非完全互连同构系统上的静态任务调度*

章军 章立生 韩承德

(中国科学院计算技术研究所高性能计算机研究中心 北京 100080)

摘要 在分布式内存多处理机DMM(distributed memory multiprocessor)系统中,不同处理机上运行的任务之间的通信开销仍然很大,有时甚至抵消了多处理机并行所带来的好处.为了使并行程序在DMM系统上能得以高效的执行,必须采用合理的调度技术将任务分配给处理机.文章首先分别给出了任务调度系统中的任务模型、处理机模型以及调度问题的形式化描述,然后在此基础上研究了任务调度中3个最重要的问题,即(1) 如何顺序选择参与调度的任务,(2) 如何选择路由,(3) 如何分配任务给处理机.其中,路由选择是按存储转发寻径与虫蚀寻径两种不同的方式来讨论的.最后,根据上述3个问题的解决策略,构造了一个非完全互连同构系统上的静态任务调度算法.

关键词 静态任务调度,任务模型,处理机模型,存储转发寻径,虫蚀寻径.

中图法分类号 TP338

在分布式内存多处理机(distributed memory multiprocesor,简称DMM)系统中,所有处理机经网络互连起来,任意两台处理机之间的通信都必须经过互连网络.近30年来,即使通信技术的发展突飞猛进,但与处理机的运算速度相比,通信开销仍然很大,是制约消息传递型系统性能进一步提高的瓶颈.

为了使并行程序在DMM系统上能得以高效的执行,必须采用合理的调度技术将不同的任务在适当的时刻分配给处理机去执行.对于静态任务调度而言,处理机的任务分配是在程序执行之前完成的.有关任务的计算量、任务之间的依赖关系及通信情况、每个处理机的处理能力以及它们之间的互连拓扑在编译时假定是已知的.另外,任务一旦分配给某个处理机,便只能在该处理机上执行,即任务的执行是非抢占式的(nonpreemptive).一般说来,静态调度的目标是最小化整个应用的执行时间.

本文考虑的问题要求处理机系统是同构的,对处理机的个数以及处理机之间是否完全互连不作限制.另外,在任务调度时,不允许任务复制.本文第1节给出了任务调度系统模型,它包含任务模型与处理机模型.第2节给出了调度问题的形式化描述.第3节讨论了非完全互连同构系统上任务调度中的3个最重要问题的解决策略,并构造了相应的调度算法,这3个问题分别是:(1) 如何顺序选择参与调度的任务,(2) 如何选择路由,(3) 如何将处理机分配给任务.其中,路由选择是按存储转发寻径与虫蚀寻径两种不同的方式来讨论的.

1 调度系统模型的形式化描述

调度系统模型包含任务模型与处理机模型.下面分别给出任务模型与处理机模型的形式化描述.

通常,任务依赖图是经程序划分后获得的,可用图理论中的有向无环图(directed acyclic graph,简称DAG)来表示.任一有向无环图(DAG)可用四元组$TG=(V,E,A,D)$来定义.其中:

(1) $V=[v_i]$表示任务图中结点(任务图中的结点即指任务,故在本文中有时结点与任务混用)的集合,v_i表示第i个任务,$|V|$表示图中结点数目;

---

* 本文研究得到国家自然科学基金和国家攀登计划项目基金资助.
作者章军,1971年生,博士,主要研究领域为并行计算,并行任务调度,计算机体系结构.章立生,1962年生,副研究员,主要研究领域为计算机体系结构,计算机网络,并行处理.韩承德,1940年生,研究员,博士生导师,主要研究领域为计算机体系结构,并行处理.

