基于防危核(壳)的安全关键硬实时系统响应时间的分析

黎忠文[1,3],陈 亮[1],熊光泽[2]
(1.厦门大学信息科学与技术学院,福建厦门 361005; 2.电子科技大学计算机科学与工程学院,四川成都 610054;
3.电子科技大学中山学院,广东中山 528402)

摘 要: 基于响应时间的可调度分析是设计硬实时系统的基础.防危核(壳)是安全关键硬实时系统的一种新防危措施,现有的普适响应时间分析方法不能适用于这类系统.本文的贡献在于采用检测点时间冗余容错和优先级提升思想,通过分析内/外错误、隐/显错误,提出了一种合适的响应时间分析方法,从而为这类系统提供有效的离线可调度分析工具,同时也为这类系统设计了一种基于检测点的容错方案.本方法稍作修改也可为其它安全关键系统服务.最后通过仿真实验验证了本方法的有效性和优越性.

关键词: 防危核(壳); 安全关键系统; 容错; 检测点; 响应时间分析
中图分类号: TN915.01 **文献标识码**: A **文章编号**: 0372-2112 (2006) 04-0647-06

Response Time Analysis for Safety-Critical Hard Real-Time Systems Based on Safety Kernel/Shell Scheme

LI Zhong-wen[1,3], CHEN Liang[1], XIONG Guang-ze[2]
(1. Information Science and Technology College, Xiamen University, Xiamen, Fujian 361005, China;
2. Computer Science and Engineering College, UEST of China, Chengdu, Sichuan 610054, China;
3. Zhongshan Institute, UESTC, Zhongshan, Guangdong 528402, China)

Abstract: Safety-critical hard real-time systems (in brief, we call them SCHRS below) often operate in harsh environmental conditions that necessitate fault-tolerant computing and safety assurance techniques. Various hardware as well as software safety assurance techniques are employed in these systems among which safety kernel (shell) is a new scheme in application layer and it's correctness can be proofed by formal tool. Appropriate response time analysis schemes (in brief, we call them RTA below) are fundamental to the design of predictable hard real-time systems. However few RTA schemes is suitable to SCHRS based on safety kernel or shell. The main contribution of this paper is two-fold. First, we propose an appropriate RTA to improve system fault resilience. Compared with other response time analysis polices, the proposed scheduling analysis takes into account the fact that the recoveries of tasks in safety kernel or shell may be executed at higher priority levels, and there are several kinds of error, such as internal, external, hidden and visible error. This characteristic is very important in SCHRS based on safety kernel or shell. Second, we present a suitable checkpointing fault-tolerant scheme for SCHRS based on safety kernel or shell. The emphasis here is on utilizing this RTA as an off-line design support tool. The effectiveness of the proposed approach is evaluated by simulation.

Key words: safety kernel (shell); safety-critical systems; fault-tolerant; checkpointing; response time analysis

1 引言

软件的安全性已经成为关系现代安全关键系统安危的主要因素[1].软件的大量应用一方面导致系统的复杂性呈指数级增长,安全隐患严重;另一方面,改变了系统原有的故障模式,使设计型软件错误成为事故的主要来源[2,3].然而传统防危(即 safety保障技术.为与 security相区别,把它称为防危)和可靠技术对此却渐感力不从心.据统计

收稿日期: 2004-08-22; 修回日期: 2005-10-26
基金项目:福建省 2003年青年科技人才创新基金 (No. 2003J020);福建省 2004年自然科学基金 (No. A0410004);厦门大学院士基金 (No. 0630-E23011);厦门大学新世纪优秀人才支持基金 (No. 0000-X07116);厦门大学 985二期信息创新平台项目 (No. 2004-2007)

,就一般复杂度的软件,用测试法只能使软件错误率降低到每小时 10^{-4}个[4],对于复杂度稍高的软件,测试法还达不到这种效果.而与防危相关的控制系统的错误率要求是每小时 10^{-9}个,甚至有的是每小时 10^{-10}个[4].事实上目前还有许多系统(包括国家级)仍然建立在"脆弱"的软件之上[5].因此,软件新防危技术成为研究热点.目前已在该领域取得的具有代表性的研究成果是 Levenson等提出的防危核(Safety Kernel).它是针对软件设计缺陷而提出来的一种应用级防危机制.由于构思巧妙,它一直倍受研究. Rushby利用形式化方法对防危核和防危策略进行了描述及正确性验证[6],而 Kevin则为两个模型系统 MSS(Magnetic Stereotaxis System)和 UVAR(University of Virginia Research Reactor)构建了相应的防危核,并在原型系统中得到了检验[7].近年来,我们对防危核在分布式系统中的关键实现技术进行了一定的研究[8].后来 Anderson等人把防危核改进为防危壳(safety shell)[9].防危壳在功能上与防危核相似,只是内部组成结构不同.具体实现时,防危壳把防危策略进一步细分为三个子模块,分别在状态监视器、时间监视器和防危异常处理器内实现.另外基于防危核(壳)的软件构架及与编程语言的关系也是研究热点,如基于反射机制 Open C++和 CML的实现技术[10]等.

