DOI:10.3880/j.issn.1006-7647.2017.02.009

基于 Fork/Join 多核并行框架的梯级水库群优化调度

王 森[1,2],马志鹏[1,2],李善综[3],王凌河[1,2],熊 静[1,2]

(1.珠江水利科学研究院,广东 广州 510611; 2.水利部珠江河口动力学及伴生过程调控重点实验室,广东 广州 510611;
3.水利部珠江水利委员会技术咨询中心,广东 广州 510611)

摘要:为了满足大规模梯级水库群优化调度精细化管理需求,解决决策计算耗时长及求解效率低等困难,提出了基于 Fork/Join 多核并行框架的梯级水库群优化调度并行求解方法,并以离散微分动态规划方法并行化为例,给出了梯级水库群优化调度方法在 Fork/Join 框架下的并行化实现方式。红水河大规模梯级水库群长期发电优化调度测试结果表明,并行计算能够充分发挥多核处理器的加速性能,有效缩短计算耗时,提高求解效率;选择合理的 Fork/Join 框架规模控制阈值是充分发挥并行优势的关键因素。

关键词:梯级水库群;优化调度;Fork/Join 并行框架;多核处理器;并行计算

中图分类号:TV697.1+2 **文献标志码**:A **文章编号**:1006-7647(2017)02-0048-07

Optimal operation of cascaded reservoirs based on Fork/Join multi-core parallel framework//WANG Sen[1,2], MA Zhipeng[1,2], LI Shanzong[3], WANG Linghe[1,2], XIONG Jing[1,2] (1. *Pearl River Hydraulic Research Institute*, *Guangzhou* 510611, *China*; 2. *Key Laboratory of the Pearl River Estuarine Dynamics and Associated Process Regulation*, *Ministry of Water Resources*, *Guangzhou* 510611, *China*; 3. *Technical Advisory Center of Pearl River Resources Commission*, *Ministry of Water Resources*, *Guangzhou* 510611, *China*)

Abstract: In order to meet the refined management demand of optimal operation of large-scale cascaded reservoirs and solve the problems of long running time and low computational efficiency, a parallel method based on the Fork/Join multi-core parallel framework is proposed for optimal operation of large-scale cascaded reservoirs. A parallel discrete differential dynamic programming (PDDDP) was designed to describe the parallelization scheme for optimal operation of cascaded reservoirs based on the Fork/Join framework. The long-term power generation optimal operation of large-scale cascaded reservoirs on the Hongshui River was used as a case study. The testing results show that the parallel method can make full use of the acceleration performance of the multi-core processor, significantly reducing the computation time, and improving the computational efficiency. Moreover, the choice of the reasonable scale control threshold of the Fork/Join framework is critical to taking full advantage of parallel computation.

Key words: cascaded reservoirs; optimal operation; Fork/Join parallel frame; multi-core processor; parallel computing

近年来,我国水利事业发展迅猛,形成了或将形成一大批具有电站级数多等显著特点的大规模梯级水库群。这类大规模梯级水库群优化调度问题具有非常显著的高维数、非线性、多阶段、约束强耦合等特点[1],动辄十几级甚至几十级电站规模使“维数灾”问题变得尤为突出,求解难度越来越大,计算效率越来越低,采用传统优化方法求解呈现一定的局限性,无法满足流域机构、电网和水电公司等部门对水库群的精细化调度需求,探索合理高效的求解算法是大规模水库群调度工作亟待研究的重要课题。

随着计算机科学高速发展以及硬件配置不断升级和更新,无论个人电脑还是高性能运算服务器,多核处理器已成为当今主流计算机配置,为并行技术的推广和应用奠定了基础。同时,由于串行方法只能调用单一线程进行计算,无法充分利用多核配置的加速性能,易造成多核资源浪费。因此,将串行方法改造成高性能并行方法是避免资源闲置、提高方法计算效率的必然趋势。

在水库调度学科领域,学者们一直非常重视方法并行化研究与应用,且取得了丰硕的成果,大幅度提高了优化方法计算效率。常见的串行方法并行化方式主要有粗粒度和细粒度两种。陈立华等[2-3]利

基金项目:水利部公益性行业科研专项(201401013,201501010)

作者简介:王森(1986—),男,工程师,博士,主要从事大规模梯级水库群联合优化调度和水利信息化等研究。E-mail:wangsen19861227@163.com
通信作者:马志鹏(1979—),男,高级工程师,博士,主要从事水文水资源系统分析等研究。E-mail:zhipengma@163.com

用群体智能算法的天然并行性特点,将算法初始种群划分为多个规模大小相同的子种群,并平均分配到不同内核中同时计算,实现了群体算法的粗粒度并行化计算;张忠波等[4-6]根据动态规划算法中各状态变量组合的天然并行性,提出了细粒度并行设计的并行动态规划方法。上述成果普遍利用了已成熟的 OpenMP、MPI 或. NET 等能够有效兼容 Fortran、C++、C#等语言的并行模式或框架,且为 Fortran、C++、C#等语言开发的串行方法并行化提供了便利,但是这些框架并不能有效适应 Java 语言开发的串行方法,难以应用于 Java 编码的程序并行化开发和改造。因此,实现串行方法并行化,选择合适的并行模式或框架非常重要,不仅需要考虑编程语言的支持类型、并行任务的应用性以及平台兼容性等因素,而且开发人员对不同编程语言的熟悉程度也对并行框架的选择有重要影响。

