基于栅格分层的逐栅格汇流算法并行化研究*

刘军志[1,2],朱阿兴[1,3],刘永波[4],秦承志[1],陈腊娇[1,2],吴 辉[1,2],杨 琳[1]
(1. 中国科学院 资源与环境信息系统国家重点实验室,北京 100101;
2. 中国科学院大学,北京 100049;
3. Department of Geography, University of Wisconsin-Madison, Madison, WI 53706, USA;
4. Department of Geography, University of Guelph, 50 Stone Road East, Guelph, Ontario N1G 2W1, Canada)

摘 要:分布式水文模型中的逐栅格汇流算法计算量大,需要借助并行计算以满足大流域长历时模拟的要求。针对目前鲜有对基于隐式有限差分的逐栅格汇流算法进行并行计算研究的情况,基于栅格分层的思想提出一种适用于共享内存并行计算环境的逐栅格汇流并行算法。该算法首先根据流向进行栅格分层,使同一层中栅格的计算相互独立,然后将同一层中栅格的计算任务分配到多个计算单元并行计算。采用C++编程语言与OpenMP并行编程库实现了该算法,并选择河北省清水河流域为实验区,在不同数据规模(30m、90m、270m分辨率)、不同核数(2~20个)以及不同栅格分层方法的情况下对算法性能进行了测试。实验结果表明本文提出的并行算法具有较好的加速比和并行效率,且并行效率随数据规模的增大而增大。栅格分层方法对算法并行性能有明显影响,从上到下的分层方法比从下到上的方法具有更高的并行效率。

关键词:逐栅格汇流;隐式有限差分;栅格分层;并行计算;OpenMP
中图分类号: P333.9 **文献标志码**: A **文章编号**:1001-2486(2013)01-0123-07

Parallelization of a grid-to-grid routing algorithm based on grids layering

LIU Junzhi[1,2], *ZHU AXing*[1,3], *LIU Yongbo*[4], *QIN Chengzhi*[1], *CHEN Lajiao*[1,2], *WU Hui*[1,2], *YANG Lin*[1]
(1. State Key Lab of Resources and Environmental Information System,
Institute of Geographic Sciences and Natural Resources Research, CAS, Beijing 100101, China;
2. University of Chinese Academy of Sciences, CAS, Beijing 100049, China;
3. Department of Geography, University of Wisconsin-Madison, Madison WI 53706, USA;
4. Department of Geography, University of Guelph, Guelph, Ontario N1G 2W1, Canada)

Abstract: Grid-to-grid routing algorithms for distributed hydrological modeling require large amount of computations which cannot be provided by sequential computation techniques. Parallel programming technology is necessary for large-scale and long-period simulations using grid-to-grid routing algorithms. There is currently little research on the parallelization of implicit finite difference based routing algorithms. A parallel implicit finite difference based grid-to-grid routing algorithm was presented, based on grids layering. In this algorithm, grids in the watershed were divided into different layers according to flow direction. The calculations of grids in a downstream layer cannot be performed until its upstream layers' calculations were completed. The calculations of grids in the same layer are independent of each other, thus can be performed in parallel. So the parallelization strategy is to assign the calculation tasks of grids in the same layer to different CPU-cores to perform parallel computing. The algorithm was implemented by using the C++ programing language and the Open Multi-processing (OpenMP) Application Programming Interface (API) and was tested in the Qingshuihe watershed of Hebei Province under different amount of input data. The result shows that this parallel algorithm had good speedup and parallel efficiency. In addition, the case study showed that the parallel efficiencies were higher for simulations with large datasets than with small datasets and the up-down layering method had better performances than the down-up layering method.

Key words: grid-to-grid routing; implicit finite difference; grids layering; parallel computing; OpenMP

逐栅格汇流的分布式水文模型是采用栅格作为空间离散单元、沿流向逐栅格进行汇流演算的分布式水文模型,如DHSVM、TOPKAPI、LISFLOOD等[1-3]。这种模型能够反映空间单元之间的相互作用,从而可以对水流运动进行较为准确的空间描述。由于逐栅格汇流分布式水文模型所需计算

* **收稿日期**:2012-07-09
基金项目:国家863计划资助项目(2011AA120305);国家自然科学基金资助项目(41023010);中国科学院知识创新工程重要方向项目(KZCX2-YW-442)
作者简介:刘军志(1984—),男,山东海阳人,博士研究生,E-mail: liujz@lreis.ac.cn;
朱阿兴(通信作者),男,研究员,博士,博士生导师,E-mail: axing@lreis.ac.cn

