突发性故障模式下实时任务的优先级分配策略

周正勇[1] 阳富民[1] 李 俊[2] 胡贯荣[1] 涂 刚[1] 张 杰[1]
(华中科技大学计算机科学与技术学院 武汉 430074)[1] (武汉数字工程研究所 武汉 430074)[2]

摘 要 在容错实时系统中,可调度性分析是确保实时任务在限定时间内完成的重要手段。分析了突发性故障模式的可调度性问题,针对该故障模式下已有策略的不足,设计了优先级分配策略,并根据策略的性质实现了容错优先级变迁因子的搜索算法。深入的分析和实验证明,这种策略能够有效地提高系统的容错能力。

关键词 容错实时系统,最坏响应时间,可调度性,突发性故障
中图法分类号 TP316 **文献标识码** A

Priority Assignment Strategy for Real-time System under Fault Bursts

ZHOU Zheng-yong[1] YANG Fu-min[1] LI Jun[2] HU Guan-rong[1] TU Gang[1] ZHANG Jie[1]
(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)[1]
(Wuhan Digital Engineering Research Institute, Wuhan 430074, China)[2]

Abstract In real-time systems, schedulers must be fault tolerant to guarantee no missed deadline. Based on the analysis of worst-case response time schedulability for real-time systems under fault bursts, we found out that fault-tolerant priority assignment strategy could improve system fault resilience effectively, compared with traditional fault-tolerant strategy. Also, we presented a fault-tolerant priority configuration search algorithm for the proposed analysis.

Keywords Fault-tolerant real-time systems, Worst-case response time, Scheduling analysis, Fault bursts

1 引言

实时系统需要保证系统的硬件或软件出现故障时实时任务仍可以继续运行并在截止期限内完成。采用容错技术可以有效地避免故障时系统崩溃,硬件冗余、软件冗余和时间冗余是有效的手段。在某些空间有限的实时系统中,硬件冗余度是有限的,时间冗余是必然的选择。从时间特性上看,故障可以分为间歇性和暂时性的,间歇性的故障将不断地在故障出现和故障良好的状态间切换,暂时性故障会在一段时间后消失。间歇性的故障则包括软件错误或者环境干扰等多种原因。假设两个故障之间有着最小时间间隔[1]的伪周期性故障模型是经典的故障模型。这种模型可以有效地管理定点故障或电磁扰动,但不能有效地管理有限时间区间潜在且随机的故障[2],即实时系统在一个有限的时间段受到多次持续的干扰。这种情况常见于传感器前的障碍或机场飞机起落对雷达波的干扰。突发性故障模型可以很好地描述这种现象。分析这种故障模型下实时任务的最坏响应时间,并提高实时系统容错能力是本文的主要目的。

2 系统模型

2.1 故障模型

伪周期故障模型是经典的故障模型。在该模型中,两个连续的故障由一个已知的最小时间间隔 T_F 分开(如图1所示)。

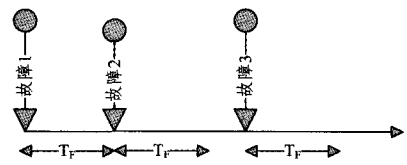

图1 伪周期性故障模型

伪周期故障模型可以处理离散的故障,通常应用于设计故障和电磁兼容性故障。事实上,图1中描述的状况是比较悲观的,故障1和故障2之间间隔时间恰好是 T_F;故障3和故障2之间的间隔大于 T_F。所以,该模型中假设故障发生的次数要多于真实情况。T_F 描述了故障发生的频率,如果故障事件的频率增加,则 T_F 减小。

本文所研究的故障模型是突发性故障模型,该模型描述如下:在一个时间间隔 ΔF 内故障的时间分布和数量是不可知的。前后两次时间间隔 ΔF 之间的最小时间用 T_F 表示。时间段 ΔF 之外不会出现故障。时间段 ΔF 内故障的出现频率和最小周期是不可知的,因此 ΔF 时间段内可以看作是一个黑盒子,黑盒子内执行的任务可能发生故障。突发性故障模型描述如图2所示。

伪周期故障模型研究的重点是故障发生的频率。突发性故障模型则侧重于系统被干扰时间段的时长;根据描述,在这个时间段内实时系统的故障是随机发生的,显然伪周期故障模型不适合这种情况。突发性故障模型是本文研究的重点。

本文受国家自然科学基金(61173049)资助。
周正勇(1979—),男,博士生,讲师,主要研究方向为实时调度、智能控制,E-mail:zhouzhengyong@mail.hust.edu.cn;**阳富民**(1966—),男,教授,主要研究方向为嵌入式操作系统。

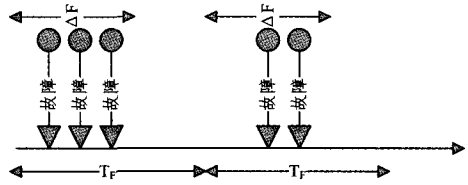

图 2　突发性故障模型

2.2　任务模型

实时系统中实时任务的集合记为 Γ，$\Gamma=\{\tau_1,\tau_2,\cdots,\tau_n\}$。$\tau_i(1\leqslant i\leqslant n)$代表实时任务，$i$ 为下标序号；对于任意的任务 $\tau_i=(C_i,T_i,D_i)$，其中 C_i 表示任务的最坏执行时间(Worst Case Execution Time，WCET)；T_i 表示主任务的周期；D_i 表示任务的相对截止期限。本文中任务的相对截止期限小于等于 T_i。任务可以是周期性或突发性的，对于突发性任务而言，T_i 指的是任务到来的时间间隔的最小值。根据某种静态优先级调度算法 FP(Γ)来分配任务的优先级，例如：RMS[3](Rate Monotonic Scheduling)或 DMS[4](Deadline Monotonic Scheduling)，每个任务被分配到各不相同的优先级。

2.3　容错方式

对于故障模型给出下面的假设：

1. 每次突发故障持续时间为 ΔF，ΔF 内爆发故障的数量和时间分布是不可知的。

2. 两次突发故障之间的时间间隔 T_F 要大于任务集中最大截止期限。因此每个任务在一个周期内最多只可能遇到一次故障爆发。

目前实时系统可采用的软件容错技术有重复执行、N 版本程序设计、检查点和恢复块技术等[5,6]。前两种技术都是在任务结束后对结果进行错误检测。检查点和恢复块可以在执行中检测故障。系统所采取的容错技术是重复执行。一旦检测到任务 τ_i 出现故障，系统将立即调度任务 τ_i 的备选任务或者重新执行任务 τ_i 来进行容错处理。而当备选任务出现故障，所采取的容错技术是重复执行。文献[2]提出了两种容错策略，详细规定了检测到故障后调度程序的行为：简单策略(End Detection/Full Re-execution/Simple，ED/FR/S)重复执行检测到错误的任务，不处理低优先级的任务；多重策略(End Detection/Full Re-execution/Multiple，简称 ED/FR/M)则认为如果当前任务检测到错误那么其他低优先级的任务也可能出错。调度程序除了恢复当前任务，还要打断低优先级任务并执行恢复动作。

