基于OpenMP的两个地学基础空间分析算法的并行实现及优化

朱效民 潘景山 孙占全 顾卫东
(山东省计算中心 济南 250014) (国家超算济南中心 济南 250101)
(山东省计算机网络重点实验室 济南 250014)

摘 要 研究了两个基础空间分析算法(大量线段求交、点面叠加)的实现方法,并基于单机多核环境,利用OpenMP实现了并行算法。通过分析并行算法,得到了无法取得线性加速比的原因,即负载不均衡,内存管理采用全局方法,仍然是"串行"的。基于此,通过对数据进行有效的排序并利用OpenMP的动态调度方式进行调度;改进了现有的并发内存分配技术,并将其用于并行算法的内存管理。利用以上方法对并行算法进行了优化,测试表明,优化后的算法具有较为理想的近线性加速比,单机四核环境下,每个核心的计算效率不低于80%。

关键词 空间分析,求交,点面叠加,并发内存分配
中图法分类号 TP301.6 **文献标识码** A

Parallel Implementation and Optimization of Two Basic Geo-Spatial-Analysis Algorithms Based on OpenMP

ZHU Xiao-min PAN Jing-shan SUN Zhan-quan GU Wei-dong
(Shandong Computer Science Center, Jinan 250014, China)
(National Supercomputer Center in Jinan, Jinan 250101, China)
(Shandong Provincial Key Laboratory of Computer Network, Jinan 250014, China)

Abstract This paper introduced research on the methods for two basic geo-spatial-analysis algorithms: getting intersection points for large amounts of segments and point-polygon-overlay, and we implemented the two algorithms on the shared memory multi-core environment based on OpenMP. We analyzed the reason why we don't get linear speedup, and got that it is because of unbalanced load and serial memory management method. Then we sorted the input data and adopted the dynamic scheduling of OpenMP. Also, we adopted and improved the current parallel memory allocating technique to manage the memory for parallel algorithms. Based on the two methods above, we improved the algorithms. The tests show that the improved method can reach nearly linear speedup, and the efficiency of each core in a four-core node is above 80%.

Keywords Spatial analysis, Intersection, Point-polygon-overlay, Parallel memory allocate

GIS,即Geographical Information System(地理信息系统),是一种非常重要的信息系统,是处理地理空间数据的信息系统[1]。近年来,随着GIS的发展,GIS又有了新的含义,即Geographical Information Science(地理信息科学)。GIS已不仅仅是一个技术实现,而是与计算机技术、地理学、测绘学等密切相关的一门交叉学科。GIS从初期为部分专家学者使用,慢慢到为政府部门做辅助决策使用,目前GIS已经走向大众化,任何人都可以利用GIS特别是其空间分析功能来指导自己的日常生活。

空间分析是GIS的核心,是GIS的重点与难点。据专家估计,在经济建设和日常生活活动所涉及的信息中,80%与地理信息密切相关。但是,"人们被数据淹没,人们却饥饿于知识";这样的情形客观上要求大力发展空间分析理论方法,GIS空间分析功能也越来越多。随着空间分析技术的不断发展,GIS也将从一般的空间事务处理向分析型空间决策支持方向迈进[2]。矢量地图叠加分析(也叫叠置分析,即Vector Map Overlay)作为空间分析的一种,在GIS中扮演了一个重要的角色,在一定程度上决定了一个GIS产品的质量。叠加分析是GIS中最常用的提取空间隐含信息的手段之一,是GIS空间分析中一项最常用、最重要的分析方法之一。

近年来,随着Intel和AMD多核处理器市场占有率的上升,多核化的时代已经到来,多核处理器市场占有率已经接近90%。但是,目前一些常用的软件,包括常用的GIS软件的空间分析功能等一般都是串行的,无法发挥出多核CPU的计算能力,因此需要开发能够利用多核资源的程序。其方法不是重新开发适应多核平台的程序,而是对现有的串行程序进行多核并行化改造,在不改变串行程序基础架构和算法的基础上,对其局部并行化部分进行并行化设计[3]。因此,利用OpenMP进行改造是行之有效且能够快速实现的方法。