量大,且计算量随栅格数量和模拟历时的增加而急剧增加,在传统串行计算条件下,由于计算能力的限制,该类模型多用于小流域短历时模拟[4]。要将其应用拓展到大流域长历时模拟则需要借助并行计算提供更强大的计算能力[5-6]。

逐栅格汇流分布式水文模型并行化的关键是逐栅格汇流并行算法的研发和实现。逐栅格汇流算法可采用有限差分、有限元、有限体积等方法,其中有限差分由于具有原理简单、效果好等优点,是目前最常用的算法[7]。有限差分包括显式有限差分和隐式有限差分两种。

显式有限差分实现简单,不需要进行方程组求解,但计算稳定性受 Courant 条件限制[8],对时间步长要求非常严格,因而不适合进行长历时模拟。显式有限差分的并行化方面已有学者进行了相关研究,由于该方法在进行某一时刻的汇流计算时只需要其前一时刻的变量信息,当前时刻每个栅格的计算相互独立,因此多采用数据规则分块的方式并行计算。如 Neal 等[9]利用 OpenMP (Open Multi-Processing)并行编程库,采用按行分块的方式实现了 LISFLOOD-FP 模型的并行化;Yu 等[10]采用规则分块方式实现了 FloodMap 模型的并行化,并对比了不同分块方式对并行性能的影响。

隐式有限差分需要求解偏微分方程组,实现较显式差分复杂,但计算稳定性好,时间步长灵活[11],适合较长历时的模拟。隐式有限差分在计算汇流时,各栅格通过流向存在依赖关系,因此对于隐式有限差分的并行化不适合采用显式有限差分并行化中所通常采用的数据规则分块方式,而需要根据其流向依赖关系确定合适的并行策略。目前鲜有对基于隐式有限差分的逐栅格汇流算法的并行化研究。

本文针对利用隐式有限差分进行逐栅格汇流计算的并行化问题,基于栅格分层的思想提出了一种适用于共享内存并行计算环境的逐栅格汇流并行算法,基于 OpenMP 编程实现,并选择实验区在不同数据规模情况下对算法的性能进行了验证。

1 基于栅格分层的逐栅格汇流算法并行化

1.1 逐栅格汇流算法原理

逐栅格汇流算法将每个栅格作为一个坡面对待[12]。假设栅格长和宽相等,且分辨率为 a,则当流向与栅格行列方向平行时,栅格代表的坡面长宽均为 a;当流向为栅格对角线方向时,栅格代表的坡面长为$\sqrt{2}a$,宽为$\sqrt{2}a/2$。图 1 为两种情况下的坡面示意图。

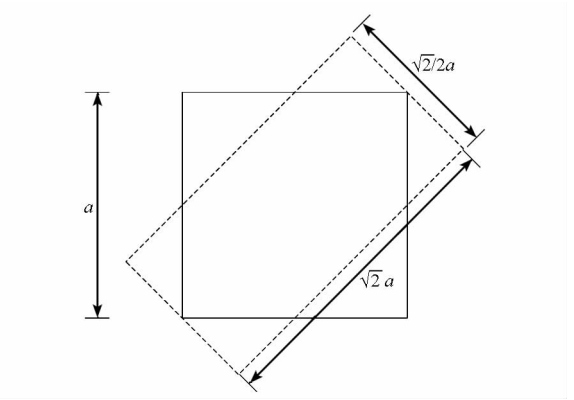


图 1 两种不同流向类型的栅格所代表坡面示意图

Fig. 1 Grid slope with two types of flow direction

对于单个坡面,本文采用一维运动波方程组结合曼宁公式进行逐栅格汇流演算。一维运动波方程组是圣维南方程组的简化,被广泛应用于当前的主流分布式水文模型中[2-3],公式如下:

$$\begin{cases}\dfrac{\partial q}{\partial x}+\dfrac{\partial h}{\partial t}=i-f\\ S_0=S_f\end{cases} \tag{1}$$

式中,q 为单宽流量(m^2/s),h 为水深(m),x 为坡长(m),t 为时间(s),i 为雨强(m/s),f 为入渗率(m/s),S_0 为坡度,S_f 为摩擦阻降。