为了方便分析，假设系统中所有的备选任务分配与主任务有相同的优先级，这个假设并不是本文模型中的局限，任务切换和调度的时间忽略不计；不存在任务之间的约束关系，每个任务都是独立的；系统采用可抢占式调度，当前运行的任务一定是可运行任务中优先级最高的；另外，假设系统在任务与对应的备选任务之间的调度切换无损耗。

3　问题的提出

在无容错需求的硬实时系统中，计算任务 τ_i 的最坏响应时间 R_i 要考虑任务自身的执行时间 C_i 和所有 $p_j>p_i$ 任务抢占时间的总和 I_i；因此，任务的最坏响应时间计算公式为[7]：

$$R_i=C_i+I_i \tag{1}$$

Audsley 等[4]对 I_i 做了进一步的分析而得到了 R_i 的详细计算公式：

$$R_i=C_i+\sum_{\tau_j\in hp(i)}\left\lceil\frac{R_i}{T_j}\right\rceil C_j \tag{2}$$

其中，$hp(i)$表示 $p_j>p_i$ 的任务集合，即 $hp(i)=\{\tau_j\in\Gamma\mid p_j>p_i\}$。对于任务 $\tau_j\in hp(i)$，在$[0,R_i]$期间，任务 τ_j 执行的最大次数为$\lceil R_i/T_j\rceil$，则任务 τ_i 在执行过程中被任务 τ_j 抢占的总时间为$\lceil R_i/T_j\rceil C_j$。

针对突发性故障模式容错实时系统中任务 τ_i 的最坏响应时间的计算公式如式(3)[2]：

$$R_i^{\Delta F}=R_i+\Delta F+I_i^{\Delta F}+F_i \tag{3}$$

其中，ΔF 是故障的时长，$I_i^{\Delta F}$ 是故障发生后备选任务被高优先级任务抢占的时间，F_i 是故障恢复的时间。$I_i^{\Delta F}$ 的计算公式如式(4)所示：

$$I_i^{\Delta F}=\sum_{\tau_j\in hp(i)}\left\lceil\frac{R_i^{\Delta F}-R_i-\Delta F}{T_j}\right\rceil C_j \tag{4}$$

ED/FR/S 和 ED/FR/M 策略在 F_i 的计算上有所不同，其计算公式分别由式(5)和式(6)给出。可以看出 ED/FR/M 和 ED/FR/S 计算最高优先级任务响应时间的结果是一样的。差别就在于其他优先级的任务，显然通过 ED/FR/M 计算出的最坏响应时间更短，可调度性更好，容错能力更强。

$$F_i=\begin{cases}2*C_i, & p_i=1\\ 2*C_i+2*\sum\limits_{\tau_j\in hp(i)}\left\lceil\frac{R_i}{T_j}\right\rceil C_j, & p_i>1\end{cases} \tag{5}$$

$$F_i=\begin{cases}2*C_i, & p_i=1\\ C_i+\sum\limits_{\tau_j\in hp(i)}\left\lceil\frac{R_i}{T_j}\right\rceil C_j+\max\limits_{\tau_j\in hp(i)}C_j, & p_i>1\end{cases} \tag{6}$$

在此对表 1 和表 2 中的容错实时任务集合分别采用 ED/FR/S 策略和 ED/FR/M 策略。由表 1 可以看出，当 $\Delta F=20$ 时，根据式(3)计算所得到的每个任务的最坏响应时间均小于各自对应的截止期限，则在 $\Delta F=20$ 时系统是可调度的，但是当 ΔF 增大为 30 时，任务 τ_3 的最坏响应时间大于其截止期限 D_3，因而导致整个系统变为不可调度。ED/FR/M 策略下任务的故障恢复时间少于 ED/FR/S 策略，相应每个任务的最坏响应时间也较少，可以满足表 1 中任务集在 $\Delta F=20$ 和 $\Delta F=30$ 时系统可调度性的要求。但是 ED/FR/M 策略仍然存在不可调度的情况，比如表 2 中当 $\Delta F=50$ 时。

表 1　ED/FR/S 的限制

任务	T_i	C_i	D_i	R_i^{20}	R_i^{30}
τ_1	300	10	300	50	60
τ_2	600	100	600	360	370
τ_3	800	110	800	800	810

表 2　ED/FR/M 的限制

任务	T_i	C_i	D_i	R_i^{30}	R_i^{40}
τ_1	300	15	300	75	105
τ_2	700	100	700	290	320
τ_3	800	150	800	790	820

分析表 1 和表 2 中任务最坏响应时间的结果可知，任务 τ_1 和 τ_2 的最坏响应时间远小于相对截止期限，这些任务还存在一些空闲时间可以挪用，但是 ED/FR/M 和 ED/FR/S 策略没有利用这些空闲时间来满足任务 τ_3 截止期限的要求。如果利用这些空闲时间就要改变备选任务的优先级，而这两种策略是简单采用任务重新执行的方式，虽然采取上述两种策略时，系统无需额外的容错调度处理机制，完全可以按照无

容错需求时所采取的 $FP(\Gamma)$ 调度机制来处理，但是，这种简单的方式可能会导致任务无法在截止期限内完成，进而使得整个系统不可调度，造成严重的损害。

文献[8,9]中的容错优先级分配策略通过提高任务的容错优先级来利用高优先级任务的空闲时间，这种策略可以增强系统的容错能力。但是文献[8,9]针对的是伪周期故障模型，而本文研究的是突发性故障模型；在不同的故障模型下，其任务响应的过程大大不同。接下来我们在突发性故障模型中运用此策略，首先分析提高任务的容错优先级对任务最坏响应时间的干扰以及干扰的规律，并根据这种规律提出任务最坏响应时间的计算公式举例说明允许容错优先级变迁的策略可以在 ED/FR/S 和 ED/FR/M 策略无法满足系统容错需要的情况下能够有效地提高系统容错能力。

4 可调度性分析

当系统监测到任务出错时，任务要重新执行并分配不同的优先级，为了方便分析，把重新执行的任务和原任务分开研究，我们称因容错而执行的任务为备选任务，原任务为主任务，下面给出备选任务和容错优先级的定义。

