

# 基于逆向分层的网格 workflow 调度算法

苑迎春<sup>1),3)</sup> 李小平<sup>1),2)</sup> 王 茜<sup>1),2)</sup> 张 毅<sup>1),2)</sup>

<sup>1)</sup>(东南大学计算机科学与工程学院 南京 210096)

<sup>2)</sup>(东南大学计算机网络和信息集成教育部重点实验室 南京 210096)

<sup>3)</sup>(河北农业大学信息科学与技术学院 河北 保定 071001)

**摘 要** 有向无环图 DAG(Directed Acyclic Graph)描述的工作流时间费用优化问题是计算网格下一个基本的且难以求解的问题. 通过分析 DAG 图中活动的并行和同步完成特征, 采取由后向前方法将活动逆向分层(Bottom Level, BL), 将 workflow 截止期转化为层截止时间, 提出截止期约束的逆向分层费用优化算法 DBL(Deadline Bottom Level). 算法中同层活动的开始时间不同于 DTL(Deadline Top Level)算法中设置相同的策略, 而是分别由其前驱活动确定, 时间浮差被平均分配到各分层, 以尽量增大活动的费用优化区间. 通过大量模拟实验将 DBL 和 MCP(minimum Critical Path)、DTL 两算法比较, 结果表明 DTL 将 MCP 的平均费用降低 15.62%, 而 DBL 将 MCP 的平均费用降低 24.74%. 最后讨论了截止期和分组参数对算法性能的影响.

**关键词** 计算网格; workflow; 有向无环图; 启发式算法; 逆向分层  
中图法分类号 TP393

## Bottom Level Based Heuristic for Workflow Scheduling in Grids

YUAN Ying-Chun<sup>1),3)</sup> LI Xiao-Ping<sup>1),2)</sup> WANG Qian<sup>1),2)</sup> ZHANG Yi<sup>1),2)</sup>

<sup>1)</sup>(School of Computer Science and Engineering, Southeast University, Nanjing 210096)

<sup>2)</sup>(Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, Nanjing 210096)

<sup>3)</sup>(Faculty of Information Science and Technology, Agriculture University of Hebei, Baoding, Hebei 071001)

**Abstract** Efficient scheduling of workflow applications represented by DAG (Directed Acyclic Graph) with the objective of time-cost optimization is fundamental and intractable in computational Grid. By analyzing parallel and synchronization properties of DAG, all tasks are divided into BL (Bottom Level) groups using a backward method. The workflow deadline can be transformed into the time intervals of all tasks based on BL groups, a bottom leveling based algorithm called DBL (Deadline Bottom Level) is proposed. In DBL, the starting time of a task in each group is determined by the maximum finish time among those of its predecessors rather than the finish time of its parent group, which is adopted by DTL (Deadline Top Level). The total time float is allocated equally to each group, which manage to enlarge the cost optimization intervals of all tasks. By comparing DBL with DTL and MCP (Minimum Critical Path), which is a cost optimization method based on the minimum critical path, experimental results show that DBL can save 24.74% average cost of MCP and DTL can only do 15.62%. In other words, DBL outperforms both DTL and MCP. As well, parameters affecting algorithms (such as leveling parameters, deadlines) are analyzed by experiments.

**Keywords** computational grid; workflow; directed acrylic graph; heuristics; bottom level

收稿日期:2006-06-13;最终修改稿收到日期:2007-10-29. 本课题得到国家自然科学基金(60672092,60504029,90412014)资助. 苑迎春,女,1970年生,博士研究生,副教授,主要研究方向为网格计算、服务组合. E-mail: nd\_hd\_yc@163.com. 李小平,男,1970年生,博士,副教授,博士生导师,主要研究领域为机器调度、项目调度和服务计算. 王 茜,女,1946年生,教授,博士生导师,主要研究领域为信息集成技术、数据库技术以及计算机协同工作. 张 毅,男,1983年生,博士研究生,主要研究方向为机器调度、服务计算.

## 1 引言

网格计算<sup>[1-2]</sup>作为下一代分布计算环境,为解决复杂计算问题提供了一种新型计算模式. 网格中的许多大型应用通常被分解成若干个彼此联系的活动(任务),活动及活动间的复杂约束关系可看作是一个工作流<sup>[3]</sup>. 有向无环图 DAG (Directed Acyclic Graph)是工作流的一种常用描述方式,广泛存在于 e-science、e-business 等网格应用领域<sup>[4]</sup>. 工作流调度就是为活动选择并分配合适的资源,并满足不同的调度目标. 活动和资源的映射实质是一个复杂的优化问题,通常情况下多为 NP-hard 问题<sup>[5]</sup>. 目前已有许多文献提出最小化完工时间(makespan)的启发式调度算法;现有的网格工作流管理系统<sup>[3,6]</sup>也多以最小化完工时间作为调度目标. 随着网格应用向商业领域延伸,网格资源由无偿服务转为有偿服务. 工作流调度不再仅以最小化完工时间为衡量标准,还要同时考虑运行应用所需的成本,即网格工作流调度是一个时间费用优化问题:在截止时间(deadline)约束下,最小化工作流费用.

时间-费用的优化问题在项目管理领域已得到广泛研究. 早期研究多集中于连续费用函数情况,即活动所需费用是其执行时间的一个非增的、连续函数. 离散时间费用优化问题(Discrete Time-Cost Tradeoff Problem, DTCTP)指活动所需费用是其执行时间的一个离散非增函数. 由于在实际问题中广泛存在,DTCTP 已引起许多学者关注. 文献[7]详细介绍了该问题的研究现状. 已有研究成果多采用分支定界、动态规划等精确方法求解<sup>[8]</sup>. 尽管 De 等<sup>[9]</sup>已证明该优化问题在通常情况下是强 NP-hard 问题,但该领域很少有人用启发式策略求解该问题. Akkan<sup>[10]</sup>基于网络图分解,利用列生成技术给出一种上下界求解方法.

