分布式实时系统的容错调度算法

刘 怀

(南京师范大学 电气与电子工程学院, 江苏 南京 210042)

[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. 将任务分配算法与单处理器的调度算法相结合, 提出基于基版本/副版本技术和非抢占式 EDF 算法的容错调度算法, 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, 给出了任务集在给定处理器集上可调度性的判定方法.

[关键词] 分布式实时系统, 容错, 基版本/副版本, 非抢占 EDF

[中图分类号] TP316.2, [文献标识码] A, [文章编号] 1672-1292(2004)02-0022-04

0 引言

随着各种应用复杂性的提高, 分布式实时系统开始越来越广泛地应用于各种领域, 如工业控制系统、武器防御控制系统、飞行控制系统、电站控制系统及实时多媒体系统等. 但是随着分布式系统中节点数目的增加, 系统出现故障的可能性增大. 在实时系统中, 每个任务都是有严格的时间约束(时限)的, 如果控制器的故障使某些任务不能在其时限内完成, 就可能会造成很大的损失. 为了避免控制器出现故障时造成严重后果, 需要在分布式实时系统中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].

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提高系统的可靠性[2]. 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[2~6], 如 RTFTRC、RTFTNO[2]、BKCL[3,4]、EBKCL[5,6]等等. 这些算法都要求所有任务的周期都相同且等于任务时限, 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. 本文给出一种基于基/副版本技术与非抢占 EDF 算法的容错算法, 任务的周期可以互不相同, 同时分析算法的可调度性.

1 任务模型及系统模型

为使系统具有容错的能力, 每一个实时任务都设有基版本和副版本, 并分配在不同的处理器上, 系统先运行基版本, 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.

本文主要研究容错调度算法, 不考虑系统中错误的诊断, 采用文献[7]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 个故障出现前, 前一个故障已经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, 系统要能及时诊断出并报告处理机故障. 本文不对故障检测作深入地讨论, 只假设故障出现后, 其它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, 因此忽略处理机之间消息的传递时间[7].

定义 **1** 系统中实时任务集表示为 $S = \{\tau_1, \tau_2, \cdots, \tau_n\}$, $n \geqslant 1$, 任务 $\tau_i \in S$ 由一个四元组表示, $\tau_i = (P, d, t^p, t^b)$, P 和 d 分别表示任务 τ_i 的周期、时限; t^p 和 t^b 分别表示任务 τ_i 的基版本和副版本, t^p 和 t^b 可由四元组描述, 即 $t^p = t^b = (C, Ts, Pr, D)$, 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.

定义 **2** 一个分布式实时系统被描述为一个处理机的集合: $\Omega = \{Pr_1, Pr_2, \cdots, Pr_k\}$, $k \geqslant 2$, $Pr_i = (\Delta, len, ta)$, 其中 Δ 和 len 分别表示被调度到该处理器上的基版本和副版本任务的总利用率, ta 表示被分配到处理器 Pr_i 上的所有任务(包括基版本和副版本)中最大执行时间和最小周期的比值.

收稿日期: 2003-09-19.
基金项目: 南京师范大学科研基金资助项目(2003KZXXGQ2B88).
作者简介: 刘怀(1971-), 博士, 副教授, 主要从事实时控制系统、综合自动化、智能控制的教学与研究. E-mail: lhhjf@sohu.com

为了使问题简化, 本文做如下假设:

(1) 各处理器的性能、结构和处理器能力完全相同, 且同一个任务的基版本和副版本执行代码相同, 即一个任务的基版本和副版本执行时间相同, 即 $\tau_i \cdot t^p \cdot C = \tau_i \cdot t^b \cdot C = C_i$.

(2) 任务集中, 各任务相互独立.

(3) 任务的时限等于其周期, 即 $\tau_i \cdot d = \tau_i \cdot P$.

(4) 所有任务的基版本时限与周期的比相同, 令 $\tau_i \cdot t^p \cdot D = a\tau_i \cdot P (0 < a < 1)$.

(5) 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[7], 所以忽略处理器之间消息的传递时间.

(6) 假设对于 $\forall \tau_i \in S$, $\tau_i \cdot t^p \cdot C + \tau_i \cdot t^b \cdot C \leqslant \tau_i \cdot P$, 这样才能保证 τ_i 的基版本因处理器故障执行失败后, 其副版本才有充分的时间被调度执行, 并在其时限前完成.

