有向图并行计算中一种新的结点调度算法

张爱清 莫则尧
(北京应用物理与计算数学研究所高性能计算中心 北京 100094)

摘 要 在基于有向图的并行计算中,给定图剖分后,如何设计结点调度方案使得并行执行时间最短,是典型的NP完全问题.针对此问题,文中提出一种新的基于顺逆交替迭代技术的启发式调度算法,并给出该算法的并行实现.严格的理论推导证明,新算法在一定的假设条件下,从任何初始调度方案出发,均可以单调收敛.在数百个处理器上的并行数值实验表明,与常用的调度算法相比,新算法可在付出很少的开销代价下显著提高整体并行效率.

关键词 有向图;并行计算;结点调度算法;顺逆交替迭代技术
中图法分类号 TP301 **DOI 号**: 10.3724/SP.J.1016.2009.02178

A New Scheduling Algorithm for Digraph-Based Parallel Computing

ZHANG Ai-Qing MO Ze-Yao
(*High Performance Computing Center, Institution of Applied Physics and Computational Mathematics, Beijing* 100094)

Abstract The scheduling algorithm is crucial for the efficiency of the digraph-based parallel computing. However, it is classically NP-complete to find a schedule with the shortest duration for a given digraph partition. In this paper, a new heuristic algorithm is presented using the well known forward-backward iterations. It is proved that this new algorithm is convergent locally under some conditions. Furthermore, the parallel implementation of the algorithm is given in detail. On hundreds processors, the benchmark results suggest that the new algorithm outperforms many scheduling algorithms widely used now.

Keywords digraph; parallel computing; scheduling algorithm; forward-backward iterations

1 引 言

在基于网格离散的科学计算应用中存在一类计算方法,它们的并行计算可以等价地转换为有向图的并行计算[1]:有向图中的结点表示计算方法中网格单元上的计算任务,结点权重表示计算开销,图中的弧表示任务间的数据依赖关系,弧权重表示数据通信开销.这类方法普遍存在于科学计算领域.例如,在核科学和惯性约束聚变数值模拟应用中,求解多群粒子输运方程的离散纵标数值方法[2-3]以及在流体力学中,求解对流占优方程的隐式迎风格式、顺流松弛方法[4-6]等,均是此类方法.针对有向图设计的并行算法,可以直接应用到这类方法的并行计算中.

在有向图中,一方面,弧对应的数据依赖关系表示弧终点必须在弧始点计算完毕之后才能开始计算,并且需要引用弧始点的最新计算结果;另一方面,没有数据依赖关系的结点可以并行计算.有向图并行计算的核心是如何设计并行算法,充分挖掘有向图中隐藏的内在并行度,缩短有向图并行计算时间.其中的关键是有向图剖分算法以及剖分后结点的调度算法.本文围绕后者展开研究工作.

收稿日期:2006-09-06;最终修改稿收到日期:2009-06-17.本课题得到国家"九七三"重点基础研究发展规划项目基金(2005CB321702)、国家自然科学基金(60533020,60603050,90718029)资助.**张爱清**,女,1976 年生,博士,副研究员,主要从事大规模科学与工程计算中并行算法与并行应用软件的研究. E-mail: zhang_aiqing@iapcm.ac.cn. **莫则尧**,男,1971 年生,博士,研究员,主要从事大规模科学与工程计算中并行算法与并行应用软件的研究. E-mail: zeyao_mo@iapcm.ac.cn.

假设有向图已经被剖分成多个互不重叠的子图,分配给不同的处理器.此时,调度算法研究的是各处理器如何安排结点的执行次序,使得在满足依赖关系的前提下并行执行时间最短.事实上,该问题可以归结为一类资源受限的任务调度问题,称之为RCPSP问题.RCPSP是最早被证明的NP完全问题之一[7],研究历史悠久,拥有丰富的启发式算法.其中,基于优先级策略的调度算法最为常用,它由两部分组成:调度生成方法和优先级策略.调度生成方法根据各个结点的优先级,安排每个结点的执行时间.常见的有S-SGS方法和P-SGS方法[8],已经成熟.优先级策略则确定各个结点的优先级,用于多个可执行结点之间的调度决策.优先级策略是调度算法的核心,对调度方案的并行执行时间起决定性作用.当前,优先级策略分为两类.一类是通用型策略,如最晚完成时间策略[9]、最晚开始时间策略[10]、最小松弛时间策略[9]、最多后继结点数策略[11]、多优先级策略[12-13]、采样策略[14-15]、Forward-Backward策略[16]、DFDS策略[17]、最短内部边界路径策略[1]等.另一类是适合具体应用问题的有向图特征的优先级策略,如针对粒子输运方程多方向输运扫描的KBA优先策略[18]、random delay策略[19]等.在这些基于优先级的调度算法中,Forward-Backward算法,简称FB算法,是当前运筹学界公认的最简单有效的调度算法之一.

除基于优先级策略的调度算法外,后启发式调度算法是当前流行的另一类调度算法,如模拟褪火法、禁忌搜索法、遗传算法等[8,20],但它们的计算复杂度较高,不适用于大型有向图,故不在本文考虑之列.

在基于网格离散的大规模科学计算应用问题中,网格的规模庞大,通常被分布存储在多个处理器中.此时,自然的做法是各个处理器并行地形成并分布存储有向图的子图.于是,调度算法必须被并行地实现,以避免聚集子图所需的昂贵的全局通信开销.然而,当前文献上提出的调度生成方法(S-SGS方法和P-SGS方法),均是串行算法.这限制了多种调度算法(如FB算法)在这类并行计算中的应用.