Fork/Join 并行框架是一种基于 Java 源码开发的多核并行框架[7],已作为标准程序集成到 Java 版本 7 的并发程序包中,能够最大效率地简化开发人员的编程工作,便于 Java 编码开发的串行方法并行化改造。目前,基于 Fork/Join 框架的并行方法研究成果相对较少[8-9],且未深入阐述其原理、特点、实现方法以及性能影响因素。本文在阐述 Fork/Join 的应用原理及方法的基础上,以离散微分动态规划(discrete differential dynamic programming,DDDP)并行化求解为例,提出基于 Fork/Join 多核并行框架的大规模梯级水库群优化调度并行求解方法,并通过计算测试分析影响 Fork/Join 性能的关键因素,提出设置 Fork/Join 阈值的适应性公式。

1 基于 Fork/Join 框架的并行化设计方法

1.1 Fork/Join 框架基本原理

Fork/Join 并行框架的核心技术继承了“分治法”的特性,通过递归分割原问题,形成若干个规模更小、相互独立且可并行计算的子问题。当各子问题进行独立并行计算后,组合各子问题的子结果即可输出原问题的最终结果,图 1 为“分治法”流程框架。Fork/Join 框架设计了独特的线程池技术,当程序开始执行时,默认创建与可用的处理器核心数目

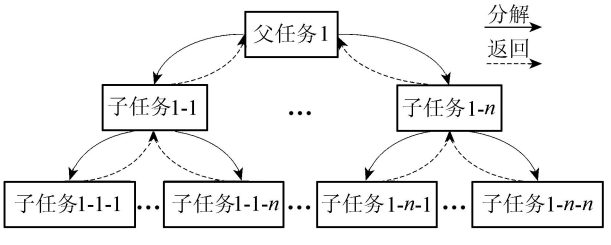


图 1 “分治法”流程框架

相同的活动线程数,便于线程的维护和管理,减小了系统资源消耗,尤其适合在单机多核配置上部署多线程程序。在“分治法”执行过程中,定义了一个可自由设置的控制子问题规模大小的阈值作为子问题规模的上限值,即当子问题规模小于或等于阈值时,“分治法”执行结束,各子问题被平均分配到不同线程中开始并行计算,图 2 为阈值控制模式下的子问题分割示意图。另外,在子问题并行计算过程中,Fork/Join 并行框架设计了独特的双端队列排序模式,并采用了“工作窃取”算法,即当一个线程的队列计算任务为空时,将从其他处于工作状态的线程队列尾部“窃取”计算任务,避免了线程处于闲置状态,提高线程的利用效率,图 3 为“工作窃取”算法示意图。

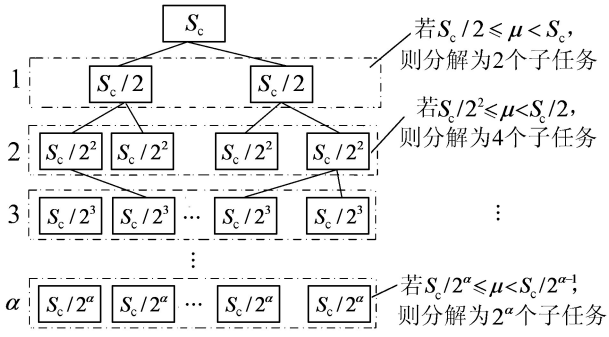


图 2 阈值控制模式下的子问题分割示意图

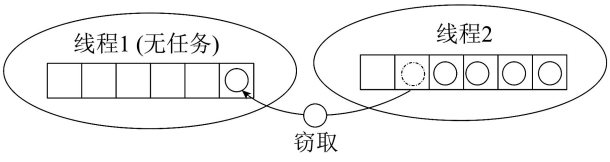


图 3 “工作窃取”算法示意图

1.2 Fork/Join 框架实现方法

Fork/Join 并行框架的接口类以及实现类封装在 Java. util. concurrent 中。虽然 Fork/Join 的并行效率与其他并行框架相比并非最优[7],但其最大优点在于提供了非常简便的应用程序实现接口。开发人员不再需要处理同步、通信等各种相关事务,可避免难以调试的死锁和数据争用等错误,只需按照给定方法接口的实现方式,编写很少的程序即可完成算法与框架接口的对接,极大地简化了编写并行程序的琐碎工作,提高了工作效率,Fork/Join 接口程序伪代码见图4。对于开发人员而言,应用 Fork/Join 并行框架实现算法并行化计算主要需关注以下 4 个问题:

a. 算法实现的代码类需继承 Fork/Join 应用接口类 java. util. concurrent. RecursiveAction 或者 java. util. concurrent. RecursiveTask。

b. 选择合适的阈值划分任务。由图 2 可知,阈值的大小直接影响子问题的分割数目。当阈值设置

```
public class ComputeAllTask extends RecursiveAction
                                                    接口类
  protected void compute() {
      if (end - start <= THRESHOLD)                 阈值
          ComputeTask(start, end);
      else {
        int pivot = (start + end) /2;               子任务划分
        this.invokeAll(new ComputeAllTask (pivot + 1, end),
              new ComputeAllTask (start, pivot ));
                                                    子任务递归
      }
  }
  publish void ComputeTask(int start, int end){
              for(int i=start; i<=end; i++)
                子任务串行计算代码;                  子任务计算
}
```

