# 最迟预分配容错实时调度算法设计与分析

雷 航，王 亮

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】提出一种多类型任务集的容错实时调度算法，详细分析该算法的调度机制，证明了该算法的正确性，并给出了该算法的可调度条件，最后通过模拟实验分析了算法的性能。实验表明，调度算法的性能与系统负载、任务出错概率、任务的计算时间等系统参数相关。

关 键 词 多任务; 预分配; 实时调度; 软件容错
中图分类号 TP302.8 文献标识码 A

# Analysis of a Real-Time Scheduling Algorithm with Fault-Tolerance

LEI Hang, WANG Liang

(School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 610054)

**Abstract** The paper presents a kind of real-time scheduling algorithm with fault-tolerance for scheduling various tasks. the scheduling scheme of the algorithm is analyzed in detail. The performance simulation and analysis of the scheduling algorithm are given. The experiment results show that the algorithm performance is related with system load, failure probability, and computation time of tasks.

**Key words** multi-task; predistribution; real-time scheduling; software fault-tolerant

容错技术是实时安全关键系统可靠性保障的重要手段[1]。在实时系统中，必须将容错和实时两种技术相结合[1-4]。以往文献中研究的容错实时调度算法只能调度单一类型的实时任务，未考虑同时调度具有容错需求的实时任务和无容错需求的实时任务，也未能同时调度周期任务和非周期任务，而且任务优先级不能根据需要人为指定。因此，本文建立了一种容错实时调度模型，并在该模型基础上，提出一种基于时间冗余和软件冗余的容错实时调度算法，对多类型的任务集合进行调度。该模型适当放宽了对任务优先级的限制，允许对有容错需求的关键任务人为安排优先级。

## 1 系统模型

多类型任务集容错实时调度模型中，有周期实时任务和非周期任务两大类。前者又分有容错需求和无容错需求两类。有容错需求的周期实时任务也称为关键任务，为检测任务执行是否正确，任务结束前将对结果进行正确性判断(开销计入运行时间)。对于无容错需求任务，假定它的每次运行都是正确的，算法只需保证在截止时间前分配所需的处理器资源而不考虑容错相关的问题。

采用双版本策略对关键任务进行容错，每个关键任务都有两个相互独立的可执行版本(称为主/副版本)。主版本功能复杂、计算量大、运行结果精度高，但由于其较高的复杂度和较大的资源需求，导致可靠性相对较低。副版本功能相对简单、计算量小、运行结果精度不高，因此可靠性较高。当任务的主版本运行出错或系统负载过大引起处理器资源紧张时，为保证任务的时间约束，让副版本运行，使任务仍然能在时限内满足用户的最低需求，提供基本的服务。

定义 1 周期实时任务集合有两个子集，$\tilde{A}_{\mathrm{pd}}=\tilde{A}_{\mathrm{ft}}\cup\tilde{A}_{\mathrm{nft}}=\{\tau_1,\tau_2,\cdots,\tau_{n+m}\}$，其中，集合$\tilde{A}_{\mathrm{ft}}$包含具有容错需求的周期实时任务；集合$\tilde{A}_{\mathrm{nft}}$包含无容错需求的周期实时任务。

假设任务$\tau_i$的周期$T_i$和截止时间$D_i$相等。超周期$T=\mathrm{LCM}(T_1,T_2,\cdots,T_{n+m})$，是$\tilde{A}_{\mathrm{pd}}$中所有任务周期的最小公倍数。由于任务调度在每个超周期$T$中是重复的，所以仅以时间段$[0,T]$作为研究对象。

定义 2 有容错需求任务集合(即关键任务集)$\tilde{A}_{\mathrm{ft}}=\{\tau_1,\tau_2,\cdots,\tau_n\}$；$\forall\tau_i\in\tilde{A}_{\mathrm{ft}}$是一个六元组，即

收稿日期：2005 –03 –21
作者简介：雷 航(1960 – )，男，博士，教授，主要从事嵌入式实时系统方面的研究.