(7) 根据容错要求, 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执行过程没有时间重叠, 其形式化描述为:

$$(\tau_i \cdot t^p \cdot Pr \neq \tau_i \cdot t^b \cdot Pr) \text{ and } (\tau_i \cdot t^p \cdot Ts + \tau_i \cdot t^p \cdot D \leqslant \tau_i \cdot t^b \cdot Ts), \quad \forall \tau_i \in S \tag{1}$$

2　调度算法的设计

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.

2.1　处理器局部任务的调度算法

分布式系统中的每一个处理器都采用非抢占式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, 但高优先级任务不能抢占正在执行的低优先级任务.

任务的基版本和副版本时限的设置方法如下: 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. 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 $\tau_i \cdot t^p \cdot D \leqslant \tau_i \cdot P - \tau_i \cdot t^b \cdot C$, 使其在 $\tau_i \cdot t^p \cdot D$ 前完成. 如果基版本因处理器 $\tau_i \cdot t^p \cdot Pr$ 出现故障而未能在其时限前完成, 则通知副版本所在的处理器 $\tau_i \cdot t^b \cdot Pr$ 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 $\tau_i \cdot P - \tau_i \cdot t^p \cdot D$, 这样可以使其在任务 τ_i 实例的时限 $\tau_i \cdot P$ 前完成. 如果基版本 $\tau_i \cdot t^p$ 正确执行, 则通知处理器 $\tau_i \cdot t^b \cdot Pr$ 取消对该基版本对应的副版本任务实例的调度.

基于此给出任务集 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 $S = \{\tau_1, \tau_2, \cdots, \tau_n\} (n \geqslant 1)$ 和处理器集 $\Omega = \{Pr_1, Pr_2, \cdots, Pr_k\} (k \geqslant 2)$, 如果各参数满足

$$\sum_{i=1}^{n} \frac{C_i}{\tau_i \cdot P} \leqslant 0.25k\left(1 - \frac{2C_{\max}}{P_{\min}}\right) \tag{2}$$

则, 任务集 S 在处理器集 Ω 中是可调度的. 当(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 $\tau_i \cdot t^p \cdot D = \frac{1}{2}\tau_i \cdot P (\forall \tau_i \in S)$. 式中 $C_{\max} = \max\{C_1, C_2, \cdots, C_n\}$, $P_{\min} = \min\{\tau_1 \cdot P, \tau_2 \cdot P, \cdots, \tau_n \cdot P\}$.

证明　对于 $\forall \tau_i \in S$, 根据假设(4) 有 $\tau_i \cdot t^b \cdot D = (1 - a)\tau_i \cdot P$. 对于处理器 $Pr_i (1 \leqslant i \leqslant k)$, 假设在某一时刻 t, 其它处理器出现故障, 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. 由 Baker[9] 的分析可以得到处理器 Pr_i 中任务可调度的条件为:

$$\sum_{\tau_q \cdot t^p \cdot Pr = Pr_i} \frac{C_q}{\tau_q \cdot t^p \cdot D} + \sum_{\tau_j \cdot t^b \cdot Pr = Pr_i} \frac{C_j}{\tau_j \cdot t^b \cdot D} + \max_{\tau_l \cdot Pr = Pr_i}\left[\frac{B_l}{\tau_l \cdot D}\right] \leqslant 1 \tag{3}$$

式中, $B_l = \max_{\substack{\tau_m \cdot Pr = Pr_i \\ m \neq l}} \{C_m\}$, $\tau_l \cdot Pr$ 表示任务 τ_l 的基版本或副版本所在的处理器.

令 $C'_{\max, i} = \max_{\tau_m \cdot Pr = Pr_i} \{C_m\}$, $D_i = \min_{\tau_m \cdot Pr = Pr_i} \{\tau_m \cdot D\}$, 若任务满足下式:

$$\sum_{\tau_q \cdot t^p \cdot Pr = Pr_i} \frac{C_q}{\tau_q \cdot t^p \cdot D} + \sum_{\tau_j \cdot t^b \cdot Pr = Pr_i} \frac{C_j}{\tau_j \cdot t^b \cdot D} + \frac{C'_{\max, i}}{D_i} \leqslant 1 \tag{4}$$

必满足(3) 式, 即任务是容错可调度的. 将所有处理器的可调度条件((4) 式) 左右两边同时相加, 有:

$$\sum_{i=1}^{k}\left(\sum_{\tau_q \cdot t^p \cdot Pr = Pr_i} \frac{C_q}{a\tau_q \cdot P} + \sum_{\tau_j \cdot t^b \cdot Pr = Pr_i} \frac{C_j}{(1 - a)\tau_j \cdot P} + \frac{C'_{\max, i}}{D_i}\right) \leqslant k \tag{5}$$

