基于负载执行紧迫度的实时补偿任务调度策略 TSCTTL

夏家莉[1] 曹重华[1] 王文乐[2] 陈 辉[1]
(江西财经大学软件与通信工程学院 南昌 330032)[1] (江西师范大学软件学院 南昌 330022)[2]

摘 要 针对支持补偿性的实时任务模型,分析实时任务的系统负载执行紧迫度,进而提出基于负载执行紧迫度的实时补偿任务调度策略 TSCTTL;通过实验仿真表明,依据实时任务的负载执行紧迫度来调度补偿任务,降低了系统任务的截止期错失率,并提高了系统收益。

关键词 阈值抢占,补偿任务,硬实时任务,任务调度
中图法分类号 TP311.1,TP316.2 **文献标识码** A

Real-time Task Scheduling Strategy Based on Load Execution Urgency

XIA Jia-li[1] CAO Zhong-hua[1] WANG Wen-le[2] CHEN Hui[1]
(School of Software and Communication Engineering, Jiangxi University of Finance & Economics, Nanchang 330032, China)[1]
(School of Software, Jiangxi Normal University, Nanchang 330022, China)[2]

Abstract For the compensatory support real-time task model, this paper analyzed real-time task system load execution degree of urgency, then put forward real-time compensation task scheduling strategy TSCTTL based on the load execution degree of urgency. The simulation results show that scheduling compensation tasks according to the load execution degree of urgency of the real-time tasks reduces the system task deadline miss ratio and improves the system returns.

Keywords Threshold preemption, Compensation task, Hard real-time tasks, Task scheduling

1 引言

嵌入式实时系统广泛应用于工业、航天军工的各种控制设备的智能数据处理系统中。实时任务分为硬实时任务、软件实时任务、固实时任务,实时任务调度的目的是为了保证所有硬实时任务足截止期,而且尽量降低软实时任务的截止期错失率。

为了提高嵌入式实时系统的自适应性,Nag 等[1]将任务分为两个部分:主任务和补偿任务,当主任务不能成功执行时由其补偿任务使之安全结束。但是补偿任务不具备实时性,对于实时应用存在局限性。夏家莉[2]提出了基于替代/补偿的事务模型 ACTM,并提出了针对 ACTM 的实时调度策略,使补偿任务具有实时性,消除了实时任务所造成的外部影响。

基于抢占阈值的调度策略有利于减少无效抢占,并节约了系统资源,提高了系统成功率。金宏等[3]针对不确定的任务集和任务优先级情况,提出动态阈值的调度策略,来降低任务截止期错失率。李琦等[4]针对软实时任务提出了两种基于动态模糊阈值的调度策略,即通过调整任务截止期至动态阈值来提高任务完成率。

本文针对 ACTM 模型中实时任务的特点,分析任务的相对负载对任务执行紧急度的影响,并结合抢占阈值方法,提出了一种基于负载执行度的补偿任务调度策略 TSCTTL(Task Scheduling for Complement Task Based on Task Load)。该策略提高了系统资源利用率,并提高了任务执行的成功率。实验仿真中,采用截止期最早最优先算法 EDF (Earliest Deadline First)[5]、空余时间最短者最优先 LSF (Least Slack First)[6],对补偿任务与主任务捆绑调度、主任务夭折后调度补偿任务、基于任务的负载执行紧迫度的补偿调度 3 种调度策略进行了实验对比,结果表明根据系统的任务负载执行度来调度补偿任务,能够提高系统成功率和系统收益。

2 任务模型

假设实时系统中实时任务集合为 $T=\{T_1, T_2, \cdots, T_n\}$,对于存在补偿的硬任务 T_i,它由主任务 Tk_{im} 与补偿任务 Tk_{ic} 组成;没有补偿的任务 T_i 只包含主任务 Tk_{im},Tk_{im} 与 Tk_{ic} 可称为 T_i 的子任务,Tk_{im} 与 Tk_{ic} 成为调度的对象,任务之间不存在冲突;因此当前系统中实时任务集可以表示为:

$TK=\{TK_i \mid TK_i \in (TK_{im}, TK_{ic}), 0<i<n\}$

对于每一个子任务 TK_i,它有如下属性:

D_i:任务 TK_i 的截止期;

A_i:任务 TK_i 的到达时间;

c_i:任务 TK_i 的估计执行时间;

