Windows 环境下河网水流多线程并行计算

王船海,曾贤敏

(河海大学水文水资源与水利工程科学国家重点实验室,江苏 南京 210098)

摘要:基于多任务 Windows 操作系统的线程和事件机制,利用 Windows 系统 API 函数创建多个线程,并对共享内存式的多个线程运行互斥与同步实行统一管理,从而实现了河网水流的多线程并行计算.测试结果表明,采用与 CPU 内核数相同的线程数进行并行计算,可以达到最佳的并行计算效果.

关键词:河网水流;Windows 环境;并行计算

中图分类号:TV131.4 文献标识码:A 文章编号:1000-1980(2008)01-0030-05

并行程序实现的主要模式有 2 种:共享内存模式和消息传递模式[1-2].共享内存模式可通过 OpenMP 来实现,该模式将并行化指令添加到程序代码中,由编译器实现自动并行化;消息传递模式可通过 MPI 来实现,该模式利用消息传递机制实现多个计算单元的并行计算[1].这 2 种模式在 Linux 和 Unix 环境下的应用较多,而在 Windows 环境下的应用较少.相对于 Unix 和 Linux 系统,Windows 系统的用户和应用软件更多,用户对 Windows 系统也更熟悉,常见的小规模科学计算也都是在 Windows 环境下进行的.

河网水流计算是一种具有连续过程性质的计算,若采用 MPI 模式实现并行,则每个计算单元拥有各自的内存单元,各并行线程间通信量很大,网速和通信量对并行性能有很大的影响,网络传输花费的时间甚至会超过模型求解时间,而利用 OpenMP 指令可以实现共享内存的并行编程.但利用 OpenMP 指令对现有程序进行改造,几乎要重写所有程序代码,编程量很大.

本文利用 Visual C++6.0 开发工具和多任务操作系统 Windows 线程机制创建了多个线程,采用面向对象的程序设计方法设计了一个管理线程的互斥与同步的线程管理类,在不破坏现有程序结构的前提下,对现有程序求解模块进行了修改,从而实现了多 CPU 环境下共享内存的河网水流并行计算[3].

1 河网水流模型

1.1 一般求解方法

为便于问题的求解,假定河网流域由河道、湖泊和“联系”3 种基本要素构成[4].其中“联系”是指控制水流的堰、闸和行洪区口门等,可用宽顶堰公式来描述,而通过“联系”的流量可表示为“联系”首、末节点水位的线性组合[5-6].描述河道水流的数学模型为一维圣维南方程组,其求解步骤是:建立双追赶方程;构建递推关系;将河道各断面的水位和流量表示为河道首末节点水位的线性组合[5-7].根据水量平衡原理,在河网节点处建立以节点水位为基本未知变量的线性方程组,解出节点水位,回代即可求出整个流域河网的水位和流量.

1.2 并行求解方法

并行计算方式有 3 种:数据并行、功能并行和流水线[1].数据并行是指不相关的任务对数据集中不同元素执行相同操作;功能并行是指不相关的任务对数据集中不同元素执行不同操作;流水线是指将单个任务分几个阶段来执行,即几个阶段的任务串行执行,当多个任务同时执行时,就可以实现多个任务不同阶段的并行.对河网水流并行求解而言,程序设计时一般只采用数据并行方式.

若采用文献[7]的方法来求解河网水流,则主要的求解工作是求解追赶方程系数、求解节点水位方程和回代求解水位流量,求解流程如图 1 所示.图中:T 表示当前的计算时间步;E 表示需要计算的总步数;N 表示线程数.

收稿日期:2006-11-06

作者简介:王船海(1963—),男,江苏海安人,副教授,主要从事水文水资源、计算水力学和 GIS 研究.

