维普资讯 http://www.cqvip.com

计算机科学 2001Vol. 28№. 9

# 群机系统内核外实现进程迁移的研究

Research of Process Migration outside the Kernel in Cluster

张 怡 胡 凯 胡建平

(北京航空航天大学计算机科学与工程系 北京 100083)

**Abstract** The network-based clusters will get a wide application in the network computing. One of important guarantee of system performance is to actualize the migration of process. This paper discusses a way of migrating process outside the kernel in the system of cluster, analysises the process, mechanism and limitations of storing the state of process in the way of checkpoint, and restarts the process on the remote machine in detail.

**Keywords** Checkpoint, Process migrate, Remote system calls

## 1 引言

群机系统(cluster)是实现网络高性能计算的一种有效方式。它是通过网络将一些工作站或 PC 机连接起来,并在之上附加一些管理软件,将连接的机器协调起来共同工作,从而获得高吞吐量计算。目前,已经有 Condor, LSF, Condine 和 Beowulf 等多个商业和科研软件包支持群机系统的运行。群机系统的基本思想是通过占用网络上空闲资源以获得高吞吐量的网络计算能力。它可以处理批处理任务和交互任务,在其上安装 PVM 或 MPI 等并行环境后,多数群机系统都支持并行计算。当在群机的任何一台主机上提交任务后,中央管理器将在系统中寻找空闲主机,并将任务分派到不同的空闲主机上,利用这些空闲的 CPU 资源来协同完成任务。在这种群机系统中,利用进程迁移可以实现系统内部的动态负载平衡、提高并行计算能力、实现高效容错、减少通信开销等应用[1]。

由于系统实现高性能计算是利用网络上的空闲主机资源,因此必须同时保证构建群机系统的其他主机的自主性,主机所有者不能感觉到自己的主机被其他人占用了。当空闲主机被所有者重新占用,使其负载加重时,系统必须能够及时将外来任务撤出,即将正在运行的进程迁移到群机的其他主机上继续运行(动态进程迁移)来保证整体任务的完成。这是进程迁移在群机系统中最直接和最重要的需求。同时,进程迁移还是群机系统进一步实现动态负载平衡的基础。

## 2 内核外实现进程迁移的策略

进程迁移是实现群机高吞吐量计算的基础机制,特别适合于面向利用网络空闲资源的群机系统,它保证了分派到其他机器上的任务能够高效率执行完毕。早在 80 年代,进程迁移就作为分布式操作系统的一种系统功能来开发,DEMOS/MP、LOCUS、V 和 SPRITE 系统都在一定程度上实现了进程迁移,这些系统主要提供内核级的支持,通过内核原语访问进程状态,设立检查点和透明迁移应用进程。但是目前流行的群机系统,多数都是依赖于原有低层操作系统,并在其上附加一定的管理软件来达到群机协同工作的效果,并不从低层重新开发自己专用的操作系统,因此,在群机中多采用在用户级实现进程迁移的方法,以维持原有操作系统的通用性和一致性。例如 CONDOR[2]、CONDINE、COCHECK 等系统采用让应用程序重新链接相关的检查点库,无需修改源代码的方法,而 CLIP[3]等系统则采用更为直接的方法,需要将检查点库函数插入到相关的源代码中。本文讨论的内核外实现进程迁移的方法,无需修改源代码,只需在程序链接时链接入附加库,并在实现时做相应的调整处理。

### 2.1 进程状态

在传统的 UNIX 系统中,进程被抽象为上下文信息。一个进程的上下文是由用户空间的内容、硬件寄存器的内容以及与该进程有关的数据结构组成,更严格地说,进程上下文由它的用户级上下文、寄存器上下文和系统上下文组成[4]。

张 怡 博士生,主要研究领域:分布式操作系统、分布式并行计算和网络计算。胡 凯 博士生,主要研究领域:分布式操作系统。胡建平 博士生导师,主要研究领域:分布式系统、网络计算、知识工程等。