τ_i:任务 TK_i 的已执行时间;

V_i:任务 TK_i 的价值,表示任务成功执行将给系统带来

到稿日期:2013-04-22 返修日期:2013-07-25 本文受国家自然科学基金:基于替代/补偿的并发控制机制研究(60763002)资助。
夏家莉(1965—),女,博士,教授,主要研究方向为实时数据库、软件工程、商务智能;曹重华(1977—),男,博士生,讲师,主要研究方向为实时数据库、软件工程;王文乐(1985—),男,博士,讲师,主要研究方向为实时数据库、实时调度,E-mail:wwle1985@126.com;陈 辉(1976—),男,博士,副教授,主要研究方向为实时数据库、数据挖掘。

的收益。

在 t_0 时刻，根据任务在系统中的执行状态，可以将系统中的实时任务划分为如下几类：

- 执行任务，表示正在占用 CPU 的任务；
- 就绪任务，表示因优先级低而处于等待 CPU 执行的任务；
- 等待任务，表示已经到达，但因相对负载执行度低而未加入调度队列的补偿任务。

3 负载紧迫度分析

实时任务 Tk_j 的当前可用于执行的时间 D_j-t_0 越接近于剩余执行时间 $C_j-\tau_i$，任务的执行紧迫度越强，如果系统负载低，没有其他任务与 T_{ij} 竞争，即使 D_j-t_0 等于 $C_j-\tau_i$，任务也能保证执行完；如果负载高，D_j-t_0 即使很长，任务执行时间也可能被其他任务抢占，从而影响任务的成功提交。因此实时任务的调度执行不仅与任务本身的属性截止期与剩余执行时间有关，还与系统的当前负载密切相关。因此，引入任务执行紧迫度的概念。

假设系统中当前就绪任务集为 $TKR=\{Tk_1, Tk_2, \cdots, Tk_n\}$，对于任务 Tk_j，其截止期为 D_j，由截止期小于 D_j 的就绪任务组成的集合记为 $TKR_{<j}=\{Tk_i \mid Tk_i \in TKR \wedge D_i \leqslant D_j \wedge Tk_i \neq Tk_j\}$。

定义 1 在 t_0 时刻，系统从当前时刻 t_0 到任务 Tk_j 的截止期 D_j 这段时间，执行完 Tk_j 和 $TKR_{<j}$ 中所有任务所需时间的总和与系统可用执行时间 D_j-t_0 的比值称为任务 Tk_j 的执行紧迫度，记为 LU_{Tk_j}：

$$LU_{Tk_j}=\frac{\left(\sum_{Tk_i \in TKR_{<j}}(c_i-\tau_i)\right)+(c_j-\tau_j)}{D_j-t_0}$$

3.1 系统相对超载时紧迫度分析

定理 1 在 t_0 时刻，若系统对于任务 Tk_j 的执行紧迫度大于 1，则任务 Tk_j 的执行紧迫度 LU_{Tk_j} 随等待时间增加而增大。

证明：假设在 t_0 时刻，任务 Tk_j 的执行紧迫度为 $LU_{Tk_j}=\frac{\left(\sum_{Tk_i \in TKR_{<j}}(c_i-\tau_i)\right)+(c_j-\tau_j)}{D_j-t_0}>1$，到时刻 t_1 $(t_1<D_j)$，任务 Tk_j 累积等待了 x 个时间单位，其中 $y(0\leqslant y\leqslant x)$ 个时间单位用于执行 $TKR_{<j}$ 中的任务，$x-y$ 个时间单位用于执行 $TKR_{<j}$ 外的任务。

如果 $x>D_j-t_0-(c_j-\tau_j)$，则任务 Tk_j 的空余时间少于剩余执行时间，任务 Tk_j 将夭折，因此等待时间 x 必须满足 $0<x\leqslant D_j-t_0-(c_j-\tau_j)$。

