安全关键实时系统高可信集成技术的研究

杨仕平, 熊光泽, 桑 楠
(电子科技大学计算机科学与工程学院, 四川成都 610054)

摘 要: 为增强安全关键实时系统的可信性, 在分析高可信保障机制现状的基础上, 提出了一种支持多级关键度子系统共享同一系统资源的集成式高可信保障体系结构. 为防止不同关键度子系统间的有害干扰, 提出了基于两级结构化调度方法的时间隔离机制. 为实现时间隔离, 论文首先针对集成式多级关键高可信保障机制建立了调度模型, 然后进行了子系统及其任务的可调度性分析, 最后用实例进行了例证. 本文的研究成果也可应用于其他非关键领域, 具有较大的实用性.

关键词: 安全关键; 多级关键度; 高可信; 调度分析; 时间隔离
中图分类号: TP302. 8 文献标识码: A 文章编号: 0372-2112 (2003) 08-1237-05

Research on High Dependability Integration Technology of Safety Critical Real Time Systems

YANG Shi-ping, XIONG Guang-ze, SANG Nan
(School of Computer Science and Engineering, UEST of China, Chengdu, Sichuan 610054, China)

Abstract: After analyzing status quo of high dependability safeguard mechanism, an integrated high dependability safeguard systematic framework supporting multi-level criticality in which several subsystems share the same system resources is brought forward. In order to avoid deleterious interferences between subsystems with different criticality, a kind of high dependability temporal isolation mechanism based on two level scheduling approach is brought forward. In order to realize this mechanism, integrated high dependability safeguard mechanism supporting multi-level criticality based on single node computer is emphatically researched. The scheduling model is constructed. The schedulability of tasks and subsystems are analyzed. The correctness of time isolation mechanism is proved by several samples. The conclusion of researching on high dependability safeguard may be applied to the other no safety-related areas, and has great practicability.

Key words: safety critical; multi-level criticality; high dependability; schedulability analysis; temporal isolation

1 引言

在后 PC 时代, 随着我国国民经济信息化的迅速发展, 实时计算技术日益广泛地应用于航空航天、国防、交通运输、核电能源和医疗卫生等诸多安全关键实时系统(Safety Critical Real Time Systems, SCRTS), 这些系统之所以称为 SCRTS, 是因为它们一旦失效将会导致生命财产的重大损失以及环境可能遭受严重的破坏, 如震惊全世界的前苏联切尔诺贝利核事故, 美国亚利安娜 V 型火箭发射失败及其最近哥伦比亚号航天飞机的坠毁都给人们留下了惨痛的教训. 为减少或防止 SCRTS 发生灾难性事故, 增强系统的可信性[1] (Dependability) 是重点所在, 其中可信性是指系统在任务开始时且可用性给定的情况下, 在规定的时间和环境内能够使用且能完成既定功能的能力, 即系统/ 动则成功0 的能力. 然而, 现代社会的高速发展及世界不稳定因素的存在, 使得 SCRTS 日益庞大和复杂. 与此同时, SCRTS 的应用环境也更加复杂和恶劣, 从陆地、海洋到天空、太空, SCRTS 的应用环境不断扩展和更加严酷. 除此, SCRTS 要求持续无故障任务时间加长, 如太空探测器的长时间无故障飞行要求、通信网络的关键任务不停机要求等, 迫使 SCRTS 必须具有良好的可靠性、可维修、可在线升级等专门特性. 由此可见, 关于 SCRTS 高可信保障机制的研究具有极大的挑战性, 同时也具有较大的研究价值.