围绕大规模有向图并行计算,本文开展以下工作:(1)提出一种新的启发式调度算法.新算法基于顺逆交替迭代技术,在原有FB算法的基础上,采用了一种新的更为有效的优先级策略.(2)给出新调度算法的收敛性证明,为新算法提供一定的理论保障.(3)针对分布存储的有向图,给出新算法的并行实现.(4)以粒子输运并行计算中的调度问题为例,在数百个处理器上,将新算法和最晚开始时间策略、DFHDS策略、最短内部边界路径策略以及FB算法进行对比.测试数据说明,新算法可在付出很少的开销代价下显著提高整体并行效率.

本文第2节详细介绍有向图并行计算中的调度模型;第3节给出新的调度算法;第4节给出新算法的局部收敛性证明;第5节给出新算法的并行实现;第6节为数值实验;第7节总结全文.

2　调度模型

假设带权有向图 $\mathcal{G}=(\mathcal{V},\mathcal{E},\boldsymbol{p},\boldsymbol{W})$ 由结点集 $\mathcal{V}=\{1,2,\cdots,n\}$ 和弧集 $\mathcal{E}=\{(i_1,j_1),\cdots,(i_m,j_m)\}$ 构成.非负向量 $\boldsymbol{p}$ 称为结点权重向量,元素 p_i 代表结点 i 的时间开销;非负矩阵 $\boldsymbol{W}$ 称为弧权重矩阵,元素 $w_{k,j}$ 表示弧 (k,j) 的时间开销.如果弧 $(k,j)\in\mathcal{E}$,则称该弧与结点 k,j 相关联;并称结点 k 是结点 j 的直接前驱,结点 j 是结点 k 的直接后继.没有直接前驱的结点称为源点;没有直接后继的结点称为汇点.定义有向路径为一组结点序列 $(u_1,u_2,\cdots,u_s)$,满足 $(u_i,u_{i+1})\in\mathcal{E}$,任意 $1\leq i<s$.如果 $u_1=u_s$,则称此路径为圈.不存在圈的有向图称为无圈有向图.如果图 $\mathcal{G}$ 中存在从结点 k 到结点 j 的有向路径,则称结点 k 是结点 j 的前驱,结点 j 是结点 k 的后继.路径长度指路径上所有结点和弧的权重总和.关键路径指图中的最长有向路径.

有向图 $\mathcal{G}$ 的最短可能执行时间等于关键路径的长度.图中结点时间权重总和与关键路径长度的比值称为理想加速比.理想加速比刻画了有向图本身的内在并行度.

并行计算中,有向图 $\mathcal{G}$ 被剖分成 P 个互不重叠的子图,分配给 P 个处理器.记 $\boldsymbol{r}$ 为结点映射向量,元素 $1\leq r_i\leq P$ 表示结点 i 所归属的处理器.称处理器 r_i 是结点 i 的属主,结点 i 是处理器 r_i 的本地结点.通过弧与本地结点关联的非本地结点称为影像结点.根据两端结点的分配情况,弧可分为两类:

(1)截弧.弧的两端结点被分配给不同的处理器;

(2)内部弧.弧的两端结点分配在同一处理器内.

一般地,内部弧的权重总是设为0;而截弧的权重代表通信开销,总是大于或等于0.

对于给定的有向图,在图剖分确定情形下,如何

安排结点的执行次序,使得并行执行时间最短,可以归结为如下形式的结点调度问题:

$$\begin{aligned}&\min[\boldsymbol{f}]\\&\text{subject to}\\&f_i+w_{i,j}\leq f_j-p_j,\ i\in\mathcal{D}_j,\ j=1,2,\cdots,n\\&|\mathcal{A}_k(t)|\leq 1,\ k=1,2,\cdots,P,\ \forall t\end{aligned}\tag{1}$$

其中,向量 $\boldsymbol{f}=(f_1,f_2,\cdots,f_n)$ 称为完成时间向量,或称结点调度方案;记号 $[\boldsymbol{f}]=\max\limits_{1\leq j\leq n}\{f_j\}-\min\limits_{1\leq j\leq n}\{f_j-p_j\}$ 表示方案的执行时间,即方案中从起始结点开始计算到终止结点完成计算所经历的时间长度;$\mathcal{D}_j$ 为结点 j 的直接前驱集合;$\mathcal{A}_k(t)=\{j:f_j-p_j\leq t<f_j\}\bigcap\{j;r_j=k\}$,指处理器 k 安排在 t 时刻执行的结点集合.

问题(1)的第 1 个约束条件称为顺序限制,指后继结点必须在前驱结点计算完成,并且收到前驱结点的计算结果之后,才能开始计算;第 2 个约束条件表示资源限制,即任何时刻任何处理器一次只能执行一个结点.满足以上两个约束条件的调度方案称为可行调度方案.执行时间最短的可行调度方案称为最优调度方案.

定义调度方案的加速比为:结点时间权重总和与方案的执行时间的比值.显然,可行调度方案的加速比≤min{理想加速比,P}.

易知,问题(1)存在解的充要条件是:有向图 $\mathcal{G}$ 中不存在圈.本文中,我们总是假设有向图是无圈的.

问题(1)是个 NP 完全问题,在实际应用中必须采用启发式调度算法来构造次优的可行调度方案.于是调度算法需从两个方面来评价:(1)所构造的调度方案执行时间越短,则算法越优;(2)算法本身的计算复杂度越低,则越优.但通常两者是矛盾的.综合而言,当调度方案的执行时间加上调度算法本身的执行时间越少,算法才是越优的.