1.2.1　追赶方程系数的求解

在追赶方程系数的求解过程中，不同河道追赶系数的求解工作互不相关，因此可利用数据并行方式求解追赶系数．由于求解计算工作量与河道断面数成正比，按河道断面数将计算工作量平均分配给各个线程，以使所有线程负载基本达到平衡[1]．但是，将河道断面数平均分配给线程数后，可能会导致同一河道断面追赶方程系数的计算在不同线程上进行，若直接并行求解，线程间的通信量会很大．因此，先按河道断面数均分的分配原则，然后把跨越线程的河道断面系数计算调整到同一个线程上．这样，既大大减小了线程间的依赖性，又考虑到了线程间的负载平衡．此外，在构建节点水位方程时，一些公用变量会被多个线程同时更新，串行程序中这种更新不会出错，但在多线程并行计算时会出现覆盖正确值的现象．因此，并行计算时一般会在涉及公用变量更新的地方设置临界区保护，以使各个线程互斥访问公用变量．这样会造成线程间的等待，影响并行效率．为解决这一问题，本文采用了以空间换时间的方法，即让每个线程都拥有一组公用变量拷贝，待所有并行求解完成后再对公用变量做一次归约运算．采用这样的处理方法，在追赶方程构建这一步，就可以完全消除线程间的依赖关系．

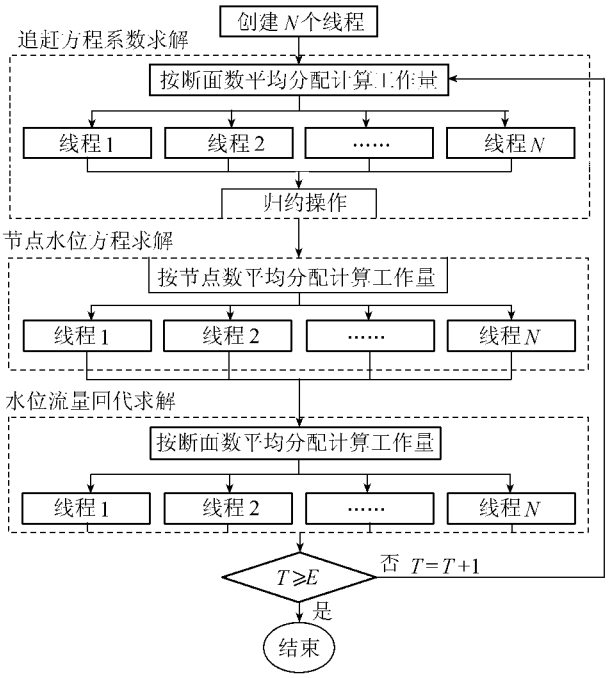

图 1　河网水流求解流程

Fig.1　Flow chart of river network flow

1.2.2　节点水位方程的求解

由于节点水位方程的系数矩阵主对角占优，本文采用矩阵标识法[7]，即仅对非零元进行运算．这极大地提高了求解效率．线程计算工作量的分配方法是把总的节点数平均地分配给每个线程．由于是迭代求解，节点水位迭代与顺序无关，线程间不存在依赖关系，可完全并行计算．

1.2.3　水位流量的回代求解

同线程间计算工作量的分配思路与 1.2.1 节所述分配思路相似，但线程间不存在依赖关系，不需要做额外改造，可以直接并行求解．

2　程 序 设 计

2.1　Windows 系统线程创建函数和同步控制类的功能[8]

Windows 系统创建的线程有 2 种：用户界面线程和工作线程．由于河网水流求解只涉及数值计算，没有用户界面交互，故选用 CreateThread()创建工作线程．实现并行计算主要需要解决的问题是线程间的互斥及同步．线程间的同步与互斥控制有多种实现方法，由于采用事件方法可以解决同步与互斥的控制问题，故采用 MFC 的 CEvent 类．Windows API 创建线程函数 CreateThread()和 CEvent 类的功能[4]分别是：(a) CreateThread()——Windows 线程创建函数，用来创建并发执行的线程对象；(b) CEvent——MFC 中的同步操作类，某事件发生时向其他线程发送消息．

