第7卷第5期 软件学报 Vol. 7, No. 5
1996年5月 JOURNAL OF SOFTWARE May 1996

基于Transputer网络的进程迁移技术研究*

陈勇 刘心松 苏森

(电子科技大学计算机系 成都 610054)

摘要 进程迁移是分布式系统研究的重要内容,它对实现系统容错和负载平衡起到很重要的作用.本文描述了Transputer网络系统中进程迁移的设计思想、方法及技术实现.首先对所采用的硬件环境及整个容错并行操作系统的层次结构进行了介绍,然后对进程迁移所涉及的关键技术进行了详细讨论.该平台的建立为动态任务调度及容错处理等技术的进一步研究提供了强有力的支持.

关键词 并行处理,进程迁移,工作空间,运行环境,重定位.

"进程迁移"是一种将某处理机上的进程运行环境传送到另外的处理机上,并使其从"断点"处继续执行的方法.它作为分布并行处理领域的一项关键技术已引起人们的广泛注意,目前对它的研究非常活跃,很多分布并行处理系统都将其作为一个设计目标,如Condor[1],V-System[2],Sprite[3],DEMOS/MP[4]等.

在Condor系统中,每个处理机都有一个局部调度器,并且其中一台上还有一个协调器.局部调度器负责监视本机的工作负载情况,协调器则定时(间隔2分钟)向其它处理机发出信息以查询各机的工作负载,并根据收到的信息来决定将把重负载处理机上的进程迁移到哪些空闲处理机上.一旦决定了空闲处理机则通知重负载处理机,再由这些重负载处理机上的局部调度器按照一定的原则选取相应的进程并迁移到分配的空闲机上.可以看出,Condor系统采用的是界于集中式和分布式之间的方法来实现进程迁移,因而容错性能较差.

V-System所采用的进程迁移技术也有其特点.在该系统中,进程运行环境的迁移采用了预拷贝(Pre-copying)技术.当迁移某个进程时,并不立即终止该进程的执行,而是在进程运行过程中拷贝其地址空间并迁移到目标处理机上,若在迁移期间其地址空间又做过修改,则又把修改过的页迁移到目标处理机上,直到最近修改过的页很少时才中断该进程,然后再迁移修改过的页以及其它必要的运行环境.不过,当进程对页面修改操作较频繁时,系统开销较大.

* 作者陈勇,1965年生,副教授,主要研究领域为并行处理,神经网络系统,操作系统.刘心松,1940年生,教授,主要研究领域为分布式计算机系统,并行处理,操作系统及通信软件.苏森,1971年生,硕士,主要研究领域为分布式计算机系统,容错处理系统.

本文通讯联系人:陈勇,成都 610054,电子科技大学计算机系

本文1995-03-06收到修改稿

但是由于进程迁移本身与所处的硬件平台有紧密的关系,因此它们的实现技术也不相同. Transputer 是英国 INMOS 公司推出的一种功能强、性能好、技术先进的处理器芯片[5],它强大的多任务能力以及 4 条高速通信链路使得它非常适合建立点对点通信的分布并行处理系统;其所特有的 OCCAM[6]语言为最大程度发挥它的性能提供了良好的程序设计基础. 但在其上实现进程迁移在国内外尚未见到报导. 在"八五"攻关项目的支持下,我们对此进行了研究和探索,并将其作为所开发出的容错并行操作系统的一部分. 将进程迁移技术用于容错系统是我们开发的系统的重要特点,本文就是我们初步研究结果的总结.

1　硬件环境

作为基础,我们选用了 Quintek Fast9 板作为研究的硬件平台. Quintek Fast9 是 1 个符合 INMOS 数据传输控制通用标准的可互连的板一级 Transputer 产品,板上含有 9 个 T800 以及 1 个 CrossBar Switch(C004),IBM PC 机可用作宿主开发机,如图 1 所示.

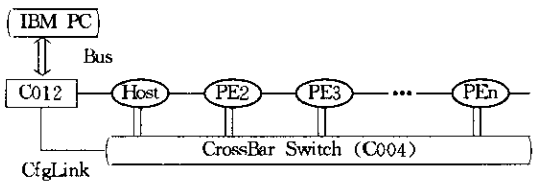

图1 硬件平台结构

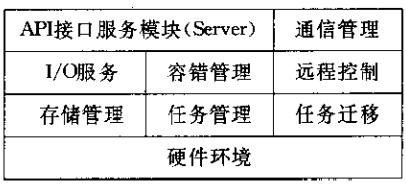

图2 FTPOS核心层次结构

• 宿主机:作为 Transputer 网络的文件服务器以及 I/O 控制.