图 4　Fork/Join 接口程序伪代码

偏小时,意味着子问题的规模更小且分割数量更多,易造成资源管理消耗过大;当阈值设置偏大时,意味着子问题的规模更大且分割数量更少,而当子问题数量小于工作线程数时,易导致部分工作线程处于闲置状态。为了保证所有工作线程均能分配到子问题,通用的阈值设置公式可表示为

$$\mu = \lceil \frac{S_c}{w} \rceil \tag{1}$$

式中:⌈ ⌉为对计算结果取上整数;μ 为规模控制阈值;S_c 为原问题规模;w 为多核处理器逻辑线程数(具有"超线程"技术的处理器可在 1 个内核中模拟 2 个逻辑线程)。另外,框架应用接口类提供了划分子问题的方法接口,开发人员只需在接口程序中提供合适的阈值,不需要关心程序任务划分以及线程分配如何实现。

c. 实现 Fork/Join 接口类的 void compute( )方法。该方法接口主要实现各子任务的计算,即将算法可并行化部分的代码程序写入该方法接口。

d. 设置任务划分方式。每次任务划分可将父任务划分为多个子任务,而划分的子任务个数由开发人员编写代码实现。一般来说,采用对半分解父任务的方式有利于硬件实现负载均衡。

1.3　DDDP 并行化设计方法

DDDP 方法是由动态规划方法衍生的一种改进方法,主要通过减少离散点数以及不断缩小搜索范围缓解"维数灾"问题,方法原理及求解流程详见文献[10]。由于 DDDP 方法与动态规划方法及其改进方法的核心均为递推方程[1],以下以 DDDP 方法并行化为例,描述所有离散组合在递推方程中的求解计算并行化设计方法。

1.3.1　DDDP 方法并行性分析

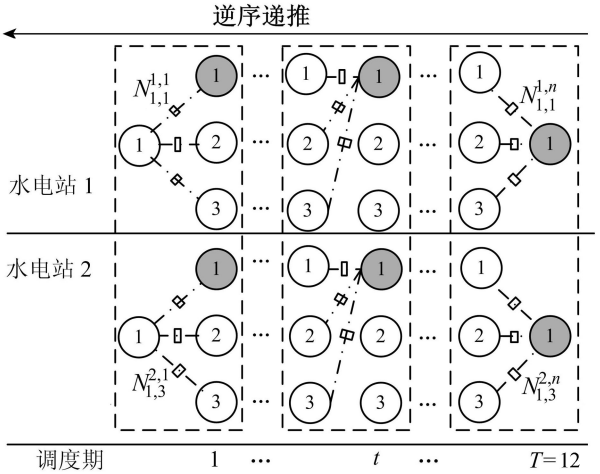
以计算时段为月、调度周期为 1 年以及始末水位固定的 2 库水电系统长期优化调度为例,采用 DDDP 方法求解,则在任意廊道内一次迭代过程的计算任务规模为 $3^2+10\times3^4+3^2$,见图 5。假设一个水电系统长期优化调度问题包含 M 个水电站,调度周期 T,则 DDDP 方法一次迭代过程的计算规模可表示为

$$3^M + (T-2) \times 3^{2M} + 3^M \tag{2}$$

从图 5 可知,由于 DDDP 方法采用离散状态变量的寻优思想,且计算每个多维组合状态在递推方程中的求解计算都是独立的,具有天然并行性,因此,每次廊道内所有变量组合在递推公式中的求解可作为父任务,并通过 Fork/Join 递归划分为多个子任务,实现 DDDP 方法的并行计算。

1.3.2　DDDP 方法并行化设计

根据 Fork/Join 的实现方法,与 DDDP 方法求解相结合,并行离散微分动态规划(parallel discrete differential dynamic programming, PDDDP)方法设计流程如图 6 所示。根据 PDDDP 方法设计流程,优化



