第 28 卷第 4 期　　四川理工学院学报(自然科学版)　　Vol. 28　No. 4
2015 年 8 月　　Journal of Sichuan University of Science & Engineering(Natural Science Edition)　　Aug. 2015
文章编号:1673-1549(2015)04-0033-04　　DOI:10.11863/j.suse.2015.04.07

嵌入式 Linux 内存管理的优化

于国龙[a,b], 崔忠伟[a,b], 左　羽[a,b]

(贵州师范学院 a. 数学与计算机科学学院;b. 贵州省高校工业物联网工程技术研究中心, 贵阳　550018)

摘　要:内存管理是影响嵌入式 Linux 实时性的一个关键因素,为了提高嵌入式 Linux 的实时性,对其内存管理进行了优化。首先为系统中的重要任务分配了专用的内存区域,使重要任务在内存不足时不被置换出去,以保障重要任务优先执行;然后通过利用系统空闲时间来扫描系统内存的方法,使得任务在执行时尽量减少缺页中断的发生,从而提高系统的实时性;最后通过实验对比 OPT 最优算法、LRU 算法、优化后的 LUR 算法的缺页中断数和任务截止期错失率,发现优化后的 LUR 算法的缺页中断数和任务截止期错失率在三者中最低,说明通过以上的内存优化方法使得嵌入式 Linux 的实时性得到了提高。

关键词:嵌入式 Linux;实时性;重要任务;页面置换;缺页中断

中图分类号:TP316.2　　**文献标志码**:A

引　言

近年来,随着电子计算机硬件和软件系统的发展,越来越多的嵌入式产品走进人们的日常生活,如音乐播放器、手机、数字电视等。嵌入式系统主要是面向专用领域的系统,对实时性能有着较高的要求,并且大多数的嵌入式系统都搭载了嵌入式操作系统。嵌入式 Linux 操作系统以其特有的开放性、与生俱来的网络特性,在嵌入式领域得到了广泛应用,目前已经被成功地应用到了 PDA、移动电话、音乐播放器等各种嵌入式设备上。但在国防、航空、工业控制等对实时性要求非常严格的场合中,嵌入式 Linux 的实时性还有待改进,因此,提高嵌入式 Linux 的实时性成为嵌入式操作系统的一个重要研究方向[1-2]。

目前嵌入式 Linux 实时性的解决方案非常多,但总的来说,在构造嵌入式 Linux 系统时,主要从细粒度微定时器、内核抢占机制、实时调度策略、中断机制、内存管理优化等几个方面来改善系统的实时性。内存管理是系统实时性的重要部分,快速有效的存储能提高系统的实时性[1,3-5]。内存管理中,内存不足时页面置换的效率是影响系统实时性的关键,目前嵌入式 Linux 中,主要采用 LRU(最近最少使用)算法、FIFO(先进先出)算法、NFU(最不经常使用)算法等作为内存不足时的页面置换算法[6-8]。其中 LRU 算法用的最多,也最高效,但是 LRU 算法在嵌入式系统中的应用也存在一定不足,如页面置换中断多,重要的任务不能得到优先内存保障等,针对这些不足,文章采用重要任务专用内存,并对内存定时间隔扫描以减少缺页中断的发生,因为中断的执行时间不确定,且往往耗时较多,严重影响系统的实时性,下面就是对嵌入式 Linux 内存管理的优化。

1　嵌入式 Linux 内存页面置换算法

嵌入式 Linux 的进程运行过程中,若其所要访问的页面不在内存中时,需把它们调入内存,若内存已无空闲空间时,为了保证该进程能正常运行,系统必须从内存中调出一页程序或数据,送外存储器的对换区中。但应将哪个页面调出,就需要根据一定的内存页面置换算法来确定,常用的内存页面置换算法见表 1,其中,最为常用的算法有 OPT 算法、LRU 算法、NFU 算法等[9-11]。

收稿日期:2015-06-01

基金项目:贵阳市科技计划项目(筑科合同[2013101]10-6);贵州师范学院校级项目(14ZC009)