3 新的顺逆交替迭代调度算法

给定结点优先值后,调度生成方法可用两种次序构造调度方案:顺序或逆序.以 S-SGS 方法[8]为例,顺序 S-SGS 方法从源点开始,顺着弧的方向逐个遍历结点,尽早地安排每个结点的执行时间.逆序 S-SGS 方法则从汇点开始,逆着弧的方向逐个遍历结点,在给定的时间期限前,尽晚地安排每个结点的执行时间.

顺逆交替调度是一种有效的调度技术.该技术以迭代的形式,不断更新结点优先级,以及交替采用顺序和逆序调度生成方法,以求生成一序列逐步改进的调度方案.该技术的特点是每次更新优先级时,总是利用前一反序构造的旧方案获取启发式信息.Li 等人于 1992 年提出第一个基于顺逆交替技术的调度算法,称为 FB 算法[16].FB 算法采用前个方案中结点的完成时间作为优先值,然后交替调用顺序和逆序 S-SGS 方法更新调度方案.文献[21]则基于另一种比较复杂的优先级策略 LCBA,给出另一个顺逆交替迭代算法.以下为方便描述,总假设初始调度方案通过顺序调度得到,并记为 $\boldsymbol{f}^0$;第 h 个迭代过程中,先后通过逆序调度和顺序调度得到的方案分别记为 $\boldsymbol{f}^{h-1/2}$ 和 $\boldsymbol{f}^h$.

本文基于顺逆交替迭代技术,提出一种新的优先级策略,称为截弧优先策略.处理器的空闲/忙碌状态极大程度取决于截弧所代表的通信能否及时完成,因此截弧的执行时间至关重要.原则上,如果截弧的后继计算越多,则该截弧及其相关前驱结点应越先被执行.而逆序调度所得方案中截弧的最晚允许开始时间,是对截弧后继计算量的一种有效估计.鉴于此,针对顺序调度(基于旧方案 $\boldsymbol{f}^{h-1/2}$ 构造新方案 $\boldsymbol{f}^h$ 时),本文提出如下截弧优先策略:旧方案中结点 i 所有后继截弧的最晚允许开始时间记为 α_i,α_i 值越小的结点优先级越高;若 α_i 值相等,则记结点 i 在旧方案中的开始时间 $s_i^{h-1/2}=f_i^{h-1/2}-p_i$,$s_i^{h-1/2}$ 越小的结点优先级越高.

α_i 的计算公式如下:

$$\alpha_i=\min\{\{s_j^{h-1/2}-w_{k,j}:(k,j)\in\mathcal{SC}_i\}\bigcup\{+\infty\}\}\tag{2}$$

其中 $\mathcal{SC}_i$ 指与结点 i 或结点 i 的本地后继结点相关联的截弧的集合;$s_j^{h-1/2}-w_{k,j}$ 是截弧 (k,j) 在旧方案中的最晚允许开始时间.特别地,根据式(2),如果 $\mathcal{SC}_i$ 为空,则该结点将被赋于最低优先级.这是因为此结点不影响非本地处理器的计算.

类似地,逆序调度时,如果截弧的前驱计算越多,则该截弧的后继结点应优先安排在越晚的时间段里执行.记方案 $\boldsymbol{f}^{h-1}$ 中结点 i 所有前驱截弧的最早可能完成时间为 β_i.于是,在逆序调度构造方案 $\boldsymbol{f}^{h-1/2}$ 时,截弧优先策略为:β_i 值越大的结点优先级越高;若 β_i 值相同,则 f_i^{h-1} 越大的结点优先级越高.其中 β_i 的计算公式如下:

$$\beta_i=\max\{\{f_j^{h-1}+w_{j,k}:(j,k)\in\mathcal{DC}_i\}\bigcup\{-\infty\}\}\tag{3}$$

这里,$\mathcal{DC}_i$ 指与结点 i 或结点 i 的本地前驱结点相关

联的截弧的集合;$f_j^{h-1}+w_{j,k}$表示截弧(j,k)在旧方案$\boldsymbol{f}^{h-1}$中的最早可能完成时间.

基于上述截弧优先策略的顺逆交替迭代算法称为CAP-FB算法.该算法接收一个初始调度方案$\boldsymbol{f}^0$,然后以迭代的形式逐步改进调度方案.每次迭代分4步,具体如图1所示.迭代的终止条件为$|[\boldsymbol{f}^{h-1/2}]-[\boldsymbol{f}^h]|<$阈值$\varepsilon$或迭代次数$h$等于最大允许迭代次数.

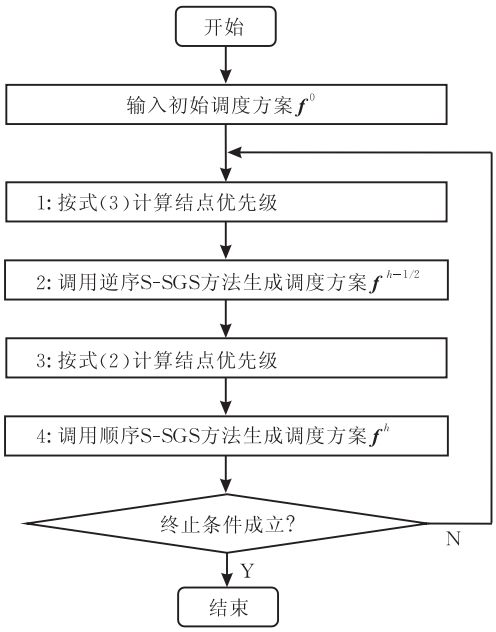
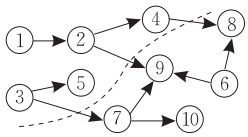


图1　CAP-FB算法流程图

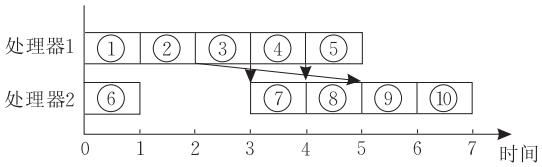
图2是CAP-FB算法求解过程的一个简单实例.对于图2(a)和图2(b)所示的调度问题和初始调度方案,CAP-FB算法通过一次顺逆迭代,将调度方案调整到最优.