曼宁公式为

$$q=\frac{1}{n}\sqrt{S_f}h^{5/3}=\frac{1}{n}\sqrt{S_0}h^{5/3} \tag{2}$$

式中,n 为曼宁系数,其他变量含义与上文相同。

汇流算法的数值解法采用得到普遍认可的四点隐式有限差分[13]。采用该方法,对于一个坡面,只需知道其上游的入流边界条件就可以采用牛顿迭代法求解[14]。对于整个流域,则需要从最上游没有入流的栅格开始按流向关系从上游到下游依次进行单坡面汇流计算,上游栅格的出流作为下游栅格的入流(上边界条件)。

1.2 逐栅格汇流算法的并行化

(1)并行计算环境的选择

目前的并行编程模型主要包括两类:共享内存方式和消息传递方式[15]。共享内存方式的并行需要共享内存硬件平台的支持,它通过所有处理器共享内存进行通信,编程模型简单且通信开销低,适用于细粒度的并行;消息传递方式的并行一般用于分布式内存的并行计算环境,它通过消息传递的方式来进行通信,编程模型较复杂且通信开销大,适用于粗粒度的并行。本文研究的逐栅格汇流并行算法在栅格层次上进行并行计算,并行粒度较小,因此选择共享内存的并行计算环

境和编程模型。

(2)并行化策略

栅格之间的汇流计算顺序受上下游关系影响,下游栅格只有当其上游栅格计算完成后才能开始计算。在进行并行计算时,不存在上下游关系的栅格可以并行计算,而存在上下游关系的栅格只能串行计算。

本文采用栅格分层的方式寻找能够并行计算的栅格集合,即根据流向关系将栅格分为有序的多个层,并使同一层中的栅格不存在上下游关系。这样同层中栅格的计算是相互独立的,因此可以分配到多个计算单元并行计算。

栅格分层方法可分为从上到下和从下到上两种。从上到下的分层方法[12]把没有入流的最上游栅格作为第一层,第一层栅格的下游栅格为第二层,其他栅格的层号为其相邻上游栅格的最大层号加 1,以此类推,直到出口栅格(如图 2(b))。从下到上的分层方法[16]把出口栅格作为第一层,第一层的直接上游栅格作为第二层,其他栅格的层号为其相邻下游栅格的层号加 1。为保持两种分层结果从上游到下游具有相同的编号,本文将从下到上方式分层结果的层号进行倒置(例如将 7 -6 -5 -4 -3 -2 -1 倒置为 1 -2 -3 -4 -5 -6 -7),这样最终得到的从下到上方式的分层结果如图 2(c)所示。

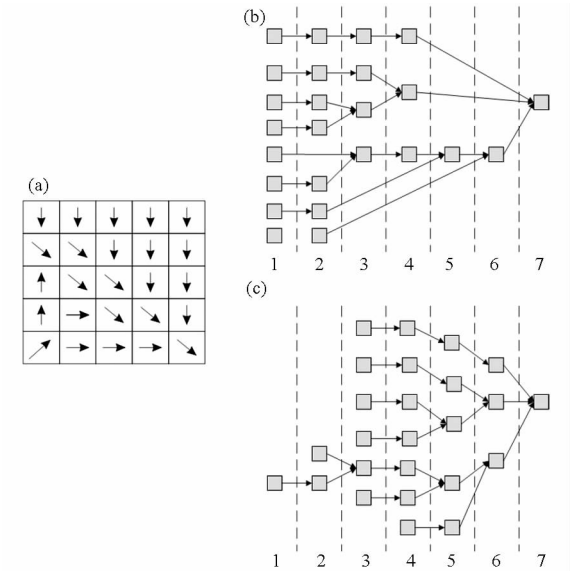


图 2　(a)流向网络　　(b)从上到下的分层结果

(c)从下到上的分层结果

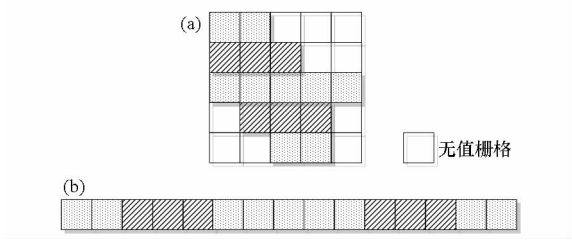
Fig. 2　(a) Flow path network

(b) The set of grid layers using the up-down method

(c) The set of grid layers using the down-up method

(3)数据结构

对于坡度、流向等栅格数据,传统方法一般在内存中采用二维矩阵对其进行组织。但由于流域形状不规则,在流域边界外有很多无值区域,采用二维矩阵存储既浪费了内存空间,又需要在每个时间步长判断当前栅格是否为空值,浪费了计算时间。更为重要的是,无值栅格由于其空间分布不规则,容易在并行计算导致负载不平衡。因此,本文将表示栅格数据的二维矩阵按行展开为一维数组,并剔除无值栅格(如图 3),栅格的位置信息(行列号)和流向关系信息通过全局的查找表存储,从而避免了上述问题。



