第 38 卷 第 4 期
Vol.38 No.4

计 算 机 工 程
Computer Engineering

2012 年 2 月
February 2012

·开发研究与设计技术· 文章编号：1000—3428(2012)04—0275—03 文献标识码：A 中图分类号：TP316

基于动态抢占阈值的 LSF 调度算法

任小西，赵公怡

(湖南大学信息科学与工程学院，长沙 410082)

摘 要：在最小空闲时间优先(LSF)调度算法中，当任务集中有多个任务的优先级相同或相近时，过多的上下文切换会产生“颠簸”现象，从而大幅增加系统开销。为此，结合 LSF 算法的特点，通过设计合理的动态抢占阈值，提出一种改进的调度算法 DPTLSF。仿真结果表明，改进的算法能够大幅减少“颠簸”现象的发生，降低任务集的截止期错失率。

关键词：实时调度；上下文切换；颠簸；抢占阈值；最小空闲时间优先

Least Slack First Schedule Algorithm Based on Dynamic Preemption Threshold

REN Xiao-xi, ZHAO Gong-yi

(College of Information Science and Engineering, Hunan University, Changsha 410082, China)

【**Abstract**】In Least Slack First(LSF) schedule algorithm, when there are multiple tasks which have the same or similar task priority, too much context switch resulting “bump” phenomenon that will greatly increase the system overhead. Based on preemption threshold, combined with the characteristics of LSF algorithm, through the rational design of dynamic preemption threshold, this paper proposes an improved algorithm named DPTLSF. Simulation results show that the improved algorithm can effectively reduce the “bumps” from occurring and reduce the missed deadline percentage of task set.

【**Key words**】real-time schedule; context switch; bump; preemption threshold; Least Slack First(LSF)

DOI: 10.3969/j.issn.1000-3428.2012.04.090

1 概述

实时系统是一类必须在规定时限内完成对事件处理的计算机应用系统。系统正确性不仅要保证计算结果的正确性，还必须满足任务的截止时间。实时调度是实时系统中重要的研究内容，自从 Liu 和 Layland 在 1973 年提出著名的 RM 调度算法以来，人们对实时调度算法及其应用做了大量研究，提出了许多实时调度算法。最小空闲时间优先(Least Slack First, LSF)调度算法是一种常用的动态优先级实时调度算法，是对最早截止期优先(Earliest Deadline First, EDF)调度算法的改进。最小空闲时间优先算法中任务优先级由任务的预测完成时刻到截止期之间的空闲时间决定，这段空闲时间越小，优先级越高。

由于等待队列中的任务的空闲时间随时间严格递减，因此当前执行任务的空闲时间不变，等待任务的优先级随时可能超过当前执行任务的优先级，产生频繁的任务抢占，降低系统性能，这是抢占式调度算法中普遍存在的问题[1]。

为了减少抢占式多道系统中 CPU 带宽的浪费及内存总开销，Express Logic 公司的 W.Lamie 于 1997 年提出了抢占阈值调度思想，并运用到公司开发的商用实时系统 ThreadX 中。研究证明，抢占阈值思想能够明显减少过多的上下文切换，提高系统资源利用率。

本文针对空闲时间较小时“颠簸”现象对任务集的截止错失率及上下文切换会产生重大影响的问题，设计一种新的抢占阈值确定方法，并在此基础提出 LSF 的改进算法 DPTLSF，通过仿真实验验证了 DPTLSF 算法的改进效果。

2 相关工作

在抢占阈值调度思想提出之前，实时调度算法要么是抢占式的，要么是非抢占式的。抢占式调度算法的优点是注重紧急任务的执行，缺点是造成资源的浪费和较高的任务截止错失率；非抢占式调度算法的优点是较低的任务错失率和较少的上下文切换，缺点是对紧急任务的响应不够及时。

