DOI:10.13196/j.cims.2014.05.tanwenan.1070.8.2014059

基于并行分层的工作流调度优化算法

谭文安[1,2],路广振[1+],孙 勇[1]
(1.南京航空航天大学 计算机科学与技术学院,江苏 南京 210016;
2.上海第二工业大学 计算机与信息学院,上海 201209)

摘 要:针对给定截止时间约束下用有向无环图描述的工作流时间费用优化问题,逆向分层算法未考虑工作流中各个任务在实际执行过程中的并行性而带来相对较多的时间碎片,提出一种基于并行分层的工作流调度算法——并发级别工作流调度算法。该算法将工作流在实际执行过程中的某个任务和其他一个(或具有依赖关系的相邻多个)并行执行的任务尽量划分到同一层,并根据各层的并行度分配冗余时间,对每层中存在具有依赖关系的多个相邻任务采用 Markov 决策过程算法进行时间费用优化。对平衡结构和非平衡结构的有向无环图所描述的工作流进行大量模拟实验,对比最小临界路径算法、逆向分层算法和期限顶级算法,实验结果表明所提算法具有较显著的优势。

关键词:并行分层;工作流调度;有向无环图;启发式;优化算法
中图分类号:TP393 **文献标识码**:A

Workflow scheduling optimization based on concurrent level

TAN Wen-an[1,2], *LU Guang-zhen*[1+], *SUN Yong*[1]
(1. School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China;
2. School of Computer and Information, Shanghai Second Polytechnic University, Shanghai 201209, China)

Abstract: Aiming at the workflow time cost optimization problem described by scheduling Directed Acyclic Graph (DAG) under deadline constraint, the Deadline Bottom Level (DBL) algorithm didn't considered the concurrence in the real executing process which caused much more shatter time. For this reason, a novel heuristics algorithm of Concurrent Level Workflow Scheduling (CLWS) was proposed. In this algorithm, some tasks of workflow in actual executive process were divided into the same level with the other concurrent tasks. The redundant time was distributed according to the concurrent degree of each level, and the multiple adjacent tasks with dependency relationship in each level were optimized by using Markov Decision Process (MDP) algorithm. Based on the workflow described by balanced structure and unbalanced structure of DAG, a number of simulation experiments were made to compare three algorithms of Minimum Critical Path (MCP), DBL and Deadline Top Level (DTL), and the results showed that CLWS algorithm had significant improvement.

Key words: concurrent level; workflow scheduling; directed acyclic graph; heuristics; optimization algorithm

0 引言

云计算、智能计算和面向服务的体系架构(Server Oriented Architecture,SOA)等新技术和新概念的广泛应用,有效促进了计算机支持与协同工作的实现[1-2]。工作流技术作为计算机完成协同工作的抽象化手段,在企业流水线生产、办公自动化和科研领域等方面扮演着重要的角色。工作流调度是

收稿日期:2013-01-16;**修订日期**:2013-04-15。Received 16 Jan. 2013;accepted 15 Apr. 2013.
基金项目:国家自然科学基金资助项目(61272036)。**Foundation item**: Project supported by the National Natural Science Foundation, China (No. 61272036).

工作流管理系统中的关键技术问题,目前工作流的服务资源从无偿服务转换为有偿服务,可根据工作流的服务质量(Quality of Service, QoS)特性,如服务的执行时间、费用和可靠性等,选择最优的服务资源来动态地满足不同用户的业务需求[3]。基于 QoS 的工作流调度算法是一个 NP-hard 问题[4]。工作流调度算法考虑服务的执行时间和费用两个因素,在工作流截止时间一定的条件下动态地选取任务中的优秀服务,从而使整个工作流的执行费用达到最小。因此,工作流调度问题是工作流关于时间与费用的优化问题。

目前使用模拟退火算法[5]、遗传算法[6]和混合粒子群优化算法[7]等实现工作流调度的方法存在算法时间开销比较大的问题。基于启发式工作流调度算法正逐步被关注,越来越多的基于启发式的时间费用优化算法[8-9]被提出。其中,基于工作流对应的有向无环图(Directed Acyclic Graph,DAG)结构和任务资源 QoS 特性进行分层的思想得到大家的推崇。Yu 等[9]提出基于截止时间约束的时间费用优化算法——期限顶级(Deadline Top Level,DTL)算法,根据工作流入口任务到其余任务的跳数(每个任务到其可达任务所经过的任务最大数)进行正向分层,将工作流截止时间划分为层截止时间,每层中各个任务的执行时间区间设定为对应层的层开始时间和层截止时间,在指定的时间区间内为各个任务选择相对较优的服务。

DTL 算法是一种相对简单且执行效率较高的启发式算法,但其存在一定不足:①不能合理地将具有同步执行特性的活动分配到同一层中;②将任务开始时间设置为其所在层的层开始时间,对任务的时间区间增加受限条件会产生较多的时间碎片。针对 DTL 算法的不足,Yuan 等[8]提出基于逆向分层(Deadline Bottom Level,DBL)的调度算法,该算法根据工作流中的各个任务到工作流出口任务的跳数进行逆向分层。

