优先级驱动的短任务优先 RTOS 进程调度算法*

李正平，程八意，陈军宁
（安徽大学 电子信息工程学院，合肥 230601）

摘 要：针对目前普遍采用的基于静态优先级轮转调度算法的不足，提出一种静态优先级驱动的短任务优先动态时间片轮转调度算法，称之为 LC 调度算法。该算法采用短任务优先的调度策略，动态分配时间片，为每个静态优先级设置两个就绪队列 RQ_0 和 RQ_1，将 burst time 短的进程插入 RQ_1，长的插入 RQ_0。当 RQ_0 上有进程等待时间过长则会自动提升到相应的 RQ_1 上。模拟实验表明：LC 算法在各种条件下都能保持优秀的性能，它充分考虑了调度的开销、性能、响应速度和公平性，并且对 burst time 预测精度不敏感，相对于其他改进的 RR 算法具有更好的表现。

关键词：操作系统；轮转调度；突发时间；双就绪队列；优先级提升
中图分类号：TP301.6 **文献标志码**：A **文章编号**：1001-3695(2014)04-1020-03
doi:10.3969/j.issn.1001-3695.2014.04.015

New proposed priority driven short job first scheduling algorithm for RTOS

LI Zheng-ping, CHENG Ba-yi, CHEN Jun-ning
(*School of Electronics & Information Engineering*, *Anhui University*, *Hefei* 230601, *China*)

Abstract: To overcome the shortages of current widely used static priority-driven round robin scheduling algorithm in RTOS, this paper proposed a static priority-driven short job first dynamic round-robin scheduling algorithm, called LC algorithm. The algorithm used SJF scheduling policy and dynamically allocated time slice, it set two ready queues RQ_0 and RQ_1 for each static priority, inserted short burst time processes into RQ_1 and long burst time processes into RQ_0. LC algorithm calculated waiting time for long burst time processes in RQ_0, if the waiting time was too long, it reinserted the first RQ_0 process into the tail of RQ_1. Simulation test shows that the LC algorithm can maintain excellent performance in all cases. LC algorithm makes a good tradeoff between schedule overhead, performance, response speed and fairness. Moreover, it isn't sensitive to inaccuracy of burst time prediction, so it is better than some other improved RR algorithms.

Key words: operating system; round robin; burst time; double ready queue; priority upgrade

0 引言

实时系统的关键特征是能够在规定的时间内响应事件的请求并给出正确的处理结果。因此，实时操作系统的进程调度算法首要目标是满足系统内实时任务的时间约束，合理地调度并执行这些任务，使之能在各自的规定时限之前完成操作。衡量调度算法的主要指标有平均周转时间、平均等待时间、CPU 利用率、响应速度和公平性。经典的基于静态优先级的速率单调调度算法（RMS）是单处理器下的最优静态调度算法，算法简单便于实现，但其处理器利用率较低，实际应用中往往难以精确确定任务的执行周期。基于动态优先级的最早任务期限优先（EDF）调度算法虽然具有处理器利用率高的优点，但瞬时过载时，系统行为不可预测，而且实际中任务的期限往往难以准确确定。轮转（round robin）调度算法基于静态时间片简单易实现，但静态时间片的长度难以确定，太短和太长均不合适，容易导致系统性能和响应速度下降。近年来有很多基于动态时间片的轮转调度算法的研究[1~9]，如 PBDRR[2] 算法能根据每个进程的 burst time 值动态地分配时间片的大小，但是该算法性能完全依赖于对 burst time 的精确估计之上，当 burst time 估算不准确时，该算法的性能会很糟糕。FJFDRR[3] 算法根据优先级和 burst time 为每个进程计算出一个因子 f，f 值小的就可以先得到执行，不过该算法用到浮点运算等方式，调度开销大。其他如 SRBRR[6]、MDTQRR[7] 和 AN[4] 等改进型 RR 算法虽然克服了传统轮转调度算法的一些不足，但算法本身往往过于复杂，调度开销大，或者公平性不足，易导致长任务饥饿，有的算法甚至依赖于实际难以满足的前提，从而使得无法实际应用。最短作业优先（SJF）调度算法性能优越，但存在容易导致长作业饥饿和不能保证紧急性作业得到及时处理的缺点。其他如最紧急优先（MUF）等算法也都存在类似的问题，从而实际中难以应用。

