并行环境下基于多处理机任务的调度模型与调度算法*)

Multiprocessor-job Scheduling: Models and Algorithms

黄金贵[1,2] 陈建二[2] 陈松乔[2]

(湖南师范大学图像识别与计算机视觉研究所 长沙 410081)[1]

(中南大学信息工程学院计算机理论与软件研究所 长沙 410083)[2]

Abstract Many jobs in parallel systems usually depend on more than one resource or more than one processor. This kind of jobs are called multiprocessor jobs. In this paper, we study the scheduling model $P_k|fix|C_{max}$ based on multiprocessor jobs. When $k\geqslant 3$, the problem is NP-hard in the strong sense thus it does not have a fully polynomial time approximation scheme unless P=NP. We develop a linear time approximation algorithm of ratio 5/3 for the $P_4|fix|C_{max}$ problem, which improves the best previous ratio 2 for practical algorithms for the problem. Our techniques are also useful for multiprocessor job scheduling problems on systems with more than four processors. We will also introduce a generalized and more practical scheduling model.

Keywords Parallel system, Multiprocessor job, Job scheduling, Approximation algorithm

1 引言

目前所研究的并行系统中的任务调度问题，大都针对于单处理机任务进行。所谓单处理机任务就是指所有被调度的任务都只需要一个处理机，而且可以是任意一个处理机。然而，在网络环境下，由于各个处理机的性能和功能不尽相同，一个任务往往需要一个或多个处理机同时执行才能完成，称这样的任务为多处理机任务。如果一个任务需要 r 个处理机同时执行，我们就称这个任务为 r-处理机任务。这里所说的"处理机"实际上是一个广义的概念，它可以是计算机系统中的 CPU、内存、驱动器或打印机等，也可以是网络系统的节点、链路和带宽等。一个并行工程，无论是上层的作业，还是底层的进程，它们都需要拥有不同的各种资源组合。并行系统中多处理机任务的调度，就是充分合理地利用所有可利用的资源，依照某种顺序串行或并行地调度所有待处理的多处理机任务，使得系统的总执行时间尽可能小。

一般地，多处理机系统定义为 k 个处理机的集合 $P=\{p_1,p_2,\cdots,p_k\}$，所需要该系统调度执行的多处理机任务集合为 $J=\{j_1,j_2,\cdots,j_n\}$，其中每个任务 $j_i(1\leqslant i\leqslant n)$具有二元属性 (s_i,t_i)，$s_i\subseteq P$ 是执行任务 j_i 需要拥有的一组处理机，通常称之为处理机模式或处理机类型；ti 是这组处理机同时处理该任务所需要的时间量，称为时间长度。每一个任务都只能在它所需要的所有处理机都空闲的情况下，才能被调度执行，而且在它被执行的期间内，它所占用的任一处理机不能被剥夺、中断或执行其它任务(但未被它占用的处理机可以同时地执行其它任务)。我们的目标就是构造一个调度，调度该系统中的 k 个处理机以最短的时间完成所有给定的多处理机任务，这就是典型的多处理机任务调度问题，记为 $P_k|fix|C_{max}$[1]，P_k 是 k 个处理机的集合，多处理机任务集合 J 是该调度问题的一个实例(Instance)。

关于多处理机任务调度问题 $P_k|fix|C_{max}$ 无论是在理论研究还是实际应用上，都引起了越来越广泛的兴趣和关注。近年来，关于其可行性和近似算法，已有大量的研究成果。Hoogeveen 等人证明：当 $k\geqslant 3$ 时，该问题是强 NP-难的，因此，除非 P=NP，不存在完全多项式时间近似的调度算法[2]。为了寻求一种可行的近似解，人们首先关注的是当 k=3 时的特殊情形。Blazewicz 等人为 $P_3|fix|C_{max}$ 构造了多项式时间的近似比为 4/3 的近似调度算法[3]，后又被 Dell'Olmo 等人[4]把近似比改进为 5/4。这些算法都是基于一种称为规则调度(normal schedule)的特殊调度，即所有类型相同的多处理机任务连续调度。Goemans[5]通过把一组类型相同的 1-处理机任务分开调度，得到了逼近性能为 7/6 的线性时间调度算法。但是，这些算法都很难推广到一般模型 $P_k|fix|C_{max}$，哪怕是 k=4，这些算法也无能为力。近年来，Chen 和 Miranda[6]对于固定的 k，给出了 $P_k|fix|C_{max}$ 的近似比为 $1+\epsilon$ 的多项式时间算法，遗憾的是，该多项式时间含有很高的次数。即使 $k=3,\epsilon=0.2$，时间就可以达到 $O(n^{50})$，显然这是不实用的。当 $k\geqslant 4$ 时，目前最好的可行解是 Chen 和 Lee[7]给出的线性时间近似调度算法，其近似比为 k/2。

