**Windows操作系统多核CPU内核线程管理方法**

摘要:Windows 是采用[CPU](http://product.dzsc.com/product/searchfile/5590.html" \t "_blank) 时间片轮转多任务分配机制的非实时操作系统，无法满足实时性要求比较高的任务需要。而随着[CPU](http://wiki.dzsc.com/info/2456.html" \t "_blank)技 术的快速发展，当前市场上双核甚至多核CPU 已成为主流，这使得在Windows 平台上用多核CPU 的某些核独立执行任务，从而有效地提高软件的实时性成为可能。本文介绍一种在Windows 内核模式下编写Windows 驱动程序，有效分配多核CPU 资源从而提高软件实时性的方法。

**1 引言**

　　本文分析了Windows 系统的进程调度机制,并设计了一种基于Windows 操作系统内核驱动的多核CPU 线程管理方法,实现了一个基于Windows 内核驱动的线程管理服务系统，它能让用户根据每一个任务线程对CPU 资源的需要程度和对实时性的要求，在多核CPU上合理为线程分配CPU 核。

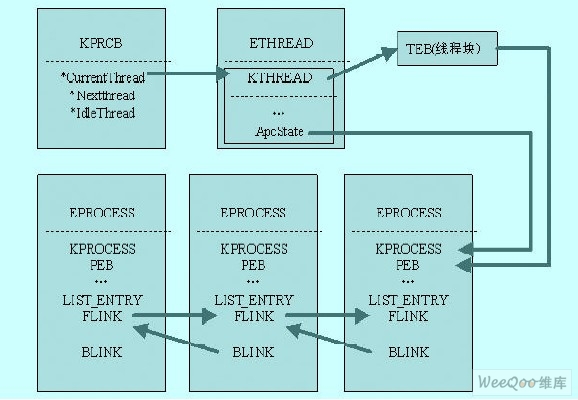


图1 Windows 内核调度结构体关系图

**2 Windows 系统的进程调度方法分析**

　　Windows NT 中的每一个进程都是EPROCESS 结构体。此结构体中除了进程的属性之外还引用了其它一些与实现进程紧密相关的结构体。例如，每个进程都有一个或几个线程,线程在系统中就是ETHREAD 结构体。简要描述一下存在于这个结构体中的主要的信息,这些信息都是由对内核函数的研究而得知的。首先，结构体中有KPROCESS 结构体,这个结构体中又有指向这些进程的内核线程(KTHREAD)链表的指针(分配地址空间),基优先级,在内核模式或是用户模式执行进程的线程的时 间,[处理器](http://product.dzsc.com/product/searchfile/8213.html)affini ty(掩码，定义了哪个处理器能执行进程的线程)，时间片值。在ETHREAD 结构体中还存在着这样的信息:进程ID、父进程ID、进程映象名。

　　在E P R O C E S S 结构体中还有指向P E B 的指针。

　　ETHREAD 结构体还包含有创建时间和退出时间、进程ID 和指向EPROCESS 的指针,启动地址，I/O 请求链表和KTHREAD 结构体。在KTHREAD 中包含有以下信息:内核模式和用户模式线程的创建时间，指向内核堆栈基址和顶点的指针、指向服务表的指针、基优先级与当前优先级、指向APC 的指针和指向T E B 的指针。

　　KTHREAD 中包含有许多其它的数据,通过观察这些数据可以分析出KTHREAD 的结构。图1 描述了这些结构体之间的关系。

　　通过遍历KPROCESS 结构体中的ETHREAD,找到系统中当前所有的KTHREAD 结构,这个结构中的偏移量为0x124 处的Affinity 域(Windows XP sp3)即为设置CPU 亲缘性掩码的[内存](http://product.dzsc.com/product/searchfile/4072.html" \t "_blank)地 址。在此重点解释CPU 亲缘性的概念,CPU 亲缘性就是指在系统中能够将一个或多个进程或线程绑定到一个或多个处理器上运行,这是期待已久的特性。也就是说:“ 在1号处理器上一直运行该程序”或者是“在所有的处理器上运行这些程序，而不是在0 号处理器上运行”。然后, 调度器将遵循该规则，程序仅仅运行在允许的处理器上。在Windows 操作系统上，给程序员设定CPU 亲缘性的[接口](http://product.dzsc.com/product/searchfile/5540.html" \t "_blank)是 用一个32 位的双字型数表示的, 它被称为亲缘性掩码(Affinity bitMask)。亲缘性掩码是一系列的二进制位，每一位代表一个CPU 单元是否可执行当前任务。例如一个在具有四个CPU 的PC 机上( 或四核CPU) ,亲缘性掩码的形式的二进制数如下式所示:

　　0000000000000000000000000000XXXXB

　　其中自右向左，每一位代表0 到31 号CPU是否可用，由于本机只有四个CPU, 所以只有前四个位可用,X 为1则代表当前任务可执行在此位代表的CPU 上，X 为0 则代表当前任务不可执行在此位代表的CPU 上, 例如:

　　00000000000000000000000000000010B

　　代表当前任务只能执行在1 号 CPU 上(CPU 下标记数从0 开始),又如0x00000004 代表当前任务只能执行在2 号CPU 上,0x00000003 代表当前任务可以运行在0号和1 号CPU 上。

　　Windows 的进程调度代码是在它的[System](http://www.dzsc.com/s/SYSTEM.html" \t "_blank) 进程下的，所以它不属于任何用户进程上下文。调度代码在适当的时机会切换进程上下文,这里的切换进程上下文是指进程环境的切换, 包括[内存](http://wiki.dzsc.com/info/1471.html" \t "_blank)中 的可执行程序, 提供程序运行的各种资源.进程拥有虚拟的地址空间,可执行代码, 数据, 对象句柄集, 环境变量, 基础优先级, 以及最大最小工作集等的切换。而Windows 最小的调度单位是线程, 只有线程才是真正的执行体，进程只是线程的容器。Windows 的调度程序在时间片到期，或有切换线程指令执行(如Sleep,KeWaitForSingleObject 等函数)时, 将会从进程线程队列中找到下一个要调度的线程执行体，并装入到KPCR(Kernel ' s Processor Contr ol Re g i o n , 内核进程控制区域) 结构中,CPU 根据KPCR 结构中的KPRCB 结构执行线程执行体代码。而在多核CPU 下，当Windows 调度代码执行时,从当前要调度执行的KTHREAD 结构中取出Affinity,并与当前PC 机上的硬件配置数据中的CPU 掩码作与操作,结果写入到指定的CPU，例如双核CPU 的设备掩码为0x03,如果当前KTHREAD 里的Affinity 为0x01,那么0x01&0x03=0x01,这样执行体线程会被装入CPU1的KPRCB 结构中得以执行,调度程序不会把这个线程交给CPU2 去执行。此过程如图2 所示。这就是为线程选择指定CPU 核的原理。

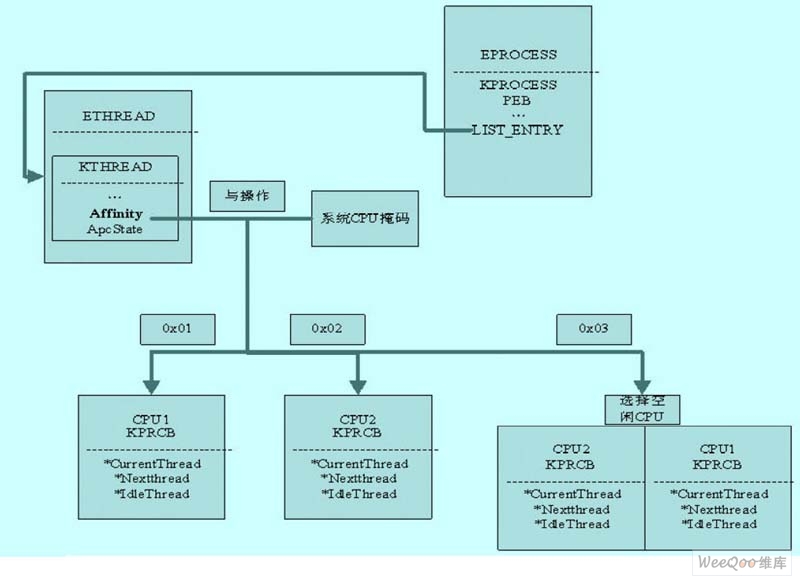


图 2 Windows 内核亲缘性调度原理图。

　　那么控制线程在指定CPU 上运行的突破口就是修改Windows 内核结构体KTHREAD 下的Affinity 域。然而Windows 内核结构被放在虚拟内存线性地址的高2G(不同版本Windows 下也可能是1G)地址空间,用户模式下的应用程序是无法访问这段内存空间的,所以必须编写Windows 驱动程序,来访问Windows 内核内存空间, 这也是本文将要描述的重点。

**3 线程管理服务系统**

　　整个系统的结构如图3 所示。该系统由两大部分组成，分别是内核模式下的管理服务系统设备驱动程序，和用户模式下的管理服务系统应用程序。管理服务系统应用程序通过调用 Win32 子系统API,向内核下的管理服务系统驱动程序传递IRP,内核收到IRP 后,跟据收到的IRP 的内部信息，执行相应的派遣函数,对相应内存进行读写,从而给管理服务系统应用程序提供可用的系统信息。

