

# 基于任务复制的分簇与调度算法

何 琨<sup>1),2)</sup> 赵 勇<sup>2)</sup> 黄文奇<sup>1)</sup>

<sup>1)</sup> (华中科技大学计算机科学与技术学院 武汉 430074)

<sup>2)</sup> (华中科技大学控制科学与工程系 武汉 430074)

**摘 要** 针对并行与分布式系统中相关任务的静态调度问题,以最小化调度长度为主要目标,以减少资源数为次要目标,对待复制的重要祖先集定义了新的选择策略,提出了基于任务复制的动态关键前驱调度算法,改进了粒度的定义,证明了对任意 DAG,算法有优于前人的性能下界.实验结果优于典型任务复制算法,特别是对经典 EZ 算例的解(调度长度为 8)好于前人认为的理论最优解(调度长度为 8.5),并证明了新的解为最优解.定义了 DAG 的补图,讨论了不允许任务复制时树型 DAG 的 2-优度算法.

**关键词** 任务复制;任务分簇;调度算法;DAG;任务粒度

中图法分类号 TP301

## A Clustering and Scheduling Algorithm Based on Task Duplication

HE Kun<sup>1),2)</sup> ZHAO Yong<sup>2)</sup> HUANG Wen-Qi<sup>1)</sup>

<sup>1)</sup> (College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074)

<sup>2)</sup> (Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074)

**Abstract** This paper addresses an important and classic scheduling problem, the static scheduling of dependent tasks in homogeneous environment. It is NP hard even when the resources are unbounded, and finds many applications in the parallel and distributed computation area. Dependent tasks are usually denoted by Directed Acyclic Graph (DAG), and solving heuristics are commonly categorized to priority list based, cluster based and task duplication based schemes. Task-duplication-based (TDB) algorithms are of better performance than non-duplication ones. A new TDB clustering and scheduling algorithm, called the dynamic critical predecessor (DCP) algorithm, is proposed in this paper. DCP algorithm defines a new selective strategy for important ancestors to be duplicated. The primary aim is to get the shortest schedule length, and the next is to utilize as less resources as possible. Based on an improved definition of granularity, DCP algorithm achieves a better performance guarantee for arbitrary DAG than relative works reported in the literature. Experimental results on several benchmarks show that DCP algorithm is quite effective and it exceeds other classic TDB algorithms. Especially for the classic EZ benchmark, DCP algorithm gets an optimal solution with 8 makespan, which is better than the optimal result taken for before with 8.5 makespan. Complement graph of a DAG is defined, and a similar algorithm is developed to produce a 2-optimal schedule for tree graph if task duplication is not allowed for the tasks.

**Keywords** task duplication; task clustering; scheduling algorithm; DAG; task granularity

收稿日期:2005-11-08;最终修改稿收到日期:2007-12-20. 本课题得到国家自然科学基金(70471077)、国家“九七三”重点基础研究发展规划项目基金(2004CB318000)和中国博士后科学基金(20070420174)资助. 何 琨,女,1972年生,博士,主要研究方向为分布式计算、NP 难问题现实求解和算法优化. E-mail: brooklet60@gmail.com. 赵 勇,男,1967年生,博士,教授,博士生导师,主要研究领域为决策分析、复杂系统建模与优化. 黄文奇,男,1938年生,教授,博士生导师,主要研究领域为 NP 难问题现实求解和算法优化.

## 1 引言

调度相关任务以最小化完工的时间问题是并行与分布式系统中的一个重要问题. 相关任务可表示大量的数值计算、公式推导、 workflow 项目、并行算法等, 如快速傅立叶变换、高斯估计的计算. 通过将任务分解并分配到多个资源(如处理机)上并行执行, 可大幅提高执行效率. 即使资源同构、充分, 多且任务允许被复制, 此类问题仍是 NP 完全的<sup>[1]</sup>. 同构问题是研究异构问题的基础, 且对于求解其它的单模项目调度问题有参考价值, 因此得到了广泛的研究.

本文研究同构问题的一个典型模型, 即已知一组有优先关系约束的任务和一组相同数量的同构资源, 求一种调度方案, 满足以下约束:

(1) 任务约束. 每个任务需要分配到一个资源上执行, 任务的执行时间给定且在所有的资源上相同; 任务是原子的, 其执行过程不可中断.

(2) 链路约束. 有优先关系约束的两个任务, 只有前一任务完成且生成的数据传输到后一任务所在资源, 后一任务才能开始; 传输时间仅与两任务间的传输量有关, 若两个任务在同一资源上执行, 则传输延迟为 0; 否则为一给定的值.

(3) 资源约束. 分配到同一资源上的任务, 其执行时间不能重叠.

目标为如何将任务分配到资源, 并给出任务在资源上的开始时间, 使任务集的最早开始时间与最晚完成时间之间的差尽可能的小, 即最小化调度长度 (*makespan*). 此类问题的一个典型应用为分布式内存机器中的任务调度问题.

