多道相关任务系统的一种并行调度方法*

许曰滨 逯昭义

(青岛大学计算机系 青岛 266071)

摘要 该文针对分布式系统提出了一种描述任务动态特征的数据结构指派表AT(assignment table)及一个并行调度算法DRA(dynamic readjusting algorithm).经仿真运行,证明该算法能使分布式系统的并行处理能力得到提高.

关键词 多处理机系统,算法,并行处理,数据结构,分布式计算机.

中图法分类号 TP315

在一个由$n(n>0)$台处理机构成的分布式系统$P=\{P_1,\ldots,P_n\}$,任务调度的主要目标是将任务系统$(T,<)$分布到P上求解.其中T可以分解为$m(m>0)$个子任务.即$T=\{T_1,\ldots,T_m\}$,$<$是T上的偏序关系.

1977年,E. C. Horvath提出了将任务按偏序关系规定位级数,并通过位级数的大小制定分布策略的方法.文献[1]在此基础上提出一个改进的位级数抢先调度算法.文献[2,3]对多机相关任务调度的优化策略与组织作了深入研究.这些算法大都以压缩调度长度、减少任务间的通讯量作为追求的目标.

由于$(T,<)$中诸任务之间存在着相互关联、相互制约的关系,任务调度过程不可避免地产生一些处理机的闲置时间.[4]为了提高资源利用率,增加系统吞吐能力,本文提出一种多道相关任务系统并行调度方法,旨在有效地利用处理机时间,为多道任务系统服务.

1 概念及表示法

一个任务系统的前趋图(Precedence Graph)是表示任务系统$(T,<)$的有向图.图中的一个结点对应T中的一个任务.结点权量是任务的运行长度,表示任务需用的单位执行时间UET(unit executing time)数.图中的有向线表示任务间的偏序关系.令$s,t\in T$,若在前趋图中存在一条从s到t的有向线,则表示s是t的直接前趋,t是s的直接后继,记作$s<t$.

定义1. 处理机$P_i\in P$的一个指派(Assignment)是一个三元组$\langle name,st,\tau\rangle$.其中

$name$: P_i承担的任务名称.

st: 在P_i上的开工时间.从当前时刻起经历的系统UET数.

τ: 任务$name$的运行长度.

多机系统中,每台处理机可分布多个任务.每个任务对应一个指派.各处理机的若干指派组成指派表.P_i的指派表记作AT_i,其中第j个指派记作AT_{ij},它的3个分量分别记作$AT_{ij}[name]$,$AT_{ij}[st]$和$AT_{ij}[\tau]$.则P_i在第$AT_{ij}[st]+AT_{ij}[\tau]-1$个UET结束时完成任务$AT_{ij}[name]$.

定义2. 处理机$P_i\in P$的一个闲置周期是一个二元组$\langle x,y\rangle$.其中x是P_i的闲置开始时间,y是其闲置长度.

系统初启,所有AT_i的状态为Φ,表示P_i有一个闲置周期$\langle 0,\infty\rangle$.任务定序程序将根据P_i分布的任务及偏序关系,为AT_i添加若干指派,并按开工时间的先后排列成有序表.

对于P_i的任二指派AT_{ij}和AT_{ik},为方便计,不妨令$AT_{ij}[st]<AT_{ik}[st]$,若不存在一个指派AT_{ir},使得$AT_{ij}[st]<AT_{ir}[st]<AT_{ik}[st]$成立,则称$AT_{ij}$和$AT_{ik}$是相邻的.

定理. 对于$P_i\in P$的任意两个相邻的指派AT_{ij}和AT_{ik},令$AT_{ij}[st]<AT_{ik}[st]$,若满足$AT_{ij}[st]+AT_{ij}[\tau]<AT_{ik}[st]$,则$P_i$在$AT_{ij}$和$AT_{ik}$之间必存在一个闲置周期$\langle x,y\rangle$.其中

---

* 本文研究得到山东省自然科学基金资助.作者许曰滨,1950年生,副教授,主要研究领域为计算机网络与分布式系统.逯昭义,1942年生,教授,主要研究领域为计算机网络体系结构.

本文通讯联系人:许曰滨,青岛 266071,青岛大学计算机系

本文1997-09-16收到原稿,1998-03-03收到修改稿

$$x=AT_{ij}[st]+AT_{ij}[\tau],$$
$$y=AT_{ik}[st]\quad (AT_{ij}[st]\mid AT_{ij}[\tau]).$$

证明从略.