文献[2]证明了在固定优先级、确定任务集的环境下，抢占阈值调度优于抢占式和非抢占式调度。最优的抢占阈值如何确定是抢占阈值思想的核心问题。因为偏大的阈值会导致紧急任务的响应受到很大影响，所以偏小的阈值不能将过多的上下文切换完全消除。文献[3]提出了一种任务集不确定、优先级动态变化的情况下，将抢占阈值思想引入 LSF 中的算法。文献[4]提出一种按比例确定阈值方法。文献[5]提出将优先级继承协议集成到抢占阈值调度中，使得在最坏情况下内容切换次数最少。文献[6]提出抢占阈值由任务裕度和任务重要度 2 个特征参数决定，并用云模型理论进行优化。文献[7]提出了阈值的一种线性计算算法。然而，以上设计的阈值确定算法没有考虑到 LSF“颠簸”现象影响调度性能的根本原因，改进算法的上下文切换次数仍然比较多，任务截止错失率仍然比较高，因此，应该改进 LSF 算法以进一步减少过多

基金项目：中央高校基本科研业务费专项基金资助项目(0148)

作者简介：任小西(1978－)，女，副教授、博士，主研方向：嵌入式系统，编译器；赵公怡，硕士

收稿日期：2011-07-12 **E-mail：**zhaogongyi.jxjian@163.com

的上下文切换，降低任务截止错失率，使 LSF 算法在纯抢占式调度和非抢占式调度之间获得平衡，达到性能最优。

3 LSF 算法

3.1 任务模型

任务集由 N 个相互独立的周期性任务 $\tau_i(1\leqslant i\leqslant N)$构成，任务 τ_i 由六元组$(T_i, C_i, S_i, D_i, P_i, H_i)$确定。其中，$T_i$ 为任务的周期；C_i 为表示任务最坏情况下的执行时间；D_i 为任务的绝对截止期；S_i 为任务的空闲时间；P_i 为任务的优先级；H_i 为任务的抢占阈值。

3.2 LSF 算法的“颠簸”现象

在 LSF 算法中，任务的优先级根据空闲时间来分配，任务在 t 时刻的空闲时间定义如下：

$$S(t)=D_i-t-C_i$$

其中，C_i 为剩余部分任务最坏情况执行时间。空闲时间反映了任务的紧急程度。由上式可知，任务的最小空闲时间随时间变化，对于等待任务，D_i 和 C_i 为常数，所以，任务的空闲时间单调递减；对当前运行的任务，运行前后的空闲时间不变。任务空闲时间的初始值为任务周期减去最坏情况执行时间。空闲时间值小于 0 表示任务已经无法按时完成，应该被夭折。调度时，调度器选择具有最小空闲时间的任务运行，若多个任务的最小空闲时间相同，则选择绝对截止期小的任务运行。当任务集中有 2 个以上的任务具有相同或相近优先级时，运行过程中空闲时间的变化容易导致过多的上下文切换，产生“颠簸”现象。例如，假设对于给定的 3 个任务 τ_1、τ_2、τ_3，各参数如下：

