一个并行计算支持环境的设计与实现*

陶 杰 鞠九滨

(吉林大学计算机科学系 长春 130023)

摘要 介绍了一个并行计算的支撑环境PCSE(parallel computation supporting environment).通过引入FORK与JOIN原语,使得无论是使用过程程序设计语言,还是使用逻辑程序设计语言编写的用户源程序都能在本系统上并行执行,从而为用户进行高速计算提供了一个很好的软件基础.目前,PCSE支持FORTRAN,C以及PROLOG应用程序的并行执行.

关键词 并行计算,分布式程序设计,网络工作站.

中图法分类号 TP311

计算机的一个重要的应用领域是科学计算.天气预报、人造卫星、飞机和轮船的外形设计以及物质结构分析等复杂的运算都离不开计算机.然而,科学计算又是非常费时的.许多问题即使在大型机、巨型机上求解也需花费几天甚至几十天的时间;若是在微机上计算,耗时会更长.目前,大型机、巨型机由于价格昂贵,还不能普遍应用,人们更多的是依靠微机(尤其是工作站)进行科学计算.UNIX工作站已用得非常普遍,很多用户拥有一批网络连接的工作站,但这些工作站大部分时间是空闲的.因此,人们提出了使用网络空闲工作站进行并行计算的方法.然而,目前人们所研制的并行系统都是针对某一种或一类程序设计语言,还没有一个统一的、适合于各种语言(包括过程程序设计语言和逻辑程序设计语言)的并行系统.科学计算使用的语言种类繁多,有必要设计一个统一的计算模型.本文所描述的支撑环境PCSE(parallel computation supporting environment)即是为此目的而研制的.通过提供两条功能强大的进程操作原语FORK和JOIN,使得无论是用FORTRAN,C,还是用PROLOG语言书写的程序都能够在网络工作站群上并行执行,从而提高了计算速度.

1 FORK和JOIN的语义

包含并行性的程序在网络环境下并行执行时,需要进行并行任务的派生与汇合.并行任务的派生就是使一个任务在执行的同时,派生出可与它并行执行的其他任务,分配给不同的处理机完成.这些任务的执行时间可能是不同的,需要等它们全部完成后将结果汇合起来才能进行后续任务.后续任务可以是单一的,也可以是新的并行任务.如果是新的并行任务,又要进行派生与汇合,如此下去,直至整个程序结束.

FORK语句的形式是FORK l,其中l是新进程的标识符.在FORTRAN程序中,l是一个语句标号;在C程序中,l是一个特定的标记;在PROLOG程序中,l是一个谓词.执行一个FORK l语句时,派生出标识符为l的新进程.具体地说,就是准备好启动和执行该进程所必需的有关信息,将空闲的处理机分配给该进程.如果没有可用的空闲处理机,则让该进程进入排队栈等待.FORK派生的计算还可以再派生,因此,FORK可嵌套,以便充分利用处理机资源,使各处理机都处于忙碌状态.特别是对于PROLOG程序,同一子句中能并行(本系统只实现了AND并行)的子目标不多,但程序中递归现象却很常见.因此,对于PROLOG程序,必须允许FORK嵌套,这样才能获得较高的性能.

与FORK语句相配合,JOIN语句的形式是JOIN n,其中n为已派生出的并发进程的个数.JOIN语句附有

---

* 本文研究得到国家自然科学基金资助.作者陶杰,女,1966年生,副教授,主要研究领域为分布计算系统.鞠九滨,1935年生,教授,博士生导师,主要研究领域为分布计算系统,计算机网络.

本文通讯联系人:陶杰,长春 130023,吉林大学计算机科学系

本文1997-06-09收到原稿,1998-03-03收到修改稿

一个计数器，其初始值为0.每当一个被派生的进程得到结果后，计数器的值加1，并与n比较.若计数器的值小于n，表明此进程不是并发进程中的最后一个，则将该进程占用的处理机释放出来，以执行正在排队栈中等待的其他任务，如果排队栈已空，则让该处理机空闲.如果计数器的值等于n，则表明所有并发进程都已得到结果，此时，将结果汇合起来，继续执行后续任务.