阶段	出力决策组合	计算规模
T	$\{(N_{1,1}^{1,n},N_{1,1}^{2,n}),(N_{1,1}^{1,n},N_{2,1}^{2,n}),\cdots,(N_{3,1}^{1,n},N_{2,1}^{2,n}),(N_{3,1}^{1,n},N_{3,1}^{2,n})\}$	3^2
⋮	⋮	⋮
t	$\{(N_{1,1}^{1,t},N_{1,1}^{2,t}),(N_{1,1}^{1,t},N_{1,2}^{2,t}),(N_{1,1}^{1,t},N_{1,3}^{2,t}),\cdots,(N_{1,3}^{1,t},N_{1,1}^{2,t}),(N_{1,3}^{1,t},N_{1,2}^{2,t}),(N_{1,3}^{1,t},N_{1,3}^{2,t}),$ $(N_{1,1}^{1,t},N_{2,1}^{2,t}),(N_{1,1}^{1,t},N_{2,2}^{2,t}),(N_{1,1}^{1,t},N_{2,3}^{2,t}),\cdots,(N_{1,3}^{1,t},N_{2,1}^{2,t}),(N_{1,3}^{1,t},N_{2,2}^{2,t}),(N_{1,3}^{1,t},N_{2,3}^{2,t}),$ $(N_{1,1}^{1,t},N_{3,1}^{2,t}),(N_{1,1}^{1,t},N_{3,2}^{2,t}),(N_{1,1}^{1,t},N_{3,3}^{2,t}),\cdots,(N_{1,3}^{1,t},N_{3,1}^{2,t}),(N_{1,3}^{1,t},N_{3,2}^{2,t}),(N_{1,3}^{1,t},N_{3,3}^{2,t}),$ ⋮ $(N_{3,1}^{1,t},N_{1,1}^{2,t}),(N_{3,1}^{1,t},N_{1,2}^{2,t}),(N_{3,1}^{1,t},N_{1,3}^{2,t}),\cdots,(N_{3,3}^{1,t},N_{1,1}^{2,t}),(N_{3,3}^{1,t},N_{1,2}^{2,t}),(N_{3,3}^{1,t},N_{1,3}^{2,t}),$ $(N_{3,1}^{1,t},N_{2,1}^{2,t}),(N_{3,1}^{1,t},N_{2,2}^{2,t}),(N_{3,1}^{1,t},N_{2,3}^{2,t}),\cdots,(N_{3,3}^{1,t},N_{2,1}^{2,t}),(N_{3,3}^{1,t},N_{2,2}^{2,t}),(N_{3,3}^{1,t},N_{2,3}^{2,t}),$ $(N_{3,1}^{1,t},N_{3,1}^{2,t}),(N_{3,1}^{1,t},N_{3,2}^{2,t}),(N_{3,1}^{1,t},N_{3,3}^{2,t}),\cdots,(N_{3,3}^{1,t},N_{3,1}^{2,t}),(N_{3,3}^{1,t},N_{3,2}^{2,t}),(N_{3,3}^{1,t},N_{3,3}^{2,t})\}$	$3^{2\times2}$
⋮	⋮	⋮
1	$\{(N_{1,1}^{1,1},N_{1,1}^{2,1}),(N_{1,1}^{1,1},N_{1,2}^{2,1}),(N_{1,1}^{1,1},N_{1,3}^{2,1}),\cdots,(N_{1,3}^{1,1},N_{1,1}^{2,1}),(N_{1,3}^{1,1},N_{1,2}^{2,1}),(N_{1,3}^{1,1},N_{1,3}^{2,1})\}$	3^2