本文通讯联系人:韩承德,北京100080,北京2704信箱25分箱

本文1998-08-28收到原稿,1998-11-30收到修改稿

(2) $E=[e_{i,j}]$表示任务图中边(任务图中的边即指消息，故在本文中有时与消息混用)的集合，$e_{i,j}$表示由 v_i 指向 v_j 的有向边，$|E|$表示图中边的数目；

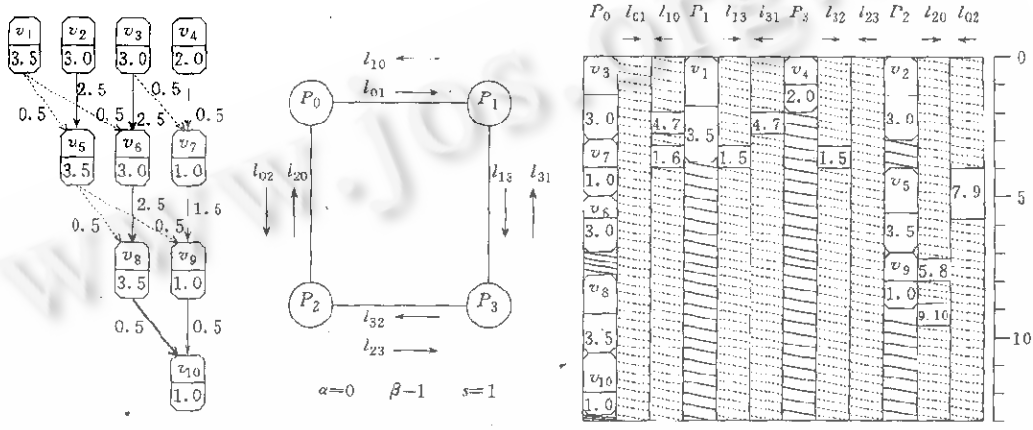
(3) $A=[a_i]$表示任务的计算量集合，a_i 表示任务 v_i 的计算量；

(4) $D=[d_{i,j}]$表示任务之间通信的数据量集合，$d_{i,j}$表示任务 v_i 发送给任务 v_j 的数据量.

为了表示图 TG 中任务之间的互相依赖情况，下面给出两个优先(precedence)关系：⇒及→.

定义 1.1. 在图 TG 上，如果 v_i 是 v_j 的父结点(或 v_j 是 v_i 的子结点)，则有 $v_i \Rightarrow v_j$；如果 v_i 是 v_j 的祖先结点(或 v_j 是 v_i 的子孙结点)，则有 $v_i \rightarrow v_j$.

由上面的定义可知，关系→具有传递性，而关系⇒不具有传递性，$e_{i,j} \in E$ 等价于 $v_i \Rightarrow v_j$. 在图 1(a)中，有 $v_3 \rightarrow v_6$，$v_2 \rightarrow v_8$ 等满足上述两个关系.



(a) 有向无环图(DAG)　　(b) 处理机模型　　(c) 执行结果

图1　调度系统模型

由任务 v_i 的父结点与祖先结点构成的集合分别记为 $PARENT(v_i)$ 与 $PRED(v_i)$；由任务 v_i 的子结点与子孙结点构成的集合分别记为 $CHILD(v_i)$ 与 $SUCC(v_i)$. 在图 1(a)中，$PARENT(v_6)=\{v_1,v_3\}$，$CHILD(v_3)=\{v_6,v_7\}$，$PRED(v_8)=\{v_1,v_2,v_3,v_5,v_6\}$，$SUCC(v_5)=\{v_8,v_9,v_{10}\}$.

定义 1.2. 对于任务 v_i，若 $PARENT(v_i)=\varnothing$，则 v_i 称为入结点(entry node)；若 $CHILD(v_i)=\varnothing$，则 v_i 称为出结点(exit node). 记所有的入(出)结点构成的集合为 ENTRY(EXIT).

这种任务执行模型又称为编译时的宏数据流模型(compile time macro data flow model)[1~3]. 在该模型中，每个任务只有在它所需要的数据到齐之后，才可以开始执行，即任务的开始执行是以它所需要的数据的到齐来驱动的. 任务执行完后，立即将其后代所需要的数据发送给它们. 通常，数据的发送是非堵塞(nonblocking)方式的，而数据的接收是堵塞(blocking) 方式的.

处理机模型可以用五元组 $PG=(P,L,S,I,R)$来描述.

(1) $P=[p_i]$表示所有处理机的集合，$|P|$表示处理机的数目.