调度分析是设计硬实时系统的基础[11],它对安全关键系统具有特别重要的意义[12].响应时间的分析是广泛采用的调度分析方法,它通常用来确定无错的可预测系统中,有固定优先级的各实时任务在最坏情况下的响应时间,从而预测系统设计的可行性,即任务的可调度性. Burns等人把它用于分析基于任务复制的硬实时容错系统[13].文献[14]在主任务与替代任务优先级不相同这个更接近真实系统的条件下,对文献[13]进行改进.后来响应时间分析方法还被用于分析基于软件检测点容错的实时系统[12].但是这些方法并不适于既有检测点又有核(壳)控任务需及时响应要求的安全关键硬实时系统.如关闭反应堆这类出错处理任务的优先级就应该非常高.文献[12]不能反应这种切实需要.而文献[14]在没有检测点的情况下只考虑了一个内部错误发生时,替代任务优先级高于主任务时的响应时间分析.事实上,目前我们尚没有查到针对基于防危核(壳)的安全关键系统响应时间分析的资料.因此本文将对此展开研究,为这一类安全关键系统提供一种离线的静态分析方法和工具,对该方法进行修改也可用于其它硬实时系统.

2 计算模型

设主任务集 Ψ 是 n个任务集合,$\Psi = \{\tau_1, \tau_2, \cdots, \tau_n\}$,每个任务 τ_i 可用四元组 $\langle C_i, D_i, T_i, p_i \rangle$ 表示.其中 T_i 为任务 τ_i 两个实例间最小间隔时间,τ_i 可以是周期和非周期任务;C_i、D_i 和 p_i 分别为任务 τ_i 的执行时间、死限和优先级,且满足 $C_i \leq D_i \leq T_i$, $i = 1, 2, \cdots, n$.鉴于固定优先级的调度机制在安全关键系统领域已逐步取代过去常用的静态调度机制[11],主任务按优先级进行调度.为简化,设 Ψ 内各任务是按优先级降序排序,即如果 $i < j$,则 $p_i > p_j$.主任务 τ_i 设置 T_i 个检测点 T_i, T_2, …, T_i,任务的始点不设检测点,T_i 在任务尾,因此这些检测点把任务 τ_i 分成了 T_i 段.每一段起始和终止点为相邻的两个检测点,比如 T_{k-1}和 T_k,这里记为 $(T_{k-1}, T_k]$,称为任务 τ_i 的第 T_k 个子任务段.第一子任务段比较特殊,为任务始点到检测点 T_1.任务 τ_i 的每一子任务段分别对应一个替代任务,若主任务在执行中发生了错误,则调用防危核(壳)内相应的替代任务用于容错,即主任务将从发生错误的下一子任务段的始点开始执行.注意段 $(T_{k-1}, T_k]$ 为半开半闭的区间,它表示 T_{k-1}属于第 T_{k-1}段,而不属于第 T_k 段.因此当错误发生在 T_{k-1}点时,第 T_{k-1}段的替代任务将就绪;而当错误发生在 T_k 点时,第 T_k 段的替代任务将被调用.显然每个主任务都有多个替代任务.为了让错误能得到及时的纠正,替代任务的优先级应不低于主任务,且同一主任务的多个替代任务的优先级相同,当主任务与替代任务的优先级相同时,先执行替代任务.为了便于分析,令 $\bar{\tau}_i$ 是主任务 τ_i 的执行时间最长的替代任务,$\bar{C}_i$ 是其执行时间.主任务、替代任务和检测点之间的关系如图 1所示.

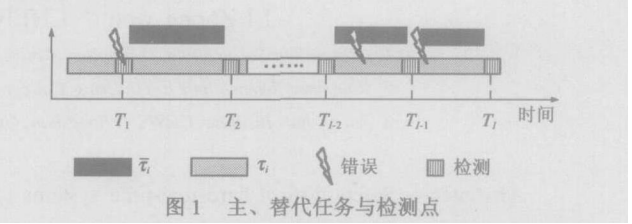

图1 主、替代任务与检测点

检测点的设置与任务段粒度有关,一般而言为避免多米诺效应,产生错误的繁殖,检测点可以设置在任务通信之前,另外在本安全关键系统中还需设置在系统受保护组件被访问之前.令 O_i 是在任务 τ_i 建立一个检测点的开销,显然 O_i 至少包括 O_i^1 和 O_i^2 两项,其中 O_i^1 是在该检测点进行有效测试时所需时间开销;O_i^2 则为保持检测点状态所需的开销,比如存贮器读写等.和文献[12]一样,对任务 τ_i 检测点的开销我们统一考虑为 O_i,而不进行细分, $i = 1, 2, \cdots, n$.