由于 $C_{\max} \geqslant C'_{\max, i}$, 令 $D_{\min} = \min\{D_1, D_2, \cdots, D_k\}$, 如果参数满足下式, 必满足(5)

$$\sum_{i=1}^{k}\left(\sum_{\tau_q\cdot t^p\cdot Pr=Pr_i}\frac{C_q}{a\tau_q\cdot P}+\sum_{\tau_j\cdot t^b\cdot Pr=Pr_i}\frac{C_j}{(1-a)\tau_j\cdot P}\right)+k\frac{C_{\max}}{D_{\min}}\leqslant k \qquad (6)$$

显然存在

$$\sum_{i=1}^{k}\sum_{\tau_q\cdot t^p\cdot Pr=Pr_i}\frac{C_q}{a\tau_q\cdot P}=\sum_{q=1}^{n}\frac{C_q}{a\tau_q\cdot P}$$

$$\sum_{i=1}^{k}\sum_{\tau_j\cdot t^p\cdot Pr=Pr_i}\frac{C_j}{(1-a)\tau_j\cdot P}=\sum_{j=1}^{n}\frac{C_j}{(1-a)\tau_j\cdot P} \qquad (7)$$

将(7)带入(6)中可得

$$\sum_{q=1}^{n}\frac{C_q}{a\tau_q\cdot P}+\sum_{j=1}^{n}\frac{C_j}{(1-a)\tau_j\cdot P}+k\frac{C_{\max}}{D_{\min}}\leqslant k$$

由于$0<a<1$,在不等式两边同时乘以$a(1-a)$并整理得

$$\sum_{q=1}^{n}\frac{C_q}{\tau_q\cdot P}\leqslant k\left(1-\frac{C_{\max}}{D_{\min}}\right)\left[\frac{1}{4}-\left(\frac{1}{2}-a\right)^2\right] \qquad (8)$$

上式为任务集S可在处理器集Ω的条件,其左边为任务集S的利用率,显然等$a=\frac{1}{2}$时,其利用率取最大值为$\frac{1}{4}k$,此时$D_{\min}=\frac{1}{2}P_{\min}$,代入上式即得结论.证毕.

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,根据任务模型,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$S_p$和$S_b$在处理器$Pr_i$上可调度的条件为:

$$\sum_{j=1}^{m}\frac{C_j}{\tau_j\cdot P}+U_m+\frac{C_{\max,i}}{P_{\min,i}}\leqslant 0.5 \qquad (9)$$

式中,$U_m=\max_{Pr_q\in\Omega,q\neq i}\left\{\sum_{\tau'_j\cdot t^p\cdot Pr=Pr_q}\frac{C_j}{\tau'_j\cdot P}\right\}$,

$C_{\max,i}=\max_{\tau_m\in S_p\cup S_b}\{C_m\}$,$P_{\min,i}=\min_{\tau_m\in S_p\cup S_b}\{\tau_m\cdot P\}$.

证明　由前面分析可知,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$\forall Pr_q(Pr_q\in\Omega,j\neq i)$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[9]的分析可以得到处理器Pr_i中任务可调度的条件为:

$$\sum_{j=1}^{m}\frac{C_j}{0.5\tau_j\cdot P}+\sum_{\tau'_j\cdot t^p\cdot Pr=Pr_q}\frac{C_j}{0.5\tau'_j\cdot P}+\max_{\tau_l\in S_p\cup S_b}\left[\frac{B_l}{\tau_l\cdot P}\right]\leqslant 1 \qquad (10)$$

显然

$$\max_{\tau_l\in S_p\cup S_b}\left[\frac{B_l}{\tau_l\cdot P}\right]=\frac{C_{\max,i}}{P_{\min,i}} \qquad (11)$$

由于Pr_q是任意的,显然当$\sum_{\tau'_j\cdot t^p\cdot Pr=Pr_q}\frac{C_j}{0.5\tau'_j\cdot P}$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(10)式,即任务是可调度的,则处理器Pr_i上的任务在任一时刻都可调度.令

$$U_m=\max_{Pr_q\in\Omega,q\neq i}\left\{\sum_{\tau'_j\cdot t^p\cdot Pr=Pr_q}\frac{C_j}{0.5\tau'_j\cdot P}\right\} \qquad (12)$$

再由(10)式和(11)式可得结论.证毕.