根据叠加图层的类型不同,叠加分析一般包含点面、线面和面面叠加。面面、线面叠加都要以线段求交为基础,且整个算法过程中80%~90%的时间为求交时间。大量点与多边

到稿日期:2012-06-05 返修日期:2012-09-01 本文受国家自然科学基金青年科学基金项目(61004115),山东省科学院科技发展基金项目(科基合字2011第12号)资助。

朱效民 博士,助理研究员,主要研究方向为空间计算、并行计算,E-mail:zhuxm@keylab.net。

形关系的运算也是一个基础算法，不仅是点面叠加的基础，线面、面面叠加时，对于没有产生交点的要素计算时，通常选取一个代表点，然后利用点面关系的计算，完成对没有产生交点的要素的处理。因此，这两个算法是基础的核心算法，对这两个算法进行并行的实现能够提高整个叠加分析功能模块的效率。

综上，我们以大量线段的求交、大量点与多边形的关系计算这两个算法为研究内容，在单机多核环境下基于 OpenMP 进行实现，并进行分析、优化。这正是本文的工作所在。

1 基础模块

本文的算法实现过程中，依赖于一些基础的计算模块，包括 r-tree 空间索引、并发内存分配技术；还利用 OpenMP 进行并行实现。本节将简要介绍 r-tree 的基本原理、并发内存分配技术以及 OpenMP。

1.1 r-tree

空间索引是指在存储空间数据时依据空间对象的位置和形状或者空间对象之间的某种空间关系、按一定顺序排列的一种数据结构，其中包含空间对象的概要信息如对象的标识、外接矩形及指向空间对象实体的指针。作为一种辅助性的空间数据结构，空间索引介于空间操作算法和空间对象之间，通过筛选，大量与特定空间操作无关的空间对象被排除，从而提高了空间操作的效率。空间索引的作用是为了在空间数据库、磁盘文件、内存结构中快速地定位所选中的空间要素，从而提高空间操作的速度和效率。

r-tree 是最流行的空间索引。它是 B-树在多维空间的扩展，由 Guttman 于 1984 年提出。我们以以上 r-tree 的理论为基础，实现了一个鲁棒高效的 r-tree 索引。它利用空间索引来管理内存中的空间数据，提供数据的插入以及利用 MBR (Minimum Bounding Rectangle)对两个主要功能进行检索查询。索引的插入对象可以是几何体(点线面等)，也可以是几何体的组成部分(即线段)。此外，针对插入对象，建立了两层 ID 标记策略，从而可以方便地标记线段(对应两个 ID，分别为几何体的 ID 以及线段的 index)，对以线段为基本索引对象的计算过程使用较为方便。

1.2 并发内存分配

并行、多线程的应用程序得到越来越广泛的应用，对于这些应用程序，在共享内存环境下，内存分配已经成为制约程序性能的瓶颈。并行优化一个很重要的改造内容就是对内存分配机制进行改造。如果使用普通的非线程分配器，那么多线程程序中的内存分配将成为一个严重的瓶颈，因为每个线程都会在一个全局锁上发生竞争，而在单一全局堆上执行内存分配操作和内存释放操作时都需要使用锁。例如，在文献[4]中，当单机多核下并行实现点面叠加分析算法时，通过 Intel Vtune Performance Analyzer 来分析内存分配与释放等基础操作，在 1、2、3、4 个线程时其基础的内存分配操作的相对值分别为(以单线程时的内存分配为基础)：1：1.61：1.86：2.11。可见，单机多线程时，这些基础的内存元操作都较单线程耗时，因此无法达到线性加速比。共享内存的并行环境对计算密集型的过程具有较好的加速，而频繁申请、释放内存的计算过程则需要利用一些内存分配算法提高并行的效率。

面对这种状况，人们提出了许多针对多线程应用程序的内存分配算法，其中比较有影响力的有 Hoard[5]、Streamflow[6]、PtMalloc[7]、NBMalloc[8]等，这些多核内存分配程序从不同的角度解决了多核内存管理问题，核心思路是为每个核建立一个局部的堆，每个线程可以在私有堆内申请释放内存。具体实现时，有一些不同，如：Hoard 还有一个全局堆，PtMalloc 线程堆与线程不是一一对应的，而是动态变化的。文献[8]中综述并比较了这些不同的内存分配方法。

