基于 EDF 的分布式控制系统容错调度算法*

刘 怀[1+], 费树岷[2]

[1](南京师范大学 电气与电子工程学院,江苏 南京 210042)

[2](东南大学 自动化研究所,江苏 南京 210096)

A Fault-Tolerant Scheduling Algorithm Based on EDF for Distributed Control Systems

LIU Huai[1+], FEI Shu-Min[2]

[1](School of Electrical & Electronic Engineering, Nanjing Normal University, Nanjing 210042, China)

[2](Research Institute of Automation, Southeast University, Nanjing 210096, China)

+ Corresponding author: Phn: 86-25-5481296, E-mail: smfei@seu.edu.cn

http://210.29.130.227

Received 2002-12-14; Accepted 2003-04-09

Liu H, Fei SM. A fault-tolerant scheduling algorithm based on EDF for distributed control systems. *Journal of Software*, 2003,14(8):1371~1378.

http://www.jos.org.cn/1000-9825/14/1371.htm

Abstract: In recent result, the fault-tolerant scheduling algorithm almost requires that all task's periods are the same and equal to their deadlines, but in fact the periods are not the same in many cases. According to the characteristics of distributed control systems and the technique of primary/backup copies, based on EDF algorithm the novel fault-tolerant scheduling algorithm is proposed in this paper. The algorithm can deal with the different periods of all tasks. By using setting their deadlines the problem that execution times of primary and backup copies are not overlap can be controlled. The method for setting deadlines of primary and backup copies is given and the schedulability of task set is analyzed. The maximal utilization of task set and the minimal number of processor are investigated. The result of simulation shows that the algorithm is effective.

Key words: distributed control system; real-time task; fault-tolerant; primary copy/backup copy; EDF

摘 要: 现有的分布式实时系统的容错调度算法要求系统中所有任务的周期相同且等于其时限,而实际中任务的周期常常是互不相同的.根据控制系统中任务的特点,结合任务分配算法与处理器的调度算法,提出了基于基版本/副版本技术和 EDF 算法的容错调度算法.该算法不要求任务的周期都相同,并通过设置基版本/副版本任务时限控制它们的执行时间不重叠,给出了基版本/副版本任务时限的设置方法,并对任务集的可调度性进行了分析.当任务集可调度时,给出其最大利用率和最小处理器个数的约束条件.最后给出一个仿真实例,结果表明了算法的有效性.

关键词: 分布式控制系统;实时任务;容错;基版本/副版本;EDF

中图法分类号: TP316 **文献标识码**: A

第一作者简介: 刘怀(1971－),男,河北承德人,博士,副教授,主要研究领域为实时控制系统,综合自动化系统.

随着各种控制系统复杂性的提高,分布式控制系统已越来越多地应用于各种实际控制领域,如工业控制系统、武器防御控制系统、飞行控制系统、电站控制系统等.但是随着控制器数目的增加,系统控制器出现故障的可能性也相应增大.在控制系统中,每个任务都有严格的时间约束(时限),如果控制器的故障使某些任务不能在其时限内完成,就有可能造成很大的损失.为了避免控制器出现故障时造成严重后果,需要为分布式控制系统提供一定的容错能力,保证任务仍可以满足其时限,以提高整个系统的可靠性[1,2].

实时容错调度算法是一种通过软件解决分布式实时系统容错问题的有效方法.该方法的优点是不需要额外的硬件代价实现实时系统的可靠性.它是在分布式系统的容错调度算法的基础上发展起来的.分布式系统目前所采用的典型的容错技术主要是分布式投票技术[3]、反转恢复技术[4]和基/副版本技术[5],然而这些技术没有考虑任务的实时性问题.而现有的分布式实时系统的容错技术主要是在基/副版本技术上发展起来的[6~12],如BKCL[7,11],EBKCL[8,9],RTFTRC,RTFTNO[10]等等.这些方法都要求所有任务的周期相同并等于它们的时限,而实际情况下任务的周期往往是不相同的,这就限制了它们的应用范围,且该算法没有与处理器的局部调度算法结合起来.因此本文提出一种基于基/副版本技术与处理器调度算法相结合的容错算法,任务的周期可以互不相同,同时分析算法的可调度性.

