$P_4|fix|C_{max}$ 问题的最优规则调度算法

黄金贵 李荣珩
(湖南师范大学计算机教学部 长沙 410081)

摘 要 多处理机任务调度问题 $P_m|fix|C_{max}(m\geq 3)$是典型的强 NP 难问题,由于其在并行环境中的实际意义而受到越来越多的关注.但在一般情形下,寻求该问题的较为理想的近似算法是极其困难的,通常从较少处理机数的系统着手研究.对于 $m=4$ 的情形,文中研究了 $P_4|fix|C_{max}$的规则调度算法,通过引入组调度技术,给出了该问题的一个线性时间的 4/3-近似算法,并证明了该算法是 4-处理机系统中的最优规则调度算法.

关键词 多处理机任务调度;规则调度;近似算法;NP-难问题
中图法分类号 TP393 **DOI 号**: 10.3724/SP.J.1016.2009.01631

An Optimal Algorithm for Normal Scheduling on $P_4|fix|C_{max}$

HUANG Jin-Gui LI Rong-Heng
(*Department of Computer Education, Hunan Normal University, Changsha* 410081)

Abstract With the advanced of the heterogeneous parallel computing technology, Multiprocessor-job scheduling problem has attracted much attention recently. Because of complexity of the general multiprocessor system, it is impossibility to find the approximation scheduling algorithm with the ideal performance. The paper is focused on the smaller processors system, that 4-processor system and its scheduling problem $P_4|fix|C_{max}$. With introduced the Normal scheduling, and Group scheduling, a linear time algorithm is developed with the 4/3 approximation ratio. It is proved that normal scheduling come from the algorithm is the optimal normal scheduling in 4-processor systems.

Keywords multiprocessor-job scheduling; normal scheduling; approximation algorithm; NP-hard problem

1 引 言

并行计算系统中的任务可能同时依赖于多种资源(或统称为处理机),这类需要多个处理机同时运行的任务被称为多处理机任务.多处理机任务调度问题的研究已开始受到国内外学者越来越多的关注.Hoogeveen[1] 等人证明了该问题是强 NP 难(Strong NP-hard)的,Huang 等人曾证明不存在多项式时间的常数近似比的近似算法[2].对于这样极其困难的问题,人们一般都从少数处理机系统入手进行研究[3-4].本文研究 4-处理机系统中的这类问题.

给定 4-处理机系统 $P=\{p_1,p_2,p_3,p_4\}$(或简记为$\{1,2,3,4\}$)和 n 个依赖于该系统的多处理机任务的集合 $J=\{j_1,j_2,\cdots,j_n\}$,其中任务 $j_i=(M_i,t_i)$,$1\leq i\leq n$, M_i是处理机集合 P 的一个非空真子集,称为该任务的处理机模式,代表任务 j_i运行所需要的处理机资源;t_i是该任务在这些处理机上运行所需要的时间.这些多处理机任务的调度,就是采取优化的执行顺序,使得系统总的时间跨度(最后一个任务

收稿日期:2007-09-07;最终修改稿收到日期:2009-06-01.本课题得到国家自然科学基金(60872039,10771060)资助.**黄金贵**,男,1964 年生,博士,副教授,研究方向为网络优化及软件过程技术,包括网络并行计算、路由算法、可信网络、资源管理与调度、NP 难问题近似优化、小参数计算、软件过程技术等. E-mail: hjg@hunnu.edu.cn. **李荣珩**,男,1963 年生,博士,教授,研究领域为排序、组合优化及计算理论.

的完成时刻)尽可能地小.为了简化问题,暂不考虑优先约束、时间延迟等,也不允许剥夺调度.这类调度问题被归为 $P_4|fix|C_{\max}$[5].设该问题的最优调度 S_{OPT},并以最短时间跨度为调度目标,为此,要寻求多项式时间的近似调度 S_{APX},使得近似比 $R=S_{APX}(J)/S_{OPT}(J)$尽可能接近 1.

为了简化问题,Blazewicz[6]等提出了规则调度方法,即将具有相同处理机模式的任务合并到一起连续调度.文献[7]解决了 $P_3|fix|C_{\max}$的最优规则调度问题,而对于 4 处理机系统,Chen 和 Lee[8]给出了近似比为 2 的线性时间规则调度算法,文献[3]将其改进为 3/2.本文提出基于组调度的规则调度方法,解决 4-处理机系统的最优规则调度问题.