目前多数实时操作系统如 freeRTOS、RT-thread 均采用了基于静态优先级的调度算法，甚至 Linux 针对实时进程也是采用相同的调度算法[10,11]。该算法按照重要性为进程分配不同的优先级，调度器总是选择处于就绪状态的最高优先级进程运

收稿日期：2013-07-07；**修回日期**：2013-08-19 **基金项目**：国家核高基重大专项项目（2009ZX01031-001-004）

作者简介：李正平（1979-），男，安徽宣城人，副教授，硕导，博士，主要研究方向为嵌入式系统、电路与系统；程八意（1988-），男，安徽芜湖人，硕士研究生，主要研究方向为嵌入式系统（chengbayi100@126.com）；陈军宁（1953-），男，安徽芜湖人，教授，博导，博士，主要研究方向为超大规模集成电路与系统设计、嵌入式系统及系统解决方案.

行,如果有多个进程优先级相同,则采用轮转算法调度。该算法在同优先级进程间的调度由于采用了传统 RR 算法,表现一般,存在不足。

本文在此基础上提出了一种静态优先级驱动的短任务优先动态时间片轮转调度的算法,称之为 LC 调度算法。该算法重点改进了同优先级进程间的轮转调度,考虑了调度的开销、性能和公平性,以期在众多指标中取得良好的平衡。

1　算法原理

LC 调度算法是基于静态优先级的调度算法。开发者综合进程的重要性、deadline 和执行周期等因素为每个进程设置优先级,优先级在系统运行过程中不会动态调整。高优先级进程就绪时立刻抢占低优先级进程,同优先级的进程采用短任务优先动态时间片轮转调度算法进行调度。

LC 算法为每个优先级设置两个就绪队列 RQ_{i0} 和 RQ_{i1}。每当进程 k 就绪时,首先估计其 burst time(BT_k),并与该优先级就绪队列中所有进程的剩余 burst time 平均值 BT_{avg} 比较。如果 $BT_k < BT_{avg}$ 或小于 25 ms,则将该进程插入 RQ_{i1} 队尾,反之放入 RQ_{i0} 队尾。LC 算法在选择进程运行时,首先寻找具有就绪进程的最高优先级,再检查该优先级对应的 RQ_{i1} 就绪队列。如果该队列非空,则选择队列上第一个进程运行;如果 RQ_{i1} 为空,则检查 RQ_{i0},选择其上第一个进程运行。假设某个优先级的就绪队列中有 N 个进程,SR 代表所有进程剩余 BT 的总和,BT_{avg} 是 SR 的平均值,WT_i 等于当前 RQ_{i0} 的等待时间。LC 算法的伪代码如下:

a)当一个优先级为 j 的进程就绪,计算其 BT,if (($BT < BT_{avg}$) ‖ ($BT < 25$ms)),将其插入 RQ_{j1},否则插入 RQ_{j0};更新 SR、BT_{avg}、N。

b)确定就绪队列非空的最高优先级(假设为 i),$WT_i = 0$。

c)if (RQ_{i1} ! = null)

{

(a)取得 RQ_{i1} 上第一个进程,设置其时间片 $TQ = \max(25, BT_{avg})$,运行;

(b)if(BT 在 TQ 之前结束)更新 SR、BT_{avg},将进程从 RQ_{i1} 中删除,N - - ;

else 将该进程移到 RQ_{i0} 队尾,更新 SR 、BT_{avg};

(c)WT_i = 本次执行时间 + WT_i;

if($i > 2BT_{avg}$)　将 RQ_{i0} 第一个进程移到 RQ_{i1} 尾部,$WT_i = 0$;

}

else {

(d)取得 RQ_{i0} 上第一个进程,设置其时间片 $TQ = \max(25, BT_{avg})$,运行;

(e)if(BT 在 TQ 之前结束)　计算 BT,更新 SR、BT_{avg},将进程从 RQ_{i0} 中删除,N - - ;

else 将该进程移到 RQ_{i0} 队尾,更新 SR、BT_{avg}

}

end

进程的时间片 TQ(time quantum)是动态的。每当选择一个进程运行时,设置其时间片为当前的 BT_{avg}。为了避免时间片过小而导致过多的上下文切换,算法为进程的时间片设定一个下限,根据 Matarneh[5] 的研究,其下限设为 25 ms,即如果计算出的时间片小于 25 ms,则将其调整为 25 ms,保证任何情况下进程的时间片都不小于 25 ms 的下限值。