推论 1. 对于 $P_i\in P$ 上开工最早的指派 AT_{ij}，若满足 $AT_{ij}[st]>0$，则 P_i 存在一个闲置周期$\langle 0,AT_{ij}[st]-1\rangle$.

推论 2. 对于 $P_i\in P$ 的一个指派 AT_{ij}，若 $AT_{ij}[st]=\mathrm{Max}(AT_{i1}[st],AT_{i2}[st],\cdots)$成立，则 P_i 存在一个闲置周期$\langle AT_{ij}[st]-AT_{ij}[\tau],\infty\rangle$.

显然，在 P_i 的最后一个指派结束时刻 t，必有闲置周期$\langle t,\infty\rangle$. 该周期内可以排得下 P_i 上的任何任务.

2　算　法

2.1　目标函数

令 TCB 是存储任务控制信息的集合，TCB_r 对应系统中的一个任务 r. $TCB_r[P]$保存 r 分布的处理机号. $TCB_r[\tau]$是 r 的运行长度，$TCB_r[\mathrm{wait}]$是 r 因偏序关系制约应当等待的时间. 它受制于 r 的直接前驱任务中结束最迟的任务 s，其值由下式计算

$$TCB_r[\mathrm{wait}]=TCB_s[\mathrm{wait}]+TCB_s[\tau].$$

由于 P_i 有 $n(n>0)$个包括$\langle t,\infty\rangle$在内的空闲周期，因而必存在一个空闲周期$\langle x,y\rangle$能够满足任务 r 的运行要求，即

$$x+y\geqslant TCB_r[\mathrm{wait}]+TCB_r[\tau]\wedge y\geqslant TCB_r[\tau] \tag{1}$$

让 $Early_i(x)$表示 P_i 上的一个满足式(1)，且为最早的空闲周期的起始时间，则根据定义、定理，选择下面的目标函数，以计算任务开工时间

$$\mathrm{Max}(Early_i(x),TCB_r[\mathrm{wait}]) \tag{2}$$

该目标函数在兼顾任务本身的偏序关系及处理机空闲情况的同时，为任务安排最早的开工时间，以免因安排不紧凑而拖延后续任务的运行.

例：设处理机 P_1 已有两个任务 u 和 v. 对应的指派为$\langle u,5,1\rangle$和$\langle v,9,2\rangle$. 因而，它有 3 个空闲周期$\langle 0,5\rangle$，$\langle 6,3\rangle$和$\langle 11,\infty\rangle$. 若 P_1 上分布了一个新任务 w，且有

$$TCB_w[\mathrm{wait}]=[0],TCB_w[\tau]=1.$$

显然，P_1 上的 3 个空闲周期都能满足式(1)，且$\langle 0,5\rangle$是最早的，则根据式(2)，开工时间应确定为 3. 即

$$\mathrm{Max}(Early_i(x),TCB_j[\mathrm{wait}])=\mathrm{Max}(0,3)=3.$$

这样，P_1 的指派表将变为$\langle w,3,2\rangle$，$\langle u,5,1\rangle$和$\langle v,9,2\rangle$.

2.2　任务定序算法 DRA 描述

S1：从任务系统$(T,<)$中找到任一尚未确定开工时间，且其所有直接前驱任务皆已排定的任务 t，做

$$i:=TCB_t[P].$$

S2：从头至尾查找 AT_i，找到第 1 个满足式(1)的空闲周期$\langle x,y\rangle$. 做

$$te:=\mathrm{Max}(x,TCB_t[\mathrm{wait}]).$$

S3：按指派表的开工时间的升序关系，将一个新指派$\langle t,te,TCB_t[\tau]\rangle$插入 AT_i. 即

$$Insert(AT_i,\langle t,te,TCB_t[\tau]\rangle).$$

S4：对任务 t 的所有直接后继任务 r，做

$$TCB_r[\mathrm{wait}]:=\mathrm{Max}(TCB_r[\mathrm{wait}],te+TCB_t[\tau]).$$

S5：若$(T,<)$尚有未确定开工时间的任务，则转 S1.

S6：结束.

2.3　算法分析

安排任务的开工时间，除了受任务系统本身的偏序关系制约外，还受处理机闲置周期的制约. 当任务 t 在 P_i 上安排了最早的开工时间，t 的安排才是紧凑的. 在 S2 中，任务开工时间 te 的确定分为下述 3 种情况.

