

嵌入式 Linux2.6 的实时性能研究与优化

李少雪, 吾守尔·斯拉木, 郭凯敏

(新疆大学 信息科学与工程学院, 新疆 乌鲁木齐 830046)

摘 要: 以 Linux2.6 内核为对象, 对 Linux 系统实时性进行了研究. 针对 Linux 中时钟粒度过大、临界区的执行造成系统反应延迟的不确定性和进程调度算法在任务调度过程中的公平性体现不够等, 提出根据实时应用的特点设计了一个并行高精度定时器以提高时钟精度; 采用自旋锁双模块控制方式对临界区进行管理以减少时延; 修改 0(1) 调度算法的双队列操作, 同时增加时间片轮询装置和采用动态计算通用进程的优先级和时间片的方法以提高调度的公平性和系统快速响应时间. 实验测试表明: 改进方案有效提高了系统的实时性, 扩充了 Linux 在嵌入式领域的应用性.

关键词: linux2.6; 嵌入式系统; 实时性; 内核; 进程调度

中图分类号: TP316

文献标识码: A

文章编号: 1000-7180(2010)08-0210-04

Research and Optimization on Real-Time Performance of Embedded Linux2.6 System

LI Shao-xue, Wushour. Silamu, GUO Kai-min

(Institute of Information Science and Engineering, Xinjiang University, Urumqi 830046, China)

Abstract: The paper analyzes the real time performance of Linux2.6 kernel, because the key restrict factors which are clock, critical section and process scheduling, a new method called high-resolution timer is designed and implemented, spin-lock twin module-control method for critical section be adopted, and modify the 0(1) scheduling algorithm, all which are in order to improve the real-time performance of the Linux system efficiently. The test result indicates the improvements are successful to improve the real-time performance of the Linux system, and the application range in embedded field is expanded.

Key words: Linux2.6; embedded system; real-time; kernel; process scheduling

1 引言

Linux 自问世至今, 短短十几年间以发展成为一个功能强大且设计较完善的操作系统. 并且 Linux 遵循 GPL, 开放源代码, 具有丰富的软件资源, 同时拥有功能强大的内核, 性能高效、稳定、易于剪裁以及支持多种体系结构^[1]和能适应很多硬件平台等优点, 使其在嵌入式系统领域得到了突飞猛进的发展. 然而 Linux 系统本身只是一个通用的分时系统, 在设计之初是以高的吞吐量和公平性为追求目标, 基本上没有考虑系统在实时应用方面的时间约

束需求, 虽然后来在实时性方面有了很大的改进, 但仍不能满足嵌入式系统在实时性方面的广泛需求. 文中基于 Linux2.6 内核, 从时钟精度、临界区访问、调度算法三方面对其实时性能的影响进行分析并提出改进方案.

2 Linux2.6 内核对实时性能的影响

2.1 时钟粒度

Linux2.6 内核采用 jiffies 计数的时钟, 在每次时钟中断上半部, 给 jiffies_64 变量增加 1, 更新墙上时间 xtime; 在时钟中断下半部, Linux 采用 softirq

收稿日期: 2010-05-13; 修回日期: 2010-06-20

基金项目: 国家自然科学基金项目(60507005); 新疆教育厅高校科研计划科学研究重点项目(XJEDU2005I04)

软中断机制推迟中断处理,执行已到期的动态定时器.这个周期的时钟中断是导致低时间精度的一个重要原因.在频率为 1000Hz 时,也只能达到 1ms 的时钟粒度,其粒度还是较大.任务相应延迟,而实时应用一般都需要微秒甚至纳秒级的相应精度,显然其不能满足实时应用的需求.

2.2 临界区访问

在 Linux2.6 内核中虽采用插入了抢占点^[2]的技术,改善了系统的实时性.但是内核对临界区操作时采用中断屏蔽方式,中断服务程序的执行和进程在临界区中的执行都是不可抢占,从而使系统的行为难以预测,实时进程的时限不能得到保证,响应时间延迟.