本文设错误为暂态错误,即经过执行替代任务可以消除的错误.除了和文献[14]一样把错误归于内部错误和外部错误之外,我们把错误还定义为隐式和显式错误,其中显式错误指环境检测中检测到的错误,例如由硬件检测到的"执行非法指令"、"算术溢出"和"保护违规"等,以及由实时编程语言的运行支持系统检测出的诸如"数组越界"和"引用空指针"等错误.这类错误一旦出现,系统将产生中断,它的表现域一般在两个检测点间.而隐式错误是指应用程序自己检测到的错误,比如在检测点设置看门狗定时器,合理性检查性等,这类错误可能在子任务段开始后不久就已发生,但在检测点进行测试时才被发现.隐、显式

错误的出错和表现域如图 2所示. 内部错误和外部错误是从替代任务的角度来划分任务的,而隐式和显式错误是以错误的表现时间来划分的.

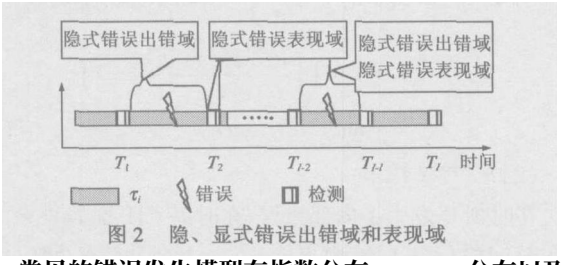

图 2　隐、显式错误出错域和表现域

常见的错误发生模型有指数分布、Weibull分布以及它们的变种. 为方便分析,我们和文献[12]和[14]一样,采用最小错误间隔时间 T_E.

计算模型的假设条件:

(1)和文献[12]一样,检测点被均匀的设置在任务中,即任务 t_i 的每一子任务段的计算时间均为 $\lceil C_i/T_i \rceil$;

(2)只考虑暂态错误,系统生命期中它们占绝大多数;

(3)错误间的最小间隔时间 T_E 是已知的;

(4)不会产生错误的繁殖;

(5) 100%的故障诊断率;

(6)错误恢复过程中如再发生故障,则重复执行替代任务段就可.

3　现有分析方法的局限性

实时系统的可调度性、时间性分析方法有很多种,文献[11]已详细讨论了这些主要方法不适用于安全关键系统的原因. 因此本文只对与本计算模型最接近的新成果 PB[12]进行对比分析. Punnekkat S 和 Burns A 等在文献[12]中把硬件容错方法-checkpointing应用于软件容错,并把它与基于优先级的调度算法结合在一起. 他们在任务 t_i 的实例中均匀地插入 m_i 个检测点,当实例运行至检测点时,系统就对该实例进行检测,如出错,则该实例退回到最近的上一个检测点,任务的优先级不变,系统重新调度. PB方法扩大了任务的可调度空间. 其任务的最坏响时间为公式(1):

$$R_i = C_i + m_i \times O_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \times (C_j + m_j \times O_j) + \left\lceil \frac{R_i}{T_E} \right\rceil \times \max_{\tau_j \in hp(i) \cup \{i\}} \left(\left\lceil \frac{C_j}{m_j} \right\rceil + O_j \right) + B_j^* \tag{1}$$

其任务 t_i 的最坏响时间为公式(1):

其中, $hp(i) = \{t_j \mid t_j \in \Gamma \wedge p_j > p_i\}$; B_j^* 为阻塞时间,文献[12]在分析中对它忽略不计.

该分析算法不适于本计算模型的原因:

(1)安全关键系统中,替代任务的优先级一般应比主任务高,因为错误需及时处理,否则后果极为严重. 比如关闭反应堆这类替代任务的优先级就应该非常高. 文献[12]的方法不能反应安全关键系统的这种切实需要.

(2)当替代任务的优先级高于主任务时,公式(1)不能反应任务的最坏响应时间. 比如任务 t_i 发生错误,当其替代任务优先级 $\bar{p}_i$ 大于 p_j 时,虽然 t_j 就绪并且 $p_j > p_i$,但 t_j 必须等待 $\bar{t}_i$ 的执行,因此与公式(1)相比, R_j 会增加,而 R_i 会减小.