(a)二维数组方式　　(b)本文采用的一维数组方式

(a) Two-dimensional array and

(b) One-dimensional array adopted in this paper

图 3　栅格数据在内存中的存储方式:

Fig. 3　Storage of raster data in memory

对于栅格分层信息,本文采用变长二维数组存储(如图 4),数组中每一行对应一个栅格层,每行的第一个元素存储该层的栅格数目(图 4 中的 C_i),之后的元素存储该层所包含栅格在一维数组中的索引号(图 4 中的 i_{ij})。在确定了栅格分层方法的情况下,多次运行算法用到的栅格分层方案相同,所以对于一个流域栅格分层方案只需计算一次。本文将栅格分层方案作为算法的输入数据,以避免算法运行时重复进行分层计算。

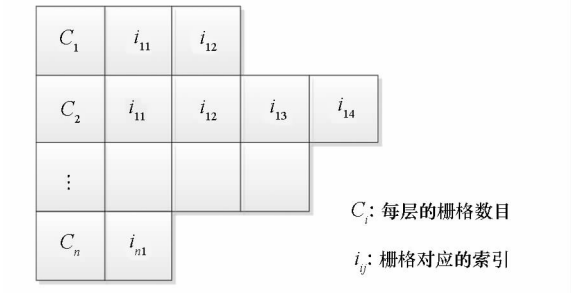


图 4　内存中栅格分层信息的存储方式

Fig. 4　Storage of grid layer information in memory

(4)基于 OpenMP 的并行算法

OpenMP 是支持共享内存程序设计的工业标准,支持 C/C++ 和 Fortran 编程语言。它采用 fork-join 的并行模式,一个 OpenMP 程序从单个主线程开始执行,当主线程遇到并行区域时,主线程创建额外的线程组成一个线程组并行执行。在并

行区域的末尾,线程组中的线程将等待,直到线程组中的全部线程到达并行区域末尾,然后这些线程会合(Join)到一起,继续执行主线程,直到遇到下一个并行区域或程序结束[17]。采用这种方式编程者可以保持原来串行程序的结构不变,减小了并行编程的难度。

OpenMP 通过编译指令对并行计算进行描述,本算法利用 OpenMP 的 parallel for 语句将分层后同一层内的栅格分配到多个线程进行并行计算。并行计算所用线程数利用 omp_set_num_threads 函数进行设置。

2 实验与结果分析

2.1 实验区

选择河北省清水河流域作为实验区验证并行算法的有效性。该流域位于东经 114°46′ ~ 115°30′,北纬 40°47′ ~41°17′,是海河流域上游支流。流域面积 2300km^2,属于半干旱气候区,年均降水量 480mm,降水时间分布不均匀,其中 6 ~9 月份降水占全年总降水量的 80% 以上。

2.2 并行计算实验

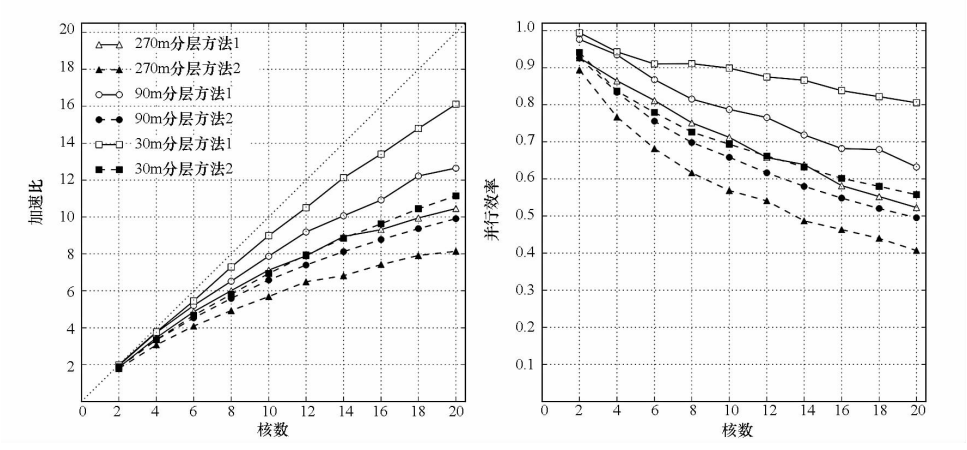
收集整理了实验区 30m、90m 和 270m 三种不同分辨率的数据集,其中 90m 和 270m 分辨率的数据集为 30m,数据集通过重采样得到。30m、90m 和 270m 三种数据集所包含的栅格数分别为 2 381 602, 268 860 和 29 850,栅格分层数分别为 2 875 971 和 308(不同分层方法的层数相同)。本文利用三套数据集在不同分层方法下对并行算法进行了测试。实验中的降水输入为重现期为 100 年的设计暴雨,曼宁系数等输入参数均采用默认值[18]。