定义 1 容错实时系统中实时任务的备选任务集合记为 $\overline{\Gamma}$，$\overline{\Gamma}=\{\overline{\tau}_1,\overline{\tau}_2,\cdots,\overline{\tau}_n\}$。

$\overline{\Gamma}$ 代表容错硬实时主任务集合 Γ 的备选任务集合，$\overline{\tau}_i$ $(1\leqslant i\leqslant n)$ 对应于任务 τ_i 的备选任务。$\overline{\tau}_i=(\overline{C}_i,\overline{D}_i)$，其中，$\overline{C}_i$ 表示 $\overline{\tau}_i$ 的最坏执行时间，$\overline{D}_i$ 表示 $\overline{\tau}_i$ 的截止期限。显然 $\overline{D}_i=D_i$，$\overline{C}_i=C_i$。

定义 2 容错实时系统中实时任务 τ_i 的优先级记为 p_i，p_i 表示任务 τ_i 根据某种静态优先级调度算法 $FP(\Gamma)$ 所分配的优先级别，$p_i\in\{1,2,\cdots,n\}$，对于 p_i 来说，$\forall i,j,i\neq j,p_i\neq p_j$。若 $p_i=n$，表示 τ_i 的优先级最高；而当 $p_i=1$ 时，表示 τ_i 的优先级最低。

$\overline{p}_i$ 表示 τ_i 的容错优先级，即备选任务 $\overline{\tau}_i$ 的优先级。其中，$\overline{p}_i\in\{1,2,\cdots,n\}$。对于 $\overline{p}_i$ 来说，$\forall i,j,i\neq j,\overline{p}_i\neq\overline{p}_j$。

4.1 优先级改变的影响

当系统允许主任务和备选任务的优先级不一致时，任务的执行过程肯定会发生变化。下面举例说明容错任务的执行情况。

假设任务集合 $\Gamma=\{\tau_1,\tau_2\}$。其中任务 τ_1 的周期小于任务 τ_2，根据 RMS 算法可知，$p_1>p_2$。假设任务 τ_1 和任务 τ_2 在完成之前都遇到了故障爆发。根据 ED/FR/S 和ED/FR/M 策略可得 $\overline{p}_1>\overline{p}_2$，两种策略在容错过程中都是先执行任务 τ_1 的备选任务 $\overline{\tau}_1$，然后再执行任务 τ_2 的备选任务 $\overline{\tau}_2$；而且 $\overline{\tau}_2$ 在执行过程中还可能被任务 τ_1 抢占。图 3(a)和(b)所示的正是任务 τ_2 的执行经过。由于任务 $\overline{\tau}_2$ 要等待任务 τ_1 执行完毕，这导致了 $\overline{\tau}_2$ 不能满足其实时性要求，任务 τ_2 不可调度。如果改变备选任务的优先级 $\overline{p}_2>p_1$，则任务 τ_1 的执行不再抢占 $\overline{\tau}_2$ 的执行，任务 τ_1 要等待任务 τ_2 恢复完成后才能被响应执行。这种策略我们称之为容错优先级变迁策略。同样我们将恢复故障的动作分为两类：一种是简单方式即检测到任务错误后，只执行备选任务，不处理低优先级的任务中潜在的故障；另一种是多重方式，即在当前任务检测到错误后执行备选任务，还要打断低优先级任务并执行相应的备选任务。这两种方式定义为 End Detection/Full Re-execution/Simple/Change Priority(SCP) 和 End Detection/Full Re-execution/Multiple/Change Priority(MCP)，如图 4(a)和(b)所示，采用 SCP 策略和 MCP 策略就能使 $\overline{\tau}_2$ 满足系统的要求，τ_2 能被调度。但是也能看出 τ_2 能满足截止实现的要求是因为占用了 τ_1 的运行时间或者是备选任务 $\overline{\tau}_1$ 的运行时间。所以，SCP 策略和 MCP 策略相对于 ED/FR/M 策略和 ED/FR/S 策略在同样运行环境中，τ_1 的最坏响应时间延长，而 τ_2 的最坏响应时间则变短了。随着任务数量的增多，优先级变迁的方式也随之增多，容错优先级的改变将对任务故障的回复时间产生很大的影响；不同的优先级变迁方式也会作用于任务出现故障后的抢占情况和被抢占情况，使得任务的调度情况迥然有别。

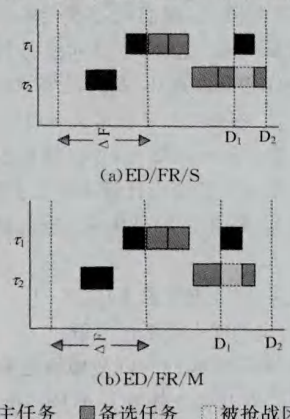

图 3 ED/FR/S 和 ED/FR/M 策略下任务的执行过程

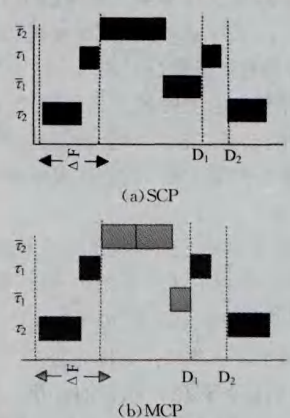

图 4 SCP 和 MCP 策略下任务的执行过程

4.2 计算任务最坏响应时间

由于不同的优先级变迁方式影响任务的最坏响应时间，本节先给出定义来描述不同的优先级变迁，然后再讨论最坏响应时间的计算。

定义 3 容错实时系统中优先级变迁因子 P_x 是一个多元组 $\langle h_{x,1},h_{x,2},\cdots,h_{x,n}\rangle$，其中 $h_{x,i}=\overline{p}_i-p_i$，且 $1-n\leqslant h_{x,i}\leqslant n-1$。

根据 P_x 的定义，$\overline{p}_i$ 表示相对于 p_i 的减量，使 $\overline{p}_i$ 的取值范围在最低优先级与最高优先级之间。当 $P_x=\langle 0,0,\cdots\rangle$ 时，其所表示的就是优先级不变迁的策略。这就意味着 ED/FR/S 和 ED/FR/M 策略是 SCP 和 MCP 策略的特例。而当 P_x 取不同的值时，任务的最坏响应时间是不同的，因而，任务的最坏响应时间 $R_i^{\Delta F}(x)$ 又与 P_x 相关。

