分布式计算系统中进程迁移的方法*

肖 红 邱毓兰 彭德纯

(武汉大学计算机科学系,武汉 430072)

摘要 在分布式计算系统中,进程迁移是一种在进程生命周期内将它从一台处理机传送到另一台处理机上,并使进程从其"断点"继续运行下去的方法.本文详细地讨论了在Wulor—75/32系统和国外几个典型的分布式计算系统中实现进程迁移的方法,以及为实现进程迁移,对系统的其它机制——文件系统、通信机制和内核中有关部分所作的扩充.

关键词 分布式系统,进程迁移,寻找空闲机,进程通信机制,性能优化.

"进程迁移"是一种在松耦合的分布式系统中,将正在某处理机(源处理机)上运行的进程传送到另外的处理机(目的处理机)上,并使进程在目的机上从"断点"起继续运行的方法.

提高分布式系统整体性能的一条途径是将系统负载尽可能均匀地分布到系统中各个处理机上,也即实现负载均衡,或负载共享,这样能提高系统的并行度,加快运行速度.而通过对处理机作业的静态分配很难达到负载及资源的平衡,因为当一个进程突然申请大量的某种资源时,或者生成一个带有未意料到的资源需求的新进程时,都可能破坏曾经达到的平衡.如果能动态地访问系统负载,并能在进程的生命周期内重新合理分布进程,那么尽管移动进程需要一定的开销,一般情况下,系统吞吐量仍可获得较大的提高.理论和模型研究表明:通过进程的重定位可获得良好的系统性能[9-12].

进程迁移机制还可用于故障恢复.进程迁移提供了透明地中止一个进程、传送其环境并在另外的机器上恢复其运行的能力.如果迁移一个进程所必需的信息存储在静态存储器中,处理机故障时,我们就可能在它完全瘫痪前,将其上运行的进程迁移到其他仍能工作的处理机上.当然,这种故障应是逐渐的,或由部分软件引起的,且能够为迁移进程提供必要的时间.

"进程迁移"是目前国际上非常活跃的研究领域,很多分布式系统将它作为一个设计目标,但成功的例子并不很多.较著名的系统有:Sprite[4,6]、DEMOS/MP[5]、Condor[3]、LOCUS[8]、V—System[7]等.在国内,我们为研制"Wulor—75/32负载共享远程执行系统"(国家"七五"项目[1,2])首次成功地实现了"进程迁移".

* 本文1991年4月18日收到,1991年9月4日定稿

作者肖红,硕士,主要研究领域为计算机软件,现在深圳赛格软件技术有限公司工作.邱毓兰,副教授,主要研究领域为计算机系统软件、分布式并行处理操作系统.彭德纯,教授,主要研究领域为计算机系统结构,分布式并行处理(计算)等研究工作.

本文通讯联系人:彭德纯,武汉 430072,武汉大学计算机系

进程迁移的实现是一件很困难的工作.其中一个主要问题是如何将进程从其老的工作环境中分离出来,又如何在目的机上为进程的继续运行创造必要的工作环境.其次,涉及到另一活跃的研究领域——寻找空闲处理机的方法.另外,还必需扩充文件系统和通讯机构,使进程迁移后仍能访问所有资源,并正确与其他进程通讯.本文第 1—3 节将分别讨论这些问题,且着重描述了 Wulor－75/32、Sprite、Conder 和 DEMOS/MP 系统的解决方法,第 4 节则讨论了各系统中为降低迁移开销所提出的优化方法,最后提出一些结论.

1 迁移进程的方法

进程迁移机制中最困难的工作无疑是进程运行环境的保护及恢复.因为这涉及到对操作系统内核的修改扩充.

