating sorted contour strings[J]. Computers and Geosciences,2000,26(7):831-837.

[9] 王志斌,万玉发,罗　兵,等. 一种等值线填充并行算法[J]. 计算机工程与应用,2012,48(28):61-65.

[10] 周　博,沈夏炯,郑逢斌,等. 等值面快速生成方法的改进[J]. 计算机工程,2010,36(12):291-292.

[11] 汤子东,郑明玺,王思群,等. 一种基于三角网的等值线自动填充算法[J]. 中国图象图形学报,2009,14(12):2577-2581.

[12] 陈国良,孙广中,徐　云,等. 并行算法研究方法学[J]. 计算机学报,2008,31(9):1493-1502.

[13] 吴石磊,安　虹,李小强,等. 组网雷达估测降水系统并行化方案的设计与实现[J]. 计算机科学,2012,39(3):271-275.

[14] 张蔺廉,范其平. 用规则四方矩阵网格资料分析等值线的一种方法[J]. 气象,2006,32(6):75-78.

[15] 王之元,胡庆丰,陈　娟. 能耗并行加速比:高性能计算系统综合性能的有效度量[J]. 计算机工程与科学,2009,31(11):113-116.