2 安全关键实时系统高可信保障机制的现状

在 SCRTS 中, 软件的设计缺陷常常是造成系统发生灾难性事故的主要根源, 而现有的防错、预错、排错或容错等技术都不足以保证复杂系统中的错误被完全排除, 因此研究 SCRTS 新的可信性保障机制是十分必要的. 为此, 国内外许多研究机构展开了积极的研究, 目前已取得了大量的研究成果. 在这些研究成果中, 最具代表性的是 Rushby 等人提出的防危

收稿日期: 2002-12-08; 修回日期: 2003-04-18
基金项目: 国防科技预研基金项目(No. 2000J6. 7. 1. DZ0206); 国家十五项目(No. 41315040106)

核[1](safety kernel),防危核在 SCRTS 中的地位与作用如图 1 左侧所示.防危核的思想是把整个应用软件按功能的不同分割为关键与非关键两部分,其中关键部分在防危核中实现,非关键部分在其余应用软件中实现,且防危核起着隔离应用软件与关键设备的作用.实际运行时,防危核利用一组专门定义的防危策略,监控应用软件及操作人员对关键设备的操作请求,并拒绝可能导致生命、财产损失的操作请求.这样便可避免由于错误的设备操作请求而引起关键设备错误动作,从而实现系统的防危性需求.

由图 1 可以看出,防危核保障机制的优点在于可把重点集中于关键部分(防危核)的开发与验证之上,而对非关键部分则不需太多的关注.这样既可提高整个系统的可信性,又降低了系统的开发与验证成本.对于大而复杂的 SCRTS,具体实现时,则让防危核与其余应用软件分别在物理位置相隔的两台不同计算机上运行,并通过可靠实时网络相连,如图 1 右侧所示.由防危核的防危保障原理可知,基于防危核的高可信保障机制特别适合于核电控制系统,因为这样的系统有足够的地理空间用于关键部分与非关键部分间的分布式实现,且能有效防止非关键部分对关键部分的干扰.但是这种保障机制也存在一些缺陷,如其两级关键度的划分方式较单一,虽然在一定程度上降低了软件验证的开销,但随着 SCRTS 复杂度的增加,防危核也将逐渐变得庞大,验证其正确性的开销也会逐渐增加.

图 1 防危核在 SCRTS 中的地位与作用及其分散式实现

3 支持多级关键度的集成式 SCRTS

由第 2 节可知,基于防危核的高可信保障机制存在关键度划分单一的缺陷.对于大而复杂的 SCRTS,很有必要把整个系统按关键度等级的不同划分为几个子系统[2],如可按美军标 DOD2178B 把整个系统划分为 A、B、C、D、E 五级.同防危核保障机制的实现方式一样,为防止不同关键度子系统间的相互干扰,许多支持多级关键度划分的 SCRTS 仍让各子系统分别独立运行在物理位置相互隔离的不同计算机上,我们可以称这种可信性保障机制为分布式多级关键保障机制,其优点在于:(1)可有效防止设计错误在不同子系统间传播;(2)可根据各子系统的实际需要采用不同的操作系统、不同的调度算法.然而这种保障机制也存在诸多不足之处:(1)分布式的实现方式在一定程度上浪费了操作系统及 CPU 等系统资源,同时也造成重量、体积、能耗、成本的增加,特别不适合于航空航天器、汽车电子、核潜艇等对重量、体积、能耗、成本有特殊限制的 SCRTS;(2)当不同关键度子系统间需要紧密协作时,其支撑网络的带宽将成为通信的瓶颈;(3)支撑网络的实时性及可靠性将在很大程度上影响 SCRTS 的可信性.由上面的分析可知,对于航空航天器等系统,合理的设计是把整个系统按关键度等级的不同划分为多个子系统,并让各子系统尽可能地运行在相同的处理器上,在此本文称该可信性保障机制为集成式多级关键保障机制[3].在该高可信保障机制下,可根据各子系统关键度等级的不同采取与之相应的可信性措施来满足其可信性需求.这样既可集中精力保证最关键部分的可信性,又能提高整个系统的可信性,同时也可降低系统的开发与验证费用.