图 5　3 离散点 DDDP 方法一次迭代计算规模示意图

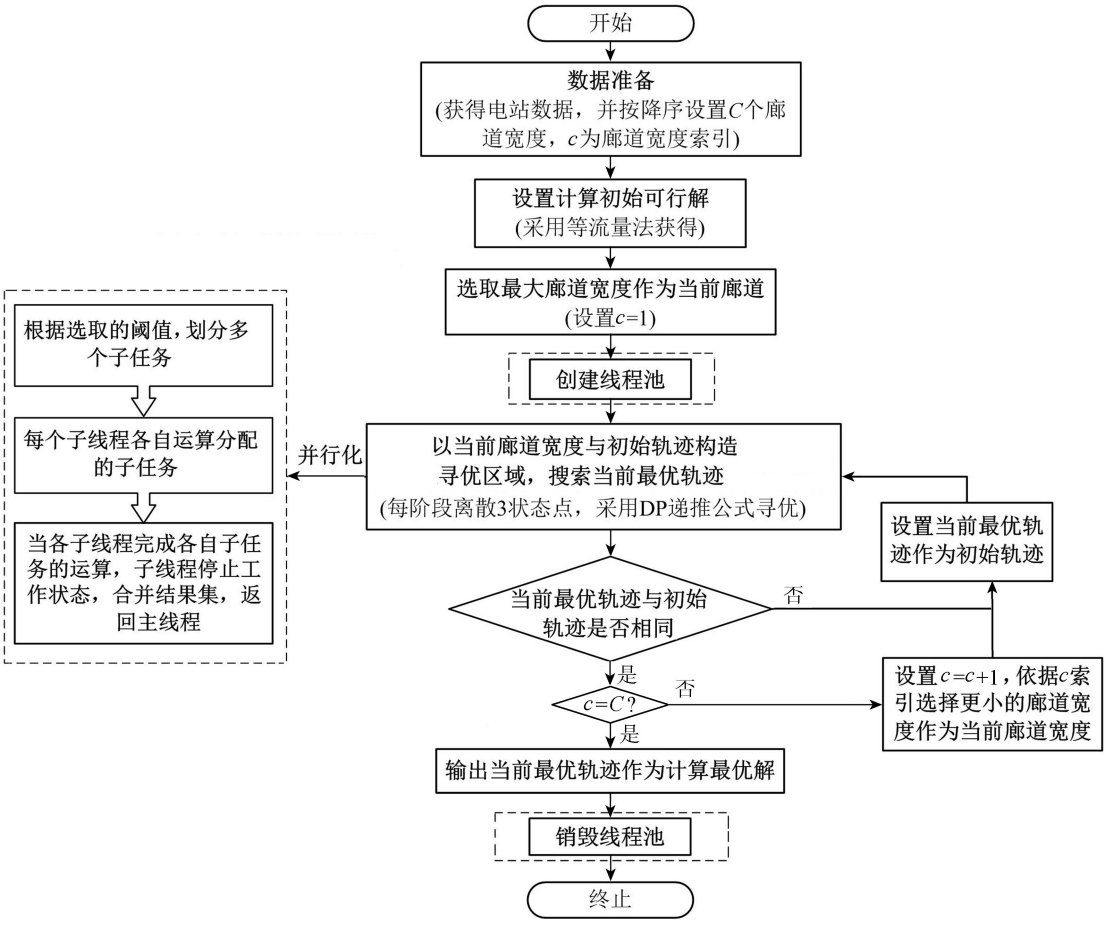


图 6　PDDDP 方法设计流程

设计时下述 5 个方面需要重点关注：

a. 在优化方法进入并行求解步骤前，应先创建线程池，并且程序设计时需避免线程池反复创建，导致资源浪费；在优化搜索结束后，应销毁线程池，避免不必要的资源占用。

b. 父任务的规模由水电系统的电站个数以及调度时段数决定，阈值需根据父任务规模及测试配置，采用式(1)计算得到。这种方式可使分解的子任务数等于配置的逻辑线程数，保证每个逻辑线程都处于计算状态，避免资源闲置。

c. 对每个多维组合状态按 1 ~ S_c 进行标序，利用 this. invokeAll( )方法接口实现任务递归分解，则父任务将分解为 $\left[1,\frac{S_c}{w}\right]$、$\left[\frac{S_c}{w}+1,\frac{2S_c}{w}\right]$、…、$\left[\frac{(w-2)S_c}{w}+1,\frac{(w-1)S_c}{w}\right]$、$\left[\frac{(w-1)S_c}{w}+1,S_c\right]$共 w 个子任务，其中$\left[\frac{S_c}{w}+1,\frac{2S_c}{w}\right]$表示子任务的计算区间，即每个子任务包含$\frac{S_c}{w}$个多维组合状态。当子任务划分结束时，框架将自动分配子任务到不同线程中进行计算。

d. 数据相关是并行程序设计过程中常出现的错误，为了避免这类错误产生，各工作线程执行所需的动态变量(如各状态组合下的计算水位、决策出力、出库流量等)需在各线程中单独定义，而静态变量(如水位-库容曲线、尾水位-泄流曲线等)则可从主线程中直接调用。

e. 各子线程的子任务执行结束后，需汇总各子任务的计算结果，即得到本次迭代过程中所有多维组合状态对应于递推公式的返回值，输出当前最优路径。

2　计算实例

2.1　测试方案

为了验证 Fork/Join 多核并行框架的计算效率，以红水河梯级水库群为调度对象，构建计算时段为月的库群长期发电量最大模型进行测试，并采用 PDDDP 方法求解。水库群由 10 座梯级电站组成，其中天生桥一级、龙滩、岩滩为长期调节性能电站，参与优化迭代计算，其他为日调节或径流式电站。水库群长期发电量最大模型较为常见，目标函数及约束条件的数学表达式见文献[11-14]，流域梯级拓扑结构见图 7，各电站特性值见文献[15]。另外，为了测试

不同计算规模条件下的算法并行性能,采用3种不同径流序列情景方案进行仿真计算,各方案见表1。

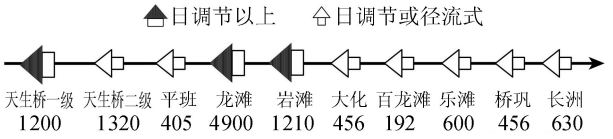


图7 流域梯级拓扑结构(单位:MW)

表1 测试方案

测试方案	径流序列	一次迭代状态组合规模数
方案1	50%水平年	7344
方案2	1981—2000年	173556
方案3	1951—2000年	435996

2.2 机器配置及并行指标

测试机器采用32核服务器,操作系统为Windows Server 2008,超线程状态关闭。衡量算法并行效率[16]的两个重要指标加速比 S_p 和效率 E_p 计算公式为