进程运行的每一步都是与一定的环境相联系的,包括处理机状态、进程状态、通用寄存器中的数据、虚地址空间(正文段、数据段和栈段)、各打开文件的状态、所收到的信号、家族联系及资源占用等多项内容,这些环境都是由操作系统内核来管理的.进程迁移时,进程在源处理机上停止运行,如要使它能在目的机上继续运行,就必须将目的机上的环境恢复成与该进程停止时完全相同的状况.现有的多数操作系统都没有这种能力,因此需要在内核中扩充对环境的保护和恢复功能,这需要进行大量的工作.如果不改动内核,而象在 Bulter 系统[14]中那样,在源处理机仅仅简单地"杀死"进程,然后在目的机上重新启动,那么进程在源处理机上所做的工作就全浪费了,特别是进程可能已经运行了相当长的时间,这种浪费就更大,这样,无疑会降低系统性能.所以对内核的扩充工作是值得进行的.

不同系统迁移进程时,基本上都需要如图所示的几步:

源处理机	目地处理机
中断被迁进程的运行 保护进程中止时的环境 (传送进程的环境)	
	(传送进程的环境) 申请资源,恢复环境 恢复进程运行

不同系统中,进程运行的环境有所不同,但基本上都由进程控制块、核心栈、通用寄存器中的数据、收到的信号和打开的文件等组成.

Sprite 系统是使用标准信号机构来中断被迁移进程的运行.Wulor－75/32 和 DEMOS/MP 则是陷入核心;将被迁进程的状态变为"迁移态"——不被调度也不被换出,并将它从运行队列中删去.

Wulor－75/32 系统是在 Sun－3/60、260、280 工作站连成的 Ethernet 网络上实现的[1].网上工作站运行 Sun OS 3.5,我们分析了其内核的有关内容,发现 Sun OS 的性能与实现方法都类似于 UNIX BSD 4.2,也是分时系统.进程切换时,核心将现运行进程的一些环境,包括通用寄存器的内容、核心栈及收到的信号保护到 U 区中,当进程再次被调度运行时,又将这些环境从 U 区中恢复.利用这套机制能简化环境的保护和恢复.于是我们在每

个处理机上设置一个"顾客进程"和一个"服务器进程",顾客进程负责调度作业并迁移外来进程,而服务器进程则负责执行顾客和其它服务器所要求的服务.让核心从顾客进程陷入,这时要迁移的进程已被切换,有关环境已被保护到U区中;而在目的机上,只需将这些环境恢复到新创建进程的U区中,当新进程被调度运行时,这些环境就自动恢复了,这样对寄存器、核心栈和信号等环境的保护与恢复代之以对U区的保护与恢复.使用这种方法,迁移进程的响应时间取决于顾客进程被调度运行的时间.为缩短此响应时间,我们将外来进程设置为后台进程,而顾客进程设置为前台进程,根据UNIX的切换算法,顾客进程一旦启动,就能立即抢占运行.

在Condor系统中,每台处理机上有一个本地调度器(local scheduler),其中一台上有一个协调器(Coordinator).在有外来进程的处理机上的本地调度器每隔1/2秒检查本地用户是否恢复工作,若是,则立即停止外来进程的执行,并将进程的"检查点"(checkpointing)保存在存储器中."检查点"也就是进程恢复运行所必需的环境,它并不立即传送,而是等到协调器为它分配了一个空闲机,且本地调度器又决定将此进程迁移时再传送.

环境的传送方法依赖于系统的机间通信机构.在Wulor－75/32系统中,是利用基于管座(socket)的顾客—服务器模型(Client—Server Model),由"源"和"目的"处理机协同完成.而DEMOS/MP和Sprite则是利用远程过程调用(RPC)实现的,其中DEMOS/MP是由目的处理机调用RPC,而Sprite正好相反.这三个系统中,环境的保护、传送和恢复都可以并行进行.

在Wulor－75/32系统中,一旦环境恢复,该进程就被置为"就绪态",并插入运行队列,当它被切换到"运行态"后,就能从原断点继续执行.而Sprite系统则是由源处理机向目的机发一条核—核(Kernel—Kernel)的RPC,告知环境传送结束并恢复被迁进程的工作,这种RPC机构是Sprite特别为进程迁移实现的.

