Sidle的远程执行设备*

徐高潮 鞠九滨

(吉林大学计算机系,长春 130023)

摘要 Sidle系统是运行在SUN工作站网络上的一组实用程序,利用空闲的处理机资源进行大粒度的并行计算.同其它远程执行设备相比,它能支持程序内部并行和嵌套的远程执行,允许一个服务员机接受多个远程执行任务.本文介绍了这些特点和透明性的实现方法.

关键词 工作站网络,空闲工作站,大粒度并行,远程执行的透明性,程序内部并行,嵌套的远程执行.

我们的分布式系统由19台SUN4工作站经以太网连接而成,其中9台SUN4/65,10台SUN4/20,总的处理能力在2.5亿次/秒以上.Sidle系统是运行在此环境上的一组实用程序,支持用户利用网络上的空闲的处理机资源实现大粒度的并行计算.

目前国际上已发表的同类工作中比较著名的有:Stanford大学研制的V系统的可抢先的远程执行设备[1],Xerox PARC研制的Cedar系统的进程服务员[2],贝尔实验室研制的NEST系统中的负载共享与透明的远程执行设备[3],Carnegie-Mellon大学研制的Andrew分布式系统的Butler系统[4],Wisconsin大学研制的Locox网络中的Condor调度系统[5],加拿大Victoria大学研制的REM远程执行系统[6],以及Ohio州立大学研制的Stealth分布调度程序[7].

同以上这些系统相比,Sidle具有一些新功能.Sidle系统的远程执行设备不仅能支持命令一级的任务并行而且还能支持PROLOG程序的内部并行.有些程序的分布并行执行要求远程执行设备能够支持嵌套的并行执行,例如并行make和C-PROLOG的分布并行执行.由于以上系统都不支持嵌套的远程执行,不可避免地限制了系统的应用范围.现代工作站的处理能力都很强,一个服务员机在接受一个外来任务之后往往还有空闲时间.到目前为止所发表的系统还没有实现在单个服务员机上执行多个外来任务的功能,只有Stealth有这种思想,当前还在实现中.

空闲工作站的共享和其它资源共享有着不同的要求和特点.在文件透明访问的问题上,V系统采用文件远程拷贝的办法来实现,Butler系统使用共享文件系统来实现,Cedar系统使用自己研制的CFS(Cedar File System)来完成文件的远程访问,其主要策略是在本地缓存远程文件.由于NFS(网络文件系统)具有很好的透明性和坚定性,所以Sidle使用NFS

* 本文1992-04-11收到,1992-07-24定稿

作者徐高潮,28岁,助教,主要研究领域为分布计算系统.鞠九滨,59岁,教授,主要研究领域为分布式计算系统.
本文通讯联系人:徐高潮,长春 130023,吉林大学计算机系

来实现文件的透明远程访问.

V 系统和 NEST 系统都提出了逻辑主机的概念.所谓逻辑主机就是属于一个物理主机的进程(包括本地的和本地所派生的远程进程)的集合.V 系统和 NEST 系统使用网络内进程的全局唯一命名来保持远程执行进程原有的父子关系和进程组关系,这种方法需要修改原有的操作系统.Sidle 系统在远程执行设备中定义了逻辑父子关系和逻辑组关系,在不修改原有操作系统的前提下保持了远程执行进程原有的父子关系和组关系,并实现了物理主机间信号的透明传送.

1　程序内部的并行执行

Sidle 系统的支持程序内部并行的远程执行设备由以下几个部分组成:远程执行服务员(RES—Remote Execution Server),程序内部并行执行的接口程序—通信服务员 CS.通信服务员 CS 是并行程序同远程执行服务员之间的接口.图 1 说明了支持程序内部并行的远程执行设备的组成.

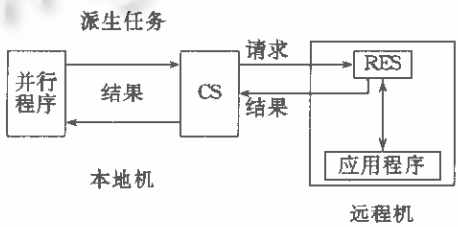

图1　支持程序内部并行的远程执行设备的组成

通信服务员 CS 将并行程序的远程执行申请转发给远程机的远程执行服务员,后者派生一子进程执行并行程序所派生的子任务,远程子任务的执行结果送回基地,结果由通信服务员 CS 保存,并行程序需要子任务的执行结果时,从通信服务员 CS 处获得.