一般用加权的有向无回路图(Directed Acyclic Graph, DAG)  $G(V, E, \mu, \lambda)$  来表示相互依赖的任务集, 如图 2(a) 所示, 顶点集  $V = \{1, 2, \dots, n-1, n\}$  为任务集, 顶点的权  $\mu_i$  表示相应任务的执行时间, 有向边集  $E = \{e_{ij} : i, j \in V; i \rightarrow j\}$  表示任务间的优先关系, 边的权  $\lambda_{ij}$  表示前后两任务不在同一资源上执行时的传输延迟.

目前代表性的调度方法为启发式算法<sup>[1-10]</sup>, 主要包括基于任务复制的调度<sup>[1-7]</sup>、基于优先级列表的调度<sup>[8-9]</sup>和基于簇的调度<sup>[10]</sup>三类. 基于任务复制(Task-Duplication-Based, TDB)的调度算法通常要优于基于优先级列表和基于簇的调度算法<sup>[1, 11]</sup>. TDB 算法的理论基础是采取以空间换时间的策略, 通过冗余分配任务到多个资源以减少通信开销, 从而减

少总的调度长度, 因此可生成更小的调度长度. 如何准确地确定应被复制的重要任务是获得较短调度的关键, 各种 TDB 算法的主要区别也正在于此.

近年来发表的 TDB 算法包括 DSH、BTDH、LCTD、LWB、PY、MJD<sup>[1]</sup>、CPFD<sup>[2]</sup>、TDS<sup>[4]</sup>、ETDS<sup>[5]</sup>、PPA<sup>[6]</sup>、IREA<sup>[7]</sup>等. PY 和 MJD 是其中理论上最优的, 它们对任意 DAG 给出了性能保证, 而其它仅部分算法对某些特定 DAG 生成了优化调度. 对任意 DAG, PY 调度的调度长度最多为最优调度的 2 倍, MJD 调度的调度长度最多为最优调度的  $1 + \epsilon'$  倍 ( $0 < \epsilon' \leq 1$ ). MJD 算法的主要思想是以拓朴顺序计算每个任务  $v$  的开始时间下界  $e_v$  及其对应簇  $C_v$ , 然后反拓朴顺序访问 DAG 中的任务并构造调度. 在  $e_v$  的计算过程中, 数据到达时间  $c$  的计算未到任务  $v$ , 而是进入簇  $C_v$  的第一个任务, 可能错失更小调度的机会; 且未考虑其祖先得到  $e$  值的对应簇信息, 导致对任务的实际最早可能开始时间的估计不够准确.

本文提出一种基于任务复制的动态关键前驱(Dynamic Critical Predecessor, DCP) 调度算法, 并证明对任意 DAG, DCP 调度的调度长度最多为最优调度的  $1 + \epsilon$  倍 ( $0 < \epsilon \leq 1$  且  $\epsilon \leq \epsilon'$ ). 针对实际应用中处理机数目受限的情况, 算法在不增加调度长度的同时, 尽可能减少占用的资源数目. 实验表明 DCP 算法的求解质量很高.

## 2 DCP 调度算法

针对 MJD 算法的不足, DCP 算法主要做了以下改进: (1) 计算数据最早到达时间时, 不是计算到入簇的第一个任务的时间, 而是计算  $v$  的最早可能开始时间  $S_v$ . (2) 放宽加入任务的条件, 只要加入该任务不会使  $S_v$  增加, 则可以加入簇  $C_v$ . (3) 计算  $S_v$  时, 其祖先任务含相关的分簇历史信息, 即在考虑跨簇的任务时, 不仅以任务为粒度, 也以簇为粒度, 从而更准确, 并加快了计算过程.

### 2.1 计算 $S_v$ 的值

首先按拓朴顺序访问图  $G$  中的任务  $v \in V$ , 并计算其最早可能开始时间  $S_v$  及对应簇  $C_v$ . 对任务  $v$ , 设其祖先的  $S$  值及对应簇已求出, 欲求  $S_v$  及对应簇  $C_v$ , 只需找到一个包含  $v$  及其部分祖先的簇就足够了, 因为如果某任务不是  $v$  的祖先, 则从簇中移除该任务, 不会使  $S_v$  增加, 只可能使其减小.

设  $C_v$  是一个含任务  $v$  及其部分祖先  $W = \{w_1,$

$\omega_2, \dots, \omega_l\}$  的簇,  $U = \{u_1, u_2, \dots, u_k\}$  为跨簇  $C_v$  的任务.  $W$  和  $U$  中任务的  $S$  及对应簇  $C$  已经求出. 将  $S_i$  作为任务  $i$  的释放时间, 按就绪任务释放时间的先后顺序对集合  $W \cup U \cup \{v\}$  进行贪心调度, 得到  $v$  的最早可能开始时间  $S_v$ , 并记下造成  $S_v$  不能再下降的关键任务  $u$ , 称  $u$  为  $C_v$  的关键前驱  $c_{pred}(C_v)$ .

然后寻找使  $S_v$  尽可能小的簇  $C_v$ , 从只含任务  $v$  的簇开始, 每次向簇中并入  $v$  的部分祖先集, 并检查  $S_v$  是否减小. 部分祖先集的选择, 考虑了  $C_v$  的关键前驱  $u$  及其对应簇  $C_u$ , 具体过程见图 1. 算法 1 为任务的  $S_v$  及对应簇  $C_v$  的计算过程, 其中  $C.s$  和  $C.c_{pred}$  表示当前格局贪心调度后的  $S_v$  和簇  $C$  的关键前驱,  $v.s$  和  $v.C$  表示终止格局时的  $S_v$  和  $C_v$ ,  $w$  表示最后并入簇  $C$  的任务.

