基于超立方体的静态任务调度*

章 军 冯秀山 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)

E-mail: fxs@chpc.ict.ac.cn

摘要 该文给出一个基于超立方体的静态任务调度算法.在算法的设计中,首先建立了任务优先级表和处理机优先级表,任务在调度时总是顺次调度高优先级任务,然后再从处理机优先级表中选择能使该任务最早开始执行的处理机.最后,分别给出了基于LU分解的任务图与随机生成的任务图的调度结果.

关键词 静态任务调度,超立方体,有向无环图,任务优先级表,虫道寻径.

中图法分类号 TP316

在所有的多处理机互连中,超立方体是一种较为常见的拓扑结构.国外已有不少并行机系统在进行设计时采用了这种拓扑结构或其变形,如InteliPSC,nCUBE机等.本文给出了基于超立方体的静态任务调度算法.在算法的设计中,首先建立了任务优先级表与处理机优先级表,任务调度总是依次调度高优先级任务,然后再从处理机优先级表中选择能使该任务最早开始执行的处理机.由于超立方体是非完全互连结构,故在调度时需将连接(link)与处理机一样看成是资源加以分配.不同的处理机之间在进行通信时,消息传递采用的是虫道寻径技术.另外,确定任务在某个处理机上的开始执行时间时,选用了最早空闲时间空隙.最后,分别给出了基于LU分解的任务图与随机生成的任务图的调度结果.

1 超立方体多处理机系统上的静态任务调度问题

任意一个有向无环任务图可以用四元组$TG=(V,E,A,D)$来定义,其中V,E,A和D分别表示图中任务结点的集合、通信边的集合、任务计算量大小的集合以及任务之间通信量大小的集合.通信量$d_{i,j}$表示由通信边$e_{i,j}=(v_i,v_j)$引起的通信量的大小,或者说,任务v_i向任务v_j传输的数据量的大小.值a_i表示任务$v_i\in V$的计算量的大小.在任务图中,任务v_i的父结点构成的集合记为$PARENT(v_i)$,而任务v_i的子结点构成集合的记为$CHILD(v_i)$.对于任务v_i,若$PARENT(v_i)=\varnothing$,则v_i称为入结点;若$CHILD(v_i)=\varnothing$,则v_i称为出结点.由入(出)结点构成的集合记为$ENTRY(EXIT)$.

任意一个n维的超立方体多处理机可采用二元组$PG=(P,L)$,其中P,L分别表示所有处理的集合与连接的集合.若$l_{i,j}\in L$,则表示处理机p_i与p_j之间有直接连接.对于任意一个n维的超立方体多处理机,处理机的个数$|P|$为2^n,连接的数目为$n2^n$,网络的直径$D=n$.

在本文中,假定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.假定每个处理机的运算速度为s,则任务图TG中任务v_i的计算时间$\Gamma_i=\frac{a_i}{s}$.在不同的处理机之间进行通信时,消息传递采用的是虫道寻径技术.在不存在链路堵塞(link contention)的情况下,若任务v_i向任务v_j发送数据量的大小为$d_{i,j}$,通信延迟可用公式$c_{i,j}=\alpha+\frac{d_{i,j}}{\beta}$来计

---

* 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,博士生,主要研究领域为并行计算,并行任务调度,计算机体系结构.**冯秀山**,1973年生,硕士生,主要研究领域为计算机体系结构,数字信号处理.**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:冯秀山,北京100080,北京2704信箱25分箱

本文1998-10-12收到原稿,1998-12-25收到修改稿

算,其中 α 和 β 分别表示通信时的启动开销和通道的数据传输率.

在非完全互连系统上,调度的含义有两个方面,一方面是将任务分配给处理机去执行,另一方面是将连接分配给消息去使用.衡量调度性能的最主要的指标是调度长度.