通信服务员 CS 中设置了一个子任务表,用于记录本地所派生的子任务,其数据结构表示为:

```
struct subtask—list {
    struct    subtask—list *next;
    char      name[16]; /* 子任务名 */
    int       sysid;
    int       taskid;
    int       flag;
    char      *buffer;}
```

子任务标识符使用网络内全局唯一命名(sysid,taskid),flag 是子任务是否执行完毕的标志(0 表示未完成,1 表示完成),buffer 是执行结果的存放地址.

基于上述思想,我们将串行 C—PROLOG 解释程序改造成并行 DC—PROLOG 解释程序,实现了 PROLOG 应用程序的分布并行执行.

2 远程执行的透明性

2.1 文件的远程透明访问

远程进程所访问的文件系统应该是基地节点上的文件系统. 例如,远程机和基地机上都有文件"/a/b",那么远程执行的进程所访问的应该是基地上的文件"/a/b",而不是服务员机上的文件"/a/b". Sidle 的远程执行设备在每个工作站上都设置了一个远程执行目录"/rem",如果进程 P 的基地节点为 eagle,远程执行服务员为 P 建立一个根目录"/rem/eagle",利用 NFS 将进程 P 所要访问的远程文件安装在目录"/rem/eagle"下. 而本地进程的根目录则为"/". 从图 2 可以看出,本地进程和远程进程具有不同的根.

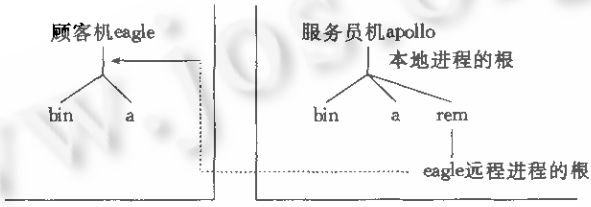

图2 远程进程和本地进程的文件系统

2.2 进程间关系保持和物理主机间信号传送

UNIX 系统的 signal 传送机制是实现进程通信与同步的,为了维持逻辑主机内进程的通信与同步,必须实现物理主机间信号的透明传送. Sidle 的远程执行设备在逻辑主机内定义了两个关系,即逻辑父子关系和逻辑组关系. 为了同 UNIX 本身所保持的进程关系相区别,称 UNIX 所保持的进程关系为实际父子关系和实际组关系.

图 3 中的 P 和 Q 两进程的关系是逻辑父子关系,Q 是 P 所派生的远程子进程.

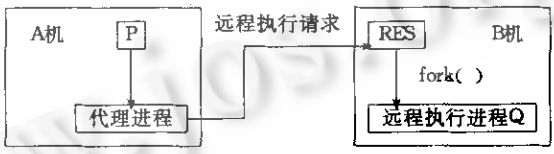

图3 进程的逻辑父子关系

逻辑主机边界内的所有进程称为一个逻辑组. 从图 4 可以看出,一个逻辑组可以包含多个实际组. 实际组 A、实际组 B 和实际组 C 同属一个逻辑组.

物理主机间信号的传送是通过信件传送来完成的,代理进程用于父子间信号的传送与接收,组代理进程用于逻辑组内信号的传送与接收. NEST 系统修改了 proc 结构,进程的命名采用全局唯一命名,进程标识符包括两部份〈sysid,pid〉,专门设置了一个网络服务员用于信号在不同物理主机间的传送.

2.3 远程控制终端

当一个程序在远程执行时,它的控制终端应该是其基地节点上的终端. Sidle 的远程控制终端是利用管道技术,输入输出重定向技术和 Socket 进程通信工具实现的.