FORK和JOIN两个原语需要由用户加入其原程序中，但这并不会给用户增加太大的负担.下面左边一列是矩阵乘的FORTRAN原程序和斐波纳契数求解的PROLOG原程序，右边一列是加入FORK和JOIN后的并行程序.

```
      DO 10 J=0,7
10    DO 20 I=0,7
      C(I,J)=0
      DO 30 K=0,7
30    C(I,J)=C(I,J)+A(I,K)*B(K,J)
20    CONTINUE
```

```
      DO 5 J=0,6
5     FORK 10
      J=7
10    DO 20 I=0,7
      C(I,J)=0
      DO 15 K=0,7
15    C(I,J)=C(I,J)+A(I,K)*B(K,J)
20    CONTINUE
      JOIN 8
```

```
fib(1,1).
fib(2,1).
fib(N,M):-N1 is N-1,N2 is N-2,
          fib(N1,M1),fib(N2,M2),
          M is M1+M2.
```

```
fib(1,1).
fib(2,1).
fib(N,M):-N1 is N-1,N2 is N-2,
          FORK(fib(N1,M1)),
          fib(N2,M2),JOIN 2,
          M is M1+M2.
```

上述FORTRAN并行程序意味着有8个进程并行执行，每个进程求解C矩阵的一列.在这里，每个进程执行的程序是一样的，即10到20之间的语句，但它们的结果并不一定在同一时刻得出.当所有进程都得出结果后，要在主处理机(用户提交任务的处理机)上对结果进行汇合，然后再进行后面的任务.上述PROLOG并行程序意味着fib(N1,M1)将远程执行，fib(N2,M2)将本地执行，而且在执行fib(N1,M1)和fib(N2,M2)时，可能还会继续派生，如此最大程度地开发并行性.

2 系统实现

2.1 预编译

用户提交的并行程序必须经过预编译的过程，以产生可并行的子任务，从而在网络工作站上并行执行.预编译时，每当扫描到一个FORK语句，就首先对与其配对的JOIN间的语句进行粒度检查.由于系统中存在着额外开销，一些小任务不适合远程执行，因此对并行任务必须进行粒度控制.文献[1]所描述的用FORK和JOIN语义实现的分布式Prolog系统，就是因为没有进行粒度控制，才导致加速比不高.对于过程程序设计语言，用户派生的任务中一般都含有循环语句，因此，我们通过循环次数和循环体内语句条数控制粒度.如果粒度低于预先给定的数值，那么，任务中即使有FORK语句，也不执行FORK操作.而对于逻辑程序设计语言，用户要派生的任务往往都是递归程序，因此，我们通过递归深度来控制粒度(详见文献[2]).

粒度检查合格后，即粒度足够大时，就对FORK与JOIN间的语句形成一个并行子任务.对于FORTRAN和C等过程程序设计语言，子任务中不仅要包含FORK标识符与JOIN间(不包括JOIN)的所有语句及其中的所有子过程(相当于C语言中的函数，下同)，还要包括FORK前的所有全局变量.全局变量作用于所有过程，为多个过程所共享.当将原程序分解为多个可并行子任务时，为确保生成的每个模块都能独立编译和正确执行，必须将原程序的全局变量作为每个子任务的全局变量.已经赋值的全局变量还需代入已得到的值[3].此外，还要在子任务中加入过程定义语句，使每个子任务以过程形式出现.而对于PROLOG这样的逻辑程序设计语言，子任务中要包含整个原程序，派生出的谓词作为提问的形式出现.出于简单性原因，子任务间不允许有数据依赖，以减少机间的通信.