如果一个进程耗尽其时间片后,其 burst time 时间仍未结束,算法将其插入 RQ_{i0} 队列的尾部,等待下一次调度。为防止突发时间长的进程饥饿,算法为每个优先级维护一个等待时间 WT_i,用于跟踪 RQ_{i0} 上长 burst time 进程等待运行的时间。当 WT_i 大于 BT_{avg} 的两倍时,系统将 RQ_{i0} 的第一个进程移到 RQ_{i1} 上,使其尽快调度。

进程的 burst time 时间无法事先确定,只能基于运行历史进行估计。由于估计运算在系统运行过程中频繁被执行,因此估计的算法不能过于复杂,否则将带来不可接受的开销。在实际应用中,可以采用如下高效估计算法:

$$BT_{k_est}^{n+1} = \frac{BT_{k_est}^{n} + BT_{k_real}^{n}}{2} \tag{1}$$

式中:$BT_{k_est}^{n+1}$ 和 $BT_{k_est}^{n}$ 分别为进程 k 本次和上次的 burst time 估计值,$BT_{k_real}^{n}$ 则为上次 burst time 的实际值。每当进程一次 burst time 结束时,LC 算法将计算出本次的 burst time 实际值,保存在进程描述符内。一般情况下只要进程的 burst time 实际长度比较稳定,估算的 burst time 将非常接近实际值。

2　算法分析

2.1　理论分析

实时操作系统的进程调度算法首要目标是保证关键进程的优先运行,而对重要性相同的进程,则需要在保证公平的基础上减少平均周转时间和平均等待时间。LC 调度算法通过优先级来描述进程的重要性,由于采用了静态优先级的调度算法,当高优先级进程就绪时总是立刻抢占低优先级进程,从而有效保证了关键进程的优先运行。

LC 算法对相同优先级的进程采用改进的轮转调度算法,使用简单高效的 burst time 估计方法,开销很小。如果能精确知道每个进程的 burst time 时间,则将 burst time 小的进程优先运行可以尽可能地减少平均等待时间和平均周转时间。LC 算法通过把 burst time 小的进程插入优先调度的 RQ_{i1} 就绪队列,降低了平均等待时间和平均周转时间。

相对于 SJF 和 FJFDRR[3] 等算法,LC 算法为每个优先级维护等待时间 WT_i,当处于 RQ_{i0} 中的进程等待运行的时间超过阈值,将被提升至 RQ_{i1} 队列优先运行,避免了长 burst time 进程长时间得不到运行,具有更好的公平性。

很多算法(如 SJF 和 FJFDRR)的有效性依赖于对进程 burst time 时间的精确预测,然而实际中这是不可能的。当 burst time 预测不准时,将严重影响这些算法的性能。LC 算法存在同样的问题,但是当一个长突发时间的进程被错误地插入到 RQ_{i1}(由于 burst time 预测值远小于实际值)后,它只需经过一次时间片为 BT_{avg} 的运行就将被调整到 RQ_{i0},缓解了由于 burst time 预测不准导致的平均周转时间和平均等待时间的延长。

2.2　模拟实验

LC 调度算法和目前 RTOS 普遍采用的基于静态优先级的调度算法相比,主要是改进了同优先级进程之间的轮转(RR)算法。本文通过模拟实验研究了改进的轮转调度算法的性能,并与传统的 RR、以及 SRBRR[6]、MDTQRR[7] 和 AN[4] 三种改进的 RR 算法进行对比。为对比各调度算法的性能,采用平均周转时间、平均等待时间和切换次数三个指标,而最大周转时间、最大等待时间、平均响应时间和最大响应时间用于衡量算法的公平性和响应速度,好的算法应该有较小的值。

进程的等待时间定义为从进入就绪队列到执行完其 burst time 退出就绪队列这段时间内等待执行的时间长度。而响应时间为进程从进入就绪队列到第一次被调度执行之间的时间间隔。

情况 1　各个进程以 burst time 递增顺序到来(表中的单位均为 ms)

