工作站网络上协作任务的调度*

齐 红 鞠九滨

(吉林大学计算机科学系 长春 130021)

摘要 在利用工作站群集系统进行的协作模式并行计算中,任务调度在很大程度上决定并行计算的性能.本文给出了一个任务调度的模型和算法,它考虑了协作模式并行计算中任务间的同步时间、通信时间、数据加载及结果收集时间.根据这个调度模型,可以选择一组并行执行时间最短的工作站,从而获得较好的并行计算性能.

关键词 工作站群集系统,协作模式并行计算,任务调度.

中图法分类号 TP311.1

支持分布式超级计算的工作站群集系统是当今并行计算的一个重要发展方向.它具有优越的性能、良好的扩展性和并行 I/O 等特点.支持工作站群集系统进行并行计算的并行程序设计的环境中,一个重要的问题就是任务调度,它在很大程度上决定并行计算的性能.

D. C. Marinescu 等[1]将分布并行计算分为两种模式.一种是基本模式,即将一个大计算量的任务分解为一些相互独立的子任务,在一组工作站上并行执行,最后将子任务结果汇集为总的结果.另一种是协作模式,即在计算过程中,子任务间需要同步,交换中间结果,因而要求基本上同时开始,并以同样速度执行.

由于在协作模式并行计算中各子任务间需要同步,因而当所使用的工作站之间负载相差悬殊时,可能造成各子任务间在计算过程中相互等待,影响并行计算的速度.因此,对于协作模式的并行计算问题,在进行任务调度时,要根据任务运行时系统中各工作站的负载情况,选择一组最佳的工作站,使任务的执行时间最短.这方面的工作有:M. J. Atallah 等[2]提出的群调度模型和算法,它可以使单个任务的执行时间最短;Kemal Efe 等[3]从系统资源利用率角度出发,提出的一种使任务的平均响应时间最短的群调度算法.

上述两个模型和算法都忽略了子任务间的通信开销以及任务的串行处理(如数据的加载和结果的收集)的开销.虽然由于高速网络的出现使通信延迟变得很小,但在实际的应用问题中通信开销往往还是很可观的.B. K. Schmidt 等[4]给出了一种度量通信开销的方法,它根据应用问题并行算法的通信模型及数据量,估算出应用问题在使用不同数目的工作站时的通信开销.将这些信息加入到调度模型中,可以使调度算法更有效,获得更好的并行计算性能.

本文在上述工作基础上,给出了一个任务调度的模型和算法,它考虑了同步开销、通信开销、数据加载及结果收集开销的影响.根据这个调度模型,我们在网络计算环境 PVM(parallel virtual mechine)[5,6]中实现了一个任务调度系统,并使用这个调度系统进行协作模式的并行计算,获得了最大加速比.

1 任务调度模型和算法

在工作站群集上的分布并行计算中,有很大一类并行算法属于协作模式的并行计算,例如:"Pipe-Multiply-Roll"方法的矩阵乘、"Split-Sort-Merge"方法的并行排序、波动方程及大型线性方程组等使用迭代方法进行并行求解的数值计算问题.

本文考虑的任务划分方法是数据划分.若系统中有 P 台工作站可用,则将整个数据域划分为 P 个子域,使 P 台工作站工作在各个子域上,运行同样的程序,即使用 SPMD 编程模型.

我们选择 CPU 利用率作为衡量工作站负载的指标.如果系统共有 Q 台工作站,每台工作站的 CPU 利用率用 η_i 表示.η_i 定义为正在执行的任务(包括本地任务和远程任务)所占用的 CPU 时间占全部 CPU 时间的百分比,则 $0 \leqslant \eta_i$

---

* 本文研究得到国家自然科学基金资助.作者齐红,女,1970 年生,助教,主要研究领域为分布并行计算.鞠九滨,1935 年生,教授,博士导师,主要研究领域为分布系统与网络软件.