对网格环境下的时间费用优化问题, Buyya 等<sup>[11-12]</sup>提出一种实现资源管理的经济模型,并基于不同 deadline/budget 约束提出求解独立任务调度的三种启发式算法; Lin 等<sup>[13]</sup>指出在一定条件下(网格服务资源足够多,保证并行任务同时执行),网格 DAG 应用的时间费用优化问题可抽象为项目管理领域中的 DTCTP 问题. 文献[14]对截止期(deadline)约束的工作流((DAG 表示)费用优化问题提出

截止期分解的求解策略 DTL(Deadline Top Level),算法借助正向分层 TL(Top Level)将全局截止期分解为层截止时间,活动在限定的时间区间内选择费用最小的服务资源,所有活动的局部费用优化之和得到全局近似最优解. 虽然 DTL 算法是一种简单且运行效率很高的启发式算法,但其仍存在以下不足: (1) TL 分层基于活动并行和同步开始特征分组,具有同步完成特性的任务不一定能划分到同一层中; (2) 截止期分解规定同层活动有相同开始时间和截止时间,层开始时间是前驱层的截止时间,这种设置策略使部分活动的费用优化区间减小,算法性能受到影响.

针对 DTL 算法的不足,本文提出一种新的启发式算法:截止期约束的逆向分层调度算法 DBL(Deadline Bottom Level). 算法基于结点的逆向深度(在 4.1 节定义)分层,使同步完成活动划分到同一组中,并利用逆向分层 BL(Bottom Level)将工作流截止期转化为活动的时间区间;DBL 仍规定同层活动有相同截止期,而开始时间分别由其前驱活动决定,尽量增大活动的时间区间,活动能更合理地局部优化费用. 模拟结果验证了算法的性能和效率.

## 2 问题描述

网格中许多大型应用通常由多个活动相互协作完成,活动及活动间的约束关系可模型化为一个工作流,有向无环图 DAG 是其常用的一种描述方式. 令 DAG 图记作  $G = \{V, E\}$ , 其中  $V = \{1, 2, \dots, n\}$  ( $|V| = n$ )表示应用中所有任务集合(结点、任务和活动在文中含义相同);  $E$  是有向边集合,表示活动间的控制或数据依赖关系,即对  $\forall (i, j) \in E$ ,  $i$  执行完后  $j$  才可执行( $i, j$  是结点编号,规定  $i < j$ ). 图中没有前驱的结点称为入口任务(entry task),没有后继的结点称为出口任务(exit task). 为算法实现方便,规定图中只有单一入口任务和出口任务(可添加虚结点使其满足要求,虚结点是执行时间和成本都为 0 的任务). 因此,结点 1 和  $n$  分别是唯一入口和出口任务. 图 1 是一个简单的工作流实例.

开放网格服务架构 OGSA(Open Grid Service Architecture)将 Web 服务引入网格中,服务作为资源的抽象形式为用户提供了一致访问接口<sup>[2]</sup>. 服务接口包含功能接口和描述服务质量 QoS(Quarlity

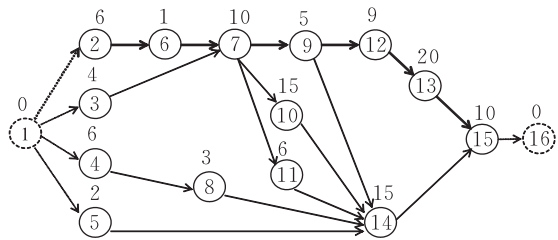


图 1 工作流应用实例(DAG 表示)

of Service)的非功能接口. 功能接口描述服务能力; 非功能接口用响应时间、价格(费用)、可靠性等 QoS 属性描述. 分布、异构的网格环境可使不同服务提供者提供相同功能、不同 QoS 的资源, 即有相同功能的不同服务执行同一个任务时需要不同的时间和费用, 且费用通常是时间的离散非增函数<sup>[14]</sup>. 因此, 对  $\forall i \in V$ , 都存在一个能够完成该任务的候选服务集, 称为  $i$  的服务池, 记作  $P(i)$ .  $l(i)$  是  $P(i)$  的集合长度, 表示能完成任务  $i$  的服务个数;  $S_{ik}$  表示能完成任务  $i$  的第  $k$  个服务;  $(c_{ik}, t_{ik})$  描述  $S_{ik}$  的费用、时间属性, 表示任务  $i$  选择服务  $S_{ik}$  执行时所需费用和执行时间, 则  $P(i) = \{S_{ik} = \{(t_{ik}, c_{ik})\}, 1 \leq k \leq l(i)\}$ . 表 1 给出了图 1 中各任务的服务池实例.

表 1 图 1 中各活动的服务池实例

活动编号	$P(i)$	$l(i)$
1	$\{(0,0)\}$	1
2	$\{(6,10), (8,8), (11,6)\}$	3
3	$\{(10,5), (12,4)\}$	2
4	$\{(5,6)\}$	1
5	$\{(10,4), (15,2)\}$	2
6	$\{(5,3), (10,2), (20,1)\}$	3
7	$\{(25,15), (30,10)\}$	2
8	$\{(30,3)\}$	1
9	$\{(14,8), (18,5)\}$	2
10	$\{(100,30), (150,20), (200,15)\}$	3
11	$\{(50,10), (80,6)\}$	2
12	$\{(18,9)\}$	1
13	$\{(40,25), (50,20)\}$	2
14	$\{(80,30), (120,20), (150,15)\}$	3
15	$\{(50,13), (60,10)\}$	2
16	$\{(0,0)\}$	1

假定工作流内部处理时间忽略不计, 且所有候选服务都能提供所承诺的服务质量, 则工作流运行时间取决于活动所选服务的运行时间. 这样, 对任意一个 DAG 表示的工作流应用, 给各活动分配一种服务执行模式, 就有一条关键路径(运行时间最长的路径)与之对应. 工作流完工时间就是关键路径的运行时间, 工作流总费用就是所有活动花费之和.