与 DTL 算法不同,DBL 算法根据 DAG 结构特性将每个任务的开始时间设定为其直接前驱任务的最大截止时间,扩大了每个任务的执行时间区间,从而优化工作流的总费用。DBL 虽能很好地解决 DTL 算法的上述不足,但也存在如下问题:①算法未能根据工作流在实际执行过程中任务间的并行执行情况进行分层,使得划分的层次较多,也会产生较多的时间碎片;②算法将冗余时间平均分配到各个层中,未能根据层间的逻辑结构和资源状况进行差异性分配;③当给定的工作流截止时间小于算法按照逆向分层后的工作流最小完成时间时,算法将不适用;④执行过程中各个任务的执行时间区间固定不变,不能根据实际执行情况动态调整。

受工作流分层思想的启发,本文提出一种新的启发式算法——并行分层工作流调度(Concurrent Level Workflow Scheduling,CLWS)算法。该算法能够针对工作流时间费用优化问题进行研究,根据工作流执行过程中的并行执行情况进行合理分层;根据每层的并行度分配冗余时间;使 CLWS 算法避免出现类似 DBL 算法不适用的情形;根据其直接前驱执行情况动态确定实际执行过程中任务的执行时间区间,尽可能地扩大其执行时间区间。

1　问题描述

通常,工作流可用 DAG 描述,DAG 的节点和弧分别表示工作流任务和任务间的关系。对于给定的工作流,令有向无环图 $G=\langle V,A\rangle$,其中:V 表示 DAG 的工作流任务节点集合,$V=\{1,2,\cdots,n\}$;A 表示 DAG 的有向弧集合,$A=\{\langle i,j\rangle \mid i\rightarrow j, i$ 是 j 的直接前驱,$i,\ j\in V\}$。$\forall\langle i,j\rangle\in A$ 表示 i 和 j 存在时间依赖关系。假定每个工作流都有唯一的入口节点 V_{entry} 和唯一的出口节点 V_{exit}。如果 DAG 中含有多个入口节点,则可增加一个虚入口节点,使之成为多个入口节点的直接前驱;同样,如果 DAG 中含有多个出口节点,则为这些出口节点增加一个直接后继作为该工作流的虚出口节点。如图 1 所示,图中入口节点为节点 1,出口节点为节点 4。

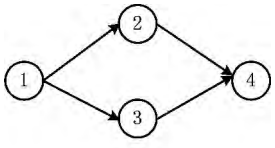

图1　简单的工作流示例图

在分布和异构网格环境下,不同服务提供者可提供相同类型的功能,但对应的 QoS 资源不同,因此完成同一个任务的不同服务需要的执行费用和时间可能不同[8]。为每个任务设定一个服务池(Service Pool,SP),用来管理对应该任务的所有服务 s。令任务节点 i 的服务池 $SP(i)=\{s_i^k=\langle t_i^k,c_i^k\rangle \mid 1\leqslant k\leqslant |SP(i)|\}$,其中:$|SP(i)|$ 表示该服务池中服务的个数,t_i^k 和 c_i^k 分别表示第 i 个节点第 k 个服务的执

行时间和费用。假设各个节点服务池是独立的，即相邻节点不会出现服务竞争。

基于时间费用的优化算法通常会考虑两个 QoS 因素，即服务的执行时间和费用（忽略任务间的通信时间和费用）。目前常见的基于 QoS 约束的工作流调度算法有基于费用约束的工作流调度算法和基于截止时间约束的工作流调度算法[10]，本文主要研究后者。后者是在给定的截止时间下，每个任务选择合适的服务，使总执行费用最少。

$$C_{\text{total}} = \min\Big\{ \sum_{\substack{i=1 \\ 1\leqslant k\leqslant |SP_i|}}^{n} c_i^k \Big\}。 \tag{1}$$

式中：i 表示任务编号，c_i^k 表示任务 i 选择其服务池中第 k 个服务的成本。

整个工作流能否正常执行，必须满足以下约束条件：

$$\beta_i + t_i^k \leqslant \beta_j； \tag{2}$$

$$\delta_n \leqslant t_{\text{deadline}}。 \tag{3}$$

式中：$\forall \langle i,j\rangle \in A$ 都需要满足式(2)，β_i 表示任务 i 的开始时间，δ_n 表示任务 n 的截止时间，t_i^k 表示任务 i 选择第 k（$1\leqslant k\leqslant |SP_i|$）个服务的执行时间，$t_{\text{deadline}}$ 表示工作流的截止时间。

通常在工作流各个任务的服务池中，可选服务的执行时间随着执行费用的增加呈离散非增变化[11]。时间费用优化算法的关键是尽量扩大每个任务的执行时间区间$[\beta,\delta]$，使每个任务在较大的时间区间内选择执行时间相对较大的服务，这样对应的执行费用相对较少。

2 算法描述

2.1 相关定义

定义 1 给定图 $G=\langle V,A\rangle$，为每个节点分配执行时间最短的服务，由关键路径法求得关键路径的总执行时间为当前 DAG 的总执行时间，记作 $t_{\min}$，也称作该工作流的下界截止时间。

定义 2 给定图 $G=\langle V,A\rangle$，为每个节点分配执行时间最长的服务，由关键路径法求得关键路径的总执行时间为当前 DAG 总的执行时间，记作 $t_{\max}$，也称作该工作流的上界截止时间。