4　本计算模型可调度性分析

为便于比较,本文采用与文献[12]和[14]一样的记法, $ip(i)$和 $sp(i)$见图 3所示:

定义 1　$hp(i) = \{t_j \in \Gamma \mid p_j > p_i\}$;

定义 2　$ip(i) = \{t_j \in \Gamma \mid \bar{p}_j \geq p_i\}$;

定义 3　$sp(i) = \{t_j \in \Gamma \mid p_j > \bar{p}_i\}$;

定义 4　$ipe(i) = \begin{cases} ip(i), & \text{if } (\bar{p}_i = p_i) \\ ip(i) - \{t_i\}, & \text{if } (\bar{p}_i > p_i) \end{cases}$

在 $T_E > 0$和给定主、替代任务优先级关系 $p(p_i, \bar{p}_i)$的条件下,任务 t_i 的最坏响应时间 $R_i(T_E, p)$的计算要受错误的性质,即显式或隐式、内或外的影响.

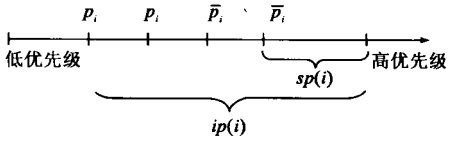

图 3　任务的优先级关系

4.1　外部错误

任务 t_i 的外部错误[14]是指与任务 t_i 并发执行的其它任务发生的错误. 令在外部错误条件下任务 t_i 的最坏响应时间为 $R_i^{ext}(T_E, p)$. 这里的错误发生情况分为外部错误中嵌套内部错误和外部错误不嵌套内部错误两种,前者记为 $R_i^{ext\text{-}int}(T_E, p)$,后者为 $R_i^{ext\text{-}ext}(T_E, p)$. 它们又可进一步分为显式和隐式错误,隐式和显式的 $R_i^{ext\text{-}ext}(T_E, p)$,分别记为 $R_i^{ext\text{-}extl}(T_E, p)$和 $R_i^{ext\text{-}ext2}(T_E, p)$. 先计算 $R_i^{ext\text{-}extl}(T_E, p)$.

设当任务 t_i 就绪以后,任务 t_ξ ($p_\xi > p_i$)执行过程中出现隐式错误,不妨设在其第 T_k 个检测点检测出错误,引起替代任务 $\bar{t}_\xi$ 的执行. 这个错误对于任务 t_i 而言就是隐式外部错误. 令任务 t_ξ 的第 T_k 个子任务段 $(T_{k-1}, T_k]$的起始和终止点分别对应 t 和 t'时刻,时间段 $[t, t']$检测出该隐式错误, $t' < t$因此 t_i 的 CPU 将被 $hp(i)$和 $ip(i) - \{t_i\}$中任务抢占,故:

$$R_i^{ext\text{-}extl}(T_E, p) = C_i + T_1 \times O_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{R_i^{ext\text{-}extl}(T_E, p)}{T_j} \right\rceil \times (C_j + T_j \times O_j) + \left\lceil \frac{R_i^{ext\text{-}extl}(T_E, p)}{T_E} \right\rceil \times \max_{\tau_i \in ip(i) - \{\tau_i\}} (\bar{C}_j + O_i) \tag{2}$$

令 $\bar{C}_k + O_i$ 为任务集 $ip(i) - \{t_i\}$中替代任务和检测点开销之和的最大者. 若 $ip(i) - \{t_i\} = \Phi$,则令 $(\bar{C}_k + O_k) = 0$.

定理 1　在任务优先级关系 $p(p_i, \bar{p}_i)$和 $T_E > 0$条件下, $R_i^{ext\text{-}ext}(T_E, p) = R_i^{ext\text{-}extl}(T_E, p)$.

证明:设当任务 t_i 就绪以后,任务 t_ξ ($p_\xi > p_i$)在第 T_k 个任务段(T_{k-1}, T_k)发生显式错误,该段起始点分别对应 t' 和 t''时刻,时间段 $[t, t'']$为任务 t_ξ 的第 T_k 个检测点的检测时间.显式错误属于环境检测范围,一旦出错后,就会产生中断,因此在时间段 $[t', t)$内相应的替代任务就会就绪,而不必推迟到时间段 $[t, t'']$,所以 $R_i^{ext\text{-}ext1}(T_E, p) \supseteq R_i^{ext\text{-}ext2}(T_E, p)$.因此 $R_i^{ext\text{-}ext}(T_E, p) = R_i^{ext\text{-}ext1}(T_E, p)$.

定理 2 若 $(\bar{C}_i + O_i) \leq (\bar{C}_k + O_k)$; $i \neq k$且 $i, k \in \{1, 2, \cdots, n\}$,则 $R_i^{ext\text{-}int}(T_E, p) = R_i^{ext\text{-}ext}(T_E, p)$.