工作流截止期  $\delta_n$  由用户给出(一般用常数表示), 它表示完成工作流所有任务所需的最晚完成时间. 调度目标就是在满足给定截止期限下, 为工作流

中各任务分配合适的服务, 使工作流所需费用最小, 形式化描述如下

$$\min \sum_{i \in V} \sum_{1 \leq k \leq l(i)} x_{ik} c_{ik} \tag{1}$$

$$\text{s. t. } \sum_{k=1}^{l(i)} x_{ik} = 1, \forall i \in V \tag{2}$$

$$f_i \leq f_j - \sum_{k=1}^{l(i)} t_{ik} x_{ik}, \forall (i,j) \in E \tag{3}$$

$$f_n \leq \delta_n \tag{4}$$

$$x_{ik} \in \{0,1\}, i \in V, 1 \leq k \leq l(i) \tag{5}$$

其中  $x_{ik}$  是布尔变量, 当任务  $i$  选择其服务池中的第  $k$  个服务执行时其值为 1, 否则为 0.  $f_i$  表示任务  $i$  的完成时间,  $f_n$  为工作流完工时间.

式(1)表示最小化工作流费用; 约束(2)是任务只能选择一个服务资源执行; 约束(3)满足偏序关系; 式(4)满足截止期约束; 式(5)说明  $x_{ik}$  是布尔变量.

已知截止期约束的工作流费用优化问题通常情况下是一个 NP-hard 问题<sup>[13]</sup>. 本文通过构造启发式算法将工作流截止期  $\delta_n$  分解为任务的截止时间  $\delta_i$ , 开始时间  $\beta_i$  由其前驱活动确定; 在时间区间  $[\beta_i, \delta_i]$  约束下任务  $i$  选择费用最小的服务执行, 工作流费用的近似最优解  $C_{\text{total}}$  就是所有任务的最优费用之和, 形式化描述为

$$C_{\text{total}} = \sum_{i \in V} \min_{1 \leq k \leq l(i)} \{x_{ik} c_{ik}\} \tag{6}$$

$$\text{s. t. } \beta_i + t_{ik} \leq \delta_i, 1 \leq k \leq l(i) \tag{7}$$

式(6)表示启发式算法的目标函数; 式(7)是任务的服务选择需满足其时间窗口约束.

因此, 使目标函数(6)取值最小的关键在于将  $\delta_n$  合理转化为任务的起止时间区间  $[\beta_i, \delta_i]$ ,  $i \in V$ . 其转化方法不仅满足任务间的偏序关系, 而且要尽量增大区间  $[\beta_i, \delta_i]$  的长度, 使任务的局部费用最小.

### 3 最小关键路径法(MCP)

首先给出工作流下界完工时间的定义.

**定义 1.** 给定图  $G=(V,E)$ , 对  $\forall i \in V$ , 从其服务池中选择运行最快的服务, 则该分配得到的关键路径称为最小关键路径, 其路径长度称为工作流下界完工时间  $L_{CT}$ , 即  $L_{CT} = \sum_{i \in CP} \min_{1 \leq k \leq l(i)} \{t_{ik}\}$  (其中  $CP$  是关键活动集).

由定义 1 可知,  $L_{CT}$  是工作流完成所有任务所需的最小时间. 因此, 用户给定的截止期  $\delta_n$  必须大于或等于  $L_{CT}$  才能保证工作流完成.

在最快服务分配下, 根据关键路径法 CPM (Critical Path Method) 可计算任务  $i \in V$  的最早开始时间  $ES_i$ . 令  $\beta_i$  表示任务  $i$  所允许的最早开始时间,  $\delta_i$  表示  $i$  所允许的最晚完成时间 (即截止时间), 则  $i$  的可执行时间区间可按式 (8) 计算:

$$\begin{cases} \beta_i = ES_i \\ \delta_i = \min_{j \in succ(i)} \{ES_j\} \end{cases} \quad (8)$$

其中  $succ(i)$  是任务  $i$  的后继活动集合. 若任务在式 (8) 确定的时间区间内选择费用最小的服务, 此时工作流完工时间仍为  $L_{CT}$ , 且非关键活动所需费用得到优化. 将这种费用优化方法称为最小关键路径法 MCP (Minimum Critical Path).

MCP 算法在保持最小完工时间的同时, 仅对非关键活动, 选择价格便宜、运行时间长一些的服务, 降低工作流总费用. 当截止期  $\delta_n$  大于  $L_{CT}$  时, 在区间  $[L_{CT}, \delta_n]$  内延迟工作流完工时间可进一步优化费用.

## 4 逆向分层算法

启发式算法的思想是将截止期  $\delta_n$  分解为任务的时间区间  $[\beta_i, \delta_i]$ , 从而将全局费用优化问题转化为活动的局部费用优化问题. 当截止期  $\delta_n$  大于  $L_{CT}$  时, 工作流存在松弛时间. 为尽量增大每个任务的时间区间, 必须将  $\delta_n$  合理地分布到每个任务上, 由此给出如下定义和性质.

### 4.1 相关定义和性质

**定义 2.** 给定图  $G=(V, E)$ , 结点  $i$  的逆向深度  $BD(i)$  定义为  $i$  到出口结点  $n$  的最长路径长度 (路径长度指边的个数).

$BD(i)$  的递归计算公式为

$$BD(i) = \begin{cases} 0, & i = n \\ \max_{j \in succ(i)} \{BD(j)\} + 1, & \text{其它} \end{cases} \quad (9)$$

由上述公式可知结点 1 的逆向深度最大.

将相同深度值的任务划分为一层 (组), 这种分组方法称为  $G$  的逆向分层 BL (Bottom Level). 第  $k$  ( $0 \leq k \leq BD(1)$ ) 个分组包含的任务为

$$BL_k = \{i \mid BD(i) = k, i \in V\} \quad (10)$$

将所有 BL 分组按深度值降序排列得到 BL 列表, 列表长度为  $G_L = BD(1) + 1$ . 表中第一层是  $BL_{BD(1)}$ , 最后一层是  $BL_0$ . 对任意  $BL_k$  ( $0 < k < BD(1)$ ), 规定其前驱层是  $BL_{k+1}$ , 后继层是  $BL_{k-1}$ , 则有如下定理成立.