2.3 调度算法

Linux2.6 采用 $O(1)$ 调度算法,与 Linux2.4 及以前版本的 $O(n)$ 调度算法相比,内核具有更好的实时性能. $O(1)$ 的含义是指:无论有多少进程,这个算法均可以在一个限定的时间内完成,该算法解决了以往 Linux 内核调度算法在任务负载过多时的不稳定性,同时也引入了很多新的特点^[3].但其作为实时系统的调度算法依然存在着许多缺点,如:以牺牲实时进程的响应时间换取任务调度的公平性;双队列在进行切换时,只有在活动队列中的所有进程都耗尽了自己的时间片时方可,在此期间所有的实时进程都无法被立即处理.另外进程的随机抢占策略,使得对实时进程的执行时间和响应时间都很难预测和控制,且该算法未根据普通进程的实际执行情况有针对性的动态分配时间片,从而使得系统资源利用率欠佳,快速响应时间延迟.

3 嵌入式 Linux2.6 实时性能的改进方案

根据上述对 Linux2.6 的实时性分析,下面针对时钟精度、临界区访问、调度算法三方面对嵌入式 Linux2.6 系统的实时性优化给出具体方案.

3.1 高精度定时器设计

Linux2.6 内核虽对时钟精度有所改进,但仍无法满足实时任务对微妙甚至纳秒级响应精度的需求.而简单地提高时钟频率,又会增加时钟中断的次数,CPU 的绝大部分时间都可能在处理时钟中断,造成不必要的开销.所以为了对实时任务进行高精度定时,在系统中增加一个与标准 Linux 核心时钟并行运行的具有高精度的核心实时时钟处理系统,同时增加一个新的定时器链表和一个实时中断处理程序,该链表中的每一项超时处理函数都由其对应

的中断服务来执行.普通时钟用于正常的周期性计时,而高精度时钟可以根据实际需要配置系统的时钟粒度,这样既满足实时需求,又减少不必要的系统开销.

由于实时时钟被频繁的使用,因而高效的时钟定时器队列处理能有效地改善时钟管理引起的瓶颈,而 Faboracci heaps^[4]是一种空间复杂度低,且高效的算法,因此采用 Faboracci heaps 的数据结构对实时定时器链表进行管理,能有效提高算法的执行效率,高精度定时器的数据结构如下:

```
typedef struct hrtimer{
    Fabo_heap_queue timer_queue; /* Faboracci heaps 队列 */
    tick_timer expires; /* 实时任务到期时间 */
    tick_timer period; /* 实时任务周期 */
    enum hrtimer_state state;
    int (*function)(void *); /* 回调函数 */
    void * data; /* 服务函数所带的参数 */
    struct list_head list; /* 到期高精度定时器链表 */
};
```

在系统中增加了一个高精度定时器 hrtimer 之后,便要维护两个与时钟有关的中断请求队列:系统时钟中断请求队列和 hrtimer 中断请求队列,它们分别对应各自的中断服务程序.如果系统中的设备驱动程序或应用程序利用了高精度定时器产生中断,那么系统便调用 hrtimer 服务程序.在加载 hrtimer 时,需要将它链入 timer_queue,所调用的内函数时为 enqueue_tmcb(),在每次时钟中断服务例程中,会将定时器的时间与当前时间比较,如果定时器时间小于或等于当前时间,则说明定时器已经到期,将其从 Faboracci heaps 队列中删除,放到一个挂起链表中,然后根据剩余的最早过期的 timer 设置事件源的下一个中断触发时间,在执行 run_local_timer()之后再激活 raise_softirq,使其执行该定时器的回调函数.

3.2 自旋锁双模块控制模型设计

在 Linux2.6 内核中,因同时存在实时和非实时任务,根据系统延迟^[5],那么假设任务 A 为普通任务,任务 B 为实时任务, $T_{execute}$ 为任务的执行时间, $T_{schedule}$ 为任务的调度时间,设 A 提出申请临界区在前,则 B 的延迟时间为 $ST(B) = T(A)_{execute} + T(B)_{schedule}$. 则设任意时刻 $T(A)_{execute} < T(B)_{schedule}$ 几率为 ρ ,则实时任务的平均延迟时间为: $SD(B) = \rho(T(B)_{schedule}) + (1 - \rho)(T(A)_{execute} - T(B)_{schedule})$,从而可知,实时任务的响应时