C_1=5, D_1=12, S_1=7；

C_2=6, D_2=13, S_2=7；

C_3=7, D_3=14, S_3=7。

由 LSF 调度算法可知 τ_1、τ_2、τ_3 交替执行，在每个时间片都发生抢占，其运行情况如图 1 所示。

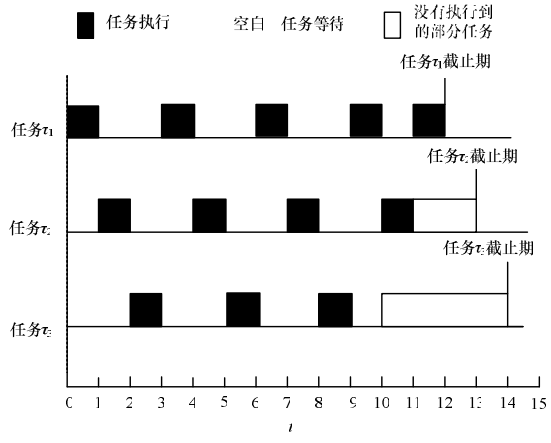

图 1 LSF 调度

由图 1 可见，运行过程中产生了过多的抢占，而且 3 个任务因为均分 CPU 造成 2 个任务错过了截止期，时间区间(0, 12)内 CPU 有效利用率仅为 41.7%，因此，任务集的截止错失率与空闲时间比较小时过多的任务抢占有直接关系；如果几个任务的空闲时间比较大时发生激烈的 CPU 抢占，虽然也会造成过多任务上下文切换，但是对任务集截止错失率的影响比较小，因为即使均分 CPU，仍然能比较从容地完成任务，这在后面的仿真实验结果中也得到了证明。在 LSF 算法中，所有错失截止期的任务都是在最小空闲时间很小的情况下没有竞争到 CPU 而夭折的，因此，趋于零的一段空闲时间区间内的频繁抢占是造成 LSF“颠簸”现象的主要原因，也是增大任务错失率的一个重要原因。

3.3 抢占阈值策略

抢占阈值策略通过合理设置抢占阈值，充分利用了抢占调度和非抢占调度的优点。特别地，当阈值取当前任务优先级时，是抢占调度；当阈值取足够大时，是非抢占调度。可见，抢占调度和非抢占调度只是抢占阈值调度中的特例。将抢占阈值策略引入 LSF 算法，分析上面的例子：约定优先级值越大，优先级越高(以下不再赘述)。

假如任务优先级大于某个值时(对应空闲时间为 7 单位时的优先级)，其抢占阈值设为无穷大或者任务集最大优先级，LSF 算法蜕化为非抢占调度，则运行情况如图 2 所示。可见，设置合理的抢占阈值后，任务连续执行，抢占次数为 0，只有一个任务错失截止期，时间区间(0, 12)内 CPU 有效利用率为 100%，效果显著。

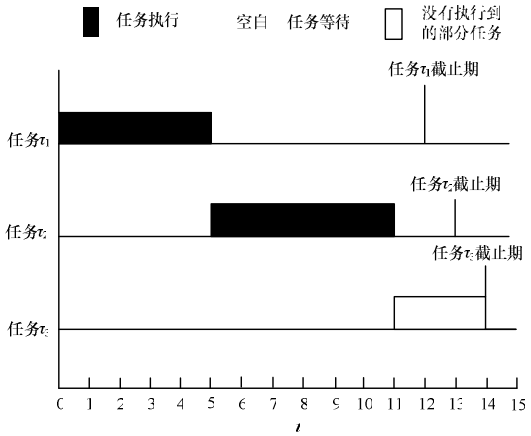

图 2 基于抢占阈值的 LSF 调度

4 LSF 算法改进

为了解决“颠簸”现象带来的频繁上下文切换及任务截止错失率问题，需要改进 LSF 算法。根据上面的分析，减少上下文切换次数可以采用抢占阈值思想，每个任务除了分配优先级外，还要分配一个抢占阈值，如果其他任务要抢占当前任务，不但要求优先级大于当前任务优先级，还必须满足优先级大于当前任务的抢占阈值。对于过度抢占带来的截止错失率问题，根据上面的分析，可以在趋于零的很小一段空闲时间区间设置对应的抢占阈值为无穷大或任务集中最大优先级，以保证任务的顺序执行，不至于因为抢占使各任务均分 CPU 资源，导致较高的任务截止错失率。DPTLSF 算法基于下面方案 2 的阈值确定方法，仍然采用 LSF 的优先级确定方法，调度时根据优先级与阈值的比较判定是否抢占，若当前执行任务完成，则选择优先级最大的任务获得 CPU。