I. 首先考虑在 t_0 至 t_1 时间段没有新的任务到达的情况：

在 t_1 时刻，Tk_j 的执行紧迫度 $LU_{Tk_j}=\frac{\left(\sum_{Tk_i \in TKR_{<j}}(c_i-\tau_i)\right)+(c_j-\tau_j)-y}{D_j-t_0-x}$，令 $f(x)=\frac{\left(\sum_{Tk_i \in TKR_{<j}}(c_i-\tau_i)\right)+(c_j-\tau_j)-y}{D_j-t_0-x}$，再令 $a=D_j-t_0$，$b=\left(\sum_{Tk_i \in TKR_{<j}}(c_i-\tau_i)\right)+(c_j-\tau_j)$，显然有 $a>0, b>0, x>0, a>x, b>x$，可得 $f(x)=\frac{b-y}{a-x}>0$，$f'(x)=\frac{b-y}{(a-x)^2}$，因 $LU_{Tk_j}=\frac{\left(\sum_{Tk_i \in TKR_{<j}}(c_i-\tau_i)\right)+(c_j-\tau_j)}{D_j-t_0}>1 \Rightarrow b>a$，而 $0<x\leqslant D_j-t_0-(c_j-\tau_j) \Rightarrow a\geqslant x$，又有 $0\leqslant y\leqslant x$，得 $b>y$，所以有 $f'(x)>0$，可知 $f(x)$ 为增函数，得 LU 关于 x 递增。

II. 当 t_0 至 t_1 时间段有截止期小于 D_j 的新的任务到达，则 $\sum_{Tk_i \in TKR_{<j}}(c_i-\tau_i)$ 增大，执行紧迫度增大，再由 I 可知，LU 关于 x 递增。

III. 当 t_0 至 t_1 时间段有截止期大于 D_j 的新的任务到达，则不影响 Tk_j 的相对执行紧迫度，证明同 I，因此，LU 也关于 x 递增。

故原命题得证。

若任务 Tk_j 的执行紧迫度 LU_{Tk_j} 大于 1，则说明系统不能满足 $TKR_{<j}$ 中所有任务的截止期，必须要有任务夭折，且紧迫度越大，任务夭折的概率就越大。由定理 1 可知，任务 Tk_j 的负载紧迫度随等待时间增大而增大，当 LU_{Tk_j} 大于一个系统给定的阈值 $\mu(\mu>1)$ 时，就夭折 $TKR_{<j}$ 中优先级最小的主任务 $Tk_{\min}$，如果 $Tk_{\min}$ 已开始执行，且存在对应的补偿任务，则立即启动其对应的补偿任务调度执行；如果 $Tk_{\min}$ 从未执行，则它不需要执行补偿，直接夭折。

3.2 系统相对不超载时紧迫度分析

定理 2 在 t_0 时刻，若系统对于任务 Tk_j 的相对执行紧迫度小于等于 1，在累积等待时间 $x(0<x<D_j-t_0-(c_j-\tau_j))$ 时间内没有新任务加入调度队列，且等待时间用于执行 $TKR_{<j}$ 中的任务，则任务 Tk_j 相对负载执行度 LU_{Tk_j} 随等待时间增加而变小。

证明：假设在 t_0 时刻，任务 Tk_j 的执行紧迫度为 $LU_{Tk_j}=\frac{\left(\sum_{Tk_i \in TKR_{<j}}(c_i-\tau_i)\right)+(c_j-\tau_j)}{D_j-t_0}<1$，到时刻 t_1 $(t_1<D_j)$，累积等待了 x 个时间单位；如果 $x>D_j-t_0-(c_j-\tau_j)$，则任务 Tk_j 的空余时间少于剩余执行时间，任务 Tk_j 将夭折，因此必须满足 $0<x<D_j-t_0-(c_j-\tau_j)$。

因在 t_0 至 t_1 时间段没有新的任务到达，若所等待的 x 个时间单位全部用于执行 $TKR_{<j}$ 中的任务，则在 t_1 时刻，Tk_j 的负载执行度 $LU_{Tk_j}=\frac{\left(\sum_{Tk_i \in TKR_{<j}}(c_i-\tau_i)\right)+(c_j-\tau_j)-x}{D_j-t_0-x}$，令 $f(x)=\frac{\left(\sum_{Tk_i \in TKR_{<j}}(c_i-\tau_i)\right)+(c_j-\tau_j)-x}{D_j-t_0-x}$，再令 $a=D_j-t_0$，$b=\left(\sum_{Tk_i \in TKR_{<j}}(c_i-\tau_i)\right)+(c_j-\tau_j)$，显然有 $a>0, b>0, x>0, a>x, b>x$，可得 $f(x)=\frac{b-x}{a-x}>0$，$f'(x)=\frac{b-a}{(a-x)^2}$，因相对负载小于 1，所以有 $b-a<0$，$f'(x)<0$，因此 $f(x)$ 为减函数，得 LU_{Tk_j} 关于 x 递减。原命题得证。