对图2中的问题以及相同的初始调度方案,Li的FB算法不能产生任何改进效果,不如CAP-FB算法.这是因为截弧优先策略从提高资源利用率的角度出发,优先处理被非本地结点依赖的结点,因此相较于FB算法无视资源利用率的结点完成时间策略,CAP-FB更能有效缩短调度方案的执行时间.

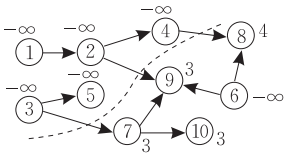
从单个迭代步的计算量上看,CAP-FB算法和FB算法的差别仅在优先级的计算上.在每个迭代步,CAP-FB算法需要根据式(2)和式(3)计算结点优先级,计算复杂度为$O(|\mathcal{V}|+|\mathcal{E}|)$,其中$|\mathcal{V}|$指结点数,$|\mathcal{E}|$指弧数;而FB算法直接采用前一方案的结点完成时间作为优先级,无此额外开销.从单个迭代步的计算复杂性上看,S-SGS方法的算法复杂度为$O(|\mathcal{V}|+|\mathcal{E}|)$,因此CAP-FB和FB算法有相同的算法复杂度$O(|\mathcal{V}|+|\mathcal{E}|)$.



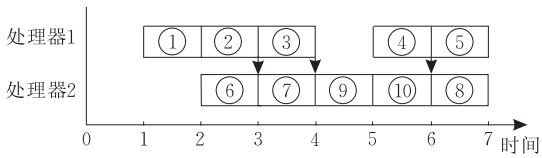
(a) 给定有向图,沿虚线分成2个子图.假设图中结点权重均为1,弧权重均为0



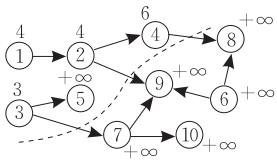
(b) 初始调度方案,执行时间为7



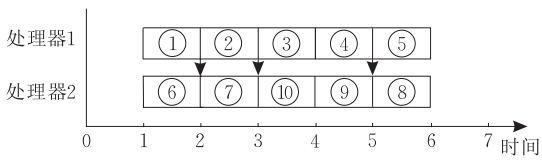
(c) 基于逆序调度的截弧优先策略和图2(b)所示的调度方案,每个结点的β优先值如图所示



(d) 根据1(c)所示优先级,执行逆序调度获得的新调度方案,执行时间为6



(e) 基于顺序调度的截弧优先策略和图2(d)所示的调度方案,每个结点的α优先值如图所示



(f) 根据1(e)所示优先级,执行顺序调度获得新调度方案,执行时间为5

图2　CAP-FB求解过程示例

4　算法收敛性分析

目前为止,尚无文献对基于顺逆交替迭代技术的调度算法证明其收敛性.这里,我们针对所有结点

等权重且所有弧权重为 0 情形下的调度问题(1),证明 CAP-FB 算法具有局部收敛性. 为方便起见,以下称上述情形下的调度问题为简单调度问题.

引理 1. 对简单调度问题,$[f^h]\leq[f^{h-1/2}]$,$h=1,2\cdots$. 其中,f^h 如图 1 所示,是 CAP-FB 算法基于调度方案 $f^{h-1/2}$ 执行步 3 和步 4 后得到的新调度方案.

证明. 记带权有向图 $\mathcal{G}=(\mathcal{V},\mathcal{E},\boldsymbol{p},\boldsymbol{W})$. 根据定理条件,取结点权重 $p_i=1$, $\forall i\in\mathcal{V}$,弧权重 $w_{i,j}=0$, $\forall(i,j)\in\mathcal{E}$. 引入记号 $|\mathcal{H}|=\sum_{i\in\mathcal{H}}p_i$. 假设新、旧调度方案 f^h 和 $f^{h-1/2}$ 的开始时间均为 0.

根据式(2)易知:若结点 j 是 k 的前驱,则优先值 $\alpha_j\leq\alpha_k$;更进一步,若结点 j 是 k 的非本地前驱,则

$$\alpha_j\leq s_k^{h-1/2}<f_k^{h-1/2}\leq\alpha_k \tag{4}$$

将所有截弧的始点和图 $\mathcal{G}$ 的汇点按优先级从高到低排列,记所得序列为 $J=\{j_1,j_2,\cdots,j_L\}$. 记图 $\mathcal{G}$ 中与 j_c 属于同一处理器,且优先级不低于 j_c 的结点集合(含 j_c)为 $\mathcal{H}_c$. 下面对下标 c 以归纳法证明:

$$\exists j'\in\mathcal{H}_c,\ \text{s.t. } f_{j_c}^h\leq f_{j'}^{h-1/2},\ c=1,2,\cdots,L \tag{5}$$

$c=1$ 时:根据 CAP-FB 算法的截弧优先策略可知,$\forall i\in\mathcal{H}_1$,$\alpha_i=\alpha_{j_1}$ 且 $f_i^{h-1/2}\leq f_{j_1}^{h-1/2}$,因此在旧方案中,$j_1$ 是集合 $\mathcal{H}_1$ 中的最晚执行的结点,故 $|\mathcal{H}_1|\leq f_{j_1}^{h-1/2}$ 成立. 由反证法可知集合 $\mathcal{H}_1$ 不存在任何前驱结点,故在顺序 S-SGS 方法下,$f_{j_1}^h=|\mathcal{H}_1|$ 成立. 因此,$f_{j_1}^h\leq f_{j_1}^{h-1/2}$,式(5)成立.