**定理 1.** 给定图  $G=(V, E)$ , 结点划分为  $G_L$  个

BL 分组, 则 (1) 对  $\forall i, j \in BL_k$ ,  $i$  和  $j$  之间无偏序约束, 可并行执行; (2)  $\forall i \in BL_k$ , 若  $i < n$ , 则至少存在一个直接后继结点  $j$ , 满足  $j \in BL_{k-1}$  且  $BD(j) = k-1$ .

证明. (1) 反证法, 假设  $i$  和  $j$  之间存在偏序约束, 则  $i$  和  $j$  之间至少存在一条可达路径. 不妨设  $i$  是  $j$  的 (直接) 前驱, 由定义 2 知  $BD(i) > BD(j)$ , 即  $BD(i) \neq BD(j)$ , 因此  $i$  和  $j$  不在同一层, 与已知矛盾.

(2) 因为  $i \in BL_k$ , 则有  $BD(i) = k$ ; 又因为  $i < n$ , 所以结点  $i$  一定存在后继结点. 根据式 (9) 知  $BD(i) = \max_{j \in succ(i)} \{BD(j)\} + 1$ , 即  $\max_{j \in succ(i)} \{BD(j)\} = k-1$ . 所以  $i$  至少存在一个后继  $j$  且  $BD(j) = k-1$ . 由定义 2 得  $j \in BL_{k-1}$ . 证毕.

定理 1 说明, 逆向分层中同层任务除具有并行执行特点之外, 还可能有相同直接后继, 即有同步完成特性的任务会划分到同层中. 对这些任务设置相同截止时间仍可保持同步完成特征. 开始时间不同, 分别由前驱任务决定. 这样, 任务的时间区间尽量被放大. 当每个任务选择最快服务执行时, BL 分组的截止时间  $\delta_{BL_i}$  及任务的  $\beta_i$  和  $\delta_i$  按如下公式设置

$$\begin{cases} \beta_j = 0, & j = 1 \\ \delta_{BL_i} = \max \left\{ \beta_j + \min_{1 \leq k \leq l(j)} \{t_{jk}\} \right\}, & j \in BL_i \\ \delta_j = \delta_{BL_i}, & j \in BL_i \\ \beta_j = \max \{\delta_i\}, & j \neq 1, i \in pred(j) \end{cases} \quad (11)$$

由式 (11) 确定的工作流完工时间称为 BL 最小完工时间, 记作  $BL_{\min}$ , 它是 BL 列表中最后一层  $BL_0$  的截止时间, 即  $BL_{\min} = \delta_{BL_0}$ .

**定理 2.** 任意给定图  $G=(V, E)$ , 都有  $BL_{\min} \geq L_{CT}$ .

证明. 由定义 1 知  $L_{CT} = \sum_{i \in CP} \min_{1 \leq k \leq l(i)} \{t_{ik}\}$ . 不妨设  $CP = \{i_1, i_2, \dots, i_m\}$ , 各任务相应的最早完成时间设为  $\{f_{i_1}, f_{i_2}, \dots, f_{i_m}\}$ .

根据定理 1, 关键活动  $i_1, i_2, \dots, i_m$  位于不同 BL 分组内. 设  $i_k \in BL_i$  ( $1 \leq k \leq m$ ), 则由式 (11) 知  $\delta_{BL_i} \geq f_{i_k}$ .

(1) 若  $\exists i_k \in CP$ , 满足  $\delta_{BL_i} > f_{i_k}$ , 则  $i_k$  的截止时间  $\delta_{i_k}$  会延迟到  $\delta_{BL_i}$ . 已知关键活动延迟会使工作流完工时间推迟, 因此必有  $BL_{\min} > L_{CT}$ ;

(2) 若对  $\forall i_k \in CP$ , 满足  $\delta_{BL_i} = f_{i_k}$ , 则关键活动的起止时间不发生变化, 即  $BL_{\min} = L_{CT}$ .

综合 (1)、(2), 可得  $BL_{\min} \geq L_{CT}$  成立. 证毕.

**定义 3.** 逆向分层浮差  $T_{ws}$  定义为全局截止期  $\delta_n$  和 BL 最小完工时间之差, 即  $T_{ws} = \delta_n - BL_{\min}$ .

4.2 DBL 算法描述

通过 4.1 节分析可知, 当截止期  $\delta_n \geq BL_{\min}$  时, workflow 任务可逆向分层并用式(11)将截止期转化为活动的开始时间和截止时间. 这种分解方法不仅满足截止期  $\delta_n$  约束, 而且还可利用逆向分层浮差  $T_{ws}$  进一步增大活动的时间区间. 设置思路如下: 将  $T_{ws}$  平均分配到各层, 令  $T_{LS}$  表示层浮差, 则

$$T_{LS} = T_{ws} / (G_L - 2) \tag{12}$$

其中虚活动组成的两个分组除外, 这样每层(不包含虚组)延长相同层浮差  $T_{LS}$ , 式(11)可转变为式(13)的截止期分解方法.

$$\begin{cases} \beta_j = 0, & j = 1 \\ \delta_{BL_i} = \max_{j \in BL_i} \{ \beta_j + \min_{1 \leq k \leq l(j)} \{ t_{jk} \} \} + T_{LS}, & i \neq 0, i \neq BD(1) \\ \delta_j = \delta_{BL_i}, & j \in BL_i \\ \beta_j = \max \{ \delta_i \}, & i \in pred(j) \end{cases} \tag{13}$$

依据式(13)确定活动开始时间和截止时间的办法不仅能满足截止期  $\delta_n$  约束, 而且进一步扩大了每个任务的时间区间, 任务能更合理地优化费用.