NBMalloc[8]是一个无锁的内存分配器，其目的是利用动态内存分配技术增强并发应用程序的并发性。其架构是受 Hoard 启发而来的，基本原理也与 Hoard 类似。实现时，以 NBMalloc 为基础进行改造与特定的优化。主要的改进措施包括：两个算法内部每次申请的内存大小往往不是随机的，常见的内存申请过程为空间索引查询中保存空间要素的 index 信息，以及保存计算得到的结果等。因此可以根据空间计算的数据特点，简化其 size_of_class，以提高内存管理的效率。

1.3 OpenMP 介绍

OpenMP[9]由 OpenMP Architecture Review Board 牵头提出，并已被广泛接受，用于共享内存并行系统的多线程程序设计。OpenMP 是作为共享存储标准而问世的，是为在多处理机上编写并行程序而设计的一个应用编程接口，包括一套编译指导语句和一个用来支持它的函数库。OpenMP 使用 Fork-Join 并行执行模型，所有的 OpenMP 程序开始于一个单独的主线程，开始执行后，主线程会一直串行地执行，直到遇到第一个并行域才开始并行执行[10]。

1.4 本文的思路

一般而言，除非面对大规模的实时计算需求或者高性能计算环境，叠加分析功能及其依赖的 GIS 软件等都是运行在单机(即共享内存)环境下的，较少在集群等环境下运行，因此本文的并行算法设计与分析都是根据叠加分析的实际应用与运行环境(即单机多核的共享内存并行计算)来进行的。本文在共享内存的单机多核环境下，利用 OpenMP 对现有的串行算法进行并行化实现，并从数据负载均衡调度、内存分配优化等角度进行优化。

2 串行算法

2.1 大量线段求交

线段求交的基础操作是对两条线段计算交点，而大量线段求交时，线段并不是两两相交的，实际的交点数量远远小于两组线段数量之积，因此大量线段求交时，最耗时的部分不是计算真实存在的交点的时间，而是对实际上没有相交的线段计算交点的过程。因此，大量线段计算交点的关键在于如何去除不必要的运算。

一般而言，大量线段求交有如下几种方法。最普通的线段求交策略即暴力求解法(Brute Force Method)，也就是对一组线段中的每一条，分别与另一组中的每一条进行计算。这是最简单的方法，但当线段数量非常多时，暴力求解法是非常低效的。常用的两种避免不必要的求交运算方案是空间划分(Spatial Partition)的策略和空间排序(Spatial Order)的策略。

空间划分策略即将线段划分到各个区域中去，只有在同一区域内的线段才可能产生交点，一般通过空间索引来实现。空间排序策略以扫描线方法[11]为代表，其核心是：所有相交的线段在一定区域内都是相邻的，相邻才有可能相交。对不

相邻的线段不会计算交点。Andrews 等在文献[12]中综述了以上这些求交点思路，并且认为 Spatial Partition 在 GIS 数据计算中要优于 Spatial Order 策略。

并行性方面，暴力求解法简单易实现，虽然效率不高，但较易实现并行。扫描线算法效率较高，但因是一种扫描的思想，一般从左到右、从下到上扫描，若并发实现，无法有效地对扫描区域进行划分，无法定义扫描的方向。而且算法实现过程中要始终使用一个红黑树结构，需要频繁地在红黑树中插入和删除(频繁申请和释放内存、做内存的拷贝)，该算法并不适合共享内存的并行实现。基于空间索引的方法的预处理阶段(r-tree 的建立与空间要素插入)不易并行，而索引查询与交点计算容易并行实现；而预处理阶段占整个计算过程的时间耗费百分比较小(小于 5%)，因此综合而言，基于 r-tree 的空间划分策略适合并行实现。

综上，从算法的效率以及可并行性两个方面考虑，我们选取空间划分的方法进行求交。具体实现时，采用 r-tree 索引的空间划分策略。其串行算法的计算流程为：

1. 统计两组线段的数量，为线段较多的一组建立索引，将其包含的所有线段插入到索引结构中去。
2. 以线段较少的一组为基础，顺次取得所有线段，并对每条线段执行下述操作：
   2-1 以当前线段的 MBR，查询索引，得到可能与当前线段相交的所有线段。
   2-2 将当前线段与查询得到的所有线段一一进行求交运算，将得到的真实存在的交点保存。