证明:同定理 1,这里只讨论隐式内错误就行了.先分析任务 t_i 只有一个子任务段,即第 T_w 子段,发生内错误的情况.显然不论是显式还是隐式内部错误,t_i 在第 T_w 子段只能发生一次内错误.在 $R_i^{ext\text{-}ext}(T_E, p)$中加入段标志参数,有 $R_i^{ext\text{-}ext}(T_E, p, T_I\text{-}T_I) = R_i^{ext\text{-}ext}(T_E, p, T_I\text{-}T_{w-1}) + R_i^{ext\text{-}ext}(T_E, p, T_w\text{-}T_w) + R_i^{ext\text{-}ext}(T_E, p, T_{w+1}\text{-}T_I)$,其中 $1\text{-}T_I$ 表示任务 t_i 的第 T_1 (即第 1 子段)至第 T_I 子段,$1\text{-}T_{w-1}$ 表示第 1 至第 T_{w-1} 子段,$T_w\text{-}T_w$ 代表它的第 T_w 子段,$T_{w+1}\text{-}T_I$ 则是第 T_{w+1} 子段到第 T_I 子段.同前面一样,设 t_i 的第 T_w 段就绪时刻为 t',在时刻 t^f 发生了内部错误,时间段 $[t, t'']$检测出该隐式错误.令在时刻 t^f 任务 $\bar{t}_i$ 执行完备.显然在时间 $[t', t'']$内任务 t_i 将被 $hp(i)$、$ip(i)\text{-}\{t_i\}$任务集中的任务抢占 CPU;最迟在 t时刻 $\bar{t}_i$ 就绪,在时间段 $(t'', t^f]$内 $\bar{t}_i$ 将被 $sp(i)$及 $sp(i)$的替代任务抢占 CPU.由于 $hp(i) \supseteq sp(i)$、$(\bar{C}_i + O_i) \leq (\bar{C}_k + O_k)$,根据公式(2)和定理 1 有 $R_i^{ext\text{-}ext}(T_E, p, T_w\text{-}T_w) \supseteq [t', t^f]$.故:$R_i^{ext\text{-}ext}(T_E, p, T_I\text{-}T_I) = R_i^{ext\text{-}ext}(T_E, p, T_I\text{-}T_{w-1}) + R_i^{ext\text{-}ext}(T_E, p, T_w\text{-}T_w) + R_i^{ext\text{-}ext}(T_E, p, T_{w+1}\text{-}T_I) \supseteq R_i^{ext\text{-}ext}(T_E, p, T_I\text{-}T_{w-1}) + [t', t^f] + R_i^{ext\text{-}ext}(T_E, p, T_{w+1}\text{-}T_I)$.

同理可证当任务 t_i 有多个子任务段发生内错误时,定理 2仍然成立.故,$R_i^{ext\text{-}int}(T_E, p) = R_i^{ext\text{-}ext}(T_E, p)$.

4.2 内部错误

下面求发生内部错误时,任务 t_i 的最坏响应时间 $R_i^{int}(T_E, p)$.对于任务 t_i 而言,内部错误的发生将引起其替代任务 $\bar{t}_i$ 的执行,这里仍然存在显式和隐式错误之分,与定理 1类似可以证明隐式错误情况下 t_i 的最坏执行时间包含了显式错误情况下 t_i 的最坏执行时间,因此这里只计算隐式内部错误时 t_i 的 $R_i^{int}(T_E, p)$.它又分成两个部份,即内部错误发生前后的最坏执行时间,分别记为 $R_i^{int0}(T_E, p)$和 $R_i^{int1}(T_E, p)$.

4.2.1 段的最坏响应时间

设任务 t_i 在第 T_k 个子段发生了隐式内部错误.令 t_i 的第 T_k 段就绪时刻为 t',在时刻 t^f 发生了内部错误、在时刻 t任务 $\bar{t}_i$ 就绪和时刻 t^f 任务 $\bar{t}_i$ 执行完备.这里使用文献[14]的方法分 $[t', t'']$和 $[t'', t^f]$两段进行讨论,分别记为 $R_i^{int0\text{-}sub}(T_E, p)$和 $R_i^{int1\text{-}sub}(T_E, p)$.

