允许多处理机故障的实时任务容错调度算法

殷进勇　　顾国昌
(哈尔滨工程大学计算机科学与技术学院　哈尔滨　150001)

摘　要：随着故障处理机个数增加，基于主/从版本技术的实时容错调度算法对处理机利用率迅速下降。论文提出了一种能够调度周期和非周期混合实时任务的容错调度算法，该算法允许多个处理机出现故障。把 DS(Deferrable Server)算法扩展到多处理机系统，可在系统中设置多个 DS 服务器来处理非周期任务。当处理机出现故障时，通过在其他处理机上回卷执行故障任务，保证了系统的容错性能。实验结果表明，该算法能够使系统接收的所有实时任务满足截止期限并有效地减少了所需的处理机数。

关键词：实时容错调度；整体调度；混合任务；延时服务器；多处理机故障
中图分类号：TP316　　**文献标识码**：A　　**文章编号**：1009-5896(2010)02-0444-05
DOI: 10.3724/SP.J.1146.2009.00263

A Real-time Fault-tolerant Scheduling Algorithm for Multiple Processor Faults

Yin Jin-yong　　Gu Guo-chang
(*College of Computer Science and Technology*, *Harbin Engineering University*, *Harbin* 150001, *China*)

Abstract: The existing fault-tolerant scheduling algorithms are mainly based on primary/backup copies technology and the utilization of processor decreases greatly with the number of faults increases. In this paper, a real-time fault-tolerant scheduling algorithm is proposed to schedule periodic and aperiodic tasks jointly and tolerate multiple processor faults. The DS (Deferrable Server) algorithm is extended to the multiprocessors system, and several deferrable servers can be set to schedule aperiodic tasks. The faults can be tolerated by tasks' rollback executing on the other processors. The experimental results demonstrate that this algorithm can guarantee all accepted tasks' deadlines and decreases the number of required processor significantly.

Key words: Real-time fault-tolerant scheduling; Global scheduling; Hybrid tasks; Deferrable server; Multiple processor faults

1　引言

在实时系统中，每个实时任务都有一个截止期限，如果由于处理机故障使得实时任务错失截止期限，则造成严重后果，所以需要为实时系统提供一定的容错能力。在某些实时系统中，周期任务和非周期任务同时存在，而现有的容错调度算法[1−4]主要针对同类任务提出的。在单处理机系统中，对周期任务和非周期任务的混合调度算法主要有基于服务器的DS(Deferrable Server)算法[5]和基于空闲时间的SSA(Slack Stealing Algorithm)算法[6]及其它们扩展版本，而在多处理机系统中多数采用分割调度(partitioned scheduling)方式，首先把任务分配到各个处理机上，再根据单处理机调度算法进行调度。文献[7]中的HFTS算法就是一种基于分割调度和主/从版本技术的混合实时任务容错调度算法。

本文把DS算法和回卷恢复技术扩展到多处理机系统，采用整体调度(global scheduling)方式调度周期任务和DS服务器，在DS服务器上调度非周期任务，并通过任务的回卷恢复保证系统的容错能力。本文内容安排如下：首先给出了系统和任务模型并在Baker[8]的基础上分析了周期任务和DS服务器的可调度性，重新推导了可调度性判定条件；其次分析了非周期任务在DS服务器上的可调度性，给出了DS-EDF(Earliest Deadline First on Deferrable Server)调度算法；再次通过设置检测点和任务回卷恢复，分析了在系统中一个或多个处理机出现故障时的实时任务的可调度性，提出了一个允许多个处理机故障的容错调度算法MFTS(Multiple Faults Tolerance Scheduling)；最后通过仿真实验证明了算法的有效性。

2　系统和任务模型

硬件系统由 m 个共享内存的处理机组成，系统调度的任务包括周期任务 $\tau=\{\tau_1,\tau_2,\cdots,\tau_n\}$，非周期任务 $J=\{J_1,J_2,\cdots,J_k\}$和服务器 $S=\{S_0,S_1,\cdots,S_h\}$。每