算法运行的硬件环境为配有 4 路 Intel Xeon E7450 六核处理器(共 24 核)和 32G 内存的服务器;软件环境为 Windows Server 2008 操作系统、VC++ 2010 编译器和 OpenMP 2.0 并行编程库。VC++ 的编译器优化选择最大化速度(/O2)选项。由于硬件平台的计算核心数为 24 个,本文分别在线程数为 2 ~20 的情况下运行所提出的并行算法,测试不同核数情况下的并行时间并与串行算法进行对比。

算法评价指标主要选取加速比和并行效率。加速比指串行程序执行时间与并行程序执行时间的比值,并行效率指加速比与参与计算核数的比值。

2.3 结果及分析

在不同栅格分层方法、不同数据集情况下的算法测试结果如图 5 所示。在各实验中,加速比均随核数的增加而增加。当参与计算的核数较少时,各实验均具有较高的并行效率,例如当核数为 2 时,并行效率在 0.89 ~1.0;当核数为 4 时,并行效率在 0.77 ~0.94。这表明该并行算法能够充分利用当前主流桌面多核计算环境的计算能力。随着核数的增加,各个实验的并行效率有所减低,这是由于调度开销随着核数的增大而增大。



(a) (b)

图 5 不同实验条件下加速比和并行效率随核数变化情况

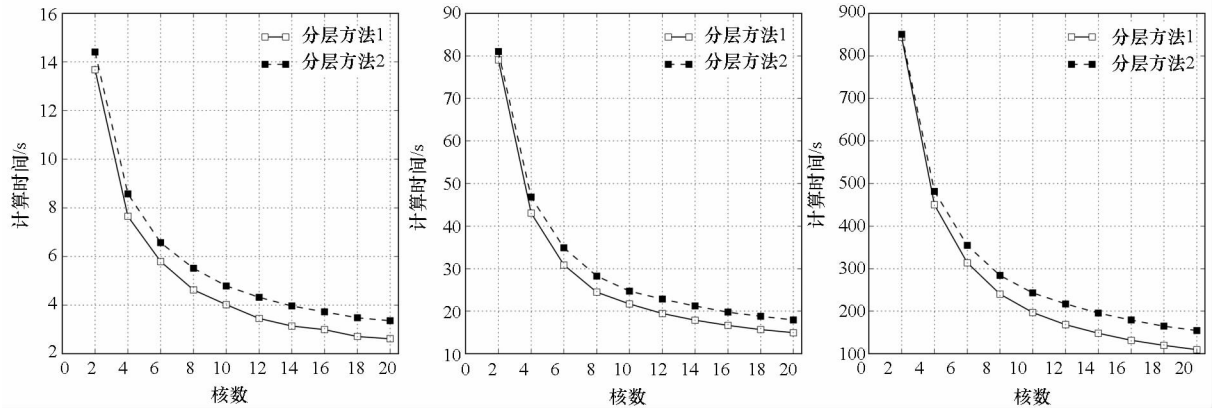
(图例中分层方法 1 指从上到下的方法,分层方法 2 指从下到上的方法)

Fig. 5 Change of speedup and parallelization efficiency with the number of cores under different experimental conditions

对于同样的栅格分层方法和核数,数据量较大的计算任务具有更高的加速比。这是由于对于给定的栅格分层方法和核数,调度开销大致是确定的,对于数据量大的计算任务,调度开销占总计算时间的比例更小,因而具有更好的并行性能。