(2) $L=[l_{i,j}]$是一个 $|P|\times|P|$的处理机互连拓扑矩阵. $l_{i,j}$只能取 0 或 1，若 $l_{i,j}=0$，则表示 p_i 与 p_j 之间没有直接连接(link)，若 $l_{i,j}=1$，则表示 p_i 与 p_j 之间有直接连接.

(3) $S=[s_i]$表示处理机的运算速度集合，s_i 表示处理机 p_i 的运算速度.

(4) $I=[a_i]$表示通信时的启动开销集合，a_i 表示处理机 p_i 通信时相应的 I/O 处理机启动开销.

(5) $R=[\beta_{i,j}]$表示两个相邻处理机之间通道的数据传输率集合，$\beta_{i,j}$表示了两个相邻处理机 p_i 与 p_j 之间通道的数据传输率.

通常，假定所有连接都是全双工的(full duplex)，每个处理单元都带有专门的硬件来支持并行通信，这样的部件称为 I/O 处理机，这种 I/O 处理机允许任务的通信与计算重叠(overlapping)地进行.

在同构系统(homogeneous system)上,所有处理机的运算速度相等(简记为 s),通信的启动开销相等(简记为 α),并且所有通道的数据传输率都相等(简记为 β),则处理机模型可用两元组 (P,L) 来描述.而任务模型可以改用四元组 $TG=(V,E,T,D)$ 来描述,其中 V,E 及 D 的定义同前,而 $T=[\Gamma_i]$ 表示任务的计算时间集合,Γ_i 表示执行任务 v_i 所需要的时间,为 a_i/s.当任务 v_i 与 v_j 被分配到不同的处理机 p_m 与 p_n 上去执行时,在采用某种路由算法确定了某条路径,且路径上的所有通道均处于空闲状态时,如果采用存储转发寻径技术,任务之间的通信时间为 $c_{i,j,m,n}=pl(m,n)\times(\alpha+d_{i,j}/\beta)$,其中 $pl(m,n)$ 为处理机 p_m 与处理机 p_n 之间路径的长度;如果采用虫蚀寻径技术,任务之间的通信时间为 $c_{i,j,m,n}=\alpha+d_{i,j}/\beta$.

2 调度问题的形式化描述

任务调度的消费者是 DAG 图中的所有任务以及任务之间通信的消息,资源是经某种拓扑互连的处理机及处理机之间的连接.调度算法考虑的是如何将处理机分配给任务及连接分配给消息.

在非完全互连系统上,形式地(文献[4]也曾给出了调度问题的形式化描述,但却没有考率任务之间通信的消息),某个调度可用函数 f 来表示,该函数的定义域为 $V\cup E\times L$,值域为 $P\times[0,\infty)\cup[0,\infty)$.对于某个任务 $v\in V$,如果 $f(v_i)=(p_j,t)$,我们称任务 v_i 被调度到处理机 p_j 上,并在 t 时刻开始执行;对于某个消息 $e_{i,j}$,如果 $f(e_{i,j},l_{m,n})=t$,则表示消息 $e_{i,j}$ 在采用某种寻径策略时需要使用连接 $l_{m,n}$,在 t 时刻,该连接可以分配给该消息使用,如果消息 $e_{i,j}$ 不需要使用连接 $l_{m,n}$,令 $t=\infty$.

衡量调度性能的最主要的指标是调度长度(schedule length,许多文献[1,3]中又称为 parallel time,简记为 PT).调度 f 的调度长度 $PT(f)=\max\{t+\Gamma_{i,j}\}$,其中 $f(v_i)=(p_j,t)$,且 $\Gamma_{i,j}=a_i/s_j$.

定义 2.1. 假定任务在被调度之前都映射到不同的处理机上,且这些处理机之间以完全互连的方式连接,这个假定的处理机系统又称为虚拟处理机系统.任务图 TG 中的静态关键路径(static critical path,简称 SCP)定义为从入结点到出结点中最长的路径(包含任务之间的通信时间).