2　寻找空闲处理机

为改善系统性能,需要根据负载的变化,重新分布进程,亦即迁移"外来进程".如何确定负载较轻的或空闲的处理机呢?

Sprite系统用一个共享文件记录每个处理机的平均负载、空闲时间以及"外来任务"的数目.("外来任务"系指那些不是由本地用户提交的任务).为寻找空闲处理机,进程通过一个库例程(library routine)锁住该共享文件,从中选择一个合适的处理机(平均负载低、至少已工作了五分钟、目前无其它外来任务在其上运行),更新此处理机的外来作业数目,然后释放此共享文件,见图1.

当外来进程放弃占用某处理机时,此共享文件再次被锁住,有关信息再次被更新.

显然,在Sprite中使用的是集中式算法,其特点是通信开销小,效率较高,每次寻找只需要两次消息传递(实际使用的是两次RPC),但其坚固性差,一旦共享文件或其服务器故障,则整个算法瘫痪,且共享文件服务器易成为整个系统的性能瓶颈.Wulor－75/32使用全分布式算法,规则是:

(1)当本地用户恢复工作并决定独占其处理机的CPU时间时,向系统发一"迁移外来

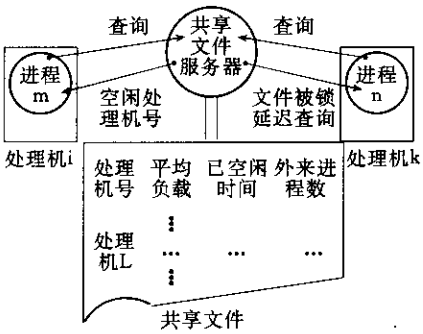

图1 Sprite中寻找空闲机模型

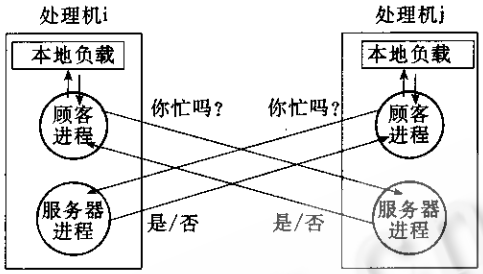

图2 Wulor－75/32中寻找空闲机模型

进程"的命令.

(2)系统调用顾客进程,轮询其他处理机,直到发现一个空闲处理机或全部询问完为止.

(3)未找到新的空闲机时,外来进程被迁移回其宿主机,即提交该进程所属的作业的那台机器上.

图2表示了这个算法的模型.

这时,为寻找空闲机,最多需要发送2(N－1)条消息(N为系统中处理机的数目),但任何处理机的故障都不会影响系统中其他部分的工作.

Condor系统采用介于集中式和全分布式之间的折中算法.本地调度器监视本机的工作负载,协调器每隔2分钟询问各机的工作负载,并根据得到的信息决定将哪些空闲处理机分配给那些需要向外调度作业或迁移进程的处理机,再由这些重负载处理机上的本地调度器选择合适的作业或进程调度、迁移到分配的空闲机上.见图3.

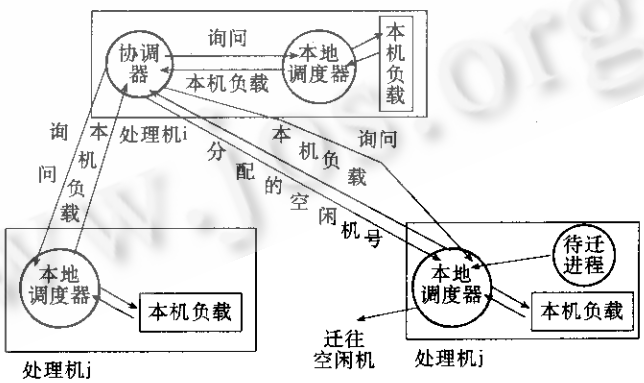

图3 Condor系统中寻找空闲机模型

为防止作业较多的用户长期占用系统所有的远程资源,为作业较少的用户提供公平的访问系统资源的机会,Condor系统的协调器采用了Up－Down算法来分配空闲机,必要的时候,剥夺重负载用户对某个远程处理机的占用,而将此处理机分配给提出申请的某个负载较轻的用户[13].