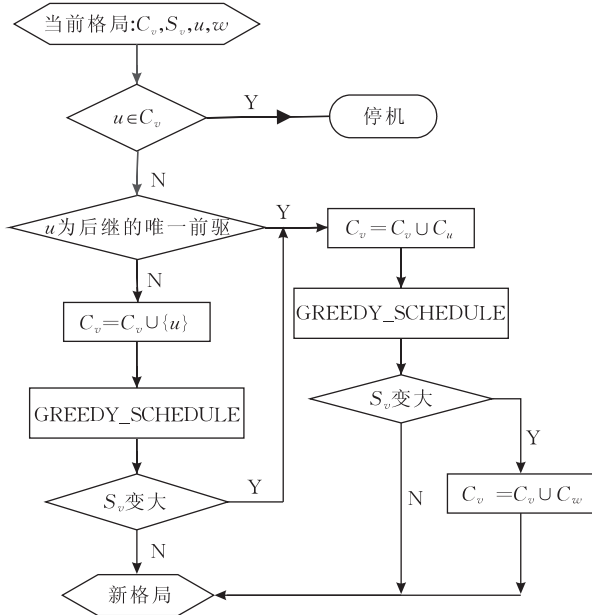


图 1 格局演化过程

#### 算法 1. COMPUTE\_S\_VALUE( $v, G$ ).

1. begin
2.  $C' \leftarrow \{v\}$ ;
3.  $C' \leftarrow \text{GREEDY\_SCHEDULE}(C', G)$ ;
4.  $s' \leftarrow C'.s$ ;  $s \leftarrow C'.s$ ;
5.  $u \leftarrow C'.c_{pred}$ ;
6.  $w' \leftarrow \emptyset$ ; (初值为空)
7. 循环, do while ( $s' \leq s$ ) {
8.  $w \leftarrow w'$ ;  $s \leftarrow s'$ ;  $C \leftarrow C'$ ;
9. 若  $u \notin C$  {
10. 若  $u$  为其后继的唯一父任务, 则 {
11.  $C' \leftarrow C \cup u.C$ ;  $w' \leftarrow \emptyset$ ;
12.  $C' \leftarrow \text{GREEDY\_SCHEDULE}(C', G)$ ;
13. } 否则 {
14.  $C' \leftarrow C \cup u$ ;  $w' \leftarrow u$ ;

15.  $C' \leftarrow \text{GREEDY\_SCHEDULE}(C', G)$ ;
16. 若  $C'.s > C.s$ , 则 {
17.  $C' \leftarrow C \cup u.C$ ;  $w' \leftarrow \emptyset$ ;
18.  $C' \leftarrow \text{GREEDY\_SCHEDULE}(C', G)$ ;
19. }
20. }
21.  $s' \leftarrow C'.s$ ;
22.  $u \leftarrow C'.c_{pred}$ ;
23. } 否则 {
24. 退出循环;
25. }
26. } //end 循环
27. 若  $w$  不为空, 则  $C \leftarrow C \cup w.C$ ;
28.  $v.s \leftarrow s$ ;  $v.C \leftarrow C$ ;
29. 返回  $v$ ;
30. end COMPUTE\_S\_VALUE.

## 2.2 构造调度

得到每个任务  $v \in V$  的最早可能开始时间  $S_v$  及对应簇  $C_v$  后, 反拓扑顺序访问  $G$  中的任务并进行分簇.

#### 算法 2. CLUSTERING( $G$ ).

1. 初始化: 结束任务为标识任务,  $\phi(G)$  为空;
2. 循环, 直到没有标识任务为止 {
3. 取出一个标识任务  $v$  并考虑其对应簇  $C_v$  {
4. 若在  $\phi(G)$  中找不到一个簇包含  $C_v$  的所有任务和边, 则 {
5. 循环, 对  $\phi(G)$  中的每个簇  $C_i$  {
6. 判断  $C_v$  与  $C_i$  合并是否会导致结束任务  $n$  的  $S_n$  增加,
7. 若  $S_n$  增加, 则  $C_v$  加入簇集  $\phi(G)$ ,
8. 否则合并簇  $C_v$  到  $C_i$ , 退出循环;
9. }
10. }
11. 否则利用已有的簇, 修改边的关联任务和关联簇;
12. }
13. 取消对  $v$  的标识并标识所有跨  $C_v$  簇的任务.
14. }
15. end CLUSTERING.

分簇完成后, 将  $\phi(G)$  中的每个簇分配到不同的资源上, 它们之间用跨簇的边集关联, 然后按任务的最早可能开始时间非降序排列进行贪心调度, 得到最终结果.