1 系统模型及设计思想

典型的数字控制系统如图 1 所示[13].系统以一个固定的频率通过 A/D 转换器从物理过程获得测量数据,经控制计算得出控制量,再通过 D/A 转换器将控制信号送到被控对象.它需要完成数据采集、控制计算、状态更新、控制输出、报警等功能.其中数据采集、控制计算、状态更新、控制输出等是周期性进行的,它们的执行时间也是固定的,是控制系统所要处理的主要任务,占用处理机的时间较多.而报警等是随机产生的,是为了处理一些意外情况或紧急事件所需要执行的任务,占用处理器时间较少.系统的性能主要由周期性任务的执行情况来决定,所以本文着重研究周期性任务的容错调度问题.对于一个控制回路来说,它需要完成的各功能之间相互联系、相互影响,所以可将这种回路看作周期性任务,分布式控制系统中要在多个处理器上处理多个这种任务.

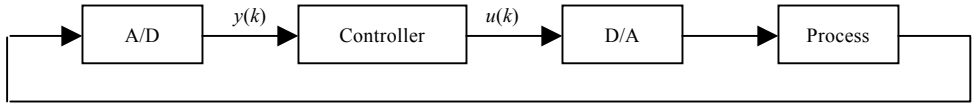

Fig.1 Overview of a digital control system

图 1 数字控制系统的结构

基于上述分析和容错算法的要求,下面给出分布式控制系统中的任务模型.设系统中实时周期性任务集表示为 $S=\{\tau_1,\tau_2,\ldots,\tau_n,\},n\geq 1$,任务 τ_i 由一个四元组表示,$\tau_i=(P,d,t^p,t^b)$,P,d,t^p 和 t^b 分别表示任务的周期、时限、基版本和副版本.$t^p=(C,Ts,\mathrm{Pr},D)$,$t^b=(C,Ts,\mathrm{Pr},D)$,其中 C,Ts,Pr 和 D 分别表示任务 τ_i 的基版本和副版本的执行时间、开始执行时间、所分配的处理机和时限.一个分布式控制系统被描述为一个处理机的集合:$\Omega=\{\mathrm{Pr}_1,\mathrm{Pr}_2,\ldots,\mathrm{Pr}_k\},k\geq 2$,$\mathrm{Pr}_i=(\Delta,len)$,其中 Δ 和 len 分别表示基版本和副版本被调度到该处理器上所有实时任务的总利用率.

为了使问题简化,本文作如下假设:

(1) 各处理器的性能、结构和处理器能力完全相同,即系统是一个同构系统,且同一个任务的基版本和副版本的执行代码相同,则实时任务的基版本和副版本的执行时间相同,即 $\tau_i.t^p.C=\tau_i.t^b.C=C_i$.

(2) 任务集中各任务相互独立.

(3) 任务的时限等于它的周期,即 $\tau_i.d=\tau_i.P$.

(4) 所有任务的基版本时限与周期的比相同.

2 调度算法的设计

为使系统具有容错能力,每一个实时任务都设有基版本和副版本,它们分配在不同的处理器上,系统首先运

行基版本,如果基版本运行正确,则其副版本不需要运行,而当基版本所在的处理器出现故障时,则该任务在其他处理器上的副版本投入运行.

调度算法分为两部分:任务分配算法和处理器的局部调度算法.为达到容错的目的,实时任务和调度算法应满足如下条件:

条件 1. 对于$\forall \tau_i \in S, \tau_i.t^p.C+\tau_i.t^b.C \le \tau_i.P$,这样才能保证$\tau_i$的基版本因处理器故障执行失败后,其副版本有充分的时间被调度执行,并在其时限前完成.

条件 2. 实时任务τ_i的基版本和副版本所在的处理器不同且它们的执行过程没有时间重叠,形式化描述为

$$(\tau_i.t^p.\mathrm{Pr} \ne \tau_i.t^b.\mathrm{Pr}) \text{ and } (\tau_i.t^p.Ts+\tau_i.t^p.D \le \tau_i.t^b.Ts), \forall \tau_i \in S. \tag{1}$$

条件 3. 由于处理器之间通信所需的时间与任务的执行时间相比非常短[6],所以忽略处理器之间消息的传递时间.