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的最长路径(包含任务之间的通信时间),由静态关键路径上的结点构成的集合记为 $SCPN$.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 $blevel(v_x)$).SCP 的长度为 $\max_{v_x \in V}\{tlevel(v_x)+blevel(v_x)\}$,任意结点 v_i 的 tlevel 与 blevel 的计算方法如下:

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

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

直观地,一个结点的 tlevel 值越小,则说明该结点越有可能先被调度;一个结点的 blevel 的值越大,则说明该结点越应先被调度.

2 调度算法介绍

迄今为止,在已发表的有关静态任务调度的文章[1,2]中,绝大多数考虑的是如何将任务调度到完全互连的多处理机系统上去,而直接将任务调度到非完全互连系统上去的文章[3]甚少.在下面给出的调度算法中,我们将直接把实际的处理机与连接看成是资源,把任务与消息看成是消费者,同时考虑如何将任务分配给处理机以及如何将消息分配给连接.基于超立方体的静态任务调度要解决的问题有 3 个,即如何按顺序选择参与调度的任务、如何选择路由以及如何将任务分配给处理机.

2.1 如何顺序选择参与调度的任务

本文中总是优先调度关键路径上的任务.为了降低算法的复杂度,在确定关键路径时,本文采用的是静态关键路径.对于非关键路径上的任务总是优先考虑 $slevel=blevel-tlevel$ 值大的任务.在任务调度中,任务的选择总是按拓扑顺序进行的.在下面的算法中给出了一个建立任务优先级表的算法.在算法的实现中,需增加两个虚结点,通过第 1 个虚结点(即下面算法中的 v_0)可以访问所有的入结点,通过第 2 个虚结点可以访问所有的出结点.

算法 1. 创建任务优先级表

(1) 从集合 $\{v_i \mid v_i \in CHILD(v_0) \wedge v_i \in SCPN\}$ 中选择任一结点 v_e,使之成为任务优先级表中的第 1 个任务.

(2) 从结点 v_e 的子结点中选择一个位于静态关键路径上的结点作为下一个候选结点 v_i,如果有多个这样的子结点,则选择与结点 v_e 有最大通信的子结点.

(3) 如果结点 v_i 的所在父结点都在任务优先级表中,则将结点 v_i 置于任务优先级表的表尾,转(4);否则,假定 v_j 是结点 v_i 的那些不在任务优先级表中的父结点之一,且 v_j 在这些父结点中具有最大的 slevel 值.如果有多个这样的父结点,则选择与结点 v_i 有最大通信的点.如果 v_j 的父结点都在任务优先级表中,则将结点 v_j 置于任务优先级表的表尾;否则递归地将 v_j 的所有父结点放入到任务优先级表中.类似地,可以采用这种方法将 v_i 的所有父结点置于任务优先级表中,然后再将 v_i 置于任务优先级表的表尾.

(4) 从结点 v_i 的子结点中选择一个位于静态关键路径上的结点作为下一个候选结点 v_i,如果有多个这样的子结点,则选择与结点 v_i 有最大通信的子结点.

(5) 重复第(3)~(4)步,直到所有的结点都进入任务优先级表为止.

很容易证明,由以上算法建立的任务优先级表总是满足拓扑顺序的,因为每个任务只有在它所有的父任务都入表之后才入表.

2.2 如何选择路由

消息路由采用的是确定性路由,确定性路由是由源与目的确定的最短路径.为了降低算法的算杂度,使用连接服从先来先服务的排队原则.对于超立方体多处理机而言,消息寻径是按确定维序进行的,首先比较源处理机

序号与目的处理机的序号,从低位到高位,每次沿着消除一位地址差异的方向前进.对于两个不同的处理机 p_n 和 p_m,二者之间的路由记为 $ROUTE(p_n,p_m)$,该路由的长度记为 $LENGTH(p_n,p_m)$.任何消息要使用路由 $ROUTE(p_n,p_m)$的条件是该路由上所有的连接均处于空闲状态.在下文中,记 $IDLE(l_{i,j})$为连接 $l_{i,j}$将处于空闲状态的时刻.