作者简介:于国龙(1981-),男,辽宁丹东人,讲师,硕士,主要从事嵌入式系统方面的研究,(E-mail)896999517@qq.com

表 1 常用的页面置换算法

序号	页面置换算法	特 点
1	OPT(最优算法)	作为基准的理想算法,不可实现。
2	LRU(最近最少用)算法	需硬件支持的算法,性能很优秀。
3	FIFO(先进先出)算法	页面没有重要度之分,性能一般。
4	NRU(最近未使用)算法	LRU 的一种粗略近似算法。
5	NFU(最不经常使用)算法	LRU 的一种粗略近似算法,比 NRU 性能要高。
6	第二次机会算法	FIFO 算法的改进版。

2 LRU 算法的优化

2.1 LRU 算法在嵌入式系统中应用的不足

在嵌入式 Linux 中大多采用 LRU 页面置换算法,当系统内存不足时,若有进程引起新的内存空间的分配需求,系统内核采用双向链表对系统内存空间进行扫描,以释放最久未使用过的内存页面满足系统需求。在这个过程中系统需要产生多个中断,以响应内外存储间的页面置换。中断是嵌入式 Linux 系统诸多操作事务中最耗时的操作之一,而且时间很难预测,因此,它对系统实时性的影响也是非常大的,如果系统中某些实时性要求较高的任务在执行过程中频繁的产生页面置换中断的话,将非常不利于系统的实时性能,甚至给系统造成灾难性的后果。

针对以上 LRU 页面置换算法在嵌入式 Linux 中应用存在的不足,可以采用以下的优化方法[12-14]:首先,在 LRU 页面置换算法中,要尽可能地保证系统中实时性要求高的那些重要任务能优先顺利地执行完成;其次,在此基础之上,还要尽可能地减少任务在执行过程中页面置换中断的发生,从而提高系统的实时性。基于以上的思想,提出了基于重要任务间隔扫描的 LRU 算法。

2.2 基于重要任务间隔扫描的 LRU 算法

基于上文提到的 LRU 算法的优化思想,需要着重解决两个问题:(1) 重要任务常驻内存;(2) 运行过程中页面置换中断。

首先要解决重要任务常驻内存的问题,可以在内存空间中,按实际需要划分出一块空间来,让系统中比较重要的任务常驻内存,不被其它任务置换出来。这样重要的实时任务就不会因为内存不足,来回的在内外存之间置换,从而提高重要任务的实时性。在嵌入式 Linux 系统中,可以通过 kswapd 进程来实现重要实时任务的内存常驻。在 kswapd 进程中增加一个内存页面区域,当系统初始化时,将系统中重要的实时任务所需的内存页面空间载入该区域,这样 kswapd 的扫描进程就不再扫描该区域,这样不仅保证了重要程序的内存空间的供给,另一方面也减少了 kswapd 进程的扫描时间,提高扫描效率,从而增强系统的实时性。

其次,通过优化对内存的扫描来减少任务运行过程中页面置换中断的发生。由于中断的执行时间较长,且比较难预测,这对实时任务的执行是非常不利的,也会影响到整个系统的实时性。针对以上的考虑,在系统处于空闲时间时,启动内存扫描,并进行页面置换,来减少任务执行过程中内存页面置换中断的发生。在系统中设定一个时间间隔,每个时间间隔到达之后,当第一次出现系统空闲时,开始扫描内存,查找那些在内存区域最近最少使用的页面,将这些内存页面置换出去,不必等到内存不足时再置换,这样会减少任务执行过程中的内存置换的次数,从而提高系统的实时性。

2.3 优化后的 LRU 算法实现

首先要实现优化后的 kswapd 进程。kswapd 进程的实现相当复杂,这不仅仅涉及复杂的页面交换技术,还涉及与磁盘相关的具体文件操作,需要调用很多函数,其中由 Swap_out( ) 和 shrink_cache( ) 一起完成到底哪些页面会被作为候选页以备换出。shrink_cache( ) 要做很多换出的准备工作[1,6],它关注两个队列:“活跃的”LRU 队列和“非活跃的”FIFO 队列,每个队列都是 struct page 形成的链表。kswapd 进程实现的伪代码如下:

```
while(系统内存不足时)
{
if(不是重要任务内存区域)
{
对“非活跃的”FIFO 队列进行扫描,然后进行页面置换。
}
else
{
跳过该区域,对其它区域进行扫描
}
}
```

接下来要实现系统对内存的间隔扫描。需要在 Linux 内核中添加一个系统调用,并指明系统主动扫描内存页面的时间间隔的参数。具体实现的伪代码描述如下:

```
While(系统到达时间间隔)
{
if(空闲内存页面 < = 设定值)
```

```
}
将非活跃队列中的最近最少使用的页面置换。
}
else
{
等待下一个时间间隔或者缺页中断
}
}
```