采用文献[6]提出的故障模型:某一时刻只有一个处理机出现故障,在第 2 个故障出现之前,前一个故障已被排除且那些因前一故障而执行失败的任务通过运行其副版本正确结束.另外,在对任务进行容错调度的同时,系统要能及时诊断并报告处理机故障.本文不对故障检测作深入地讨论,有兴趣者可见文献[6],只假设故障出现后,其他处理机能及时得到通知,由于处理机之间通信所需时间与任务的执行时间相比非常短,因此我们忽略处理机之间消息的传递时间.

处理器局部任务的调度算法:我们对单处理器采用抢占式 EDF 调度算法,即任务(包括基版本和副版本)实例的优先级采用 EDF 算法设置,根据任务实例的优先级高低确定要执行的任务.

任务的基版本和副版本时限的设置方法:主要是保证实时任务的基版本和副版本之间没有执行时间的重叠.我们对同一个任务的基版本和副版本实例采用不同任务时限和开始时间来控制.具体的控制方法是:实时任务τ_i的基版本的时限设置为$\tau_i.t^p.D \le \tau_i.P-\tau_i.t^b.C$,使其在$\tau_i.t^p.D$内完成,如果基版本因处理器$\tau_i.t^p.\mathrm{Pr}$出现故障而未能在其时限内完成,则通知副版本所在的处理器$\tau_i.t^b.\mathrm{Pr}$执行副版本,从而达到容错的目的.调度副版本时,假设当基版本通知其所在的处理器出现故障的时刻为副版本任务实例到达的时刻,并将其时限设置为$\tau_i.P-\tau_i.t^p.D$,这样可以使其在任务τ_i实例的时限$\tau_i.P$内完成.如果基版本$\tau_i.t^p$正确执行,则通知处理器$\tau_i.t^b.\mathrm{Pr}$取消对该基版本对应的副版本任务实例的调度.

基于此,我们给出任务集S在处理器集Ω中可调度的一个充分条件.

定理 1. 对给定的任务集$S=\{\tau_1,\tau_2,\ldots,\tau_n,\}(n\ge 1)$和处理器集$\Omega=\{\mathrm{Pr}_1,\mathrm{Pr}_2,\ldots,\mathrm{Pr}_k\}(k\ge 2)$,如果各参数满足

$$\sum_{i=1}^{n}\frac{C_i}{\tau_i.P} \le 0.25k\ , \tag{2}$$

则任务集S在处理器集Ω中是可调度的.当式(2)取等号时,处理器的利用率最大,所有任务的基版本的时限为

$$\tau_i.t^p.D=\frac{1}{2}\tau_i.P\ (\forall \tau_i \in S\ ).$$

证明:对于$\forall \tau_i \in S$,可令$\tau_i.t^p.D=a\,\tau_i.P(0<a<1)$.对于处理器$\mathrm{Pr}_i(1\le i\le k)$,假设在某一时刻$t$,其他处理器出现故障,则处理器$\mathrm{Pr}_i$不仅要执行分配给它的基版本任务,而且还要执行分配给它的基版本处理器是出错处理器的副版本任务.根据调度算法,基版本和副版本的时限设置是小于周期的.最坏的情况是,基版本在接近其时限时通知副版本执行,此时副版本的时限设置应为$\tau_j.P-\tau_j.t^p.D$,这样才能保证副版本在任务时限内完成.如果分配到处理器Pr_i所有任务(包括基版本和副版本)的实例都能在其时限内完成,则这些任务是可调度的,根据 Baker[14]的分析可以得到处理器Pr_i中任务可调度的条件为

$$\sum_{\tau_q.t^p.\mathrm{Pr}=\mathrm{Pr}_i}\frac{C_q}{\tau_q.t^p.D}+\sum_{\tau_j.t^b.\mathrm{Pr}=\mathrm{Pr}_i}\frac{C_j}{\tau_j.t^b.D}=\sum_{\tau_q.t^p.\mathrm{Pr}=\mathrm{Pr}_i}\frac{C_q}{a\tau_q.P}+\sum_{\tau_j.t^b.\mathrm{Pr}=\mathrm{Pr}_i}\frac{C_j}{(1-a)\tau_j.P}\le 1\ , \tag{3}$$