进程	P_1	P_2	P_3	P_4	P_5
到达时间	0	2	5	7	9
burst time	10	22	48	70	74

实验结果如表示 1 所示。最后一列是 LC 算法与 SRBRR、MDTQRR、AN 三种性能的平均值进行对比的结果,用性能提升的比率来显示。

表 1　情况 1 下的结果

比较项	算法					性能提升比率/%
	RR(25 ms)	SRBRR	MDTQRR	AN	LC	
平均周转时间	114.6	106.4	94.6	94.6	94.6	4.1
平均等待时间	69.8	61.6	50.2	49.8	49.8	7.4
切换次数	9	7	4	4	4	15.8
最大等待时间	166	141	141	141	141	0
最大周转时间	215	215	215	215	215	0
平均响应时间	32.2	47.6	49.8	49.8	49.8	-1.4
最大响应时间	73	130	141	141	141	-2.7

结果显示在 burst time 递增顺序到来情况下,LC 算法与 AN 算法性能相同,并且总体上优于其他三种算法。

情况 2　各进程以 burst time 递减顺序到来

进程	P_1	P_2	P_3	P_4	P_5
到达时间	0	6	13	21	75
burst time	73	50	23	19	5

可以看出,在此情况下 LC 算法仍然表现良好,不仅性能在五种算法中是最好的,而且反映公平性的几项指标也位居前列。与之相反,AN 算法性能退化严重,性能接近于 RR 算法,实验结果如表 2 所示。

表 2　情况 2 下的结果

比较项	算法					性能改善比率/%
	RR(25 ms)	SRBRR	MDTQRR	AN	LC	
平均周转时间	101.4	87.4	87.4	116.4	78.2	19.4
平均等待时间	67.8	53.4	53.4	82.6	45.4	28.1
切换次数	7	5	4	8	5	11.7
最大等待时间	120	114	114	122	103	11.7
最大周转时间	170	164	164	164	170	-3.6
平均响应时间	25	53.4	53.4	63.2	24.8	56.3
最大响应时间	52	114	114	100	58	47.1

情况 3　各进程以 burst time 随机顺序到来

进程	P_1	P_2	P_3	P_4	P_5
到达时间	0	6	8	9	10
burst time	7	15	90	42	8

结果显示,在这种随机顺序的情况下,SRBRR、MDTQRR 以及 LC 算法性能接近,表现都非常优秀,RR 算法性能次之,而 AN 算法在这种情况下表现相对是最糟的,如表 3 所示。

表 3　情况 3 下的结果

比较项	算法					性能改善比率/%
	RR(25 ms)	SRBRR	MDTQRR	AN	LC	
平均周转时间	72	52	52	82.4	58.8	5.3
平均等待时间	39.6	19.6	19.6	50	26.4	11.2
切换次数	8	5	4	7	4	25.0
最大等待时间	71	64	64	126	64	24.4
最大周转时间	154	154	154	154	154	0
平均响应时间	23	19.6	19.6	33	26.4	-9.6
最大响应时间	62	64	64	90	64	11.9

情况 4　进程以 burst time 随机顺序到来,但到达时间与情况 3 不同

进程	P_1	P_2	P_3	P_4	P_5
到达时间	0	6	8	25	29
burst time	7	15	90	42	8

情况 4 与情况 3 的差别在于各进程的到达时间不同,burst time 时间两者完全相同。对比情况 3 和 4 可见,进程的到达时间会影响算法的性能与公平性,但 LC 算法表现依然非常优越,所有指标都是位居前列。RR 算法的性能此时比 SRBRR、MDTQRR、AN 算法的表现都要好些,如表 4 所示。

表 4　情况 4 下的结果

比较项	算法					性能改善比率/%
	RR(25 ms)	SRBRR	MDTQRR	AN	LC	
平均周转时间	65	71	71	76	43	40.8
平均等待时间	32.6	38.6	38.6	43.6	10	70.3
切换次数	8	4	4	5	5	-15.3
最大等待时间	64	95	95	108	66	33.6
最大周转时间	154	137	137	137	154	-12.4
平均响应时间	16	38.6	38.6	42	10.6	73.3
最大响应时间	43	95	95	108	38	61.7