定义 3 给定图 $G=\langle V,A\rangle$，工作流正常执行，其截止时间 t_{deadline} 需满足约束条件

$$t_{\min} \leqslant t_{\text{deadline}}。 \tag{4}$$

定义 4 给定图 $G=\langle V,A\rangle$，为每个节点分配执行时间最短的服务，求出此时的关键路径，用 CP 表示关键路径节点的集合，每个任务的实际开始执行时间为其最早开始执行时间，将与关键节点并行执行的一个（或具有依赖关系的相邻多个）节点划分到同一层，使每层中关键节点的数目尽量最少，并且层中任意节点的执行时间区间都在该层关键路径的执行时间区间范围内，如式(5)所示。将工作流的并行分层记作 CL，$|CL|$ 表示工作流的总层数。

$$\begin{cases} \beta_{CL_i} = \min\limits_{\substack{V_k\in CL_i\cap CP \\ 1\leqslant k\leqslant |CL_i\cap CP|}} \{\beta_{V_k}\}； \\ \delta_{CL_i} = \max\limits_{\substack{V_k\in CL_i\cap CP \\ 1\leqslant k\leqslant |CL_i\cap CP|}} \{\delta_{V_k}\}； \\ \beta_{V_i} \geqslant \beta_{CL_i}, \forall V_i \in CL_i； \\ \delta_{V_i} \leqslant \delta_{CL_i}, \forall V_i \in CL_i。 \end{cases} \tag{5}$$

式中：CL_i 表示第 i 层节点的集合，β_{CL_i} 表示第 i 层的开始时间，δ_{CL_i} 表示第 i 层的截止时间，β_{V_i} 表示节点 V_i 的开始时间，δ_{V_i} 表示节点 V_i 的截止时间。将入口节点和出口节点分别单独设为一层。

定义 5 给定图 $G=\langle V,A\rangle$，将每个并行分层中的单个节点或具有依赖关系的多个相邻节点定义为一个子路径，记作 $subPath$。

如果 $subPath$ 中的最后一个节点 V^* 在该层中没有直接后继，则令该 $subPath$ 的截止时间为

$$\delta_{subPath_j} = \delta_{CL_i}, \forall\, subPath_j \in CL_i。 \tag{6}$$

如果 $subPath$ 中的最后一个节点 V^* 在该层中有直接后继，则令该 $subPath$ 的截止时间为

$$\delta_{subPath_j} = \min_{\substack{V_k\in CL_i\cap succ(V^*) \\ 1\leqslant k\leqslant |CL_i\cap succ(V^*)|}} \{\beta_{V_k}\}, \quad \forall\, subPath_j \in CL_i。 \tag{7}$$

式中 $succ(V^*)$ 表示节点 V^* 直接后继的集合。

给定图 $G=\langle V,A\rangle$，并行分层后，对于每层中的所有 $subPath$，如果 $subPath_j$ 中只有一个节点 V^*，则将该节点的时间区间设置为

$$\begin{cases} \beta_{V^*} = \max\limits_{\substack{V_k\in CL_i\cap pred(V^*) \\ 1\leqslant k\leqslant |CL_i\cap pred(V^*)|}} \{\delta_{V_k}\}； \\ \delta_{V^*} = \delta_{subPath_j}, subPath_j \in CL_i。 \end{cases} \tag{8}$$

式中 $pred(V^*)$ 表示节点 V^* 直接后继的集合。

如果 subPath 中存在具有依赖关系的多个相邻节点，则根据该 subPath 的串行性，可采用 Markov 决策过程（Markov Decision Process，MDP）算法[12]，在实际执行过程中动态调整各个任务的执行区间，使其都能尽量选择相对较优的服务。

算法 1　CLWS 算法。

步骤 1　初始化工作流,为 DAG 各节点分配服务资源。

步骤 2　由定义 1 计算工作流的下界截止时间 $t_{\min}$。

步骤 3　根据定义 4 进行工作流并行分层。

步骤 4　由定义 5 求出工作流中每层的 $subPath$ 集合。

步骤 5　如果给定的工作流截止时间不满足式(4),则转步骤 8,输出提示信息,说明工作流的截止时间不能满足要求;否则计算层浮差,根据式(13)将每层的层浮差分配到各层中,重新计算每层的时间区间以及每层 $subPath$ 的时间区间。

步骤 6　遍历每层的 $subPath$ 集合,如果当前 $subPath$ 集合只含有唯一节点,则根据式(8)计算该节点的时间区间;否则结合式(10),采用 MDP 算法为 $subPath$ 中的节点计算最优服务。

步骤 7　计算工作流的总执行费用。

步骤 8　结束。

算法在实际执行过程中,如果节点的直接前驱提前完成,则可适当调节该节点的开始执行时间,增大该节点的时间区间;步骤 6 中,如果某节点在对应层中的直接后继已经完成服务的分配,则调整该节点的截止时间,使其始终不能超过直接后继的开始时间。

定义 6　给定图 $G=\langle V,A\rangle$,对于每层中具有依赖关系的多个相邻节点的 $subPath$,令 $subPath=\{V_1,V_2,\cdots,V_r\}$,将节点 V_i 选择第 $k(1\leqslant k\leqslant|SP_i|)$ 个服务后转向前一个节点 $V_j(j<i)$ 的过程用函数式 $f(V_i,S_i^k,V_j)$ 表示,其函数值表示第 k 个服务的执行费用。

$$f(V_i,S_i^k,V_j)=\begin{cases}c_i^k, & \beta_{V_i}\geqslant\beta_{subPath};\\ \infty, & \text{其他}。\end{cases}\tag{9}$$