2 规则任务与组调度

对于 4-处理机系统中的任意任务集 J,最多可得到 $2^4-2=14$ 个规则任务,记为 $J_M=(M,t_M)$,其中指标 M 为该规则任务的处理机模式,t_M为该规则任务的执行时间.这 14 个规则任务可以根据需要的处理机个数分为 3 类:

1-处理机任务:J_1,J_2,J_3,J_4;

2-处理机任务:$J_{12},J_{13},J_{14},J_{23},J_{24},J_{34}$;

3-处理机任务:$J_{123},J_{134},J_{124},J_{234}$.

将 10 个规则 2-处理机任务和 3-处理机任务按其互补关系分组,得到 7 个独立的组(Group),用 G_M表示这些组,其中 M 为 3-处理机任务的模式,或者是 2-处理机任务组中最大一个任务的模式,即

$$G_M=\begin{cases}\{J_M\}, & |M|=3\\ \{J_M,J_{\{1,2,3,4\}-M}\}, & |M|=2\text{ 且 }t_M\geq t_{\{1,2,3,4\}-M}\end{cases} \tag{1}$$

在不引起混淆的情况下,G_M既表示该组任务的集合,也表示该组任务的时间跨度.

定义组调度(Group scheduling)如下:依次调度式(1)中的若干组,保证同组中的任务或者是同时开始、或者是同时结束.显然在组调度中,任何两个来自不同组的任务都是不可并行的,而且组与组之间的顺序不影响调度的时间跨度,于是得到最优调度时间跨度的一个下界 LB(Lower Bound).

引理 1. 令

(1) $T_i(i=1,2,3,4)$为系统要完成所有任务时处理机 p_i实际总执行时间量,即 $T_i=\sum_{i\in M}t_M$;

(2) $T_0=t_{123}+t_{124}+t_{134}+t_{234}+\max\{t_{12},t_{34}\}+\max\{t_{14},t_{23}\}+\max\{t_{13},t_{24}\}$;

(3) $LB=\max\{T_i|i=0,1,2,3,4\}$,

则 $S_{OPT}(J)\geq LB$.

下面考察式(1)定义的 7 个组.不失一般性,我们不妨令 $t_{12}\geq t_{34}$ 且 $t_{23}\geq t_{14}$.事实上,若 $t_{12}<t_{34}$,则将处理机的标号 1 与 3 互换,2 与 4 互换,即得 $t_{12}\geq t_{34}$.若 $t_{12}\geq t_{34}$但 $t_{23}<t_{14}$,则将处理机的标号 1 与 2 互换,3 与 4 互换,即得 $t_{12}\geq t_{34}$ 且 $t_{23}\geq t_{14}$.因此,这 7 组可以按 t_{13}与 t_{24}的大小关系分为两类,用 α 标识,当 $t_{13}\geq t_{24}$时,$\alpha=1$(图 1);否则 $\alpha=0$(图 2).

每组 G_M都可能引起一个或两个处理机上的空闲时间段,称为间隙(gap),用 g_M表示:

$$g_M=\begin{cases}t_{P-M}, & |P-M|=3(\text{其中 }B_{P-M}=\{J_{P-M}\})\\ t_{P-M}-t_M, & |M|=2(\text{其中 }B_{P-M}=\{J_M,J_{P-M}\})\end{cases} \tag{2}$$

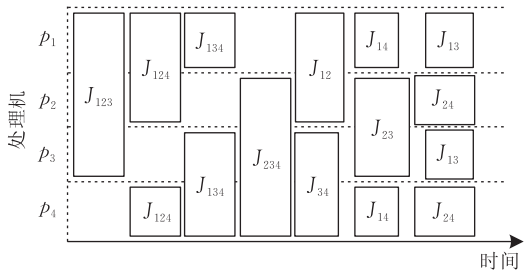

图 1 当 $t_{12}\geq t_{34},t_{23}\geq t_{14},t_{13}\geq t_{24}(\alpha=1)$时的组调度 GS_1

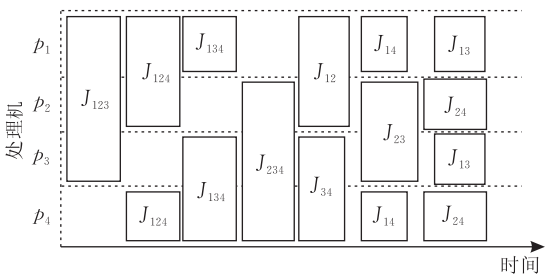

图 2 当 $t_{12}\geq t_{34},t_{23}\geq t_{14},t_{13}<t_{24}(\alpha=0)$时的组调度 GS_0

称由 3-处理机任务引起的间隙为 3-处理机任务间隙,由 2-处理机任务组引起的间隙称为 2-处理机任务间隙.显然 $g_M\leq G_{P-M}$,若 $g_M=0$,这时假定存在一个零间隙.将这些间隙映射到每个处理机上,得知在组调度中每个处理机至多有 4 个间隙,且其中包含至多一个 3-处理机任务间隙.记每个处理机 $p_i(I=1,2,3,4)$上的间隙为 $Gap(i)$,则

$$Gap(i)=\{g_M|i\in M\text{ and }M\in\{\{1\},\{2\},\{3\},\{4\},\{3,4\},\{1,4\},(1-\alpha)\cdot\{1,3\}+\alpha\cdot\{2,4\}\}\} \tag{3}$$

引理 2. $g_1+g_2+g_3+g_4+g_{34}+g_{14}+\alpha g_{24}+(1-\alpha)g_{13}\leq LB$.