2009-03-04 收到，2009-09-25 改回
国家自然科学基金(60873036)资助课题
通信作者：殷进勇　yinjinyong@yahoo.com

个周期任务 $\tau_i \in \tau$ 包含执行时间 C_i，相对截止期限 D_i 和周期 T_i，$C_i \leqslant D_i \leqslant T_i$。不失一般性，设 $D_1 \leqslant D_2 \leqslant \cdots \leqslant D_n$。任务 τ_i 在每个周期内执行一次，称为一个作业。用 τ_{ij} 表示任务 τ_i 在第 j 个周期内的作业，τ_{ij} 的到达时间 $A_{ij}=(j-1)\times T_i$，绝对截止期限 $d_{ij}=A_{ij}+D_i$。每个非周期任务 $J_i \in J$ 包含到达时间 A_i，执行时间 C_i 和相对截止期限 D_i，绝对截止期限 $d_i=A_i+D_i$。服务器 S 是一组优先级最高的周期任务，每个服务器 $S_i \in S$ 包含执行时间 C_i^s 和周期 T_i^s，并且 $T_0^s=T_1^s=\cdots=T_h^s$，$C_0^s<T_0^s, C_1^s=T_1^s, \cdots, C_h^s=T_h^s$。在 S_i 的每个周期内，只要执行时间 C_i^s 没有耗尽，非周期任务可在本周期内的任意时刻执行。

3　周期任务可调度性分析

定义 1(需求量)　在时间区间 $(t,t+\Delta)$ 上的需求量定义为绝对截止期限包含在区间 $(t,t+\Delta)$ 内的所有作业在区间 $(t,t+\Delta)$ 内的执行时间总和。

定义 2(负载)　在时间区间 $(t,t+\Delta)$ 上的负载定义为 W/Δ，其中 W 为在时间区间 $(t,t+\Delta)$ 上的需求量。

定义 3　(λ-忙碌期)　如果在时间区间 (t_1,t_2) 上的负载大于等于 $m(1-\lambda)+\lambda$，那么称时间区间 (t_1, t_2) 为 λ-忙碌期。如果不存在 $t_0<t_1$，使得区间 (t_0, t_2) 上的负载大于等于 $m(1-\lambda)+\lambda$，称 (t_1, t_2) 为最大 λ-忙碌期。

如果任务 τ_k 的第 j 个作业 τ_{kj} 错失了截止期限，并且是系统中第一个错失截止期限的作业，称 τ_k 为问题任务，区间 $[A_{kj},d_{kj}]$ 是一个 C_k/D_k-忙碌期，问题任务 τ_k 存在惟一最大 C_k/D_k-忙碌期[8]，称为问题窗口，用 $(t,t+\Delta)$ 表示且 $\Delta \geqslant D_k$，其中 $t+\Delta$ 等于作业 τ_{kj} 错失的截止期限 d_{kj}。如图 1 所示，在问题窗口内，每个任务 τ_i 包含头部，主体和尾部三部分，其中头部为区间 $[t, t+\min(\Delta, T_i-\Phi)$，$\Phi=t-t'$，$t'$ 为任务 τ_i 某个作业的释放时间且 $t'<t<t'+T_i$；如果 τ_i 的某个作业在 t'' 释放，且 $t''<t+\Delta<t''+T_i$，称 $[t'',t+\Delta)$ 为尾部，用 δ 表示；窗口的剩余部分为主体。根据 Baker 的分析，在 τ_k 的问题窗口内，任务 $\tau_i (i<k)$ 头部的需求量至多为 $\max(0,C_i-\lambda\Phi)$，总需求量 W_i 至多为 $nC_i+\max(0,C_i-\lambda\Phi)$，其中，$\Phi=nT_i+C_i-\Delta$，$n=\lfloor(\Delta-C_i)/T_i\rfloor+1$，$\lambda=C_k/D_k$，任务 τ_i 对 τ_k 的负载 W_i/Δ 至多为

$$\beta_i=\begin{cases}\dfrac{C_i}{T_i}\left(1+\dfrac{T_i-C_i}{D_k}\right), & \dfrac{C_i}{T_i}\leq\lambda\\[2ex] \dfrac{C_i}{T_i}\left(1+\dfrac{T_i-C_i}{D_k}\right)+\dfrac{C_i-\lambda T_i}{D_k}, & \dfrac{C_i}{T_i}>\lambda\end{cases} \tag{1}$$