2.2 进程结构

在PCSE中,我们把能够派生出子任务的进程称做顾客进程,不能派生出子任务的进程称做服务进程.由于派生的任务要有粒度控制,因此,必须对顾客进程中的FORK原语做粒度检查,如果粒度不够,FORK将不被执行,子任务也不能派生.所以,一个顾客进程可能会变成一个服务进程.在用户提交任务的主机上运行一个主进程和一个中央进程.主进程是一个顾客进程,它接受用户提交的任务.中央进程负责任务分配和资源管理.当FORK被执行时,它向中央进程请求要远程执行一个进程,中央进程将为其选择一个空闲处理机.如果没有空闲处理机,中央进程则将该请求送入任务队列中排队.中央进程内维持着一个任务队列、一个空闲处理机表、一个进程阻塞表和一个顾客进程表.任务队列存放等待执行的并行任务;空闲处理机表记录可用的空闲处理机,其初始值由用户指定;进程阻塞表中记录被JOIN原语阻塞的进程;顾客进程表中记录所有未完的顾客进程的进程号、所在处理机的名字、已收到结果的个数以及结果的存放地.由于PCSE实现了嵌套的进程派生,即一个顾客进程可能再派生出另一个或多个顾客进程,因此,为了分清进程间的父子关系,PCSE赋予每个进程一个全局唯一的任务号,用数字表示.主进程接受的用户任务作为一级任务,任务号为0;由它派生的任务作为二级任务,任务号为01,02,03等;二级任务派生的任务作为三级任务,任务号为三位数,即011,012,021,022,...,依此类推.串行任务保持其原任务号不变.去掉一个任务号的最末一位,即可知道其父任务是谁了.

此外,每台处理机上还运行一个调度员,它负责收集所在处理机上的CPU状态,一旦发现CPU空闲,即刻向中央进程报告,中央进程将向其派送任务队列中的首任务;如果任务队列已空,中央进程则相应修改其空闲处理机表.

每台主机上最多运行3个顾客进程.由于顾客进程可能中途受阻,为了让其他进程占用空闲的CPU资源,中央进程还会在其所在的处理机上启动新的顾客进程.然而,顾客进程随时可能被唤醒,从而导致多个进程争用CPU,使系统性能变坏.所以,PCSE对进程数规定了上限.

2.3 进程通信

PCSE允许嵌套派生,即一个FORK-JOIN对之间还可能存在FORK-JOIN对.因此,一个顾客进程创建后,还不能马上执行其后接受的任务,而必须检查其中的FORK,JOIN语句.具体过程如下(设顾客进程为P):

(1) 如果任务中没有FORK和JOIN对,P将变为服务进程,其上任务经编译、链接形成目标码后作为串行任务本地执行.

(2) 如果遇到FORK原语,P先将此FORK之前未执行的语句形成串行任务本地执行,然后对FORK进行粒度检查.若粒度不够,则将此FORK,JOIN对去除后,P继续对后续语句进行预编译.否则,生成一个并行子任务,向中央进程发送处理机请求信号.中央进程首先将P加入顾客进程表中,然后为其寻找空闲机.如果找到,中央进程向P回送任务号和空闲处理机的名字,同时在空闲处理机上启动一个新的顾客进程,P将子任务送给新的顾客进程.如果未找到,中央进程则向P回送忙碌信号,子任务被送往任务队列.同一级的FORK全部处理完后,P继续作为顾客进程执行自己保留的并行任务中的最后一个.

服务进程得到结果后,进程结束.如果是0号任务,则结果送给用户;否则,结果送给中央进程.中央进程修改顾客进程表并检查相应顾客进程的并行子任务结果是否到齐.如果到齐且此顾客进程又在阻塞表中,中央进程则向其发送唤醒信息,将其激活,同时删除顾客进程表和进程阻塞表中的对应项.

顾客进程执行完自己保留的并行任务后,将遇到JOIN原语.此时,顾客进程向中央进程发送结果请求信息.如果同级并行任务的结果全部得出,则顾客进程通过JOIN,继续对后续任务进行预编译;否则,进入阻塞状态,直到中央进程将其激活.每个顾客进程最终都会变成服务进程而得到结果.