维普资讯 http://www.cqvip.com

进程的用户级上下文由进程的正文段、数据段、用户段组成，它们构成了进程的虚拟地址空间；寄存器上下文包括：程序计数器 Pc、状态寄存器 Ps(记录机器与该进程关联时的硬件状态)、栈指针、通用寄存器；系统上下文包括：进程表表项、进程 u 区、本进程区表表项、区表及页表(定义了内存虚拟地址到物理地址的影射)、核心栈等。在保存进程状态以供进程迁移后恢复时，并不是所有这些信息都需要保存，实际上，只需要保存用户级上下文、寄存器上下文和部分系统上下文，遗憾的是，进程的系统级上下文是由操作系统的内核来管理的，有些信息在内核外的用户级是获取不了的，因此我们采取了一些策略来记录需保存的部分系统上下文信息，以此来完成内核外实现进程迁移。

除了以上的进程上下文信息外，进程状态还包括一些和进程运行相关的东西，例如打开文件的状态，进程的信号处理以及其他一些与主机状态相关的进程运行环境，保存这些信息将有助于进程恢复后的正常运行。

### 2.2 检查点

检查点机制最初是作为分布式系统的系统容错措施提出来的，它是一种时间冗余容错方式。当应用程序正常运行时，定期设立检查点，将进程的当前状态保存起来，如果出现系统崩溃，就马上恢复最近一次可用的检查点文件，使应用程序回退尽可能少的部分，来尽量减少运行损失。目前，群机系统中多把检查点机制作为进程迁移的基本实现策略，简单地说，利用检查点机制实现进程迁移，就是将当前运行的进程设立检查点，保存进程状态和上下文，然后将进程检查点文件转移到另一主机上，恢复进程状态，在断点处继续执行的过程。

实现检查点算法主要包括以下步骤：

(1)保存进程的虚拟内存信息：即进程此时的数据段、堆栈段、代码段内容。进程的虚拟内存如下图所示：

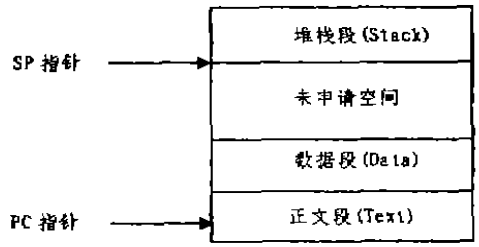


图 1 进程的虚拟内存空间

在用户层可以直接按地址读取虚拟地址空间的内容。不管是初始调用进程或重启进程，一般都使用相同的正文段，因此进程的正文段甚至可以不用保存。紧临正文段的是数据段，一般的数据段开始位置都是由操作系统指定的，可以获得。而且数据段的内容都是连续存放的，因此只要再获得末尾地址就可以存取整段地址，末尾地址可以通过低级的系统调用 sbrk()取得。进程中间的未申请空间无需保存，下面的重点是保存堆栈信息，它的开始位置也是由操作系统指定的，并且大多数类 UNIX 操作系统是由进程的虚拟内存顶端向低地址方向发展的。我们需要保存堆栈本身的内容和它的上下文内容，特别是它的堆栈指针。堆栈的上下文可以由系统调用 setjmp()和 longjmp()来协助获得，但是我们知道利用 setjmp()得到的是地址指针，而不是它的内容，因此我们仍然需要保存堆栈的内容，然后再用 setjmp()和 longjmp()来保存和恢复它的位置。存储堆栈的内容实际上也可以通过堆栈的起始和结束点来获得，堆栈的起始点是固定的，终止点可以从通过调用 setjmp()得到的存储在 JMP_BUF 中的堆栈指针中获得。综上所述，整个进程的虚拟地址空间可能被全部保存，以备迁移后恢复。