4.1 优先级的分配

在 LSF 算法中，优先级仅由任务的空闲时间决定，空闲时间越小，优先级越大。假设空闲时间为 0 时对应的优先级为最大值 P_max，空闲时间为最大值 L_max 时对应的优先级为 0。不失一般性，设计如下优先级分配函数：

$$P=kL+b, L\in[0, L\_max], k=-\frac{P\_max}{L\_max}, b=P\_max$$

不考虑空闲时间小于 0 的情况，由调度器自动夭折这些任务。

4.2 抢占阈值的确定

本文提出 2 种确定抢占阈值的方案：方案 1 主要考虑解

决过多上下文切换问题，采用一种简单的阈值设计方法，这是一种普通的改进方案，与文献[6]提出的算法相似，在本文中主要用来与方案 2 进行比较分析。方案 2 在方案 1 的基础上，设置在空闲时间较小区间内的抢占阈值为最大优先级，这是 DPTLSF 算法的关键。

方案 1 与优先级分配一样，设计抢占阈值与空闲时间满足线性关系，即：

$$P\_gate = -\frac{P\_max - M}{L\_max} L + P\_max$$
$$L \in [0, L\_max] \tag{1}$$

其中，P_gate 表示抢占阈值；L 为空闲时间；M 为当 $L=L\_max$ 时对应的阈值。

方案 2 考虑在某个很小的空闲时间范围$[0, u)$内，将抢占阈值设计为最大优先级，即在该空闲时间范围内，任务不可抢占。因此，抢占阈值与空闲时间之间满足如下函数：

$$P\_gate = \begin{cases} P\_max, L \in [0, u] \\ kL + b, L \in (u, L\_max] \end{cases} \tag{2}$$

其中，$k = \dfrac{P\_max - M}{u - L\_max}$；$b = P\_max - ku$；$M$ 为当 $L=L\_max$ 时 P_gate 的值；$[0, u]$为非抢占区间。

5　性能仿真

5.1　仿真条件

任务集中任务个数 N=5，且都为周期性任务，仿真运行持续时间为 1 000 个时间片。各任务的最坏情况执行时间 C_i 在区间[1, 10]内服从均匀分布。任务周期 T_i 按照公式 $T_i = N \times C_i / p$ 计算。其中，p 表示期望的工作负载，P_max=50，L_max=40。

5.2　仿真结果与分析

(1)仿真实验 1

令方案 1 中 M=10，方案 2 中 M=0，u=5，图 3、图 4 为基于方案 1 与方案 2 的动态抢占阈值 LSF 算法及已有的 LSF 算法在不同负载时对应的任务截止错失率与任务抢占次数曲线图。由图 3 可知，在不同负载情况下，基于方案 1 的算法与已有的 LSF 算法的任务错失率基本一样；而基于方案 2 的 DPTLSF 算法始终保持更低的任务错失率。因此，DPTLSF 算法在降低任务截止错失率方面有很大改进。图 4 表明，在不同负载情况下，基于方案 1 与方案 2 的算法都减少了任务抢占次数，DPTLSF 的任务抢占次数远少于其他 2 种算法。