## 3 实验结果

### 3.1 EZ 算例

EZ 算例是 DAG 调度的经典算例, 如图 2(a) 所

示,其最优调度的调度长度一直被认为是  $8.5^{[4]}$ . 图 2(b)为 DCP 的调度结果,调度长度为 8,使用了 3 个资源. 图 2(c)为每个任务的  $S$  值及其对应簇,图 2(d)为任务  $T_6$  的  $S$  和  $C$  的计算过程. DCP 与相

关算法的结果比较见表 1,其中 EZ、DSC、TDS 算法的结果参见文献[4],DCP 的算法复杂度说明参见文献[12]. 下面证明 EZ 算例的最优调度的调度长度为 8.

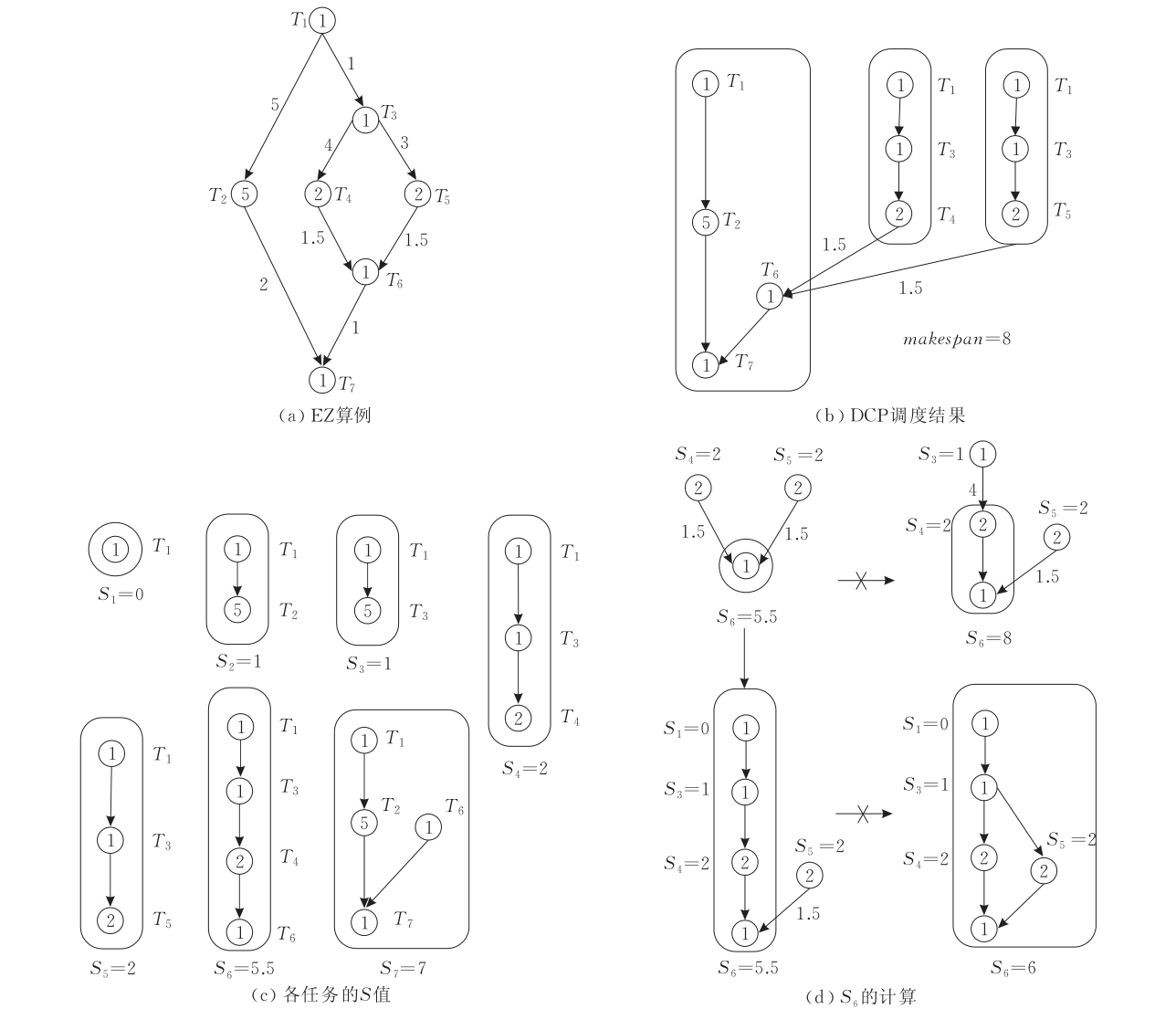


图 2

表 1 对 EZ 算例的结果比较		
算法	调度长度	算法复杂度
优化算法	8	NP 完全
EZ	10	$O( V ( V + E ))$
DSC	9	$O(\lg  V ( V + E ))$
MJD	12.5	$O( V ( V \lg  V + E ))$
TDS	8.5	$O( V ^2)$
DCP	8	$O( V ( V \lg  V + E ))$

**事实 1.** EZ 算例的最优调度的调度长度为 8.

证明. 因任务仅当其前驱任务均已完成才可能执行,所以在 DCP 的调度结果中,任务  $T_1, T_2, T_3, T_4, T_5$  均已在其绝对最早可能开始时间被调度.

由于  $T_1 \rightarrow T_2 \rightarrow T_7$  为不考虑边的权时的最长路径,所以  $T_7$  的最早开始时间不小于 6,完成时间不小于 7.