$$S_p = T_1/T_p \tag{3}$$

$$E_p = S_p/p \tag{4}$$

式中:T_1 为串行执行时间;T_p 为 p 核环境下的执行时间。

2.3 并行性能分析

表2为PDDDP方法在3个方案中不同并行环境下的计算耗时、加速比以及效率。

表2 各方案不同并行环境下的PDDDP方法测试结果

测试方案	并行核数	平均计算时间/s		加速比			效率/%	
		串行	并行	串行	并行	理想	串行	并行
方案1	1	16.8		1			100.0	
	2	16.8	10.1	1	1.66	2	50.0	83.2
	4	16.8	9.2	1	1.83	4	25.0	45.7
	8	16.8	13.2	1	1.27	8	12.5	15.9
	16	16.8	26.3	1	0.64	16	6.3	4.0
	32	16.8	38.9	1	0.43	32	3.1	1.3
方案2	1	417.3		1			100.0	
	2	417.3	240.0	1	1.74	2	50.0	86.9
	4	417.3	126.3	1	3.31	4	25.0	82.6
	8	417.3	88.8	1	4.70	8	12.5	58.7
	16	417.3	166.3	1	2.51	16	6.3	15.7
	32	417.3	333.9	1	1.25	32	3.1	3.9
方案3	1	1316.3		1			100.0	
	2	1316.3	740.0	1	1.78	2	50.0	88.9
	4	1316.3	373.7	1	3.52	4	25.0	88.0
	8	1316.3	202.9	1	6.49	8	12.5	81.1
	16	1316.3	188.4	1	6.99	16	6.3	43.7
	32	1316.3	323.6	1	4.07	32	3.1	12.7

在计算耗时上,PDDDP方法比串行方法明显减少。PDDDP方法最优加速分别缩减7.6 s、328.5 s和1127.9 s。但是,PDDDP方法各方案出现最优并行性能的测试环境有所区别,分别出现在4核、8核和16核。出现上述现象的原因在于在并行环境下随着计算核数的增加,线程管理消耗及线程间通信时间急剧上升。对于计算规模较小的方案,核数增加会导致计算效率明显下降,甚至计算耗时大于串行计算(如方案1并行环境核数大于8的测试结果)。因此,对于小规模计算任务的并行计算,当采用核数较多配置运算时,需要通过一定的测试判断最佳并行核数。

在加速比上,PDDDP方法最优加速比分别为1.83、4.70和6.99,PDDDP方法各方案最优加速比出现的测试环境与最优并行计算耗时相同。因此,随着计算任务规模的增加,并行计算的最优加速比逐步提高,并更接近理想加速比,能更好地发挥并行计算的优势,且在相同并行环境下,计算规模更大的任务能够获得更优的加速比(如在相同并行环境下,方案1、方案2、方案3的加速比逐步提升)。另外,加速比随着核数的增加并不能一直保持持续增长,会产生加速瓶颈,尤其计算规模较小的任务在核数较多的情况下,加速比无法持续增长,甚至随着核数增加导致加速比急剧下降(如方案1在4核环境下达到加速瓶颈,方案2、方案3则分别在8核、16核环境下达到)。

在效率上,随着测试环境核数的增加,效率逐渐下降,主要原因除了管理消耗以及通信时间的影响外,PDDDP方法仅在每次廊道迭代内寻优采用了并行设计,其他计算流程均为串行执行,而且为了避免数据同步错误的发生,在各工作线程中消耗了更多缓存来单独定义计算所需要的变量,势必影响算法总体的并行性能,降低计算效率。

总之,根据PDDDP方法测试结果,对于计算规模较小的任务,核数较少的配置即可获得较好的加速性能,核数较多易产生并行瓶颈,影响计算效率;对于计算规模较大的任务,核数较多可以显著提高加速性能。因此,在生产实际中,小规模计算任务的并行计算建议采用核数较少的配置,节省硬件投资;大规模计算任务的并行计算建议采用核数较多的配置,可充分发挥并行计算的优势。

2.4 Fork/Join 阈值分析

阈值的选择是影响Fork/Join并行框架充分发挥并行性能的关键因素之一。阈值设置在Fork/Join并行框架中作为一个开放接口,开发人员可自由对其进行设置。为了测试不同阈值对PDDDP方法的影响,以方案3为例,量化不同的阈值进行测试。根据式(2),其计算规模为435996,不同量化阈值条件下的任务划分示意图见图8。各量化阈值条件下并行测试结果见表3。

在表3以及图8中,不同阈值的设置在PDDDP方法中展现了不同的并行性能。当阈值设置为

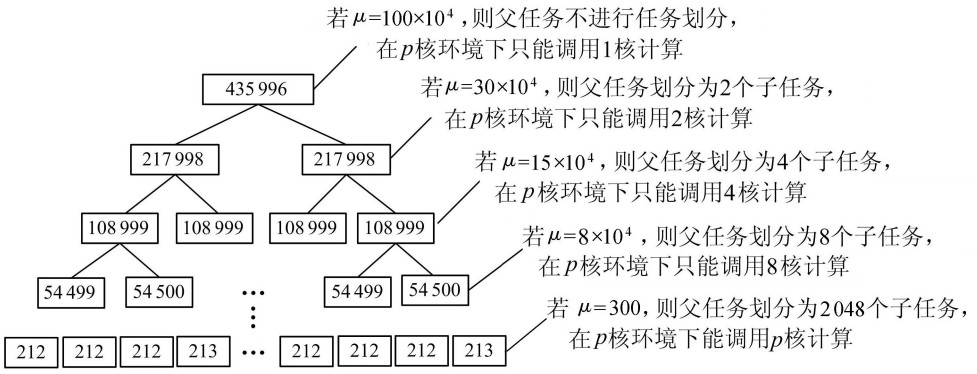


图 8　不同量化阈值条件下方案 3 任务划分示意图

表 3　各量化阈值条件下 PDDDP 方法测试结果

核数	不同量化阈值条件下计算耗时/s						
	300	2×10^4	8×10^4	15×10^4	30×10^4	100×10^4	式(1)计算阈值
2	816	774	763	724	752	1 332	740
4	429	378	385	367	729	1 309	374
8	263	195	211	381	734	1 324	203
16	235	183	198	379	748	1 319	188
32	412	181	209	369	743	1 296	324

