文章编号:1000－2367(2001)02－0078－04

多线程的实现方法①

张 利 霞

(河南师范大学计算机科学系,河南 新乡,453002)

摘　要:本文在分析了多线程的概念后,详细描述了多线程的实现方法及其在实现上的差异.

关键词:进程;线程;多线程;实现

中图分类号:TP316　　文献标识码:A

近年来,随着多处理机的发展,多线程技术引起了计算机界研究人员的高度关注.为了方便用户开发多任务并行程序,进程内多线程概念被提了出来,多线程技术致力于线程级的并行性的开发,使得用户可以开发共享地址空间的并发或并行程序.

多线程的思想最早在 B.Smith 设计的 Deneclor HEP 机器中率先采用,更早一点的历史可以追溯到 60 年代CDC6600 机器中的多功能部件计分牌.到了 80 年代初期,一些基于微内核的操作系统开始提出线程的概念并加以研究.此后,一些流行的操作系统如 Windows 95/NT 等也纷纷增加了对多线程的支持,试图用它来提高系统内程序并发执行的程度,从而进一步提高系统的吞吐量,而且在新推出的程序设计语言及其它应用软件中,也都纷纷引入了线程,来改善系统的性能.总之,由于多线程能够更好地开发并行性和提高系统性能,在体系结构、操作系统、运行库、程序设计语言各级增加对多线程的支持已成为一种趋势.

1　线程与进程

传统操作系统中常常用到一个重要概念,那就是进程.进程是一个内核级的实体,它由程序控制块 PCB (Process Control Block)、程序、数据集合组成.其中,程序部分描述了进程所要完成的任务,数据集合包括两方面内容,即程序运行时所需要的数据和工作区,数据通常是各个进程专有的,而程序可以由若干进程的描述信息和控制信息,是进程动态特性的集中反映,是系统对进程进行识别和控制的依据.在 Unix 中用 fork 创建子进程,如创建成功,则父进程与子进程处于不同的位置空间,子进程创建过程实质是产生一个父进程的副本,子进程的 proc 结构(即 PCB 中的一部分)相对于父进程的 proc 结构的个别项(如增加了父进程名、修改了该子进程产生的时间等等)作了变动.这种方式不但增加了 CPU 的开销,而且消耗了大量的内存空间.显然,要提高系统效率,实现系统的并行机制,用进程这个概念来实现是不可取的,必须要产生一个新概念,那就是线程.

关于线程尚没有一个严格而统一的定义,从程序角度来看,线程(Thread)是一段顺序指令序列.在引入线程的操作系统中,线程是进程中的一个实体,是被系统独立调度和分派的基本单位.通常一个进程都有若干个线程,至少也需要有一个线程.如果说在操作系统中引入进程的目的,是为了使多个程序并发执行,那么在操作系统中引入线程,则是为了减少程序并发执行时所应付出的时空开销,使操作系统具有更好的并发性.在传统的操作系统中,进程既是资源分配的基本单位,又是一个可以独立调度和分派的基本单位.作为资源分配的基本单位,不同的进程拥有各自独立的地址空间:代码段、数据段和栈,且拥有各自独立的资源:文件描述符表、进程表项、当前目录等.因而进程的上下文是相当庞大的,而以这种附带着庞大上下文的进程作为 CPU 调

① 收稿日期:2000－08－28.

作者简介:张利霞(1964～),女,河南洛阳人,河南师范大学讲师.

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

度的基本单位，系统在进程的创建、撤消和切换中要为之付出较大的时空开销，加重了系统管理进程的负担．使得它在开发程序的并行性方面有很大局限性，而这种局限性线程却能够很好地解决．通过把进程分解为线程，进程仍是资源分配的基本单位，使得线程成为 CPU 调度的基本单位，而线程只拥有一点在运行中必不可少的资源（如程序计数器、一组寄存器和栈）．同时进程的各线程共享地址空间及其它资源，这样使得线程上下文很轻，减轻了系统管理的负担．能够对开发并行性提供强有力的支持．现总结进程和线程间区别如下：