将所有处理器的可调度条件求左右两边同时相加,有

$$\sum_{i=1}^{k}\left(\sum_{\tau_q.t^p.\mathrm{Pr}=\mathrm{Pr}_i}\frac{C_q}{a\tau_q.P}+\sum_{\tau_j.t^b.\mathrm{Pr}=\mathrm{Pr}_i}\frac{C_j}{(1-a)\tau_j.P}\right)\le k\ , \tag{4}$$

显然存在

$$\left.\begin{aligned}\sum_{i=1}^{k}\sum_{\tau_q.t^p.\mathrm{Pr}=\mathrm{Pr}_i}\frac{C_q}{a\tau_q.P}=\sum_{q=1}^{n}\frac{C_q}{a\tau_q.P},\\ \sum_{i=1}^{k}\sum_{\tau_j.t^b.\mathrm{Pr}=\mathrm{Pr}_i}\frac{C_q}{(1-a)\tau_q.P}=\sum_{q=1}^{n}\frac{C_q}{(1-a)\tau_q.P},\end{aligned}\right\} \tag{5}$$

将式(5)带入式(4),可得

$$\sum_{q=1}^{n}\frac{C_q}{a\tau_q.P}+\sum_{j=1}^{n}\frac{C_j}{(1-a)\tau_j.P}\le k,$$

由于 $0<a<1$,在不等式两边同时乘以 $a(1-a)$,可得

$$(1-a)\sum_{q=1}^{n}\frac{C_q}{\tau_q.P}+a\sum_{j=1}^{n}\frac{C_j}{\tau_j.P}\le k(1-a)a, \tag{6}$$

简化可以得到

$$\sum_{q=1}^{n}\frac{C_q}{\tau_q.P}\le k\left[\frac{1}{4}-\left(\frac{1}{2}-a\right)^2\right], \tag{7}$$

上式为任务集 S 在处理器集Ω的条件,左边为任务集 S 的利用率,显然当 $a=\frac{1}{2}$ 时,其利用率取最大值为 $\frac{1}{4}k$. □

定理 1 给出了任务集可调度的判断条件和任务基版本时限的设置方法,但是此条件过于保守,我们的任务模型并不要求处理器上副版本都是可调度的,只要该处理器上基版本和基版本处理器出现故障的副版本可调度就可以了,基于此我们给出处理器可调度的条件.

定理 2. 设分配到处理器 Pr_i 上的基版本的实时任务为 $S_p=\{\tau_1,\tau_2,\ldots,\tau_m,\}$,而分配到其上的副版本的实时任务为 $S_b=\{\tau_1',\tau_2',\ldots,\tau_l',\}$,则任务集的可调度条件为

$$\sum_{j=1}^{m}\frac{C_j}{\tau_j.P}+U_m\le 0.5, \tag{8}$$

其中 $U_m=\max_{\mathrm{Pr}_q\in\Omega,q\ne i}\left\{\sum_{\tau_j'.t^p.\mathrm{Pr}=\mathrm{Pr}_{i_q}}\frac{C_j}{\tau_j'.P}\right\}$.

证明:由前面的分析可知,系统中在某一时刻只有一个处理器出现故障.对于处理器 Pr_i,假设在某一时刻 t,处理器$\forall\mathrm{Pr}_q(\mathrm{Pr}_q\in\Omega, j\ne i)$出现故障,则处理器 Pr_i 不仅要执行分配给其的基版本任务,而且还要执行分配给其的基版本处理器是 Pr_q 的副版本任务.根据调度算法和故障模型,基版本和副版本的时限设置小于周期.根据 Baker[14] 的分析可以得到处理器 Pr_i 中任务可调度的条件为

$$\sum_{j=1}^{m}\frac{C_j}{0.5\tau_j.P}+\sum_{\tau_j'.t^p.\mathrm{Pr}=\mathrm{Pr}_q}\frac{C_j}{0.5\tau_j'.P}\le 1. \tag{9}$$

由于 Pr_q 是任意的,显然当 $\sum_{\tau_j'.t^p.\mathrm{Pr}=\mathrm{Pr}_q}\frac{C_j}{0.5\tau_j'.P}$ 对于所有处理器(除 Pr_i 外)是最大值时,处理器 Pr_i 处于最坏的情况,如果此时满足式(9),即任务是可调度的,则处理器 Pr_i 上的任务在任意情况下都可调度.令

$$U_m=\max_{\mathrm{Pr}_q\in\Omega,q\ne i}\left\{\sum_{\tau_j'.t^p.\mathrm{Pr}=\mathrm{Pr}_q}\frac{C_j}{0.5\tau_j'.P}\right\}, \tag{10}$$

再由式(9)可得出结论. □

推论 1. 在前述故障模型和调度算法中,任务集 S 可调度时的最大利用率可大于 $0.25k$,但必小于 $0.5k$.