本文通讯联系人:齐红,长春 130021,吉林大学计算机科学系

本文 1995-12-11 收到原稿,1997-06-09 收到修改稿

≤1. 假设一次迭代在一台完全空闲的工作站($\eta_i=0$)上的执行时间是 T_o,那么当 $\eta_i>0$ 时,执行时间为 $T_o/(1-\eta_i)$. 各子任务在每次迭代后要交换中间结果,才能开始下一次迭代. 如果选择 Q 台工作站的一个子集 G 来执行整个任务,CPU 利用率最高的工作站进行一次迭代所用的时间最长,其它所有子任务都要等待它结束计算后,才能交换中间结果,开始下一次迭代. 因而子集 G 中各工作站 CPU 利用率的最大值 $\eta(G)=\max_{w_i\in G}\eta_i$ 成为系统性能的瓶颈. 这样完成一次迭代的并行执行时间为 $T_o/(1-\eta(G))$. 一般地,一个任务的并行执行时间可以表示为:

$$T(P)=T_s(P)+T_c(P)+T_r(P)+T_o(P)/(1-\eta(G)) \tag{1}$$

其中 P 表示 G 中工作站的数目,$T_s(P)$为数据的加载时间,$T_o(P)$表示 P 台工作站全部空闲时并行完成所有迭代的时间,$T_c(P)$为子任务间的通信时间,$T_r(P)$为结果收集时间.

$T_s(P)$,$T_c(P)$和 $T_r(P)$都与通信有关,所以只有定量地度量通信时间,才能准确估算任务的并行执行时间. 在基于消息传递的分布系统中,一次通信的时间是所要传递的数据量 N 的线性函数[7]:

$$T_{comm}=\alpha+\beta N$$

这里,α 是启动时间,β 是每个字节的传送时间,N 是要传送的字节数. α 和 β 可以通过实验测得. 这样,对于具体应用问题的通信模型,可以得到所要传送的字节数及通信次数,计算总的通信时间.

例如"Pipe-Multiply-Roll"方法[8]的矩阵乘,它的计算时间及通信时间分别如下:

$T_s(P)=\sqrt{P}\,(\alpha+4\beta N^2)$

$T_o(P)=\sqrt{P}\,[(N/\sqrt{P})^3T_{mult}+(N/\sqrt{P})^3T_{add}]=N^3/P(T_{mult}+T_{add})$

$T_c(P)=\sqrt{P}\,\{\sqrt{P}\,[\alpha+4\beta(N/\sqrt{P})^2]-[\alpha+4\beta(N/\sqrt{P})^2]\}=\sqrt{P}\,(\sqrt{P}+1)(\alpha+4\beta N^2/P),\sqrt{P}>1$

$T_r(P)=(\alpha+4\beta N^2/P)$

其中 T_{mult}和 T_{add}分别是单个乘和加操作的执行时间(下同),N=〈矩阵的维数〉,P=〈所用工作站数〉.

而一维波动方程[8]的计算时间及通信时间则为:

$T_o(P)=nstep*(N/P)(3T_{mult}+4T_{add})$

$T_c(P)=2*nstep(\alpha+8\beta)$

$T_r(P)=P[\alpha+8\beta(N/P)]$

其中 N=〈波动方程的空间坐标点数〉,$nstep$=〈波动次数〉,P=〈所用工作站数〉.

可见,通信开销与数据量和所用的工作站数有关,而计算时间不但与数据量和工作站数有关,还与各工作站的负载情况有关. 因而,在进行任务调度时,要同时考虑上述因素,选择使并行执行时间最短的一组工作站.

任务调度算法的目的是寻求使执行时间最短的工作站子集 G. 算法首先对系统中 Q 台工作站提供的 CPU 利用率 $\eta_{i1},\eta_{i2},\ldots,\eta_{iq}$进行排序,得到 $1,2,\ldots,q$ 的一个排列 π,使得 $\eta_{\pi(1)}<\eta_{\pi(2)}<\ldots<\eta_{\pi(q)}$. 然后根据式(1)分别计算使用不同数目工作站的并行执行时间,从中选出执行时间最短的工作站子集 G.

2　任务调度系统的实现

我们的任务调度系统是在网络计算环境 PVM 之上实现的,主要由两部分组成. 一部分是运行于系统中每台工作站上的负载服务员,它每 10s 调用系统提供的函数 rstat,根据 rstat 返回的信息,计算出 10s 之内本地 CPU 利用率的平均值,作为调度系统的负载指标,同时负载服务员还等待回答关于负载信息的请求;另一部分是以函数形式给出的任务调度实现模块,由负载收集子模块、工作站群选择子模块、任务派生子模块及结束处理子模块组成. 用户在编写应用程序时,可以象使用其它 PVM 函数一样调用这个函数. 它能在任务运行时动态地选择一组工作站,并将子任务派到这组工作站上并行执行.