2.2　并行计算中用到的主要 C ++ 线程管理类及相关数据结构

```
enum enumParellelType {ePreRiver,eRunRiver,eBackRiver};  //枚举类型：追赶、求解水位方程、回代
struct SThreadInfo//参数数据结构
{
    CRiverNetModel * m_RiverNetModel;//河网模型对象
    CEvent m_StartEvent,m_StepFinishEvent,m_EndEvent;//并行计算启动、单步完成、并行计算结束事件
    enumParellelType m_ParellelType;//标识线程执行的操作类型
    int LRiverBond,URiverBond,LNodeBond,UNodeBond;//线程分配的数组上下界
```

```
        HANDLE m_thread;//线程句柄
};
class CThreadManage//线程管理类
 {
    ……
public:
    CThreadManage(int numThread,CRiverNetModel * RiverNetModel);
    virtual ~ CThreadManage();
    void RunModel(enumParellelType ParellelType);//调用河网模型并进行计算
        BOOL StopRun();//线程结束函数
    DWORD static WINAPI RunThreadProc(LPVOID pParam);//线程启动函数
    void InitializePreRiver();//初始化模型追赶数组
    void ReductionPreRiver();//归约运算,以空间换时间
public:
    int m_NumThread;//线程数
    SThreadInfo * m_ThreadInfo;//线程信息
    CRiverNetModel * m_RiverNetModel;//河网模型对象
};
Class CRiverNetModel//河网模型类
 {
    ……
    //系数矩阵追赶
    void PreRiver(double * RA,double  * RH,double  * AA,long LBond,long UBond);
    BOOL SolveZ (CThreadManage &m_ThreadManage);//迭代求解节点水位
    void BackRiver(long LBond,long UBond);//断面水位、流量回代
    void Run();//程序启动运行函数
    CThreadManage * m_ThreadManage;
}
```

2.3 多线程程序关键代码段

```
    //创建线程,平均分配每个线程的计算工作量
    CThreadManage::CThreadManage(int numThread,CRiverNetModel  *  RiverNetModel)
     {
        ……
        for (ii = 0;ii < numThread;ii + + )
         {
          //初始化线程参数
          m_ThreadInfo[ii].m_StartEvent.ResetEvent();
          m_ThreadInfo[ii].m_EndEvent.ResetEvent();
          m_ThreadInfo[ii].m_StepFinishEvent.SetEvent();
          m_ThreadInfo[ii].m_RiverNetModel = RiverNetModel;
          //平均分配每个线程的计算工作量,该操作通过每个线程数组上下边界来实现
          m_ThreadInfo[ii].LRiverBond = ii * RiverNetModel -> m_Number/numThread;
m_ThreadInfo[ii].URiverBond = (ii + 1) * RiverNetModel -> m_Number/numThread;
          ……
          //创建线程
              m_ThreadInfo[ii].m_thread = CreateThread(NULL,0,RunThreadProc,
```

```
                    &m_ThreadInfo[ii],0,NULL);
        }
    }
    //多线程管理函数
    DWORD CThreadManage::RunThreadProc(LPVOID pParam)
    {
        //获得线程信息
        SThreadInfo * pThreadInfo = (SThreadInfo *)pParam;
        CRiverNetModel * pTaiHuRiverNetModel = pThreadInfo -> m_TaiHuRiverNetModel;
        CEvent * StartEvent, * StepFinishEvent, * EndEvent;
        enumParellelType * pParellelType = &pThreadInfo -> m_ParellelType;;
        StartEvent = &pThreadInfo -> m_StartEvent;
        StepFinishEvent = &pThreadInfo -> m_StepFinishEvent;
        EndEvent = &pThreadInfo -> m_EndEvent;
    while (TRUE)
    {
        //多线程等待执行
        ::WaitForSingleObject(StartEvent -> m_hObject,INFINITE);
            //事件指示结束多线程执行
        if (WAIT_OBJECT_0 == ::WaitForSingleObject(EndEvent -> m_hObject,0))break;
        //调用河网模型并进行计算,pParellelType 标识执行类型,pThreadInfo 线程参数信息
        m_RiverNetModel -> RunModel(*pParellelType,pThreadInfo);
        StepFinishEvent -> SetEvent();
    }
        return 0;
    }
```

2.4　并行计算相关类调用说明

在河网模型计算起始处创建线程管理对象，线程管理对象根据系统 CPU 核数创建线程数，并分配各个线程计算工作量. 当程序运行到上述需要并行计算的模块时，调用 RunModel(ParellelType)函数，其中变量 ParellelType 标识了河网模型不同的求解步骤. 由线程管理对象将求解计算工作量平均分解到各个线程、调用模型相应求解模块，同时管理各个线程间的互斥与同步. 整个河网水流求解过程结束后删除线程，释放相关资源. 相关类的调用关系如图 2 所示.