证明:定理 2 给出的是一个处理的调度条件,令 $i=1,2,\ldots,k$,并将不等式两端相加可得

$$\sum_{i=1}^{k}\sum_{\tau'_j.t^p.\mathrm{Pr}=\mathrm{Pr}_i}\frac{C_j}{0.5\tau_j.P}+\sum_{i=1}^{k}\max_{\mathrm{Pr}_q\in\Omega,q\neq i}\left\{\sum_{\substack{\tau'_j.t^p.\mathrm{Pr}=\mathrm{Pr}_q\\ \tau_j.t^b.\mathrm{Pr}=\mathrm{Pr}_i}}\frac{C_j}{0.5\tau'_j.P}\right\}\le k\ ,$$

当取等号时,任务集也是可调度的,根据定理 1 的证明有

$$\sum_{j=1}^{n}\frac{C_j}{\tau_j.P}+\sum_{i=1}^{k}\max_{\mathrm{Pr}_q\in\Omega,q\neq i}\left\{\sum_{\substack{\tau'_j.t^p.\mathrm{Pr}=\mathrm{Pr}_q\\ \tau_j.t^b.\mathrm{Pr}=\mathrm{Pr}_i}}\frac{C_j}{\tau'_j.P}\right\}=\frac{k}{2},\tag{11}$$

显然

$$0\le\sum_{i=1}^{k}\max_{\mathrm{Pr}_q\in\Omega,q\neq i}\left\{\sum_{\substack{\tau'_j.t^p.\mathrm{Pr}=\mathrm{Pr}_q\\ \tau_j.t^b.\mathrm{Pr}=\mathrm{Pr}_i}}\frac{C_j}{0.5\tau'_j.P}\right\}<\sum_{j=1}^{n}\frac{C_j}{0.5\tau_j.P},\tag{12}$$

则由式(11)和式(12)可得

$$\sum_{j=1}^{n}\frac{C_j}{\tau_j.P}<\frac{k}{2},\tag{13}$$

$$2\sum_{j=1}^{n}\frac{C_j}{\tau_j.P}>\frac{k}{2},\tag{14}$$

由式(14)可得

$$\sum_{j=1}^{n}\frac{C_j}{\tau_j.P}>\frac{k}{4},\tag{15}$$

由式(14)和式(15)可得结论. □

推论 2. 在前述故障模型和调度算法中,任务集可调度时所需的最小处理器个数为

$$k_{\min}\ge\left\lceil 2\sum_{i=1}^{n}\frac{C_i}{\tau_i.P}\right\rceil+1,\tag{16}$$

式中$\lceil x\rceil$表示$\ge x$ 的最小整数.

证明:用反证法.假设 $\exists m, m<k_{\min}, m\in N$,当处理器数目 $k=m$ 时,任务集在处理器集Ω上仍是可调度的.根据调度算法,假设任务的基版本按利用率平均分配到各个处理器上;对于一个处理器 Pr_i 上的基版本任务,它们的副版本也是按利用率平均分配到其他的处理器 $\mathrm{Pr}_j(j\neq i)$上的,则处理器 Pr_j 上的任务(包括基版本和副版本)的利用率为