定理 2 说明，任务 Tk_j 在 t_0 时刻的相对执行紧迫度小于 1 时，系统将有足够的时间执行完 $TKR_{<j}$ 中的任务，而且还会有剩余时间。如在 D_j-t_0 时间段里继续执行 $TKR_{<j}$ 中的任务，任务 Tk_j 的相对负载执行度将一直降低，如果 Tk_j 为主任务，则它总有足够的时间执行完，其对应的补偿任务可以推迟启动；如果 Tk_j 为补偿任务，它也有足够的时间执行完，为了节约系统资源，它可以等待对应的主任务夭折后才启动。

定理 3 在 t_0 时刻，若系统对于任务 Tk_j 的相对执行紧迫度小于等于 1，在累积等待时间 $x(0<x<D_j-t_0-(c_j-\tau_j))$ 时间内没有新任务加入调度队列，且等待时间用于执行集合 $TKR_{<j}$ 外的任务，则任务 Tk_j 相对负载执行度 LU_{Tk_j} 随

等待时间增加而变大。

证明：若 t_0 至 t_1 时间段没有新的任务到达，且所等待的 x 个时间单位全都不用于执行 $TKR_{<j}$ 中的任务，则在 t_1 时刻，Tk_j 的负载执行度 $LU_{Tk_j}=\frac{(\sum_{Tk_i\in TKR_{<j}}(c_i-\tau_i))+(c_j-\tau_j)}{D_j-t_0-x}$，令 $f(x)=\frac{(\sum_{Tk_i\in TKR_{<j}}(c_i-\tau_i))+(c_j-\tau_j)}{D_j-t_0-x}$，再令 $a=D_j-t_0$，$b=(\sum_{Tk_i\in TKR_{<j}}(c_i-\tau_i))+(c_j-\tau_j)$，显然有 $a>0,b>0,x>0$，$a>x,b>x$，则 $f(x)=\frac{b}{a-x}>0$，$f'(x)=\frac{b}{(a-x)^2}$，所以 $f(x)$ 为增函数。原命题得证。

定理 3 说明，即使系统对于任务 Tk_j 的相对负载紧迫度小于 1，若系统执行 $TKR_{<j}$ 集合外的任务，任务 Tk_j 的相对执行紧迫度也将随等待时间而增大。当 LU_{Tk_j} 到达 1 时，则必须执行 $TKR_{<j}$ 中的任务，否则 $TKR_{<j}$ 中将会有任务因错失截止期而夭折。如果 Tk_j 为补偿任务且没有加入调度执行，则当 LU_{Tk_j} 到达 1 时，Tk_j 就应加入调度执行，因为在支持补偿任务的系统中补偿任务优先级将大于任何主任务的优先级，所以即使当 $LU_{Tk_j}=1$，系统也能保证补偿任务 Tk_j 满足其截止期。

4 补偿任务调度时机分析

对于可补偿的实时任务而言，任务成功执行意味着主任务执行完成，该任务成功提交；或主任务不成功但其补偿任务执行完成，该任务安全结束。本节将讨论实时系统中补偿任务的调度时机问题。

当一个可补偿的实时任务到达时，如果补偿任务与主任务一起参与调度，那么补偿任务执行需要占用系统资源，而主任务并不一定会夭折，这将导致资源浪费并影响其他任务的执行，将导致错失截止期的任务增加。主任务夭折后，如果补偿任务的相对执行紧迫度小于 1，为了提高系统任务的成功率，可以考虑推迟执行补偿任务，而先执行相对执行紧迫度大的任务。因此，当存在补偿任务的实时任务到达时，如果补偿任务的相对执行紧迫度低，我们可先不考虑启动补偿任务，而调度实时任务的主任务，当补偿任务的相对负载执行度达到一定值时，才将其加入调度队列，与就绪任务一起调度执行。

任务的执行紧迫度越大，其错失截止期的概率越高，如果 Tk_j 为主任务，为了保证任务的成功执行，其补偿任务越早调度执行越好；同样地，如果 Tk_j 为补偿任务，也应该尽快调度执行。