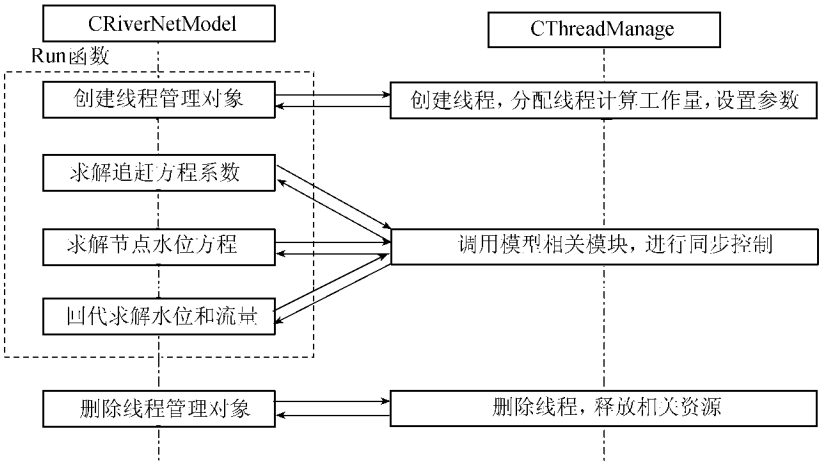

图 2　相关类图

Fig.2　Related class graphs

3 计算效率对比

首先按照上述多线程并行算法,利用 MicroSoft 公司开发工具 VC6 编写了并行计算线程管理类,并对原有的河网水流模型代码进行了改造,然后在 Windows Server 2003 操作系统、2 个 Intel Xeon 3.0G CPU(超线程)和 8G 内存计算机系统环境下,对概化后的具有 1 482 个节点的太湖流域河网水流模型进行了模拟计算,结果如表 1 所示.从表 1 可知,4 个线程并行求解时间大约是单线程求解时间的 50%,即用 4 个线程进行并行求解基本上可达到最佳运行效果(2 个 CPU 的运行时间为单个 CPU 运行时间的一半).

表 1 模型求解效率对比

Table 1 Comparison of solving efficiency of model

线程数	总运行时间/s	求解水位方程运行时间/s	加速比
1	1 122	982	
2	856	738	1.33
3	726	624	1.57
4	653	499	1.97
5	779	643	1.53

4 结 语

本文在分析河网水流求解特点的基础上,设计了实现追赶方程系数求解、节点水位方程求解以及水位和流量回代求解 3 部分并行计算的方案,并通过数值试验验证了该方案的可行性.对比计算及其分析结果表明:(a)采用 Windows 多线程技术可实现并行计算,并能提高数学模型的求解效率;(b)求解运算的线程数与系统 CPU 核数相同时,并行计算效率最高;(c)提高并行程序求解效率的关键是消除线程间的依赖关系;(d)共享内存并行计算方式是目前河网水流模型并行求解最适合的方式.

参考文献:

[1] QUINN M J. MPI 与 OpenMP 并行程序设计:C 语言版[M]. 陈文光,武永卫,译. 北京:清华大学出版社,2004:8-9.
[2] 安虹,陈国良. 并行程序设计模型和语言[J]. 软件学报,2002(1):118-123.
[3] 陈华平,安虹,黄刘生,等. 分布式任务调度算法的仿真环境研究[J]. 中国科学技术大学学报,1999(8):421-426.
[4] 李光炽,王船海. 流域洪水模拟通用模型结构研究[J]. 河海大学学报:自然科学版,2005,33(1):14-17.
[5] 李光炽,王船海. 流域洪水演进模型通用算法研究[J]. 河海大学学报:自然科学版,2005,33(6):624-628.
[6] 李光炽,王船海. 内涝型流域洪灾洪水模拟[J]. 四川大学学报,1995(1):87-96.
[7] 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.
[8] KRUGLINSKI D J,WINGO S,SHEPHERD G. Programming Visual C ++ 6.0 技术内幕:修订版[M]. 5 版. 希望图书创作室,译. 北京:希望电子出版社,2001.

Multithreading parallel computation of river network flow in Windows environment

WANG Chuan-hai, ZENG Xian-min

(*State Key Laboratory of Hydrology-Water Resources and Hydraulic Engineering, Hohai University, Nanjing* 210098, *China*)

Abstract: Based on the mechanism of thread and events of multi-task Windows operation system, multi-threads were constructed by use of API function of Windows system, and the unified management of mutually exclusive and synchronous memory-shared multithreading operation was implemented. In this way, the multithreading parallel computation of river network flow was realized. Test result shows that the optimum effect of parallel computation could be obtained by adoption of the number of threads equal to that of kernels of CPU.

Key words: river network flow; Windows environment; parallel computation