首先将任务 τ_i 的响应时间分成两部分：正常执行阶段和错误检测恢复阶段。由于本文研究的故障模型假设 $T_F \geqslant \max_{\tau_j \in \Gamma} D_i$，因此每个任务在一个周期内只可能遇到一次故障爆发。在故障爆发之前任务 τ_i 处于正常执行阶段，相应的所有的任务都应该处在正常执行状态，也没有恢复故障的动作，在这个阶段，P_x 对任务的执行情况没有影响，因此这个阶段的最坏执行时间可以用式(2)来表示。

错误检测恢复阶段可以细分为故障持续阶段和故障恢复阶段。故障持续阶段的持续时间为 ΔF，在这个阶段故障出现的次数和时间分布是不可知的，我们可以这样认为，处于这个阶段的任务是不能正常工作的。因此这个阶段的时间固定为 ΔF。在故障恢复阶段，我们通过改变备选任务的优先级来满足截止时限的需求，此时 P_x 就对任务的最坏执行时间产生作用。这种作用有好有坏，我们需要找出将系统容错能力提至最高的 P_x。根据定义 2 和定义 3 可知，假如系统中有 n 个任务，则 P_x 有 $n(n-1)/2$ 种，如何使用较短的时间从中选出最佳的优先级组合需要设计算法是第 4 节的内容。

根据上述分析，任务 τ_i 的最坏响应时间的计算公式如下所示：

$$R_i^{\Delta F}(x) = R_i + \Delta F + I_i^{\Delta F}(x) + F_i(x) \tag{7}$$

其中，x 是一组给定的优先级变迁因子。

计算任务 τ_i 的最坏响应时间故障恢复阶段的时间必须考虑高优先级任务对其备选任务 $\bar{\tau}_i$ 的抢占所引起的干扰，这些任务包括主任务和备选任务。我们需要知道哪些主任务的优先级高于 $\bar{\tau}_i$，哪些备选任务的优先级高于 $\bar{\tau}_i$。根据给定的任务集合 $\Gamma=\{\tau_1,\tau_2,\cdots,\tau_n\}$ 和优先级变迁因子 x，对任意的任务 $\tau_i \in \Gamma$，可以构造以下集合：

$hp\bar{p}(x,i)$ 用于表示主任务优先级高于 P_x 的任务子集合。这个集合中的任务在执行时会抢占 $\bar{\tau}_i$，则其对应的主任务会影响 τ_i 的最坏响应时间。其公式表示形式为：$hp\bar{p}(x,i)=\{\tau_j \in \Gamma \mid p_j \geqslant \bar{p}_i\}$。

$sep\bar{p}(x,i)$ 用于表示主任务优先级小于或等于 $\bar{p}_i$ 的任务子集合。这个集合中的主任务不会抢占 $\bar{\tau}_i$ 的执行。其公式表示形式为：$sep\bar{p}(x,i)=\{\tau_j \in \Gamma \mid p_j \leqslant \bar{p}_i\}$。

$h\overline{pp}(x,i)$ 用于表示备选任务优先级高于 $\bar{p}_i$ 的任务子集合。这个集合中的任务若在 $\bar{\tau}_i$ 执行前或执行中出现故障，则其对应的备选任务会影响 $\bar{\tau}_i$ 的响应时间。其公式表示形式为：$h\overline{pp}(x,i)=\{\tau_j \in \Gamma \mid \bar{p}_j \geqslant \bar{p}_i\}$。

$s\overline{pp}(x,i)$ 用于表示备选优先级小于 $\bar{p}_i$ 的任务子集合。这个集合中的任务若在 $\bar{\tau}_i$ 执行过程中出现故障，则不会抢占 $\bar{\tau}_i$ 的执行。其公式表示形式为：$s\overline{pp}(x,i)=\{\tau_j \in \Gamma \mid \bar{p}_j \leqslant \bar{p}_i\}$。

$he\overline{pp}(x,i)$ 用于表示主任务优先级大于任务 $\bar{\tau}_i$ 的优先级，同时其备选任务的优先级小于 $\bar{\tau}_i$ 的优先级。其公式表示形式为：$he\overline{pp}(x,i)=\{\tau_j \in \Gamma \mid \bar{p}_j < \bar{p}_i \,\&\&\, p_j > \bar{p}_i\}$。

根据集合的定义，可以得到下列公式：

$$hp\bar{p}(x,i) \cup sep\bar{p}(x,i) = \Gamma \tag{8}$$

$$h\overline{pp}(x,i) \cup s\overline{pp}(x,i) \cup \{\tau_i\} = \Gamma \tag{9}$$

影响任务 $\bar{\tau}_i$ 执行的任务就在集合 $hp\bar{p}(x,i)$ 和 $h\overline{pp}(x,i)$ 中。下面分 3 种情况考虑：

1)任务既在集合 $hp\bar{p}(x,i)$ 中也在集合 $h\overline{pp}(x,i)$ 中

满足此条件的任务 τ_k 就是主任务优先级大于任务 $\bar{\tau}_i$ 的优先级，同时其备选任务的优先级也大于 $\bar{\tau}_i$ 的优先级。

无论是主任务运行还是备选任务在运行时都会抢占任务 $\bar{\tau}_i$，我们知道 τ_k 和 $\bar{\tau}_k$ 的最坏执行时间相等，即 $\bar{C}_k = C_k$。则任务 τ_k 对任务 τ_i 执行故障恢复时的扰动为：$\left\lceil \frac{R_i^{\Delta F} - R_i - \Delta F}{T_k} \right\rceil C_k$。

2)任务在集合 $hp\bar{p}(x,i)$ 中而不在集合 $h\overline{pp}(x,i)$ 中

满足此条件的任务 τ_k 就是主任务优先级大于任务 $\bar{\tau}_i$ 的优先级，同时其备选任务的优先级小于 $\bar{\tau}_i$ 的优先级。那么只有主任务 τ_k 会抢占任务 $\bar{\tau}_i$，如果在 ΔF 阶段结束任务 σ_k 出现故障，那么 $\bar{\tau}_k$ 就会被 $\bar{\tau}_i$ 抢占，$\bar{\tau}_k$ 将在 $\bar{\tau}_i$ 之后完成，它对 $\bar{\tau}_i$ 没有影响；如果在 ΔF 阶段结束任务 τ_k 没有出现故障，那么就可能抢占 $\bar{\tau}_i$；但是 τ_k 在这个周期的执行已经完成，必须等到下一个周期之后任务 τ_k 才会再次执行。则任务 τ_k 对任务 τ_i 执行故障恢复时的扰动为：$\left\lceil \frac{R_i^{\Delta F} - R_i - \Delta F - T_k}{T_k} \right\rceil C_k$。