又因为任务的估计执行时间 C_j 一般取的是事务的最坏执行时间，它大于任务的实际需执行时间，为了提高实时系统的任务执行成功率，实时系统应允许一定的超载，设系统的补偿任务启动阈值为 $\mu(\mu>1)$。任务的补偿任务起动策略如下：

设系统当前就绪任务集为 $TKR=\{Tk_1,Tk_2,\cdots,Tk_n\}$，就绪集 TKR 中的任务按截止期的升序组成队列 RQ，对于任务 Tk_j，其截止期为 D_j，$TKR_{<j}$ 集合就是 RQ 队列中在任务 Tk_j 之前的任务。按截止期升序计算队列 RQ 中各任务的执行紧迫度 LU_{Tk_j}，根据 LU_{Tk_j} 的取值不同，系统进行补偿的调度时机不同，进行以下的补偿处理：

I. 若 $LU_{Tk_j}\leqslant\mu$，表示系统有充裕的时间保证集合 $TKR_{<j}$ 中的任务执行成功，任务成功执行的概率高，由定理 2 分析可知，$TKR_{<j}$ 中的主任务对应的补偿任务不需要立即启动执行，只有当主任务因意外夭折后才需启动补偿任务。

II. 若 $LU_{Tk_j}>\mu$，则根据定理 1 的分析，选择 $TKR_{<j}$ 中优先级最小的任务 Tk_{min} 夭折，如果 Tk_{min} 已开始执行，且存在对应的补偿任务，则需要对其进行补偿，若 $LU_{Tk_{min\_c}}<1$，则根据定理 3 的分析，可推迟起动 Tk_{min_c}，一直到 $LU_{Tk_{min\_c}}=1$ 时调度执行 Tk_{min_c}；若 Tk_{min} 还未开始执行，它对外界也不会产生后果，所以不需要对其进行补偿。

5 TSCTTL 调度算法

本节根据任务的执行紧迫度，给出支持补偿任务的基于任务负载执行紧迫度的任务调度策略 TSCTTL(Task Scheduling for Complement Task Based on Task Load)。TSCTTL 策略的具体算法如下：

(1)按就绪任务截止期的升序计算 TKR 中各就绪任务的紧迫度 LU；

(2)如果一个任务 Tk_j 的执行紧迫度 LU_{Tk_j} 大于系统给定的补偿任务启动阈值 $\mu(\mu>1)$，则从 $TKR_{<j}$ 中选出优先级最低的任务 Tk_k，并夭折任务 Tk_k；

(4)如果 Tk_k 存在对应的补偿任务 Tk_{kc}，则将补偿任务 Tk_{kc} 加入调度队列；

(5)重新计算 LU_{Tk_j}，如果 $LU_{Tk_j}\leqslant\mu$，转步骤(1)，否则转步骤(2)；

(6)执行优先级最高的任务。

(7)重复步骤(1)。

TSCTTL 调度的算法如下：

pri(Tk_j)：任务的优先级；

μ：任务紧迫度阈值；

```
Procedure LU(Tkj) //任务紧迫度计算过程
Begin
  compute LUTkj = ((Σ Tki∈TKR<j (ci−τi))+(cj−τj)) / (Dj−t0−x);
  return LUTkj;
End
Procedure: TaskScheduing //任务调度
Begin
  For Tkj in TKR
    If (LUTkj > μ) then
  Begin
    Tkmin = Tkj;
    For Tkj in TKR<j
      If (pri(Tkk) < pri(Tkmin) then
        Tkmin = Tkk;
        Abort Tkmin;
        if(isExist (Tkmin, Tkmin_c)) then//Tkmin存在对应的补偿任务 Tkmin_c
          scheduling Tkmin_c;
    End
  execute max(pri(Tkj in TKR)) //执行优先级最大的任务单位时间
End
```

6 实验仿真

6.1 实验环境及性能指标

仿真实验环境的 CPU 为双核 3.2GHz，内存为 2GB，操作系统为 Linux。实验中的参数条件如下所示：

①任务 T_i 的最坏情况执行时间 c_j 在 5～10 个单位时间内随机取值；

②任务 T_i 的周期 p_i 按照公式 $p_i=N*c_i/L$ 计算，其中 N 为一周期中任务集中的总任务数，L 为系统期望产生的工作负载；

③任务实例 T_{ij} 的周期和最坏情况执行时间分别为 P_{ij} 和 C_{ij}，其截止期 $D_{ij}=P_{ij}$，到达时间 $A_{ij}=P_{i(j-1)}$，一个任务实例代表一个任务；