2.3 如何将任务分配给处理机

本文主要考虑选择使任务最早开始执行的处理机.在将任务调度分配给处理机去执行时,总是顺序遍历已创建好的处理机优先级表,选择最合适的处理机.在创建处理机优先级表时,先由超立方体生成一棵以零号处理机为根的树,然后再以广度优先遍历的方法遍历该生成树,从而建立处理机优先级表.在由超立方体生成树时,假定树中开始时只有零号处理机,然后递归地将与树叶处理机距离为1,但尚未出现在该树中的处理机作为该树叶处理机的子处理机插入到树中.

在下文中,分别记 $P(v_i)$和 $ST(v_i,p_m)$为分配给任务 v_i 的处理机与在任务 v_i 下处理机 p_m 的最早开始执行时间.在确定任务 v_i 在处理机 p 上的最早执行时间时,应为任务分配最早的空闲时间空隙.记处理机 p 上的空闲时间空隙集合为 $ITS(p)$.

在解决了以上3个问题后,下面给出调度算法的描述.

算法2. 调度算法

(1) 初始化.

〈a〉 $\forall\ v_i\in V$,计算 $tlevel(v_i)$,$blevel(v_i)$,$slevel(v_i)$,并计算任务图的串行计算时间.

〈b〉 $\forall\ p_i\in P$,赋 $ITS(p_i)$为$[0,\infty)$.$\forall\ l_{i,j}\in L$,赋 $IDLE(l_{i,j})$为0.

(2) 创建任务优先级表与处理机优先级表.

(3) 从任务优先级表中取第1个任务,假定它为 v_i,$\forall\ p_m\in P$,计算 $ST(v_i,p_m)$.

(4) 设 $ST(v_i,p_c)=\min_{p_m\in P}ST(v_i,p_m)$.如果有多个处理机满足该条件,则根据下述方法按顺序选择处理机:

〈a〉 选择处理机 p_c,如果将任务调度到该处理机上,则通信与路由长度乘积之和较小,即 $\Sigma_{v_j\in PARENT(v_i)}(d_{j,i}\times LENGTH(P(v_j),p_c))$.

〈b〉 选择具有较高优先级的处理机 p_c.

(5) 计算任务 v_i 在处理机 p_c 上的完成时间 $FT(v_i,p_c)$,如果 $FT(v_i,p_c)$超出了程序的串行时间,则将所有的任务调度到一个处理机上去执行并退出调度.

(6) 〈a〉 修改 $ITS(p_c)$;

〈b〉 $\forall\ v_j$,如果 $v_j\in PARENT(v_i)$,则修改路由 $ROUTE(P(v_j),p_c)$上的所有连接的 $IDLE$ 值;

〈c〉 从任务优先级表中将任务 v_i 删除.

(7) 重复第(3)~(6)步,直到所有的任务都调度完毕.

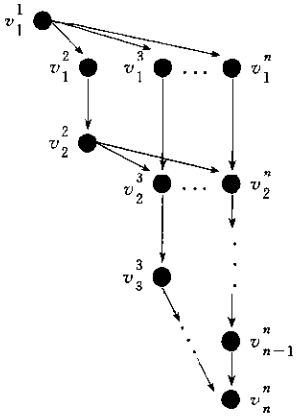
在上述算法中,最主要的时间花在第(3)~(6)步.对于每个任务,都要计算该任务在每个处理机上的消息最晚到达的时间,以寻找相应的最早空闲时间空隙.故该算法的复杂度为 $O(|P|\ |V|^2+|E|\,|P|\log_2|P|)$,其中 $\log_2|P|$为网络的直径.