本文主要考虑 $P_4|fix|C_{max}$，试图给出这类问题的一般性解决方法。首先，对于一个调度任务集实例，给出最优调度需要的时间跨度的两个下界。其一是考虑每个处理机的执行时间的总和，另一个则是基于 3-处理机任务和 2-处理机任务对而建立的。这两个下界非常简单，但是对于分析近似算法却非常有用。然后在调度 3-处理机任务和 2-处理机任务的基础上，考虑各个处理机在上述部分调度中的空闲时间，也就是上述各任务之间的间隙，在这些空闲时间间隙处调度所有 1-处理机任务，按照这种调度策略，我们得到近似比为 5/3，优于目前的近似比为 2 的最好结果。最后，本文还扩充了该模型，提出了更实际、更具一般性的模型。

2 符号与引理

考虑 4-处理机系统 $P=\{p_1,p_2,p_3,p_4\}$，简记为 $P=\{1,2,$

*)本文得到国家杰出青年学者自然科学基金和长江学者奖励基金资助。黄金贵 在职博士研究生，研究方向为计算机图形学、并行计算及系统软件。陈建二 长江学者特聘教授，博士生导师，主要研究领域为网络路由算法优化。陈松乔 教授，博士生导师，主要从事软件工程、CIMS 开发等方面的研究。

3,4}，$P_4|fix|C_{max}$问题的一个实例是一组任意给定的多处理机任务集 $J=\{j_1,j_2,\cdots,j_n\}$，其中 $j_i=(s_i,t_i)$，$1\leqslant i\leqslant k$，s_i 是{1,2,3,4}的一个非空子集，t_i 是该任务在这些处理机上运行的时间。

任务集 J 在 4-处理机系统的一个调度 S 就是让每个任务在某一时刻在需要的处理机集合上开始执行的一个指派(Assignment)。调度 S 的时间跨度是指：在调度 S 下，该 4-处理机系统完成任务集 J 的所有任务花费的总时间，记为 S(J)。任务集 J 的最优调度是 J 的所有调度中具有最小时间跨度的调度，记为 Opt，其时间跨度记为 Opt(J)，即

$Opt(J)=\min\{S(J)|S$ 是任务集 J 的任意调度$\}$

$P_4|fix|C_{max}$问题的一个近似调度算法就是为每个给定的实例给出一个调度的算法。该算法有两个评价指标：一是近似比，代表了该算法给定的调度与最优调度的逼近程度的性能；二是算法复杂度，代表了该算法的实用性。显然，算法的复杂度不应超过多项式时间。一个近似调度算法 A 的近似比 r 定义为：

$r=\{\max\{S(J)/Opt(J)|J$ 是任意一个实例，S 是算法 A 对实例 J 给出的一个调度$\}$

容易看出，在 4-处理机系统中，任何一个 4-处理机任务，无论什么时候调度都不会改变系统的时间跨度，因此，不失一般性，我们假定任务集实例 J 中不包含 4-处理机任务。于是，任务集实例 J 中最多有 $2^4-2=14$ 类不同处理机类型的任务。

为了简化调度，我们首先可以把实例 J 中所有的任务按其处理机类型归类，每一类任务组成一个任务块，调度时按任务块进行调度，即让同类任务连续调度执行。这种调度策略就是所谓的规则调度[4]，一个任务块称为一个规则任务。