定义 7　对于定义 6 中的 $subPath$,节点 V_i 在选择服务 S_i^k 时,其局部执行费用最优求解如下:

$$F(V_i)=\min_{1\leqslant k\leqslant|SP_i|}\{f(V_i,S_i^k,V_j)+F(V_j)。\}\tag{10}$$

对于式(9),如果节点 V_i 的完成时间超出所在 $subPath$ 的截止时间,则表明节点 V_i(或其在该 $subPath$ 中的前驱节点)的服务选择不合理,通过将函数值置为∞来隐式地去除相应不合理的服务选择。

2.2　并行分层算法 CLWS

由定义 3 可知,当给定工作流的截止时间大于其下界完成时间时,如果将两者之差分配到各个并行分层,则每层的执行时间区间会增大,层中各个任务的执行时间区间也会相应增大,各个任务可选择相对较优的服务,从而优化工作流的总执行费用。

定义 8　给定图 $G=\langle V,A\rangle$,将工作流的截止时间 t_{deadline} 与其下界完成时间 $t_{\min}$ 的差值称作层浮差,记作 TF,

$$TF=t_{\text{deadline}}-t_{\min}。\tag{11}$$

将式(11)计算得到的层浮差按每层节点所占比重分配到各层,各层分到的层浮差为

$$TF_{CL_i}=\frac{TF}{|V|-2}|CL_i|。\tag{12}$$

其中入口节点和出口节点所在层不进行层浮差分配。DBL 算法将层浮差平均分配到各个层,忽略各层中并行节点的差异性,将层浮差按各层节点所占比重进行差异性分配,实验结果有所改善。

将式(12)计算出的各层层浮差分配到相应层的各节点中,修改式(5)如下:

$$\begin{cases}\beta_{CL_i}=\min\limits_{\substack{V_k\in CL_i\cap CP\\1\leqslant k\leqslant|CL_i\cap CP|}}\{\beta_{V_k}\};\\ \delta_{CL_i}=\max\limits_{\substack{V_k\in CL_i\cap CP\\1\leqslant k\leqslant|CL_i\cap CP|}}\{\delta_{V_k}\}+TF_{CL_i};\\ \beta_{V_i}\geqslant\beta_{CL_i},\forall V_i\in CL_i;\\ \delta_{V_i}\leqslant\delta_{CL_i},\forall V_i\in CL_i。\end{cases}\tag{13}$$

2.3　案例说明

LAGrid-MOM 是基于面向服务的远程学习评价模型,本节以新资源发布—资源订阅为例说明 LAGrid-MOM 的工作过程[13]:①当资源提供者向网格节点上传新的学习资源时,资源发布服务产生一条包含该资源描述的元数据信息,并向系统中的所有网格节点广播该消息,这条消息的创建者为当前节点 ID,接收者为 Global 0(即全国区域 ID);②消息进入队列等待发送,该消息将发送到 Global 0 所对应的网格入口节点;③入口节点对地址 Global 0 进行动态解析,获得该区域所有网格节点支持的组织 ID 和该区域的后继区域 ID,系统自动获得不同区域 ID 以及该区域内其他网格节点所支撑的组织 ID;④网格入口节点按解析得到的地址转发消息;⑤当区域组织所在网格节点(消息转发节点)收到消息后,检查消息的接收者是全局范围,获得该区域内网格节点支撑的组织 ID 列表,并向支撑这些组织的网格节点转发该消息;⑥当一个网格节点收到消息后,将去掉消息中的多余参数,并根据消息类型

触发资源订购代理处理该消息的内容。图 2 描述了这一过程。

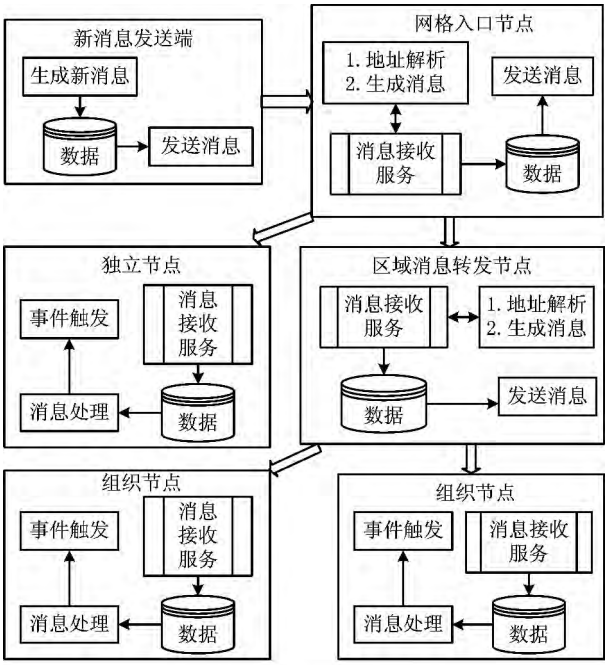

图2 LAGrid-MOM的消息处理过程

从资源发布到资源订阅成功，资源发布消息向网格中的所有节点广播，根据图 3，每个节点所属区域和支撑组织的服务提供者都会向服务注册中心发布能处理资源发布消息的服务，每个服务请求节点从服务注册中心请求服务，并对服务提供者支付相应的费用，通常服务的执行时间与其费用呈离散非增关系。资源消息的广播过程可以抽象为工作流，假定在时间有限的情况下为图 3 中每个节点选择最佳服务来完成资源订阅，使得总费用达到最少，即该过程的求解是一个基于截止时间约束的工作流调度问题。