任务调度的过程是首先应用程序启动任务调度模块,由负载收集子模块向系统中每台工作站发广播报文,其中包含发动者的主机名及 CPU 利用率的门槛值. 这个门槛值是满足用户要求的最低加速比时,工作站集 G(假定使用所有的工作站)中各处理器 CPU 利用率的最高值.

系统中各工作站的负载服务员收到广播报文后,检查本地保存的标志 id. 若 id 为 −1,表明未被占用. 再检查本地 CPU 利用率是否小于门槛值. 只有 id 等于 −1,并且 CPU 利用率低于门槛值的工作站才发回答报文给发动者. 然后将标志 id 置为请求报文中的任务号 tid,表明将使用权提交给这个任务. 这时若接收到另一个任务的请求广播,由于本地的 id 值大于 0,因而不再回答,从而避免了冲突. 而那些 id 值大于 0 或者 CPU 利用率高于门槛值的工作站则不回答.

发动者的负载收集子模块接收各工作站的回答报文后,将各工作站的主机名及 CPU 利用率等信息传给工作站群

选择子模块.后者根据前边给出的调度算法,选择一组执行时间最短的工作站,并且向系统中各工作站发一占用报文,包含有占用任务的任务号,占用的工作站数及各占用工作站的主机名.

系统中各主机的负载服务员收到报文后,检查本地id是否等于报文中的tid.若等于,则表明使用权已被提交给此任务.这时再检查host中是否有本工作站的主机名.若有,则表明将被此任务占用,进行并行计算;若没有,则表明未被选用,将本地id置为-1,以便接受其它任务的请求.若id不等于tid,则表明是其它任务占用请求,不予理采.

发动者发送占用报文后,由任务派生子模块使用PVM的进程启动函数将子任务派到所选的工作站上,开始进行并行计算.

在任务执行结束后,由结束处理子模块发释放广播报文.

各工作站的负载服务员收到这个报文后,检查本地id是否等于报文中的tid,若等于,则表明占用本工作站的任务已结束,则id置为-1,以便让其它用户使用;若不相等,则表明此时结束的任务并未占用本工作站,不予处理.

3 性 能

为了验证第1节中的计算模型,我们在由10Mbps以太网连接的9台Sun4/65工作站上运行矩阵乘和一维波动方程的求解程序,测量其执行时间,并与模型的计算值比较.

首先测量α和β值.α和β的大小与工作站的速度、所用通信协议、网络带宽及网络负载等因素有关.由于网络负载是不断变化的,因而我们进行多次实验,通过大量的实验数据计算得到,在上述网络环境中使用PVM进行通信时的α约为5.380ms,β约为0.00286ms.这样,

$$T_{comm}=5.380+0.002\ 86N\ \text{ms}$$

使用我们的任务调度系统,可以根据任务运行时系统的负载情况,估算使用不同数目的工作站的并行执行时间,从中选出一组并行执行时间最短的工作站,进行并行计算.实验证明,计算值与实际测量值接近程度很好.图1和图2分别是使用此调度系统进行矩阵乘和求解波动方程得到的测量结果和计算值的比较.

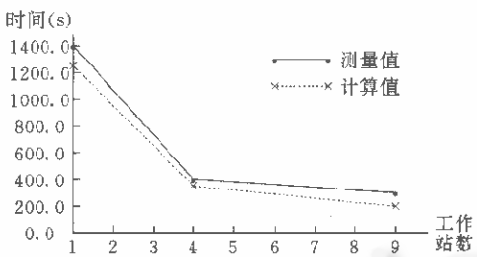

图1 计算500×500的矩阵乘的执行时间:计算值和测量值

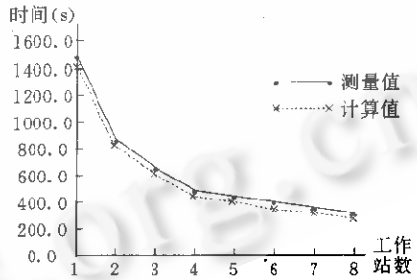

图2 求解空间坐标为100 000个点的一维波动方程的执行时间

本任务调度系统的调度目标是并行执行时间,因而并不一定使用最多的工作站,而是选择执行时间最短的一组.因为在某些负载分布下,如果不加选择地使用尽可能多的工作站,由于各工作站负载相差悬殊,造成子任务间在计算过程中相互等待,延长了整个任务的执行时间.