$c<C$ 时:假设式(5)成立.

$c=C$ 时:设结点 j_C 属于处理器 p. 分两种情形证明命题成立.

情形 1:新方案在处理器 p 的 $[0,f_{j_C}^h)$ 时间段内只排有 $\mathcal{H}_C$ 中的结点,且没有空闲时段:此时,易知 $f_{j_C}^h=|\mathcal{H}_C|$. 取 $j'=\arg\max\{f_i^{h-1/2}:i\in\mathcal{H}_C\}$,显然 $f_{j'}^{h-1/2}\geq|\mathcal{H}_C|$. 故 $f_{j_C}^h\leq f_{j'}^{h-1/2}$,式(5)成立.

情形 2:情形 1 不成立,即新方案在处理器 p 的 $[0,f_{j_C}^h)$ 时间段内有空闲时间,或排有优先级比 j_C 低的结点:此时,记 $[t_1-1,t_1)$ 为其中最晚的一个空闲的或被低优先级结点占有的时段,记 $\mathcal{K}\subset\mathcal{H}_C$ 为安排在时间段 $[t_1,f_{j_C}^h)$ 内的结点集合. 显然,$\mathcal{K}$ 非空(至少包含结点 j_C)且 $f_{j_C}^h=t_1+|\mathcal{K}|$. 记集合 $\mathcal{K}$ 在旧方案中最早和最晚执行的结点分别为 k' 和 j'. 显然 $s_{k'}^{h-1/2}+|\mathcal{K}|\leq f_{j'}^{h-1/2}$. 故以下只需证明 $t_1\leq s_{k'}^{h-1/2}$ 即可证明 $f_{j_C}^h\leq f_{j'}^{h-1/2}$.

采用假设法可以证明:存在截弧 (j_b,k'),s.t. $f_{j_b}^h\geq t_1$(否则,可以推出 k' 的所有本地或非本地前驱结点在新方案中的完成时间不晚于 t_1-1,于是此时在顺序 S-SGS 方法下,时段 $[t_1-1,t_1)$ 将优先安排结点 k',而不是空闲或者安排低优先级结点,故矛盾). 由式(4)可得 $\alpha_{j_b}<\alpha_{k'}\leq\alpha_{j_C}$,故结点 j_b 的优先级低于 j_C,所以 $b<C$. 根据归纳假设,$\exists j''\in\mathcal{H}_b$, s.t. $f_{j_b}^h\leq f_{j''}^{h-1/2}$. 再由式(4)可得:$f_{j''}^{h-1/2}\leq\alpha_{j''}\leq\alpha_{j_b}\leq s_{k'}^{h-1/2}$. 故 $s_{k'}^{h-1/2}\geq f_{j_b}^h\geq t_1$. 因此,情形 2 下式(5)亦成立.

根据归纳法,对于任意截弧的始点或图 G 的汇点 j_c,式(5)成立. 设汇点 j_{c*} 是新方案中最晚完成的结点,即 $[f^h]=f_{j_{c*}}^h$. 由式(5)知,$\exists j'$, s.t. $[f^h]\leq f_{j'}^{h-1/2}\leq[f^{h-1/2}]$. 命题得证. 证毕.

同引理 1 的证明,可得以下引理.

引理 2. 对简单调度问题,$[f^{h-1/2}]\leq[f^{h-1}]$,$h=1,2,\cdots$. 其中 $f^{h-1/2}$ 如图 1 所示,是 CAP-FB 算法基于调度方案 f^{h-1} 执行步 1 和步 2 后得到的新调度方案.

由引理 1 和引理 2,易推出以下收敛性定理.

定理 1. 对于简单调度问题,任意给定初始调度方案 f^0,CAP-FB 算法所得序列 $\{[f^h],[f^{h+1/2}]\}_{h=0}^{\infty}$ 单调收敛.

5 调度算法的并行实现

在大规模科学计算应用问题中,各个处理器并行地形成并分布存储有向图. 为适应此情形,本节给出 CAP-FB 算法基于 MPI[22] 的并行实现.

对于图 1 中的步 1 和步 3,每个进程只需保存有影像结点在旧调度方案中的完成时间,即可独立并行计算. 因此 CAP-FB 算法并行实现的关键在于步 2 和步 4 中 S-SGS 方法的并行实现.

原始的 S-SGS 方法并不适合并行,很难保证串行和并行结果的一致性. 因此本文对步 2 和步 4 采用了另一种等价的方法:调用基于结点队列的 S-SGS 方法[8](以下简称 AL-S-SGS 方法),将所有结点按优先级从高到低的次序排列,逐个地安排结点调度时间.

算法 1 是步 4 的并行实现. 其中,每个进程的输入包含本地结点和影像结点的子图、影像结点的属主、本地结点的优先级;输出是本地结点和影像结点的完成时间. 算法 1 步 1 按截弧优先策略,将本地结点按优先级从高到低排列;步 3 严格按照结点优先次序执行计算,并在其中的步 3.1 及时接收非本地前驱结点的最新数据,在步 3.4 发送其它进程需要的本地最新数据.