$$\begin{aligned}\sum_{j=1}^{m}\frac{C_j}{\tau_j.P}+U_m&=\frac{\sum_{i=1}^{n}\frac{C_i}{\tau_i.P}}{m}\left(1+\frac{1}{m-1}\right)\ge\frac{\sum_{i=1}^{n}\frac{C_i}{\tau_i.P}}{k_{\min}-1}\left(1+\frac{1}{k_{\min}-2}\right)\\&=\frac{\sum_{i=1}^{n}\frac{C_i}{\tau_i.P}}{\left\lceil 2\sum_{i=1}^{n}\frac{C_i}{\tau_i.P}\right\rceil}\left(1+\frac{1}{\left\lceil 2\sum_{i=1}^{n}\frac{C_i}{\tau_i.P}\right\rceil-1}\right)=\frac{\sum_{i=1}^{n}\frac{C_i}{\tau_i.P}}{\left\lceil 2\sum_{i=1}^{n}\frac{C_i}{\tau_i.P}\right\rceil-1}>\frac{\sum_{i=1}^{n}\frac{C_i}{\tau_i.P}}{2\sum_{i=1}^{n}\frac{C_i}{\tau_i.P}}=\frac{1}{2}.\end{aligned}$$

这就与定理 2 相矛盾,所以假设不成立,原命题正确. □

任务分配算法,就是将任务按照一定的规则分配到各处理器,这是一个 NP 难题.本文给出一个启发式分配方法.本算法在分配任务时,采用首次满足(first-fit)方法,它不仅考虑各处理器负载尽可能平衡,而且使实时任务的基版本平均分配到各处理器上,算法的描述如下:

任务分配算法.

输入:S,k.

输出:Sch_Success,处理器集合Ω.

(1) 初始化任务集 S,输入各任务的周期、执行时间;处理器集合Ω,初始化各处理器的参数 $\mathrm{Pr}_i.\Delta$=0,$\mathrm{Pr}_i.len$=0.

(2) 如果 $k<\left\lceil 2\sum_{i=1}^{n}\frac{C_i}{\tau_i.P}\right\rceil+1$ 或 $\exists\tau_i,\ \tau_i.t_i^p.C_i^p+\tau_i.t_i^b.C_i^b=2C_i>\tau_i.P$,则 Sch_Success=失败,转到步骤(7),否则转到步骤(3).

(3) 将任务 $\tau_i(1\le i\le n)$按如下步骤分配到处理器集Ω中的处理器上.

(4) 对于任务 τ_i 基版本 $\tau_i.t^p$,如果存在处理器 Pr_j 满足 $\mathrm{Pr}_j.\Delta=\min_{\mathrm{Pr}_q\in\Omega}\{\mathrm{Pr}_q.\Delta\}$,并令 $\mathrm{Pr}_j.\Delta=\mathrm{Pr}_j.\Delta+\frac{C_i}{\tau_i.P}$,如果该处理器 $\mathrm{Pr}_j.\Delta+\mathrm{Pr}_j.len>0.5$,则 Sch_Success=失败,转到步骤(7),否则将任务 τ_i 基版本 $\tau_i.t^p$ 分配到处理器 Pr_j 并转到步骤(5).

(5) 对于任务 τ_i 副版本 $\tau_i.t^b$,如果存在处理器 Pr_l ($l\ne j$)满足

$$\sum_{\substack{\tau_r.t^b.\mathrm{Pr}=\mathrm{Pr}_l\\ \tau_r.t^p.\mathrm{Pr}=\mathrm{Pr}_j}}\frac{C_r}{\tau_r.P}=\min_{\mathrm{Pr}_s\in\Omega,s\ne j}\left\{\sum_{\substack{\tau_r.t^b.\mathrm{Pr}=\mathrm{Pr}_s\\ \tau_r.t^p.\mathrm{Pr}=\mathrm{Pr}_j}}\frac{C_r}{\tau_r.P}\right\},$$

并令

$$\mathrm{Pr}_j.len=\max_{\mathrm{Pr}_q\in\Omega,q\ne i}\left\{\sum_{\substack{\tau'_j.t^p.\mathrm{Pr}=\mathrm{Pr}_q\\ \tau'_j.t^b.\mathrm{Pr}=\mathrm{Pr}_i}}\frac{C_j}{\tau'_j.P}\right\},$$

若该处理器 $\mathrm{Pr}_j.\Delta+\mathrm{Pr}_j.len>0.5$,则 Sch_Success=失败,转到步骤(7),否则将任务 τ_i 副版本 $\tau_i.t^b$ 分配到处理器 Pr_l,转到步骤(6).

(6) 所有任务分配完成,Sch_Success=成功.

(7) 算法结束.