(1) 当 $AT_i=\Phi$，处理机的闲置周期为$\langle 0,\infty\rangle$，根据式(2)，有 $\mathrm{Max}(0,TCB_t[\mathrm{wait}])=TCB_t[\mathrm{wait}]$，$te$ 完全取决于偏序关系的约束值 $TCB_t[\mathrm{wait}]$. 显然，安排 t 的开工时间为 $TCB_t[\mathrm{wait}]$是紧凑的.

(2) 当第 1 指派前有闲置周期$\langle 0,AT_{i1}[st]\rangle$，且满足式(1)时，则安排开工时间为 $TCB_t[\mathrm{wait}]$，同样也是紧凑的.

(3) 当 AT_i 有 1 个以上的指派，则从第 1 个指派开始循环向后找，若发现任二指派之间有闲置周期并满足式(1)，便予以安排. 由于 AT_i 中的指派按开工时间升序排列，循环查到的第 1 个可安排的空闲周期必为最早的，因此安排是

紧凑的.

显然,若 t 在 P_i 上的安排是紧凑的,便必为后继任务创造尽早开工的条件.在一个无环路的有向图中,若从起始任务结点出发,严格按照偏序关系前进,每安排一个任务都是紧凑的,则所有结点完成后,整个任务系统的排工是最早的.

在算法中,S1 按偏序关系确保先安排前趋再安排后继,依次进行.S2 确保每安排一个指派都是紧凑的.S4 为后继任务的安排创造条件.由于$(T,<)$的任务数量是有限的,且每循环一次,就安排一个任务,随任务的安排不断增加,循环必能有限次终止.这样,当所有任务安排完毕时,算法必获得最早的排工,即收敛于最佳解.

算法的复杂度为 $O(n^2)$.

3 调度实现

系统中的任务调度程序由时钟中断处理程序启动.系统时钟设备每隔一个 UET 产生一次中断,启动任务调度程序,作

① 将所有 $AT_{ij}[st](i\leqslant n,j\leqslant m)$的值减 1.当其减为零时,表示 $AT_{ij}[name]$可立即投入运行.若减为一个负值,表示 $AT_{ij}[name]$正在运行,且其绝对值是已运行的 UET 数.

② 检测所有处理机的状态.若 $P_i\in P$ 的处理机状态为空闲,且 $AT_{i1}[st]$为一个负值,说明 $AT_{i1}[name]$运行结束.从 AT_i 中删去 AT_{i1},以确保下次要处理的指派是 AT_i 中的第 1 指派.

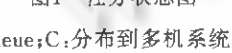


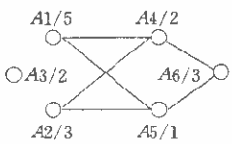
图1　任务状态图

③ 对于所有 st 为零的指派,启动响应处理机,传送任务的程序集地址及数据集地址,使任务投入运行.

一个任务系统从提交到运行大体经历 5 个阶段.如图 1 所示.其中 A:开始;B:生成 Queue;C:分布到多机系统上;D:形成指派表;E:任务调度.

4 调度示例

令系统有 3 台处理机,$P=\{P1,P2,P3\}$.设提交的任务系统为$(A,<)$.前趋图如图 2(a)所示.假定系统采用某种算法,比如均衡分布算法[4],在兼顾到系统开销(包括任务执行时间及信息交换 context-swapping 等)基础上,将$(A,<)$分布到 P 上.分布情况如图 2(b)所示.

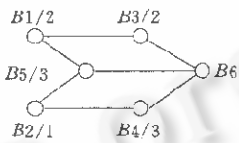


(a)

A1	A2	A3	A4	A5	A6
P1	P3	P2	P3	P2	P1

(b)

图2　任务系统$(A,<)$及其分布



(a)

B1	B2	B3	B4	B5	B6
P2	P1	P1	P3	P2	P2

(b)

图3　任务系统$(B,<)$及其分布

经过任务定序算法 DRA,生成处理机指派表为

$$AT_1=\langle A1,0,5\rangle\ \langle A6,7,3\rangle$$
$$AT_2=\langle A3,0,2\rangle\ \langle A5,5,1\rangle$$
$$AT_3=\langle A2,0,3\rangle\ \langle A4,5,2\rangle$$

当时钟中断到来时,系统令 3 个开工时间为零的任务 $A1$,$A2$ 和 $A3$ 投入运行.设想 A 在运行中不再会有新的任务提交,即相当于单道批处理系统,其调度轨迹见表 1.调度长度为 10.处理机利用率为 53%.