算法 1.　CAP-Forward ($\mathcal{G}, \boldsymbol{r}, \boldsymbol{i}, \boldsymbol{\alpha}, \boldsymbol{f}^{h-1/2}, \boldsymbol{f}^{h}$).

输入:本地子图$\mathcal{G}$,剖分映射子向量$\boldsymbol{r}$,优先级子向量$\boldsymbol{\alpha}$;
旧的本地调度方案$\boldsymbol{f}^{h-1/2}$

输出:新的本的调度方案$\boldsymbol{f}^{h}$

1. 排列本地结点$\{i_g\}_{g=1,\cdots,m}$,使得$(\alpha_{i_1}, f_{i_1}^{h-1/2}-p_{i_1})<\cdots<(\alpha_{i_m}, f_{i_m}^{h-1/2}-p_{i_m})$;

2. $I=[0,+\infty)$;　　/* 初始空闲时间段 */

3. do $g=1, \cdots, m$;　　/* m 为本地结点个数 */

3.1. for(结点i_g的每个非本地直接前驱结点j)接收f_j^h;

3.2. 按以下公式确定结点完成时间:

$$f_{i_g}^h=\min\{t:\ t\geq \max_{(j,i_g)\in E}\{f_j^h+w_{j,i_g}\}\ [t,t+p_{i_g})\subseteq I\}+p_{i_g};$$

3.3. 更新空闲时间段集合$I=I-[f_{i_g}^h-p_{i_g}, f_{i_g}^h)$;

3.4. for(结点i_g的每个非本地直接后继结点j)
发送$f_{i_g}^h$给进程r_j;

enddo

易推知:在截弧优先策略下,按优先级排列的结点序具有拓扑性质.这一点不仅保证了算法 1 构造的解满足顺序约束关系,而且保证算法 1 在步 3.1 接收数据时不会陷入死锁.

由于顺序 AL-S-SGS 方法中,每个结点的调度时间只依赖于两类结点:一是队列中位置比该结点靠前的本地结点;二是该结点的非本地直接前驱结点.因此算法 1 可以保证串行与并行结果的一致性.

同样,图 1 的步 2 可基于逆序 AL-S-SGS 方法实现并行.至此,CAP-FB 算法的并行实现已完整.FB 算法可以采用相同的方法实现并行.

6　数值实验

本节的测试实验分为两类.第 1 类为模型测试,基于给定的调度问题,比较 CAP-FB 算法和其他调度算法所得调度方案的优劣.第 2 类为实际应用测试,将调度算法应用到粒子输运并行数值模拟程序中,验证调度算法的实际效果.

6.1　模型测试

模型测试设计了两种调度模型:

(1)简单调度模型:每个结点权重为 1.0,且每条截弧权重为 0.0;

(2)复杂调度模型:每个结点和每条截弧权重不等.

其中,第 1 种为理想模型,并且实际应用中多数需要采用此种模型.第 2 种则模拟了实际运行中可能发生的复杂情形.

测试采用的有向图 A 来源于基于二维离散网格(图 3)和 24 个离散方向的粒子输运扫描算法.该图共含 14 万个结点和 56 万条弧.图中每个结点对应一个网格单元上沿某个方向的通量计算任务.这些结点随网格二维区域分解平均分配给各处理器.对应于前述两种不同的权重设置,分别记基于有向图 A 的简单调度模型和复杂调度模型为 A-1 和A-2.以下测试中,模型 A-2 的结点截弧平均权重比值取为 3∶1.这两个调度模型有非常好的内在并行度:模型 A-1 理想加速比达 666;模型 A-2 理想加速比为 555.

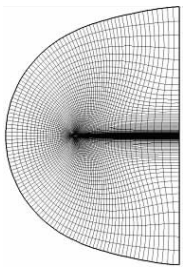


图 3　非规则网格,含 6000 个单元

最晚开始时间算法(简称 LST 算法)、DFHDS 算法、最短内部边界路径算法(简称 SBP 算法)是当前有向图并行计算中最常用的几种调度算法.本文分别以此 3 个调度算法所得的调度方案为 CAP-FB 算法的初始调度方案,对 CAP-FB 算法的调度效果进行比较.图 4 是对模型 A-1 在不同可使用处理器数目下,不同调度算法所得调度方案的加速比比较.从图 4 以及其它测试实验均可得出以下结论:CAP-FB 算法显著提高各种初始调度方案的质量,并且,其中以 SBP 算法和 CAP-FB 算法的组合为最优.以 500 个处理器数目下的模型 A-1 为例,从 SBP 算法所构造的加速比为 201 的调度方案出发,CAP-FB 算法将加速比提高到 302,效果显著.

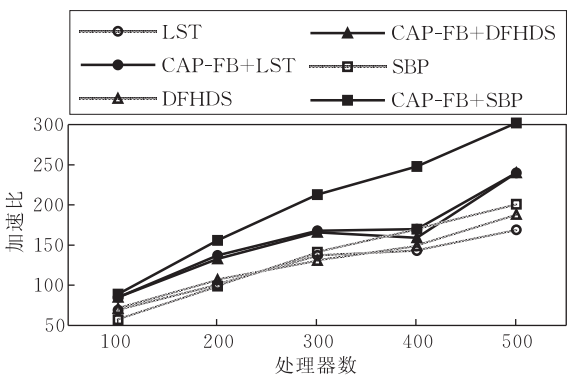


图 4　对调度问题 A-1,不同调度算法所得调度方案的加速比比较

图 5 和图 6 分别是对简单调度模型 A-1 和复杂调度模型 A-2,CAP-FB 算法和 Li 的 FB 算法所得调度方案加速比随迭代步的变化图.其中初始方案

通过最短内部边界路径调度算法构造.图中横坐标值 n 标志第 n 次顺序调度,$n.5$ 标志第 $n+1$ 次逆序调度.从两图中可以看出,CAP-FB 算法的调度能力略好于 FB 算法.如图 5 的 500CPUs 情形,FB 算法经 5 次迭代后,调度方案加速比收敛为 293;而 CAP-FB 算法仅 2 个迭代步得到的调度方案加速比为 295,最后在第 5 个迭代步加速比收敛于 303.