3)任务不在集合 $hp\bar{p}(x,i)$ 中而在集合 $h\overline{pp}(x,i)$ 中

满足此条件的任务 τ_k 就是主任务优先级小于等于任务 $\bar{\tau}_i$ 的优先级，同时其备选任务的优先级大于 $\bar{\tau}_i$ 的优先级。那么只有备选任务 $\bar{\tau}_k$ 会抢占任务 $\bar{\tau}_i$，如果在 ΔF 阶段结束任务 τ_k 出现故障，那么 $\bar{\tau}_k$ 就会抢占 $\bar{\tau}_i$；如果在 ΔF 阶段结束任务 τ_k 没有出现故障，那么就不可能抢占 $\bar{\tau}_i$。则任务 τ_k 对任务 τ_i 执行故障恢复时的扰动为：$\left\lceil \frac{R_i^{\Delta F} - R_i - \Delta F}{T_k} \right\rceil C_k$。

根据上述分析，我们可以得到故障恢复阶段的计算公式如下：

$$I_i^{\Delta F} = \sum_{\tau_j \in hpp(i)} \left\lceil \frac{R_i^{\Delta F} - R_i - \Delta F}{T_j} \right\rceil C_j + \sum_{\tau_j \in hepp(i)} \left\lceil \frac{R_i^{\Delta F} - R_i - \Delta F - T_j}{T_j} \right\rceil C_j \tag{10}$$

$F_i(x)$ 的计算公式类似于式(5)和式(6)，这里就不再一一论述了，下面给出 $F_i(x)$ 的计算公式。

$$F_i(x) = \begin{cases} 2 * C_i, & \bar{p}_i = 1 \\ 2 * C_i + 2 * \sum_{\tau_j \in hpp(x,i)} C_j, & \bar{p}_i > 1 \end{cases} \tag{11}$$

$$F_i(x) = \begin{cases} 2 * C_i, & \bar{p}_i = 1 \\ C_i + \sum_{\tau_j \in hpp(x,i)} C_j + \max_{\tau_j \in \overline{hpp}(x,i)} C_j, & \bar{p}_i > 1 \end{cases} \tag{12}$$

其中式(11)适用于 SCP 策略而式(12)适用于 MCP 策略。

根据定义 3 和上面的分析可知，ED/FR/S 和 ED/FR/M 策略其实是 SCP 策略和 MCP 策略的特例；在所有任务容错优先级未发生变化的情况下，ED/FR/S 策略和 SCP 策略所计算的最坏响应时间应该相等，这个结论也适用于 ED/FR/M 策略和 MCP 策略。下面的定理 1 和定理 2 可以证明。

定理 1 一个容错实时任务集合 $\Gamma=\{\tau_1,\tau_2,\cdots,\tau_n\}$，对于任意的故障持续时间 ΔF 和 $\tau_i \in \Gamma$，当 $P_x=\langle 0,0,\cdots,0\rangle$ 时，采用 ED/FR/S 策略，由式(3)计算所得到的最坏响应时间等于在 SCP 策略下计算式(7)所得的 $R_i^{\Delta F(x)}(x)$。

证明：当 $P_x=\langle 0,0,\cdots,0\rangle$ 时，对任意任务 $\tau_j \in \Gamma$ 都有主任务优先级等于备选任务优先级，也就是 $p_j=\bar{p}_j$。那么可知 $h\overline{pp}(x,i)=hp\bar{p}(x,i)=hp(x,i)$，$he\overline{pp}(x,i)=\emptyset$，则由式(7)、式(10)和式(11)可得：

$$R_i^{\Delta F}(x) = R_i + \Delta F + I_i^{\Delta F}(x) + F_i(x)$$

$$=R_i+\Delta F+\sum_{\tau_j\in hpp(i)}\left\lceil\frac{R_i^{\Delta F}-R_i-\Delta F}{T_j}\right\rceil C_j+F_i(x)$$

$$=R_i+\Delta F+\sum_{\tau_j\in hp(i)}\left\lceil\frac{R_i^{\Delta F}-R_i-\Delta F}{T_j}\right\rceil C_j+F_i(x)\quad①$$

$$F_i(x)=\begin{cases}2*C_i, & \overline{p}_i=1\\ 2*C_i+2*\sum\limits_{\tau\in\overline{hpp}(x,i)}C_j, & \overline{p}_i>1\end{cases}$$

$$=\begin{cases}2*C_i, & \overline{p}_i=1\\ 2*C_i+2*\sum\limits_{\tau_j\in\overline{hpp}(x,i)}C_j, & \overline{p}_i>1\end{cases}$$

$$=\begin{cases}2*C_i, & p_i=1\\ 2*C_i+2*\sum\limits_{\tau_j\in hp(x,i)}C_j, & p_i>1\end{cases}\quad②$$

综合①②，$R_i^{\Delta F(x)}(x)$的值等于式(3)计算所得的最坏响应时间。证毕。

定理 2 一个容错实时任务集合 $\Gamma=\{\tau_1,\tau_2,\cdots,\tau_n\}$，对于任意的故障持续时间 ΔF 和 $\tau_i\in\Gamma$，当 $P_x=\langle 0,0,\cdots,0\rangle$时，采用 ED/FR/M 策略，由式(3)计算所得到的最坏响应时间等于在 MCP 策略下计算式(7)所得的 $R_i^{\Delta F(x)}(x)$。

证明：当 $P_x=\langle 0,0,\cdots,0\rangle$时，对任意任务 $\tau_i\in\Gamma$ 都有主任务优先级等于备选任务优先级，也就是 $p_i=\overline{p}_i$。那么可知 $h\overline{pp}(x,i)=hp\overline{p}(x,i)=hp(x,i)$，$he\overline{pp}(x,i)=\varnothing$，则由式(7)、式(10)和式(12)可得：

$$R_i^{\Delta F}(x)=R_i+\Delta F+I_i^{\Delta F}(x)^+F_i(x)$$

$$=R_i+\Delta F+\sum_{\tau_j\in hpp(i)}\left\lceil\frac{R_i^{\Delta F}-R_i-\Delta F}{T_j}\right\rceil C_j+F_i(x)$$

$$=R_i+\Delta F+\sum_{\tau_j\in hp(i)}\left\lceil\frac{R_i^{\Delta F}-R_i-\Delta F}{T_j}\right\rceil C_j+F_i(x)\quad①$$

$$F_i(x)=\begin{cases}2*C_i, & \overline{p}_i=1\\ C_i+\sum\limits_{\tau_j\in\overline{hpp}(x,i)}C_j+\max\limits_{\tau_j\in\overline{hpp}(x,i)}C_j, & \overline{p}_i>1\end{cases}$$

$$=\begin{cases}2*C_i, & \overline{p}_i=1\\ C_i+\sum\limits_{\tau_j\in\overline{hpp}(x,i)}C_j+\max\limits_{\tau_j\in\overline{hpp}(x,i)}C_j, & \overline{p}_i>1\end{cases}$$

$$=\begin{cases}2*C_i, & p_i=1\\ C_i+\sum\limits_{\tau_j\in hp(x,i)}C_j+\max\limits_{\tau_j\in hp(x,i)}C_j, & p_i>1\end{cases}\quad②$$