在任务 τ_k 的问题窗口 $(t, t+\Delta)$ 内，服务器 S_0 的主体和尾部的需求量与一般任务相同，等于 nC_0^s。由于非周期任务在 T_0^s 的任何时刻均可执行，所以头部的最大需求量不再是 $\max(0,C_0^s-\lambda\Phi)$，而是 $\min(T_0^s-\Phi, C_0^s)$。所以 S_0 的总需求量 $W_0 \leqslant nC_0^s+\min(T_0^s-\Phi, C_0^s)$。

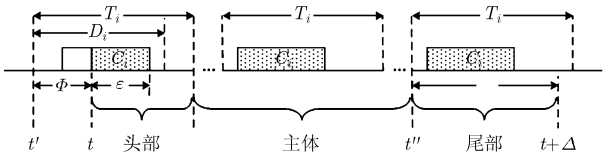

图 1　问题窗口中的头部、主体和尾部

定理 1　在任务 τ_k 的问题窗口 $(t,t+\Delta)$ 内，服务器 S_0 的负载 W_0/Δ 至多为

$$\beta_0=\frac{C_0^s}{T_0^s}\left(1+\frac{2T_0^s-C_0^s}{D_k}\right)$$

证明　$W_0 \leqslant nC_0^s+\min(T_s-\Phi, C_0^s)$

$\leqslant nC_0^s+C_0^s=(n+1)C_0^s$

$n=\lfloor(\Delta-C_0^s)/T_0^s\rfloor+1 \leqslant (\Delta-C_0^s)/T_0^s+1$

$=(T_0^s+\Delta-C_0^s)/T_0^s$

$\therefore\quad W_0 \leqslant (1+(T_0^s+\Delta-C_0^s)/T_0^s)C_0^s$

$=(C_0^s/T_0^s)(2T_0^s+\Delta-C_0^s)$

$W_0/\Delta=(C_0^s/T_0^s)(2T_0^s+\Delta-C_0^s)/\Delta=(C_0^s/T_0^s)\cdot\left(1+\dfrac{2T_0^s-C_0^s}{\Delta}\right) \leqslant \dfrac{C_0^s}{T_0^s}\left(1+\dfrac{2T_0^s-C_0^s}{D_k}\right)$　证毕

定理 2　对于周期任务 $\tau=\{\tau_1,\tau_2,\cdots,\tau_n\}$ 和服务器 S_0，如果每个任务 $\tau_i \in \tau$ 满足以等式(2)，那么 τ 和 S_0 可在 m 个处理机上被 DM(Deadline Monotonic) 算法调度。

$$\beta_0+\sum_{i=1}^{k-1}\beta_i \leq m(1-C_k/D_k) \tag{2}$$

证明过程请参阅文献[9]，这里不再赘述。

推论 1　对于周期任务 $\tau=\{\tau_1,\tau_2,\cdots,\tau_n\}$ 和服务器 $S=\{S_0,S_1,\cdots,S_h\}$，如果 τ 和 S_0 可在 m-h 个处理机上被 DM 算法调度，那么 τ 和 S 可在 m 个处理机上被 DM 算法调度。

证明　服务器 $\{S_1,\cdots,S_h\}$ 至多需要 h 个处理机，如果 τ 和 S_0 可在 m-h 个处理机上被 DM 算法调度，那么 τ 和 S 可在 m 个处理机上被 DM 算法调度。

证毕

4　非周期任务可调度性分析

虽然每个服务器可以在不同的处理机上执行，但是每个服务器上的非周期任务只能串行执行，又因为处理机完全相同，所以在服务器上调度非周期任务与在单处理机上完全相同。用 $V(t)$ 表示在 t 时刻已到达系统但截止期限还没有过期的所有非周期

任务，即 $V(t)=\{J_i|A_i\leqslant t\leqslant d_i\}$。在 t 时刻，由于 $V(t)$ 中的任务可能在前一个忙碌期内已经执行完毕，所以它对后到达任务的可调度性没有影响，用 $S(t)$表示 $V(t)$中当前忙碌期中的任务。综合利用率 $U(t)=\sum_{J_i\in S(t)}C_i/D_i$，在任意时刻 t，如果 $U(t)\leqslant 1$，那么非周期任务能被 EDF 算法在单处理机上调度[10]。