证明. 根据间隙的定义式(2)，观察调度图 1 和图 2，再应用引理 1 结论，可知 $g_1+g_2+g_3+g_4+g_{34}+g_{14}+\alpha g_{24}+(1-\alpha)g_{13}\leq t_{123}+t_{124}+t_{134}+t_{234}+t_{12}+t_{23}+\alpha t_{13}+(1-\alpha)t_{24}=T_0\leq LB$. 证毕.

3 基于组调度的规则调度

所谓规则调度(normal scheduling)就是将具有相同处理机模式的任务合并到一起连续调度，即调度规则任务. 基于组调度的规则调度，就是在组调度的基础上加入 4 个 1-处理机规则任务的规则调度. 为了描述组调度的方便，将 4 个 1-处理机任务当成一组，记为 G_0，即

$$G_0=\{J_1,J_2,J_3,J_4\} \tag{4}$$

考察基于组调度 $GS_1(\alpha=1)$ 的规则调度情况，首先重新安排图 1 中的各组之间的顺序，然后在某两组之间调度 1-处理机任务组 G_0. 如图 3 所示，是一种规则调度 NS，且记为 $NS=\{G_{13},G_{123},J_{23},G_0,J_{14},G_{124},G_{12},G_{134},G_{234}\}$，其中组 G_{23} 分解为 J_{23} 和 J_{14}，中间调度 1-处理机任务组 G_0. 容易验证该 α 规则调度 NS 的时间跨度为 $|GS|\leq B+\max\{g_1,g_{24}+g_2,g_{34}\}$.

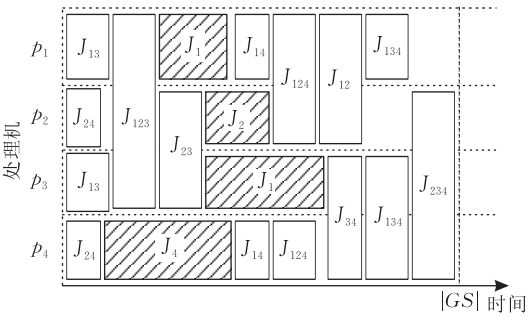

图 3　基于组调度 GS_1 的规则调度 NS

这样一来，我们可以得到若干这样的规则调度，记为 $NS(\alpha,k)$，其中 α 是组调度类型，k 是不同规则调度的序号. 表 1 列出了不同类型组调度下的规则调度，其中 α 类第 k 种调度 $NS(\alpha,k)$ 的时间跨度上界 $|NS(\alpha,k)|\leq B+h(\alpha,k)$.