情况 5　Burst time 预测不准确,假设以 burst time 随机顺序到来

进程	P_1	P_2	P_3	P_4	P_5
到达时间	0	6	8	25	29
burst time 实际值	7	15	90	42	8
burst time 预测值	8	13	20	65	9

实验结果如表 5 所示。此处 SRBRR、MDTQRR、AN 算法无法执行,所以最后一列的性能提升比率是对比普通 RR 算法的结果。

表 5　情况 5 下的结果

比较项	算法					性能改善比率/%
	RR(25 ms)	SRBRR	MDTQRR	AN	LC	
平均周转时间	129.6	无法执行	无法执行	无法执行	58.2	55.1
平均等待时间	61.2				25.8	57.8
切换次数	10				6	40
最大等待时间	143				64	55.2
最大周转时间	210				154	26.7
平均响应时间	15				9.8	34.7
最大响应时间	27				25	7.4

在现实情况下,很难对 burst time 时间进行精确预测,情况 1 ~4 均是基于精确预测 burst time 的假设,情况 5 考察由于预测不准确对几种算法的影响。为便于对比,采用的数据(进程到达时间和 burst time 实际值)与情况 4 相同,只是假定 burst time 的预测值与实际值存在偏差。模拟结果显示,当预测的 burst time 不准确时,SRBRR、MDTQRR 以及 AN 算法根本无法正常执行,因此认为 SRBRR、MDTQRR 和 AN 算法只具有理论上的意义,实际实现和应用是非常困难的。而 LC 算法此时的各项指标影响并不大,对 burst time 预测的准确性敏感性不大。

3　结束语

实时操作系统的进程调度算法需要在保证任务时间约束的基础上提高响应速度,降低等待时间和周转时间,同时兼顾公平性。目前普遍采用的基于静态优先级的 (下转第 1026 页)

4 结束语

本文针对人工蜂群算法的不足,提出了一种改进的人工蜂群算法。IABC 算法中主要有三种改进策略:a)跟随蜂阶段采用一种基于当前局部最优解(pbest)的搜索方式,使得提高了算法的局部搜索能力;b)初始化种群应用一般的反向学习方法产生,可以使种群具有较好的多样性并在一定程度上加快了算法的收敛速度;c)IABC 算法在蜜蜂进行邻域搜索时,按循环代数以线性递增的方式调整了扰动的频率,加快了算法的收敛速度。通过采用标准测试函数对改进算法进行测试实验,结果表明该算法具有较好的全局搜索和局部搜索平衡能力,算法性能得到较大改善。利用本文提出的改进算法对更多复杂函数作进一步研究是下一步的内容。另外,将改进算法应用于一些实际的连续优化问题(如聚类、数据挖掘等)也是下一步的研究方向。

参考文献:

[1] KARABOGA D. An idea based on honey bee swarm for numerical optimization, Technical Report-TR06 [R]. Kayseri: Erciyes University, 2005.

[2] KARABOGA D, BASTURK B. A powerful and efficient algorithm for numerical function optimization : artificial bee colony (ABC) algorithm[J]. Journal of Global Optimization, 2007, 39(3): 459-471.

[3] KARABOGA D, BASTURK B. On the performance of artificial bee colony (ABC) algorithm [J]. Applied Soft Computing, 2008, 8(1):687-697.

[4] KARABOGA D, BASTURK B. A comparative study of artificial bee colony algorithm [J]. Applied Mathematics and Computation, 2009, 214(1): 108-132.

[5] XU Chun-fan, DUAN Hai-bin. Artificial bee colony (ABC) optimized edge potential function (EPF) approach to target recognition for low-altitude aircraft [J]. Pattern Recognition Letters, 2010, 31(13):1759-1772.

[6] KARABOGA D, AKAY B B. Artificial bee colony algorithm on training artificial neural networks [C]//Proc of the 15th IEEE Signal Processing and Communications Applications Conference. New York: IEEE Press, 2007: 818-821.

[7] SINGH A. An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem [J]. Applied Soft Computing, 2009, 9(2): 625-631.

[8] ZHU Guo-pu, KWONG S. Gbest-guided artificial bee colony algorithm for numerical function optimization [J]. Applied Mathematics and Computation, 2010, 217(7): 3166-3173.

