第34卷 第10期 计 算 机 工 程 2008年5月
Vol.34 No.10 Computer Engineering May 2008
·软件技术与数据库· 文章编号：1000—3428(2008)10—0078—02 文献标识码：A 中图分类号：TP311;TP391

面向用户的进程调度策略研究与实现

陈 媛[1]，杨 武[2]
(1. 重庆工学院计算机科学与工程学院，重庆 400054；2. 重庆工学院工商管理学院，重庆 400054)

摘 要：针对Linux进程调度策略注重系统性能而忽视用户服务的问题，提出一种面向用户的进程调度策略，根据各用户拥有进程数量上的差别，分别调用Linux原有动态优先调度策略、公平共享法调度策略和自适应公平共享法调度策略，以达到最大限度地满足整个系统各用户需求的目的，实现进程调度公平性原则。在实现过程中，解决了决策计算的问题，引入动态权限调整机制实现了自适应公平共享算法。实验结果表明，该进程调度策略是有效的，可应用于注重用户满意度的分时系统中。

关键词：进程调度策略；面向用户；自适应公平共享法

Study and Realization of Process Scheduling Policy Facing User

CHEN Yuan[1], YANG Wu[2]
(1. Department of Computer Science and Engineering, Chongqing Institute of Technology, Chongqing 400054;
2. Department of Business Administration, Chongqing Institute of Technology, Chongqing 400054)

【Abstract】 Aiming at the problem of Linux process scheduling policy just regards the system capability and neglects user service, the paper presents a kind of process scheduling policy facing user according to the discrepancy of process quantity every user owned, which invokes Linux dynamic priority scheduling policy, fair share scheduling policy and self-adapt fair share scheduling policy in order to meet the every user's need in whole system to most degree and realize the fair principle of process scheduling. In the realization process, it settles the problem of decision-making account and introduces the dynamic right adjusting mechanism to realize self-adapting fair share arithmetic. The experimental result indicates that the process scheduling policy is effective and valuable applying on the time-sharing system which regards user's contentment degree.

【Key words】 process scheduling policy; facing user; self-adapt fair share

Linux是典型的多用户多任务操作系统，允许多个用户通过各自的终端同时使用一台主机，共享主机的各类资源和服务，每个用户又可创建多个任务(进程)，使它们并发执行[1]。用户所关心的不是进程如何执行，而是构成应用程序的一组进程是如何执行的。即使每一个用户用一个进程表示，这个概念也可以扩展到用户组。例如，在分时系统中，可能希望把某个部分的所有用户看作是同一个组中的成员，然后进行调度决策，并给每个组中的用户提供类似的服务。如果同一个部门中的许多用户登录到系统，则希望响应时间的降级只会影响这个部门的成员，而不会影响其他部门的用户。因此，基于进程组(用户)的调度决策是非常具有吸引力的。但目前Linux的进程调度都是从单个进程的角度进行讨论，调度策略的重点是高效地利用处理机，注重了系统性能而忽视了提供给用户的服务。针对Linux调度策略的这一问题，本文提出一种面向用户的进程调度策略。

1 面向用户的进程调度策略

Bach于1986年提出了一种面向用户的公平共享算法，以用户为单位平均分配CPU的占用时间，综合考虑了进程的基础优先级、进程近期使用处理器的情况以及进程所属用户的权值，来确定该进程的优先级[2-3]。在用户拥有进程数相差很大时，该算法可以有效地提高调度的公平性。但在用户拥有的进程数相差不大时，该算法就会因为多余的循环计算而使系统效率下降。为此，本文提出了一种新的面向用户的进程调度策略，它在使用公平分享算法的基础上，综合考虑各用户拥有进程数量上的差别，分别采用不同的调度算法，以提高整体满意度：