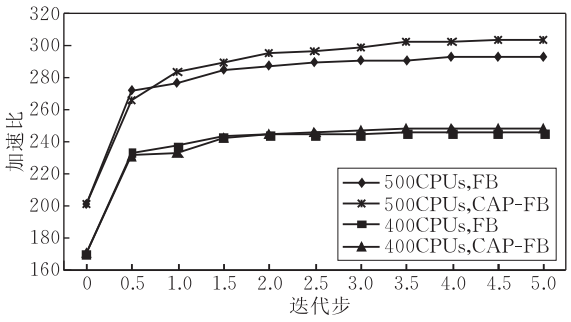


图 5 对简单调度问题 A-1,CAP-FB 算法和 Li 的 FB 算法的调度方案加速比变化情况

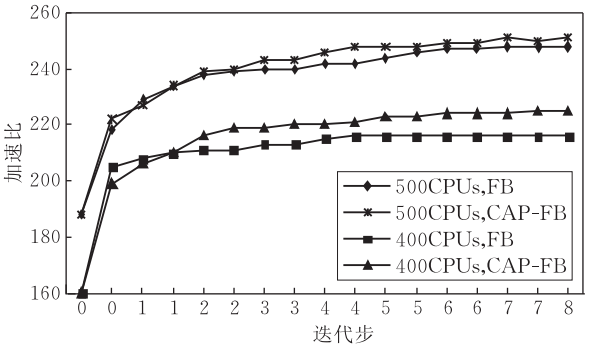


图 6 对调度问题 A-2,CAP-FB 算法和 Li 的 FB 算法的调度方案加速比变化情况

对于非等权重情形,数值实验表明(如图 6 所示),CAP-FB 算法在多个迭代步之后不能保持单调性,但仍可有效改进调度方案的质量.另外,如图 5 和图 6 所示,CAP-FB 算法和 FB 算法对调度质量的改进主要集中前 1 到 2 个迭代步.因此实际应用中,这两算法通常只需执行少量迭代步即可.

6.2 实际应用测试

为了验证实际应用效果,本文将分布式并行的 CAP-FB 算法耦合实现在中子输运并行程序[23]中,并在具体并行机上进行性能测试.测试采用的输运应用问题基于二维非结构离散网格,含 3600 个单元;离散扫描方向为 48 个,同簇方向间有依赖关系,不同簇方向不相关;离散能群为 24 个.对此应用测试问题,本文采用简单调度模型,记为模型 B.模型 B 中含 17 万个结点和近 85 万条弧,其理想加速比为 761.以下为区分起见,称简单调度模型的调度方案加速比为算法加速比;在具体并行机上实测得到的加速比称为实测加速比.

在某通信延迟约为 $10\mu s$ 的并行机上,本文采用无向图剖分软件 Chaco 提供的 IKL 方法剖分网格,然后分别调用最短内部边界路径法、Li 的 FB 算法和本文的 CAP-FB 算法,比较它们在具体应用中的表现.其中,Li 的 FB 算法和本文的 CAP-FB 均以最短内部边界路径法(SBP)作为初始方案构造算法,并且都只执行 2 个顺逆迭代步.在此 3 种调度算法下,中子输运模块执行 10 个时间步后的实测时间开销和实测加速比列于表 1 第 2~7 行.表 1 第 8~10 行则列出 3 种调度算法所得调度方案的算法加速比.如表 1 所示,在 CAP-FB 调度算法下,输运并行计算时间比在最短内部边界路径法下缩短了 9%~20%,比在 Li 的 FB 算法下少了 6%~8%.

表 1 中算法加速比和实测加速比有一定差异.其可能原因有多种,如 Cache 命中率的影响、有向图建模时权重设置的不精确等等.

表 1 不同调度算法下,中子输运并行模块的实测时间和实测加速比

CPU 数	执行时间/s			实测加速比			算法加速比		
	SBP	FB	CAP-FB	SBP	FB	CAP-FB	SBP	FB	CAP-FB
2	529.0	530.0	528.0	2.00	2.00	2.00	2.00	2.00	2.00
16	73.3	63.8	61.2	14.40	16.60	17.30	12.10	13.80	15.20
32	38.7	34.6	32.1	27.40	30.60	33.00	22.00	26.10	28.60
64	20.9	19.2	18.1	50.70	55.20	58.60	45.40	51.00	54.10
128	12.4	11.8	11.4	85.30	89.90	93.30	85.50	93.10	98.40

表 1 中的时间数据是包含了调度算法的整体时间开销.调度算法本身的开销在表 2 中单独列出.可以看出,CAP-FB 算法时间开销略多于 FB 算法,较多于 SBP 算法,不过远低于总计算时间,仅约占总时间的 2%.并且相较于该算法在调度方案上的大改进,其开销是值得的.