当  $L_{CT} \leq \delta_n < BL_{\min}$  时, BL 分组的截止期分解方法不能保证 workflow 满足截止期约束, 此时采用 MCP 算法优化费用.

当  $\delta_n < L_{CT}$  时,  $\delta_n$  是一个无效约束值, 因为 workflow 最小完工时间为  $L_{CT}$ . 需用户输入有效截止期  $\delta_n$  以保证可行性.

任务的时间区间确定后, 在其时间窗口约束下可选择费用最小的服务执行, 局部费用优化求和得到 workflow 费用的近似最优解.

综上所述, DBL 调度算法描述如下.

算法 1. DBL 算法.

- 1. 初始化并为所有任务查找相应服务资源池  $P(i)$ ,  $i \in V$ ;
- 2. 计算 workflow 最小完工时间  $L_{CT}$ ;
- 3. 根据式(9)计算任务的逆向深度, 并划分 BL 分组;
- 4. 根据式(11)计算 BL 最小完工时间  $BL_{\min}$ ;
- 5. 提示用户在截止期有效范围内输入  $\delta_n \geq L_{CT}$ ;
- 6. 当  $L_{CT} \leq \delta_n < BL_{\min}$  时, 用式(8)确定任务的  $\beta_i$  和  $\delta_i$ ;
- 7. 当  $\delta_n \geq BL_{\min}$  时, 用式(13)确定任务的  $\beta_i$  和  $\delta_i$ ;
- 8. 获取就绪任务队列  $RL$ ;
- 9. 若  $RL$  不空, 则依据约束(7)对就绪任务从其服务池中  
选择费用最小服务资源; 否则转步 11;
- 10. 发送控制或数据信息到分配的服务资源并启动执行;

- 11. 等待任务完成事件;
- 12. 若还有任务未被调度, 转步 8; 否则转步 13;
- 13. 计算 workflow 所需总费用;
- 14. 输出结果, 停止.

算法实际运行过程中, 任务的开始时间可能会因前驱结点在其截止时间之前完成而提前, 此时可调整任务开始时间使其提前执行.

DBL 算法中, 步 2, 3, 4, 7, 8 的时间复杂度最差情况下为  $O(n^2)$ ; 而步 9 到步 13 的时间复杂度为  $O(n \times len(RL) \times M)$ , 其中  $n$  是 workflow 中任务总数,  $len(RL)$  是就绪队列长度, 最大值为  $n$ ,  $M = \max_{i \in V} \{ l(i) \}$  是所有任务的最大服务池长度. 因此整个算法的时间复杂度最大不超过  $O(n^2 \times M)$ .

4.3 实例说明

为说明 DBL 的求解过程, 下面用实例说明算法的求解过程. 以图 1 给出的 workflow 应用为例, 设用户给定截止期  $\delta_n = 100$ . 各任务从表 1 给出的服务池中选择最小执行时间的服务. 图 1 结点上方的数值表示任务的最小执行时间, 关键路径为  $\{1, 2, 6, 7, 9, 12, 13, 15, 16\}$ , 下界完成时间  $L_{CT} = 61$ . 按定义 2 将任务划分为 BL 分组, 结果如表 2(a)所示.

表 2 DBL 算法应用于图 1 实例时各步骤得到的数值

(a) BL 列表

BL 号	任务
8	{1}
7	{2}
6	{3, 6}
5	{7}
4	{4, 9}
3	{5, 8, 10, 11, 12}
2	{13, 14}
1	{15}
0	{16}

(b) DBL 算法参数	
参数	值
$L_{CT}$	61
$BL_{\min}$	62
$\delta_n$	100
$T_{ws}$	38
$T_{LS}$	5.4
$T_{total}$	83
$C_{total}$	533

(c) 各任务的开始时间、截止时间及服务选择

任务	$\beta_i$	$\delta_i$	$S_{ik}$	任务	$\beta_i$	$\delta_i$	$S_{ik}$
2	0	11.4	$S_{21}$	9	33.2	43.6	$S_{91}$
3	0	17.8	$S_{31}$	10	33.2	59.0	$S_{10,2}$
4	0	43.6	$S_{41}$	11	33.2	59.0	$S_{11,1}$
5	0	59.0	$S_{51}$	12	43.6	59.0	$S_{12,1}$
6	11.4	17.8	$S_{61}$	13	59.0	84.4	$S_{13,1}$
7	17.8	33.2	$S_{71}$	14	59.0	84.4	$S_{14,2}$
8	43.6	59.0	$S_{81}$	15	84.4	100.0	$S_{15,1}$

用式(11)得到  $BL_{\min} = 62$ . 因为  $\delta_n \geq BL_{\min}$ , 用式(13)确定任务的开始时间和截止时间. 逆向分层浮差  $T_{ws} = 100 - 62 = 38$ , 按平均分配原则得到层浮差  $T_{LS} = 38 / (9 - 2) = 5.4$  (第一层和最后一层是虚组, 计算中不考虑). 各值计算结果如表 2(b)所示. 按式(13)和约束(7)得到任务的开始时间、截止时间以及

服务选择如表 2(c)所示. 得到的工作流完工时间为  $T_{\text{total}}=83$ , 满足截止期约束, 总费用是各任务所需费用之和, 即  $C_{\text{total}}=533$ .

#### 4.4 算法的进一步讨论

DBL 算法针对 DAG 表示的网格工作流应用而设计, 即算法中仅考虑了顺序和并联结构的工作流结构, 一般工作流<sup>[15]</sup>通常带有选择、循环等结构. 这些结构按一定策略预处理后, DBL 算法也可适用. 方法如下: (1) 选择结构处理. DBL 算法是将活动按逆向深度分层, 分层截止期的确定依赖于该层的最大完成时间. 因此, 选择结构可看作并联结构, 分层以及截止期分解方法不变. (2) 循环结构处理. 如文献[16]所述, 循环结构通过预先确定最大循环次数转化为顺序结构. 对上述结构预处理后, DBL 就可适用一般工作流.