若  $T_1$  与  $T_2$  不在同一簇中,则  $T_2$  的最早完成时间为 11,故  $T_7$  的最早完成时间为 12,不可能为优化调度. 故优化调度中,  $T_1$  与  $T_2$  必在同一簇中.

若  $T_7$  与  $T_2$  不在同一簇中,则完成时间不小于 9,故优化调度中,  $T_7$  一定与  $T_2$  在同一簇中.

考虑任务  $T_6$ ,只有两种可能:(1)与  $T_2, T_7$  在一个簇中,则  $T_6$  的最早可能开始时间为 6,  $T_7$  的最早可能完成时间为 8;(2)与  $T_2, T_7$  不在一个簇中,则  $T_6$  的最早可能开始时间为 5.5,  $T_7$  的最早可能完成

时间为 8.5.

可见 EZ 算例最优调度的调度长度为 8. 证毕.

### 3.2 CFPD 算例集

DCP 算法与相关的 6 个 TDB 算法比较的结果见表 2, 其中算例来源及其它 TDB 算法的计算结果参见文献[2]. 可见 DCP 的解至少与其它 TDB 算法的最好解一样好, 且若干算例得到了更优的解.

表 2 TDB 算法的调度结果比较

DAG	调度长度						
	BTDH	DSH	LCTD	LWB	PY	CPFD	DCP
Ahmad&Kwok	22	22	22	25	27	20	20
Al-Mouhamed	35	35	37	37	39	35	35
Colin&Chretienne	13	13	15	12	14	12	12
Kruatrachue&Lewis	361	361	462	761	369	361	361
Wu&Gajski	320	320	390	390	420	320	320
Yang&Gerasoulis	16	16	18	18	22	16	15
Gerasoulis&Yang	16	16	20	17	26	16	8

## 4 性能分析

本节从理论上分析 DCP 算法的性能, 改进粒度的定义, 并证明 DCP 对任意 DAG 可得到优于 MJD 的性能下界.

粒度是分析 DAG 调度算法性能的一个重要参数. 对任务  $v \in V$ , 定义

$$g_1(v) = \frac{\min\{\mu_u \mid (u, v) \in E\}}{\max\{\lambda_{uv} \mid (u, v) \in E\}},$$

$$g_2(v) = \frac{\min\{\mu_w \mid (v, w) \in E\}}{\max\{\lambda_{vw} \mid (v, w) \in E\}},$$

则  $v$  的粒度  $g_0(v) = \min\{g_1(v), g_2(v)\}$ , 图  $G$  的粒度  $g_0(G) = \min\{g_0(v) \mid v \in V\}^{[1]}$ .

本文改进了粒度的定义: (1) 只考虑前驱及输入边, 不考虑后继及输出边; (2) 取前驱及相应输入边的比值中的最小值, 而不取最小前驱与最大输入边的比值. 对任务  $v \in V$ , 定义  $v$  的粒度为  $g(v) = \min\{\mu_u / \lambda_{uv}, (u, v) \in E\}$ , 图  $G$  的粒度为  $g(G) = \min\{g(v) \mid v \in V\}$ . 例如对 EZ 算例,  $g(G) = g_0(G) = 0.2$ ; 若修改  $T_1$  的计算时间为 4, 则  $g_0(G') = 0.2$ ,  $g(G') = 0.25$ .

**引理 1.** 对任意 DAG 图  $G(V, E, \mu, \lambda)$ , 存在  $0 < \epsilon \leq 1$  满足  $g(G) = (1 - \epsilon) / \epsilon$ .

证明. 因  $\mu, \lambda$  中元素均为有理数, 由粒度的定义,  $g(G)$  为有理数运算得到, 由有理数四则运算的封闭性,  $g(G)$  也为一有理数, 故一定可写为一个分数, 设  $g(G) = a/b$ , 其中  $a, b \in N = \{0, 1, 2, \dots\}$ , 且

$b > 0$ , 则  $0 \leq g(G) < \infty$ . 定义  $\epsilon = b/(a + b)$ , 可推出  $0 < \epsilon \leq 1$ , 且  $g(G) = (1 - \epsilon) / \epsilon$ .

引理得证.

证毕.

**引理 2.** 设  $v$  为一标识任务, 且其对应簇  $C_v$  映射到同一资源, 则存在  $0 < \epsilon \leq 1$  满足  $g(G) = (1 - \epsilon) / \epsilon$ , 使  $st_v \leq (1 + \epsilon) st_{\text{opt}}(v)$ . 其中  $st_v$  为  $v$  的开始时间,  $st_{\text{opt}}(v)$  为  $v$  在最优调度时的开始时间.