对于图 1(a),当 $\alpha=0,\beta=1,s=1$ 时,该任务图中的静态关键路径为 $v_3v_6v_8v_{10}$.

定义 2.2. 在同构系统上,在部分调度的任务图 TG' 中,任务图的动态关键路径(dynamic critical path,简称 DCP)定义为 TG' 中的从入结点到出结点中最长的路径(包含任务之间的通信时间).

定义 2.3. 在同构系统上,在部分调度的任务图 TG' 中,任一结点 v_x 的 top level(简记为 tlevel)定义为从入结点到该结点(不包含该结点的权)的最长路径(记为 $tlevel(v_x)$).任一结点的 bottom level(简记为 blevel)定义为从该结点到出结点(包含该结点的权)的最长路径(记为 $blevel(v_x)$).

在同构系统上,在未调度的任务图 TG 中,SCP 的长度为 $\max_{v_x\in V}\{tlevel(v_x)+blevel(v_x)\}$,对于部分调度的任务图 TG',该式同样可用于计算 DCP 的长度.对于未调度的任务图 TG,任意结点 v_i 的 tlevel 与 blevel 的计算方法如下:

$$tlevel(v_i)=\begin{cases}0 & v_i\in ENTRY,\\ \max\{tlevel(v_x)+\Gamma_x+c_{x,i}\mid v_x\Rightarrow v_i\} & \text{otherwise.}\end{cases}$$

$$blevel(v_i)=\begin{cases}\Gamma_i & v_i\in EXIT,\\ \max\{blevel(v_x)+c_{i,x}+\Gamma_i\mid v_i\Rightarrow v_x\} & \text{otherwise.}\end{cases}$$

在同构系统上,若任务调度遵从拓扑顺序,已被调度结点的 tlevel 为该结点的开始执行时间(给出已被调度结点的 tlevel 值的计算,是因为计算未被调度结点的 tlevel 值时要用到已被调度结点的 tlevel 值),未被调度结点的 tlevel 及 blevel 值的计算也采用以上两式.

3 几个关键问题的解决策略及调度算法

本节先给出非完全互连同构系统上静态调度中的几个关键问题的解决策略,然后给出调度算法.

已经发表的非完全互连同构系统上任务调度的文章极少.其中,El-Rewini 在文献[5]中首先提出了非完全互连系统上任务调度的算法,但他的算法仅适用于存储转发寻径通信,而且也没有讨论如何去利用连接的最早的(这里的最早以及下面的较早都是针对处理机或连接上被占用的最后一段或下文中的第 δ 段时间空隙而言)

空闲时间空隙(idle time slot).Kwok 在文献[6]中提出了如何利用连接的最早空闲时间空隙的方法,但经过仔细分析,他的工作忽略了"不同连接上的空闲时间空隙分布是不一致的"这一实际情况.

非完全互连同构系统上并行任务的静态调度中要解决的问题有 3 个:(1) 如何顺序选择参与调度的任务,(2) 如何选择路由,(3) 如何将处理机分配给任务.

如何顺序选择参与调度的任务　一般认为,应优先调度关键路径上的任务.在确定关键路径时,有些算法采用静态关键路径,而有些算法采用动态关键路径.对于调度算法的性能,采用动态关键路径的算法一般优于采用静态关键路径的算法;对于调度算法的复杂度,采用动态关键路径的算法通常高于采用静态关键路径的算法.

如何选择路由　对于非完全互连系统,处理机之间的连接是有限的资源,在并行程序执行过程中,很有可能出现不同的消息争用同一连接的情况.因此,如果将连接也作为资源来静态地加以分配,在程序运行时,便不再需要考虑网络的链路堵塞以及死锁等问题的出现.

对于非完全互连同构系统,从源到目的的路径可能很多,问题是如何确定某个路由,从源发出的消息经过该路由时能尽量早地到达目的地.对于某个确定路由,如果采用某种方法能计算出消息经过该路由到达目的处理机的最早时刻,则先算出所有的路由所对应的最早时刻,便可以求出消息到达目的处理机的最早时刻并选择相应的路由.对于存储转发寻径技术,本文给出了消息经过某个路径到达目的处理机的最早时刻的计算方法;对于虫蚀寻径技术,尚没有人给出消息经过某个路径到达目的处理机的最早时刻的计算方法,本文给出了消息经过某个路径到达目的处理机的一个较早时刻的计算方法.