间的提高与 ρ 值能否提高有直接关联. 其次一般中断信号是为了那些时间不可预测的任务而设定的, 它的产生是随机的. 为了准确预测实时任务中断的开始时间 T_{real_timer} , 将中断信号的产生与 3.1 节提出的高精度时钟系统关联, 内核利用高精度时钟源记录自上一次时钟中断以来经过的时间, 从而可得下次时钟中断到来的时间. 另外如果内核中的实时任务较多, 很容易导致非实时任务等待使用临界区的时间太长, 在最坏情况下有可能出现当前非实时任务的执行时间永远大于下一个实时任务到来的时间, 导致非实时任务无法得到响应而夭折. 为了能有效保证非实时任务在可能的间隙时间内得到调度, 利用在 Linux 系统中每个临界区 i 的执行时间 $T_{lock\ i}$ 是确定值^[3]这一条件, 设置一个最小调度周期时间 P_{min} 值, $P_{min} = \max(T_{lock\ 1}, T_{lock\ 2}, \dots, T_{lock\ n})$, 即实时任务只有在其调度时间大于 P_{min} 时才允许其通过执行自旋锁使用临界区.

为提高对临界区的执行效率, 减少实时任务的延迟时间, 同时有效保证非实时任务在有限时间内得到调度, 根据以上分析提出自旋锁双模块控制模型. 即在 Linux 中增加两个模块, 一个是用于监控临界区可执行时间的自旋锁监控模块. 另一个是判别是否可以进入临界区的自旋锁控制模块. 当进程要获得自旋锁进入临界区之前, 自旋锁控制模块根据时钟标志位判别是否为实时进程, 如是实时进程, 则执行(1); 如是非实时进程, 执行(2). (1)实时进程比较自身的调度时间和 P_{min} 的值, 当 $T_{schedule} > P_{min}$ 时, 进程获取自旋锁并进入临界区, 否则进程进入工作队列等待下一次判别; (2)非实时进程, 它比较自身的执行时间 $T_{execute}$ 和下一个中断到来的时间值 T_{real_timer} , 当 $T_{execute} < T_{real_timer}$ 时, 进程获取自旋锁并进入临界区, 否则进程进入工作队列等待下一次判别. 其对自旋锁实施双管理模型的部分伪代码如下:

```
# define spin_lock(Tschedule, Pmin, Texcute, Treal_timer)
If(real_timer_on){/* real_timer_on 表示时钟标志 */
    If(Tschedule > Pmin){
        Measure_current_time();
        Preempt_disable();
        Raw_spin_lock(lock);
    }
    Else{
        Sleep_this_thread();
    }
}
```

```
Else{
    If(Texcute <= Treal_timer){
        Measure_current_time();
        Preempt_disable();
        Raw_spin_lock(lock);
    }
    Else{
        Sleep_this_thread();
    }
}
}
# define spinunlock()
Do{
    update_lock_hold_time();
    Raw_spin_unlock(lock);
    Preempt_enable();
    Add_timer();
}
}
```

3.3 0(1)算法的改进

综合 2.3 节对 0(1)算法的分析, 同时结合时间片轮转算法的优点, 对 0(1)调度算法提出如下改进:

(1) 删除结构体 `runqueue` 中的等待队列以及其相关指针, 为使优先级相同的任务具有并行处理的运行权利, 增加一个时间片轮询装置. 这样当算法通过使用优先位图快速找出系统中优先级最高的待执行队列后, 将使用时间片轮询装置对该队列中的进程进行调度; 当进程运行完设定的时间片后, 它将释放处理器以便同一级别的下一个进程被调度, 如果此时进程没有执行结果, 则将其移到它所在优先级队列的末尾, 等待下一个时间片的到来. 如此该进程的优先级不变, 因此就不需要队列切换, 只有当前最高优先级队列中的进程全部处理完成后, 才切换到剩余队列优先级最高的那个队列中去, 这样不仅在保证优先级的情况下, 减轻了上下文切换以及数组指针交换的负载, 也显著提高了实时进程的执行效率.

(2) 为了更合理地执行进程, 有针对性地分配时间片, 优化系统资源利用率. 在 `runquene` 中增加一个新的队列数组, 用于特殊处理其时间主要花费在等待 I/O 设备响应的 I/O 消耗型^[6]进程, 该数组的每一个元素指向一个队列, 每个队列对应于一种 I/O 请求. 当一个 I/O 请求处理完成后可以通过这个队列迅速地找到下一个 I/O 请求进程, 同时将完成 I/O 操作的进程, 根据其运行过程中占用 CPU 资

源的情况重新计算其优先级,占用时间长则降低其优先级,否则保持或提高其优先级,然后将其放回运行队列中.如此将在满足实时性系统要求的情况下,能够兼顾处理通用性进程,提高系统快速响应时间.