在集成式多级关键保障机制中,为保证整个系统的可信性,必须保证运行于同一处理器之上的某子系统不会对另一子系统产生有害干扰.为防止各子系统间的有害干扰,采取一定的隔离措施是十分必要的,这样便可以保证一子系统的运行不会遭受其他子系统的无意或恶意破坏,如内存覆盖、独占 CPU 等.通常可采取的隔离措施有空间隔离与时间隔离[4,5],其中空间隔离的主要目的是保证各应用子系统有各自独立的地址空间,同时防止不同地址空间内的程序发生无意或恶意越界读写等非法操作;而时间隔离则是为了确保某子系统不会长期独占或超时使用处理器而阻止或延迟其他子系统的运行.本文将在下面重点研究子系统间的时间隔离机制.

4 集成式多级关键保障机制中的时间隔离

4.1 集成式 SCRTS 的调度模型

当不同关键度的子系统集成到同一 CPU 上时,我们可以建立如图 2 所示的集成式 SCRTS 调度模型:

由图 2 可以看出,集成式 SCRTS 的调度由高、低两级调度组成,其中高级调度为各子系统中的调度,称为应用级的调度,低级调度为操作系统中的调度,称为系统级的调度.为同时实现应用级与系统级的调度,我们采用两级结构化调度机制[6]:操作系统首先使用周期性调度方法调用各子系统,即首先为各子系统分配与之相应的处理器时槽,然后各子系统在所分配的时槽内使用优先级调度算法调度各任务.下面本文

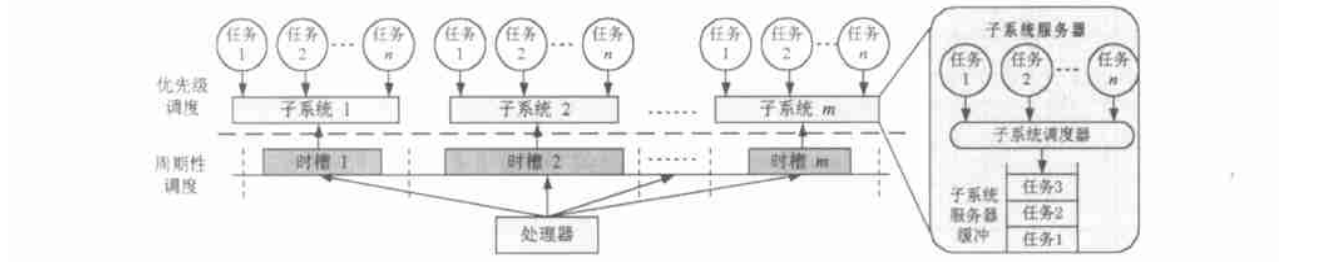

图 2 基于两级结构化调度机制的集成式 SCRTS

将详细说明两级结构化调度机制的实现方式.

4.2　子系统及其任务的可调度性分析

在进行具体调度分析之前, 本文首先假设每一子系统 A_k 中都存在一个子系统服务器 S_k, 如图 2 中右侧所示, 该服务器使用优先级抢占调度算法调度 A_k 中的各任务, 并同时和其他

定义 1　(服务器能力)　不失一般性, 假设标准处理器的总处理速率为 1, 当把服务器看作一个虚拟处理器时, 服务器能力 A_k 是指服务器的处理速率与标准处理器总处理速率的比值. 这个比值表示服务器占用标准处理器总处理速率的比例, 所以有 $0< A_k[1$.

在集成式 SCRTS 中, 假设操作系统以固定的周期 G_k 周期性地调度子系统服务器 S_k, 而在每个子系统周期 G_k 内, 子系统服务器 S_k 可使用 A_kG_k 个时间单元调度子系统内的任务, 其中 $A_k[1$, 且在其余时间间隔 $G_k- A_kG_k$ 内, 子系统服务器 S_k 被阻塞. 当子系统服务器 S_k 使用速率单调调度算法[7, 8](Rate Monotonic Algorithm, RMA) 调度各消息时, 设 S_k 内的 n 个实时任务按优先级关系排序为 $S_k[S_{n-1}[, [S_1$, 即 S_1 的优先级最高, S_n 的优先级最低. 当子系统 A_k 中的 n 个任务 $S_1, S_2, , , S_n$ 在同一时间 $t_0= 0$ 被初始化时, 则在时间段 $[0, t](t> 0)$ 内, 优先级高于或等于 S_i 的任务所要求的最坏累积处理器时间为 $W_i(t)= \sum_{j=1}^{i} C_j\lceil t/T_j\rceil$, Lehoczky[9] 证明了在时间段 $[0, t]$ 内子系统 A_k 中的任务 S_i 可被调度的条件为 $W_i(t) [t$, 即 $\sum_{j=1}^{i} C_j\lceil t/T_j\rceil [t$ 成立.