说明:此任务分配算法是静态算法,它在系统运行前将任务的基版本和副版本分配到各个处理器上,在系统运行时不再变化.但是各处理器调度其上的基版本/副版本任务实例是根据它们的优先级来调度的,它们的优先级是根据 EDF 算法动态地加以设置.

最小处理器数求解算法.

(1) 令处理器数为 $k=2$;

(2) 根据任务分配算法,如果输出结果中 Sch_Success=成功,则转至步骤(4);

(3) 令 $k=k+1$,重复步骤(2);

(4) 最小处理器数为 k;

(5) 算法结束.

3 算法模拟

我们利用计算机对前述任务分配算法和处理器调度算法进行了模拟,产生了 100 个实时任务,它们的周期是在区间[200,400](单位:ms)内随机产生的,它们的执行时间是在区间 $(0,0.5P_i)$ 内随机产生的.分两个步骤模拟.

3.1 任务集最小处理器数模拟

我们根据最小处理器数求解算法编写了程序,做了 3 组仿真实验,结果如下:

第 1 组:任务集的总的利用率为 22.145 3,根据推论 2,所需要的最小处理器数为 46 个,计算仿真得出的最小处理器数为 50 个.

第 2 组:任务集的总的利用率为 23.728 8,根据推论 2,所需要的最小处理器数为 49 个,计算仿真得出的最小

处理器数为 50 个.

第 3 组:任务集的总的利用率为 24.496 6,根据推论 2,所需要的最小处理器数为 50 个,计算仿真得出的最小处理器数为 50 个.

这说明本文的结论是正确的.

3.2 处理器出现故障时任务执行情况

本文选择上述第 3 组实验所得的任务集和处理器数在计算机上模拟任务执行,假设每隔一段时间 Δt=200ms 有一台处理器出现故障,故障持续时间服从均匀分布,参数为(50,100)(单位:ms),在模拟时间为 10 000s 时考察任务的执行情况,从结果可以看出,在处理器出现故障时,任务通过执行其副版本使其响应时间都在其时限内完成,没有实例丢失时限.

4 结 语

本文讨论了分布式实时控制系统的多任务容错调度算法,已有的分布式实时系统的容错调度算法要求所有实时任务的周期完全相同,而且没有结合单处理器调度算法.本文基于基版本/副版本技术,结合单处理器常用的 EDF 调度算法,给出了实时分布式控制系统的容错调度算法.该算法将任务基版本的时限作为副版本任务实例的到达时间,通过设置两个版本的时限来控制它们之间没有执行时间的重叠.算法分析了系统的可调度性,给出了判断任务集可调度的充分条件和处理器最小个数条件.仿真结果表明,调度算法提高了分布式控制系统的容错能力,说明算法是有效的.

References:

[1] Kieckhafer RM, Walter CJ, Finn AM, Thambidurai PM. The MAFT architecture for distributed fault tolerance. IEEE Transactions on Computers, 1988,37(4):398~405.

[2] Factor M, Gelernter DH, Kolb CE, Miller PL, Sittig DF. Real-Time data fusion in the intensive care unit. IEEE Computer, 1991,24(11):45~54.

[3] Xu LH, Bruck J. Deterministic voting in distributed systems using error-correcting codes. IEEE Transactions on Parallel and Distributed Systems, 1998,9(8):813~824.

[4] Lin TH, Shin KG. Damage assessment for optimal rollback recovery. IEEE Transactions on Computers, 1998,47(5):603~613.

[5] Davoli R, Giachini LA, Babagiu Ö, Amoroso A, Alvisi L. Parallel computing in networks of workstations with parallex. IEEE Transactions on Parallel and Distributed Systems, 1996,7(4):371~384.

[6] Zhang YJ, Zhang Y, Peng YX, Chen FJ, A multiprocessor-based fault-tolerant real-time task scheduling algorithm. Journal of Computer Research and Development, 2000,37(4):425~429 (in Chinese with English abstract).

[7] Qin X, Han ZF, Li SL, Pang LP. Efficient scheduling algorithm with fault-tolerance for real-time tasks in multi-processor systems. Journal Huazhong University of Science & Technology, 1999,27(7):14~16 (in Chinese with English abstract).

[8] Han ZF, Qin X, Pang LP, Li SL. Design of fault-tolerant scheduling algorithm for real-time tasks in distributed systems. Journal of Huazhong University of Science & Technology, 1999,27(7):12~14 (in Chinese with English abstract).