2.2 大量点与大量多边形的关系判断

大量点与大量多边形的关系判断就是对于大量的点与大量多边形，分别得到哪些点位于哪个多边形的内部，其依赖的基础算法是大量点与一个多边形的关系计算。大量点与一个多边形的关系计算采用目前已知的最高效的算法——平均条带划分的方法[4]。该方法首先对多边形的边基于平均条带进行划分，得到每个条带内的边。然后对每个点计算点所在的条带，再采用射线法，利用点与所在的条带内的边进行相交测试，最后根据测试的奇偶性得到点与多边形的关系。

大量多边形与大量点的关系计算流程：

1. 对点建立索引，并将点插入到索引结构中去；
2. 对每个多边形，采取以下操作：
   2-1 以当前多边形的外包(即 MBR)，查询索引，取得所有可能在当前多边形内部的点；
   2-2 调用单个点与多边形关系计算的流程，得到在多边形内部的点。

3 并行算法

3.1 并行算法的一些全局策略

如文献[3]所述，串行应用程序进行并行化改造主要包括流程机制、循环并行化、粒度划分、内存分配和任务调度等方面。

(1) 流程机制：即并行的计算流程，其基本思路是采用串行的计算流程，利用 OpenMP 对最耗时的 for 循环进行并行实现。

(2) 并发度的发现方式：采用 OpenMP 自动发现可用的核数。

(3) 数据划分及任务调度方式：采用 OpenMP 的默认的数据划分策略。为了防止计算不均衡，采用动态调度的方式，即 schedule(dynamic)。基本的粒度即 for 循环的粒度为点面叠加中的任意一个多边形、求交中的一条线段。数据实际划分的粒度，则为采用动态调度的方式 OpenMP 默认的调度粒度。

(4) 内存分配方面，尽量使用声明局部变量的方式。在必须进行动态内存分配时，采用自定义的内存分配与管理策略。

3.2 线线求交并行实现及优化

线段求交的串行算法的并行就是对上述串行计算的步骤 2 进行并行。如前所述，利用 OpenMP 并行实现时，如果有动态内存分配，则可能会影响并行计算的加速比。上述过程中用到动态内存分配的过程有：

(1)r-tree 建立时，需要动态分配内存，将要素插入；但是 r-tree 作为一个整体，实现并行插入较为困难。因此，r-tree 的插入采用串行方法。经过测试，r-tree 的建立耗费的时间较少，占整个求交计算的 5%左右，甚至更少。

(2)r-tree 的查询：r-tree 查询时，每查到一个位于当前 MBR 内的要素，需要将此要素对应的 ID 信息进行保存，这就需要动态申请内存空间。尤其当 MBR 内部的要素个数较多时，需要多次动态申请内存。

(3)计算得到的交点的保存。每次计算得到一个交点，需要动态申请内存，用以保存此交点。

具体的优化措施包括简单优化和动态内存分配的优化两步：

简单优化：任务调度方面，采用 OpenMP 的 dynamic 调度机制。与默认调度方式相比，动态调度增加了一定的调度开销，但这是值得的。因为空间数据具有分布不均匀的特点，而且对于求交算法而言，在某些范围内线段、交点可能稀疏，而在另外一个范围内，则较为密集，所以若采用静态调度，会造成负载不均衡。

内存优化：内存分配方面，首先尽量采用声明局部变量的方式；对于必须采用动态内存分配的计算过程，则采用改进的优化的 NBMalloc 来替换系统默认的内存分配器。

基于以上并行策略以及优化方法，我们实现了线段求交的并行算法，并选取了 3 组数据两两进行求交点运算。3 组数据情况如表 1 所列。

表 1 线圈段求交测试数据信息表

名称	对应 shpfile 大小	要素个数	线段个数
Country	4.08M	2525	226937
Road	422M	924951	17458596
Landuse	100M	164087	5784393

测试所用的机器软硬件配置为：CPU 为 Intel(R) Core (TM)2 Quad CPU Q9400，2.66GHZ(该四核 CPU 为真正的四核心 CPU，而非双核与超线程结合的假四核心)，内存 4GB，操作系统为 Windows 7 Professional(SP1，32bit)，硬盘为西数 WDC WD5000AAKS-75V0A0 (500 GB，7200 转/分，SATA-II)，所用的开发、编译环境为 Microsoft Visual Studio 2010。