注：测试结果为单次测试耗时；100×10^4 条件下只能调用 1 核计算，实质为串行计算。

100×10^4 时，阈值大于计算任务规模 435 996，计算任务不进行子任务分割，在任一核环境下仅能调用 1 核运算，计算耗时与串行计算近似相同，并未发挥并行算法优势。当阈值设置为 30×10^4 时，计算任务进行了 1 次子任务分割即划分为 2 个子任务，因此，在 2 核环境下 2 核得到充分利用，发挥了并行计算的效果，但是，在 4 核环境下仍有 2 核处于闲置状态，并未充分发挥并行性能，仅当阈值小于或等于 15×10^4 时，4 核环境下的所有核心才均得到充分利用。在 8 核、16 核、32 核环境下同样可观察到核心闲置现象。然而，当阈值设置为 300 时，划分的子任务过多，计算耗时比采用式(1)确定的阈值均有所增加。因此，在不同核数环境下，展现最优并行性能的阈值并不相同，需通过测试得到。

考虑到不同计算规模、不同并行环境下，最优阈值均不相同且难以确定，需研究发挥最优并行性能的阈值设置公式。本文采用式(1)确定测试阈值，即划分的任务数等于核数，保证每个核心均可分配到 1 个子任务，可自适应不同规模、不同核环境下的并行任务，具有一定的通用性和实用性，但是该公式确定的阈值并不一定能获得最佳并行性能。考虑到并行方法易产生瓶颈效应(当核数为 32 时，3 个 PDDDP 测试方案的加速比均不为最优)，可采用下式进一步改进式(1)，即调整父任务划分的子任务数目，使并行算法在最优核数环境下执行运算，充分发挥并行性能：

$$\mu = \left\lceil \frac{S_c}{w} \right\rceil \times 2^k \tag{5}$$

式中 k 为递归增减系数，取整数。当 $k=0$ 时，即为式(1)；当 $k>0$ 时，表示在式(1)划分的子任务数基础上，递归减少任务数。例如 $k=1$、32 核环境下，PDDDP 方法可把父任务划分为 16 个子任务，即在 32 核环境下，也只能调用 16 核进行计算，方案 3 则可获得最优并行性能。以此类推，当 $k=2$ 时，父任务划分为 8 个子任务，仅可利用 8 核实现并行，则方案 2 可获得最优并行性能。另外，由于并行算法受计算规模的影响所展现的性能可能会发生变化，针对不同规模的计算任务，k 值是否需要调整均需要通过测试得到，若计算规模较大未出现并行瓶颈时，设置 $k=0$ 即可。

总之，阈值的设置对 Fork/Join 并行算法的影响非常明显，且设置方式较为灵活。采用式(5)设置并行方法的阈值是一种可供选择的途径。通过测试结果的分析，在已有多核配置的前提下，当配置的核数较少时，建议设置 $k=0$，可保证并行算法在获得较好并行性能的基础上，自适应规模不同的计算任务，充分利用多核资源；当配置的核数较多时，尤其对于规模较小的计算任务，需要对参数 k 进行调试，测试最优并行核数，避免在核数较多环境下无法有效发挥并行算法的优势。

3　结　论

a. 通过 Fork/Join 并行框架实现方法并行化求解有利于充分发挥多核处理器的优势，大幅度缩短计算时间，加速效果显著。在当前多核配置普及的条件下，并行计算是提高求解效率的可行途径。

b. 随着并行核数的增加，并行计算的效率会出现瓶颈。对于规模相对较小的计算任务，建议使用核数较少的硬件配置实现并行计算，减少不必要的硬件投资；对于规模相对较大的计算任务，建议使用核数较多的配置实现并行计算，可充分发挥并行方法优势。

c. Fork/Join 多核并行框架规模控制阈值的选择直接影响算法的并行效率,应针对不同规模的计算任务灵活设置阈值。

参考文献:

[1] LABADIE J W. Optimal operation of multireservoir systems: state-of-the-art review [J]. Journal of Water Resources Planning and Management, 2004, 130(2): 93-111.

[2] 陈立华,梅亚东,麻荣永. 并行遗传算法在雅砻江梯级水库群优化调度中的应用[J]. 水力发电学报,2010,29(6):66-70. (CHEN Lihua, MEI Yadong, MA Rongyong. Parallel genetic algorithm and its application to optimal operation of Yalong River cascade reservoirs [J]. Journal of Hydroelectric Engineering, 2010, 29(6): 66-70. (in Chinese))

[3] 李想,魏加华,傅旭东. 粗粒度并行遗传算法在水库调度问题中的应用[J]. 水力发电学报,2012,31(4):28-33. (LI Xiang, WEI Jiahua, FU Xudong. Application of coarse-grained genetic algorithm to reservoir operation [J]. Journal of Hydroelectric Engineering, 2012, 31(4): 28-33. (in Chinese))