(1)各用户拥有进程数相当时，采用Linux原有动态优先调度策略。因为公平分享算法是将CPU时间按照用户等分的，则每个用户将分到大致相同的CPU时间，与按照以动态时间片为优先级的Linux调度策略相比，调度效果基本上一致。如果此时仍然采用公平共享法，那么执行算法所进行的优先级计算，将是冗余的，浪费了系统资源。

(2)各用户拥有进程数有一定的差别，但差别在系统的规定值之内时，采用公平共享调度策略。公平分享算法照顾了拥有进程数量少的用户，给该用户分配了相同的CPU时间，在这种情况下该算法才真正发挥其基于用户的相对公平的作用。

(3)用户之间的进程数相差很大时，采用自适应公平共享法调度策略。公平共享调度策略分配给每个用户是等量(或相对比例)的CPU时间，那么会出现拥有进程数量多的用户CPU紧张，而进程数少的用户CPU会有所浪费的现象，它只考虑这些用户之间的公平分享，即局部用户满意度，没有考虑到整体用户满意度。为了避免这种现象的出现，本文提出自适应公平共享法调度策略。该策略根据用户使用处理机的情况，动态地调整用户的权限，以此调整该用户进程的优先级，从

基金项目：重庆市科委重点科技计划基金资助项目“网络工程及信息化软件关键技术”(2004CC12)
作者简介：陈 媛(1966－)，女，副教授、硕士，主研方向：计算机软件，嵌入式系统；杨 武，教授、博士
收稿日期：2007-05-25 **E-mail：**cy@cqit.edu.cn

而调整进程占据处理机的可能性。当调度结束后，将动态权限归零，仍参照原来的静态权限参与调度。

2 面向用户的进程调度策略的实现

面向用户的进程调度策略的实现流程如图 1 所示。

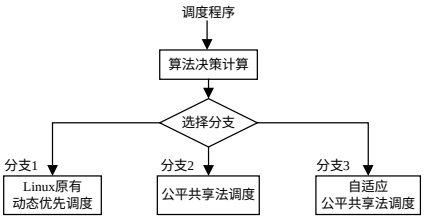

图 1 调度程序流程

2.1 算法决策的计算

根据公平分享算法的思想[4]，在实现时，用$P_j(i)$表示用户k中进程j在调度区间i开始时的优先级(值小表示优先级高)；$CPU_j(i)$表示进程j在调度区间i中CPU使用时间；$GCPU_k(i)$表示用户k在调度区间i中CPU使用时间；$Base_j$表示进程j原有优先级；W_k表示用户k的权值，则按式(1)计算$P_j(i)$的值[5]：

$$P_j(i)=Base_j+\frac{CPU_j(i-1)}{2}+\frac{GCPU_k(i-1)}{4\times W_k} \quad (1)$$

分配给每个进程一个基本的优先级。当一个进程使用处理器时以及当该进程所在的用户使用处理器时，该进程的优先级会降低。对于进程组使用的情况，用平均值除以该组的权值进行标准化。分配给组的权值越大，使用处理器对其优先级的影响就越小。

为了简化计算，在实现时采用式(2)计算$P_j(i)$。

$$P_j(i)=Base_j+\frac{CPU_j(i-1)}{2}+\frac{GCPU_k(i-1)\times Nr\_user}{4} \quad (2)$$

其中，N_r_user表示系统中目前的用户数。

任何一个调度算法都不需要绝对的公平调度，从用户相对公平入手，利用用户的进程数量N_k和用户可用CPU时间T_k来计算决策因子$F_k=T_k/W_k/N_k$。

从T_k和N_k的辨正关系可以得知，对于用户拥有进程数相当这种情况的F_k必定为一个中间值，而相差一定和相差较大的F_k值在其左右，因此可以使用该值进行调度算法的决策。若F为决策因子稳定值，用户拥有进程数相差一定和相差较大的F_k值对稳定值的最大允许偏离程度为：E_2和E_3(经验值)，则：

当$|F_k-F|\le\min(E_2, E_3)$时选择分支 1；当$F_k+E_2\le F$时选择分支 2；当$F_k-E_3\ge F$时选择分支 3，如图 1 所示。