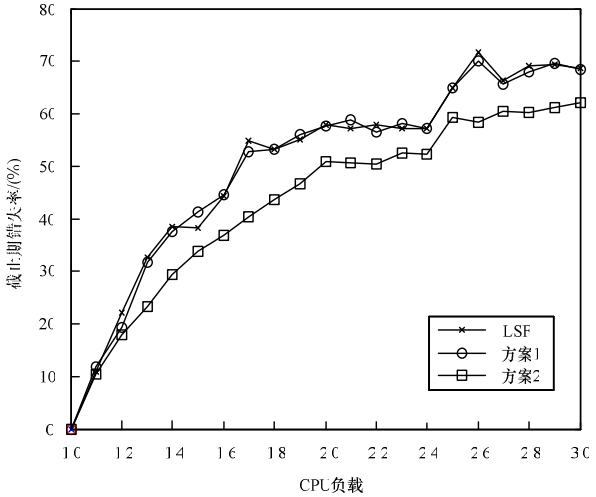

图 3　不同负载情况下的任务截止期错失率

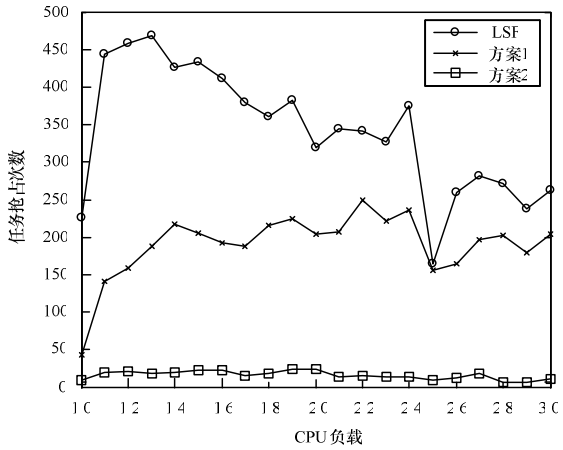

图 4　不同负载情况下的任务抢占次数

(2)仿真实验 2

令负载 ρ =1.5，非抢占区间右端点值 u=5，对式(1)和式(2)中最小阈值 M 取不同值进行实验。实验结果得出，基于方案 1 的 LSF 算法随 M 值增大，任务截止错失率先是有所增大，而后一直减小。然而基于方案 2 的 DPTLSF 算法受 M 的影响不大；基于 2 种阈值设定方案的改进算法都能够减少任务的抢占次数，特别是 DPTLSF 算法，对于不同的 M，其抢占次数平稳地维持在非常低的水平。

(3)仿真实验 3

令负载 ρ =1.5，M=0，对于方案 2，非抢占区间右端点值 u 分别取[0, 5]之间的整数，实验结果如图 5 所示。图中，MDP 为截止期错失率，CSN 为任务抢占次数。随着 u 值增加，任务截止错失率急剧下降。当 u=0 时，DPTLSF 算法蜕化为方案 1 的算法，任务错失率最高。当 u=3 时，MDP 最小，说明趋于 0 的较小空闲时间区间内的抢占对 LSF 算法的性能具有决定性作用。图 5 说明了任务切换次数随 u 值增大而迅速减小。

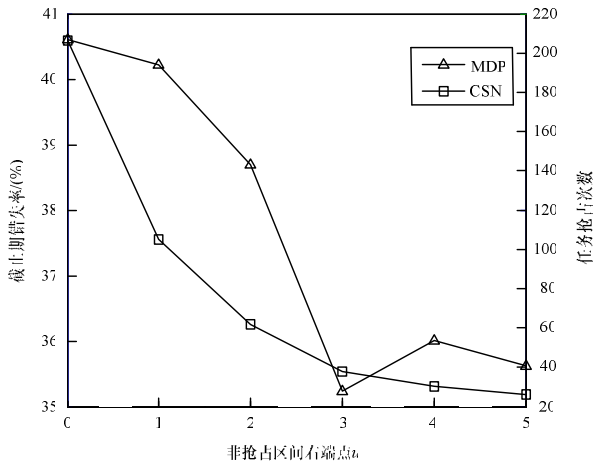

图 5　不同 u 值下的截止期错失率和任务抢占次数

6　结束语

本文针对 LSF 调度算法中的“颠簸”现象，分析了其产生原因，并通过仿真实验证明了在空闲时间较小时发生的“颠簸”现象对 LSF 算法的截止错失率及任务抢占次数起决定作用。然后基于抢占阈值思想提出 DPTLSF 算法。仿真结果表明，DPT LSF 调度算法在保留 LSF 调度算法优点的基础上，明显减少了上下文切换次数，降低了任务截止期错失率，提高了 CPU 有效利用率。

(下转第 280 页)