3　实验分析

为了验证上面的 LRU 算法优化效果,在博创嵌入式试验箱上做了相应的实验,硬件环境是 Samsung 公司基于 ARM 公司 ARM920T 处理器核的 S3C2410 处理器,它拥有 64 M 内存的内存,并在开发板上面移植了 Linux Kernel 2.6 内核。在此实验平台上,通过对 OPT 最优算法、LRU 算法、优化后的 LUR 算法三者进行页面中断次数和截止期错失率的对比实验,来分析优化后的 LRU 算法的性能。实验任务参数参见文献[6 - 8,15 - 16],实验中任务的周期(ms)、运行时间(ms)、任务运行所需的内存页数见表 2。

表 2　实验数据表

任务	周期/ms	运行时间/ms	所需内存页数	任务级别
*s*1	30	12	9567	重要
*s*2	45	13	8021	一般
*s*3	35	14	7555	一般
*s*4	50	15	6433	一般
*s*5	40	20	5155	重要
*s*6	40	15	4982	一般
*s*7	30	15	3276	一般
*s*8	45	20	2696	一般
*s*9	55	16	1532	一般

首先通过实验分析优化后的 LUR 算法在内存不足时的缺页率。对 OPT 最优算法、LRU 算法、优化后的 LUR 算法,在随着任务 *s*1,*s*2…,*s*9 依次载入系统时,所产生的缺页率进行比较,结果如图 1 所示。从图 1 可知,随着任务不断的载入系统,三种算法的系统中的内存缺页率都在不断地增加,相对 OPT 最优算法与 LRU 算法而言,优化后的 LRU 算法的内存缺页率低于 LRU 算法,接近 OTP 算法,显然,当内存不足时,在保证重要任务的前提下,缺页率得到了降低。

其次再通过实验来分析优化后的 LRU 算法对系统的实时性的影响。对 OPT 最优算法、LRU 算法、优化后的 LUR 算法,在随着任务 *s*1,*s*2…,*s*9 依次载入系统时,任务的截止期错失率进行实验分析,结果如图 2 所示。通过实验结果可以看出,三个算法随着任务数的增加,

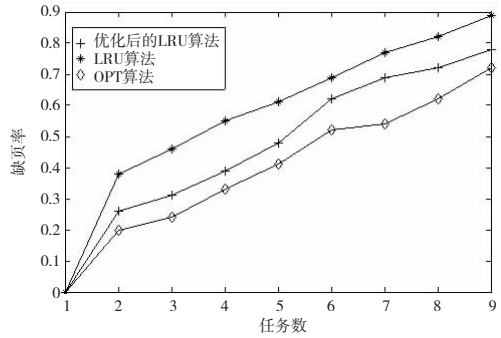

图 1　任务内存缺页中断数

截止期错失率都在增加,但优化后的 LRU 算法的截止期错失率要低于 LRU 算法,这说明优化后的 LRU 算法,使得系统的实时性有所提高,能使更多的任务实时完成。

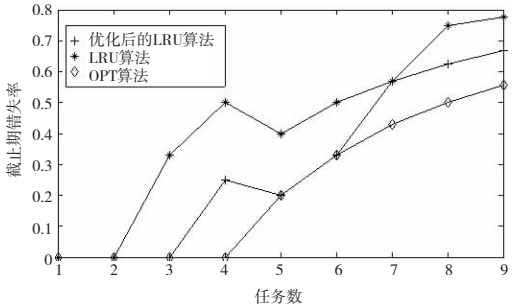

图 2　任务截止期错失率

4　结束语

通过为系统中的重要任务分配固定的内存区域,以减少重要任务因内存不足而发生的页面置换,重要任务就会常驻内存,在内存不足时不会频繁的发生页面置换中断,可以在一定程度上有效地提高重要任务的实时性,这对嵌入式系统尤为重要。同时利用系统的空闲时间,来对内存进行扫描。正常情况下,对内存扫描只发生在缺页时,要对系统扫描找出最近最少使用的页面进行扫描并进行置换,任务就需要等待内存扫描置换这个过程,降低了系统的实时性;如果在系统空闲时间对内存进行扫描,并将最近最少使用的内存置换出去,不必等到内存不足时再扫描置换,可以提高处理器的利用率,从而提高系统的实时性。针对以上思想优化后的 LRU 算法的实验结果表明,优化后的 LRU 算法页面置换中断数量要低于优化前,并且任务的截止期错失率也低于优化前,说明优化后的 LRU 算法的实时性得到了提高。