4-处理机系统中，实例 J 中所有的任务按上述方法可以归为 14 个规则任务：

$J_I=\{j_i=(s_i,t_i)|1\leqslant i\leqslant n, s_i=I\subset\{1,2,3,4\}$and $s_i\neq\varnothing\}$

于是实例 J 就是：

$J=\{J_I|I\in\{1,2,3,4,12,13,14,23,24,34,123,124,234,134\}\}$

其中每一规则任务 J_I 的处理机类型是 I，执行时间是 t_I，且 $t_I=\Sigma_{(i|s_i=I)}t_i$。令执行完实例 J 的所有任务处理机 p_i 执行的总时间为 $T_i(i=1,2,3,4)$：

$T_i=\Sigma_{I\in\{K\}_i\in K\text{ and }K\subset(1,2,3,4)\}}t_I$

显然，任一调度的时间跨度不小于任一处理机执行的总时间，我们得到如下第一个下界：

引理 1 令 $B_1=\max\{T_i|i=1,2,3,4\}$，则 $Opt(J)\geqslant B_1$。

我们说模式(处理机类型)s_1 和 s_2 是相容的，如果任一模式为 s_1 的任务和任一模式为 s_2 的任务能够并行执行(即 $s_1\cap s_2=\varnothing$)。例如，模式$\{p_1,p_2\}$和模式$\{p_3,p_4\}$是相容的，模式$\{p_1,p_2,p_3\}$和模式$\{p_4\}$也是相容的。

考虑如下的 7 组规则任务：

$$\{J_{123}\},\{J_{124}\},\{J_{134}\},\{J_{234}\},\{J_{12},J_{34}\},\{J_{13},J_{24}\},\{J_{14},J_{23}\} \quad (1)$$

其中每组任务都是相容的，而每组中的任意一个任务与其它任何一组中的每一个任务都不相容。于是，我们有如下的另一个下界：

引理 2 令 $B_2=t_{123}+t_{124}+t_{134}+t_{234}+\max\{t_{12},t_{34}\}+\max\{t_{13},t_{24}\}+\max\{t_{14},t_{23}\}$，则 $Opt(J)\geqslant B_2$。

证明：实例 J 的任何一个调度 S，都要调度式(1)中的 7 个组。从两个不同的组中分别任取任务 j 和 j'，因为任何两组都不相容，所以 j 和 j' 不能并行执行，不可同时调度，只能依次串行执行。因此，在调度 S 下，任意两组在时间上不可能有重叠，而 7 组中的任务全部完成所花费的时间不小于 B_2，由于 S 的任意性即得 $Opt(J)\geqslant B_2$。□

3 调度算法

下面是 $P_4|fix|C_{max}$问题的一个近似调度算法：

近似调度算法 SA：
输入：任务集实例 $J=\{j_1,j_2,\cdots,j_n\}$，$j_i=(s_i,t_i)(i=1,2,\cdots,n)$
输出：实例 J 的一个调度 S_0。
1. 把任务集实例 J 中的所有任务按其处理机模式归类，得到新的规则任务集
   $J=(J_1,J_2,J_3,J_4,J_{12},J_{23},J_{13},J_{14},J_{24},J_{34},J_{123},J_{234},J_{134},J_{124}\}$；
2. 计算 T_1,T_2,T_3,T_4,B_1 和 B_2；
3. For any d in$\{t_{123},t_{234},t_{134},\max\{J_{12},J_{34}\},\max\{J_{13},J_{24}\},\max\{J_{14},J_{23}\}\}$do
   {If $d\geqslant(1/3)\max\{B_1,B_2\}$then 把(1)中时间长度为 d 的一组与 4 个 1-处理机规则任务一并调度，再依次调度其余 6 组，Goto 5；}
4. 在(1)中任选两组 2-处理机规则任务与 4 个 1-处理机规则任务一并调度，再依次调度其余 5 组；
5. End.

