对 Condor 系统的分析与改进*

郭 雷 鞠九滨 张怡颖

(吉林大学计算机科学系 长春 130023)

摘要 本文分析 Condor 系统的控制软件、远程系统调用及检查点设备的实现和工作过程,讨论其任务局限性,指出实现中的不足并给出改进方案.

关键词 进程迁移,负载平衡,检查点设备.

中图分类号 TP393,TP316

Condor 系统[1~3]是 Wisconsin 大学研制的一个异构型分布式系统. 它利用将大计算量任务派至远程空闲机执行的方式,把空闲工作站的 CPU 资源分配给其他用户. Condor 的特点在于它成功地利用检查点设备实现了进程迁移以达到负载平衡和容错的目的. 一些著名的分布计算系统如 DQS, LoadLeveler[4], Load Balancer 和 PBS 等正在进行对 Condor 的开发工作,以支持它们的检查点设备和进程迁移的功能.[5]

Condor 系统监视局域网中工作站的负载情况,将处于空闲状态的机器加入动态资源缓冲区中.[1]需要 CPU 资源的用户可以申请获得空闲机器. 它具有以下特点[2]:

(1)使用 Condor 时,不需要进行特殊的程序设计.

(2)当机器 B 上的任务被派至机器 C 上远程执行时,此任务使用的是机器 B 上的执行环境. Condor 保证这个任务不会接触到机器 C 的文件系统,这样可保证机器 C 的系统安全.

(3)Condor 可以自动寻找并分配空闲工作站.

(4)工作站的主人对自己的工作站有绝对控制权.

(5)Condor 完全在操作系统核外实现.

Condor 的功能是很强的. 如果在工作站网络中适当地使用 Condor,工作站的空闲处理能力将会在保证系统安全的前提下以更高的效率得到发挥,提高系统资源利用率,缩短作业响应时间,用户也不会感到不便.[3]

由于现在 Condor 是一个免费软件,使我们有机会获得它的源程序(约七万条语句),并对它进行分析. 我们希望通过对源程序的分析,了解其系统内部机制及工作原理,发现其不

* 本文研究得到国家 863 高科技项目基金资助. 作者郭雷,1972 年生,助教,主要研究领域为计算机网络与分布式系统. 鞠九滨,1935 年生,教授,博士导师,主要研究领域为分布系统与计算机网络. 张怡颖,1971 年生,博士生,主要研究领域为分布式人工智能及语音识别.

本文通讯联系人:郭雷,北京 100088,北京邮电大学国家重点实验室

本文 1996-05-09 收到修改稿

足并加以改进. 本文研究环境为以太网连接的多台SUN4工作站.

1 Condor的控制软件

我们把在一个局域网中运行Condor的所有机器的集合称为Condor池.

Condor的任务调度结构介于静态集中式与全分散式之间. 在一个Condor池中,其中一台机器被指定为中央服务员,它负责收集池内各机器负载及任务队列信息,并仲裁空闲机的分配. 所有机器均运行任务管理进程Schedd和Startd. Schedd负责维护任务队列及更新中央服务员保存的任务队列信息,当本地节点有未完成任务时,需与中央服务员协商以获得空闲机资源. Startd周期性地检查本地节点负载状态并更新中央服务员的负载信息表. 当本地节点被指定为外来任务服务时,Startd还负责启动并管理该任务. 在运行X-window的机器上,守护进程Kbdd负责检查键盘和鼠标的状态并将结果送给Startd.

下面我们利用一个例子来说明Condor是如何调度任务的,见图1.

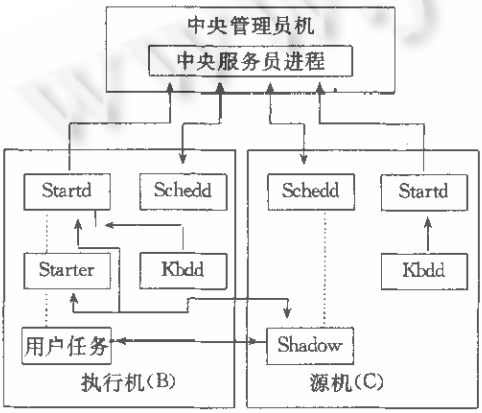

图1 Condor控制软件组成

在某一时刻,中央服务员发现机器B是空闲的,而在机器C的任务队列中有任务等待执行且B符合C机任务要求. 首先,中央服务员与C的Schedd联系,允许它在B上运行一个任务. C的Schedd从其队列中选出一个任务并启动一个Shadow进程. Shadow将通知B的Startd:机器C将要在B上运行一个任务. 如果自从最近一次对中央服务员的负载信息表更新以来,B的状态没有发生改变,即B仍是空闲的,那么它将回答"OK". 此后B上的Startd会启动一个Starter进程. C的Shadow进程向B的Starter进程传送任务的检查点文件后,Starter进程将设置定时器并启动传送来的任务. 定时器用来规定产生检查点的时间间隔.