（1）调度．在传统的操作系统中，拥有资源的基本单位和独立调度，分派的基本单位都是进程．而在引入线程的系统中，则把线程作为调度和分派的基本单位，而把进程作为资源拥有的基本单位，使传统进程的两个属性分开，线程便能轻装运行，从而可显著地提高系统的并发程度．在同一进程中，线程的切换不会引起进程切换，在由一个进程的线程切换到另一进程中的线程时，将会引起进程切换．

（2）并发性．在引入线程的系统中，不仅进程之间可以并发执引，而且在一个进程中的多个线程之间，亦可并发执行，从而使系统具有更好的并发性．

（3）拥有资源．不论是传统的操作系统，还是设有线程的操作系统，进程都是拥有资源的一个独立单位，它可以拥有自己的资源．一般地说，线程自己不拥有系统资源（也有一点必不可少的资源），但它可以访问其隶属进程的资源．亦即，一个进程的代码段、数据段以及系统资源等可供同一进程的其它所有线程共享．

（4）系统开销．在一个已存在的进程中创建或撤销一个线程比创建或撤销一个新进程开销更小．类似地，在进行进程切换时，涉及到整个当前进程 CPU 环境的保存以及新被调度运引的进程的 CPU 环境的设置．而线程切换只须保存和设置少量寄存器的内容，并不涉及存储器管理方面的操作．因此进程切换的开销远大于线程切换的开销．此外，由于同一进程中的多个线程具有相同的地址空间，致使它们之间的同步和通信的实现，也变得比较容易．

2　多线程的实现

由于线程的种种优点，操作系统、程序设计语言、运行库等都对多线程加以支持，线程已在许多系统中实现，但实现的方法各有差异．

（1）从支持多线程的层次角度上看，主要有用户级多线程实现、内核级多线程实现、混合多线程实现．

①用户级多线程实现

没有操作系统内核的支持，完全在用户级提供一个库程序来实现多线程管理，这些库提供了创建、同步、调度与管理线程的所有功能，无需操作系统特别支持，多线库实现的多线程分时地在进程中运行，由多线库线程调度器负责线程的调度与切换，在内核看来，该进程运引着用户的程序而完全不知何时在用户多线程中切换．

一个由用户利用多线库编写的多任务程序，象一般的用户程序一样作为执行文件存放于外存中，在进程创建时由操作系统将程序由数据段映射到用户进程虚空间，并从初始入口开始执行，这时初始程序作为主线程在进程中运行，主线程的栈区沿用原来进程用户栈的约定空间．当需要产生新线程去运行并完成并行任务时，主线程程序安排一个多线库调用产生一个线程，并说明要执行的过程段或函数及初始数据，多线库会建立好新线程的所有管理表格，分配栈空间和线程私有存储空间，当用户线程将控制交给了多线库后，多线库会在处理完线程请求之后作线程调度，如果没有线程切换，则控制返回调用者线程，当需要进行线程切换时，多线库为原运行的线程保护好处理机运行现场，选取由它管理的一个就绪线程，并恢复新线程的处理机运行现场．

采用用户级多线库来实现多线程，操作系统不提供对多线程的支持，线程直接在用户级产生和管理，用户级线程的所有管理控制全部由多线库承担，所有线程的管理控制用表格都存于进程的用户空间，操作系统只感觉到进程存在，除在进程创建及安排初始程序执行时由操作系统过问外，进程在各线程的切换、产生等，操作系统全然不知，操作系统进程调度程序安排进程占用处理机，而不管多线程对进程的占用．存在于用户空间的多线库对线程的调度与操作系统进程调度程序对进程的调度是分开的，系统对进程的管理表格存于系统空间，多线库对线程的管理表格存于用户空间．由于不经过操作系统内核以及不占用操作系统资源，因此开销小，适合于开发应用程序中的细粒度的大量并行性，为用户提供一个简明的同步并行编程环境．当然使用多线库来实现用户级多线程也存在明显的缺点，即进程内的线程在进程外是不可见的，这些线程不能独立调度，当

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

其中一个线程阻塞时,就会造成整个进程的阻塞.如 Pthread、informix 就提供了用户级多线程.