先定义两个将在下文中用到的对二元矢取分量的操作 fst 与 $scnd$:

定义 3.1. 若将操作 fst 与 $scnd$ 作用于二元矢(x_1,x_2),则 $fst((x_1,x_2))=x_1$,$scnd((x_1,x_2))=x_2$.

对于任务调度 f,假定任务 v_i 已被调度,v_i 要向其子任务 v_j 发送消息 $e_{i,j}$,如果在选择某条路径时需要使用连接 $l_{m,n}$.下面分两种不同的寻径方式,即存储转发寻径与虫蚀寻径来讨论如何选择路由.

(a) 在采用存储转发寻径技术时

假定在此之前已有 δ 个消息$\{e^1_{m,n},\ldots,e^{\delta}_{m,n}\}$要求顺序使用连接 $l_{m,n}$.

(1) 若 $fst(f(v_i))=p_m$,即消息刚从源处理机出发经过第 1 个连接时,则存在一些 k 满足下式:

$$f(e^{k+1}_{m,n},l_{m,n})-\max\left\{scnd(f(v_i))+\Gamma_i+\alpha,f(e^k_{m,n},l_{m,n})+\frac{size(e^k_{m,n})}{\beta}\right\}\geqslant\frac{d_{i,j}}{\beta},\tag{1}$$

其中 $k\in\{0,\ldots,\delta\}$,$f(e^0_{m,n},l_{m,n})=0$,$f(e^{\delta+1}_{m,n},l_{m,n})=\infty$,$size(e^k_{m,n})$为消息 $e^k_{m,n}$的大小,且 $size(e^0_{m,n})=0$.

对于式(1),至少 $k=\delta$ 时是成立的.如果 ι 为所有 k 中的最小值,则消息 $e_{i,j}$可以使用连接 $l_{m,n}$的最早时刻为

$$\max\left\{scnd(f(v_i))+\Gamma_i+\alpha,f(e^{\iota}_{m,n},l_{m,n})+\frac{size(e^{\iota}_{m,n})}{\beta}\right\},\tag{2}$$

(2) 若 $fst(f(v_i))\neq p_m$,假定消息 $e_{i,j}$在使用连接 $l_{m,n}$时刚刚经过了连接 $l_{l,m}$,则如果存在一些 k 满足下式:

$$f(e^{k+1}_{m,n},l_{m,n})-\max\left\{f(e_{i,j},l_{l,m})+\frac{d_{i,j}}{\beta})+\alpha,f(e^k_{m,n},l_{m,n})+\frac{size(e^k_{m,n})}{\beta}\right\}\geqslant\frac{d_{i,j}}{\beta},\tag{3}$$

对于式(3),至少当 $k=\delta$ 时是成立的.如果 ι 为所有 k 中的最小值,则消息 $e_{i,j}$可以使用连接 $l_{m,n}$的最早时刻为

$$\max\left\{f(e_{i,j},l_{l,m})+\frac{d_{i,j}}{\beta}+\alpha,f(e^{\iota}_{m,n},l_{m,n})+\frac{size(e^{\iota}_{m,n})}{\beta}\right\}.\tag{4}$$

根据该路径上的最后一段连接,不难求出消息经过该路径到达目的处理机的最早时刻.

定理 3.1. 在采用存储转发寻径技术,按式(1)~(4)选择连接的空闲时间空隙时,针对某个确定的路由,消息可以最早地到达目的处理机.

证明:在采用存储转发技术通信时,消息在经过中间结点时都必须先存储,然后再转发.按式(1)~(4)选择连接的空闲时间空隙时,消息都是最早地经过中间结点.因此,针对某个确定的路由,消息必定最早地到达目的处理机.