表 1　各种间隙类型的规则调度 $NS(\alpha,k)$

α	K	$NS(\alpha,k)$	$h(\alpha,k)$
1	1	$\{G_{23},G_{123},J_{13},G_0,J_{24},G_{124},G_{12},G_{134},G_{234}\}$	$\max\{g_1+g_{14},g_2,g_{34}\}$
1	2	$\{G_{23},G_{123},J_{12},G_0,J_{34},G_{134},G_{13},G_{124},G_{234}\}$	$\max\{g_1+g_{14},g_3,g_{24}\}$
1	3	$\{G_{23},G_{124},J_{12},G_0,J_{34},G_{134},G_{13},G_{123},G_{234}\}$	$\max\{g_1+g_{14},g_4+g_{14}+g_{24}\}$
1	4	$\{G_{12},G_{124},J_{24},G_0,J_{13},G_{134},G_{23},G_{123},G_{234}\}$	$\max\{g_1+g_{14},g_4+g_{14}+g_{34}\}$
1	5	$\{G_{123},G_{124},G_{134},G_{234},G_{12},G_0,G_{13},G_{23}\}$	$\max\{g_1+g_{14},g_2,g_3,g_4+g_{14}\}$
1	6	$\{G_{13},G_{123},J_{12},G_0,J_{34},G_{234},G_{23},G_{124},G_{134}\}$	$\max\{g_2+g_{24},g_3,g_{14}\}$
1	7	$\{G_{13},G_{124},J_{12},G_0,J_{34},G_{234},G_{23},G_{123},G_{134}\}$	$\max\{g_2+g_{24},g_4+g_{14}+g_{24}\}$
1	8	$\{G_{12},G_{124},J_{14},G_0,J_{23},G_{234},G_{13},G_{123},G_{134}\}$	$\max\{g_2+g_{24},g_4+g_{34}+g_{24}\}$
1	9	$\{G_{13},G_{123},J_{23},G_0,J_{14},G_{124},G_{12},G_{134},G_{234}\}$	$\max\{g_2+g_{24},g_1,g_{34}\}$
1	10	$\{G_{123},G_{124},G_{134},G_{234},G_{12},G_0,G_{23},G_{13}\}$	$\max\{g_2+g_{24},g_1,g_3,g_4+g_{24}\}$
1	11	$\{G_{12},G_{123},J_{23},G_0,J_{14},G_{134},G_{13},G_{134},G_{234}\}$	$\max\{g_3+g_{34},g_1,g_{24}\}$
1	12	$\{G_{12},G_{123},J_{13},G_0,J_{24},G_{234},G_{23},G_{134},G_{234}\}$	$\max\{g_3+g_{34},g_2,g_{14}\}$
1	13	$\{G_{13},G_{134},J_{14},G_0,J_{23},G_{234},G_{12},G_{123},G_{124}\}$	$\max\{g_3+g_{34},g_4+g_{34}+g_{24}\}$
1	14	$\{G_{12},G_{134},J_{13},G_0,J_{24},G_{234},G_{23},G_{123},G_{124}\}$	$\max\{g_3+g_{34},g_4+g_{34}+g_{14}\}$
1	15	$\{G_{123},G_{124},G_{134},G_{234},G_{12},G_{23},G_0,G_{13}\}$	$\max\{g_3+g_{34},g_1,g_2,g_4+g_{34}\}$
0	1	$\{G_{12},G_{123},J_{23},G_0,J_{14},G_{124},G_{24},G_{134},G_{234}\}$	$\max\{g_1+g_{13},g_2,g_{34}\}$
0	2	$\{G_{23},G_{123},J_{13},G_0,J_{24},G_{124},G_{12},G_{134},G_{234}\}$	$\max\{g_1+g_{14},g_2,g_{34}\}$
0	3	$\{G_{23},G_{123},J_{12},G_0,J_{34},G_{234},G_{24},G_{124},G_{134}\}$	$\max\{g_3+g_{13},g_2,g_{14}\}$
0	4	$\{G_{12},G_{123},J_{13},G_0,J_{24},G_{234},G_{23},G_{134},G_{234}\}$	$\max\{g_3+g_{34},g_2,g_{14}\}$
0	5	$\{G_{24},G_{124},J_{12},G_0,J_{34},G_{234},G_{23},G_{123},G_{134}\}$	$\max\{g_4+g_{14},g_2,g_{13}\}$
0	6	$\{G_{12},G_{124},J_{14},G_0,J_{23},G_{234},G_{24},G_{123},G_{134}\}$	$\max\{g_4+g_{34},g_2,g_{13}\}$
0	7	$\{G_{123},G_{124},G_{134},G_{234},G_{12},G_0,G_{23},G_{24}\}$	$\max\{g_1+g_{13},g_2,g_3+g_{13},g_4\}$
0	8	$\{G_{123},G_{124},G_{134},G_{234},G_{12},G_0,G_{24},G_{23}\}$	$\max\{g_1+g_{14},g_2,g_3,g_4+g_{14}\}$
0	9	$\{G_{123},G_{124},G_{134},G_{234},G_{12},G_{23},G_0,G_{24}\}$	$\max\{g_1,g_2,g_3+g_{34},g_4+g_{34}\}$
0	10	$\{G_{23},G_{123},J_{12},G_0,J_{34},G_{134},G_{24},G_{124},G_{234}\}$	$\max\{g_1+g_{13}+g_{14},g_3+g_{13}\}$
0	11	$\{G_{12},G_{124},J_{24},G_0,J_{13},G_{134},G_{23},G_{123},G_{234}\}$	$\max\{g_4+g_{14}+g_{34},g_1+g_{14}\}$
0	12	$\{G_{12},G_{134},J_{14},G_0,J_{23},G_{234},G_{24},G_{123},G_{124}\}$	$\max\{g_3+g_{13}+g_{34},g_4+g_{34}\}$
0	13	$J_2,J_{13},J_{14},G_{134},G_{12},J_1,J_3,J_4,J_{23},J_{24},G_{234},G_{123},G_{124}$	$\max\{t_{13}+t_{14}+t_{123},t_2-t_{134},t_{14}+t_{24}+t_{124}\}$
0	14	$J_2,J_{13},J_{34},G_{134},G_{23},J_1,J_3,J_4,J_{24},J_{12},G_{124},G_{123},G_{234}$	$\max\{t_{13}+t_{34}+t_{234},t_2-t_{134},t_{13}+t_{12}+t_{123}\}$
0	15	$J_2,J_{34},J_{14},G_{134},G_{24},J_1,J_3,J_4,J_{12},J_{23},G_{123},G_{234},G_{124}$	$\max\{t_{34}+t_{14}+t_{124},t_2-t_{134},t_{34}+t_{23}+t_{234}\}$