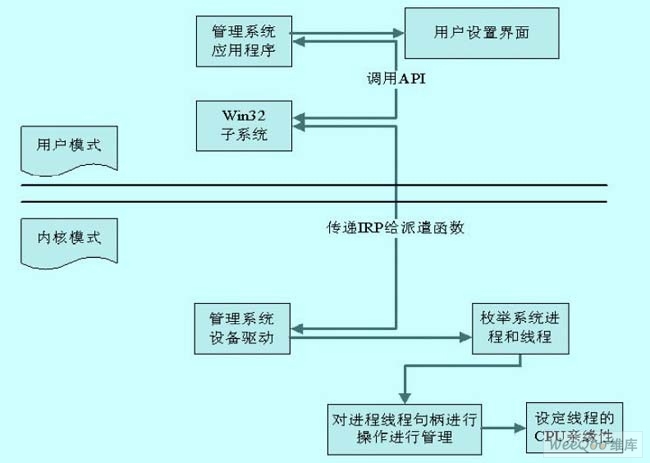


图3 管理系统总体结构图。

**3.1 内核模式下读取系统信息**

　　线程管理服务系统驱动程序中，读取系统信息的方法用到了微软没有公开文档的内核服务函数,ZwQuerySystemInform[ATI](http://ati.dzsc.com/)on， 这个函数被封装在ntdll.dll模块中,通过链接ntdll.lib 可得到此函数地址。通过一个枚举量SystemProcessInformation 来得到进程线程相关信息，填入到第二个输入参数SYSTEM\_PROCESS\_INFORMATION结构中, 这样就获得了当前系统关于进程线程的信息。

**3.2 内核模式下枚举系统进程线程**

　　SYSTEM\_PROCESS\_INFORMATION结构中存储了进程及其线程的所有相关信息，表1 列出了它的具体内容，包括结构内域的地址偏移, 数据类型和描述。

　　SYSTEM\_PROCESS\_INFORMATION的第一个DWORD型是下一个进程 SYSTEM\_PROCESS\_INFORMATION相对于当前结构地址的偏移量，可以通过地址偏移来遍历所有的进程结构，当遇到某一个进程结构的0 x 0 0 0 0 处的DWORD 型值为0 时，说明这个结构体是系统内最后一个结构体。线程管理服务在它的派遣函数中通过这种方式遍历所有进程，从中提取有用的信息，填入两个自定义结构体中。如图 4 所示，描述了一个具有两个线程的进程的数据结构，首先在MY\_PROCESS\_INFO 结构中填入进程的相关信息，然后根据此进程所有的线程数，向系统申请足够大的分页内存空间，PVOID 型指针指向的是第一个线程结构所在的地址空间，然后向线程结构体中\_MY\_THREAD\_INFO 中填入线程信息，再由线程结构体中的PVOID 型指针指向第二个线程结构体所在的地址空间，以此类推，最后一个线程结构体的PVOID型指针指向[NULL](http://www.dzsc.com/n/NULL.html" \t "_blank)。这样一个过程描述了一个进程及其所属的所有线程的枚举过程，通过对所有进程的遍历，可以得到系统中的一个完整的进程线程表，存在一段分页内存中，这样在应用程序中便可以得到这些信息。