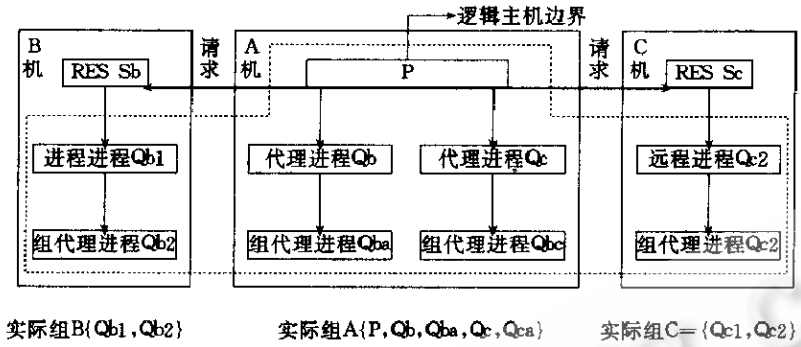

图4 进程的逻辑组关系

3 远程进程管理

Sidle 系统的远程执行设备能接受多个远程执行请求并派生多个远程执行进程在某服务员机上同时执行,设置远程进程表来管理多个远程进程是比较直接的办法,表的数据结构可表示为:

```
struct remote—proc {
    struct    remote—proc  * next;
    int       pid; / * 进程号 * /
    char      home—host[16]; / * 基地主机名 * /
    char      file—server[16]; / * 文件服务员名 * /
    int       apid; / * 用户接口进程号 * /
    int       agpid; / * 组代理进程号 * /
    int       euid; / * 有效用户号 * /
    int       egrpid; / * 有效用户组号 * /
    int       socket—r; / * 接收信件的通信口 * /
    int       socket—s; / * 发送信件的通信口 * /
    int       pipin[2]; / * 输入管道 * /
    int       pipout[2]; / * 输出管道 * /
    int       piperr[2]; / * 差错管道 * / }
```

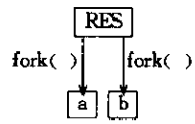

图5

每个进程都有一个有效用户标识符和有效组用户标识符来记录一个进程的访问权限.例如图 5,用户 A 和用户 B 向同一空闲机分别派生了远程执行任务 a 和 b,如果不改变 a 和 b 的有效用户号和有效用户组号,那么 a 和 b 所继承的有效用户号和有效用户组号和远程执行服务员 RES 的有效用户号和有效用户组号相同.实际上,a 进程的有效用户号和有效用户组号应该是用户 A 的用户号和用户组号;同样,b 的有效用户号和有效用户组号应该是用户 B 的用户号和用户组号.RES 派生完进程 a 和 b 后应该改变它们的有效用户号和有效用户组号,使其具有正确的访问权限.

多个远程进程在同一个服务员机上执行时,还应该处理好它们的进程组关系.例如图 5

中的a和b,它们应该属于不同的逻辑组,但是,由于a和b都是同一程序RES所派生的,所以a和b又属于同一实际组,这同我们所定义的逻辑组关系相矛盾.为解决这一矛盾,RES在派生完进程a和b后,必须给它们赋以不同的进程组号,使得它们不属于同一实际进程组.

此外,多个远程进程需要访问不同的文件系统,因此多个远程执行的进程还应该具有不同的根目录和当前工作目录.

在远程执行设备只为一个远程进程服务的情况下,远程执行设备只需同一个顾客机建立通信通道.Sidle的远程执行设备可能会和多个顾客机建立通信通道,所以Sidle的远程执行设备的通信通道必须是可扩展的.远程执行服务员接受一个外来任务时便为其建立一个通信进程,负责同基地节点建立通信通道,当远程执行任务完成时便撤消此通信进程.

4 嵌套的远程执行

一个任务P派生出子任务Q到远程机上执行,子任务Q又派生出自己的子任务R到另一远程机上执行,我们称这种情况为嵌套的远程执行.在嵌套的远程执行中,任务P同其所派生的多个后代任务构成一树形结构,例如图6.从图6中我们可以看出,一子任务所在的节点可能会和以下几类节点相联系:文件服务员节点、根节点、父节点和儿子节点.如果根节点是无盘工作站,那么文件服务员节点和根节点就不属于同一节点,否则就是同一节点.

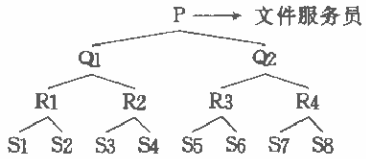

图6 嵌套的子任务派生

子任务同文件服务员的联系主要是使得子任务能访问文件服务员上的文件系统.由于NFS不允许文件的嵌套远程访问,因为嵌套的远程访问会大大地降低NFS的效率,所以子任务必须直接访问文件服务员上的文件系统而不能通过其父节点及祖先节点层层嵌套访问文件服务员.