3 实例测试

我们使用了5个例子来测试我们所实现的并行计算支撑环境PCSE的性能.实验环境是10Mbps以太网连接的8台SUN 4/20工作站.初始状态为:每个处理机上运行1个调度员,主机上还运行1个主进程和1个中央进程.主进程在等待用户任务.

前两个例子是用PROLOG语言编写的斐波纳契数求解程序与梵塔问题处理程序.这两个例子都包含递归

关系且并行粒度极高，所以系统性能良好. 第3和第4个例子是用FORTRAN 77编写的重磁资料转换程序GMPS和Vlasov方程求解程序VLAS. 这两个程序来自实际应用. GMPS利用位场频率域处理原理实现重磁异常处理. 整个处理过程包括原始数据的扩充、二维正傅立叶变换、频率域位场异常转换处理和二维反傅立叶变换. 其中二维正、反傅立叶变换都可进行并行化处理，频率域位场转换处理也可部分并行. VLAS使用蒙特卡洛试验粒子方法求解Vlasov方程. 程序的核心部分是密度计算，这部分也是最费时的. 我们将该部分划分为多个独立的子部分，并在PCSE上进行了并行计算. 最后一个例子是用C语言编写的矩阵乘计算程序MULT，矩阵规模为500 * 500. 选择此例的目的是观察系统开销对系统性能的影响. 采用并行计算后，本系统引入了一些额外的开销. 其中最大的开销是在各处理机上启动顾客进程所需的时间，该时间值大约为1.2s. 其次是预编译时间，该时间随程序长度的不同而不同，在以上3个例子中，最长的是GMPS的0.8s. 此外，子任务、结果、控制信号在网络上的传递以及结果的重定向等操作也需花费一定的时间，不过，这些时间并不长，在某些情况下可忽略不计. 由于MULT的执行时间短，额外开销占的比例较大，因此，系统性能较差. 表1给出了实验结果. 其中，加速比被定义为单机运行时间与多机运行时间的比值.

表1　不同系统规模下各测试程序的加速比

加速比		处理机个数							
		1	2	3	4	5	6	7	8
测试程序	FIB	1	1.75	2.6	3.58	4.44	5.32	6.19	6.96
	HANOI	1	1.84	2.77	3.65	4.52	5.43	6.22	7.13
	GMPS	1	1.42	2.15	2.92	3.54	4.23	4.96	5.54
	VLAS	1	1.53	2.37	3.26	3.89	4.67	5.33	5.94
	MULT	1	1.28	1.65	1.77	1.82	1.89	1.94	1.97

4　结　论

实验结果表明，我们所实现的并行计算支撑系统是可行的. 在没有增加用户太大负担的情况下，使用户的应用程序在并行计算中赢得了速度. 但系统仍有一些不足之处. 预编译部分还需优化，以缩短其开销. FORK和JOIN语义还需扩大，以使更多种类的程序(如PASCAL，LISP等)能够在PCSE上运行.

参考文献

1　Carlton M, Roy R V. A distributed Prolog system with and parallelism. IEEE Software, 1988,5(1):43～51

2　Tao Jie, Ju Jiu-bin. Executing Prolog programs in parallel on network workstation. Chinese Journal of Advanced Software Research, 1994,1(1):30～38

3　Tian Lai-sheng, Huang Lian-shu, Tao Jie. The implementation of DC in a loosely-coupled distributed environment. Chinese Journal of Advanced Software Research, 1995,2(3):227～236

Design and Implementation of a Parallel Computation Supporting Environment

TAO Jie　JU Jiu-bin

(*Department of Computer Science　Jilin University　Changchun*　130023)

Abstract　A parallel computation supporting environment named PCSE(parallel computation supporting environment) is described in this paper. By introducing from two process operation primitives——FORK and JOIN, any program whether written in procedure programming language or logic programming language can be executed in parallel, PCSE provides an excellent software interface for the users to carry out on high-speed calculation. At present, PCSE supports FORTRAN, C and PROLOG applications executed in parallel.

Key words　Parallel computing, distributed programming, networked workstations.