[9] Zhang KL, Qin X, Han ZF, Pang LP. Study of the model for fault-tolerant scheduling algorithm in heterogeneous distributed real-time systems. Journal of Huazhong University of Science & Technology, 2000,28(8):17~18 (in Chinese with English abstract).

[10] Qin X, Han ZF, Pang LP. Real-Time scheduling with fault-tolerance in heterogeneous distributed systems. Chinese Journal of Computers, 2002,25(1):49~56 (in Chinese with English abstract).

[11] Qin X, Han ZF, Pang LP, Li SL. Design and performance analysis of a hybrid real-time scheduling algorithm with fault-tolerance. Journal of Software, 2000,11(5):668~693 (in Chinese with English abstract).

[12] Kieckhafer RM. Fault-Tolerant real-time task scheduling in the MAFT distributed system. In: Proceedings of the 22nd Annual Hawaii International Conference on System Sciences. IEEE CS Press, 1989,I:143~151.

[13] Cervin A. Improved scheduling of control tasks. In: Proceedings of the 11th Euromicro Conference on Real-Time Systems. IEEE Computer Society Press, 1999. 4~10.

[14] Tovar E, Vasques F. Non pre-emptive scheduling of messages on SMTV token-passing networks. In: Proceedings of the 12th Euromicro Conference on Real-Time Systems (RTS 2000). IEEE CS Press, 2000. 209~218.

附中文参考文献:

[6] 张拥军,张怡,彭宇行,陈福接.一种基于多处理机的容错实时任务调度算法.计算机研究与发展,2000,37(4):425~429.

[7] 秦啸,韩宗芬,李胜利,庞丽萍.多处理机系统的高效实时容错调度算法.华中理工大学学报,1999,27(7):14~16.

[8] 韩宗芬,秦啸,庞丽萍,李胜利.分布式系统的实时容错任务调度算法设计.华中理工大学学报,1999,27(7):12~14.

[9] 张坤龙,秦啸,韩宗芬,庞丽萍.异构分布式实时系统中容错调度模型的研究.华中理工大学学报,2000,28(8):17~18.

[10] 秦啸,韩宗芬,庞丽萍.基于异构分布式系统的实时容错调度算法.计算机学报,2002,25(1):49~56.

[11] 秦啸,韩宗芬,庞丽萍,李胜利.混合型实时容错调度算法的设计和性能分析.软件学报,2000,11(5):668~693.

Call for Papers

December 7~10 , 2003, Shanghai, China

http://www.cs.sjtu.edu.cn/gcc2003/index.htm

The Second International Workshop on Grid and Cooperative Computing (GCC2003) is to be held from December 7~10, 2003 in Shanghai, China. GCC2003 is the follow-up of the highly successful GCC2002 Workshop held in SanYa, HaiNan. It will serve as a forum to present current and future work as well as to exchange research ideas by researchers, developers, practitioners, and users in Grid computing, Web services and cooperative computing.

TOPICS OF INTEREST

The main topics of interest include, but not limited to:

Grid Computing and Grid Security
Grid Information Services
Grid Middleware and Toolkits
Grid Monitoring, Management and Organization Tools
Grid Applications
Information Grid and Knowledge Grid
Advance Resource Reservation and Scheduling
Performance Evaluation and Modeling
Web Services and Web Security
P2P Computing
Cooperative Middleware
Software Integration Technologies
Software Engineering Support for Cooperative Computing
Computer-Supported Cooperative Work

SUBMISSION

GCC 2003 invites authors to submit original and unpublished work. Papers should not exceed 10 pages of text using 10 point size type on 8.5×11 inch paper. Papers will be refereed and accepted on the basis of their scientific merit and relevance to the conference topics. Submission of full technical papers is called. Papers should be written in English and submitted in PDF format. All papers are to be electronically submitted to (Please also include the address, telephone, FAX, and email of the primary contact person in a separate sheet): gcc2003@cs.sjtu.edu.cn

IMPORTANT DATES

Electronic submissions due:	September 15, 2003
Acceptance notification due:	October 15, 2003
Camera-ready due:	November 5, 2003
GCC2003 Workshop:	December 7~10, 2003