(2)保存进程所打开文件的状态。这些信息被系统保存在内核中，不易剥离，我们就采取了一些特殊的手段来保存，任何被进程打开的文件及其状态都在打开的同时被记录下来。这一过程当然不会被操作系统自动记录，所以在编译应用程序前，先进行程序的预处理过程，将所有打开文件的系统调用都用相应的包裹函数代替，包裹函数在执行原来的系统调用前执行一些附加的操作，例如：将打开文件的文件描述符、文件打开模式、文件偏移量等信息记录下来，存储在固定的数据结构中。因此，这些信息就被正确地保存起来了。

(3)保存进程的其他一些相关状态。这些状态包括：进程 ID 号、进程使用的信号和信号处理、CPU 状态等进程运行环境信息。这些信息有些可以被直接读取，有些就需要采取一定的技巧来取得，此处就不再一一赘述。

保存的状态信息可以被存储成文件，等到系统找到合适的新空闲主机后，再转移到新主机；也可能由于在收集状态时，已经选定了新主机，从而将状态信息直接发送到某个设定的 socket 连接端口。进程状态被转移到新主机后，将依照保存时的相反顺序恢复，进程将从上次中断的地方继续运行。

### 2.3 进程远程执行

当系统找到了合适的空闲主机后，进程将会派到这一主机上运行，本次执行将会附带一个特殊的参数，表示不是首次执行，而是执行迁移的中断进程。带参数的进程先由控制进程接管，派生新进程之后，恢复检查点文件，即将新进程的正文段、数据段、堆栈段的内容恢复成保存的内容，并设置堆栈指针和程序计数器指针到保存的位置，然后恢复进程其他的保存状态信息，

维普资讯 http://www.cqvip.com

让进程从断点处恢复执行。

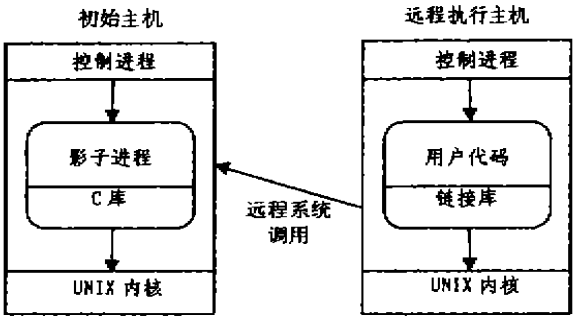


图2 进程远程执行

进程迁移到其他主机上运行，有可能由于主机状态改变而导致一系列复杂的问题。例如：由于文件访问位置改变而导致无法访问；某些和位置有关的系统调用无法实现等。本文讨论一种利用转发技术实现的方法，称为远程系统调用。转发技术，往往针对一些与位置有关的操作，如部分系统调用、信号处理等。这些操作有的无法在新的目的节点上独立完成，有的虽可完成但困难较大，因此采用这种在目的节点上拦截操作，转送回源节点进行处理后再将结果返回到目的节点的方法。

所谓远程系统调用，就是系统在链接原始代码时，链接了一些检查点库，其中包括对某些系统调用的处理链接，如果发现进程是继续执行的迁移进程，执行这些系统调用时，将不会直接调用本地的C函数库，而是利用远程过程调用（RPCs）将操作转移到原始提交任务的主机上执行，为了支持这种远程系统调用，在进程提交主机上，将会派生一个影子进程，来完成其后可能用到的调用工作。运行在原始主机上的影子进程负责执行传来的过程调用，并将结果打包传回运行主机，这样，和位置有关的系统调用就象在本地执行一样，得到正确的结果。通过远程系统调用还可以使文件系统的访问独立于位置。进程迁移需要某一进程从不同的主机位置上可以访问到一致的文件。持续在原始主机上运行的影子进程就担当了迁移进程文件访问代理的作用，迁移进程对文件的访问就可以重定向到原始主机的影子进程上来执行。这种重定向被定位到系统调用的级别，实际上任何直接使用读写、打印命令或间接调用这些命令的例程都应该做如上的处理。