(b) 在采用虫蚀寻径技术时

假定消息 $e_{i,j}$要经过的连接顺序为 $l_{a,b},l_{b,c},\ldots,l_{y,z}$,很显然,$fst(f(v_i))=p_a$,$fst(f(v_j))=p_z$.对于任一连接,

如 $l_{m,n}$，若已有 $\delta_{m,n}$ 个消息 $\{e^1_{m,n},\ldots,e^{\delta_{m,n}}_{m,n}\}$ 要求顺序使用该连接，令 $\Gamma^k_{m,n}=f(e^k_{m,n},l_{m,n})$，$\zeta^k_{m,n}=\Gamma^k_{m,n}+\frac{size(e^k_{m,n})}{\beta}$，则在连接 $l_{m,n}$ 上已被分配的时间空隙 $TS_{m,n}$ 为

$$TS_{m,n}=[\gamma^1_{m,n},\zeta^1_{m,n}]\cup\ldots\cup[\eta^{\delta_{m,n}}_{m,n},\zeta^{\delta_{m,n}}_{m,n}],$$

令

$$TS=TS_{a,b}\cup TS_{b,c}\cup\ldots\cup TS_{y,z}.$$

令 TS 对应着时间轴上被占用的时间空隙数为 δ，为了便于问题的讨论，令

$$TS-[\eta_1,\zeta_1]\cup[\eta_2,\zeta_2]\cup\ldots\cup[\eta_\delta,\zeta_\delta],$$

其中 $[\eta_i,\zeta_i]$ 表示在该时间段至少有一个或多个消息在使用连接 $l_{a,b},l_{b,c},\ldots,l_{y,z}$ 中的一个或多个，则存在一些 k 满足下式，

$$\eta_{k+1}-\max\{scnd(f(v_i))+\Gamma_i+\alpha,\zeta_k\}\geqslant\frac{d_{i,j}}{\beta},\tag{5}$$

其中 $k\in\{0,\ldots,\delta\}$，$\zeta_0=0$，$\eta_{\delta+1}=\infty$.

可知，至少 $k=\delta$ 是满足不等式(5)的，假定 ι 为所有 k 中的最小值，则消息 $e_{i,j}$ 到达目的处理机的较早时刻为

$$\max\{scnd(f(v_i))+\Gamma_i+\alpha,\zeta_\iota\}+\frac{d_{i,j}}{\beta}.\tag{6}$$

根据式(5)和式(6)，不难在所有的路由中确定一个能使消息较早地到达目的处理机的路由.

在采用虫蚀寻径技术时，消息是以异步流水方式到达目的处理机的，不同的连接上对应的时间空隙是不一致的，因此很难用统一的式子来选择不同连接上的空闲时间空隙以使消息最早地到达目的处理机.

本文约定，如有多个路由都能使消息最早(较早)地到达目的地，则选择最短路由中的任意一条.

如何将处理机分配给任务 一般认为，应选择使任务最早开始执行的处理机. 下面给出某个任务在某个处理机上的最早开始执行时间的计算方法.

先定义 $DAT(v_x,p_y)$ 为任务 v_x 从其父结点所接收的消息中最晚到达处理机 p_y 的消息到达的时间，又假定在任务 v_i 分配给处理机 p_j 时已有 δ 个任务 $\{v_{j_1},v_{j_2},\ldots,v_{j_\delta}\}$ 要求顺序使用处理机 p_j，则如果存在一些 k 满足下式：

$$scnd(f(v_{j_{k+1}}))-\max\{scnd(f(v_{j_k}))+\Gamma_{j_k},DAT(v_i,p_j)\}\geqslant\Gamma_i,\tag{7}$$

其中 $k\in\{0,\ldots,\delta\}$，$scnd(f(v_{j_0}))=0$，$scnd(f(v_{j_{\delta+1}}))=\infty$，可知，至少 $k=\delta$ 是满足不等式(7)的. 假定 ι 是所有 k 中满足上述不等式的最小值，则任务 v_i 在处理机 p_j 上的最早开始执行时间为

$$\max\{scnd(f(v_{j_\iota}))+\Gamma_{j_\iota},DAT(v_i,p_j)\}.\tag{8}$$

根据式(7)和式(8)，便可从所有的处理机中选择使该任务能最早开始执行的处理机.