现在我们假设子系统 A_k 中的 n 个任务 $S_1, S_2, , , S_n$ 不是在能力为 A_k 的子系统服务器 S_k 中被调度, 而是在处理器能力为 A_k 的专用处理器上被调度. 由于 $A_k[1$, 所以任务的计算时间将增加, 即为 C_i/ A_k. 根据 Lehoczky 在文献[9] 中的充分必要调度条件可以得知, 如果子系统 A_k 内的各任务 $S_1, S_2, , , S_n$ 可被处理器能力为 A_k 的专用处理器调度, 则存在时间 $t I H_i= \{lT_j| j= 1, 2, , , i; l= 1, 2, , , \lfloor D_i/T_j\rfloor\} G \{D_i\}$ 使 $W_i(A_k, t)= \sum_{j=1}^{i} \frac{C_j}{A_k}\lceil t/T_j\rceil [t$ 成立, 其中 $W_i(A_k, t)$ 表示在时间段 $[0, t]$ 内且处理器的能力为 A_k 时, 优先级高于或等于 S_i 的任务所要求的最坏累积处理器时间. 为便于下面讨论, 本文在此定义 $B_i(A_k)= \max_{t I H_i}\{t- W_i(A_k, t)\}$, $B_0(A_k)= \min_{i= 1, 2, , , n} B_i(A_k)$. 由定义可知, 当任务 S_i 可调度时, $B_i(A_k)$ 表示在时间段 $[0, D_i]$ 内处理器没有运行任何优先级高于或等于 S_i 的任务的时间, 可称 $B_i(A_k)$ 为时间段 $[0, D_i]$ 内的 i 级非活动周期, 同时称 $B_0(A_k)$ 为子系统 A_k 中的最小非活动周期.

由前可知, 当采用分布式多级关键高可信保障机制时, 各子系统 A_k 都运行在专用的处理器上, 且假设 A_k 运行在处理能力为 $A_k(A_k[1)$ 的专用处理器上时, 子系统 A_k 中的诸任务均可被调度. 当采用集成式多级关键高可信保障机制时, 假设此时的处理器的总能力为 1. 0, 则我们需要证明在每个周期 G_k 中, 当分配给子系统服务器 S_k 的处理器能力为 A_k 时, 子系统 A_k 中的各任务均可被调度, 即通过比较运行于子系统服务子系统中的服务器一起共享处理器. 另设子系统 A_k 由 n 个任务 $S_1, S_2, , , S_n$ 组成, 每个任务 S_i 可用三元组 (C_i, D_i, T_i) 表示, 其中 T_i 为任务 S_i 的周期, C_i 为任务 S_i 的最坏执行时间, D_i 为任务 S_i 的死限, 且满足 $C_i[D_i[T_i$. 为便于后面分析子系统服务器 S_k 的可调度性, 我们首先作如下定义:

器 S_k 与处理器能力为 A_k 的专用处理器上任务集的可调度性, 证明下面的定理 1 成立.

定理 1　在集成式多级安全关键高可信保障机制中, 子系统 A_k 可被周期为 G_k、服务器能力为 A_k 的子系统服务器 S_k 调度的条件为:

(1) 子系统 A_k 可在处理器能力为 A_k 的专用处理器上被调度;

(2) 不等式 $G_k- G_kA_k[B_0(A_k)$, 即 $G_k[B_0(A_k)/(1- A_k)$ 成立.