④系统中具有补偿任务的实时任务的概率为 0.1，补偿任务的最坏情况执行时间 c_j 在 1～3 个单位时间内随机取值，补偿任务的截止期为其对应的主任务截止期 $+c_j*\mathrm{round}(L+1)$；

在仿真实验中，用最坏情况执行时间作为任务的实际执行时间，所有仿真结果都是由 100 次独立仿真实验获得的统计平均值，实验运行的持续时间为 5000 个时间单位。

实验分别采用截止期最早最优先的优先级分派策略的支持补偿的实时任务调度策略（EDF-TSCTTL）、空闲时间最短最优先的优先级分派策略的支持补偿的实时任务调度策略（LSF-TSCTTL），及传统的 EDF 与 LSF 优先级实时任务调度策略进行实验；并将系统任务截止期错失率 MDR（Miss Deadline Ratio：系统中主任务与补偿任务均失败的任务数目与系统中所有任务总数的比值）作为评价性能指标。

6.2 性能分析

在本小节中，我们首先比较了 4 种算法在静态系统中的性能，然后又比较了其在动态系统中的性能。

（1）特定负载下，不同 μ 值对 MDR 的影响

取每周期任务个数 $N=20$ 和系统期望产生的工作负载 $L=1.5$，图 1 是阈值 μ 在区间[1,2]（X 轴）对系统截止期错失率（Y 轴）的影响情况。

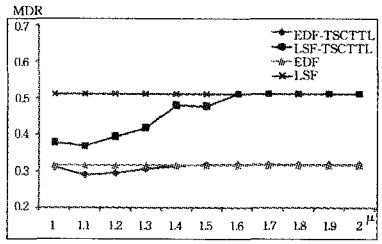

图 1 不同阈值 μ 下系统 MDR 变化图

从图 1 中可知，TSCTTL 调度策略优于同种类型的优先级分派策略，因此采用 TSCTTL 调度策略可有效降低系统的截止期错失率。当 μ 大于系统负载 L 时，采用 TSCTTL 调度策略的系统截止期错失率等于传统的调度策略的截止期错失率，因为把 μ 值设为大于系统负载 L 时，任何任务的相对负载都小于系统负载 L，μ 值就不起作用，TSCTTL 调度策略就等同于传统的实时任务调度策略。

（2）不同的工作负载下采用 EDF 优先级算法

取 $N=5$，采用截止期最早最优先（EDF）的优先级分派算法进行调度，系统期望产生的工作负载分别在 0.5～3 取值，在每一个工作负载条件下，当 $\mu=1.4$ 时，EDF-TSCTTL 的 MDR 最小，小于采用 EDF 调度算法的，说明采用 TSCTTL 算法能降低系统的截止期错失率，满足 $\mu<L$；而当 $\mu=2.0$ 时，EDF-TSCTTL 的 MDR 最大，MDR 大于采用 EDF 优先级分派的传统调度算法的，这是因为 μ 值接近于系统负载 L 时，可能会使能成功的任务提前夭折，从而增加了系统的截止期错失率。由图 2 可知，对比不同负载下的补偿任务的启动阈值 μ，在系统过载时应满足 $\mu<L$，因为 MDR 值与系统负载 L 紧密相关，而 TSCTTL 策略可以控制 L 至阈值 μ，在超载时及时夭折不能完成的任务；而若 $\mu=1.0$，则会使系统负载 L 小于 1，导致可能完成的任务夭折了。为了充分利用系统资源，系统应使阈值 μ 在区间 $(1,L)$ 之内。因此在特定的负载下，只要 μ 值设置得当，系统采用 TSCTTL 策略的 MDR 值小于 EDF 策略的，可获得更好的性能。