(1) $R_i^{int1\text{-}sub}(T_E, p)$

在时刻 t'',任务 $\bar{\tau}_i$ 就绪.这时影响 $\bar{\tau}_i$ 响应时间的任务集为 $sp(i)$,因此

$$R_i^{int1\text{-}sub}(T_E,p) = \bar{C}_i + O_i + \sum_{\tau_j \in sp(i)} \left\lceil \frac{R_i^{int1\text{-}sub}(T_E,p)}{T_j} \right\rceil \times (C_j + T_j \times O_j) + \left(\left\lceil \frac{R_i^{int1\text{-}sub}(T_E,p)}{T_E} \right\rceil - 1 \right) \times \max_{\tau_j \in sp(i) \cup \{\tau_i\}} (\bar{C}_j + O_j) \quad (3)$$

(2) $R_i^{int0\text{-}sub}(T_E, p)$

在时刻 t^f 发生了内部错误、在时刻 t''任务 $\bar{\tau}_i$ 就绪.因此在时间段(t, t''),影响 $R_i^{int0}(T_E, p)$的因素有 $hp(i)$及 $hp(i)$中对于任务 τ_i 而言的外部错误和相应的替代任务.由于在时刻 t''之后,任务 $\bar{\tau}_i$ 就绪,而 $hp(i)$包括两个子集合:$subl_hp(i) = \{\tau_j \in \Gamma \mid p_j > p_i \wedge p_j \leq \bar{p}_i\} = hp(i)\text{-}sp(i)$和 $sub2\_hp(i) = \{\tau_j \in \Gamma \mid p_j > p_i \wedge p_j > \bar{p}_i\} = sp(i)$.集合 $sub2_hp(i)$在 t''时刻以后仍然要抢占任务 $\bar{\tau}_i$,因此要进行分别计算.

$$R_i^{int0\text{-}sub}(T_E,p) = \lceil C_i/T_I \rceil + O_i + \sum_{\tau_j \in hp(i)\text{-}sp(i)} \left\lceil \frac{R_i^{int0\text{-}sub}(T_E,p)}{T_j} \right\rceil \times (C_j + T_j \times O_j) + \sum_{\tau_w \in sp(i)} \left(\left\lceil \frac{R_i^{int\text{-}sub}(T_E,p)}{T_w} \right\rceil - \left\lceil \frac{R_i^{int1\text{-}sub}(T_E,p)}{T_w} \right\rceil \right) \times (C_w + T_w \times O_w) + \left(\left\lceil \frac{R_i^{int\text{-}sub}(T_E,p)}{T_E} \right\rceil - \left\lceil \frac{R_i^{int1\text{-}sub}(T_E,p)}{T_E} \right\rceil \right) \times \max_{\tau_k \in ip(i)} (\bar{C}_k + O_k) \quad (4)$$

(3)错误段最坏响应时间:

$$R_i^{int\text{-}sub}(T_E,p) = R_i^{int0\text{-}sub}(T_E,p) + R_i^{int1\text{-}sub}(T_E,p)$$

4.2.2 段内不发生内错误时,该段最坏响应时间

$$R_i^{ext\text{-}sub}(T_E,p) = \left\lceil \frac{C_i}{T_I} \right\rceil + O_i + \sum_{\tau_j \in hp(i)} \left\lceil \frac{R_i^{ext\text{-}sub}(T_E,p)}{T_j} \right\rceil \times (C_j + T_j \times O_j) + \left\lceil \frac{R_i^{ext\text{-}sub}(T_E,p)}{T_E} \right\rceil \times \max_{\tau_k \in ip(i) - \{\tau_i\}} (\bar{C}_k + O_k)$$

4.2.3 内错误时,任务 τ_i 的最坏响应时间

从 4.2.1 可以得出任务 τ_i 各子任务段的最坏时间的计算与任务其它段没有关系,因此任务 τ_i 的 $R_i^{int}(T_E, p)$为各子任务段的最坏响应时间之和:

$$R_i^{int}(T_E,p) = \sum_{i=1}^{T_i} \max(R_i^{int\text{-}sub}(T_E,p), R_i^{ext\text{-}sub}(T_E,p)) \quad (5)$$

4.3 任务 τ_i 的最坏响应时间 $R_i(T_E, p)$

定理 1 已证明,当$((\bar{C}_i + O_i) \leq (\bar{C}_k + O_k)) \wedge (ip(i)\text{-}\{\tau_i\} \neq \Phi)$时,$R_i^{ext}(T_E, p) \geqslant R_i^{int}(T_E, p)$,因此这时:$R_i(T_E, p) = R_i^{ext}(T_E, p) = \max(R_i^{ext}(T_E, p), R_i^{int}(T_E, p))$.但当$((\bar{C}_i + O_i) \leq (\bar{C}_k + O_k)) \wedge (ip(i)\text{-}\{\tau_i\} = \Phi)$或者$(\bar{C}_i + O_i) > (\bar{C}_k + O_k)$时,$R_i^{ext}(T_E, p)$与 $R_i^{int}(T_E, p)$的大小与 $\bar{p}_i$、p_i、

hp(i)、ip(i)和 sp(i)等因素有关,只有通过计算才知道.故:

$$R_i(T_E, p) = \max(R_i^{ext}(T_E, p), R_i^{int}(T_E, p)) \quad (6)$$

5 实例

下面用本算法与 PB 算法进行比较.设主任务集 $\Psi = \{t_1, t_2, t_3\}$.其中 t_1:〈$C_1 + O_1 = 10, D_1 = 60, T_1 = 60, p_1 = 3$〉; t_2:〈$C_2 + O_2 = 15, D_2 = 100, T_2 = 100, p_2 = 2$〉; t_3:〈$C_3 + O_3 = 20, D_3 = 160, T_3 = 160, p_3 = 1$〉.令(0, 1, 1)和(0, 0, 2)分别代表当错误发生时各替代任务的优先级提升指标,比如三元式(0, 1, 1)表示任务 t_2 之替代任务的优先级为在其主任务优先级的基础上提升一个单位,即 $\overline{p_2} = p_2 + 1 = 3$.(0, 1, 1)和(0, 0, 2)的剩余含义为:$\overline{p_1} = p_1 + 0 = 3, \overline{p_3} = p_3 + 1 = 2$; $\overline{p_1} = p_1 + 0 = 3, \overline{p_2} = p_2 + 0 = 2, \overline{p_3} = p_3 + 2 = 3$.

表 1　任务集 Ψ_1 的最坏执行时间

t_i	D_i	T_I	$T_E = 34$					$T_E = 35$					$T_E = 39$					$T_E = 40$					$T_E = 42$				
			PB	(0, 1, 1)		(0, 0, 2)		PB	(0, 1, 1)		(0, 0, 2)		PB	(0, 1, 1)		(0, 0, 2)		PB	(0, 1, 1)		(0, 0, 2)		PB	(0, 1, 1)		(0, 0, 2)	
			R_i	R_i^{int}	R_i^{ext}	R_i^{int}	R_i^{ext}	R_i	R_i^{int}	R_i^{ext}	R_i^{int}	R_i^{ext}	R_i	R_i^{int}	R_i^{ext}	R_i^{int}	R_i^{ext}	R_i	R_i^{int}	R_i^{ext}	R_i^{int}	R_i^{ext}	R_i	R_i^{int}	R_i^{ext}	R_i^{int}	R_i^{ext}
t_1	60	1	20	20	25	20	30	20	20	25	20	30	20	20	25	20	30	20	20	25	20	30	20	20	25	20	30
t_2	100	1	55	50	95	60	95	55	50	95	60	95	55	50	75	60	75	44	40	75	40	75	44	40	75	40	75
t_3	160	1	180	105	100	105	100	180	105	100	105	100	180	105	100	105	100	160	105	100	80	100	160	105	100	80	100

表 2　任务集 Ψ_2 的最坏执行时间

t_i	D_i	T_I	$T_E = 18$					$T_E = 20$				
			PB	(0, 1, 1)		(0, 2, 1)		PB	(0, 1, 1)		(0, 2, 1)	
			R_i	R_i^{int}	R_i^{ext}	R_i^{int}	R_i^{ext}	R_i	R_i^{int}	R_i^{ext}	R_i^{int}	R_i^{ext}
t_1	30	1	12	12	16	12	16	12	12	16	12	16
t_2	70	2	72	64	44	64	44	72	64	38	64	38
t_3	100	1	88	84	88	84	88	88	74	78	74	78

从表 1可以看出,当 $T_E = 34$、$T_E = 35$、$T_E = 39$时,若采用 PB 算法,则由于 t_3 的最坏响应时间为 180,大于其死限,因此任务集不可调度.但若采用本算法,根据公式(6): $R_i(T_E, p) = \max(R_i^{ext}(T_E, p), R_i^{int}(T_E, p))$,则无论是优先级(0, 1, 1)还是(0, 0, 2)任务集均可调度.在实验中我们发现当 $T_E < 40$时,使用 PB 算法任务集都不可调度.而当 $T_E > 34$时,采用本算法对替代任务的优先级进行提升后,任务集就可调度.这一结果与理论是相符的,因为通过优先级的提升就可让低优先级的任务抢占高优先级的 slack时间,从而减少其响应时间.因此本算法比 PB 算法更优.表 2是另一组主任务集 $\Psi_2 = \{t_1, t_2, t_3\}$.其中

t_1:〈$C_1 + O_1 = 6, D_1 = 30, T_1 = 50, p_1 = 3$〉

t_2:〈$C_2 + O_2 = 20, D_2 = 70, T_2 = 90, p_2 = 2$〉;

t_3:〈$C_3 + O_3 = 60, D_3 = 100, T_3 = 150, p_3 = 1$〉.