[4] 张忠波,吴学春,张双虎,等. 并行动态规划和改进遗传算法在水库调度中的应用[J]. 水力发电学报,2014,33(4):21-27. (ZHANG Zhongbo, WU Xuechun, ZHANG Shuanghu, et al. Parallel dynamic programming and improved genetic algorithm and their application to reservoir operation[J]. Journal of Hydroelectric Engineering, 2014, 33(4): 21-27. (in Chinese))

[5] 王丽萍,孙平,蒋志强,等. 并行多维动态规划算法在梯级水库优化调度中的应用[J]. 水电能源科学,2015,33(4): 43-47. (WANG Liping, SUN Ping, JIANG Zhiqiang, et al. Application of parallel multi-dimensional dynamic programming algorithm in cascaded reservoirs optimal operation[J]. Water Resources and Power, 2015, 33(4): 43-47. (in Chinese))

[6] 蒋志强,纪昌明,孙平,等. 多层嵌套动态规划并行算法在梯级水库优化调度中的应用[J]. 中国农村水利水电,2014(9): 70-75. (JIANG Zhiqiang, JI Changming, SUN Ping, et al. The application of multilayer nested parallel dynamic programming algorithm in cascade reservoirs operation optimization [J]. China Rural Water and Hydropower, 2014(9): 70-75. (in Chinese))

[7] LEA D. A Java fork/join framework[C]//Proceedings of the ACM 2000 conference on Java Grande. New York: ACM Press, 2000: 36-43.

[8] 邹强,王学敏,李安强,等. 基于并行混沌量子粒子群算法的梯级水库群防洪优化调度研究[J]. 水利学报,2016,47(8):967-976. (ZOU Qiang, WANG Xuemin, LI Anqiang, et al. Optimal operation of flood control for cascade reservoirs based on parallel chaotic quantum particle swarm optimization [J]. Journal of Hydraulic Engineering, 2016, 47(8): 967-976. (in Chinese))

[9] 王森,武新宇,程春田,等. 梯级水电站群长期发电优化调度多核并行机会约束动态规划算法[J]. 中国电机工程学报,2015, 35(10): 2417-2427. (WANG Sen, WU Xinyu, CHENG Chuntian. Multi-core parallel chance constrained dynamic programming for long-term generation operation of cascaded reservoirs [J]. Proceedings of the CSEE, 2015, 35(10): 2417-2427. (in Chinese))

[10] HEIDARI M, CHOW V T, KOKOTOVI P V, et al. Discrete differential dynamic programming approach to water resources systems optimization [J]. Water Resources Research, 1971, 7(2): 273-282.

[11] 原文林,曲晓宁. 混沌蚁群优化算法在梯级水库发电优化调度中的应用研究[J]. 水力发电学报,2013,32(3): 47-54. (YUAN Wenlin, QU Xiaoning. Application study on chaos ant colony optimization for operation of cascade reservoir power generation [J]. Journal of Hydroelectric Engineering, 2013, 32(3): 47-54. (in Chinese))

[12] 王建群,贾洋洋,肖庆元. 狼群算法在水电站水库优化调度中的应用[J]. 水利水电科技进展,2015,35(3): 1-4. (WANG Jianqun, JIA Yangyang, XIAO Qingyuan. Application of wolf pack search algorithm to optimal operation of hydropower station [J]. Advances in Science and Technology of Water Resources, 2015, 35(3): 1-4. (in Chinese))

[13] 徐刚,夏甜. 基于改进与优化调度图的梯级电站联合调度[J]. 水利水电科技进展,2014,34(3): 44-49. (XU Gang, XIA Tian. Joint dispatching of cascade hydropower station based on improved and optimized reservoir operation chart [J]. Advances in Science and Technology of Water Resources, 2014, 34(3): 44-49. (in Chinese))

[14] 刘攀,赵静飞,李立平,等. 水库优化调度中的异轨同效问题[J]. 水利水电科技进展,2013,33(2): 5-8. (LIU Pan, ZHAO Jingfei, LI Liping, et al. Equivalence of reservoir optimal operation [J]. Advances in Science and Technology of Water Resources, 2013, 33(2): 5-8. (in Chinese))

[15] 王森,程春田,武新宇,等. 自适应混沌整体退火遗传算法在水电站群优化调度中的应用[J]. 水力发电学报,2014,33(5): 63-71. (WANG Sen, CHENG Chuntian, WU Xinyu, et al. Application of self-adaptive chaos whole annealing genetic algorithm to optimal operation of hydropower station groups [J]. Journal of Hydroelectric Engineering, 2014, 33(5): 63-71. (in Chinese))

[16] 王丽萍,孙平,蒋志强,等. 基于并行云变异蛙跳算法的梯级水库优化调度研究[J]. 系统工程理论与实践,2015, 35(3): 790-798. (WANG Liping, SUN Ping, JIANG Zhiqiang, et al. Study on cascade reservoirs optimal operation based on parallel normal cloud mutation shuffled frog leaping algorithm[J]. Systems Engineering: Theory & Practice, 2015, 35(3): 790-798. (in Chinese))

(收稿日期:2016-02-12 编辑:熊水斌)