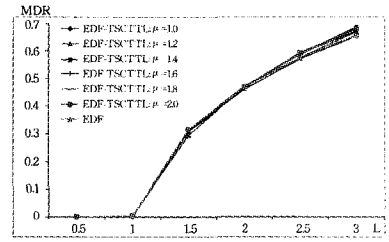

图 2 EDF 策略 MDR 对比图

（3）不同的工作负载下采用 LSF 优先级算法

对于采用 LSF 的优先级调度策略的实验任务属性，我们采用与 EDF 优先级调度策略相同的任务属性。获得的截止期错失率变化图如图 3 所示。由图 3 可知，在系统工作负载小于 1 时，MDR 都接近于零；同样在系统工作负载为 2.5 时，当 $\mu=1.4$，LSF-TSCTTL 的 MDR 最小，小于采用 LSF 优先级分派的传统调度算法的，采用 LSF-TSCTTL 调度策略对比 LSF 调度策略能降低系统的截止期错失率；而当 $\mu=2.0$ 时，LSF-TSCTTL 的 MDR 最大。

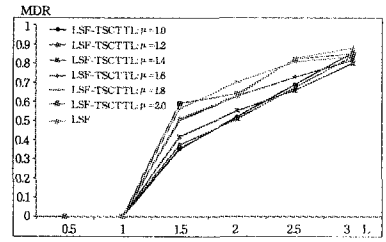

图 3 LSF 策略 MDR 变化图

对比图 2 与图 3，在同一系统工作负载下，MDR 最小值的 μ 值相同，MDR 最大值的 μ 值也相同，说明 μ 值与系统工作负载紧密相关，而与是否采用 EDF 或 LSF 的关系不大，这是因为 TSCTTL 调度算法与负载紧密相关，而 EDF 与 LSF 优先级分派算法与负载不存在直接关系。

结束语 本文给出了实时任务的系统相对执行紧迫度，研究了实时任务执行紧迫度与等待时间的关系，并基于任务的相对负载执行度讨论了实时任务的补偿任务的调度时机；然后将 EDF 优先级分派算法、LSF 优先级分派算法分别集成到 TSCTTL 调度策略中。仿真实验显示：在系统超载时，μ 值与系统工作负载紧密相关，系统在某一特定负载下存在一个恰当的 μ 值，使得支持补偿任务的实时系统采用 TSCTTL

（下转第 225 页）

(1)判断当前对象是否是顺序图中的对象；

(2)判断该对象 Obj_i 当前交互信息 m_j 是否是结束信息；

(3)判断该信息是否是并发/选择行为；

(4)如果该信息是并发/选择行为，则相应地增加子进程标识 $Temp_k$；

(5)否则就是顺序行为，则将该信息顺序添加到 FSP 模型；

(6)递增到下一个信息。

3.2 接车进路用例的形式化模型生成

接车进路用例对应的各对象的 FSP 进程代数模型如下：

Signaller=(m_1 −> when (c1) m_{22} | when (!c1&c2&!c3&c4)m_{10} | when(!c1&c2&c3)m_{11} | when(!c1&(c2&!c3&!c4|!c2)&c5)m_{16} | when(!c1&(c2&!c3&!c4|!c2)&!c5&c6)m_{18} | when(!c1&(c2&!c3&!c4|!c2)&!c5&!c6)m_{17} −>m_{21})

Interlocking System=(m_1 −>when(c1)m_{22} | $Temp_1$)

$Temp_1$=(m_2 −>m_3 −>when(c2)$Temp_{11}$ | $Temp_{14}$)

$Temp_{11}$=(m_4 −>m_5 −>when(!c3)$Temp_{12}$ | m_{11})

$Temp_{12}$=(m_6 −>m_7 −>when(!c4)$Temp_{13}$ | m_{10})

$Temp_{13}$=(m_8 −>m_9 −>$Temp_{14}$)

$Temp_{14}$=(m_{12} −>m_{13} −>when(!c5)$Temp_{15}$ | m_{16})

$Temp_{15}$=(m_{14} −>m_{15} −>when(!c6)$Temp_{16}$ | m_{18})

$Temp_{16}$=(m_{17} −>m_{19} −>m_{20} −>m_{21})

Switch=(when (!c1) m_2 −> m_3 −> when (c2&!c3) $Temp_2$)

$Temp_2$=(m_6 −>m_7 −>when(!c4)$Temp_{21}$)

$Temp_{21}$=(m_8 −>m_9)

Section=(when(!c1&!c2)$Temp_3$ | when(!c1&c2)m_4 −>m_5 −>$Temp_3$)

$Temp_3$=(m_{12} −>m_{13} −>when(!c5)$Temp_{31}$)

$Temp_{31}$=(m_{14} −>m_{15})

Signal=(m_{19} −>m_{20})