证明. 由引理 1, 存在  $0 < \epsilon \leq 1$  满足  $g(G) = (1 - \epsilon) / \epsilon$ . 下面证明  $st_v \leq (1 + \epsilon) st_{\text{opt}}(v)$ , 以标识任务  $v$  的深度为递推值, 进行数学归纳证明:

(1) 若  $v$  为开始任务,  $st_v = st_{\text{opt}}(v) = 0$ , 命题成立;

(2) 设命题对  $v$  的祖先中的标识任务均成立. 设另一个簇的标识任务  $u$  为任务  $w \in C_v$  和簇  $C_v$  的关键任务, 则  $u$  为  $w$  的所有前驱中数据到达最晚的前驱, 所以  $st_w = st_u + \mu_u + \lambda_{uw}$ ; 由于祖先任务  $u$  为一标识任务, 由假设  $st_u \leq (1 + \epsilon) st_{\text{opt}}(u)$ , 所以

$$st_w \leq (1 + \epsilon) st_{\text{opt}}(u) + \mu_u + \lambda_{uw}.$$

因  $w$  为  $u$  的后继,  $st_{\text{opt}}(w) \geq st_{\text{opt}}(u) + \mu_u$ . 因此,

$$st_w \leq st_{\text{opt}}(u) + \mu_u + \epsilon \cdot st_{\text{opt}}(u) + \lambda_{uw},$$

$$st_w \leq st_{\text{opt}}(w) + \epsilon \cdot st_{\text{opt}}(u) + \lambda_{uw} \quad (1)$$

由粒度定义和引理 1,

$$\mu_u / \lambda_{uw} \geq g(w) \geq g(G) = (1 - \epsilon) / \epsilon,$$

$$\epsilon(\mu_u + \lambda_{uw}) \geq \lambda_{uw} \quad (2)$$

由式(1)、式(2)得

$$st_w \leq st_{\text{opt}}(w) + \epsilon(st_{\text{opt}}(u) + \mu_u + \lambda_{uw}),$$

而  $st_{\text{opt}}(v) \geq st_{\text{opt}}(u) + \mu_u + \lambda_{uw}$ , 可推出

$$st_w \leq st_{\text{opt}}(w) + \epsilon st_{\text{opt}}(v) \quad (3)$$

式(3)对任意属于  $C_v$  的任务成立, 所以  $st_v \leq (1 + \epsilon) st_{\text{opt}}(v)$ .

引理得证.

证毕.

**定理 1.** 对任意 DAG 图  $G(V, E, \mu, \lambda)$ , 存在  $0 < \epsilon \leq 1$  满足  $g(G) = (1 - \epsilon) / \epsilon$ , 且 DCP 调度结果最多为最优调度的  $1 + \epsilon$  倍.

证明. 设  $M_{\text{opt}}$  为  $G$  的最优调度结果, 对  $G$  中所有无后继的点  $v$ ,  $M_{\text{opt}} \geq \max\{st_{\text{opt}}(v) + \mu_v\}$ . 因每个无后继的点均为标识任务, 由引理 2:

$$\begin{aligned} makespan(G) &\leq \max\{(1 + \epsilon) st_{\text{opt}}(v) + \mu_v\} \\ &\leq \max\{(1 + \epsilon) [st_{\text{opt}}(v) + \mu_v]\} \\ &\leq (1 + \epsilon) \max\{st_{\text{opt}}(v) + \mu_v\} \\ &\leq (1 + \epsilon) M_{\text{opt}}. \end{aligned}$$

调度结果最多为最优调度的  $1 + \epsilon$  倍. 证毕.

**定理 2.** 对任意 DAG, DCP 算法的性能下界优于 MJD 算法的性能下界.

证明. 由

$$\min \left\{ \frac{\mu_u}{\lambda(u,v)}, (u,v) \in E \right\} \geq \frac{\min \{ \mu_u \mid (u,v) \in E \}}{\max \{ \lambda(u,v) \mid (u,v) \in E \}},$$

得出  $g(v) \geq g_1(v)$ , 而  $g_1(v) \geq g_0(v)$ , 故  $g(v) \geq g_0(v)$ , 因此  $g(G) \geq g_0(G)$ .

由引理 1, 存在  $0 < \epsilon \leq 1$  满足  $g(G) = (1 - \epsilon) / \epsilon$ ; 由文献[1], 存在  $0 < \epsilon' \leq 1$  满足  $g_0(G) \geq (1 - \epsilon') / \epsilon'$ ; 可推出  $(1 - \epsilon) / \epsilon \geq (1 - \epsilon') / \epsilon'$ , 即  $\epsilon \leq \epsilon'$ .

对任意 DAG 图  $G(V, E, \mu, \lambda)$ , 由定理 1, DCP 算法的  $makespan(G) \leq (1 + \epsilon) M_{opt}$ ; 由文献[1], MJD 算法的  $makespan(G) \leq (1 + \epsilon') M_{opt}$ . 而  $\epsilon \leq \epsilon'$ , 故 DCP 算法的性能下界优于 MJD 算法的性能下界.

因此命题成立.

证毕.

## 5 相关讨论

在实际应用中, 某些任务不允许被复制, 如银行取款任务, 而典型的 DAG 图 Out-tree 和 In-tree 应用非常广泛, 如分而治之 (divide-and-conquer) 算法、常见的数值计算等. 本节讨论不允许复制时 DAG 的特点, 并证明存在多项式时间的算法, 对 Out-tree 和 In-tree 的调度结果最多为最优调度的  $1 + \epsilon$  倍且  $0 < \epsilon \leq 1$ .