证明　设运行于子系统服务器 S_k 中的任务除子系统 A_k 的 n 个任务 $S_1, S_2, , , S_n$ 外, 还包括一个额外的任务 S_0. 额外任务 S_0 在每个周期 G_k 内都被调度, 计算时间为 $C_0= G_k- A_kG_k$, 且任务 S_0 的优先级最高, 即有 $S_n[S_{n-1}[, [S_1< S_0$, 于是任务 S_0 可抢占子系统 A_k 中的其他任务. 则我们需要证明的是: 即使由于任务 S_0 的存在而导致 A_k 中的任务被抢占, A_k 中的任意任务 S_i 也能在其死限 D_i 到达之前完成. 根据文献[9] 中的充分必要调度条件可以得知, 如果存在 $t I H_i G G_i$, 其中 $G_i= \{lG_k| l= 1, 2, , , \lfloor D_i/G_k\rfloor\}$, 使 $\sum_{j=1}^{i} C_j\lceil \frac{t}{T_j}\rceil + C_0\lceil \frac{t}{G_k}\rceil [t$ 成立, 则任务 S_i 在服务器 S_k 上可调度.

另外假设任务 S_i 可在处理器能力为 A_k 的专用处理器上被调度, 则存在时间 $t_i^* I H_i$ 使 $B_i(A_k)= t_i^* - W_i(A_k, t_i^*) \setminus B_0(A_k) \setminus 0$ 成立, 其中 $i= 1, 2, , , n$. 由前可知, $W_i(A_k, t_i)$ 是时间 t_i 的非递减函数, 可假设 $t_i^* = mG_k+ D$, 其中 $D< G_k$, 如果 $D\setminus B_0(A_k)$, 则有:

$$\begin{aligned}
&\sum_{j=1}^{i} C_j\lceil \frac{t_i^*}{T_j}\rceil + C_0\lceil \frac{t_i^*}{G_k}\rceil = A_kW_i(A_k, t_i^*) + (m+ 1)C_0\\
&[A_k(t_i^* - B_0(A_k)) + (m+ 1)C_0\\
&= A_k(t_i^* - B_0(A_k)) + (m+ 1)(1- A_k)G_k\\
&[A_k(t_i^* - B_0(A_k)) + (1- A_k)(t_i^* - D) + B_0(A_k)\\
&= t_i^* + (1- A_k)(B_0(A_k) - D)\\
&[t_i^* \qquad (1)
\end{aligned}$$

相反, 如果 $D< B_0(A_k)$, 那么在 $t_c= mG_k< t_i^*$ 时, 则有:

$$\begin{aligned}
&\sum_{j=1}^{i} C_j\lceil \frac{t_c}{T_j}\rceil + C_0\lceil \frac{t_c}{G_k}\rceil [A_k(t_i^* - B_0(A)) + mC_0\\
&[A_k(t_i^* - D) + m(1- A_k)G_k\\
&= A_kt_c + (1- A_k)t_c\\
&= t_c \qquad (2)
\end{aligned}$$

由于 $t_c I G_i$, 所以当分配给子系统 A_k 的处理器能力为 A_k 时, 子系统 A_k 中的各任务可被服务器 S_k 调度. 由条件(1) 的证明过程可知, 通过比较服务器 S_k 上的任务执行序列与处理能力为 A_k 的专用处理器上的任务执行序列可以看出, 在每

个子系统周期 G_k 末, 服务器 S_k 使用了与专用处理器相同的处理能力(大小为 A_k)运行 A_k 中的各任务, 也即定理 1 中的条件(1)成立.

另外, 当 A_k 中的各任务运行于专用处理器上时, 由于此时不存在假定的特殊任务 S_0, A_k 中的各任务将不会被任务 S_0 阻塞, 因而它们可在每个周期 G_k 内更早的完成. 当各任务运行于服务器 S_k 之上时, 由于服务器 S_k 还要与系统中的其他子系统服务器竞争处理器, 所以很有必要增加额外的条件限制服务器中各任务的完成时延, 该限制条件即为定理 1 中的条件(2), 即任务的完成时延等于各任务死限之前的最小非活动周期, 于是定理 1 得证.