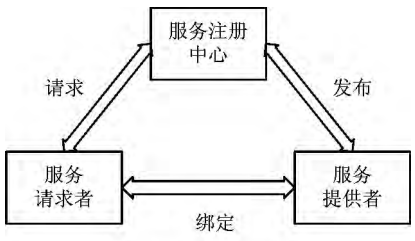

图3 网格服务的体系结构

上述实际案例的业务，从资源发布到资源订阅，可抽象化为一个如图 4 所示的工作流模型，表 1 所示服务池的数据表示各个节点完成消息处理的可用服务及其时间和费用（忽略消息传输过程中的时间和费用），服务处理相同消息时，所需处理的时间与其费用呈非增关系。图 4 中的节点 1 和节点 16 分别表示资源发布和资源订阅，即工作流入口节点 V_{entry} 和出口节点 V_{exit}，$SP(V_{\text{entry}})=SP(V_{\text{exit}})=\{\langle 0,0\rangle\}$。

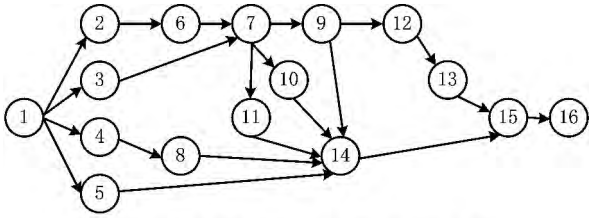

图4 资源发布到资源订阅过程的工作流图

表 1 图 4 中工作流的所有任务的服务池

任务	服务池 SP	任务	服务池 SP
1	$\{\langle 0,0\rangle\}$	9	$\{\langle 8,14\rangle,\langle 5,18\rangle\}$
2	$\{\langle 10,6\rangle,\langle 8,8\rangle,\langle 6,11\rangle\}$	10	$\{\langle 30,100\rangle,\langle 20,150\rangle,\langle 15,200\rangle\}$
3	$\{\langle 5,10\rangle,\langle 4,12\rangle\}$	11	$\{\langle 10,50\rangle,\langle 6,80\rangle\}$
4	$\{\langle 6,5\rangle\}$	12	$\{\langle 9,18\rangle\}$
5	$\{\langle 4,10\rangle,\langle 2,15\rangle\}$	13	$\{\langle 25,40\rangle,\langle 20,50\rangle\}$
6	$\{\langle 3,5\rangle,\langle 2,10\rangle,\langle 1,20\rangle\}$	14	$\{\langle 30,80\rangle,\langle 20,120\rangle,\langle 15,150\rangle\}$
7	$\{\langle 15,25\rangle,\langle 10,30\rangle\}$	15	$\{\langle 13,50\rangle,\langle 10,60\rangle\}$
8	$\{\langle 3,30\rangle\}$	16	$\{\langle 0,0\rangle\}$

假设用户给定的截止时间为 100，为每个任务分配执行时间最小的服务后，得到关键路径节点集合 $CP=\{1,2,6,7,9,12,13,15,16\}$，实例工作流的下界截止时间 $t_{\min}=61$。根据定义 4 和定义 5，可得实例工作流的并行分层和每层中的 $subPath$ 集合，如表 2 所示。

表 2 实例工作流的并行分层

层编号	层中 $subPath$ 集合
1	$\{\{1\}\}$
2	$\{\{2\},\{3\},\{4\},\{5\}\}$
3	$\{\{6,7\},\{8\}\}$
4	$\{\{9,12,13\},\{10,14\},\{11\}\}$
5	$\{\{15\}\}$
6	$\{\{16\}\}$

计算上述工作流，绘制实例工作流实际执行过程的甘特图，如图 5 所示，其中粗实线表示关键节点的执行过程。实例工作流甘特图能直观地展现出各个任务的实际执行过程。

在工作流的截止时间内满足定义 3 的约束条

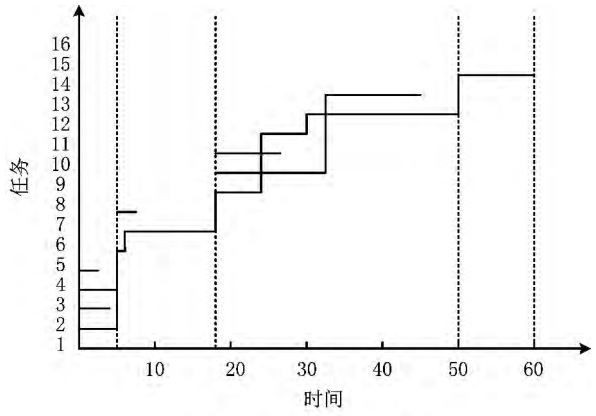

图5　实例工作流甘特图

件,由式(11)计算总层浮差 $TF=100-61=39$。由式(12)计算每层的层浮差(保留两位小数),根据 CLWS 算法最终获得各个任务的服务选择和实际执行时间区间,如表 3 所示,实例工作流的总执行费用为 493。