当机器B的Starter进程定时结束时,它将向用户任务发送一个SIGTSTP信号,中断用户任务的执行. 由Starter进程负责建立在这个检查点的检查点文件(暂时存放在机器B上)并设置定时器重新启动任务执行. 反复至任务结束,通知任务的提交者.

如果在任务的运行过程中,B由于主人的使用变得繁忙,Startd进程将向Starter进程发送"挂起"信号,暂时挂起用户任务. 这是由于通常机器上的用户只在一段时间内比较活跃,例如用户查看信件. 若机器B的繁忙情况持续了很长时间(如5m),Startd进程将向Starter进程发送"VACATE"信号. Starter进程终止C机上用户任务的运行,并将最新的检查点文件返回给C上的Shadow进程. 如果此任务在B上的运行时间很短,还没有达到一个检查点,那么将只是终止任务的运行. B上的Starter进程周期性地对用户任务的检查点文件进行更新,而不是在发送"VACATE"信号后更新. 这是为了避免当工作站上的用户十分活跃时,进行更新检查点文件这样的大规模I/O操作. 虽然周期性的更新会造成效率降低,而且会损失一段时间的计算结果,但B的主人还是很高兴的.

2　Condor 远程系统调用的实现

每一个 UNIX 用户的程序,无论是否用 C 语言编写的,最后都需要与函数库 libc.a 进行链接.这个函数库中包括基本 I/O 函数和与系统核心接口函数.系统核心接口函数一般实现为"stub"形式,调用时将参数和标识功能的调用号压入用户栈中,然后执行机器定义的超级用户调用(supervisor call)指令.UNIX 系统中的系统调用都对应着 libc.a 中的一个函数.图 2 说明了 UNIX 系统的系统调用机制.

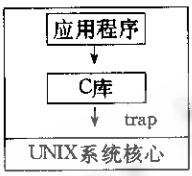

图2　UNIX的系统调用机制

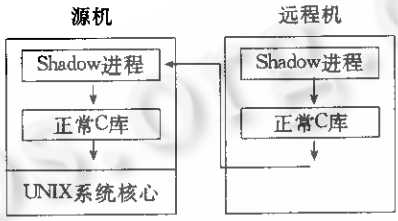

图3　远程系统调用实现机制

Condor 提供的函数库 libcondor.a 替换了 libc.a 中所有核心接口函数.提交给 Condor 的任务在编译时必须与此库链接.当一个任务被派至远程执行时,源机上运行着相应的 Shadow 进程.Shadow 是任务进行远程系统调用的代理.远程系统调用时,新的接口函数将把参数和标识功能的调用号打包发送给对应的 Shadow 进程.Shadow 会以正常的调用方式执行这个系统调用,然后把结果返回给远程运行的任务.这个方案使远程过程调用具有高度透明性.图 3 说明了远程系统调用的实现机制.

3　Condor 的检查点机制

Condor 通过向正在运行的任务发送信号来设定检查点.任务进程接收到信号 SIGTSTP 后,利用其信号处理程序保存当前打开文件状态并生成系统核心映像文件 Core.Condor 构造检查点文件时利用了上一次检查点文件中的代码段及符号表和本次检查点生成的 Core 文件中的栈段、数据段、当前打开文件状态表及寄存器内容.如图 4 所示.

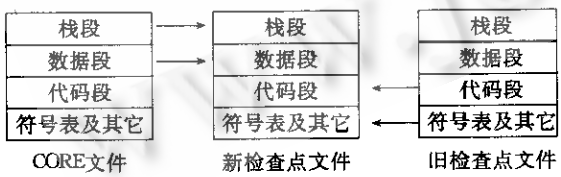

图4　构造新检查点文件

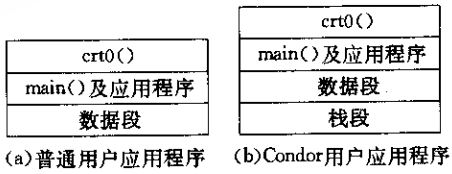

图5

Condor 保存进程状态信息的很多操作都是在信号 SIGTSTP 的处理程序中完成的.但是,在 UNIX 系统中,当用户任务进程接收到信号 SIGTSTP 时将终止该进程的运行.这样在用户的应用程序中就需要调用 signal()函数来修改信号 SIGTSTP 的处理程序指针,使它指向检查点所提供的信号处理程序.为了避免用户使用检查点时需要特殊的程序设计,Condor 修改了程序起始模块(program prologue module)——crt0().它将原 UNIX 系统的 crt0.o 文件中调用用户程序主函数 main()的部分改写为调用函数 MAIN().MAIN()函数

是由 Condor 提供的，其主要功能就是对检查点设备所使用的全局变量及信号处理程序进行初始化，在函数 crt0()与 main()之间起到桥梁的作用.

任务在经过检查点后所获得的检查点文件符合 UNIX 系统的可执行文件格式，只是增加了栈段(如图 5 所示)，完全可以直接再次运行.