4.3 子系统周期及服务器能力 A_k 的确定

由 4.2 节的内容可知, 定理 1 实质上提供了一种可确定参数对(A_k, G_k)的方法, 即确定子系统服务器 S_k 被调度的频率(大小为 $1/G_k$)以及应在每周期 G_k 内分配多大的处理器能力 A_k 去调度子系统 A_k 中的各任务. 本节将详细研究如何确定参数对(A_k, G_k).

为方便起见, 假设子系统服务器 S_k 使用 RMA 算法调度子系统 A_k 中的各任务, 同时对处理器能力进行归一化处理, 即令处理器的处理能力为 1.0. Liu 证明了当处理器能力为 1 且使用 RMA 算法调度各任务时, 处理器的利用率 Q_k 不会超过 $n(2^{1/n}-1)$, 即有式 $Q_k=\sum_{i=1}^{n}\frac{C_i}{T_i}$[$n(2^{1/n}-1)$ 成立[7]. 于是可推知当处理器能力为 A_k(A_k[1) 且使用 RMA 调度算法调度各任务时, 由于此时处理器的处理速率降低, 则任务的计算时间将增加, 即由 C_i 改变为 C_i/A_k, 于是可得此时处理器的利用率 Q_{kc} 为 $\sum_{i=1}^{n}\frac{C_i}{A_kT_i}$, 且有 $Q_{kc}=\sum_{i=1}^{n}\frac{C_i}{A_kT_i}$[$n(2^{1/n}-1)$ 成立. 然而当各子系统集成于处理器能力为 1.0 的处理器上时, 则各子系统内的任务可调度时处理器的利用率为 Q_k(而不是 Q_{kc}), 于是可推知在集成式多级关键高可信保障机制中, 不等式 Q_k[$A_kn(2^{1/n}-1)$ 成立, 即 $Q_k/(n(2^{1/n}-1))$[A_k 成立. 由此可知分配给子系统服务器 S_k 的服务器能力 A_k 应不小于 $Q_k/(n(2^{1/n}-1))$. 这样便确定了子系统服务器 S_k 所需的最小处理器能力 A_{kmin}, 即有 $A_{kmin}=Q_k/(n(2^{1/n}-1))$. 在各 A_k 之和不大于 1.0 的前提下, 可适当增加各子系统服务器 S_k 的 A_{kmin} 值, 这样便可最终确定出各子系统服务器 S_k 的 A_k. 在确定好各 A_k 的值后, 便可根据定理 1 中的条件(2): G_k[$B_0(A_k)/(1-A_k)$ 确定出与 A_k 相对应的子系统周期 G_k, 到此便完全确定出了各子系统服务器 S_k 的参数对(A_k, G_k).

5 实例分析

为了用实例证明定理 1 的正确性, 假设存在如表 1 所示的四个子系统, 各子系统中包含几个周期性任务(用三元组 C_i, D_i, T_i4表示). 为方便起见, 假设各任务的死限 D_i 等于其周期 T_i, 且在每个周期 G_k 内使用 RMA 算法调度子系统 A_k 中的各任务.