②内核级多线程实现

内核级线程是由操作系统支持实现的线程,操作系统负责内核级线程的各种管理表格、负责线程在处理机上的调度和切换,用户层上无需内核级线程的管理代码,操作系统提供了一系列系统调用界面让用户程序请求操作系统作线程创建、撤销等.内核级线程用于运行并行执行的用户任务,当它被外部中断打断或自陷进入操作系统时,也运行操作系统内核程序.

当一个进程被创建时,系统同时创建一个内核级线程,用户初始程序即在该内核级线程序上运行,称之为主内核线程,当需要创建新的线程去运行并行任务时,主内核线程运行的程序安排一个线程创建系统,调用产生一个新的内核级线程,并说明要执行的过程段或函数及初始数据,同时也为该线程提供一个用户栈空间,线程的用户栈空间都在进程用户虚空间内,除第一个线程栈空间在线程初始化时按约定预留外,其它线程栈空间由用户态运行程序自行分配,内核建好内核级线程的线程控制块 TCB 等管理数据结构,为线程分配一个核心空间.线程在运行用户程序时在用户栈上工作,当中断或自陷进入操作系统核心运行时,则转到核心栈上工作.

采用内核级多线程的实现方法,操作系统内核必须为每个线程保存一系列描述信息,线程被用作处理机的执行单位,操作系统以线程为单位进行处理机分配,因此,内核级线程可以并行地在系统的不同处理机上运行,操作系统支持内核级线程与处理器之间的捆绑,让用户干预内核级线程选定处理机运行,当内核级线程从用户程序转操作系统核心程序运行后,如有 I/O 请求等引起线程阻塞,或者有更高优先级的线程可以运行,操作系统线程调度程序会作线程切换处理.由于线程的产生和管理经过操作系统内核,并且要占作操作系统资源,这就限制了一个系统中的内核级线程的数目,因此它适合于开发粗粒度的并行性,但是与用户级线程相比,它在进程外是可见的,可独立参与调度,从而避免了用户级多线程中一个线程阻塞导致整个进程阻塞的情形.如:Windows 95/NT,Mach、os/2 都提供内核级多线程.

③混合多线程实现

用户级线程和内核级线程实现方式各有优缺点,现代系统中通常提供混合多线程的实现方式.这样既有利于用户编写并行程序又能够最大限度地发挥多处理机的并行性.图 1 给出了混合实现方式示意图.这是一个两层结构,下面一层由内核提供内核级线程,上面一层由库程序提供用户级线程,中间轻型进程(LWP)是内核级线程支持用户级线程的接口.