基于分组(1)的 7 组规则任务，组与组之间只能依次调度，每组内所有任务同时调度。图 2 所示是分组(1)中的 3-处理机规则任务组和 2-处理机规则任务组的一个调度。易知其时间跨度不超过 B_2。

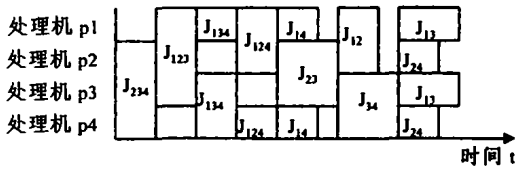

图 1 关于 2-处理机任务和 3-处理机任务的 7 个分组的调度

算法 SA 中第三步把(1)中的一组或两组与 4 个 1-处理机规则任务的一并调度是按图 3 的方式进行的。分 3 种情形描述如下：

(a)一个 3-处理机规则任务与 4 个 1-处理机规则任务的合并调度，如图 2(a)所示。

(b)一组 2-处理机规则任务与 4 个 1-处理机规则任务的合并调度，如图 2(b)所示。

(c)两组 2-处理机规则任务与 4 个 1-处理机规则任务的合并调度，如图 2(c)所示，4 个 1-处理机规则任务在两组 2-处理机规则任务$\{J_{12},J_{34}\}$和$\{J_{14},J_{23}\}$之间调度。且使 J_{12},J_{34}有相同的起点，J_{14},J_{23}有相同的终点，向内与 1-处理机任务靠紧，则至少有一个处理机没有空闲时间(例如图中的 p2)。

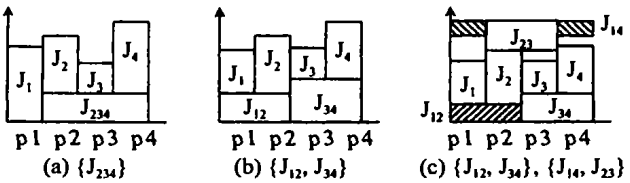

图 2 2-处理机任务或 3-处理机任务组与 1-处理机任务的合并调度

显然，上述每种情形下的时间跨度均不超过 $\max\{T_1,T_2,T_3,T_4\}=B_1$。下面我们分析算法 SA 的性能。

定理 1 算法 SA 的时间复杂度为 O(n)，且对于任一任务集实例 J，$SA(J)/Opt(J)\leqslant 5/3$。

证明：算法 SA 的时间复杂度是显然的，现证其逼近性能。令 d 是一组中的各任务的最大时间长度，则由算法 SA 的第三步知，如果存在一组使得 $d\geqslant(1/3)\max\{B_1,B_2\}$，则其余 6

组按图1的方式调度的时间跨度≤(2/3)max{B_1,B_2},而把该组和4个1-处理机任务按图2(a)或(b)的方式调度的时间跨度≤B_1,根据引理1和引理2,从而使得调度任务集J的总时间跨度

$SA(J)\leqslant B_1+(2/3)\max\{B_1,B_2\}$

$\leqslant Opt(J)+(2/3)Opt(J)=(5/3)Opt(J)$

另一方面,如果不存在一组使得 $d\geqslant(1/3)\max\{B_1,B_2\}$,即每一组的时间跨度均≤(1/3)max{$B_1$,$B_2$}。由算法SA的第四步,任取两组2-处理机任务与4个1-处理机任务按图2(c)的方式调度,其余的5组按图1的方式调度,则此时在图2(c)的调度方式下没有空闲的处理机 p_r 在整个任务集J的调度执行过程中最多只有两次空闲时间,一次为不需要处理机 p_r 的3-处理机任务的时间,另一次等于图2(c)调度方式未被调度的第三组2-处理机任务的时间长度。而两次空闲时间长度 $\leqslant 2*(1/3)\max\{B_1,B_2\}=(2/3)\max\{B_1,B_2\}$。所以,根据引理1,调度任务集J的总时间跨度:

$SA(J)\leqslant T_r+$两次空闲时间长度

$\leqslant B_1+(2/3)\max\{B_1,B_2\}\leqslant Opt(J)+(2/3)Opt(J)$

$=(5/3)Opt(J)$

综上所述,定理得证。 □

4 一般的调度模型

一般的调度模型SM(Schedule Model)定义为如下的5元组:

$SM=(PS,JI,PC,JC,EC)$

其中各元素代表的含义如下:

· PS(Processor system set):并行环境下的处理机系统,$PS=\{p_1,p_2,\cdots,p_k\}$;