2.2　任务分配算法

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:

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$Pr_i\cdot\Delta=0$,$Pr_i\cdot len=0$,$Pr_i\cdot ta=0$.

(2)如果$\exists\tau_i$,$\tau_i\cdot t^p\cdot C+\tau_i\cdot t^b\cdot C=2C_i>\tau_i\cdot P$,则Sch_Success=失败,转到(7),否则转到(3).

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

(4)对于任务τ_i基版本$\tau_i\cdot t^p$,如果存在处理器Pr_j满足$Pr_j\cdot\Delta=\min_{Pr_q\in\Omega}\{Pr_q\cdot\Delta\}$,并令$Pr_j\cdot\Delta=Pr_j\cdot\Delta+\frac{C_i}{\tau_i\cdot P}$,$Pr_j\cdot ta=\max_{\tau_m\cdot Pr=Pr_j,m\neq l}\left\{\frac{C_m}{\tau_l\cdot P}\right\}$;如果该处理器的任务不满足$Pr_j\cdot\Delta+Pr_j\cdot len+Pr_j\cdot ta\leqslant 0.5$,则Sch_Success=失败,转到(7),否则将任务τ_i基版本$\tau_i\cdot t^p$分配到处理器Pr_j并转到(5).

(5)对于任务τ_i副版本$\tau_i\cdot t^b$,如果存在处理

器 $Pr_l(l \neq j)$ 满足

$$\sum_{\substack{\tau_r^b \cdot Pr = Pr_l \\ \tau_r \cdot P' \cdot Pr = Pr_j}} \frac{C_r}{\tau_r \cdot P} = \min_{Pr_s \in \Omega, s \neq j} \left\{ \sum_{\substack{\tau_r^b \cdot Pr = Pr_s \\ \tau_r \cdot P' \cdot Pr = Pr_j}} \frac{C_r}{\tau_r \cdot P} \right\},$$

并令 $Pr_l \cdot len = \max_{Pr_q \in \Omega, q \neq i} \left\{ \sum_{\substack{\tau_j \cdot t^p \cdot Pr = Pr_q \\ \tau_l' \cdot t^b \cdot Pr = Pr_i}} \frac{C_j}{\tau_j' \cdot P} \right\}$,

$Pr_l \cdot ta = \max_{\substack{\tau_m \cdot Pr = Pr_l \\ \tau_s \cdot Pr = Pr_l \\ m \neq s}} \left\{ \frac{C_m}{\tau_s \cdot P} \right\}$; 若该处理器 $Pr_l \cdot \Delta + Pr_l \cdot len + Pr_l \cdot ta > 0.5$, 则 Sch_Success = 失败, 转到(7), 否则将任务 τ_i 副版本 $\tau_i \cdot t^b$ 分配到处理器 Pr_l 转到(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非抢占 EDF 算法动态设置.

3　结论

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非抢占式 EDF 调度算法, 给出了实时分布式控制系统的容错调度算法. 该算法通过以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.

[参考文献]

[1] Kieckhafer R M, Walter C J, Finn A M, *et al*. The MAFT architecture for distributed fault tolerance[J]. IEEE Trans Computers, 1988, 37(4): 398-405.

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

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

[4] 韩宗芬, 秦啸, 庞丽萍, 等. 混合型实时容错调度算法的设计和性能分析[J]. 软件学报, 2000, 11(5): 686-693.

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

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

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

[8] Oh Yingfeng, Song Sang H. Scheduling hard real-time tasks with tolerance of multiple processor failures[J]. Microprocessing and Microprogramming, 1994, 40: 193-206.

[9] Baker T P. Stack-Based Scheduling of Real-time Processes [J]. The Real-Time Systems Journal, 1991, 3(1): 67-100.

Fault-Tolerant Scheduling Algorithm for Distributed Real-Time Systems

LIU Huai

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

Abstract: Fault-tolerant scheduling algorithms at present almost require that the periods of all tasks are the same, but in paretice this is not always the case. The fault-tolerant scheduling algorithm in combination with tasks assignment and scheduling algorithm for unipiocessor was proposed basd on primary/backup copies technique and non pre-emptive EDF. Given the excution times of primary and backup copies being not overlapped by setting their deadlines, the method for setting deadlines of primary and backup copies was given and the schedulability of task set was analyzed.

Key words: Distributed Real-Time System, Fault-Tolerant, Primary Copy/Backup Copy, Non Pre-emptive EDF

[责任编辑: 严海琳]