• 主 Transputer:完成程序从宿主机到各个从结点的加载以及各从结点向宿主机提交的 I/O 请求; 按需调整 CrossBar Switch,实现各种拓扑的网络结构.

• 从 Transputer:用作并行处理计算单元,每单元有其私有存储器 1MB.

• CrossBar Switch:接收来自主 Transputer 的配置信息,实现拓扑变换.

• Link adaptor:C012 完成主 Transputer 与宿主机 I/O 数据的串并转换;将 C004 的 Config Link 映射到主 Transputer 的地址空间上.

2　软件的层次结构

对进程迁移的研究除了需要任务管理方面的必要支持外,还需要一些额外的支持,如存储管理、通信管理等. 为此首先介绍整个操作系统 FTPOS 的层次结构,如图 2 所示.

其中 API 接口服务模块(Server)是应用程序与系统打交道的唯一接口,通过它应用程序可以请求系统为其服务;通信管理模块负责结点间的可靠信息交换,它是进程迁移的基础;远程控制机构为进程的迁移提供必要的原语,如发送、接收原语等;迁移管理模块负责在源结点上对迁移进程运行环境进行拾取以及在目标结点上对其重新恢复,它是本文的核心;任务管理负责对结点内的任务进行创建、调度以及撤消等;存储管理则负责对局部存储器进行分配、回收等处理.

这里有必要说明一下所采用的容错方法. 由于我们的设计目标只对硬件故障进行容错,因此采用每个结点上的任务在其右邻结点中都有其备份环境(最右边结点上的任务备份在最左边的结点上),而应用程序通过调用 API 函数 Migrate()在程序某处或多处设置备份

点，当程序执行到该 API 函数时，服务进程 Server 便把应用进程的运行环境迁移到备份结点中，当系统检测到某结点发生硬件故障后，便把在其备份结点上的进程运行环境恢复并执行，从而达到容错目的. 这部分工作由容错管理模块来完成，具体实现将另文详述.

3 进程迁移技术

3.1 进程迁移思想

我们知道，进程迁移就是将一处理机上的运行进程环境迁到另外的处理机上，并使其从“断点”处继续执行的过程. 在我们设计的系统中，进程迁移过程如图 3 所示.

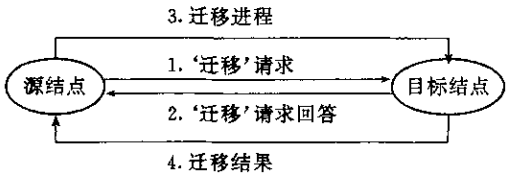

图3 进程迁移过程描述

它主要由 4 个处理阶段完成：(1)寻找阶段：确定某结点 i 的备份结点号 j；

(2)协商阶段：当某结点 i 需要向结点 j 迁移进程环境时，向其发出请求信息，若得不到回答信息(如结点 j 发生故障)或收到‘拒绝’的回答信息(如结点的资源不能满足请求要求)，则进行失败处理；若收到‘同意’的回答信息，则进入下一阶段；

(3)传输阶段：把进程运行环境按要求的格式调用发送原语发往结点 j；

(4)善后阶段：传输成功后，对相关数据结构进行处理.

为完成上述 4 个阶段的工作，系统提供了 4 个处理过程：

• FindNode(What:action) 寻找备份结点；

• Negotiation(Where from:Node; Where to:Node; Memory:Size) 协商接口，用于确认备份结点是否有足够的资源；

• Transport(Where from:Node; Where to:Node; Which: process) 传输进程运行环境；

• Postprocess(What:action) 善后处理.

3.2 进程迁移实现

进程迁移在技术实现上与硬件有紧密的关系，抽象来看，它涉及到 2 个主要方面：

(1)所迁进程运行环境的拾取；

(2)所迁进程运行环境在目标处理机上的恢复.

对前者的需要是很显然的，若不知进程的运行环境何谈进程迁移；对后者来说，是个重点而且也是难点，它所涉及的主要问题是对所迁进程运行环境进行重定位，因为所迁进程在目标结点机上所处的地址空间很可能与原来不同. 下面我们就这 2 个问题分别加以讨论.