利用以上数据划分及调度策略、内存分配策略，基于 OpenMP 的实现进行优化，串行、简单并行以及优化并行的测试数据如表 2 所列。测试的两组数据都是采用动态调度之后

的测试结果。通过表 2 可得，使用自定义的内存分配后，效率有所提升，在四核情形下，采用自定义的内存分配策略后，效率提升约为 5%～8%，而在 2 个、3 个核心时，效率提升并不明显。这是因为整个算法是计算密集型，申请释放等内存操作占整个计算过程的比例较低。综合而言，基于 OpenMP 进行并行实现并优化后，达到了近线性加速比，有些甚至超过了线性的加速比（通过查阅相关资料可得，其原因是多核计算时，分别利用了每个核心的缓存，相当于增加了缓存的大小，因此提高了获取数据的速度，进而提高了计算速度）。

表 2　线段求交并行测试

数据	串行计算的时间(s)	核心数对应的计算时间(s)			方法	交点个数
		2	3	4		
Country Road	89.2	46.3	31.2	23.6	默认内存分配	155743
		44.6	29.8	21.8	自定义内存分配	
		3.8%	4.6%	8.3%	效率提升	
		2/1	2.99/1	4.1/1.025	加速比/效率	
Country Landuse	185	95.5	63.0	52.1	默认内存分配	276958
		92.8	60.8	48.0	自定义内存分配	
		2.9%	3.6%	8.5%	效率提升	
		1.99/0.997	3.04/1.01	3.85/0.96	加速比/效率	
Landuse Road	7143	3623	2509	1891	默认内存分配	1346251
		3530	2376	1795	自定义内存分配	
		2.6%	5.6%	5.3%	效率提升	
		2.02/1.01	3.0/1.0	3.97/0.99	加速比/效率	

3.3　点面叠加并行实现及优化

点面叠加的并行实现就是对串行算法的步骤 2 进行并行实现。我们选择以多边形为基础进行并行，而不是在大量点与一个多边形内部进行并行。其原因是，如果以大量点与多边形关系计算内部并行，则只能基于点并行计算。而多边形的预处理也需要耗费一定的时间，且其并行难以实现。因此，从更高层次实现并行，实现简单，容易控制，且只要使得负载接近均衡，就会具有较高的效率。具体的负载均衡策略及优化方法为：

对多边形按照其外包大小进行排序，首先计算外包大的，其次计算外包小的。一般而言，计算耗时与点的个数、多边形的边数都是线性关系。而点的个数一般与 MBR 的大小相关。因此，为了使得负载均衡，查询时，采用按照外包大小排序的方法，外包 MBR 较小的放在后面，这样到最后即使负载不均衡，查询的点也较少，负载不均衡度非常小。虽然计算与多边形的边数也有关，但是实现时，进行了简化，仅仅按照外包的大小进行排序，而不考虑多边形的边数。

改为按照外包大小排序后，并采用 dynamic 调度，使得大外包对应的计算首先完成，最后的计算基于小外包的多边形，容易控制负载均衡。

内存管理采用自定义的内存分配与管理策略。其用到的动态内存分配的步骤主要包括：

(1)索引查询时的要素插入(可能包含的要素较多)；

(2)点与多边形关系计算时，对多边形的预处理。

测试时，我们选取了一个点数据、两个面数据。其中点数据信息如表 3 所列。面数据即是上述的 country 与 landuse 数据。测试的机器配置与线段求交时的环境相同。

表 3　点数据信息表

名称	对应 shpfile 大小	要素个数
居民点(Respt)	17.7M	649592

我们分别用点数据与两个面数据进行叠加，计算得到每个面内包含哪些点，即计算得到点位于哪个面内。串行以及内存分配优化前后对应的测试数据如表 4 所列。其中，效率提升是指采用动态内存分配后采用默认内存分配的效率提升，加速比和效率是指最优方法的并行加速比和效率。

表 4　并行点面关系算法测试表

数据	串行计算的时间(ms)	核心数对应的计算时间(ms)			方法
		2	3	4	
Respt/Country	341	192	140	114	默认内存分配
		179	128	108	自定义内存分配
		7%	9.4%	5.6%	效率提升
		1.9/0.95	2.66/0.89	3.16/0.79	加速比/效率
Respt/Landuse	3602	2425	1669	1305	默认内存分配
		1928	1353	1086	自定义内存分配
		25.8%	22.9%	20.1%	效率提升
		1.86/0.93	2.66/0.89	3.32/0.83	加速比/效率