由于服务器 S_0 的执行时间 C_0^s 小于周期 T_0^s，所以可能在总的执行时间足够的情况下，由于执行了早到达的低优先级任务而导致晚到达的高优先级任务错失截止期限。如图 2 所示，$T_0^s=10$，$C_0^s=6$，非周期任务 J_1 和 J_2 的到达时间，执行时间和相对截止期限分别为 $A_1=0$，$C_1=14$，$D_1=28$ 和 $A_2=13$，$C_2=4$，$D_2=7$。在 EDF 算法下，由于过早的执行了 J_1，使得 J_2 错失了截止期限，而在 S_0 的第 3 个周期内还有一个单位的执行时间没有利用。

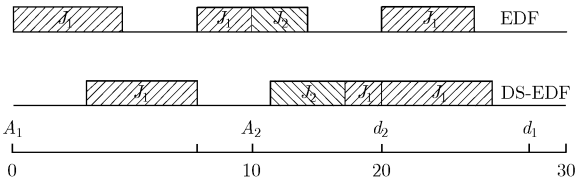

图 2 任务 J_1 和 J_2 在服务器 S_0 上的调度结果

由此可见，问题的关键是如何分配服务器 S_0 的执行时间，为此提出了 DS-EDF 算法。算法的主要思想是在满足截止期限的前提下，尽量推迟任务的执行。在服务器的每个周期内，通过设置任务的释放时间来控制任务的执行。在服务器的第 j 个周期内，非周期任务 J_i 的释放时间 r_{ij} 由式(3)确定。在 DS-EDF 算法下，非周期任务 J_1 和 J_2 均在其截止期限内完成。

$$r_{ij}=\begin{cases}\max(A_i,\,(j-1)T_0^s), & d_i<jT_0^s\\ \max(A_i,\,jT_0^s-C_0^{\mathrm{res}}), & d_i<jT_0^s\end{cases} \tag{2}$$

其中 C_0^{res} 为服务器 S_0 在本周期内的剩余执行时间。

定理 3 假设处理机 P 和 P' 的完全相同，到达 P 的非周期任务为 $J=\{J_1,J_2,\cdots\}$，由 J 形成的忙碌期依次记为 $Bp_1,Bp_2,\cdots$；P' 上有一个服务器 S_0，到达 S_0 的非周期任务为 $J'=\{J_1',J_2',\cdots\}$，$A_i'=A_i(i=1,2,\cdots)$并采用 DS-EDF 算法进行调度。如果 $C_i'=C_iC_0^s/T_0^s$，那么 S_0 在忙碌期 Bp_i 内可提供给非周期任务的执行时间 $C_i^p\geqslant|Bp_i|C_0^s/T_0^s$，$|Bp_i|$表示忙碌期 Bp_i 的长度。

证明 如图 3(a)所示，每个忙碌期 Bp_i 包括 h_i，b_i 和 t_i 三部分，分别用 C_{hi}^p，C_{bi}^p 和 C_{ti}^p 表示在 h_i，b_i 和 t_i 内 S_0 可提供的执行时间；分别用 C_{hi}^c，C_{bi}^c 和 C_{ti}^c 表示在 h_i，b_i 和 t_i 内非周期任务消耗的时间。为方便，令 $\alpha=C_0^s/T_0^s$，很明显 $C_{bi}^p=\alpha\,|b_i|$，$C_{ti}^p=\min(C_0^s,|t_i|)\geqslant\alpha\,|t_i|$。如果 $C_{hi}^p\geq\alpha\,|h_i|$，那么 $C_i^p=C_{hi}^p+C_{bi}^p+C_{ti}^p\geqslant\alpha\,(|h_i|+|b_i|+|t_i|)=\alpha\,|Bp_i|$。下面采用数学归纳法证明 $C_{hi}^p\geq\alpha\,|h_i|$。