子任务同根节点也应保持联系,子任务的显示信息(包括标准输出信息和标准差错信息)应该直接送到根节点,如果层层嵌套传送会增加通信负担和传输环节,信息丢失也不易检测.直接将子任务的显示信息传送到根节点并在根节点的终端上显示出来可以避免这些不利因素,在实现上也比较简单.

子任务同父节点的关系.在嵌套的远程执行中,父进程往往需要用到子进程的执行结果,子任务所用到的控制信息从父进程那儿取得.这可由远程执行设备中的通信服务员CS来负责这一通信关系.

在嵌套的远程执行中,进程P同其所派生的所有后代进程都属于同一逻辑组,组信号的传送应该使用广播形式,如果使用点到点的信件传送需要大量的信件和较长时间的延迟.

从以上分析可以看出,嵌套的执行环境涉及多个节点,建立一个嵌套的远程执行环境要

比建立一般的远程执行环境要复杂得多.

程序迁移对嵌套的远程执行会带来较大的影响. 如果 P 是一远程进程,迁移 P 必将影响 P 的所有后代进程的执行. 要迁移程序 P,首先停止 P 的所有后代进程的执行,然后再迁移 P.

4　结束语

DC－PROLOG 的并行机构对用户来说是完全透明的,其使用和单机上使用 PROLOG 的方法完全一样. 使用 8 台工作站求解斐波纳契数,加速率可达 6.7. 使用 9 台工作站求解梵塔问题,加速率可达 7.6. 对于 PROLOG 来说,一个应用程序经常产生多个远程子任务,一个远程子任务又经常产生多个远程子任务,所以驱逐一个 PROLOG 程序会引起多个子任务运行中止,这必将影响 PROLOG 的执行效率. 所以我们规定 PROLOG 应用程序一旦在某远程机上运行,远程机不能驱逐此 PROLOG 应用进程,直到它运行结束为止.

Sidle 为远程执行一任务所花开销主要包括以下几个部分:建立和删除远程执行环境,安装和撤消远程目录,启动和加载远程执行程序等. 其中远程加载 100KB 程序所用时间为 175ms. 实际测量远程执行最小程序的时间为 1.5 秒,其中 CPU 时间不大于 0.5 秒.

有如下几个方面的进一步工作. 第一,系统中进程的通信手段使用 Socket 进程通信机制,如果使用 SUN OS 的轻量进程编程和 RPC 进程通信工具可大大改进系统进程通信和进程调度的性能. 第二,开发其它语言的并行执行,例如 C 语言,并提供与 Sidle 的相应接口. 第三,扩大应用范围,解决有关应用的用户接口,以及作业调度算法,着重在人工智能方面的应用以及在调度算法中引入人工智能机制.

参考文献

1　Theimer M M, Lantz K A, Cheriton. Preemptable remote execution facilities for V－system. In: Proceedings of 10th Symposium on OS ACM, 1985:2－12.

2　Hagmann R. Process server: sharing processing power in a workstation environment. Proc. 6th International Conference on Distributed Computing System, 1985:260－267.

3　Agrawal R, Ezzat A K. Location independent remote execution in NEST. IEEE Translation on Software Engineering, 1987, **SE－13(8)**:905－912.

4　Nichols D A. Using idle workstations in a shared computing environment. In: 11th ACM Symposium on OS Principles, 1987:5－12.

5　Litzkow M T. Condor －a hunter of idle workstations. In: Proc. 8th International Conference on Distributed Computing Systems, 1988:104－111.

6　Shoja G C. A distributed facility for load sharing and parallel processing among workstations. Journal of System & Software, 1991, **14(3)**:163－172.

7　Krueger P, Chawla R. The stealth distributed scheduler. Proceedings of 11th ICDCS, IEEE, 1991:336－343.

REMOTE EXECUTION FACILITIES OF SIDLE

Xu Gaochao and Ju Jiubin

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

Abstract The sidle system is a set of programs running on network—based workstations that allows users to do their large—grained parallel computations using idle workstations. Compared with other systems, it supports parallel computation within a program and recursive remote execution, allows a server machine to accept more than one remote job. This paper presents these features and transparent implementation of remote execution.

Key words Workstation networks, idle workstations, large grain parallel processing, transparency of remote execution, parallel computation within a program, recursive remote execution.