综合①②，$R_i^{\Delta F(x)}(x)$的值等于式(3)计算所得的最坏响应时间。证毕。

4.3 实例分析

现对表 1 和表 2 中的任务集合采取 SCP 和 MCP 策略，并指定优先级变迁因子为 $P_x=\langle 0,1,-1\rangle$，也就是提高任务 τ_3 备选任务的优先级，并降低任务 τ_2 备选任务的优先级。计算得到的任务最坏响应时间如表 3 和表 4 所列。当 $\Delta F=30$ 时，系统在 ED/FR/S 策略下不可调度，而在 SCP 策略 $P_x=\langle 0,1,-1\rangle$下可调度；可以看出，提高任务 τ_3 备选任务的优先级，缩短了任务 τ_3 的最坏响应时间，并让任务 τ_3 在其截止期限内完成，虽然增加了任务 τ_2 的最坏响应时间，但也能满足截止期限的要求，重要的是整个系统变得可以调度。当增加 ΔF 时间到 40 时，系统在 SCP 策略下仍然可以调度，虽然任务 τ_2 的响应时间增大了，但这种代价是值得的。

比较 ED/FR/M 策略和 MCP 策略，我们可以得到相似的结论：某些在 ED/FR/M 和 ED/FR/S 两种策略下不可调度的系统，使用 SCP 和 MCP 策略后变得可以调度，这证实优先级变迁在某些情况下可以提高系统的容错性能。

表 3 SCP 的执行情况

任务	T_i	C_i	D_i	R_i^{20}	R_i^{30}	R_i^{40}
τ_1	300	10	300	50	60	70
τ_2	600	100	600	590	600	610
τ_3	800	110	800	710	720	730

表 4 MCP 的执行情况

任务	T_i	C_i	D_i	R_i^{30}	R_i^{60}	R_i^{70}
τ_1	300	15	300	75	105	115
τ_2	700	100	700	590	620	630
τ_3	800	150	800	755	785	795

5 优先级变迁因子搜索算法

前面对容错优先级变迁的策略下任务可调度性进行了分析，下一步的主要任务就是设计优先级变迁因子搜索算法；因为对一组任务，使用不同优先级变迁因子会得到不同的 ΔF，使系统的容错能力更强。下面引入函数 $\Delta F(x)$，其表示在给定 P_x 的情况下，ΔF 所能达到的最大值。

定义 4 容错实时系统中最大故障时长函数为 $\Delta F(x)$，对容错实时任务集合 $\Gamma=\{\tau_1,\tau_2,\cdots,\tau_n\}$，对于任意的 $\tau_i\in\Gamma$ 都有 $R_i^{\Delta F(x)}(x)\leqslant D_i$；存在任务满足 $R_i^{\Delta F(x)+1}>D_i$。

也就是说，若 ΔF 比 $\Delta F(x)$的值更大，那么一些任务可能不可调度。特别地，当 $\Delta F=\Delta F(x)+1$ 时，至少存在一个任务 τ_i，找不到一组优先级变迁因子 x 不满足 $R_i^{\Delta F(x)+1}(x)<D_i$。则称这样的任务为临界任务，用 $D(x)$来表示临界任务集合，即 $D(x)=\{\tau_i\in\Gamma|R_i^{\Delta F(x)+1}>D_i\}$。

下面根据 MCP 策略的性质，设计了一种容错优先级变迁因子搜索算法。

假如系统中有 n 个任务，则 P_x 共有 $n(n-1)/2$ 种，如果要找出一组最佳的变迁因子，最直接的办法就是遍历所有的组合并测出每种组合最大的 ΔF，显然这种做法耗时巨大。遍历每种组合并从小到大逐一测试 ΔF 的改变，但是测试过程中某些组合会被淘汰掉。ΔF 值的取值范围在零到$\min\limits_{\tau_i\in\Gamma}D_i-1$之间，那么这样的算法的计算任务集最坏响应时间的循环次数为$(\min\limits_{\tau_i\in\Gamma}D_i-1)*n(n-1)/2$，也就是 $O(n^2)$。显然这是不可接受的。接下来分析提高系统容错能力的根本因素，并在此基础上设计并实现容错优先级变迁因子搜索算法。

下面根据式(10)－式(12)，分析 SCP 策略和 MCP 策略对最高优先级任务、最低优先级任务和处于中间优先级任务 3 种不同优先级任务响应时间的影响，这里假设 τ_1 的优先级最高，τ_n 的优先级最低，τ_i 表示任意一个中间优先级任务。

1)最高优先级任务 τ_1

由于 $hp(1)=\varnothing$，则必然有 $hp(1)\subseteq hp\overline{p}(x,1)$，同样 $he\overline{pp}(x,1)=\varnothing$，SCP 策略和 MCP 策略计算得到的结果肯定是大于或等于 ED/FR/S 策略和 ED/FR/M 策略，增加的部分是其他份备选任务抢占备选任务 $\overline{\tau}_1$ 的响应时间。因此，采取 SCP 策略和 MCP 策略只会延长 τ_1 的响应时间，或者不会改变。

2)最低优先级任务 τ_n

对于 τ_n 来说，$p_j\leqslant\overline{p}_j$。则根据式(10)和式(11)可知，$\overline{\tau}_n$ 的响应时间有可能减少，因为 $\overline{\tau}_n$ 的优先级提高，就会抢占其

他主任务或备选任务的执行时间。采取 SCP 策略和 MCP 策略会减少 τ_n 的响应时间。

3)中间优先级任务 τ_i

τ_i 的情况比较复杂，它的容错优先级可以提高也可以降低，但是它的容错优先级的提高不一定会减少任务 τ_i 的响应时间，根据 $I_i^{\Delta F}(x)$ 的计算公式可知，容错优先级的提高会减少计算项 $\left\lceil \frac{R_i^{\Delta F}-R_i-\Delta F}{T_k} \right\rceil C_k$ 但是会增加计算项 $\left\lceil \frac{R_i^{\Delta F}-R_i-\Delta F-T_k}{T_k} \right\rceil C_k$，所以不能确定容错优先级提高后的效果。而降低 τ_i 的优先级则一定会增加任务 τ_i 的响应时间。