3 模拟试验

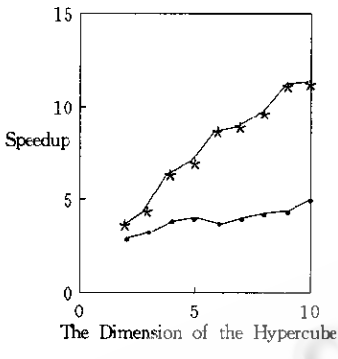
图1(a)是在不考虑选主元情况下LU分解的DAG图.假定该矩阵的大小为10000×10000,分块矩阵的大小为200×200,故在该任务图中共有1 275个任务.图中每个任务的计算量大小以及任务之间通信量的大小可由手工分析得出.假定处理机的运算速度为50Mflops,浮点加减与乘除均花费一个时钟周期,计算获得任务图的串行执行时间为13 337.62s,由入结点到出结点不考虑任务之间通信时间的最大路径长度为606.41s.

图1(b)分别给出了在慢速与快速通信两种情况下,运行调度程序获得的加速比与超立方体维数的关系图,其中加速比等于串行时间除以调度长度.所谓的慢速通信是假定通信时的启动开销为0.5ms,数据传输率为每秒100 000个浮点数/s.而快速通信是假定通信时的启动开销为500ns,数据传输率为1 000 000个浮点数/s.总的来说,随着处理机个数的增加,加速比有增大的趋势,快速通信时获得的加速比高于慢速通信时的加速比.另外,图中还出现了波动现象.经分析,这不仅与确定性路由寻径有关,还因为在处理机个数较多时,一开始任务便

分散到不同的处理机上去，导致以后调度产生的通信开销较大.



(a) LU分解的任务图



(b) LU分解的调度结果

图1

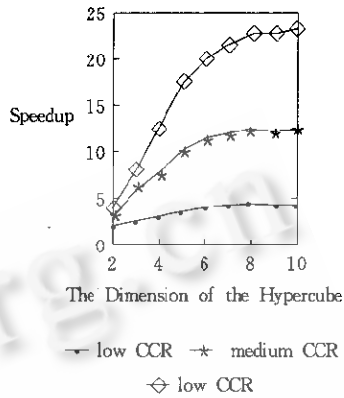


图2 随机产生的任务图的调度结果

随机生成的任务依赖图可以通过控制结点数、结点的平均计算时间与平均通信时间的比值(CCR)、任务图的层数、结点的平均入度等参数自动生成.在试验中，选定结点数目的范围为1 200～1 600，任务图的层数为30～40，每个结点的平均入度为1～3，通信时的启动开销$\alpha=0.5$，数据传输率$\beta=1$，处理机的计算速度$s=1$.试验的结果见图2，图中的3条曲线分别对应着3种不同的CCR，低CCR为0.25～0.33，中等CCR为1～1.1，高CCR为3～4，CCR值反映了任务划分的粒度.针对每种CCR值，都随机地生成50个任务图，然后将任务图调度到n维的超立方体多处理机上去，故图中每一点的加速比都对应着50个加速比的平均值.由图中的调度结果可知，CCR值越高，则调度所能获得的加速比越高，故调度的长度在很大程度上受到任务划分的粒度影响.另外，随着处理机个数的增加，加速比有增大的趋势.但是，在处理机个数增加到一定程度后，加速比的增长变缓，这受到了任务图中固有的并行度的限制.

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 Hesham El-Rewini *et al*. Scheduling parallel program tasks onto arbitrary target machines. Journal of Parallel and Distributed Computing, 1990,9(21):138～153

Static Task Scheduling for Hypercube Interconnected Multicomputers

ZHANG Jun FENG Xiu-shan HAN Cheng-de

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

Abstract In this paper, an effective static task scheduling algorithm for hypercube interconnected multicomputers is presented. In the design of this algorithm, two priority lists are built for tasks and processors. During the schedule, tasks are selected from the task priority list in sequence. For a selected task, the processor on which it can be executed the earliest is assigned. In the end, the schedule results of LU decomposition and randomly generated task graphs are given and analyzed.

Key words Static task scheduling, hypercube, directed acirclic graph (DAG), task priority list, wormhole routing.