3　对通信机构和文件系统的扩充

进程迁移机构应能对用户提供透明性,这意味着无论进程被迁移到哪台机器上,都应产生相同的结果,能象在本地执行时一样地访问文件、设备和其他系统设施,并且送往此进程的消息都能正确送到.因此,需要对文件系统和通信机构加以扩充.

Sprite 系统的解决方法是设计并实现了共享的文件系统,所有文件统一命名,因而通过一个文件名总能指向同一文件,与场地无关,这样用户能透明地访问系统中的所有设备及文件.进程间的通信也使用文件系统,利用管道(pipe)和伪设备(pseudo－device)进行,Sprite 的文件系统自动地转送进程间的通信.与场地相关的系统调用则是通过宿主机转发的.

DEMOS/MP 和 V－System 是基于消息的系统,进程之间以及进程与系统之间的所有交互作用都是通过发送和接收消息来实现的.在这些系统中,消息机构已提供了一定程度的场地透明性,但必须扩充消息转发机构.

在 DEMOS/MP 系统中,消息的发送者是通过由操作系统内核管理的通信信道—链(link)来确定消息的接收者的,每条链有一个在全网络唯一的标识符,称为消息进程地址(message process address).其结构如图 4 所示.

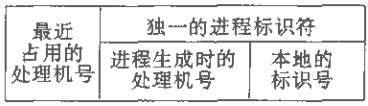

图4 DEMOS/MP中链的标识符结构

消息总是被送到链上记录的最近占用的处理机上,如果进程已被迁移,则记录的这个处理机转发这条消息,并通知此消息的发送者更新链标识符中第一项的内容.因此进程被迁移后还必须在源处理机上留下一个附属进程(dependency),以转发消息,见图 5.

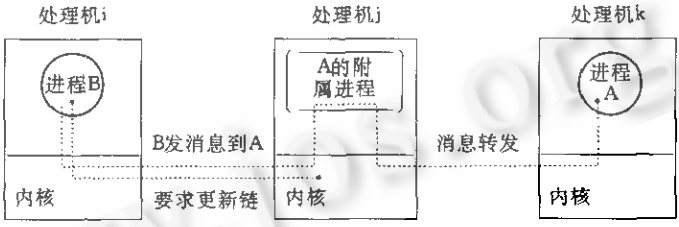

图5 DEMOS/MP的消息转发机构

这种方法的缺陷是如果进程被多次迁移,则消息需要多次转发,并且在进程曾运行的每个处理机上都留有附属进程,何时才能撤消这些附属进程仍存在问题.

V－System 中则保存一张进程到其正占用的处理机的映射表,进程被迁移后,发往此进程的消息可能不能正确送达,此时核心就在网络中广播一条"寻找当前正运行此进程的处理机"的消息,然后更新映射表,并重发那条消息,对于具有广播机构的通讯网络来说,这种方法开销较小.

Wulor－75/32 系统所基于的用 Sun－3 工作站连成的 Ethernet 网上已提供了网络文件系统,与 Sprite 系统相似.

比较而言,在分布式系统中,实现共享文件系统或实现共享虚拟内存(Shared Virtual Memory)是一种较完全的解决方法.