4 Condor 任务的局限性

检查点设备和远程系统调用的使用，使 Condor 所支持的任务受到了很多的限制：

① Condor 仅支持单进程的任务. 在 Condor 中，不支持使用 fork 及类似系统调用的任务. 使用这些系统调用的任务经常在父子进程间进行通信，协调进行工作. 而 Condor 无法为多个进程生成检查点文件.

② Condor 不支持使用信号和信号处理程序的应用程序. 由于在 Condor 的检查点设备的实现中使用了许多信号处理程序，如果用户应用程序中使用同样的信号，那么由用户定义的信号处理程序将取代检查点设备定义的信号处理程序，造成检查点设备失效.

③ Condor 不支持用户应用程序使用进程间通信. 如果用户应用程序使用了 SOCKET 机制进行进程间通信，就有可能出现在某个检查点时刻，用户应用程序正在进行进程间通信的情况，这时将会因造成通信的混乱而失败.

④ 用户应用程序使用的每个文件均应是只读的或是只写的. 那些使用可读可写文件的任务有可能会造成错误. 这是 Condor 定时生成检查点文件造成的.

5 对 Condor 实现的几点改进

Condor 对于任务的局限性是由于系统设计造成的，改进难度相当大，不在本文中讨论. 本文仅对 Condor 实现过程中的一些不足设计了改进方案.

5.1 源机在任务已派出而结果尚未返回时关机，派出的任务将成为“孤儿”

这个问题的产生原因在于当源机正常关机时，任务对应的 Shadow 进程将被终止. Shadow 进程接收到 SIGTERM 信号时，只对任务队列作了保护工作，并未通知在远程执行的相应任务进程. 当这个任务不使用系统调用，不需要与 Shadow 联系时，只有在运行结束或需要进行迁移时才会发现无法与源机联系，所做的计算工作都将报废，造成巨大的 CPU 资源的浪费.

我们可以通过在 Shadow 中加入信号 SIGTERM 的处理程序，来解决上面的问题. Shadow 接收到 SIGTERM 信号后：

① 由 Shadow 进程向远程机上任务相应的 Starter 进程发送信息包.

② 由任务相应的 Starter 进程向任务进程发送信号 SIGTSTP 生成检查点文件.

③ 由 Starter 进程向相应的 Shadow 进程传送检查点文件.

④ 由 Shadow 进程对任务队列进行维护.

上面的方案在关机时终止进程会花费较长时间. 但是这只是在源机上花费了时间，源机上提交任务的用户是应该能够容忍以一定的时间来换取任务的提前完成的. 此方案仅适用于使用 HALT 命令正常关机的情况，这时会产生关机信号，为 SIGTERM 处理程序所截获.

5.2 中央服务员机制的改进

虽然Condor中央服务员的功能十分有限,其它机器对服务员的依赖性不是很强,但它使系统的可靠性很差.如果中央服务员机发生了故障,则需要对Condor池内所有机器的配置文件进行修改后,才能够恢复正常运行.这将使系统的效率明显降低.

本文设计的可迁移式服务员可以通过"竞选"产生.当一台属于Condor池的机器启动时,它首先发广播寻找服务员机.如果当前还没有Condor池中其它机器启动,那么它就定义自己为服务员机并启动中央服务员进程.当服务员机需要关机时,它发送一个广播给所有正在运行的Condor池中的机器.其它机器接收到广播后,向服务员发送应答信息,服务员将第一个作出响应的机器指定为新的服务员.

这个方案会增加一定的开销,但大大增加了系统的可靠性.

6 结　论

本文对Condor系统的控制软件、远程系统调用及检查点设备的实现和工作过程进行了分析,并讨论了其任务局限性的起因.针对Condor实现中的不足,提出了改进方案.实现这些方案并进行性能比较是我们正在进行的工作.

参考文献

1 Litzkow Michael, Livny Miron, Mutka Matt W. CONDOR—a hunter of idle workstations. Proceedings of 8th ICDCS, 1988. 104～111.

2 Bricker Allan, Litzkow Michael, Livny Miron. Condor Technical Summary Version 4.1b, 1992.

3 Litzkow Mike, Livny Miron. Experience with the CONDOR distributed batch system. 1992.

4 Kaplan J A, Nelson M L. A comparison of queuing cluster and distributed computing systems. Technical Memorandum of NASA Langley Research Center, June. 1994.

5 Dikken L, Linden F, Vesseur J *et al*. DynamicPVM. International Conference on High Performance Computing and Networking, Apr. 1994.

ANALYZING AND IMPROVING THE CONDOR

GUO Lei　JU Jiubin　ZHANG Yiying

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

Abstract　In this paper the principles and implementations of the control software, remote system call and checkpoint facility in the Condor are described. Limitations of tasks in the Condor are discussed. The shortcomings and suggestions in implementing are pointed out.

Key words　Process migration, load balance, checkpoint facility.

Class number　TP393, TP316