栅格分层方法对并行性能的影响明显,从上

到下的分层方法(方法 1)比从下到上的分层方法(方法 2)具有更高的加速比和并行效率。图 6 给出了在两种分层方法情况下不同数据集的并行计算时间。从图 6 中可以看出,在不同数据规模下方法 1 的并行计算时间均小于方法 2,从而证明方法 1 确实比方法 2 具有更好的性能。

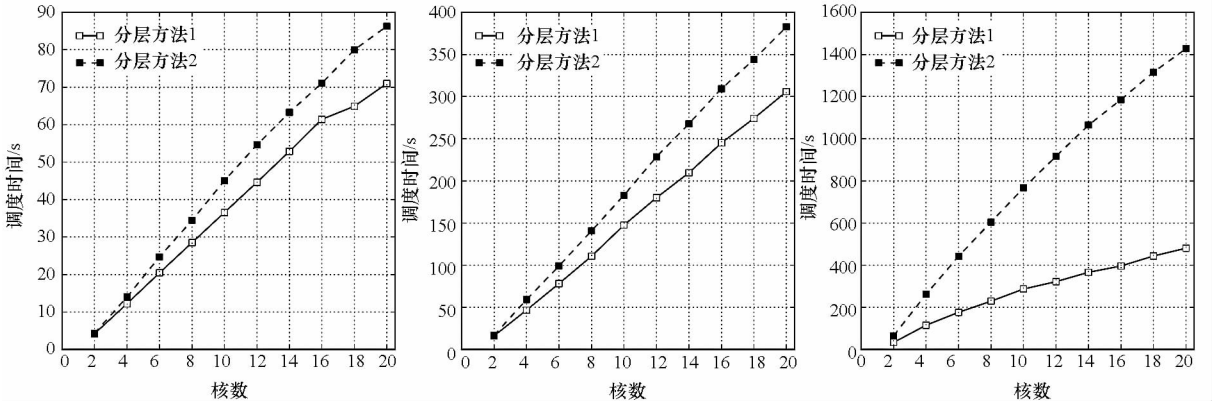


(a) 270m 数据集　(b) 90m 数据集　(c) 30m 数据集

(a) The 270m dataset　(b) The 90m dataset　(c) The 30m dataset

图 6　不同数据规模和分层方法情况下的并行计算时间

Fig. 6　The parallel computation time of different layering methods for different datasets



(a) 270m 数据集　(b) 90m 数据集　(c) 30m 数据集

(a) The 270m dataset　(b) The 90m dataset　(c) The 30m dataset

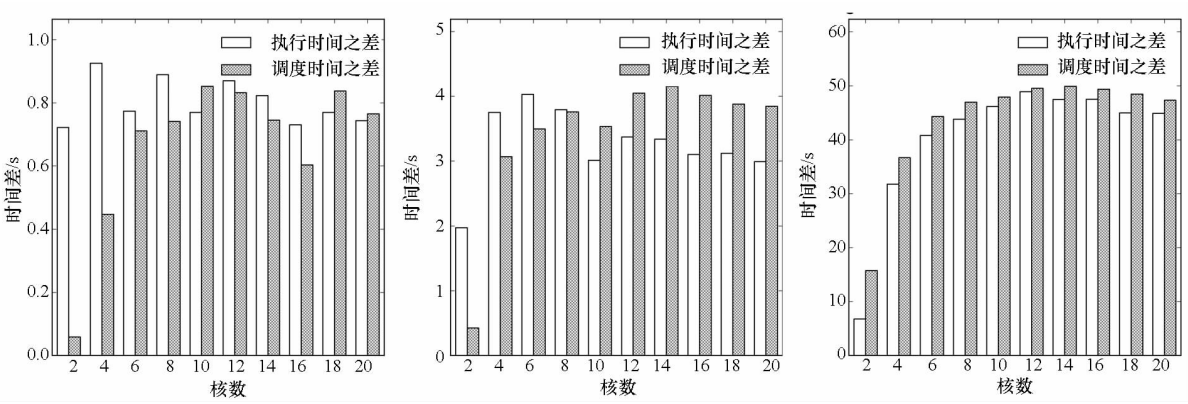
图 7　不同数据规模和分层方法情况下的并行调度时间

Fig. 7　The scheduling overhead of different layering methods for different datasets