(1)在系统启动前没有任务执行，所以 $C_{h1}^p=\min(|h_1|,C_0^s)\geqslant\alpha\,|h_1|$。

(2)当 $i>1$ 时，假设 $C_{hi-1}^p\geq\alpha\,|h_{i-1}|$，下面证明 $C_{hi}^p\geq\alpha\,|h_i|$。

在 h_{i-1} 内，处理机 P 和 P' 上到达的任务分别记为 J_{hi-1} 和 J_{hi-1}'。由于 $\sum_{J_i\in J_{hi-1}}C_i\geqslant|h_{i-1}|$，$C_i'=\alpha\,C_i$，所以 $\sum_{J_i'\in J_{hi-1}'}C_i'\geqslant\alpha\,|h_{i-1}|$。根据假设，$C_{hi-1}^p\geqslant\alpha\cdot|h_{i-1}|$，所以任务集 J_{hi-1}' 在 h_{i-1} 内消耗的时间 $C_{hi-1}^c\geqslant\alpha\,|h_{i-1}|$。在忙碌期 Bp_{i-1} 内，处理机 P' 上总的执行时间为 $\sum_{J_i'\in Bp_{i-1}}C_i'=\alpha\sum_{J_i\in Bp_{i-1}}C_i=\alpha|Bp_{i-1}|$，所以非周期任务在 t_{i-1} 内执行的时间 $C_{ti-1}^c\leq\alpha\,|t_{i-1}|$。如图 3(b)所示，如果 t_{i-1} 和 h_i 不在服务器的同一个周期内，那么 h_i 与 h_1 的相同，$C_{hi}^p\geq\alpha|\mathrm{head}_i|$。如果 t_{i-1} 和 h_i 在服务器的同一个周期内，如图 3(c)所示，那么 $C_{hi}^p=C_0^s-C_{ti-1}^c\geq C_0^s-\alpha|t_{i-1}|=\alpha T_0^s-\alpha|t_{i-1}|=\alpha(T_0^s-|t_{i-1}|)\geqslant\alpha|\mathrm{head}_i|$。

综上所述，$C_{hi}^p\geq\alpha|\mathrm{head}_i|$，所以 $C_i^p\geq\alpha|Bp_i|$。证毕

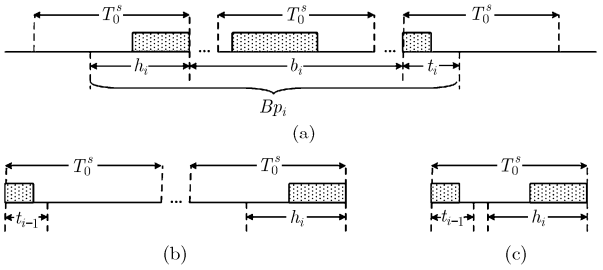

图 3 包含头部、主体和尾部的忙碌期

定理 4 对于非周期任务 $J=\{J_1,J_2,\cdots,J_k\}$，如果在任意时刻 t 满足不等式(4)，那么 J 可在服务器 S_i 上被 DS-EDF 算法调度。

$$U(t)\leq C_i^s/T_i^s \tag{4}$$

证明 不失一般性，设任务集 J 中的任务按绝对截止期限的不降顺序排列，即 $d_i\leqslant d_{i+1}(1\leqslant i<k)$。根据 DS-EDF 算法，只要有等待处理的任务，服务器 S_i 的执行时间不会流失，并且在 DS-EDF 算法下，低优先级任务不会影响高优先级任务的调度，所以只要在时间区间 $[A_{\min},d_i](i=1,\cdots,k,\ A_{\min}=\min\{A_j|j=1,\cdots,i\})$ 内，服务器 S_i 提供的执行时间 $C_i^p\geqslant$