3.2.1 进程运行环境的提取

作为研究的初期阶段，一开始我们尽量对问题进行简化，为此作出如下假设：

假设 1：一个进程对应磁盘上的一个文件；

假设 2：一个进程内部不再有并发执行单位；

假设 3：进程使用系统提供的 API 函数相互通信，且不对物理链路直接操作；

假设 4：进程通过系统提供的 API 函数来分配内存空间；

假设 5：进程只在程序最外层调用函数 Migrate().

对 Transputer 来说，进程的运行环境主要包括进程的代码空间、工作空间(相当于数据区)以及指令指针 IP 等. 由 INMOS 公司提供的 Transputer 开发系统所开发的可执行代码中包含了进程运行所需环境参数，通过分析发现其格式如图 4 所示.

```
Struct ExeFile {
  INT   code—description—length;      ——代码接口描述长度
  BYTE code—description[];            ——代码接口描述
  INT   compiler—identifier—length;   ——编译器标识符长度
  BYTE compiler—identifier[];         ——编译器标识符
  INT   target—machine;               ——目标机器
  INT   interface—version;            ——接口版本号
  INT   work—space—requirement;       ——所需工作空间大小
  INT   vector—space—requirement;     ——所需向量空间大小
  INT   code—entry—offset;            ——代码入口偏移
  INT   code—length;                  ——代码长度
  BYTE code[];                        ——代码
}
```

图4 可执行文件格式

基于上述文件格式，任务管理模块在创建进程时对每一个进程创建一个进程控制块(PCB)，其数据结构如图5所示.

NextPcbPtr	TaskID	CodeAreaPtr	CodeAreaSize
EntryOffset	TaskStatus	WorkSpacePtr	WorkSpaceSize
BreakPointWs	MemBlkPtr	DataQueuePtr	Reserved

图5 PCB 数据结构

其中 CodeAreaPtr，CodeAreaSize 确定了进程代码所在空间的地址及大小；而 WorkSpacePtr，WorkSpaceSize 则确定了进程运行所需工作空间以及向量空间的地址及大小；EntryOffset 为程序执行入口点指针；TaskID 域为进程标识，它由2个字构成：高位字为结点编号，低位字为结点内部编号，因此只要保证内部编号唯一就可保证进程标识唯一；TaskStatus 描述了进程当前状态，有4种主要状态：就绪(Ready)、运行(Run)、阻塞(Block)、挂起(Suspend)，正在被迁移的进程处于挂起状态；BreakPointWs 域用于迁移，其后将详细说明；同一进程在运行过程中申请的内存块是以链表形式链接起来的，该链表的首指针保存在 MemBlkPtr 域中，至于 DataQueuePtr 域用于进程间的数据通信.

基于上述 PCB 结构，任务管理模块便可对进程的运行环境进行管理和跟踪. 有关这方面的资料较多，不再赘述.

3.2.2　迁移进程运行环境的恢复

Transputer 的地址空间为线性空间，它对代码的访问是以地址 $\#8\times10^7$ 为基准的，而对工作空间的访问则是以工作空间地址 ws 为基准. 因此，对所迁进程环境的恢复应作如下工作：对工作空间地址的重新调整；对迁移进程的指令指针 IP 的调整以及对过程调用返回地址的调整. 对于前者，首先需要知道迁移进程工作空间地址的值，然后根据进程迁移前后的工作空间地址之差(△ws)对其进行重新调整；对于后者，在3.2.1节中的假设条件下，就是用进程迁移前后的代码空间地址之差(△ip)对当前指令指针 IP 进行调整，以及对进程调用

函数 Migrate()的返回地址进行调整.

由文献[5]可知,进程因对通道操作而处于等待状态时,该通道保存了进程的工作空间的地址 ws,显然在进程迁移后应该对其调整,而且通过分析发现:[ws－1]单元保存了下一条指令指针,[ws－3]单元保存了需要交换信息的地址,很明显这2个单元也都需要调整.

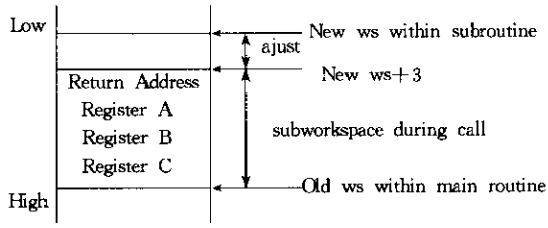

图6 过程调用机制

由3.2.1节中假设可知,迁移进程只调用了函数 Migrage(),因此只需对其返回地址进行调整即可,通过分析发现,Transputer 过程调用机制如图6所示.