两组测试数据对应的加速比、效率值示意图如图 1 所示。

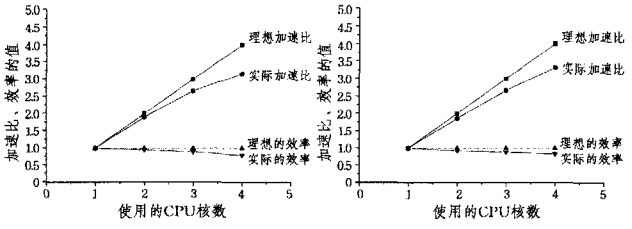

图 1　点面关系计算并行加速比及效率

测试可得，引入动态内存分配策略后，效率有明显的提升，特别是第二组测试数据。这是因为，查询 r-tree 时，每次包含的要素较多，也就是需要频繁地动态申请与释放内存，所以采用动态内存分配后，各个线程自己管理内存，只会造成极小的冲突，也就取得了理想的加速比。此外，整个计算过程中，射线法的相交测试的效率较高，耗时百分比较低，也就是并发内存分配耗时的比例相对较高，因此效率提升的效果较为明显。而并发内存分配所占比例较高，使得该程序不是计算密集型，因此最终的最优效果的各 CPU 核心的效率在 0.8，较线段求交的效率低。

综合以上两组测试数据可得，当内存分配占整个流程百分比较高时，利用并发内存分配技术能有效提高算法的效率；但是，一般仍然无法达到线性加速比，多核计算时的效率在 80%左右。对于计算密集型的算法，利用并发内存分配能够提高算法的效率，但是效果不是很明显，因为内存分配占整个计算流程的百分比较低；不过，又因为整个过程为计算密集型，通过利用并发内存分配进行优化后，能够接近或达到线性加速比。

结束语　通过以上串行程序的并行实现可得，利用 OpenMP 对现有的串行程序进行并行实现是较为理想的选择，无需对算法进行重新设计，只需要一些简单的编译制导语句即可实现，方法较为简单。

(下转第 39 页)

在后续工作中，我们将实现社会活动组织服务系统，结合激励机制对丢包问题作进一步研究，并在实际生活中进行实验。

参 考 文 献

[1] 熊永平，孙利民，牛建伟，等. 机会网络[J]. 软件学报，2009，20(1)：124-137

[2] Pan H，Chaintreau A，Scott J，et al. Pocket switched networks and human mobility in conference environments[C]// Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking. 2005：244-251

[3] Beale R. Supporting Social Interaction with SmartPhones [J]. IEEE Pervasive Computing，2005，4(2)：35-41

[4] Lenders V，Karlsson G，May M. Wireless Ad hoc podcasting[C]// Sensor，Mesh and Ad Hoc Communications and Networks. 2007：273- 283

[5] Yoneki E，Pan Hui，Chan Shu-yan，et al. A Socio-Aware Overlay for Publish/Subscribe Communication in Delay Tolerant Networks [C]//MSWiM'07. 2007：225-234

[6] Mokhtar S B，Mashhadi A J，Capra L，et al. Aself-organising directory and matching service for opportunistic social networking [C]//Proceedings of the 3rd Workshop on Social Network Systems. 2010

[7] Vahdat A，Becker D. Epidemic routing for partially connected ad hoc networks[R]. CS-2000-06. Duke University，2000

[8] 郭斌，於志文，张大庆，等. 机会物联——兼谈物联网的社会性[J]. 中国计算机学会通讯，2011，7(12)：52-56

[9] Cui He-ming，Suman S，Henning S. ONEChat：Enabling Group Chat and Messaging in Opportunistic Networks[C]// Eleventh Workshop on Mobile Computing Systems and Applications (HotMobile). 2010

[10] Pan Hui，Crowcroft J，Yoneki E. BUBBLE Rap：Social-based Forwarding in Delay Tolerant Networks[C]// Proceedings of the 9th ACM international symposium on mobile ad hoc networking and computing. 2008：241-250