由表 1 可知, 运行完子系统 1 中的各任务后所得到的处理器利用率 Q_1 为 0.24, 即 $Q_1=\sum_{i=1}^{5}\frac{C_i}{T_i}=\frac{3}{100}+\frac{8}{110}+\frac{9}{160}+\frac{13}{260}+\frac{10}{330}=0.24$, 同理可计算出其他子系统的处理器利用率分别为 0.17、0.26、0.03. 为确定出各子系统服务器 S_k 的参数对(A_k, G_k), 应采用 4.3 节中的方法首先确定出各子系统服务器 S_k 的处理能力 A_k. 如对于子系统 1, 分配给子系统服务器 S_1 的处理器能力 A_1 不应小于 0.32, 即使用 $A_{1min}=Q_1/(n(2^{1/n}-1))$ 计算得 $A_{1min}=0.24/(5@(2^{1/n}-1))=0.32$. 同样可计算出分配给子系统服务器 S_2、S_3、S_4 的最小处理器能力 A_{2min}、A_{3min}、A_{4min}分别为 0.22、0.33、0.04. 在各 A_k 的总和不大于 1 的情况下可适当增大所计算出的最小处理器能力 A_{kmin}, 如 $A_1=0.32$、$A_2=0.28$、$A_3=0.34$、$A_4=0.06$ 便是 4 个子系统的一组可行处理器能力分配. 当分配好各子系统的处理器能力 A_k 后, 便可由定理 1 中的条件(2)确定出与 A_k 相应的子系统周期 G_k. 最后便可以得到 4 个子系统参数对(A_k, G_k)的一组可行分配为 (0.32, 36)、(0.28, 59)、(0.34, 28)、(0.06, 57).

表 1 四个子系统中的任务参数

	子系统 1 (Q_1= 0.24)	子系统 2 (Q_2= 0.17)	子系统 3 (Q_3= 0.26)	子系统 4 (Q_4= 0.03)
任务 (C_i, D_i, T_i)	(3, 100, 100)	(3, 50, 50)	(6, 78, 78)	(1, 80, 80)
	(8, 110, 110)	(4, 90, 90)	(9, 110, 110)	(3, 140, 140)
	(9, 160, 160)	(4, 120, 120)	(16, 160, 160)	
	(13, 260, 260)	(6, 170, 170)		
	(10, 330, 330)			

6 操作系统中周期性调度器的实现

本文在前面分析了子系统及其任务的可调度性, 主要目的在于确定参数对(A_k, G_k). 由于分配给各子系统的执行周期并不一定需要连续, 因而可以在操作系统中灵活地构建周期性调度器. 为简单起见, 可使操作系统中的调度周期 T_{OS} 为各子系统调度周期 G_k 的最小值[10], 即 $T_{OS}=\min(G_k)$. 对于表 1 中的 4 个子系统, 前面已经得到了参数对(A_k, G_k)的一个可行分配为{(0.32, 36)、(0.28, 59)、(0.34, 28)、(0.06, 57)}, 由于此时的最小子系统周期 G_k 为 28, 所以操作系统中调度器的调度周期 T_{OS}可设为 28, 同时 4 个子系统的可行分配将从{(0.32, 36)、(0.28, 59)、(0.34, 28)、(0.06, 57)}转换为{(0.32, 28)、(0.28, 28)、(0.34, 28)、(0.06, 28)}. 当操作系统以每 28 个时间单元调度四个子系统时, 分配给四个子系统的时间单元分别为 8.96= 28@0.32, 7.84= 28@0.28, 9.52= 28@0.34, 1.68= 28@0.06. 图 3 为操作系统中调度器的调度周期 T_{OS}等于 28 个时间单元时的子系统调度序列. 除了操作系统周期性地调度子系统服务器外, 子系统服务器还要调度其内部的各任务. 为了说明怎样使用优先级调度算法调度每个子系统内的任务流, 我们仍使用表 1 中的例子. 由表 1 可知, 在子系统 3 内包含三个实时任务, 处理器将在每 28 个时槽单元内为子系统 3 的子系统服务器 S_3 分配 9.52 个时间单元. 假设子系统服务

器 S_3 使用 RMA 调度算法调度子系统内的各任务, 则在所分配的 9.52 个时间单元内, 三个实时任务的调度序列如图 3 的上半部所示.