下面，根据上文提供的策略，构造一个非完全互连同构系统上的静态任务调度算法. 本算法在选择参与调度的任务时，采用的是静态关键路径.

算法 1. 调度算法

① 根据静态关键路径，创建任务优先级表.

(a) $\forall\ v_i$，计算 $tlevel(v_i)$ 与 $blevel(v_i)$，并确定一条静态关键路径.

(b) 置任务优先级表为空，取静态关键路径上的第 1 个结点作为当前结点 v_i.

(c) 如果 v_i 的入度为 0，则将该结点作为任务优先级表中最后一个结点插入表中，并把该结点的子结点的入度减 1，转(d). 如果 v_i 入度不为 0，则先对其所有的尚未进入任务优先级表的父结点 v_j 按 $tlevel(v_j)+\Gamma_j+c_{i,j}$ 的大小由大到小排序. 以第 1 个父结点及其尚未进入任务优先级表的祖先结点为子图构成一个有向无环图，递归地创建该子图的任务优先级表，并将该子图的任务优先级表添加到要创建的任务优先级表的末尾. 用类似的方法处理其余父结点构成的子图. 最后再将结点 v_i 加到任务优先级表的末尾，并把该结点的子结点的入度减 1.

(d) 如果当前结点为最后一个结点，则任务优先级表创建完成，否则取静态关键路径上的下一个结点作为当前结点，执行(c).

② 根据每个处理机连接数目的高低，创建处理机优先级表.

③ 取任务优先级表中的第 1 个任务作为当前任务,执行④~⑧步,为该任务选择最早开始执行的处理机.

④ 将处理机优先级表中的第 1 个处理机作为当前处理机.

⑤ 如果当前任务在任务图中不是入结点,则按其父结点完成的先后顺序计算每个父结点发送给当前结点的消息到达当前处理机的最早时间.具体计算分成两种情况,若父结点也在当前处理机上执行,则消息的最早到达时间为父结点的完成时间;若父结点不在当前处理机上执行,则应按照式(1)~(4)或(5)与(6)计算消息的最早到达时间.根据所有消息到达的时间,求出最晚消息到达时间.如果当前任务在任务图中是入结点,则设其最晚消息到达时间为 0.

⑥ 根据式(7)和式(8)计算当前任务在当前处理机上的最早开始时间.如果当前处理机为处理机优先级表中的最后一个处理机,则转⑦;否则从处理机优先级表中取下一个处理机作为当前处理机,转⑤.

⑦ 如果当前任务为任务优先级表中的最后一个任务,则结束;否则,取下一个任务作为当前任务,执行④~⑥步,为该任务选择最早开始执行的处理机.

这里,假定不同处理机之间的通信采用虫蚀寻径技术.根据以上调度算法分析一下如何将图 1(a)中的 DAG 图调度到图 1(b)中的多处理机系统上去.图 1(a)中的静态关键路径为 $v_3v_6v_8v_{10}$,根据算法第 1 步创建的任务优先级表为 $v_3v_1v_6v_2v_5v_8v_4v_7v_9v_{10}$.根据算法第 2 步创建的处理机优先级表为 $p_0p_1p_2p_3$.先调度 v_3,因为它是入结点,故将 v_3 调度到 p_0 上时便能最早地执行.类似地,v_1 被调度到 p_1 上执行.在调度 v_5 时,因为要从父结点 v_1 接收消息 $e_{1,5}$,v_5 在 p_0 上的最早开始时间为 4.0.而 v_5 在 p_1,p_2,p_3 上的最早开始时间均为 5.5,故 v_5 被调度到处理机 p_0 上执行.类似地,v_2,v_6,v_8,v_4 被分别调度到处理机 p_2,p_2,p_0,p_3 上.在调度 v_7 时,它可以被调度到处理机 p_0 上,插到任务 v_3 与 v_6 之间执行,而消息 $e_{4,7}$ 可以插在 $e_{1,6}$ 之前占用连接 $l_{1,0}$.类似地,v_9,v_{10} 被分别调度到处理机 p_2,p_3 上.最后,获得整个 DAG 图的调度长度为 12.5,而整个 DAG 图的串行执行时间为 24.0,调度结果如图 1(c)所示.