4 性能优化

在分布式系统中,实现负载共享、进程迁移能改善系统性能.但迁移进程时,也会给系统带来一些额外的开销,这取决于两个因素:迁移进程的次数与每次迁移所花费的时间.根据Wulor－75/32和其他一些系统的实践经验,迁移一个活动进程所需的时间不是"毫秒"级的,而是以"秒"为量级,它包括中断进程、保护和传送进程环境及恢复进程运行所需要的时间.其中用于传送打开文件及虚地址空间的开销占全部开销的绝大部分,这种额外开销必然会抵销部分甚至全部由于实现了进程迁移而带来的系统性能上的收益,因而对此各系统提出了不同的优化方案:

·共享文件系统

Sprite和LOCUS中实现了共享文件系统,因此只需要传送打开文件的有关信息以及被迁移进程对各打开文件修改了的部分.但是维护共享文件一致性的算法必需占用一定的开销,详见[16].

·预拷贝(pre－copying)技术

在V－System中,迁移一个进程时,并不立即中止该进程的运行,而是在进程运行过程中拷贝其地址空间并传送到目的处理机上,然后拷贝并传送在前一段运行中修改过的页,这个过程一直持续到最近修改的页的数目很小时才停止.中断该进程并传送其余的进程环境及最后一次修改过的页.Theimer报告说这样能将被迁进程挂起的时间降低到与将进程换入内存所需的时间相当[7],但这种方法降低了进程迁移的响应速度.

·以懒怠方式(Lazy fashion)传送内存空间

Zayas在[15]中提出一种能缩短迁移时间的方法,即仅当内存被引用时,再由目的处理机回溯到源处理机并传送.这种方法把虚地址空间的传送开销分摊到进程的执行开销之中.但是这种方法要求进程被迁移后,仍占用源处理机的内存,空间开销较大.

懒怠传送技术和预拷贝技术都需要修改操作系统内核的进程控制子模块.

·避免传送新近运行的进程的环境

在LOCUS和V－System中,对已运行时间很短的进程,不进行迁移而是重新启动,因为对于这样的进程来说,迁移它们所需要的工作量甚至超过了避免重复工作所能节省的开销.

·调度合适的作业到远程机上运行

前面所述优化方案无疑忽略了作业本身的差异及这些差异对进程迁移开销的影响.按作业类型,我们可以把作业分为"计算"型的和"信息处理"的.前者除必要的输入输出外,绝大部分工作用于与文件无关的"计算";而后者通常需要进行大量的文件处理工作,传送打开文件需占用很大的开销.因此调度"计算"性强的作业到远程机执行,留着需进行文件处理的作业在本地运行,必能降低总的迁移开销.另外,按作业的形式,还可分为"交互式"的和"后台"作业两类.显然,后台作业更适合于调度到外地运行,这样可以避免由于进程迁移在网络

间传送交互作用信息而造成的时间延迟.

为负载共享，我们在 Wulor－75/32 系统中采用了“任务加权参数算法”(Job－Weight Parameter Algorithm)进行作业调度[1,2]，这对于进程迁移亦为有益. 我们给系统一个额定值 Wt；对用户作业，根据其打开文件的数目和规模以及该作业与控制台交互作用的程度，给每个作业赋予适当的权值 R(f,i) . 若 R＞Wt，表明此作业计算性强，与用户的交互作用较少，因而系统可将这种作业优先调度到远程机上运行，且可以随时再迁移；反之，若 R＜Wt，则表明作业只适合于在本地运行，不宜进行进程迁移. 尽管这样可能产生负载的不平衡，但由于做了这种限制，使一个作业在远程机执行过程中要被迁移时，只需将进程的程序段 text、数据段 data、栈段 stack 以及 proc 结构、user 结构、text 结构等的内容传送到另一台选择的空闲机上，恢复成一个可执行进程继续从原断点做下去，而避免了做很复杂的文件处理工作. 这种方案对于降低进程迁移开销十分有效，可降低 30%左右，且易于实现[17].

迁移进程的次数与各处理机本地用户提交作业的时间及作业运行时间有关，难以控制. 同时，由于一般在大型计算机网络中，空闲处理机数目相当多，因而迁移次一般不超过 2 次. 故性能优化主要集中在如何降低每次迁移的开销上.