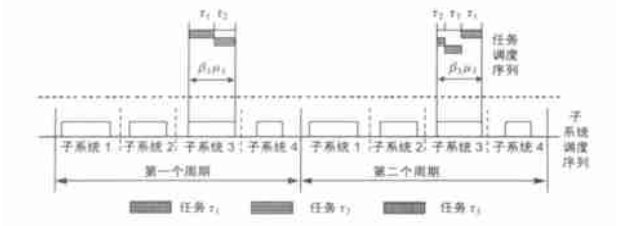

图 3　操作系统中的周期性调度序列

7　结束语

为增强航空航天、核潜艇等对体积、能耗、重量及开发成本有特殊限制的 SCRTS 的可信性. 本文在分析总结以防危核为代表的分布式多级关键保障机制的基础上, 提出了适用于航空航天等系统的集成式多级关键保障机制. 然而高可信集成各子系统的前提是子系统间不会相互干扰, 这就需要通过空间隔离与时间隔离来保证, 而本文所研究的重点在于时间隔离, 并通过两级调度机制实现了时间隔离. 然而, 本文所作的研究工作只能算是沧海一粟, 有关 SCRTS 高可信保障机制的研究还有大量的工作需要去作. 在信息安全大热潮的今天, SCRTS 的可信性似乎被人冷落, 美国的 9.11 事件、俄罗斯的核潜艇事故或许给我们敲响了警钟, 原来 SCRTS 与信息安全系统同样重要, 它们的同步发展才会从根本上提高人民的生活质量.

参考文献:

[1] Wika K J. Safety Kernel Enforcement of Software Safety Policies[D]. Charlottesville, VA: Department of Computer Science, University of Virginia, 1995.

[2] Totel E, Blanquart J.2P, Deswarte Y, Powell D. Supporting Multiple Levels of Criticality[C]. IEEE Symposium on Fault Tolerant Computing Systems(FTCS-28), Munich, 1998.

[3] N Audsley, A Wellings. Partition scheduling in APEX runtime environment for embedded avionics software[J]. Proc. of IEEE Real2Time Computing Systems and Applications, Oct. 1998. 103-109.

[4] M Younis, M Aboutabl, D Kim. An approach for dporting software partitioning and reuse in integrated modular avionics[J]. Proc. of IEEE Real2time Technology and Applications Symposium, 2000.

[5] Ben L, Di Viro. A formal model of partitioning for integrated modular avionics[R]. NASA contractor Report NASA/CR2199822087 03, August 1998.

[6] Z Deng, J W S Liu. Scheduling real2time applications in an open environment[J]. Proc. IEEE Real Time Systems Symposium, 1997, 12: 308-319.

[7] Liu C L, Layland J W. Scheduling algorithms for multiprogramming in a hard real time environment[J]. ACM, 1973, 20(1): 46-61.

[8] N Audsley, A Burns, M Richardson, A Wellings. Hard real2time scheduling: the deadline monotonic approach[A]. Eighth IEEE Workshop on Real2time Operating Systems and Software[C]. 1991. 133-137.

[9] J Lehoczky, L Sha, Y Ding. The rate2monotonic scheduling algorithm: exact characteristics and average case behavior[J]. Proc. IEEE Real2Time Systems Symposium, 1989, 12: 166-171.

[10] C2C Han, K2J Lin, C2J Hou. Distance constrained scheduling and its applications to real2time systems[J]. IEEE Trans. on Computers, 1996, 45(7): 814-826.

作者简介:

杨仕平　男, 1974 年 5 月出生于四川省阆中市, 2001 年 4 月于电子科技大学攻读计算机应用博士学位至今, 长期从事高可信实时操作系统和分布式安全关键实时系统方面的研究工作. 先后参与过多项国防科技预研基金及/十五0等重大项目. Email: yangsp@uestc.edu.cn.

熊光泽　男, 1938 年出生于四川省丹棱县, 教授、博导, 1962 年毕业于成都电讯工程学院计算机专业, 1982 年至 1987 年先后在美国硅谷和澳洲 CIT 从事嵌入式实时软件研发, 主要研究领域: 嵌入式实时计算、软件安全与可靠性等. Email: gzxiong@uestc.edu.cn.