$\sum_{j=1}^{i} C_j$，那么集合 J 可调度。

当 $i>0$ 时，$C_i^s=T_i^s$，如果(4)式成立，那么命题成立。下面证明当 $i=0$ 时，命题也成立。为方便，令 $\alpha = C_0^s / T_0^s$，设非周期任务 $J' = \{J_1', J_2', \cdots, J_k'\}$，且 $A_i' = A_i$，$C_i' = C^i / \alpha$，$D_i' = D_i$。那么 $U'(t) = \sum_{J_s' \in S(t)} C_i' / D_i' = \left(\sum_{J_i \in S(t)} C_i / D_i\right) / \alpha = U(t)/\alpha \le 1$，所以任务集 J' 可在单处理机上被 DS-EDF 算法调度。设 $\{J_1', J_2', \cdots, J_i'\}$ 形成的忙碌期为 $Bp_1, \cdots, Bp_n (n \ge 1)$，由于 $\{J_1', J_2', \cdots, J_i'\}$ 可调度，所以 $\{Bp_1, \cdots, Bp_n\} \subset [A_{\min}, d_i]$，并且 $\sum_{j=1}^{n} \left|Bp_j\right| = \sum_{j=1}^{i} C_j'$。根据定理 3，服务器 S_0 在 $[A_{\min}, d_i]$ 内提供的执行时间 $C_0^p \ge \sum_{j=1}^{n} \alpha \left|Bp_j\right| = \alpha \sum_{j=1}^{n} \left|Bp_j\right| = \alpha \sum_{j=1}^{i} C_j' = \sum_{j=1}^{i} \alpha C_j' = \sum_{j=1}^{i} C_j$。　　证毕

5　MFTS 算法

当某个处理机出现故障时，此处理机上执行的任务通过回卷到最近的检测点，重新调度到另外一个处理机上继续执行，使其在截止期限内顺利完成。每个检测点包含两部分：保存部分和恢复部分，其开销分别用 C_{sa} 和 C_{re} 表示。假设任务 τ_i 设置了 m_i 个检测点且检测点间隔 C_{in} 相等即 $C_{\text{in}}=C_i/(m_i+1)$，那么 τ_i 的最长回卷长度为 $C_i^R = C_{\text{sa}}+C_{\text{re}}+C_{\text{in}}$。正常执行时，$\tau_i$ 的执行时间为 $C_i^N = C_i + m_i C_{\text{sa}}$。

定理 5　对于周期任务 $\tau = \{\tau_1, \tau_2, \cdots, \tau_n\}$，如果每个任务 $\tau_i \in \tau$ 满足不等式(5)，那么 τ 可在 m 个处理机上被 DM 算法调度并允许 f 个处理机出现故障。

$$\Delta\beta^r + \sum_{i=1}^{k-1} \beta_i \le (m-f)(1-\lambda),\ (r = 0, \cdots, f) \quad (5)$$

其中 β_i 为式(1)定义并用 C_i^N 代替 C_i，$\lambda = C_k^r / D_k$，$\Delta\beta^r = r \max\{C_i^R / D_i\}$，$C_k^r = C_k^N + (f-r) C_k^R$。

证明　只需证明在 f 个处理机出现故障时，τ 可在 $m-f$ 个处理机上被 DM 算法调度。在任务 τ_k 执行的过程中，如果其他周期任务所在的处理机出现 r 次故障，根据式(1)，任务 τ_k 的负载最大增加量为 $\Delta\beta^r$，τ_k 所在的处理机最多出现 $f-r$ 故障，τ_k 的最长执行时间为 C_k^r。如果不等式(5)成立，那么 τ 可在 $m-f$ 个处理机上调度被 DM 算法调度。　　证毕

定理 6　对于非周期任务 $J=\{J_1, J_2, \cdots, J_k\}$，如果在任意时刻 t 满足不等式(6)，那么 J 可在服务器 S_i 上被 DS-EDF 算法调度并允许 f 个处理机出现故障。

$$\Delta U(t) + U(t) \leqslant C_i^s / T_i^s \quad (6)$$

其中 $\Delta U(t) = f\max(C_i^R / D_i)$，$U(t) = \sum_{J_i \in S(t)} C_i^N / D_i$。

证明　在没有处理机出现故障时，任务 J_i 的执行时间为 C_i^N，综合利用率 $U(t) = \sum_{J_i \in S(t)} C_i^N / D_i$；如果有处理机出现故障，至多有 f 个故障，综合利用率至多增加 $\Delta U(t) = f\max(C_i^R / D_i)$。根据定理 4，如果不等式(6)成立，那么任务集 J 可调度。　证毕