根据以上的分析，可以得出 SCP 策略和 MCP 策略的性质：在 P_x 给定的情况下，如果最高优先级任务不可调度时，则改变最高优先级任务的容错优先级是不能缩短它的响应时间的；而当中间优先级任务和最低优先级任务不可调度时，则通过提高优先级任务的容错优先级可能会缩短它的响应时间。所以在设计容错优先级变迁因子搜索算法时，主要针对的是处于中间优先级任务和最低优先级任务不可调度的情况。

假设 τ_i 不可调度，则可以考虑两种手段：第一种是将在集合 $hp\bar{p}(x,i)$ 中且在集合 $h\overline{pp}(x,i)$ 中的任务 τ_k 的容错优先级降低至小于 τ_i 的容错优先级，则任务 τ_k 对备选任务 $\bar{\tau}_i$ 的干扰由 $\left\lceil \frac{R_i^{\Delta F}-R_i-\Delta F}{T_k} \right\rceil C_k$ 降至 $\left\lceil \frac{R_i^{\Delta F}-R_i-\Delta F-T_k}{T_k} \right\rceil C_k$。第二种是将任务不在集合 $hp\bar{p}(x,i)$ 中而在集合 $h\overline{pp}(x,i)$ 中的任务 τ_k 的容错优先级降低至小于 τ_i 的容错优先级，这样，则任务 τ_k 对备选任务 $\bar{\tau}_i$ 的干扰由 $\left\lceil \frac{R_i^{\Delta F}-R_i-\Delta F}{T_k} \right\rceil C_k$ 降至零。综合以上分析，我们用 *reductions* 来表示这两种情况下需要改变并降低容错优先级任务的集合。而且 *reductions* 中任务的容错优先级只要降低到 $\bar{p}_i$ 以下就可以缩短任务的响应时间，最简单的就是交换 τ_k 和 τ_i 的容错优先级。

根据 4.3 节的分析可知，当某些任务集合在 ED/FR/S 策略或 ED/FR/M 策略下均不可调度时，可以采取 SCP 策略或 MCP 策略，使任务集合可调度。满足要求的优先级变迁因子可能有很多组。为了最大限度地提高系统的容错能力，设计了一种容错优先级变迁因子搜索算法 FPFCS，描述如下：

算法 1 fault-tolerant priority change factor configuration search algorithm

(1)根据 FP(Γ)分配 p_i，i=1,2,…,n

(2)τ_{max}←最高优先级任务

(3)τ_{min}←最低优先级任务

(4)P_x←(0,0,…,0)；P_x^*←P_x

(5)$L \leftarrow \min_{\tau_i \in \Gamma} D_i - 1$

(6)ΔF←ΔF(x)；ΔF*←ΔF

(7) while (TRUE)

(8) 计算 $R_i^{\Delta F}(x)$，i=1,2,…,n

(9) if ($\forall \tau_i \in \Gamma, R_i^{\Delta F}(x) \leqslant D_i$) then

(10) $P_x^* \leftarrow P_x$

(11) ΔF*←ΔF

(12) ΔF←ΔF+1；

(13) if (ΔF>L) exit while

(14) if ($\tau_{max} \in D(x)$) exit while

(15) τ_i 为 D(x)中的一个任务

(16) if (reductionset(x,i)=∅) exit while

(17) τ_j 为 reductionset(x,i)中的一个任务，$h(x,j) \leftarrow \bar{p}_i - p_j$，$h(x,i) \leftarrow \bar{p}_j - p_i$

(18) end if

(19) end while

(20)$P_x \leftarrow P_x^*$

(21)ΔF←ΔF*

容错优先级变迁因子搜索算法可分为两步：首先计算出给定的任务集合 Γ 在 ED/FR/S 策略或 ED/FR/M 下的 ΔF 和 P_x 的值(见算法第 1 行－第 6 行)，然后在此时就对 Γ 采取容错任务优先级交换的方法来测试能否提高系统的容错能力(见算法第 9 行－第 21 行)。

6 仿真实验

为了说明 SCP 策略和 MCP 策略在提高系统容错能力方面的性能，本文对 $\Delta F(0)$ 和 $\Delta F(x)$ 进行比较，因而将 $f(\Delta F)=(\Delta F(0)-\Delta F(x))/\Delta F(0)$ 作为衡量系统容错能力好坏的一个性能指标。很显然，在 ED/FR/S 策略或 ED/FR/M 策略下，$\Delta F(0)=0$。这里采用 RMS 调度算法来分配任务的优先级。

本实验是在主频为 3.0GHz 的 CPU，1GB 内存的 PC 机上的进行的。模拟了 10000 组任务数据，每组有 5 个任务，每个任务的执行时间 C_i，周期 T_i，截止期限 D_i 是随机产生的。但是对于任务组 $\Gamma_m(m=1,2,\cdots,10000)$需满足以下条件：

(1) $\forall \tau_i \in \Gamma_m, T_i \in [5*\lceil m/10 \rceil, 50*\lceil m/10 \rceil]$；

(2) $\forall \tau_i \in \Gamma_m, 10*C_i \leqslant D_i \leqslant T_i$；

(3) $\forall \tau_i \in D(0), p_{min} < p_i < p_{max}$，其中 p_{min} 表示最低的优先级，p_{max} 表示最高的优先级。

对每组任务采用 FPFCS 算法搜索最佳容错变迁因子并计算 $f(\Delta F)$，实验结果如图 5 和图 6 所示。图 5 是比较策略 SCP 和 ED/FR/S 得到的 $f(\Delta F)$ 散点图，图 6 是比较策略 MCP 和 ED/FR/M 得到的 $f(\Delta F)$ 散点图。横坐标表示处理器的利用率，纵坐标是 $f(\Delta F)$ 的值。

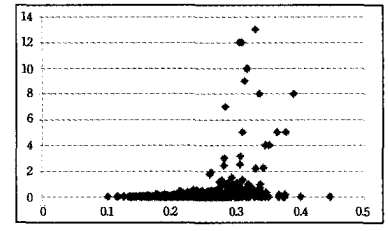

图 5 SCP 和 ED/FR/S 策略的 $f(\Delta F)$ 的散点图

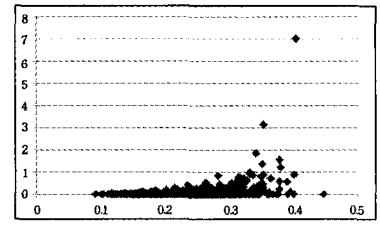

图 6 MCP 和 ED/FR/M 策略的 $f(\Delta F)$ 的散点图

分析图中的数据，除了部分组任务 $f(\Delta F)$ 等于零以外，大部分任务组的 ΔF 都得到了提高。以图 5 中的数据为例，

(下转第 14 页)