为了证明上述事实,我们设计了下面的实验.系统中有8台工作站($W1$~$W8$),它们的负载分布如表1所示.使用本调度系统进行并行计算的执行时间和使用全部8台工作站的执行时间如表2所示.比较这两个执行时间,可以看出,使用本文的调度系统可以获得最短的执行时间.

表1 各次测试系统中各主机的CPU利用率(%)

	$W1$	$W2$	$W3$	$W4$	$W5$	$W6$	$W7$	$W8$
第1次	<5	<5	<5	<5	<5	<5	<5	30
第2次	<5	<5	<5	<5	<5	<5	30	60
第3次	<5	<5	<5	<5	<5	30	60	80

表 2　在不同负载分布情况下,使用本调度系统的执行时间与使用全部工作站的执行时间的比较

	使用本任务调度系统			使用全部 8 台工作站的测量值(s)
	所选工作站	计算值(s)	测量值(s)	
第 1 次	W1～W7	35.77	36.30	50.50
第 2 次	W1～W6	38.67	43.80	53.63
第 3 次	W1～W5	44.69	45.28	56.55

4　相关工作

本文的工作是在网络计算环境 PVM 中实现了一个任务调度系统,它可以有效地支持协作模式的并行计算.

与本文的调度算法类似的工作有 M. J. Atallah 等人提出的 Static 算法[2]以及 Kemal Efe 等人提出的 Freelist 算法.[3]与这两个群调度算法比较,主要的差别有两点.首先,使用的负载指标不同.本调度算法使用 CPU 利用率作为负载指标.Static 算法使用工作周期作为负载指标,它定义为本地任务所使用的周期与大计算量任务可用周期之比.Freelist 算法的负载指标定义为本地任务所用周期占整个周期的百分比.与这两个负载指标相比,CPU 利用率更加直观,并且更容易获得.其次,上述两个算法都忽略了通信开销,而本调度算法则将通信开销进行了定量的分析,因而在调度协作模式并行计算任务时,会更有效.

B. K. Schmidt 等人[4]精确测量了常数 α 和 β,给出几类典型应用问题的通信模型,定量地分析了这些应用问题的并行执行时间,但在分析实际任务的计算时间时,未考虑各工作站的负载情况.在协作模式并行计算中,负载的均匀程度直接影响并行执行时间.本文的工作则定量分析了系统中的负载分布对计算时间的影响,因而更加符合实际情况,对并行执行时间的估算更为准确.

5　结　论

本任务调度系统适用于协作模式的并行计算问题,使用它进行任务调度,可以获得较好的并行计算性能.当前我们只实现了相同负载分布,即将一个任务划分为相同大小的子任务,进行并行计算.这对于一般的计算问题足以获得较好的性能.实现不均匀负载分布可以更加有效地发挥资源的利用率,但难度很大.

参考文献

1　Marinescu D C *et al.* Distributed supercomputing. In: Proceedings of the 10th International Conference on DCS, 1990. 381～387

2　Atallah M J *et al.*. Co-scheduling compute-intensive tasks on a network of workstations: model and algorithms. In: Proceedings of the 11th International Conference on DCS, 1991. 344～352

3　Efe K, Schaar M A. Performance of co-scheduling on a network of workstations. The 13th. International Conference on DCS, 1993. 525～531

4　Schmidt B K, Sunderam V S. Empirical analysis of overheads in cluster environments. Concurrency: Practice and Experience, 1994,6:1～32

5　Beguelin A. A user's guide to PVM(parallel virtual machine) technical report. ORNL/TM-11826

6　Geistand G A, Sunderam V S. Network based concurrent computing on the PVM system. Concurrency: Practice and Experience, 1992,4(4):293～311

7　Bomans L, Roose D. Benchmarking the iPCS/2 hypercube multi-processor. Concurrency: Practice and Experience, 1989,1(1): 3～18

8　Fox G C *et al.* Solving problems on concurrent processors. Englewood Cliffs: Prentice Hall, 1988

Scheduling Cooperative Tasks on a Network of Workstations

QI Hong　JU Jiu-bin

(*Department of Computer Science　Jilin University　Changchun*　130021)

Abstract　In the cooperative concurrent computing on a workstation cluster system, task scheduling determines the performance in large degree. This paper gives a model and an algorithm of task scheduling, in which the costs of synchronization, communication, loading data and collecting results are considered. Using this scheduling model, a group of workstations with the shortest executing time of jobs can be selected to obtain better performance of parallel computing.

Key words　Workstation cluster, cooperative concurrent computing, task scheduling.

Class number　TP311.1