[9] GAO Wei-feng, LIU San-yang. Improved artificial bee colony algorithm for global optimization[J]. Information Processing Letters, 2011, 111(17): 871-882.

[10] GAO Wei-feng, LIU San-yang. A modified artificial bee colony algorithm [J]. Computers & Operations Research, 2012, 39(3): 687-697.

[11] BANHARNSAKUN A, ACHALAKUL T, SIRINAOVAKUL B. The best-so-far selection in artificial bee colony algorithm [J]. Applied Soft Computing, 2011, 11(2): 2888-2901.

[12] 暴励,曾建潮. 一种双种群差分蜂群算法[J]. 控制理论与应用, 2011, 28(2): 266-272.

[13] RAHNAMAYAN S, TIZHOOSH H R, SALAMA M M A. Opposition-based differential evolution [J]. IEEE Trans on Evolutionary Computation, 2008, 12(1): 64-78.

[14] WANG Hui, WU Zhi-jian, RAHNAMAYAN S. Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems [J]. Soft Computing, 2011, 15(11): 2127-2140.

[15] KARABOGA D, BAHRIYE A. A modified artificial bee colony algorithm for real-parameter optimization [J]. Information Sciences, 2012,192(1):120-142.

[16] 卢滢宇,王冰,李学文,等. 深层加速搜索的蜂群算法[J]. 计算机应用研究, 2012, 29(12): 4445-4447.

(上接第 1022 页)轮转调度算法,在同优先级进程间采用传统的 RR 算法导致性能不佳。本文在此基础上提出了一种改进的静态优先级驱动轮转调度算法(LC 算法),综合考虑了调度算法的复杂性、调度的性能、公平性和可实现性。模拟实验显示该算法在各种情况下均表现良好,相对于其他轮转调度改进方案具有综合性能优异、易于实现的优点。

参考文献:

[1] BEHERA H S, MOHANTY R, PANDA J. Experimental analysis of a new fair share wighted time slice(FSWTS) scheduling algorithm for real time systems[J]. Journal of Global Research in Computer Science,2011,2(2):54-60.

[2] MOHANTY R, BEHERA H S. Priority based dynamic round robin (PBDRR) algorithm with intelligent time slice for soft real time systems[J]. International Journal of Advanced Computer Science and Applications,2011,2(2):46-50.

[3] MOHANTY R, DAS M. Design and performance evaluation of a new proposed fittest job first dynamic round robin (FJFDRR) scheduling algorithm[J]. International Journal of Computer Information Systems,2011,2(2):23-27.

[4] NOON A, KALAKECH A. A new round robin based scheduling algorithm for operating systems: dynamic quantum using the mean average [J]. International Journal of Computer Science Issues,2011,8(3):224-229.

[5] MATARNEH R J. Self-adjustment time quantum in round robin algorithm depending on burst time of the now running processes[J]. American Journal of Applied Sciences,2009,6(10):1831-1837.

[6] MOHANTY R, BEHERA H S, PATWARI K,*et al*. Design and performance evaluation of a new proposed shortest remaining burst round robin (SRBRR) scheduling algorithm [C]//Proc of International Symposium on Computer Engineering & Technology. 2010:10-15.

[7] BEHERA H S, MOHANTY R. Comparative performance analysis of multi-dynamic time quantum round robin (MDTQRR) algorithm with arrival time[J]. Indian Journal of Computer Science and Engineering,2011,2(2):262-271.

[8] BEHERA H S, MOHANTY R, SAHU S, *et al*. Design and performance evaluation of multi cyclic round robin(MCCR) algorithm using dynamic time quantum[J]. Journal of Global Research in Computer Science,2011,2(2):48-53.

[9] SINGH A, GOYAL P, BATRA S. An optimized round robin scheduling algorithm for CPU scheduling[J]. International Journal on Computer Science and Engineering ,2010,2(7):2383-2385.

[10] STALLINGS W. 操作系统精髓与设计原理[M]. 陈向群,陈渝,译. 北京: 机械工业出版社,2010.

[11] 陈莉君. Linux 内核设计与实现[M]. 北京: 机械工业出版社, 2006.

[12] YASSHUWANTH C, RAMESH R. A new scheduling algorithm for real time system[J]. International Journal of Computer and Electrical Engineering,2010,2(6):1104-1106.