表 3　实例工作流各任务的服务选择和执行时间区间

任务	选择的服务	实际执行时间区间	任务	选择的服务	实际执行时间区间
1	S_1^1	[0,0]	9	S_9^1	[36.5,44.5]
2	S_2^1	[0,10]	10	S_{10}^1	[36.5,66.5]
3	S_3^1	[0,5]	11	S_{11}^1	[36.5,46.7]
4	S_4^1	[0,6]	12	S_{12}^1	[44.5,53.5]
5	S_5^1	[0,4]	13	S_{13}^1	[53.5,78.5]
6	S_6^1	[17.14,20.14]	14	S_{14}^2	[66.5,86.5]
7	S_7^1	[20.14,35.14]	15	S_{15}^2	[87.20,97.20]
8	S_8^1	[17.14,20.14]	16	S_{16}^1	[99.98,99.98]

3　模拟实验

工作流的 DAG 图有平衡和非平衡两种结构[10],如图 6 所示。针对图 6 中两种不同结构的工作流,分别采用 CLWS,DBL,DTL 和最小临界路线(Minimum Critical Path,MCP)算法[8]进行模拟实例测试与分析,评价 CLWS 算法的性能。

3.1　实验环境

工作流的 DAG 图均采用 DAG 生成器自动生成。为了模拟不同规模的工作流调度问题,假设 DAG 的任务数 $|V|\in\{10,20,30,40,50,60,70,80,90,100\}$,对于平衡结构的 DAG,并行分层中子路径的节点数目 $|subPath|\in\{1,2,3,4\}$,节点的出度数 $outDegree\in\{1,2,3\}$;对于非平衡结构的 DAG,节点的出度数 $outDegree\in\{1,2,3,4\}$。两种结构的

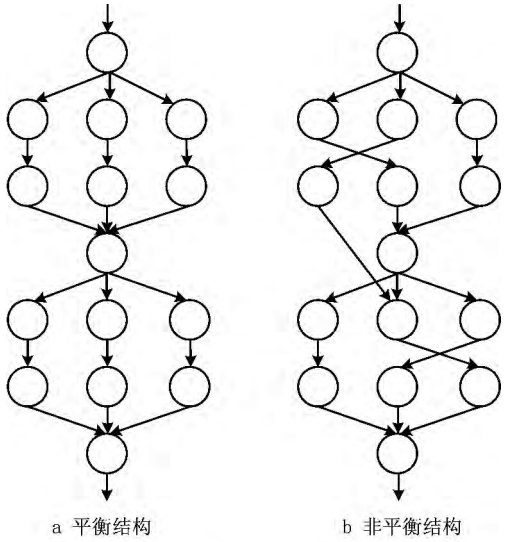

图6　DAG结构的分类

DAG 中任务的服务池长度在区间[5,10]内服从均匀分布,服务的执行时间在区间[5,30]内随机产生,为了模拟任务执行费用与执行时间呈离散非增关系,假设执行费用等于 1 000 除以执行时间,以模拟执行费用是随执行时间的增加呈离散的非递增变化。两种结构的 DAG 截止时间 $t_{deadline}=t_{min}+\mu\times(t_{max}-t_{min})$,其中 $\mu\in\{0.05,0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45,0.5\}$,$\mu$ 的取值分别对应 10 个实例。

3.2　实验结果与分析

假设算法 A 和算法 B 执行相同的工作流实例,得到总执行费用分别为 C_A 和 C_B,则算法 B 相对于算法 A 的执行费用提高率为 $E_{B|A}=\dfrac{(C_B-C_A)}{C_A}\times 100\%$;实验最终结果取每组实例的平均值。

由 DAG 生成器随机生成不同规模的工作流实例,针对每组实例分别采用 CLWS,DBL,DTL 和 MCP 算法计算相应工作流的总费用,并计算前三种算法相对于 MCP 算法的执行费用提高率。图 7 所示为不同规模的非平衡结构工作流,以及三种算法相对于 MCP 算法的总费用提高率。CLWS 算法采用并行分层方法,将工作流在实际执行过程中存在并行性的单个节点(或具有依赖关系的多个相邻节点)分到同一层,以减少工作流的时间碎片,优化工作流的总费用,从图 7 可知 CLWS 算法的执行性能比 DBL 和 DTL 都高。观察图 6 中平衡结构 DAG 的结构特性可知,DBL 和 DTL 算法的执行过程相同,因此实验阶段仅计算 CLWS 和 DBL 算法相对于 MCP 算法的执行费用提高率。图 8 所示为不同规模的平衡结构工作流下两种算法相对 MCP 算法

的执行费用提高率。观察非平衡结构的 DAG 图，可以直观地判断单个节点(或具有依赖关系的多个相邻节点)间的并行性，CLWS 算法的分层数明显比 DBL 算法的分层数要少，较少的分层可以降低工作流的时间碎片；此外，CLWS 算法将具有依赖关系的多个相邻节点分到同一层，可采用 MDP 算法优化各个层的执行费用，相比 DBL 算法能够得到较好的优化结果。从图 8 可以看出，针对平衡结构工作流，CLWS 的执行性能明显比 DBL 高。