MFTS 算法包括离线和在线两部分。离线部分完成判定周期任务可调度性，设置服务器的个数和执行时间等功能；在线部分调度实时任务到每个处理机上执行，算法描述如下：

(1)离线部分

(a)根据定理 5，求出周期任务 $\tau = \{\tau_1, \tau_2, \cdots, \tau_n\}$ 所需要的最小处理机个数 g，如果 $g>m$，那么周期任务不可调度，算法结束。

(b) $\beta_0 = \min\left\{(g-f)(1-\lambda) - \sum_{i=1}^{k-1} \beta_i - \Delta\beta^r\right\}$，根据定理 1 求出服务器 S_0 的执行时间 C_0^s；设置其他 $h=m-g$ 个服务器的执行时间 $C_i^s = T_i^s$。

(2)在线部分。

(a)调度周期任务：采用 DM 调度算法调度周期任务和分配了非周期任务的服务器。

(b)判定非周期任务可调度性：当非周期任务 J_i 到达时，根据定理 6，采用 Fist-fit 策略把分配到能够调度它的服务器 S_i 上，如果不存在这样的 S_j，则拒绝 J_i。

(c)调度周期任务：采用 DS-EDF 算法调度每个服务器上的非周期任务。

6　实验结果及分析

为了验证算法的有效性，分两组实验对算法进行了验证：第 1 组实验验证 MFTS 算法对周期任务的容错调度；第 2 组实验是在选定一组周期任务和处理机前提下，验证 MFTS 算法对非周期任务的容错调度。

在第 1 组实验中，随机产生 10 组任务，每组包含 50 个周期任务，任务的周期 T_i 在[200, 300]内均匀分布，执行时间 C_i 在$(0, 0.3T_i]$内均匀分布，相对截止期限 $D_i = T_i$，求出所需的处理机个数 M 和处理机平均利用率 $U = \sum_{i=1}^{50} (C_i / T_i) / M$。HFTS[7]算法是一种基于主/从版本技术的混合实时任务容错调度算法，如果一个任务具有多个从版本，那么 HFTS 算法也能允许多个处理机故障。实验结果如图 4 所示，随着处理机故障的增加，HFTS 算法对处理机的利用率明显的下降，而 MFTS 算法下降的不明显。主要原因是每增加一个处理机故障，HFTS 算法就要为每个任务增加一个备份，系统的负载快速增加；而 MFTS 算法采用整体调度和任务回卷执行的方式调度任务，每增加一个处理机故障系统的负载仅增

加一个任务的回卷长度，对算法的影响不明显。

从图 4 可看出，在有两个处理机出现故障的情况下，HFTS 算法和 MHTS 算法对处理机的利用率基本相同，HFTS 算法使用了 20 个处理机而 MFTS 算法使用了 21 个处理机。在系统提供 21–25 个处理机的情况下，测试了对非周期任务的调度。非周期任务的到达时间为强度λ=0.1 的泊松流，与周期任务相对应，非周期任务的相对截止期限 D_i 在[200, 300]内均匀分布，执行时间 C_i在(0, $0.3D_i$]内均匀分布，任务接收率为系统接收的非周期任务个数与非周期任务总数的比值。实验结果如图 5 所示，在系统提供 21 个处理机时，HFTS 和 MFTS 对任务的接收率基本相同，因为 50 个周期任务在 HFTS 算法下至少需要 20 个处理机，而在 MFTS 算法下，至少需要 21 个处理机；随着系统提供的处理机数目的增多，HFTS 算法对任务的接收率基本上为 MFTS 算法的1/2，原因是在 HFTS 算法下，非周期任务的从版本全部为主动从版本，非周期任务的负载增加了 1 倍，而在 MFTS 算法下，非周期任务的负载只增加了任务的一个回卷长度。