这种远程调用的方法对源节点和目的节点都会造成一定的影响，它产生对源节点的剩余依赖性。但是，我们所讨论的进程迁移不是用在系统容错上，而是实现利用网络空闲主机的应用上，因此采用这种方法还是比较适合的。而且，系统还采用了一些限制措施，例如当进程发生多次迁移时，将检查点文件送回原始主机，再由原始主机转移到合适的新主机的方法来避免对中间主机的多重依赖。

### 2.4 群机系统中进程迁移的实现和作用

以上我们所讨论的迁移方法力求适合一种松散偶合的群机系统，整个群机系统由多个主机以网络连接的方式组合起来，我们在考虑系统整体工作能力的同时，还必须考虑单个用户的自主性、网络的不确定性等因素，在不影响机器主人工作的前提下，充分利用网络主机的空闲CPU时间来完成需多个机器协同解决的大运算量任务。因此，进程迁移在这样的系统中主要起保证主人优先使用原则和实现系统负载共享的作用。

需多机协同解决的大运算量任务可以在系统内的任何一台主机上提交、提交的任务可能被派遣到系统的任何一台空闲主机上运行，同时提交主机将会产生轻负载的影子进程来协同工作。如果支持检查点和进程迁移的功能选项为打开（此功能可选），在进程进行链接时，就会自动链接检查点库，并在最初执行时，执行一些额外的操作：设置检查点信号的处理程序。当远程主机的主人返回使用机器，并使该机负载达到一定的指标时（此负载指标可以人工设定），中央服务器将产生迁移命令、远程运行进程将得到检查点信号，并进入信号处理来执行检查点操作。系统根据定期的交换信息，将能够发现合适的新空闲主机，检查点信息将被转移到新主机上。当进程迁移到其他机器上重新运行时，进程首先执行恢复例程而不是执行原始用户代码，恢复例程将负责恢复迁移进程状态，中断进程将继续执行。这些操作对于用户来说都是透明的。下图描述了一次典型的进程迁移总体实现过程：

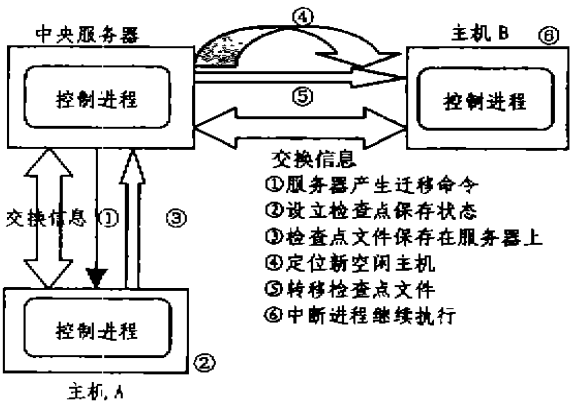


图3 典型的进程迁移总体实现过程

由上面的叙述可以看出，进程迁移模块实际上是由更高层的负载共享模块调度的，负载共享模块使用这种动态剥夺式的方法实现整个系统的负载分担，得到更高的执行加速比。

维普资讯 http://www.cqvip.com

## 3 系统的局限性及其相关研究

UNIX 内核外实现进程迁移具有很重要的优点，它不依靠任何的内核代码，实现相对简单，又具有很好的可移植性，原则上，大多数的类 UNIX 系统都可以支持。以占用网络空闲资源来获得高吞吐量的非专用群机系统，由于要保持其软硬件平台的通用性，尤其适合于采用在内核外实现进程迁移的方法。