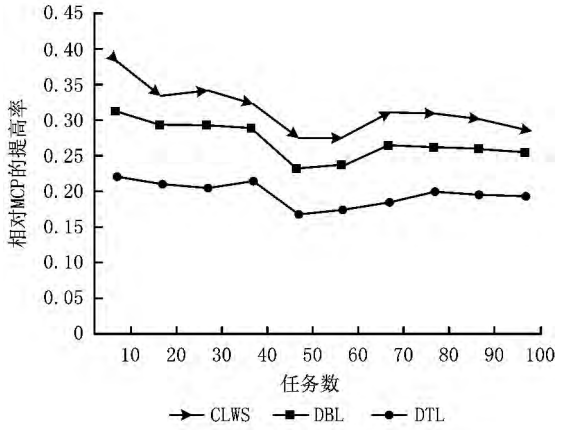

图7 不同规模非平衡工作流算法间的对比

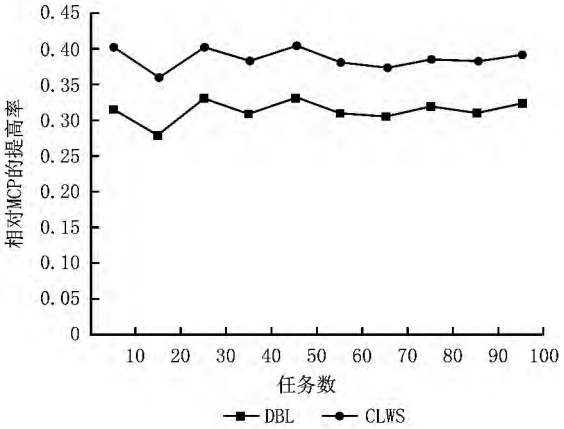

图8 不同规模非平衡工作流算法间的对比

3.3 参数对并行分层算法性能的影响

工作流的截止时间 $t_{deadline}$ 是完成该工作流的最小时间。假设工作流的截止时间大于完成该工作流的下界完成时间，CLWS 算法、DBL 算法和 DTL 算法均采用层浮差方法实现工作流总执行费用的优化。

针对任务数为 80 的工作流进行实验结果分析，图 9 所示为不同截止时间情形下非平衡结构工作流采用 CLWS，DBL，DTL 和 MCP 四种算法的执行费用对比情况。分析图 9 可知：随着截止时间系数 μ 的不断增加，即工作流的截止时间不断增加，MCP 算法的执行费用保持不变；CLWS，DBL 和 DTL 算法的执行费用均在不同程度上随着截止时间的增加而逐渐减少。随着截止时间的不断增加，工作流的层浮差也相应增加，因为 CLWS 算法对层浮差的利用率相对于 DBL 算法和 DTL 算法较高，所以工作流的总执行费用较少；对于 DBL 算法和 DTL 算法，当给定的工作流截止时间小于满足分层所需的工作流最小完成时间时，两种算法均不适用，需采用 MCP 算法优化工作流的总费用。而 CLWS 算法则不会出现该情形，并且相对于 MCP 算法，CLWS 算法能明显优化工作流总执行费用。根据 3.2 节的分析，针对平衡结构工作流，DBL 算法和 DTL 算法的执行过程相同，实验对比分析了 CLWS，DBL 和 MCP 三种算法的工作流执行费用，图 10 所示为不同截止时间情形下平衡结构工作流三种算法的执行费用。当截止时间系数 μ 不断增加时，MCP 算法的执行费用不变，CLWS 算法和 DBL 算法的执行费用在不同程度上逐渐减小，并且 CLWS 算法不会出现工作流截止时间小于满足分层所需的工作流最小完

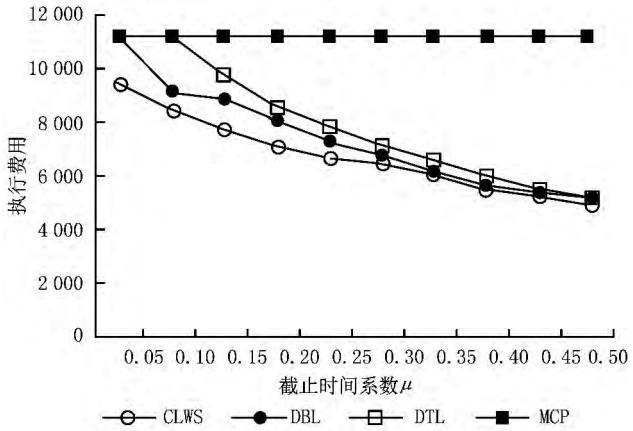

图9 不同截止时间情形下非平衡结构工作流执行费用对比

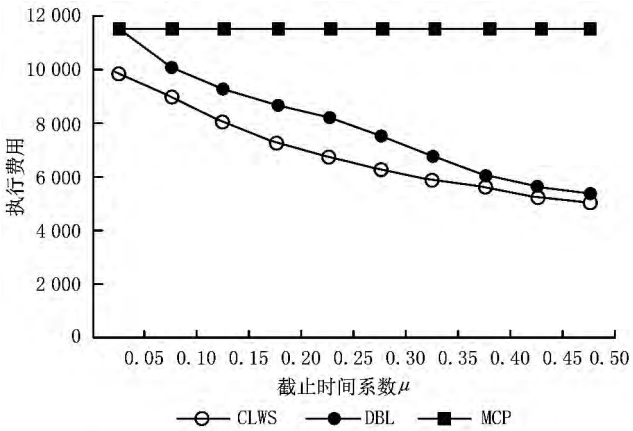

图10 不同截止时间情形下平衡结构工作流执行费用对比

成时间的情形,其执行性能相对提高。

4　结束语

本文针对工作流分层调度问题提出 CLWS 算法,该算法使工作流并行执行的任务尽可能分到同一层,结合工作流的 DAG 结构合理地为每个任务节点分配时间区间,以保证每个任务能选择相对较优的服务。相对于 DBL 算法和 DTL 算法,CLWS 算法能减少分层数,增加全局层浮差的利用率,提高任务执行的并行性,减少总的执行费用。对于每层中具有依赖关系的多个相邻任务,采用 MDP 算法可进一步优化工作流总执行费用。CLWS 算法避免了 DBL 算法和 DTL 算法在特定环境下不适用的情形,能在一定程度上减少工作流的时间碎片。针对平衡和非平衡两种结构工作流进行模拟实验分析,实验结果表明 CLWS 算法的平均执行性能优于 DBL 算法和 DTL 算法。