对于两种分层方法,由于计算任务相同,计算时间的差别应该主要来自调度开销的差别。利用 Intel VTune™性能分析器在不同数据规模下测试了两种分层方法的调度开销(如图 7),调度开销包括 NtYieldExecution 函数、SwithToThread 函数和 vcomp100. dll 动态库中全部函数的运行时间。从图 7 中可以看出方法 1 的调度时间明显低于方法 2。图 8 给出了两种分层方法执行时间之差与平均调度时间(总调度时间除以核数)之差的对比,可以看出二者基本一致,说明总执行时间的差别主要由调度开销不同引起。

调度开销主要受线程数影响。对于核数为 N 的实验,本文利用 omp_set_num_threads(N)函数将线程数设为 N。但对于每个汇流层,实际执行的线程数并不总为 N。当汇流层中的栅格数大于 N 时,实际线程数为 N;而当汇流层中栅格数小于 N 时,实际线程数等于汇流层中的栅格数目。

对于不同分层方法得到的分层结果,每层中栅格数目的分布相差较大。对于方法 1(从上到下分层),大部分栅格集中于位于上游的少数汇流层,而下游的大部分汇流层包含很少甚至只有一个栅格;对于方法 2(从下到上分层),各层栅格分布比较平均。以 90m 分辨率数据集为例(如图 9),在全部 971 个汇流层中,对于方法 1,91.4% 的栅格集中于上游的 10 层,848 层栅格数少于 20,379 层只包含一个栅格;对于方法 2,只有 18 层栅格数少于 20,只有 1 层栅格数为 1。由于方法 1 层内栅格数少于指定线程数的层数更多,所以其实际线程数少于方法 2,从而具有更小的调度开销。

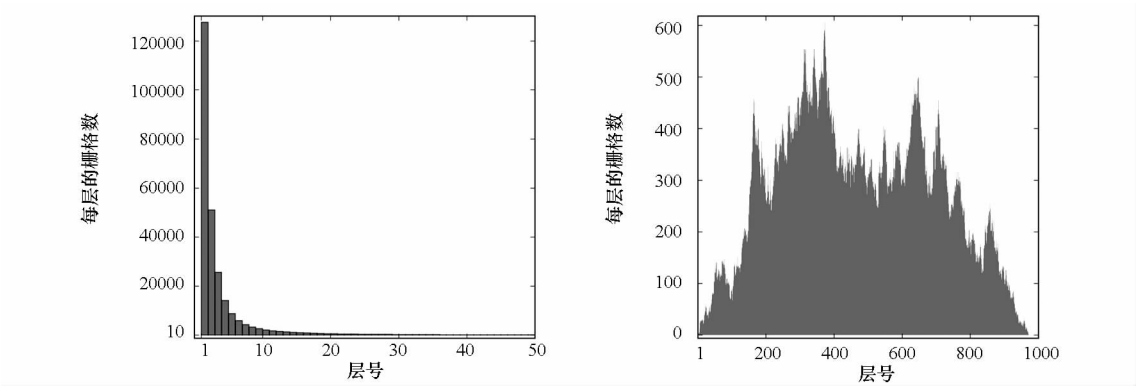


(a) 270m 数据集 (b) 90m 数据集 (c) 30m 数据集

(a) The 270m dataset (b) The 90m dataset (c) The 30m dataset

图 8 不同数据规模下两种分层方法执行时间之差与平均调度时间之差

Fig. 8 The execution time difference and the average scheduling time difference between the two layering methods for different datasets



(a)方法 1(只显示了上游的 50 层) (b)方法 2

(a) The up-down method (only the upstream 50 layers are displayed) (b) The down-up method

图 9 不同分层方法每层栅格数目分布

Fig. 9 The number of grids within each routing layer for different layering methods

3 总结

本文针对逐栅格汇流算法用于大流域长历时模拟的计算瓶颈问题,基于栅格分层的思想提出一种适用于共享内存并行计算环境、基于隐式有限差分的逐栅格汇流并行算法,基于 OpenMP 编程实现,并选择河北省清水河流域作为实验区对算法进行了验证。结果表明,该算法具有较好的加速比和并行效率,且对于数据量大的计算任务更能体现其优势;栅格分层方法对算法并行性能有明显影响,从上到下的分层方法比从下到上的方法具有更高的并行效率。在下一步工作中,一方面拟在该算法基础上耦合其他流域过程模块,实现逐栅格汇流分布式水文模型的并行化;另一方面,由于共享内存方式并行计算在可扩展性方面不如消息传递方式,为进一步提高该算法的可伸缩性,拟在共享内存并行算法的基础上,进一步在子流域层次实现消息传递方式的并行计算,最终实现子流域 - 栅格双层并行算法。

参考文献(References)

[1] Wigmosta M S, Vail L W, Lettenmaier D P. A distributed hydrology-vegetation model for complex terrain [J]. Water Resources Research, 1994, 30(6): 1665 - 1679.