运行队列的结构体定义如下:

```
Struct runqueue{
Spinlock_t lock; /* runqueue 的自旋锁 */
Unsigned long nr_running; /* 运行队列中的进程数 */
Struct prio_array_t * active, array; /* 运行队列指针及队列容器 */
Struct prio_array arrays; /* 优先级队列 */
Atomic_t nr_iowait; /* 记录本 CPU 因等待 I/O 而处于休眠状态的进程 */
}
Struct prio_array{
Int nr_active;
Struct list_head queue[MAX_Prio]; /* 每个优先级的进程队列 */
Unsigned long bitmap[BITMAP_SIZE]; /* 进程队列的索引位图 */
}
```

4 嵌入式 linux 内核实时性能测试

测试硬件平台:博创 UP-TECHPXA270, CPU: Intel XScale 结构芯片 PXA270, 主频 520MHZ;内存 SDRAM: HY561620, 64M; I-Cache 16K, D-Cache 16K. 测试 2.6.11 内核版本为 ucLinux 内核和改进的 uclinux 内核.

4.1 HackBench 测试

HackBench 测试是由 Rusty Russell 提出的、用于评估 Linux 进程调度器的调度性能、负载性和可扩展性的一种 Benchmark 标准测试.在实验中,分别创建了 20、60、120、200、250、300、400、500、600 组进程,分别在 ucLinux2.6.11 和改进后的 ucLinux2.6.11 上作测试.实验结果如图 1 所示.改进后的内核其多任务调度完成时间总体上要低于未改进的内核.特别是在进程数大幅增加时,改进后的内核其性能未受到大的影响.

4.2 SleepTest 测试

用来评估内核的延迟以及稳定性.在本试验中, sleepTest^[7]进行 1000 次睡眠循环,每次睡眠 1s,最后统计出最大、最小以及平均的睡眠延迟时间,测试结果如表 1 和表 2 所示.从结果可见改造过的内核延迟比改造前有明显降低,且延迟相当稳定,保持在低延迟状态.

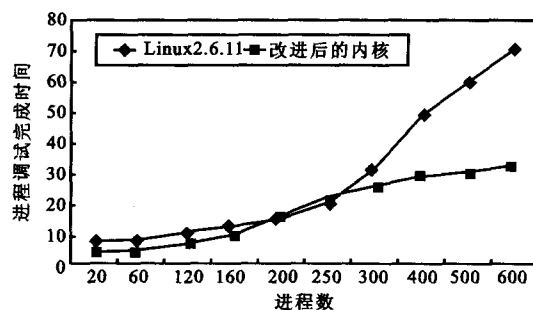


图 1 实验结果对比图

表 1 没有负载的结果 (μs)

内核	最大延迟时间	最小延迟时间	平均延迟时间
2.6.11	2340	1213	1698
改进后的内核	1324	651	912

表 2 以两个 find 命令作为负载的结果 (μs)

内核	最大延迟时间	最小延迟时间	平均延迟时间
2.6.11	9876	3215	6481
改进后的内核	1545	767	1024

5 结束语

通过对 Linux2.6 内核实时性能的分析,归纳出其主要的制约因素,然后有针对性的从内核时钟、临界区、调度算法三方面对 Linux2.6 内核的实时性进行了改进.实验测试表明,其改进有效提高了系统的实时性能.

参考文献:

- [1] 刘文峰,李程远,李善平. 嵌入式 Linux 操作系统的研究[J]. 浙江大学学报:工学版,2004,38(4):447-452.
- [2] Michael B. A linux based real time operating system[D]. Socorro, New Mexico: New Mexico Institute of Mining and Technology, 1997.
- [3] Love Ro. Linux kernel development[M]. 2nd ed. Indiana: Sams Publishing, 2005.
- [5] Michael L Fredman, Robert E Tarjan. Fibonaeci heaps and their uses in improved network optimization algorithms [C]// Proc 25th FOCS. California, 1984:338-346.
- [6] Abeni L, Goel A, Krasic C, et al. A measurement-based analysis of the real-time performance of linux [C]// The 8th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2002). San Jose, 2002.
- [6] 马超,何炎祥,王波. 一种实时性 O(1) 调度改进算法[J]. 计算机应用研究, 2009, 26(7): 2450-2453.
- [7] 胡强,蔡自兴. 一种基于改造时钟系统的 linux 实时化方案[J]. 计算机工程, 2009, 29(18): 271-273.