最后 3 种 $NS(0,13)$,$NS(0,14)$,$NS(0,15)$比较特殊,将组调度中的组做了较多的拆分. 下面以 $NS(0,13)$为例来说明,如图 4 所示,$NS(0,13)=\{J_2, J_{13}, J_{14}, G_{134}, G_{12}, J_1, J_3, J_4, J_{23}, J_{24}, G_{234}, G_{123}, G_{124}\}$.

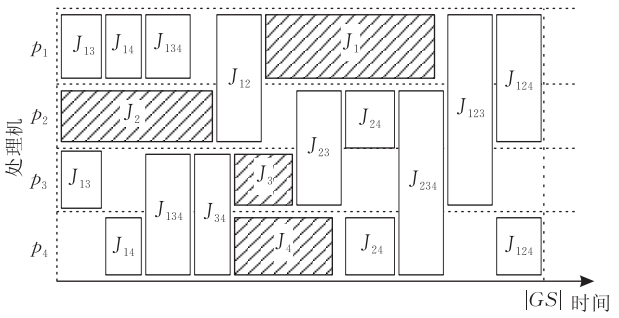

图 4 规则调度 $NS(0,13)$

为了计算该规则调度的时间跨度$|NS(0,13)|$,将图 4 转换为如下的有向图(图 5),图的每个结点表示一个规则任务,有向边表示任务的先后关系. 调度的时间跨度就是该有向图的关键路径(即最大路径),即

$$
\begin{aligned}
&|NS(0,13)|=\\
&\quad \max\{t_{13}+t_{14}+t_{134}+t_{12}+t_{23}+t_{24}+t_{234}+t_{123}+t_{124},\\
&\quad t_2+t_{12}+t_1+t_{123}+t_{124},\\
&\quad t_2+t_{12}+t_{23}+t_{24}+t_{234}+t_{123}+t_{124},\\
&\quad t_{13}+t_{14}+t_{134}+t_{34}+t_3+t_{23}+t_{24}+t_{234}+t_{123}+t_{124},\\
&\quad t_{13}+t_{14}+t_{134}+t_{34}+t_4+t_{24}+t_{234}+t_{123}+t_{124}\}\\
&=\max\{t_{13}+t_{14}+T_0, t_2-(t_{13}+t_{14}+t_{134})+T_1,\\
&\quad T_2, t_{14}+t_{24}+t_{124}+T_3, t_{13}+t_{123}+T_4\}\\
&\le LB+\max\{t_{13}+t_{14}, t_2-t_{134}, t_{14}+t_{24}+t_{124}, t_{13}+t_{123}\}\\
&\le LB+\max\{t_{13}+t_{14}+t_{123}, t_2-t_{134}, t_{14}+t_{24}+t_{124}\}.
\end{aligned}
$$

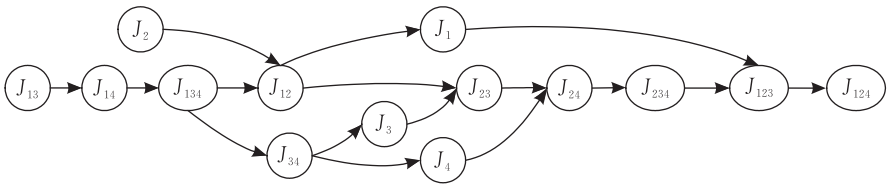

图 5 规则调度 $NS(0,13)$的调度图

引理 3. 对任意 $\alpha\in\{0,1\}$,必存在 $k\in\{1,2,3,\cdots,15\}$,使得 $h(\alpha,k)\le LB/3$.

证明.

(1) 当 $\alpha=1$ 时,根据引理 2,在 g_1+g_{14},g_2+g_{24},g_3+g_{34}中必有一个不超过 $LB/3$. 不妨设 $g_1+g_{14}\le LB/3$,考虑 $k\in\{1,2,3,4,5\}$. 若 $h(1,1)>LB/3$ 且 $h(1,2)>LB/3$,即 $\max\{g_2,g_{34}\}>LB/3$ 且 $\max\{g_3,g_{24}\}>LB/3$,则根据引理 2 易知,在 $g_4+g_{14}+g_{24}$,$g_4+g_{14}+g_{34}$及 $\max\{g_2,g_3,g_4+g_{14}\}$中必有一个不超过 $LB/3$. 即证得当 $\alpha=1$ 时引理成立.