另外, 由于网络的动态性, 任务在执行期间会遭受一些服务失效、服务策略变更等非预测因素的影响, 工作流执行往往不能再满足用户需求. 因此须建立重调度机制适应动态的网格环境, 增强算法的健壮性. 本文重调度算法的思想是尽量在局部范围内调整任务的候选服务分配, 以保证算法的效率并满足用户需求. 已知 DBL 将工作流截止期转化为任务的可执行时间窗口. 当某个任务所选服务发生异常情况时, 算法可在其时间窗口内选择其他候选服务. 若候选服务不存在, 可对其后继层活动更新候选服务, 使这些活动在后继层的截止期前完成, 并且所需费用最小. 否则逐层递推, 直到条件满足为止. 因此, 截止期分解的费用优化策略能较好地处理运行中的服务失效、服务不可用等一些异常情况. 有关重调度机制的研究将是下一步工作的重点.

### 5 实验结果

为比较 DBL 算法、MCP 算法和 DTL 算法, 对大量不同 DAG 结构的工作流应用进行模拟测试. 随机 DAG 图生成器需设置任务数  $V$  和任务最大出度  $\text{out\_degree}$  等参数, 通过组合不同参数的取值生成不同特征的 DAG 图, 实验中结点数  $|V| \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ , 图的最大出度  $\text{out\_degree} \in \{1, 3, 5, |V|\}$ . 任务需要的服务提供者数量服从均匀分布(均值等于 10), 取值范围在区间  $[5, 15]$  内; 服务的执行时间分别在  $[5, 50]$ ,  $[50, 100]$  区间随机产生, 价格为 2000 除以服务执行时间, 以模拟不同粒度和不同 QoS 值服务. 每种参数组合产

生 10 个不同实例, 截止期以工作流下界完工时间为准, 每个实例按 10% 递增取 10 个不同截止期, 实验结果为所有实例的平均值.

需说明的是, DTL 算法中 TL 分组后的工作流最小完工时间  $TL_{\min}$  同样不再是  $L_{CT}$ , 而是每个分组的最大完工时间之和, 所以  $L_{CT} \leq TL_{\min}$ . 当  $L_{CT} \leq \delta_n < TL_{\min}$  时, TL 分组也无效. 本文采用与 DBL 相同方法修改 DTL, 即当  $L_{CT} \leq \delta_n < TL_{\min}$  时, 采用 MCP 算法保证 DTL 在  $\delta_n \geq L_{CT}$  范围内均有效.

表 3 是 3 个算法在不同任务数量下工作流执行费用情况(3 个算法都有很高的运行效率, 其平均运行时间不到 1 ms, 表中不再比较).

表 3 不同算法的费用情况比较

任务数	MCP	DTL		DBL	
	费用	费用	提高率/%	费用	提高率/%
10	1005	905	9.98	823	18.15
20	2173	2069	4.82	1949	10.30
30	3199	2612	18.32	2324	27.35
40	4083	3277	19.72	2899	28.99
50	5637	4706	16.52	4179	25.86
60	7157	6138	14.23	5383	24.78
70	7965	6682	16.10	5885	26.10
80	8484	6717	20.82	5984	29.46
90	10132	8150	19.56	7089	30.03
100	11070	9283	16.14	8154	26.34
平均			15.62		24.74

由于 MCP 算法以  $L_{CT}$  作为工作流截止期约束确定任务开始时间和截止时间, 只对非关键任务结点优化费用, 没有利用可延迟的截止期约束, 故得到的工作流执行费用明显偏高. 而 DBL 算法和 DTL 算法采用不同分组方法并充分利用时间浮差进一步优化费用, 所以得到的工作流执行费用较低. 从表中还可看出 DBL 算法的费用优化效果明显优于 DTL 算法. 其原因在于 DBL 充分利用 DAG 图中任务间的并行和同步完成特性, 合理设置任务开始时间和截止时间, 使每个任务的优化区间较大. 相对 MCP 算法, DTL 费用平均降低 15.62%, 而 DBL 的费用平均降低约 24.74%, 即 DBL 相对 DTL 算法性能平均提高约 9.12%, 这对于总费用通常较大的工作流应用而言, 节省的费用相当可观.

### 6 其它参数对算法性能的影响

#### 6.1 截止时间对算法性能的影响

截止期  $\delta_n$  是用户期望的工作流最晚完工时间, 通常情况下,  $\delta_n$  越大, 工作流花费越小, 即不同截止期对算法将产生不同的影响. 图 2 是包含 200 个任



务的工作流实例所需费用随有效截止期的变化情况.

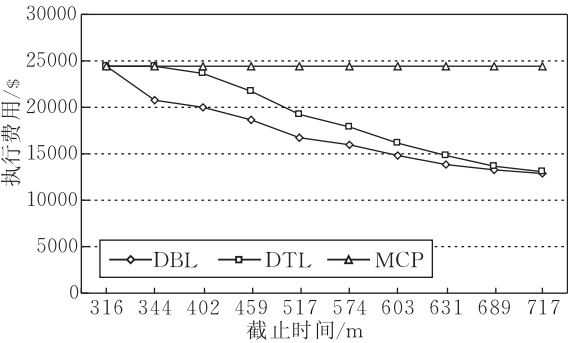


图 2 算法的执行费用随截止时间的变化情况  
( $n=200, L_{CT}=287, TL_{\min}=382, BL_{\min}=340$ )

由图 2 可以看出,随着截止期的增加,DBL 和 DTL 两个算法所需费用都减少,而 MCP 算法不随截止期的变化而变化.因为 MCP 算法以  $L_{CT}$  作为 workflow 执行截止期,对不同的有效截止期约束,算法的完工时间不会改变;而 DBL 和 DTL 两个算法将时间浮差分配到每个任务上并进行优化费用.当  $\delta_n < TL_{\min}$  且  $\delta_n < BL_{\min}$  时,三算法使用相同的起止时间设置策略,所需费用值相同(图的左侧起始位置);随着  $\delta_n$  的增大,DTL 和 DBL 分别采用 TL 和 BL 分组设置任务的时间窗口,所需费用明显降低.