部分去重,去重的目的是将多个关键字对应的网页进行整理.

定义 z_i 表示 s_i 的整数部分,同时也可得 $s_i - z_i$ 表示 s_i 的小数部分.利用 HashTable<key, Value> 且 key 为唯一的特性,进行去重,如下伪代码:

```
if(hashtable.Contains( $z_i$ ))
{
    hashtable[ $z_i$ ].value +=  $s_i - z_i$ ;
    //将相同整数部分的值进行小数部分相加,小数部分
    就是出现的次数
}
else
{
    hashtable.add( $z_i$ ,  $s_i - z_i$ );
    //如果没有将其新添加
}
```

定义 A 为 S 去重整理后的新集合.

定义 a_i 是集合 A 中每一个元素的小数部分.将所有的 a_i 根据需要返回的数据条数从高到低进行排序(例如:需要向页面返回 10 条数据,那么只需排前 10 条数据即可).并对应输出其整数部分,整数部分即为 h_m ,即得到的是排序以后的文档 d_i .

最后根据排序后的文档 d_i 返回数据,包括:标题、数据、网页链接、收录日期、网页大小、网页快照.

3 实验结果及分析

本系统测试实验以新疆大学站点 (<http://www.xju.edu.cn>) 为例,新疆大学站点共有网页量为 2 万余页.如表 3 所示.

表 3 实验结果

输入关键字	返回结果/条	返回时间/s
大学	11827	0.051
大学生活	2580	0.020
校园大学	3501	0.022
校长信箱网址	250	0.005

由以上统计信息可以得出,此站内搜索引擎可

以充分挖掘站点数据,随着关键字的增多,搜索定位越准备,提高了搜索的准确度和搜索速度,提升了网站形象.

4 结束语

文中介绍了一个基于站内搜索的数据挖掘研究与实现,不同于传统的搜索引擎.(1)基于高效的站内搜索,自主实现逆向最大匹配分词,扩展了网页排序方法.(2)利用数据挖掘的原理,实现倒排索引,在各个细节中都优化查找方式.(3)深入细化,准确到位,体贴用户心理,发现用户最关注的内容,寻找用户的潜在需求.

参考文献:

[1] 卢亮,张博文. 搜索引擎原理、实践与应用[M]. 北京:电子工业出版社,2007.
[2] 梁斌. 走进搜索引擎[M]. 北京:电子工业出版社,2007.
[3] 黄魏,高兵,刘异,等. 基于词条组合的中文文本分词方法[J]. 科学技术与工程,2010,10(1):85-89.
[4] 基于 LUCENE 的站内搜索引擎的设计实现论文[EB/OL]. (2010-05-26) [2010-06-01]. <http://www.souku.com.cn/viewtitle.jsp?url=66541652>.
[5] 银杏高科. 专业的站内搜索提供商[EB/OL]. (2010-04-20) [2010-06-01]. <http://www.ginkgotek.com/aq2.html>.
[6] 严蔚敏,吴伟民. 数据结构:C语言版[M]. 北京:清华大学出版社,2005.
[7] 周娅,张振宇. 数据结构[M]. 重庆:重庆大学出版社,2003.

作者简介:

刘凡平 男,(1988-),研究方向为搜索引擎以及信息处理.
高艳华 女,(1981-),硕士研究生.研究方向为信息处理与数据挖掘.
于 炯 男,(1964-),博士,教授.研究方向为计算机分布式系统、网络安全.
张 伟 男,(1984-),研究实习员.研究方向为电子商务.

(上接第 213 页)

作者简介:

李少雪 男,(1983-),硕士研究生.研究方向为嵌入式系统、网络.

吾守尔·斯拉木 男,(1942-),教授,博士生导师.
郭凯敏 男,(1985-),硕士研究生.研究方向为嵌入式系统、多文种信息处理.