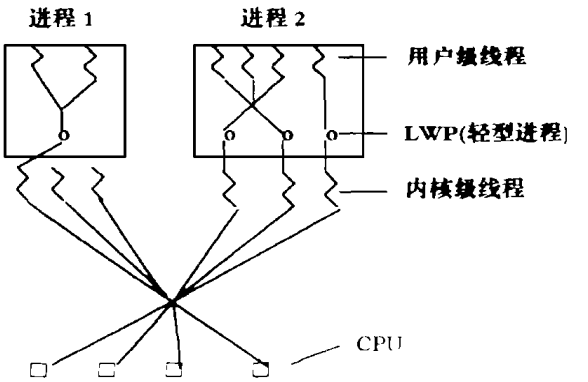

图 1 混合多线程实现示意图

用户利用多线库函数编写的并行程序作为用户执行文件,在进程创建时,被映射到进程用户虚空间中,第一个内核级线程随进程同时被创建,用户程序的主函数在该线程上运行,当需要时用户程序可以调用多线库创建新的用户级线程,也可以请求多线库到内核去申请更多的内核级线程,每当多线库申请创建一个内核级线程,多线库让该内核级线程运行及作好相关的初始化工作后即转到用户线程调度程序运行,多线库调度程序即可以再调度用户级线程,将控制交给被调用户线程.内核支持多个内核级线程的并发执行.这些内核级线程都执行内核代码,访问内核数据结构,它们共享同一地址空间,相互之间切换开销小.而且,内核还提供了仅供内核线程内部使用的同步原语,保证数据结构的完整性.内核级线程由内核进行调度,它选择优先权高的线程到某个 CPU 上运行.基于内核线程,内核通过轻量级进程来实现用户线程,这些内核线程都有一个 LWP 与之对立.从上面一层,同一进程的用户线程来共享地址空间,库提供了用户线程控制、通讯、同步机制,尽可能地减少系统调用,以减小开销.另外,库还对用户线程进行调度,选择优先权高的用户线程到某个 LWP 上运行,这时 LWP 可看作是虚拟的 CPU.

采用混合多线程实现方式,用户利用多线库产生足够的用户级线程.内核级线程的操作系统调用界面只提供给多线库使用,用户可以完全不知多线库使用了几个内核级线程,也可以通过多线库的接口申请更多的

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

内核级线程，所有内核级线程都由多线库管理使用，由多线库选择用户级线程在哪一个内核级线程上运行. 由于内核和用户级同时提供对多线程的支持，用户运行库调度用户线程映射到内核线程，核心调度内核线程执行，它同时具备了用户级和内核级多线程的优点. 但是，由于这两级的相互独立性会导致调度冲突，在一定程序上降低了系统的性能. 提供混合多线程的有 SoLaris2.x 等.

(2)从应用编程接口角度看，主要有：

①JAVA 线程接口. JAVA 是一种针对跨平台和基于 WWW 应用而开发的新型语言，它内部集成了对多线程的支持. 其线程由 3 部分组成：虚拟 CPU，封装在 Java. lang. Thread 类中；CPU 所执行的代码，传送给 Thread 类；CPU 所处理的数据，传送给 Thread 类. JAVA 语言内部提供了 Runnable 接口实线多线程. 其实现方法如下：在 Thread 的构造函数中，有一个带 Runnable 参数的构造函数，只要先产生一个定义好的具有 Runnable 接口的对象，然后再将其对象当成参数传给 Runnable 产生一个 Thread 的对象，那么就可通过该 Thread 对象来得到 Thread 所提供的所有功能.

②POSIX 线程接口. 1995 年制定的 POSIX 100. 1C 标准，定义了可移植的线程程序设计接口，但它没有描述底层设计机制，这样不论是在用户级还是内核级提供多种程支持，或是两级都支持，或是使用第三方开发多线程软件包，只要遵循这个标准，就能保证应用程序的可在多线程方面的移植性.

③MICROSOFT 线程接口. 微软 Windows 95/NT 提供的线程接口，它与 POSIX 接口有很大区别.

3　结束语

目前，线程机制已经成为并发程序设计语言和操作系统的主要特征. 在共享主存多处理机系统的软件设计中更是如此，如 C/C＋＋提供了用户级的线程库，而 Java 是用类来生成和同步线程. 多线程编译系统和多线程操作系统已经成为系统软件发展的主流. 但是多线程结构比原来的结构更加复杂，复杂性涵盖了硬件现场规模、现场切换机制、调度算法等软硬件的诸多方面. 如何既保持各级多线程的优点，又提高系统性能，围绕线程的管理、通讯和同步等方面的研究还在继续.

参　考　文　献

1　汤子瀛等. 计算机操作系统[M]. 西安：西字电子科技大学出版社，1996. 52～56

2　Kavi K. Multithreading Implementations[M]. Technical Report，1998. 1～125

3　Maurice J. Bach. 陈葆钰等译. UNIX 操作系统设计[M]. 北京：北京大学出版社，1989. 18～107

4　董慧等. JAVA 实用编程例解[M]. 北京：电子工业出版社，1999. 45～107

Implementation Method of Multithreading

ZHANG Li-xia

(Department of Computer Science，Henan Normal University，Henan Xinxiang，453002)

Abstract：This paper analyzes the concept of multithreading，describes several kinds of multithreading implementation methods in details and its implementation differences.

Key words：process；thread；multithread；implementation

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net