DTL 算法与 DBL 算法相比:当  $\delta_n$  取值最初超过  $TL_{\min}$  或  $BL_{\min}$  时,DBL 的费用优化效果明显优于 DTL 算法;当  $\delta_n$  增大到一定程度时,两算法的费用值趋于相同.其原因在于 BL 基于任务同步完成特性分组,而 TL 基于任务同步开始特征分组.在  $\delta_n$  相对较小但超过  $TL_{\min}$  和  $BL_{\min}$  时,任务具有明显的同步完成特征,BL 分组及截止期分解方法的优势也明显,DBL 所需费用较低;当  $\delta_n$  逐渐变大时,分给任务的起止区间相对较大,松弛时间增加,任务的同步特征逐渐减小,BL 分组优势不再具有明显优势,两算法的费用优化效果相似.

6.2 分组参数对算法性能的影响

从 6.1 节分析可以看到, $BL_{\min}$  和  $TL_{\min}$  是 DBL 和 DTL 算法的两个重要参数,有效截止期  $\delta_n$  落入不同的区间范围,算法会采用不同的起止时间确定方法. $L_{CT}, TL_{\min}, BL_{\min}, \delta_n$  之间的关系如图 3 所示.

当  $\delta_n < BL_{\min}$  ( $TL_{\min}$ ) 时,两个分组算法均以公式(8)确定任务开始时间和截止时间,即用 MCP 算法计算费用优化解.从第 5 节的讨论结果看,MCP 算法的优化效率远不及分组后的优化效率.因此, $BL_{\min}$  ( $TL_{\min}$ ) 到  $L_{CT}$  的距离对算法性能会产生重要影响.由此定义分层最小完工时间增加比:  $R_{BL} =$

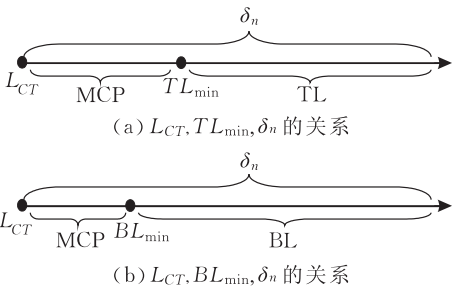


图 3 几个时间参数之间的关系

$\frac{(BL_{\min} - L_{CT})}{L_{CT}} \times 100\%$  ( $R_{TL} = \frac{(TL_{\min} - L_{CT})}{L_{CT}} \times 100\%$ ).  $R_{BL}$  ( $R_{TL}$ ) 取值越小,对算法的平均性能影响越小.表 4 是分组参数随任务数的变化情况比较.

表 4  $TL_{\min}$  和  $BL_{\min}, R_{TL}$  和  $R_{BL}$  的比较

任务数	$L_{CT}$	$TL_{\min}$	$BL_{\min}$	$R_{TL}$	$R_{BL}$
10	43	44	43	2.33	0
20	71	79	77	11.27	8.45
30	91	102	95	12.09	4.40
40	115	128	121	11.30	5.22
50	113	129	123	14.16	8.85
60	132	150	142	13.64	7.58
70	136	155	148	13.97	8.82
80	160	185	180	15.63	12.50
90	166	191	186	15.06	12.05
100	194	224	216	15.46	11.34
平均				12.49	7.92

从表 4 列出的数据可看出,两个参数随任务数量增加而增大,而且相同任务数情况下  $TL_{\min}$  的平均取值均比  $BL_{\min}$  大.  $TL_{\min}$  与  $L_{CT}$  最大时间增加比为 15.46%,最小是 2.33%,平均增加为 12.49%;而  $BL_{\min}$  较接近  $L_{CT}$ ,最大时间增加长度为 12.50%,最小是 0,平均增加 7.92%.由于 DBL 算法中任务开始时间依赖于其前驱任务的最大截止时间,而不是其前驱组的截止时间(通常大于等于前驱任务的截止时间).因此任务的完成时间也会较早,相应的分组截止时间也会减小,即  $BL_{\min} \leq TL_{\min}$ .然而由于 DAG 图结构的复杂性和不同服务粒度,也可能会出现  $BL_{\min} > TL_{\min}$  的工作流实例.实验统计结果表明出现这种情况的几率在 15% 以内,平均情况下  $BL_{\min} < TL_{\min}$ .

综上,DBL 取 MCP 作为优化解的概率比 DTL 算法要小.即 DBL 比 DTL 适应更大的有效截止期范围.而且当用户给定的截止期  $\delta_n$  平均超过  $L_{CT}$  的 8% 时,DBL 算法有明显优势.

7 结论与展望

针对截止时间约束的工作流费用优化问题,将整个工作流的截止期转化为任务的截止期,提出基

于最小关键路径优化费用的算法 MCP; 基于 DAG 图中任务执行约束特征, 提出截止时间约束的逆向分层调度算法 DBL, 算法采取由后向前的方法计算结点深度并划分为 BL 分组, 使同步完成特征的任务划分到同一组中; 组内任务的开始时间由其前驱任务确定, 截止时间相同(为该组中的任务最大完成时间与该组的组时间浮差之和); 改变了 DTL 算法中为每个分组规定相同开始时间的策略, 克服了正向分层的缺点。大量模拟实验表明 DBL 的平均性能优于 MCP 和 DTL, DBL 将 MCP 的平均费用降低了 24.74%, 比 DTL 的 15.62% 有较大幅度改进。同时也讨论了分组参数和截止时间等参数对算法的影响。由于分层改变了部分工作流活动的偏序约束, 在有效截止期范围内 DBL 并不能一致改进 MCP 算法和 DTL 算法。当截止期平均超过工作流最小完工时间的 8% 时, DBL 能获得较好的性能。