· JI(Jobs set instance):处理机系统PS能够处理执行的任务集实例,它包含一组各种处理机类型的任务,每个任务都规定了它的执行时间。

· PC(Processor constraints set):处理机约束集。如处理机是否不可剥夺、是否相互区别、是否有相同的起点等。

· JC(Job constraints set):任务或任务之间的约束集。如任务本身是否可分解、是否带有时间延迟,任务之间是否有同步、互斥、优先等相关性约束。

· EC(Evaluate criterion set):调度的评价指标集。如调度的时间跨度、平均周转时间、系统响应比等。

在这种定义下,调度模型 $P_k|fix|C_{max}$ 可描述为:

PS={{$p_1,p_2,\cdots,p_k$},JI={多处理机任务},PC={不可剥夺、相互区别(即每个任务的处理机模式不能改变)},JC={任务不可分解},EC={调度的时间跨度}}。

在调度模型 $P_k|fix|C_{max}$ 中,若每个任务的处理机模式可以改变,则成为模型 $P_k|set|C_{max}$[6],该模型也已有一些研究,首先证明了当k=2时是NP-难的,当k≥3时是强NP-难的[2]。

调度模型SM的求解就是寻求一个调度S,使得任务集实例JI的每个任务在处理机系统PS中都有一个确定的指派,并且在满足各种处理机约束PC和任务约束JC的情况下,使得系统处理完成所有任务后的评价标准EC(S,JI)能够尽可能达到最优指标。一般来说,这是NP-难的,只能寻求逼近最优调度的调度方法。任务集实例JI的最优调度Opt定义为:

$Opt=\{S|EC(S,JI)=\min\{EC(S',JI)|S'\text{ 是 JI 的任一调度}\}\}$

调度S的逼近性能由近似比Ratio来度量,近似比定义为:

$Ratio=EC(S,JI)/EC(Opt,JI)$

当 Ratio→1 时,逼近性能愈好。

近似调度算法A是对任意一个实例JI,在多项式时间内构造一个满足上述条件且逼近最优调度的调度 S_0。近似调度算法A的近似比r就是对任意一个实例JI,有

$EC(S_0,JI)/EC(Opt,JI)\leqslant r$

结束语 在实际的并行系统中,任务的调度是一个很复杂的问题,该问题的解决直接影响着高性能并行计算的实现。本文基于多处理机任务研究了调度模型 $P_k|fix|C_{max}$,并采用了一种分组调度的技术,这种技术适用于一般情形的调度问题,当k=4时,我们得到了近似比为5/3的线性调度方法,并给予了严格的证明。最后,我们还在此基础上,描述了一个更复杂更接近实际的模型,如何寻求该问题的可行解,是一个很有意义而又有待于深入研究的问题。

参考文献

1 Hall L A. Approximation algorithms for scheduling. Approximation algorithms for NP-hard problems, PWS Publishing Company, 1997. 1～45

2 Hoogeveen J A, et al. Complexity of scheduling multiprocessor tasks with prespecified processor allocations. Discrete Applied Mathematics, 1994, 55: 259～272

3 Blazewicz J, et al. Scheduling multiprocessor tasks on the three dedicated processors. Information Processing Letters, 1992, 41: 275～280

4 Dell'Olmo P, et al. Efficiency and effectiveness of normal schedules on three dedicated processors. Discrete Mathematics, 1997, 164: 67～79

5 Goemans M X. An approximation algorithm for scheduling on three dedicated machines. Discrete Applied Mathematics, 1995, 61: 49～59

6 Chen J, Miranda A. A poynomial time approximation scheme for general multiprocessor job scheduling. In: Proc. 31st Annual ACM Symposium on Theory of Computing (STOC'99), Final version to appear in SIAM J. Comput. 1999. 418～427

7 Chen J, Lee C-Y. General multiprocessor tasks scheduling. Naval Research Logistics, 1999, 46: 59～74

8 Huang J G. Chen J, Chen S Q. A simple and improved approximation algorithm for scheduling multiprocessor-jobs on 3-processor systems. In: 16th World Computer Congress-International Computer Software: Theory and Practices. 2000. 801～806