**定义 1**(In-tree). 除了唯一的结束任务, 每个任务只有一个子任务的 DAG.

**定义 2**(Out-tree). 除了唯一的开始任务, 每

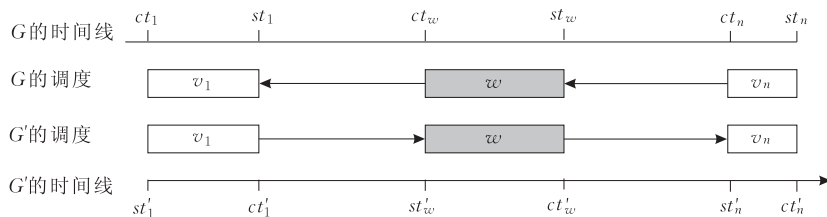


图 3  $G$  和  $G'$  的调度示意图

**定理 3.** 不允许复制时, 存在一多项式时间的算法对 Out-tree 和 In-tree 的调度结果最多为最优调度的  $1 + \epsilon$  倍且  $0 < \epsilon \leq 1$ .

证明.

(1) 由定理 1, 算法 DCP 可对 In-tree 生成最多为最优调度的  $1 + \epsilon$  倍的解. 而 In-tree 的每个任务最多只有一个子任务, 由 DCP 算法, 实际上没有任务被复制. 故不允许复制时, DCP 对 In-tree 的 DAG 可生成  $1 + \epsilon$  优度的解.

(2) 任意 Out-tree 的补图为 In-tree, 对补图  $G'$  调度得到解  $S'$ , 并定义原图  $G$  的调度为  $st_v(S) =$

个任务只有一个父任务的 DAG.

**定义 3**(补图  $G'$ ). 对任意给定的 DAG 图  $G$ , 其补图  $G'$  与  $G$  的唯一不同为  $G'$  中每条边的方向与  $G$  中对应边  $u \rightarrow w$  的方向正好相反, 为  $w \rightarrow u$ .

**引理 3.** 不允许复制时, 对任意给定的 DAG 图  $G$  及其补图  $G'$ , 设  $S'$  为  $G'$  的一个调度, 可得到  $G$  的一个调度  $S$ :  $S$  中每个任务  $v$  的匹配资源与  $S'$  一致, 且开始时间为  $st_v(S) = makespan(S') - ct_v(S')$ , 则  $G$  的调度  $S$  为一合理的调度, 且与其补图  $G'$  的调度  $S'$  有相同的调度长度.

证明. 首先证明  $S$  为一合理的调度. 设  $G'$  开始任务为  $v_1$ , 结束任务为  $v_n$ , 在调度  $S'$  中的开始时间与结束时间为  $st'_1, ct'_1, st'_n$  和  $ct'_n$ . 设任意  $w \in V$  在  $G'$  的调度  $S'$  中的开始时间和结束时间分别为  $st'_w$  和  $ct'_w$ . 则在  $G$  的调度  $S$  中:

$$st_n = makespan(S') - ct'_n = ct'_n - ct'_n = 0,$$

$$ct_n = st_n + \mu_n = \mu_n,$$

$$st_w - st_n = st_w = makespan(S') - ct'_w = ct'_n - ct'_w,$$

$$ct_w - st_n = ct_w = st_w + \mu_w = ct'_n - (ct'_w - \mu_w) = ct'_n - st'_w.$$

可见在  $S$  中对  $w$  的调度满足约束关系, 为一合理的调度.

$$st_1 = makespan(S') - ct'_1 = ct'_n - ct'_1,$$

$$makespan(S) = ct_1 = st_1 + \mu_1 = ct'_n - (ct'_1 - \mu_1) =$$

$$ct'_n - st'_1 = makespan(S').$$

$G$  的调度  $S$  与补图  $G'$  的调度  $S'$  有相同的调度长度. 引理得证. 证毕.

$makespan(S') - ct_v(S')$ . 则  $G$  中没有任务被复制, 且由引理 3,  $S$  为一合理的调度, 与其补图的调度  $S'$  有相同的调度长度.

不允许复制时, 原图与补图的最优解的调度长度是相同的. 所以上面构造的 Out-tree 的解最多为最优调度的  $1 + \epsilon$  倍.

因此命题成立.

证毕.

## 6 结 论

算法 DCP 以跨簇任务的最早可能开始时间为

相应任务的释放时间进行贪心调度,并选择导致当前任务最早可能开始时间不能再减少的关键任务或关键簇为待复制的重要任务集.在理论与实践上,DCP 算法均优于典型的 TDB 算法,对任意 DAG 有优于前人的性能下界.因此,DCP 算法对高性能应用的静态调度是一个较优的选择.无任务复制时调度典型 DAG 图 In-tree 和 Out-tree 的  $1+\epsilon$  优度的算法对不允许任务复制的树型应用有很好的参考价值.

同构问题起源于分布式内存机器中相互依赖多任务的优化调度问题,并在多处理机、机群等系统中得到进一步研究,相互依赖的多任务可以是 workflow 项目、并行算法等.本文的研究适用于带宽足够大,不考虑通信竞争造成的延迟的同构环境.网络环境为异构的情况在现实中更为常见,如何修改 DCP 算法以解决异构环境下的 DAG 静态调度问题有待作进一步的研究.