表1 SYSTEM\_PROCESS\_INFORMATION 结构



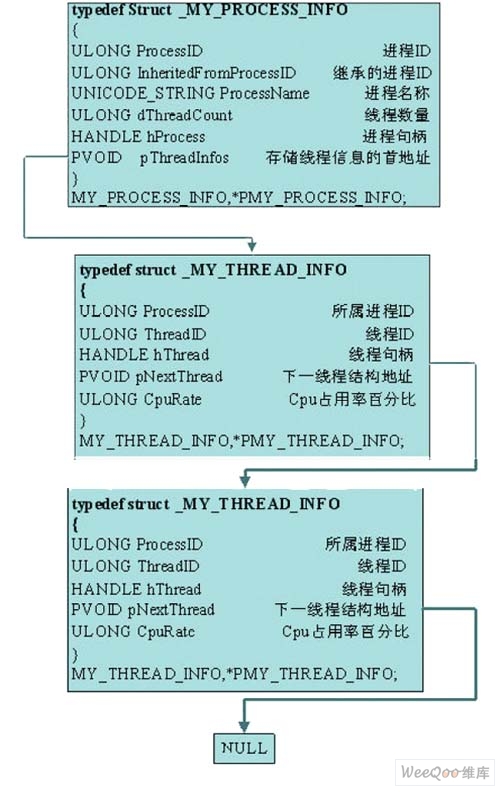


图4 进程线程的两种数据结构。

**3.3 线程管理服务系统应用程序设计**

　　进程管理服务系统应用程序是要通过调用Win 32子系统的API 函数DeviceIoControl 来向线程管理服务系统驱动程序发送IRP 的，然后在IRP 结束之后把驱动程序中读出的所有有用进程线程信息填入到指定的内存中。这样线程管理服务系统应用程序就可以根据所获得的系统信息句柄来对线程CPU 亲缘性属性进行设置。首先为DeviceIoControl 中的InputBuffer 申请一段内存空间传入给驱动程序，驱动程序读取内核空间进程线程信息写入到这段内存中，应用程序读到信息并显示给用户。

　　在系统中应用程序为每一个CPU 维护一个结构体，内容包括该CPU 是否运行实时线程，该CPU 上运行的线程数（如果是实时线程CPU线程数为1），以及在此CPU上运行的线程结构数组的首地址。系统通过对此CPU 结构数组的解析来对线程进行管理。并通过DeviceIoControl函数把设置后的CPU 结构交给驱动程序内核。

**3.4 修改Windows 内核结构体**

　　在驱动程序读回应用程序下用户的设置结果后，就需要按照用户的设定修改KTHREAD 下的Affinity 域的掩码值了。首先要找到KTHREAD 的线性内存空间，PsGetCurrentProcess()内核函数可以返回内核下当前进程空间的E P R O C E S S 结构。E P R O C E S S 结构下的ActiveProcessLinks 域是LIST\_ENTRY 结构，通过它可以遍历所有的ETHREAD 结构，那么那到KTHREAD 下的Affinity 域就不难了，可以使用两个循环嵌套来得到所有线程的Affinity 域并将其值设为应用程序中用户的设定值。线程CPU 掩码就被成功的修改了。当CPU 被设定为运行实时线程的CPU 时,在它上面运行的线程只能是一个实时线程，这时的运行线程数被设定为1; 当CPU被设定为非实时线程的时候，上面有可能除了任务线程运行之外,还有Windows 系统进程下的线程。

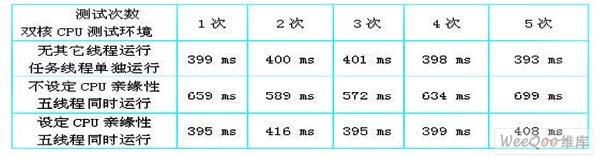
**4 软件使用及性能测试**

**4.1 驱动的加载及软件的使用**

　　首先需要把本系统的驱动sys 文件加载到Windows的服务管理器中，加载成功后打开应用程序，用户可以通过应用程序中显示出的当前系统内的进程和线程进行选择,并在GUI 图形界面中对其CPU 占用率及CPU亲缘性进行设置。

**4.2 设置 CPU 亲缘性测试**

　　测试运行在双核CPU 的PC 机上，系统运行一个要测试的任务线程(任务线程为一个108 次加法运算),四个其它线程(为测试方便，设为while 循环线程),限定了循环线程的CPU 亲缘性掩码为0x0001,任务线程的CPU亲缘性为0x0002,这样任务线程与其它线程分别在两个核上运行，分别测试了任务线程单独运行，任务线程与 其它线程不设定CPU 亲缘性，任务线程与其它线程设定CPU 亲缘性三种情况下下任务线程的运行总时间如表2 所示。



　　从表2 分析, 设定任务线程的CPU 亲缘性与其它线程所占用的CPU 分开，真正意义上的实现了任务的异步执行,非常有效的提高了实时线程对CPU 资源的使用率。

**5 结束语**

　　本文分析了Windows 系统的内核进程线程调度表2CPU 亲缘性设定三种情况下任务线程运行时间表机制，并在此基础上设计了一种基于Windows 操作系统内核驱动的多核CPU 线程管理方法, 实现了这样一个软件系统。首先在Windows 内核层获取系统进程线程信息，然后再把信息传入应用层，由应用层上的应用程序根据获取的信息句柄,对进程进行操作,用户在图形界面下按照仿真任务对CPU 资源的不同需求,进行相应的设置，可以为指定线程设置CPU 亲缘性的功能。在一定程度上为Windows 系统下的任务合理地分配了CPU 资源，为对实时性要求较高的任务提供了一个可靠的运行环境。

  来源:[ROSE](http://bbs.dzsc.com/space/viewspacepost.aspx?postid=87508)