2.2 自适应公平分享算法

在用户中设两个字段分别记录该用户的静态权限和动态权限。自适应公平分享算法的执行流程如图 2 所示。

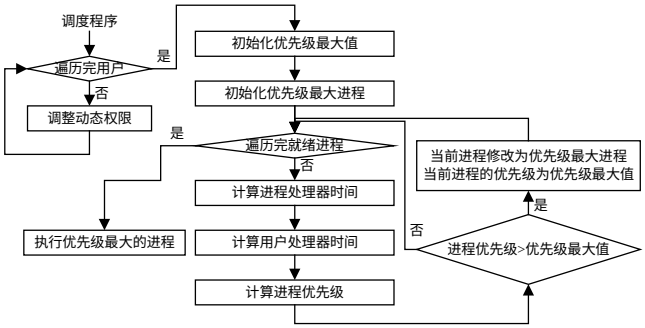

图 2 自适应公平分享算法流程

2.3 在 Linux 内核中实现面向用户的进程调度策略

在Linux进程调度中每个进程都有一个task_struct结 构[6]，task_struct在内核中的形式就是通常所说的PCB，它是对进程控制的唯一也是最有效的手段。计算进程当前优先级的是goodness()函数。真正执行进程调度的函数是schedule()。要在Linux中实现面向用户的进程调度，必须修改的部分有kernel的部分代码、task_struct结构和schedule以及goodness函数。算法实现步骤如下：

步骤 1 数据结构数据成员的修改。

在 PCB 中增加基础优先级、区间 CPU 时间成员(文件目录 include/linux/sched.h)：

```
Struct task_struct{
…
unsigned long fss_base_prio;//基础优先级(根据进程性质定为 60 //300 500)
unsigned long fss_cpu_ticks;//该进程区间 CPU 时间(初始值为 0)
…}
```

在用户结构中添加数据成员(文件目录 include/linux/sched.h)：

```
struct user_struct {
    atomic_t __count; //系统中该用户的使用计数
    atomic_t processes; //该用户拥有地进程数量
    atomic_t files; //该用户打开的文件数量
    struct user_struct *next, **pprev; //哈希表中后继和前驱 //指针
    uid_t uid; //系统分配的唯一的用户标志号
    struct user_struct *f_next,*f_prev; //用户结构双链表的后继 //和前驱指针
    unsigned long gcpu_ticks; //该用户区间 CPU 时间
    }
```

步骤 2 用户数据结构操作函数的修改。

修改 kernel/user.c 文件中的锁机制、uid_hash_insert()函数、uid_hash_remove()函数、alloc_uid()函数、free_uid()函数。

步骤 3 修改进程创建函数 do_fork()。

步骤 4 区间 CPU 时间的增加：kernel/time.c。

硬件每发出一次时钟中断，中断服务程序就要调用 do_timer() 函数来处理系统时间。在该函数中又调用 update_process_time 来递减动态优先级，使得其他进程有被调度的可能。

步骤 5 修改进程调度 schedule()和 goodness()函数。

3 实验

本文是在 Linux2.4.29 内核的基础上修改实现的，在 CPU：1.8 GHz，内存：512 MB 的单 CPU 的 x86 主机上直接对内核进行编译、调试和安装，最后进行了实验。实验中，分别在原内核上和修改内核后，7 个终端用户分别在表 1 所示的 3 种进程数情况下运行进程，这些进程完成相同的工作。在情况 1 下修改内核平均等待时间无明显改变，在情况 2 和在情况 3 下实验结果如图 3 所示。

表 1 各用户运行进程数

	用户 1	用户 2	用户 3	用户 4	用户 5	用户 6	用户 7
情况 1	100	100	100	100	100	100	100
情况 2	85	90	95	100	105	110	115
情况 3	5	40	75	100	125	160	195

(下转第 82 页)