当调用子过程时,硬件自动创建子工作空间并保存4个字:返回地址和3个堆栈寄存器(A,B 和 C),ajust 指令用于为子过程的局部变量分配空间,对 Migrate()函数来说,只分配3个字的局部变量空间,此时 New ws 即为子过程的新工作空间地址.

根据上述分析结果,我们在系统设计中,通过定义软通道 SystemChan,使应用进程与 API 接口服务进程(Server)进行通信,并最终实现进程迁移.现给出函数 Migrate()的源程序对其加以说明.

```
PROC Migrate()
  CHAN OF ANY SystemChan:      --定义通信软通道
  PLACE SystemChan AT SystemChanAddress:      --定义软通道地址
  INT Resume:
  SEQ
    SystemChan ! Migrate.Api      --通知 Server 备份进程运行环境
    SystemChan ? Resume      --等待 Server 激活该进程
:
```

当进程调用该函数后,通过语句 SystemChan ! Migrate.Api 请求 Server 的服务,而自己则通过执行语句 SystemChan ? Resume 进入等待状态,此时通道 SystemChan 保存了该进程(也即过程 Migrate())的工作空间地址.Server 接到请求后,便把进程运行环境(此时进程已处于等待状态)包括 SystemChan 的值一同迁移到备份结点.备份结点收到迁移进程运行环境后,则为其分配内存空间,计算出△ws 和△ip,然后用△ws 对所传 SystemChan 通道值调整,并通过它找到新的工作空间地址 ws,将其保存到 PCB[BreakPointWs]中,随后用△ws 对[ws－3]单元进行调整、用△ip 对单元[ws－1]和[ws＋3]进行调整,此时便完成了对迁移进程的重定位.当需要运行该迁移进程时,只需定义一个软通道(例如 ResumeChan),把该进程的工作空间地址(保存在 PCB[BreakPointWs])作为该软通道的值,然后对该通道输出一个 INT 数据即可.

4 进一步讨论

以上我们详细描述了在 Transputer 网络环境下进程迁移的技术实现.在分布并行处理系统中,有效的进程迁移的确能够改善系统的性能,但是进程迁移所带来的额外开销也可能

抵消它所带来的好处.这种额外开销主要由2方面决定：①每次迁移进程运行环境所需要的时间，这往往取决于系统通信能力的强弱以及进程运行环境本身的大小；②进程迁移的频繁度，这又取决于系统设计者对进程迁移算法的设计.我们所设计的是基于容错的进程迁移，只有当检测到某处理机发生故障情况下才迁移进程，不过该技术同样可以用于均衡系统的负载.在进程迁移的通信传送开销上，我们进行了初步实验，在 Transputer 高速通信链路的支持下(20MBit/S)，若迁移环境大小为32K，目标结点与源结点间的路径为3，进行一次迁移的时间开销约为270ms.不过，若系统的通信更繁忙的话，这种时间开销还会有所增加.

进程迁移技术涉及到系统的很多反面，目前尚有许多问题有待解决.目前我们正在对其做进一步的研究，放宽3.2.1节中的假设条件，使其更加完善.最后，作者衷心感谢曾帮助过我们的同事和朋友.

参考文献

1 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.

2 Theimei MM *et al*. Preemptable remote execution facilities for the V—system. In: Proceedings of the 10th Symposium on Operating Systems Principles, ACM, 1985. 2～12.

3 Fred Douglis, John Ousterhout. Process migration in the sprite operating system. In: Proceedings of the 7th International Conference on Distributed Computing Systems, 1987. 18～25.

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

5 The Transputer Databook. UK: INMOS Group of Companies, 1989.

6 OCCAM 2 reference manual. UK: Prentice—Hall International Hemel Hempstead, 1989.

7 Yeshayahu Artsy. Designing a process migration facility. IEEE Trans. Computer, September 1989. 47～56.

8 Hoare. Communicating sequential process. Comm. ACM, 1978, **21(8)**: 666～677.

RESEARCH ON PROCESS MIGRATION BASED ON TRANSPUTER NETWORK

Chen Yong　Liu Xinsong　Su Sen

(*Department of Computer Science　Electronics Science and Technology University　Chengdu*　610054)

Abstract　Process migration is an important research domain of distributed computer systems. It plays an important role in realizing fault tolerant and load balance of systems. This paper describes a process migration facility based on Transputer network. First, the hardware environment adopted and the architecture of the operating system are introduced. Then some key technologies about the process migration are discussed in detail. This platform will provide efficient support for the further research in dynamic task scheduling and fault tolerant processing.

Key words　Parallel processing, process migration, work space, execution environment, relocation.