表 2 不同调度算法的时间开销

CPU 数	执行时间/s		
	SBP	FB	CAP-FB
2	1.80	11.20	12.10
16	0.30	1.20	1.30
32	0.09	0.56	0.60
64	0.03	0.24	0.29
128	0.01	0.19	0.24

7 结　论

本文面向有向图并行计算,在图剖分确定情形下,提出一种新的结点调度算法(CAP-FB).理论证明说明,CAP-FB 对简单调度问题具有局部收敛性.并且数百个处理器上的实验数据说明,只需少量几个迭代步,CAP-FB 算法就可大幅改进由其它调度算法(如最晚开始时间算法、DFHDS 算法和最短内部边界路径算法)构造的调度方案,并且改进效果优于 Li 的 FB 算法.

本文还给出基于结点队列的 S-SGS 调度生成算法的 MPI 并行实现,从而实现并行的 CAP-FB 算法和 FB 算法.这些并行调度算法耦合在粒子输运模拟程序中后,有效改进了数值模拟程序的并行性能.

参　考　文　献

[1] Mo Ze-Yao, Zhang Ai-Qing, Cao Xiao-Lin. Towards a parallel framework of grid-based numerical algorithms on DAGs// Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium. Greece, 2006

[2] Lewis E E, Miller W F. Computational Methods of Neutron Transport. New York: John Wiley & Sons Publisher, 1984

[3] Wareing T A, McGhee J M et al. Discontinuous finite element S_n methods on three-dimensional unstructured grids. Nuclear Science and Engineering, 2001, 138: 256-268

[4] Bey J, Wittum G. Downwind numbering: A robust multigrid method for convection diffusion problems on unstructured grids. Applied Numerical Mathematics, 1997, 23(1): 177-192

[5] Han H et al. Analysis of flow directed iterations. Journal of Computational Mathematics, 1992, 10(1): 57-76

[6] Hackbusch W, Probst T. Downwind Gauβ-Seidel smoothing for convection dominated problems. Numerical Linear Algebra with Applications, 1997, 4(2): 85-102

[7] Błůazewicz J, Lenstra J et al. Scheduling subject to resource constraints: Classification and complexity. Discrete Applied Mathematics, 1983, 5: 11-24

[8] Kolisch R, Hartmann S. Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis//Weglarz J. Project Scheduling: Recent Models, Algorithms and Applications. Boston: Kluwer Academic Publishers, 1999: 147-178

[9] Davis E, Patterson J. A comparison of heuristic and optimum solutions in resource-constrained project scheduling. Management Science, 1975, 21: 944-955

[10] Kolisch R. Project Scheduling Under Resource Constraints: Efficient Heuristics for Several Problem Classes. Berlin: Physica-Verlag, 1995

[11] Alvarez-Valdés R, Tamarit J M. Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis//Słowiński R, Węglarz J. Advances in project scheduling. Amsterdam: Elsevier, 1989: 113-134

[12] Boctor F. Some efficient multi-heuristic procedures for resource-constrained project scheduling. European Journal of Operational Research, 1990, 49: 3-13

[13] Thomas P, Salhi S. An investigation into the relationship of heuristic performance with network-resource characteristics. Journal of the Operational Research Society, 1997, 48(1): 34-43

[14] Cooper D. Heuristics for scheduling resource-constrained projects: An experimental investigation. Management Science, 1976, 22(11): 1186-1194

[15] Drexl A. Scheduling of project networks by job assignment. Management Science, 1991, 37(12): 1590-1602

[16] Li K Y, Willis R J. An iterative scheduling technique for resource-constrained project scheduling. European Journal CLF Operational Research, 1992, 56: 370-379

[17] Pautz S. An algorithm for parallel Sn sweeps on unstructured meshes. Nuclear Science and Engineering, 2002, 140: 111-136

[18] Koch K R, Baker R S, Alcouffe R E. A parallel algorithm for 3D S_n transport sweeps. Los Alamos National Laboratory: Technical Report LA-CP-92-406, 1992

[19] Anil Kumar V S et al. Provable algorithms for parallel sweep scheduling on unstructured meshes//Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium. Denver, 2005

[20] Kolisch R, Hartmann S. Experimental investigation of Heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research, 2006, 174(1): 23-37

[21] özdamar L, Ulusoy G. An iterative local constraint based analysis for solving the resource constrained project scheduling problem. Journal of Operations Management, 1996, 14(3): 193-208

[22] Mo Ze-Yao, Yuan Guo-Xing. Parallel Programming with MPI. Beijing: Science Press, 2001(in Chinese)
(莫则尧,袁国兴. 消息传递并行编程环境 MPI. 北京:科学出版社, 2001)

[23] Mo Ze-Yao, Fu Lian-Xiang. Parallel flux sweep algorithm for neutron transport on unstructured grid. The Journal of Supercomputing, 2004, 30(1): 5-17

ZHANG Ai-Qing, born in 1976, Ph. D., associate researcher. Her research interests include parallel algorithms and parallel application software for large scale scientific and engineering numerical simulations.

MO Ze-Yao, born in 1971, Ph. D., professor. His research interests include parallel algorithms and parallel application software for large scale scientific and engineering numerical simulations.

Backgound

This paper is supported by a grant from the National Key Basic Research Special Fund (2005CB321702), and the National Natural Science Foundation of China (Nos. 60533020, 60603050, 90718029). The projects mainly focus on the parallel algorithms for large scale scientific numerical simulations.

A closely related work of the research group is the development of a parallel framework of grid-based numerical algorithms where data dependencies between grid zones can be modeled by a directed acyclic graph. The parallel framework consists of three parts on how to partition, order and calculate the vertices of digraph. This paper focuses on the second part, and proposes a new scheduling algorithm to improve the parallel efficiency of the numerical algorithms based on the digraphs.