表1　单任务系统$(A,<)$调度轨迹

	0	1	2	3	4	5	6	7	8	9
P1	A1	A1	A1	A1	A1			A6	A6	A6
P2	A3	A3				A5	B5			
P3	A2	A2	A2			A4	A4			

表2　二任务系统$(A,<)$,$(B,<)$并行调度轨迹

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
P1	A1	A1	A1	A1	A1	B2		A6	A6	A6	B3	B3		
P2	A3	A3	B1	B1		A5	B5	B5	B5				B6	B6
P3	A2	A2	A2			A4	A4	B4	B4	B4				

经过一个 UET 的运行后,所有指派的开工时间皆减 1.现在,我们假设此刻又提交了一个新任务系统$(B,<)$,如图 3(a)所示,其任务分布情况如图 3(b)所示.

经算法 DRA，指派表增至

$$AT_1=\langle A1,-1,5\rangle\ \langle B2,4,1\rangle\ \langle A6,6,3\rangle\ \langle B3,9,2\rangle$$

$$AT_2=\langle A3,-1,2\rangle\ \langle B1,1,2\rangle\ \langle A5,4,1\rangle\ \langle B5,5,3\rangle\ \langle B6,11,2\rangle$$

$$AT_3=\langle A2,-1,3\rangle\ \langle A4,4,2\rangle\ \langle B4,6,3\rangle$$

正常情况下，这两个任务系统的并行调度轨迹呈表 2 的形式. 可以看出，它们的总调度长度为 14. 处理机利用率接近 70%.

若采用串行调度方式，即当$(A,<)$调度完成之后再调度$(B,<)$，总调度长度可达 17. 处理机利用率仅为 57%. 串行调度轨迹见表 3.

表3　二任务系统$(A,<)$，$(B,<)$串行调度轨迹

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
P1	A1	A1	A1	A1	A1			A6	A6	A6	B2		B3	B3			
P2	A3	A3				A5					B1	B1	B5	B5	B5	B6	B6
P3	A2	A2	A2			A4	A4						B4	B4	B4		

5　结束语

多道任务系统的并行调度是提高处理机利用率的主要途径. 本文提出的任务定序算法在兼顾到任务间的偏序关系及处理机空闲情况的同时，为任务安排最早的开工时间，以免因安排不紧凑拖延后续任务的运行. 我们将本文提出的调度方案实现于 1 个由 3 台处理机组成的多机系统中，经仿真测试，结果与第 4 节的示例一致.

参考文献

1　尹祚明. 带后继位级跟踪的抢先位级调度. 计算机学报，1989，12(1)：10～16
(Yin Zuo-ming. Preemptive level schedule with successors level tracking. Chinese Journal of Computers, 1989, 12(1): 10～16)

2　杨羽，鄢伶俊. 多机相关任务调度的优化策略与组织方法. 计算机学报，1993，16(9)：661～669
(Yang Yu, Yan Ling-jun. Optimization strategy for the scheduling of dependent tasks in multiprocessors. Chinese Journal of Computers, 1993, 16(9): 661～669)

3　赵致琢. 分布式问题分解与分布算法研究. 计算机学报，1993，16(8)：606～613
(Zhao Zhi-zhuo. Research on distributed problem decomposition and distribution algorithms. Chinese Journal of Computers, 1993, 16(8): 606～613)

4　许曰滨. 多机相关任务的均衡调度算法. 计算机学报，1996，19(1)：77～80
(Xu Yue-bin. The equilibrium scheduling algorithm for dependent tasks in multiprocessors. Chinese Journal of Computers, 1996, 19(1): 77～80)

A Method of Scheduling Parallel Multiple Dependent-Task System

XU Yue-bin　LU Zhao-yi

(Department of Computer Science　Qingdao University　Qingdao　266071)

Abstract　A parallel scheduler for distributed system is described. This scheduler uses a data-structure AT (assignment table) to record the tasks' dynamic characteristics, and schedules processors by an algorithm DRA (dynamic readjusting algorithm). This paper shows that the capacity of the distributed system is improved.

Key words　Multiprocessor systems, algorithms, parallel processing, data structure, distribution computers.