对已经生成的 FSP 进程代数模型进行组合后，得到系统的有穷状态机模型，如图 4 所示。

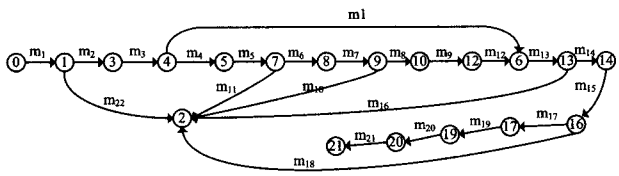

图 4 系统的有穷状态机模型

结束语 本文采用 UML 时序图描述系统需求场景，通过对 UML 顺序图中的消息前/后置条件进行分析，得到一致的需求场景；在识别 UML 顺序图中对象交互的行为序列及其类别的基础上，通过系统形式化模型生成算法将 UML 顺序图转换成 FSP(Finite State Process)，并得到系统的有穷状态机模型。该方法为安全苛求领域面临的形式化建模难这一问题的解决提供了有效途径，也为形式化验证与分析过程中系统的形式化建模提供了新思路，从安全质量方面改善了安全苛求软件的设计与开发，丰厚了基于模型的软件形式化开发方法。

参 考 文 献

[1] 王铁江，郦萌. 计算机联锁软件的 Z 规格说明[J]. 铁道学报，2003，25(4)：62-66

[2] 吴芳美. 计算机联锁软件测试评估[J]. 铁路计算机应用，1999，8(1)：7-10

[3] 李颖. 基于 UML 的车站信号软件建模[D]. 北京：北京交通大学，2008

[4] Nakarnatsu K，Kiuchi Y，et al. Intelligent Railway Interlocking Safety on Annotated Logic Program and Verification Based its Simulation[C]// Proceedings of the 2004 IEEE International Conference on Networkinp，Sensing & Control. Taipei，Taiwan，2004

[5] 吴芳美. 计算机联锁软件基于测试的安全性评价基准研究[J]. 铁道学报，2005，27(3)：97-101

[6] Blom S，Ioustinova N，Pol J，et al. Simulated Time for Testing Railway Interlockings with TTCN-3[C]//Proceedings of the 5th International Workshop on Formal Approaches to. Testing of Software. LNCS 3997，2006：1-15

[7] Garmhausen V H，Campos S，Cimatti A. Verification of a safety-critical railway interlocking system with real-time constraints [J]. Elsevier Science of Computer Programming，2000(36)：1546-1563

[8] 王曦，徐中伟，梅萌. 基于模型检测的软件安全性验证方法[J]. 武汉大学学报，2010，56(2)：156-160

[9] 赵志熙. 计算机联锁系统技术[M]. 北京：中国铁道出版社

[10] Arlow J，Neustadt J. UML2 and the Unified Process[M]. China Machine Process

[11] 王帅，吉吟东，杨士元. 一种基于场景的 CTCS-3 列车控制系统建模方法研究[J]. 铁道学报，2011，23(9)：55-61

[12] Magge J，Krammer J. Associated Concurrency：State Models and Java Programs[M]. Wiley，1999

(上接第 218 页)

调度策略可降低实时系统的任务截止期错失率。接下来的工作是研究适合 TSCTTL 调度策略的优先级分派策略，及采用 TSCTTL 策略对 CPU 利用率的影响情况。

参 考 文 献

[1] Nagy S，Bestavros A. Admission control for soft-deadline transactions in ACCORD[C]// Proceedings of the 3rd IEEE Real-Time Technology and Applications Symposium. Montreal，Canada，1997：160-165

[2] 夏家莉. 支持替代/补偿的实时调度策略[J]. 小型微型计算机系统，2005，26(2)：248-251

[3] 金宏，王强，王宏安，等. 基于动态抢占阈值的实时调度[J]. 计算机研究与发展，2004，41(3)：393-398

[4] 李琦，巴巍. 两种改进的 EDF 软实时动态调度算法[J]. 计算机学报，2011，34(5)：943-950

[5] Liu C L，Lavland J W. Scheduling algorithm for multipro gramming in a hard real-time environment[J]. Journal of ACM，1973，20(1)：40-61

[6] Semghouni S，Amanton L，Sadeg B，et al. On new scheduling policy for the improvement of firm RTDBSs performances [J]. Data & Knowledge Engineering，2007，63(2)：414-432