当 $T_E = 18$、$T_E = 20$时,若采用 PB 算法,则由于 t_2 的最坏响应时间为 72,大于其死限,因此任务集不可调度.但若采用本算法,则无论是优先级(0, 1, 1)还是(0, 2, 1)任务集均可调度,说明本算法比 PB 算法更优.

6 总结

基于防危核(壳)的安全关键系统的特点在于,应用软件对系统受保护组件的访问,都必须经过防危核的检测,合法者予以支持,反之则采取相应的防危处理措施,这些措施通过失效树分析,将造成系统误操作的失效源分为若干类,并对不同的失效源制定相应的防危措施.因此我们把这类系统的应用层软件分成两组,一是完成系统功能的应用软件,称为主任务;另一组用于错误处理的防危核(壳)内的防危措施,称为替代任务.本文的贡献在于为基于防危核(壳)的安全关键系统提出了一种可行的可调度时间分析方法,该方法是以提升替代任务优先级,使系统及时完成出错处理为基础提出来的,符合安全关键系统的安全需求.这里我们采用设置检测点的方法,在应用软件访问受保护组件之前,就插入一些检测点,定期对应用软件进行检测,以便发现错误后及时纠正.这种方法的优势在于它与基于检测点的已有算法相比,具有替代任务优先高,适合于安全关键系统的需要,与基于优先级的算法相比,它具有多个检测点,能满足防危核(壳)的需要,而且其内部错误不局限于一个,更加接近真实的系统.最后通过仿真实验验证了本方法的优越性.我们下一步将研究检测点的优化和任务优先级结构的设计问题,从而安全关键系统的性能.

参考文献:

[1] Zalewski J, Ehrenberger W, Saglietti F, et al Safety of computer control systems: challenges and results in software development[J]. Annual Reviews in Control, 2003, 27(1): 23 - 37.

[2] Leveson N G. System safety in computer-controlled automotive systems[R]. California: SAE Congress, 2000.

[3] Benveniste A, Astrom K J. Meeting the challenge of computer science in the industrial applications[J]. IEEE Trans automatic control, 1993, 38(7): 1004 - 1009.

[4] Butler R W, Finelli G B. The infeasibility of quantifying the reliability of life-critical real-time software[J]. IEEE Tran on software engineering, 1993, 19(1): 3 - 12.

[5] President's information technology advisory committee reports to the president[R]. Washington: Information technology research: investing in our future, 1999.

[6] Rushby J. Kernels for safety[A]. Rushby J. Safe and secure computing systems sympoium[C]. London: Blackwell Scientific Press, 1989. 210 - 220.

[7] Kevin R. Safety kernel enforcement of software safety policies[D]. USA: University of Virginia, 1995.

[8] 黎忠文,熊光泽,李乐民. 分布式系统安全保障新体系的研究[J]. 电子学报, 2003, 31(4): 564 - 568.
Li Zhong-wen, Xiong Guang-ze, Li Le-min. Research on New Security and Safety Assurance Structure of Distributed System[J]. Acta Electronica Sinica, 2003, 31(4): 564-568. (in Chinese)

[9] Sahraoui A E, Anderson E, Katwijk V, et al Formal specification of a safety shell in real-time control practice[A]. Vila J 25th IFAC workshop on real-time programming (WRTP'2000)[C]. Oxford: Elsevier Press, 2000. 117 - 123.

[10] Sanz R, Zalewski J. Pattern-based control systems engineering[J]. IEEE Control Systems, 2003, 23(3): 43 - 60.

[11] Bate I, Burns A. An Integrated approach to scheduling in safety-critical embedded control systems[J]. Real-Time Systems, 2002, 25(1): 5 - 37.

[12] Punnekkat S, Burns A, Davis R. Analysis of checkpointing for real-time systems[J]. Real-Time Systems, 2001, 20(1): 83 - 102.

[13] Burns A, Punnekkat S. Probabilistic schedulig guarantees for fault-tolearant real-time systems[A]. Rushby J Proc Seventh intenational working conf. Dependable computing for critical application (DCCA-7)[C]. California, USA: IEEE Computer Society Press, 1999. 339 - 356.

[14] George M A, Burns A. An optimal fixed-priority assignment algorithm for supporting fault-tolerant hard real-time systems[J]. IEEE Transactions on computers, 2003, 52(10): 1332 - 1346.

作者简介:

黎忠文 女, 1970年出生于四川丹巴, 2001年毕业于电子科技大学计算机学院,获博士学位, 2001～2003在电子科技大学信息与通信系博士后流动站工作. 2003出站后到厦门大学工作. 研究领域:实时系统高安全和高可靠技术. E-mail: lizw@xmu.edu.cn

陈 亮 男, 1978年出生于浙江舟山,硕士. 研究领域:软件安全技术.

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.