现在,分析一下以上算法的复杂度.本算法的复杂度主要体现在③~⑦步,对于每个消息,都要计算从源到达 $|P|-1$ 个目的处理机的最早时间;对于每个任务,都要计算该任务在每个处理机上的最早开始执行时间.在采用存储转发技术时,要计算消息在经过每个中间连接时的最早时间;在采用虫蚀寻径技术时,对于某个确定的路由,要先找出已经占用该路由中任一连接的所有消息,因为不同连接上已占用的消息可能部分相同,故可创建一棵 AVL 树,将不同连接上的消息按占用时间空隙的起止时间为关键字插入到该树中,然后再为要经过该路由的消息分配时间空隙.故在采用存储转发技术时,算法的复杂度为 $O(|P||V|^2+|P||D||R||E|^2)$;在采用虫蚀寻径技术时,算法的复杂度为 $O(|P||V|^2+|P||D||R||E|^2\log(|E|))$,其中 $|D|$ 为网络直径,$|R|$ 为网络中任意两个处理机之间的路由总数的最大值.

4 结束语

在一般情况下,任务调度是 NP-完全问题.目前,仅仅在某些条件非常苛刻的情况下获得了最优解[4].完全按照本文提供的策略去设计非完全互连同构系统上的静态任务调度算法是不难的,但是算法的复杂度可能会令人难以接受.对于如何设计一个较低复杂度,同时又能提供满意调度性能的算法,我们已在这些领域取得了一些进展,对此,我们将在以后的文章中加以介绍.

参考文献

1 Hwang Jing-jang *et al*. Scheduling precedence graphs in systems with interprocessor communication times. SIAM Journal of Computing, 1989,18(2):244~257

2 Wu Min-you *et al*. Hypertool: a programming aid for message-passing systems. IEEE Transactions on Parallel and Distributed Systems, 1990,1(3):330~343

3 Yang T *et al*. PYRROS: static scheduling and code generation for message passing multiprocessors. In: Proceedings of the 6th ACM International Conference of Supercomputing. 1992. 428~437

4 Hesham El-Rewini. Partitioning and scheduling. In: Albert Y H Zomaya ed. Parallel Distributed Computing Handbook. McGraw Hill, Inc., 1996. 239~302

5 Hesham El-Rewini *et al*. Scheduling parallel program tasks onto arbitrary target machines. Journal of Parallel and Distributed Computing, 1990,9(2):138~153

6 Kwok Yu-Kwong *et al*. Bubble scheduling: a quasi dynamic algorithm for static allocation of tasks to parallel architectures. In: Proceedings of the SPDP'95. Available from: http://www.cs.ust.hk

Static Task Scheduling for Incompletely Interconnected Homogeneous Systems

ZHANG Jun ZHANG Li-sheng HAN Cheng-de

(*High Performance Computing Center Institute of Computing Technology The Chinese Academy of Sciences Beijing* 100080)

Abstract In the distributed memory multiprocessor (DMM) systems, communication overhead, involved among tasks run on different processors, is still large which even offsets the advantages brought by multiprocessor parallelism. In order to execute a parallel application efficiently, it is necessary to choose an appropriate scheduling technology to allocate processors to tasks. In this paper, formal description of general scheduling system including task model, processor model and schedule problem is presented. Three most important problems, concerning schedule in incompletely interconnected homogeneous systems, are studied, which are: (1) How to choose scheduling tasks in sequence, (2) How to choose a route, (3) How to allocate processors to tasks. In addition, the problem on how to choose a route is studied according to store-and-forward and wormhole routing respectively. In the end, a static scheduling algorithm for incompletely interconnected homogeneous systems is constructed according to the solutions to the above three problems.

Key words Static task scheduling, task model, processor model, store-and-forward routing, wormhole routing.