但是这种方法也具有一定的局限性，这是因为在用户级不能全部获取所有内核维护的信息，在内核外能够获取的信息对于多数进程来说是足够的，但是，对于一些特殊类型的进程就"够了，例如使用 fork()或 exec()派生多个子进程的进程，和某些利用套接字、管道通信的进程。这就有可能需要用户选择适合的方法编制程序来配合进程迁移，或者采用分步处理的方法，先采用只发消息并缓存的方法处理完进程间的通信关系，然后执行检查点操作。后一种方法特别适合应用于并行环境中。群机系统如果要支持并行，则还需要在底层操作系统之上安装并行软件，例如 PVM 或 MPI，执行检查点时，首先考虑并行进程的一致性问题，保证它们之间不丢失或重叠消息，并合理地处理好它们之间的通信关系，做完这些操作后，再退出并行环境，在并行环境外执行状态收集和保存，恢复运行时，也是先在并行环境外恢复进程状态，然后进入并行环境，恢复进程间的通信关系。并行环境下的检查点和进程迁移的实现超出了本文所讨论的范围，但是，它和我们以上提出的检查点和进程迁移方法相结合，却是解决迁移一个或多个彼此之间相互通信的进程的有效方法之一。

**性能与结论** 内核外实现进程迁移的方法特别适合于处理需要占用大量 CPU 时间的大计算量的科学计算任务。这是由于比较进程寿命与迁移检查点文件的总体开销的结果，一般来说，进程的可迁移性主要与迁移开销有关，我们得到经验公式，若进程满足：

age＞A＊迁移开销，0.1＜A＜1。(age：表示进程已使用的 CPU 时间)

则迁移这样的进程都会获得较为满意的效果。另外，单次检查点设置的开销也是检验迁移效率的主要指标之一，由于主要面向长运行周期的计算任务，一般单位检查点设置时间开销都不会超过 1%。因此，采用内核外进程迁移也可以获得较好的性能。目前，我们正在 LINUX 操作系统之上实现以上的系统，整个系统的核心是利用网络空闲主机资源来获得高性能计算，进程迁移作为其中的一个模块，协助解决群机系统的负载平衡、负载共享及其主人优先使用原则的问题[4]，它是检查点算法的一个延伸和扩充，除了实现单任务的检查点和进程迁移算法，在群机系统中支持并行是我们的另一个重要目标。因此，研究和实现支持 PVM 和 MPI 的检查点和进程迁移机制也是我们下一步的主要工作。

## 参 考 文 献


1 Barker M, Buyya R. Cluster Computing as a Glance. Available at: http://www.dgs.monash.edu.au/~rajkumar/cluster/v1chapl.ps.

2 Bricker A, Litzkow M, Livny M. Conder Technical Summary. Available at: http://www.cs.wisc.edu/condor/doc/ Oct. 1991

3 Chen y, Plank J S, Li K CLIP: A Checkpointing Library for Intel Paragon. SuperComputing'97, San Jose, CA, 1997

4 胡凯，王强，胡建平．机群并行计算中负载共享的关键技术．计算机科学，2000，27


(上接第 50 页)


5 Greenberg M S Mobile Agents and Security. IEEE Commun. Mag., 1998, 36(7): 76～85

6 Faber T. ACC: Using Active Networking to Enhance Feedback Congestion Control Mechanisms. IEEE Network, May. 1998

7 Calderon M, et al. Active Network Support for Multicast Applications. IEEE Network, 1998(May): 46～52

8 Denneth L, et al. Directions in Active Networks. IEEE Communication Magazine, 1998(Oct.): 72～78

9 Kasera S K, et al. Scalable Fair Reliable Multicast Using Active Services IEEE Network, 2000(Jan/Feb): 48～57

10 Putzolu D, et al. The Phoenix Framework: A Practical Architecture for Programmable Networks. IEEE Communication Magazine, 2000(Mar.), 160～165

11 Decasper D S, et al. A Scalable High-Perfomance Active Network Node. IEEE Network, 1999(Jan/Feb). 8～19

12 Alexander D S. The Switch Ware Active Network Architecture. IEEE Network, 1998, 12(3): 29～36

13 Alexander D S. A Secure Active Network Environment Architecture: Realization in Switch Ware. IEEE Network, 1998, 12(3): 37～45

14 Andrew T, et al. A Survey of Programmable Networks, ACM Computer Commun. Review, 1998

15 Schwartz B, et al. Smart Packets for Active Network, BBN Tech. http://www net-tech. bbn. com/smtpkts/smart. ps. gz, 1998