(2) 当 $\alpha=0$ 时,分 $g_2\le LB/3$ 和 $g_2>LB/3$ 两种情况讨论.

①$g_2\le LB/3$ 时,考察 $h(0,k)$,$k\in\{1,2,\cdots,9\}$. 根据引理 2 可知,g_{13}、g_{14}、g_{34}中最多只有两个超过 $LB/3$. 若 g_{13}、g_{14}、g_{34}中有两个超过 $LB/3$,不妨设为 g_{13}和 g_{14},此时由引理 2 易知,$h(0,9)\le LB/3$;若 g_{13}、g_{14}、g_{34} 中有且仅有一个超过 $LB/3$,不妨设 $g_{13}>LB/3$,则由引理 2 易知,$h(0,2)\le LB/3$ 或 $h(0,4)\le LB/3$;若 g_{13}、g_{14}、g_{34}都不超过 $LB/3$,则由引理 2 可得,$h(0,1)$、$h(0,4)$、$h(0,5)$中必有一个不超过 $LB/3$.

②$g_2>LB/3$ 时,考察 $h(0,k)$,$k\in\{10,11,\cdots,15\}$. 若 $g_1+g_{13}+g_{14}$、$g_4+g_{14}+g_{34}$、$g_3+g_{13}+g_{34}$中至少有一个不超过 $LB/3$,不妨设 $g_1+g_{13}+g_{14}\le LB/3$,则根据引理 2 可知,在 $g_4+g_{14}+g_{34}$、g_3+g_{13}中必有一个不超过$\le LB/3$,即有 $h(0,10)\le LB/3$ 或 $h(0,11)\le LB/3$.

若$g_1+g_{13}+g_{14}>LB/3$ 且$g_4+g_{14}+g_{34}>LB/3$ 且 $g_3+g_{13}+g_{34}>LB/3$,则此时 $g_1+g_2+g_3+g_4+2(g_{13}+g_{14}+g_{34})>(4/3)LB$,结合引理 2,有 $g_{13}+g_{14}+g_{34}>LB/3$,从而有

$$t_{134}+t_{24}+t_{23}+t_{12}-t_{13}-t_{14}-t_{34}>(2/3)LB \quad (5)$$

但由引理 1,$t_{123}+t_{124}+t_{134}+t_{234}+t_{12}+t_{23}+t_{24}=T_0\le LB$,所以

$$t_{123}+t_{124}+t_{234}+t_{13}+t_{14}+t_{34}\le LB/3 \quad (6)$$

考察 $h(0,k)$,$k\in\{13,14,15\}$. 由引理 1,$t_2-t_{134}+t_{134}+t_{12}+t_{23}+t_{24}\le T_2\le LB$,结合式(5)即得 $t_2-t_{134}\le LB/3$;又根据引理 2 可知,在 $t_{14}+t_{24}+t_{124}$、$t_{13}+t_{12}+t_{123}$、$t_{34}+t_{23}+t_{234}$中必有一个不超过 $LB/3$;再结合式(6)即可得知,$h(0,13)$、$h(0,14)$及 $h(0,15)$中必有一个不超过 $LB/3$. 证毕.

4 最优规则调度算法

根据上文的分析,可以设计一个规则调度算法 NSA(Normal Scheduling Algorithm),基本思路是,适当选择两组或多组与 4 个 1-处理机任务块进行混合调度,而剩下的组实行组调度. 算法具体描述如下.

算法 1.　NSA.

1. 系统调度预处理：
   1.1. 将原始任务规则化得到规则任务集 $J=\{J_1, J_2, J_3, J_4, J_{12}, J_{23}, J_{13}, J_{14}, J_{24}, J_{34}, J_{123}, J_{234}, J_{134}, J_{124}\}$；
   1.2. 若 $t_{12}<t_{34}$，则交换处理机 p_1 和 p_3 的编号，p_2 和 p_4 的编号；
   1.3. 若 $t_{23}<t_{14}$，则交换处理机 p_1 和 p_2 的编号，p_3 和 p_4 的编号.
2. 调度初始化：
   2.1. 若 $t_{13}\geq t_{24}$，则令 $\alpha=1$；否则令 $\alpha=0$；
   2.2. 将规则任务中的 2-处理机任务和 3-处理机任务按(*)分组得 G_M；
   2.3. 计算最优调度的下界 LB 以及每组包含的间隙长度 g_{P-M}；
3. 确定规则调度：
   3.1. 计算 $h(\alpha,i), i=1,2,3,\cdots,15$；
   3.2. 若 $h(\alpha,k)=\min\{h(\alpha,i)\,|\,i=1,2,3,\cdots,15\}$，则选择规则调度 $NS(\alpha,k)$；
4. 实施规则调度 $NS(\alpha,k)$；
5. End.