参 考 文 献:

[1] 吴志君,段富海.嵌入式 Linux 系统内存优化使用方

法研究[J].甘肃科学学报,2012,24(1):139-142.
[2] 武建平,方攀,凌明,等.面向 Linux 内核的片上存储优化[J].微电子学,2012,36(2):82-84.
[3] 吕广锋,陈蜀宇.一种新的适用于 Nandflash 的 Linux 内存交换模型[J].计算机应用研究,2010,27(10): 97-100.
[4] 杨峰.基于 Linux 内核的动态内存管理机制的实现[J].计算机工程,2010,36(9):85-88.
[5] 高超,韩锐,倪宏.嵌入式 Linux 平台内存管理方案[J].小型微型计算机系统,2011,32(4):29-31.
[6] 张昌昌,林满山,宋威,等.基于缺页的 Linux 任务管理器设计与实现[J].计算机工程与设计,2011,32(9): 3059-3062.
[7] 黄仁,李建章,程平.基于 RM 与 EDF 的实时混合调度算法研究[J].电子技术应用,2010,36(12):29-31.
[8] 张建,刘青昆,王异奇.Linux 实时化方法的研究与实现[J].计算机工程,2011,37(11):253-256.
[9] 陈何杰,郑灵翔.ARM Linux 中断系统移植研究[J].厦门大学学报,2010,49(3):341-344.
[10] Song Kai, Yan Liping. Improvement of real-time performance of Linux 2.6 Kernel for embedded application [J]//Proceedings of 2009 International Forum on Computer Science-Technology and Applications, Chongqing, China, December25-27,2009:71-74.
[11] 赵连玉,靳飞.嵌入式计算机系统 Bootloader 的设计与实现[J].天津理工大学学报,2011,27(1):22-26.
[12] Lee C,Lim S H.Caching and deferred write of metadata for Yaffs2 flash file system[C]//Proceedings of International Conference on Embedded and Ubiquitous Computing, Melbourne, Australia, October 24-26, 2011: 41-46.
[13] Purdila O, Grijincu L A, Tapus N. The Linux Kerne Library [C]//Proceedings of Roedunet International Conference (RoEduNet), Sibiu, Romania, June 24-26, 2010:316-319.
[14] Liu Yinli,Yu Hongli,Zhang Pengpeng.The implementation of embedded acquisition based on v4l2[C]//Proceedings of 2011 International Conference on Electronics,Communications and Control,Ningbo,China,September 9-11,2010:549-552.
[15] 周上群.一种新的嵌入式系统存储器测试算法及应用[J].桂林电子科技大学学报,2013,33(3):29-32.
[16] Liu Yijun, Chen Wenbin, He Xiaoman. Research and implementation of embedded graphic user interface based on Linux[C]//Proceedings of 2010 3rd IEEE International Conference on Computer Science and Information Technology(ICCSIT),Chengdu,China,July 9-11,2010: 693-696.

The Optimization of Embedded Linux Memory Management

YU Guolong[a,b], *CUI Zhongwei*[a,b], *ZUO Yu*[a,b]

(a. School of Mathematics and Computer Science; b. Guizhou Province University Industrial Networking Engineering Technology Research Center, Guizhou Normal College, Guiyang 550018, China)

Abstract: Memory management is a key factor to impact the instantaneity of embedded Linux. In order to improve the instantaneity of embedded Linux, its memory management is optimized. At first, the important task in the system is assigned a dedicated memory area, so that important task will not be swapped out when the memory is lacking, which can protect the important task precedence; Then through the method that use the system idle time to scan the system RAM, to reduce the frequency of missing page interrupt in the task execution, thereby the system instantaneity is improved. Last, the missing page interrupts numbers and the task deadline miss ratios of OPT optimization algorithm, LRU algorithm and optimized LUR algorithm are compared through experiment, and it is found that the missing page interrupts number and the task deadline miss ratio of optimized LUR algorithm is the least among the three algorithms, which indicates that the instantaneity of embedded Linux is improved through the above described memory optimization method.

Key words: embedded Linux; instantaneity; important task; page replacement; missing page interrupt