参 考 文 献

[1] Palis M A, Liou J C, Wei D S L. Task clustering and scheduling for distributed memory parallel architectures. *IEEE Transactions on Parallel and Distributed Systems*, 1996, 7(1): 46-55

[2] Ahmad I, Kwok Y K. On exploiting task duplication in parallel program scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 1998, 9(9): 872-892

[3] Kwok Y K, Ahmad I. Benchmarking and comparison of the task graph scheduling algorithms. *Journal of Parallel and Distributed Computing*, 1999, 59(3): 381-422

[4] Darbha S, Agrawal D P. Optimal scheduling algorithm for distributed-memory machines. *IEEE Transactions on Parallel*

and Distributed Systems, 1998, 9(1): 87-95

[5] Park C I, Choe T Y. An optimal scheduling algorithm based on task duplication. *IEEE Transactions on Computers*, 2002, 51(4): 444-448

[6] Zhou Shuang-E, Yuan You-Guang, Xiong Bing-Zhong, Ou Zhong-Hong. An algorithm of processor pre-allocation based on task duplication. *Chinese Journal of Computers*, 2004, 27(2): 216-223(in Chinese)

(周双娥,袁由光,熊兵周,欧中红.基于任务复制的处理器预分配算法. *计算机学报*, 2004, 27(2): 216-223)

[7] He Kun, Zhao Yong, Chen Yang. Analysis and solutions for multitasks scheduling in distributed environments. *Systems Engineering — Theory and Practice*, 2007, 27(5): 119-125 (in Chinese)

(何琨,赵勇,陈阳.分布式环境下多任务调度问题的分析与求解. *系统工程理论与实践*, 2007, 27(5): 119-125)

[8] Radulescu A, Van Gemund A J C. Low-cost task scheduling for distributed-memory machines. *IEEE Transactions on Parallel and Distributed Systems*, 2002, 13(6): 648-658

[9] Shi Wei, Zheng Wei-Min. The balanced dynamic critical path scheduling algorithm of dependent task graphs. *Chinese Journal of Computers*, 2001, 24(9): 991-997(in Chinese)

(石威,郑伟民.相关任务图的均衡动态关键路径调度算法. *计算机学报*, 2001, 24(9): 991-997)

[10] Yang T, Gerasoulis A. DSC: Scheduling parallel tasks on an unbounded number of processors. *IEEE Transactions on Parallel and Distributed Systems*, 1994, 5(9): 951-967

[11] Bajaj R, Agrawal D P. Improving scheduling of tasks in a heterogeneous environment. *IEEE Transactions on Parallel and Distributed Systems*, 2004, 15(2): 107-118

[12] He Kun. Research on the multi-tasks scheduling problem [Ph. D. dissertation]. Wuhan: Huazhong University of Science and Technology, 2006(in Chinese)

(何琨.多任务调度问题的研究与实现[博士学位论文].武汉:华中科技大学,2006)



**HE Kun**, born in 1972, Ph. D. . Her research interests include distributed computing, NP hard problem solving and algorithm optimization.

**ZHAO Yong**, born in 1967, Ph. D. , professor, Ph. D. supervisor. His research interests include decision analyzing, complex system modeling and optimization.

**HUANG Wen-Qi**, born in 1938, professor, Ph. D. supervisor. His research interests include NP hard problem solving and algorithm optimization.

Background

This work is supported by the National Natural Science Foundation of China under grant No.70471077 and by the National Basic Research Program of China (973 Program)

under grant No.2004CB318000.

The problem of static scheduling dependent tasks in homogeneous environment for minimizing the task makespan

is a classic scheduling problem in the parallel and distributed computation area. The problem is NP-hard in theory and finds many applications in practice. It has attracted a considerable amount of attention over the years, and various approaches have been developed to handle it efficiently. Dependent tasks are usually denoted by Directed Acyclic Graph (DAG), and heuristics are the representative solving methods. The commonly categorized schemes are priority list based, cluster based and task duplication based schemes. In general, task duplication schemes (TDB) are observed to be better than the other two. Among the TDB algorithms reported in the literature, PY algorithm and MJD algorithm are theoretically best for the reason that they give performance guarantees for arbitrary DAG, while just some of the others give performance guarantees for certain DAG. For arbitrary DAG, PY could produce a schedule at most 2 times the optimal, and MJD could produce a schedule at most  $1+\epsilon'$  times

the optimal ( $0<\epsilon'\leq 1$ ).

This paper proposes a new efficient approach, the dynamic critical predecessor (DCP) algorithm, for solving the problem. Based on an improved granularity definition proposed in this paper, a proof is given that DCP algorithm can produce a schedule at most  $1+\epsilon$  times the optimal and  $0<\epsilon\leq \epsilon'$  for arbitrary DAG. Experimental results show that DCP algorithm exceeds other classic TDB algorithms. Especially on the classic EZ benchmark, DCP algorithm achieves an optimal scheduling with 8 makespan. The result is better than the optimal result taken for before with 8.5 makespan. Further, theoretical proof is given that the makespan of EZ benchmark is 8 for its optimal scheduling. Finally, complement graph of a DAG is defined, and a similar algorithm is developed to produce a 2-optimal schedule for graphs like In-tree and Out-tree if task duplication is not allowed for the tasks.