[11] Boldrini C，Conti M，Passarella A. Exploiting users' social relations to forward data in opportunistic networks：The HiBOp solution[J]. Pervasive and Mobile Computing，2008，4(5)：633-657

[12] Boldrini C，Conti M，Passarella A. ContentPlace：Social-aware Data Dissemination in Opportunistic Networks[C]// MSWiM'08. 2008：203-210

[13] Urpi A，Bonuccelli M，Giordano S. Modeling cooperation in mobile ad hoc networks：A formal description of selfishness[C]// Modeling and Optimization in Mobile，Ad Hoc，and Wireless Networks (WiOPT'03). 2003

[14] Panagakis A，Vaios A，Stavrakakis I. On the effects of cooperation in DTNs[C]//Proc. IEEE Comsware. 2007：1-6

[15] 陆音，石进，谢立. 基于重复博弈的无线自组网络协作增强模[J]. 软件学报，2008，19(3)：755-768

[16] Wellman B，Wortley S. Diffferent Strokes from Different Folks：Community Ties and Social Support[J]. American Journal of Sociology，1990，96(3)：558-588

[17] Zhang Da-qing，Wang Zhu，Guo Bin，et al. A Dynamic Community Creation Mechanism in Opportunistic Mobile Social Networks [C] // IEEE International Conference on Social Computing. 2011：509-514

[18] Eagle N，Pentland A，Lazer D. Inferring Social Network Structure using MobilePhone Data[J]. Proceedings of the National Academy of Sciences (PNAS)，2009，106(36)：15274-15278

(上接第 11 页)

此外，串行算法并行实现后，还需要根据数据的特点，利用不同的 OpenMP 自带的数据调度方式。对于一些频繁申请释放内存的程序，往往无法取得线性加速比，这就需要自定义内存管理方法来接管操作系统的内存管理策略。利用现有的技术，并根据算法的特点进行改造与定制后，并行算法取得了理想的加速比。这些都为目前一些桌面版的软件的并行实现及优化提供了一种行之有效的思路。Herb Sutter 所说的免费的午餐已经结束[13]，我们利用 OpenMP 对现有的算法进行改造，以较低的代价充分利用了计算资源，虽然得到的不是完全免费的午餐，但是付出的代价也极为低廉。这也为单机多核下的程序的并行实现及优化提供了一个指导思路与案例，即对核心、基础算法进行并发实现，能够以极低的代价提高整个系统的效率。

参 考 文 献

[1] 黎夏，刘凯. GIS 空间分析-原理与方法[M]. 北京：科学出版社，2007

[2] 黄杏元，马劲松，汤勤. 地理信息系统概论[M]. 北京：高等教育出版社，2001

[3] 曹婷婷. 基于多核处理器串行程序并行化改造和性能优化[D]. 成都：西南交通大学，2009

[4] 朱效民. 矢量地图叠加分析算法研究[D]. 北京：中国科学院研究生院

[5] Berger E D，McKinley K S，Blumofe R D，et al. Hoard：A scalable memory allocator for multithreaded applications[C] // The Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX). 2000

[6] Schneider S，Antonopoulos C D，Nikolopoulos D S. Scalable locality-conscious multithreaded memory allocation[C]// Petrank E，Moss J E B，eds. Proceedings of the 5th international symposium on Memory Management. ACM，2006

[7] Michael M M. Scalable lock-free dynamic memory allocation[J]. Conference on Programming Language Design and Implementation，2004，39(6)

[8] Gidenstam A，Papatriantafilou M，Tsigas P. NBMALLOC Allocating Memory in a Lock-Free Manner[J]. Algorithmica，2010，58：304-338

[9] OpenMP[OL]. http://www.openmp.org/

[10] 基于混合包围体的 OpenMP 并行化碰撞检测算法[J]. Journal of Software，Supplement，2008，19：190-201

[11] Bentley J L，Ottmann T A. Algorithms for reporting and counting geometric intersections[J]. IEEE Transactions on Computers，1979，28(9)：643-647

[12] Andrews D S，Snoeyink J，Boritz J，et al. Further Comparisons of Algorithms for Geometric Intersection Problems[C]// Proceedings of the 6th International Symposium on Spatial Data Handling. 1994：709-724

[13] Sutter H. The Free Lunch Is Over[OL]. http://www.gotw.ca/publications/concurrency-ddj.htm