位时间内成功识别的标签个数。当标签编码位数为 8bit 时，QTA、QT-CBP 和 BTD 算法在标签个数较少时吞吐率相近，新算法在标签饱和度较低时，能保持较高的吞吐率。随着标签饱和度升高，新算法较其他 3 种算法的优势明显，因为新算法根据堆栈顺序，维码数依次出栈，减少了不必要的搜索时隙。当编码位数相同时，新算法较 QT-CBP 和 BTD 算法，吞吐率平均增加了 59.3%，78.8%。

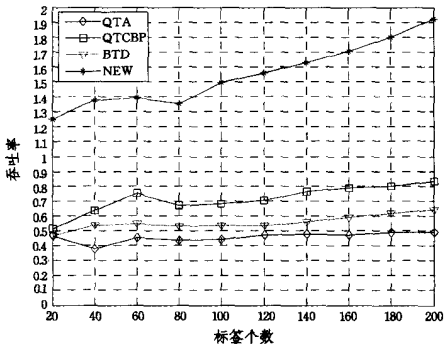

图 6　8bit 吞吐率对比图

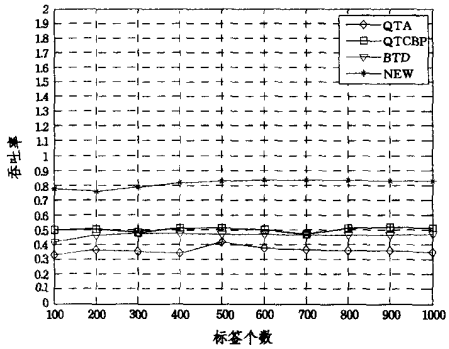

图 7　16bit 吞吐率对比图

结束语　本文结合智能分离的增强型时隙二进制算法(IS-ESBT)和自适应维分编码的思想，提出了一种基于维码数的自适应混合防碰撞算法。新算法较传统算法有以下几点优势：1)对可读范围内的标签进行逻辑分组，自适应选择分维位数，实现标签的有序识别；2)维码数堆栈阶段，只回复其维码数，减少了阅读器和标签之间的通信量；3)根据堆栈的维码数，采用不同的策略推算其维 ID，减少了空白时隙的产生。仿真实验证明，改进算法具有较大的优越性。

参 考 文 献

[1] 黄玉兰. 射频识别(RFID)核心技术详解[M]. 北京：人民邮电出版社，2010：2-5

[2] 康东，石喜勤，李勇鹏，等. 射频识别(RFID)核心技术与典型应用开发实例[M]. 北京：人民邮电出版社，2008：165-178

[3] 陈颖. 一种新的多阅读器防碰撞算法的研究[J]. 杭州电子科技大学学报，2013，32(5)：112-115

[4] 石封茶，崔琛，余剑. 基于标签运动的一种新型 RFID 防碰撞算法[J]. 计算机科学，2013，40(6)：76-79

[5] Abramson N. The ALOHA System-Another Alternative for Computer Communications[J]. Fall Joint Computer Conference, AFIPS Conference Proceedings，1970，37：281-285

[6] 萧耀友，胡钢，魏钦伟，等. 基于二进制树分解的动态防碰撞算法[J]. 通信技术，2011，44(1)：99-101

[7] 郑嘉利，覃团发，倪光南. Tree-based backoff protocol for fast RFID tag identification[J]. 中国邮电高校学报(EI 源期刊)，2013，20(2)：37-41

[8] 宋瑞玲，高仲合. RFID 防碰撞算法研究[J/OL]. 计算机工程与应用，http://www.cnki.net/kcms/doi/10.3778/j.issn.1002-8331.1309-0084.html，2014-02-13

[9] 韦冬雪，郑嘉利，李亮亮，等. 一种新颖的自适应多叉树防碰撞算法的研究[J]. 计算机科学，2013，40(10)：52-55

[10] 周信，刘晔. 一种基于码距反演的 RFID 防碰撞算法[J]. 计算机工程与应用，2012，48(8)：214-217

[11] 李致金，周杰，乔杰，等. 自适应维分编码 RFID 防碰撞算法研究及优化[J]. 通信学报，2013，34(9)：185-190

[12] Lee H，Kim J. QT-CBP：A new RFID tag anti-collision algorithm using collision bit positioning[M]. Emerging Directions in Embedded and Ubiquitous Computing. Springer Berlin Heidelberg，2006：591-600

(上接第 6 页)

其提高的比例达到了 60%，持平的比例仅为 40%，没有 ΔF 降低的现象出现，处理器的利用率在 0.2～0.3 时 ΔF 提高的比率最大。图 6 中得到提高的比例达到了 50%，持平的比例为 50%，最大提高幅度出现在图 5 中。相比而言，SCP 策略比 ED/FR/S 策略提高得更多一些。

结束语　本文的主要贡献是，针对突发性故障容错策略所存在的缺陷，提出可以改变容错任务优先级策略 SCP 和 MCP，分析计算了采用新策略下实时任务的最坏情况响应时间，经过研究和实验，证明了这两种策略能够有效地提高实时系统在突发性故障模式下的容错能力。

参 考 文 献

[1] Burns A，Davis R. I，Punnekkat S. Feasibility analysis of fault-tolerant real-time task sets[C]//Proceedings of the Eighth Euromicro Workshop Real-Time Systems. L'Aquila，1996：29-33

[2] Many F，Doose D. Scheduling analysis under fault bursts[C]// 2011 17th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS). Chicago，IL，2011：113-122

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

[4] Audsley N C，et al. Hard Real-Time Scheduling：The Deadline Monotonic Approach[C]//Proceedings of Eighth IEEE Workshop on Real-time Operating Systems and Soft ware. Atlanta，CA，USA，1991：133-137

[5] 邓建波，张立臣，邓惠敏. 异构分布式系统混合型实时容错调度算法[J]. 计算机科学，2011，38(3)：87-92

[6] 郭锐锋，刘娴，丁万夫，等. 回卷恢复模型下容错实时系统的可调度性分析[J]. 计算机科学，2013，34(6)：1334-1338

[7] Joseph M，Pandya P. Finding response times in a real-time system[J]. The Computer Journal，1986，29(5)：390-395

[8] 李俊，阳富民，卢炎生. 一种可行的容错实时系统可调度性分析[J]. 软件学报，2005，16(8)：1513-1522

[9] Lima G 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