算法 NSA 的时间复杂度和性能分析有如下定理.

定理 1.　算法 NSA 具有 $O(n)$时间复杂度，近似比性能为 4/3，即 $S_{NSA}(J)/S_{OPT}(J)\leq 4/3$.

证明.　算法的时间复杂度是显然的，下面证明算法的性能. 根据算法步 3 的 3.2 可知，

$S_{NSA}(J)\leq LB+\min\{h(\alpha,i)\,|\,i=1,2,3,\cdots,15\}$，

但由引理 3，$\min\{h(\alpha,i)\,|\,i=1,2,3,\cdots,15\}\leq LB/3$. 又根据引理 1，$S_{OPT}(J)\geq LB$，所以

$S_{NSA}(J)/S_{OPT}(J)\leq(LB+LB/3)/LB=4/3$，

即定理结论成立.　　证毕.

定理 2.　4-处理机系统中，对任意的任务实例 J，由算法 NSA 构造出的规则调度是最优的规则调度.

证明.　如果能找到一个最优规则调度的例子，使得其近似比恰为 4/3，则可以说明 4-处理机系统中的任何规则调度算法的近似比都不会小于 4/3，从而证明本文规则调度算法 NSA 是 4-处理机系统中的最优规则调度算法.

为此，取一个包含 9 个原始任务的实例 $J=\{j_1^1, j_1^2, j_3^1, j_3^2, j_4^1, j_4^2, j_{12}, j_{23}, j_{24}\}$，其中下标表示该任务的处理机模式，上标表示该任务在相同处理机模式的任务中的序号，且每个任务的处理时间均为 1. 该实例有一个最优调度 S_{OPT}，如图 6 所示，且其时间跨度为 $S_{OPT}(J)=3$.

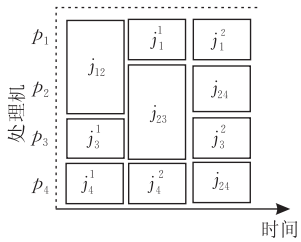

图 6　$J=\{j_1^1, j_1^2, j_3^1, j_3^2, j_4^1, j_4^2, j_{12}, j_{23}, j_{24}\}$的最优调度

实例 J 的规则任务集为 $J=\{J_1, J_3, J_4, J_{12}, J_{23}, J_{24}\}$，其中下标表示该规则任务的处理机模式，各规则任务的处理时间分别为 2、2、2、1、1、1. 容易得知对于该规则任务实例，存在最优规则调度(Optimal Normal Schedule) S_{ONS}，如图 7 所示，且调度的时间跨度为 $S_{ONS}(J)=4$.

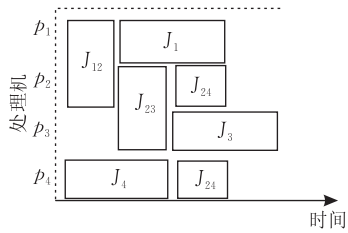

图 7　一个最优规则调度的例子

由此得到实例 J 的最优规则调度的近似比为 $S_{ONS}(J)/S_{OPT}(J)=4/3$. 这说明，对于 4-处理机系统中的多处理机任务调度问题，使用规则调度时的最好性能结果也不会好于 4/3. 而本文算法 NSA 已经达到了 4/3，因此定理 2 结论成立.　　证毕.

本节给出的定理 1 和定理 2，是本文的主要结论，并得到了严格的理论证明. 最后将给出实际验证测试，主要验证两点：(1) 规则调度算法 NSA 对任何实例的近似比不超过 4/3；(2) 在所有的规则调度算法中，本文给出的算法 NSA 是最优的.

为了严格起见，本文对于算法的近似性能都是在最坏的情况下进行比较的. 在定理 2 的证明中已经给出了一个最坏情况的实际例子，说明了无论采取何种规则调度，都不可能得到小于 4/3 的近似比，因此第(2)点无需再验证. 针对第(1)点，我们在双核 PC 服务器上利用 VC++进行了 1000 次模拟实验，每次实验中包括 1000 个任务，任务的处理机模式和时间长度都是随机给出的，得到的结果数据如表 2 所示.