结　论：进程迁移的实现是一项非常复杂的工作. 在分布式系统中引入进程迁移机制，需要对系统进行多方面的扩充，包括文件系统、通信机制以及操作系统的内核，还必须实现“寻找空闲处理机”设施. 但是，实现进程迁移又能较大地改善系统性能，并为用户提供远远超过他们自己的处理机所能提供的计算能力和资源，增强系统的容错能力和坚固性，因而被许多系统作为一项设计目标，值得我们继续探索，提出更新的、更有效的方法.

参考文献

1　彭德纯等. Wulor－75/32 负载共享和远程执行系统. 武汉大学研究报告(亦为鉴定会技术报告)，武汉，1990. 10.

2　邱毓兰等. Sun 工作站网络环境中的负载共享策略初探. 全国 UNIX 及工作站会议论文集，北京，1990：18－21.

3　Litzkow Michael J *et al*. Condor——a hunter of the idle workstations. In：Proceedings of the 8th International Conference on Distributed Computing Systems，1988：104－111.

4　Douglis F *et al*. Process migration in the sprite operating system. In：Proceedings of the 7th International Conference on Distributed Computing Systems，Berlin，West Germany，1987：18－25.

5　Powell M L *et al*. Process migration in DEMOS/MP. In：Proceedings of the 9th Symposium on Operating System Principles，1983：110－119.

6　Douglis F. Experience with process migration in sprite. In：Proceedings of the Workshop on Exeperience with Distributed and Mutiprocessor Systems，Florida，1989：59－72.

7　Theimei M M *et al*. Preemptable remote execution facilities for the V－System. In：Proceedings of the 10th Symposium on Operating Systems Principles，ACM，1985：2－12.

8　Walker B *et al*. The LOCUS distributed operating system. In：Proceedings of the 9th Symposium on Operating Systems Principles，ACM，1983：49－70.

9　Stone H S. Multiprocessor scheduling with the aid of network flow Algorithms. IEEE Trans. on Software Engineering，1977：85－93.

10　Stone H S *et al*. Control of distributed processes. Computer，1978：97－106.

11　Robinson J T. Some analysis techniques for asynchronous multiprocessor algorithms. IEEE Trans. on Software Engineering，1979.

12　Bokhard S H. Dual processor scheduling with dynamic reassignment. IEEE Trans. on Software Engineering，

1979.

13 Mutka M W *et al*. Scheduling remote processing capacity in a workstation processor bank computing system. In: Proceedings of 1st International Conference on Distributed Computing Systems, Berlin, 1980:2—9.

14 Nichols D A. Using idle workstations in a shared computing environment. In: Proceedings of the 11th Symposium on Operating Systems Principles, ACM, 1987:5—12.

15 Zayas E. Attacking the process migration bottleneck. In: Proceedings of the ACM Symposium on Operating Systems Principles, 1987:13—20.

16 Nelson M *et al*. Caching in the sprite network file system. ACM transactions on Computer Systems, 1988:134—154.

17 肖红等. 自适应的负载共享策略研究. 计算机杂志, 1992; **20**(1—2):21—25.

PROCESS MIGRATION METHODS IN DISTRIBUTED COMPUTING SYSTEMS

Xiao Hong, Qiu Yulan and Peng Dechun

(*Department of Computer Science, Wuhan University, Wuhan* 430072)

Abstract In a distributed computing system, process migration is a method by which executing processes may be transferred between nodes during their lifetime and resumed from the switch—off—point in the destination node. This paper addresses at length the implementation ways of process migration in the Wulor—75/32 and other typical distributed computing systems, as well as extensions to other system facilities: file systems, communication mechanisms and OS kernel for implementing process migration.

Key words Distributed system, process migration, finding idle machines, interprocess communication mechanism, performance optimization.