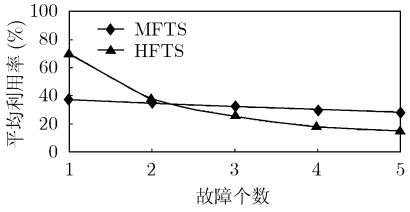

图 4 在不同故障数下的处理机利用率

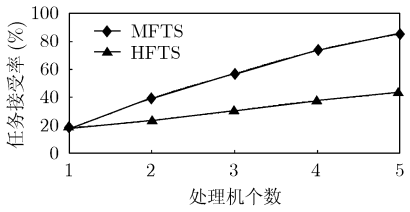

图 5 在不同处理机个数下的任务接收率

7 结论

本文提出了一种能够允许多个处理机出现故障的混合实时任务容错调度算法 MFTS，MFTS 算法采用多个延时服务器处理非周期任务，采用整体调度方式调度周期任务和延时服务器并通过任务回卷执行对处理机故障进行容错。实验结果表明对有多个处理机出现故障的情况，MFTS 算法对处理机的利用率和对非周期任务的接收率有明显的提高，但在只有一个处理机出现故障的情况下，MFTS 算法对处理机的利用率较低，需要改进对周期任务的判定方法。

参 考 文 献

[1] 罗威, 阳富民, 庞丽萍等. 基于延迟主动副版本的分布式实时容错调度算法 [J]. 计算机研究与发展, 2007, 44(3): 521–528.
Luo Wei, Yang Fu-min, and Pang Li-ping, *et al.*. A real-time fault-tolerant scheduling algorithm for distributed systems based on deferred active backup-copy [J]. *Journal of Computer Research and Development*, 2007, 44(3): 521–528.

[2] 罗威, 阳富民, 庞丽萍等. 异构分布式系统中实时周期任务的容错调度算法[J]. 计算机学报, 2007, 30(10): 1740–1749.
Luo Wei, Yang Fu-min, and Pang Li-ping, *et al.*. A real-time fault-tolerant scheduling algorithm of periodic tasks in heterogeneous distributed systems[J]. *Chinese Journal of Computers*, 2007, 30(10): 1740–1749.

[3] 吴俊. 基于双优先级队列的异构分布式控制系统容错调度算法[J]. 东南大学学报(自然科学版), 2008, 38(3): 407–412.
Wu Jun. Fault-tolerant scheduling algorithm for heterogeneous distributed control systems based on dual priorities queues [J]. *Journal of Southeast University* (*Natural Science Edition*), 2008, 38(3): 407–412.

[4] 潘雪增, 姚鑫骅, 傅建中等. 基于回卷恢复的数控系统实时容错调度策略[J]. 浙江大学学报(工学版), 2007, 41(12): 2011–2016.
Pan Xue-zeng, Yao Xin-zua, and Fu Jian-zhong, *et al.*. Tolerant real-time scheduling strategy for nc system based on rollback recovery [J]. *Journal of Zhejiang University* (*Engineering Science*), 2007, 41(12): 2011–2016.

[5] Strosnider J A, Lehoczky J P, and Sha L. The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments [J]. *IEEE Transactions on Computers*, 1995 , 44(1): 73–91.

[6] Lehoczky J P and Thuel S R. An optimal algorithm for scheduling soft-aperiodic tasks in fixed-priority preemptive systems [C]. Proceedings of the 13th IEEE Real-Time Systems Symposium, Phoenix, Arizona: 1992, 110–123.

[7] 阳春华, 桂卫华, 计莉. 基于多处理机的混合实时任务容错调度[J]. 计算机学报, 2003, 26(11): 1480–1486.
Yang Chun-hua, Gui Wei-hua, and Ji Li. A fault-tolerant scheduling algorithm of hybrid real-time tasks based on multiprocessors [J]. *Chinese Journal of Computers*, 2003, 26(11): 1480–1486.

[8] Baker T P. Multiprocessor EDF and deadline monotonic schedulability analysis [C]. Proceedings of the 24th IEEE International Real-Time Systems Symposium, 2003: 120–129.

[9] Baker T P. An analysis of deadline-monotonic scheduling on a multiprocessor. technical report TR-030301, Florida State University Department of Computer Science, Tallahassee, Florida (February 2003).

[10] Abdelzaher T F and Sharma T F. A utilization bound for aperiodic tasks and priority driven scheduling [J]. *IEEE Transactions on Computers*, 2004, 53(3): 334–350.

殷进勇： 男，1978 年生，博士生，研究方向为实时任务调度、可重构计算.

顾国昌： 男，1946 年生，教授，博士生导师，研究方向包括人工智能、嵌入式系统、可重构计算等.