表 2　算法 NSA 的实验结果数据列表

算法	处理机数	实验次数	每次实验任务数	每次规则任务数	平均花费时间/μs	最大近似比	平均近似比	最好近似比
NSA	4	1000	1000	≤14	0.839	1.3333	1.1957	1.0000

在实验中,为了不影响调度结果而简化调度,我们去掉了模式是 4 个处理机的任务.近似比计算所依赖的最优时间跨度取引理 1 给出的下界.算法时间主要花费在对 1000 个任务的规则化,得到的 14 个规则任务,如果进行全排列选择最优规则调度需要进行 $14!\approx 8.7\times 10^{10}$ 轮调度,但在本文算法 NSA 中根据规则任务间隙特征和参数的取值,只需在 15 种调度中选择即可,所以花费时间很小.实验结果显示,在给定的最坏情况下,近似比不超过 4/3,平均花费时间 0.839μs.由于实验的次数毕竟有限,不能列出所有的实例,但通过定理 1 的证明,已经从理论上保证了 NSA 算法的近似比都不会超过 4/3.

5 结束语

本文详尽地讨论了 4-处理机系统中基于组调度 *GS* 的规则调度,构造出线性近似调度算法 NSA,并证明了该算法达到的近似比为 4/3,定理 2 还证明了 4/3-近似调度算法是规则调度中的最好结果.

参 考 文 献

[1] Hoogeveen J A, van de Velde S L, Veltman B. Complexity of scheduling multiprocessor tasks with prespecified processor allocations. Discrete Applied Mathematics, 1994, 55: 259-272

[2] Huang Jin-Gui, Chen Jian-Er, Chen Song-Qiao. Parallel-job scheduling on cluster computing systems. Chinese Journal of Computers, 2004, 27(6): 765-771(in Chinese)
(黄金贵, 陈建二, 陈松乔. 网络集群计算系统中的并行任务调度. 计算机学报, 2004, 27(6): 765-771)

[3] Huang Jin-Gui, Chen Jian-Er, Chen Song-Qiao. A simple linear time approximation algorithm for multiprocessor job scheduling on four processors. Journal of Combinational Optimization, 2007, 13(1): 33-45

[4] Chen J, Huang J G. Semi-normal schedulings: Improvement on goemans//Proceedings of the 12th Annual International Symposium on Algorithm and Computation (ISAAC'01). Lecture Notes in Computer Science, New Zealand, 2001: 48-60

[5] Graham R L, Lawler E L, Lenstra J K, Rinnooy Kan A H G. Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics, 1979, 5: 287-326

[6] Blazewicz J, Dell'Olmo P, Drozdowski M, Speranza M. Scheduling multi-processor tasks on the three dedicated processors. Information Processing Letters, 1992, 41: 275-280

[7] Dell'Olmo P, Speranza M, Tuza Z. Efficiency and effectiveness of normal schedules on three dedicated processors. Discrete Mathematics, 1997, 164(1): 67-79

[8] Chen J, Lee C-Y. General multiprocessor tasks scheduling. Naval Research Logistics, 1999, 46(1): 57-74

HUANG Jin-Gui, born in 1964, Ph. D., associate professor. His research interests focus on network computing optimal and software process technology, inclusive of network parallel computing, network routing algorithm, trusted network computing, resource management and scheduling, NP approximation optimization, parameter computing, and software process technology etc.

LI Rong-Heng, born in 1963, Ph. D., professor. His research interests include combinational optimal and computing theory.

Background

This paper is a product from the research in the projects supported by the National Natural Science Foundation of China, under grants No. 60872039 and No. 10771060. The objective of the projects is to study the computational complexity and algorithmic techniques for intractable optimization problems in computer networks and in other applications. The research has achieved significant progress recently. The authors have studied the multiprocessor job scheduling algorithms on distributed network systems.

Multiprocessor-job scheduling problem has attracted much attention recently. Because of complexity of the general multiprocessor system, The polynomial time approximation schemes for these problems are of great theoretical significance. However, most of these algorithms are based on extensive enumerations of certain kinds of scheduling together with either dynamic programming or linear programming techniques. This makes this kind of algorithms practically slow and difficult to implement. So researchers have asked for practically efficient and easy-implement able approximation algorithms for the parallel job-scheduling problem for systems with small number of processors. The optimal normal scheduling for 3-processor system is given and it is a 5/4-approximation algorithm for the multiprocessor job scheduling in 3-processor system

The paper is focused on the smaller processors system, that 4-processor system and its scheduling problem $P_4|fix|C_{\max}$. With introduced the Normal scheduling, and Group scheduling, a linear time algorithm is developed with the 4/3 approximation ratio. It is proved that normal scheduling come from the algorithm is the optimal normal scheduling in 4-processor systems.