当然, 还有较多问题值得进一步研究, 如更有效的截止时间确定策略、工作流应用的截止期和总费用的权衡<sup>[17]</sup>、服务故障失效问题的处理等。

## 参 考 文 献

- [1] Foster I, Kesselman C. The Grid: Blueprint for a Future Computing Infrastructure. USA: Morgan Kaufmann Publishers, 1999
- [2] Foster I, Kesselman C, Nick J M, Tuecke S. Grid service for distributed system integration. IEEE Computer, 2002, 35(6): 37-46
- [3] Deelman E, Blythe J et al. Mapping abstract complex workflows onto grid environments. Journal of Grid Computing, 2003, 1(1): 25-39
- [4] Yu J, Buyya R. Taxonomy of scientific workflow systems for Grid computing. Sigmod Record, 2005, 34(3): 44-49
- [5] Blythe J, Jain S, Deelman E et al. Task scheduling strategies for workflow-based applications in grids//Proceedings of the IEEE International Symposium on Cluster Computing and Grid. Cardiff, Wales, UK, 2005: 759-767
- [6] Frey J, Tannenbaum T et al. Condor-G: A computation management agent for multi-institutional grids. Cluster Computing, 2002, 5(3): 237-246
- [7] De P, Dunne E J, Ghosh J B, Wells C E. The discrete time-cost tradeoff problem revisited. European Journal of Operational Research, 1995, 81(2): 225-238
- [8] Demeulemeester E, Herroelen W, Elmaghraby S E. Optimal procedures for the discrete time/cost trade-off problem in project networks. European Journal of Operational Research, 1996, 88(1): 50-68
- [9] De P, Dunne E J, Ghosh J B, Wells C E. Complexity of the discrete time—cost tradeoff problem for project networks. Operations Research, 1997, 45(2): 302-306
- [10] Akkan C, Drexler A, Kimms A. Network decomposition-based benchmark results for the discrete time—cost tradeoff problem. European Journal of Operational Research, 2005, 165(2): 339-358
- [11] Buyya R, Abramson D, Giddy J, Stockinger H. Economic models for resource management and scheduling in grid computing. Concurrency and Computation: Practice and Experience Journal (Special Issue on Grid Computing Environments), 2002, 14(13-15): 1507-1542
- [12] Abramson D, Buyya R, Giddy J. A computational economy for grid computing and its Implementation in the Nimrod-G resource broker. Future Generation Computer Systems (FGCS) Journal, 2002, 18(8): 1061-1074
- [13] Lin M, Lin Z X. A cost-effective critical path approach for service priority selections in grid computing economy. Decision Support Systems, 2006, 42(3): 1628-1640
- [14] Yu J, Buyya R, Tham C K. Cost-based Scheduling of workflow applications on utility grids//Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing. Melbourne, Australia: IEEE Press, 2005: 140-147
- [15] Senkul P, Toroslu H I. An architecture for workflow scheduling under resource allocation constraints. Information Systems, 2005, 30(5): 399-422
- [16] Zeng L Z, Benattallah B, Ngu A H H, Dumas M, Kalagnanam J, Chang H. QoS-aware middleware for Web services composition. IEEE Transactions on Software Engineering, 2004, 30(5): 311-327
- [17] Jin Hai, Chen Han-Hua, Lu Zhi-Peng, Ning Xiao-Min. QoS optimizing model and solving for composite service in CGSP job manager. Chinese Journal of Computers, 2005, 28(4): 578-588(in Chinese)  
(金海, 陈汉华, 吕志鹏, 宁小敏. CGSP 作业管理其合成服务的 QoS 优化模型及求解. 计算机学报, 2005, 28(4): 578-588)



LI Xiao-Ping, born in 1970, Ph. D., associate profes-

YUAN Ying-Chun, born in 1970, Ph. D. candidate, associate professor. Her main research interests include grid computing and service composition.

sor, Ph. D. supervisor. His main research interests include machine scheduling, project scheduling, service computing.

WANG Qian, born in 1946, professor, Ph. D. supervisor. Her main research interests include information integration, database technology and CSCW.

ZHANG Yi, born in 1970, Ph. D. candidate. His main research interests include machine scheduling, service computing.



## Background

The aim of this work is to implement the efficient scheduling of workflow applications in computational Grids. This work is part of National Natural Science Foundation of China under Grant "the research and implement for AMS data computational environment". Grid computing has emerged as a promising distributed computing paradigm for solving large-scale computational and data intensive problems in science, engineering and commerce. This project intends to implement efficiently the storage, management, access and operation of AMS massive data. In order to meeting users' requirements, the resource management and application scheduling are the key problems in Grids. For the scheduling of workflow applications represented by Directed Acyclic Graph (DAG), which widely exist in different scientific domains such as bioinformatics and astronomy, Buyya et al. have proposed some constructive heuristics (e. g. DTL) for solving cost optimization with deadline constraints. The work still focuses on the cost optimization problem for workflows with deadline constraints. A heuristic called Deadline Bottom Level (DBL)

is proposed. By analyzing the properties of DAG, a new concept called Bottom Depth (BD) is defined. Those tasks with same bottom depth are divided into a level, which comprehensives the parallel and synchronization properties of DAG. Thus, the overall deadline can be transformed into the time intervals of all tasks. All tasks can select the appropriate service resources to minimize the total cost with their time intervals. Experimental results show that DBL can save 9.12% average cost of DTL. Moreover, the heuristic DBL can also applied to general workflow applications considering choice and iteration structure if pre-processing strategies is executed. In addition, the deadline division strategy based on level is very suitable for dynamic grid environment. The scheduler may re-adjust unexecuted task level's sub-deadlines level-by-level and re-compute their optimal schedules. Thus, the minimum number of tasks is only considered to reallocation other candidate services. This work is also supported by the National Natural Science Foundation of China under grants No. 60672092 and No. 60504029.