在实际工作流执行过程中,由于任务资源竞争导致某些任务调度失败或者任务延迟完成等现象,都会对最终的执行结果产生一定影响,后续研究中将引入工作流动态调度机制[14-15]来处理一些服务调度故障,以保证工作流的正常执行。

参考文献:

[1] KLLAPI H,SITARIDI E,TSANGARIS M M, et al. Schedule optimization for data processing flows on the cloud[C]//Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. New York, N. Y., USA: ACM, 2011:289-300.

[2] TAN Wen'an, JIANG Chuanqun, LI Ling, et al. Role-oriented process-driven enterprise cooperative work using the combined rule scheduling strategies[J]. Information System Frontiers,2008,10(5):519-529.

[3] LA ROSA M,TER HOFSTEDE A H M,WOHED P, et al. Managing process model complexity via concrete syntax modifications[J]. IEEE Transactions on Industrial Informatics, 2011,7(2):255-265.

[4] BLYTHE J, JAIN S, DEELMAN E, et al. Task scheduling strategies for workflow-based applications in grids[C]//Proceedings of the IEEE International Symposium on Cluster Computing and Grid. Washington, D. C., USA: IEEE, 2005: 759-767.

[5] JIN Hai, CHEN Hanhua, LYU Zhipeng, et al. QoS optimizing model and solving for composite service in CGSP job manager[J]. Chinese Journal of Computers, 2005, 28(4): 578-588 (in Chinese). [金　海,陈汉华,吕志鹏,等. CGSP 作业管理器合成服务的 QoS 优化模型及求解[J]. 计算机学报,2005,28(4):578-588.]

[6] LIN Jianning, WU Huizhong. Scheduling in grid computing environment based on genetic-algorithm[J]. Journal of Computer Research and Development, 2004, 41(12): 2190-2194(in Chinese). [林剑柠,吴慧中. 基于遗传算法的网格资源调度算法[J]. 计算机研究与发展,2004,41(12):2190-2194.]

[7] YU Mingyuan, ZHU Yihua, LIANG Ronghua. A grid service-workflow scheduling using hybrid particle swarm[J]. Journal of Huazhong University of Science and Technology: Natural Science, 2008, 36(4): 45-47(in Chinese). [于明远,朱艺华,梁荣华. 基于混合微粒群算法的网格服务工作流调度[J]. 华中科技大学学报:自然科学版,2008,36(4):45-47.]

[8] YUAN Yingchun, LI Xiaoping, WANG Qian, et al. Bottom level based heuristic for workflow scheduling in grids[J]. Chinese Journal of Computers, 2008, 31(2): 282-290(in Chinese). [苑迎春,李小平,王　茜,等. 基于逆向分层的网格工作流调度算法[J]. 计算机学报,2008,31(2):282-290.]

[9] YU Jia, BUYYA R, THAM C K. Cost-based scheduling of scientific workflow applications on utility grids[C]//Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing. Washington, D. C., USA: IEEE Press, 2005: 140-147.

[10] YU Jia, BUYYA R, KOTAGIRI R. Workflow scheduling algorithms for grid Computing[J]. Lecture Notes in Computer Science, 2008, 146: 173-214.

[11] DEMEULEMESTER E L, HERROELEN W S, ELMAGHRABY S E. Optimal procedures for the discrete time/cost trade-off problem in project networks[J]. European Journal of Operational Research, 1996, 88(1): 50-68.

[12] SULISTIO A, BUYYA R. A gird simulation infrastructure supporting advance reservation[C]//Proceedings of the 16th International Conference on Parallel and Distributed Computing and Systems. Cambridge, Mass., USA: MIT Press, 2004.

[13] XU Jun, SHI Meilin, LI Yushun, et al. Grid computing and e-learning grid: architecture, key technologies and applications[M]. Beijing: Science Press, 2005: 126-145(in Chinese). [许　骏,史美林,李玉顺,等. 网格计算与 e-Learning Grid:体系结构,关键技术,示范应用[M]. 北京:科学出版社,2005:126-145.]

[14] ZHAO Henan, SAKELLARIOU R. A low-cost rescheduling policy for dependent tasks on grid computing systems[M]. Berlin, Germany: Springer-Verlag, 2004.

[15] YU Zhifeng, SHI Weisong. An adaptive rescheduling strategy for gird workflow applications[C]//Proceedings of the International Symposium on Parallel and Distributed Processing. New York, N. Y., USA: IEEE Press, 2007.

作者简介:

谭文安(1965－),男,湖北荆州人,教授,博士,研究方向:软件工程及其开发环境技术、企业动态建模与优化、智能信息系统等,E-mail: twajsj@sohu.com;

＋路广振(1989－),男,安徽界首人,硕士研究生,研究方向:协同计算、服务计算等,通信作者,E-mail: lgzhnuaa@163.com;

孙　勇(1977－),男,湖北咸宁人,博士研究生,研究方向:软件工程、协同计算、服务计算等。