[2] Ciarapica L, Todini E. TOPKAPI: a model for the representation of the rainfall-runoff process at different scales [J]. Hydrol. Process., 2002, 16(2): 207 - 229.

[3] Van Der Knijff J M, Younis J, De Roo, et al. LISFLOOD: a GIS-based distributed model for river basin scale water balance and flood simulation[J]. International Journal of Geographical Information Science, 2010, 24(2):189 - 212.

[4] Zhang Z Q, WANG S P, SUN G, et al. Evaluation of the MIKE SHE model for application in the Loess Plateau, China [J]. Journal of the American Water Resources Association, 2008, 44(5): 1108 - 1120.

[5] Vivoni E R, Mascaro G, Mniszewski S, et al. Real-world hydrologic assessment of a fully-distributed hydrological model in a parallel computing environment[J]. Journal of Hydrology, 2011, 409(1 - 2):483 - 496.

[6] Wang H, Zhou Y, Fu X, et al. Maximum speedup ratio curve (MSC) in parallel computing of the binary-tree-based drainage network[J]. Computers & Geosciences, 2012, 38(1):127 - 135.

[7] 李丽. 分布式水文模型的汇流演算研究[D]. 南京:河海大学,2007.
LI Li. Study on flood routing of distributed hydrologic models [D]. Nanjing: Hehai University, 2007. (in Chinese)

[8] Huang J X, Song C C S. Stability of dynamic flood routing schemes[J]. Journal of Hydraulic Engineering, ASCE, 1985, 111(12):1497 - 1506.

[9] Neal J C, Fewtrell T J, Trigg M. Parallelisation of storage cell flood models using OpenMP[J]. Environmental Modelling & Software, 2009, 24(7): 872 - 877.

[10] Yu D P. Parallelization of a two-dimensional flood inundation model based on domain decomposition[J]. Environmental Modelling & Software, 2010, 25(8): 935 - 945.

[11] 李光炽, 王船海. 大型河网水流模拟的矩阵标识法[J]. 河海大学学报, 1995, 23(1): 36 - 43.
LI Guangzhi, WANG Chuanhai. Matrix mark method for large-scale river network flow modeling[J]. Journal of Hehai University (Natural Sciences), 1995, 23(1): 36 - 43. (in Chinese)

[12] Liu J T, Chen X, Zhang J B, et al. Coupling the Xinanjiang model to a kinematic flow model based on digital drainage networks for flood forecasting[J]. Hydrological Processes, 2009, 23(9):1337 - 1348.

[13] Kazezylmaz-Alhana C M, Medina Jr M A, et al. On numerical modeling of overland flow [J]. Applied Mathematics and Computation, 2005, 166(3):724 - 740.

[14] Chow V T, Maidment D R, Mays L W. Applied hydrology [M]. New York: Mc Graw-Hill Professional, 1988.

[15] 吴建平, 王正华, 李晓梅. 利用混合编程改善 SMP 机群上并行矩阵乘法的性能[J]. 国防科技大学学报, 2006, 28(4):68 - 72.
WU Jianping, WANG Zhenghua, LI Xiaomei. Improve the performance of parallel matrix multiplication on clustered SMP systems through hybrid programming[J]. Journal of National University of Defense Technology, 2006, 28(4):68 - 72. (in Chinese)

[16] 王纲胜, 夏军, 牛存稳. 分布式水文模拟汇流方法及应用[J]. 地理研究, 2004, 23(2):175-182.
WANG Gangsheng, XIA Jun, NIU Cunwen. Flow routing method and its application in distributed hydrological modeling [J]. Geographical Research, 2004, 23(2):175 - 182. (in Chinese)

[17] 周伟明. 多核计算与程序设计[M]. 武汉:华中科技大学出版社, 2009.
ZHOU Weiming. Multicore computing and programming[M]. Wuhan: Huazhong University of Science & Technology Press, 2009. (in Chinese)

[18] Liu Y B, Gebremeskel S, De Smedt, F, et al. Predicting storm runoff from different land use classes using a GIS-based distributed model[J]. Hydrological Processes, 2006, 20: 533 - 548.