$\tau_i=(T_i, P_{RI_i}, P_i, B_i, p_i, b_i)$，其中$T_i$ 为周期；$P_{RI_i}$为指定的任务优先级；$P_i$和$B_i$分别为 $\tau_i$的主版本和副版本，$p_i$ 和$b_i$分别为$P_i$和$B_i$的计算时间($p_i \geq b_i$，$1 \leq i \leq n$)。

定义 3　若无容错需求任务集合为$\tilde{A}_{nft}=\{\tau_{n+1}, \tau_{n+2}, \cdots, \tau_{n+m}\}$；$\forall \tau_i \in \tilde{A}_{nft}$，$\tau_i$仅有一个运行版本，于是$\tau_i$为一个三元组= ($T_i$, $B_i$, $b_i$)，其中$T_i$ 为周期；$B_i$是$\tau_i$的副版本；$b_i$是$B_i$的计算时间。

定义 4　设$\tilde{A}_{ap}$表示随机性非周期任务集合，$\tilde{A}_{ap}=\{\tau_{n+1}, \tau_{n+2}, \cdots, \tau_{n+m}\}$，$\forall \tau_j \in \tilde{A}_{ap}$是一个二元组，$\tau_j=(A_j, C_j)$，其中$A_j$是$\tau_j$的到达时间；$C_j$是$\tau_j$的计算时间。

定义 5　设$S$为处理机的状态集合，$S=\{S_{\text{free apeirod}}, S_{\text{pd\_primary}}, S_{\text{pd\_backup}}\}$。$S_{\text{free}\mid\text{apeirod}}$表示处理器处于空闲状态或正在处理非周期任务；$S_{\text{pd_primary}}$表示处理器在运行周期实时任务的主版本；$S_{\text{pd_backup}}$表示处理器在运行周期实时任务的副版本。

定义 6　预分配时间段$T_{sp}$表示周期任务预留的处理器执行时间，用以执行副版本。$T_{sp}=(\text{start}, \text{end}, \tau)$，start和end分别为预分配时间段的开始和结束时间；$\tau$指明该$T_{sp}$被分配给哪个任务。

$P(i,j)$为任务实例$I_{ij}$预分配的$T_{sp}$的集合，设其元素个数为$k$，且$k \geq 1$，则：

$$P(i,j)=T_{sp} \mid T_{sp}.\tau=\tau_i=(x_1 y_1 \tau_i),(x_2 y_2 \tau_i),\cdots,(x_k y_k \tau_i) \quad (1)$$

式中 $x_1 < y_1 < x_2 < y_2 < \cdots < x_k < y_k$。

将$T_{sp}$按照时间递增顺序排列，得到预分配时间段表$P_{\text{list}}$，设$P_{\text{list}}$由$N_p$个$T_{sp}$组成。定义如下：

$$P_{\text{list}}=T_{\text{sp}_1} \rightarrow T_{\text{sp}_2} \rightarrow \cdots \rightarrow T_{\text{sp}_{N_p}}$$

式中 $N_p=\sum_{i=1}^{m+n}\sum_{j=1}^{T/T_i}\left|p(i,j)\right|$，且有：

$$\forall r \in [1, N_p] \rightarrow T_{\text{sp}_r} \in \bigcup_{(1\leq i\leq m+n)\wedge(1\leq j\leq (T/T_i))} P(i,j)$$

采用FCFS调度算法为非周期任务分配处理器资源，模型具有以下基本性质。

性质 1 分配给某任务实例的若干个$T_{sp}$，一定处于其就绪时间和截止时间之间的区域内，即在(1)式中，有$x_1 \geq (j-1)\,T_i$，且$y_k \leq jT_i$。

性质 2　为某任务实例预分配的时间段总长度等于其副版本的计算时间，即在式(1)中有：

$$\sum_{r=1}^{k}(y_r - x_r)=(\sum_{r=1}^{k} y_r - \sum_{r=1}^{k} x_r)=b_i$$

性质 3　预分配时间段之间不存在交叉关系，所以有：

$$\forall T_{\text{sp}_i}, T_{\text{sp}_j} \in P_{\text{list}} \begin{pmatrix} i \neq j \rightarrow T_{\text{sp}_i}.\text{end} \leq T_{\text{sp}_j}.\text{start} \\ \text{or} \\ T_{\text{sp}_j}.\text{end} \leq T_{\text{sp}_i}.\text{start} \end{pmatrix}$$

## 2　算法介绍

### 2.1　算法目标及关键技术

调度算法的目标是：保证每个实时任务(主版本或副版本)在截止时间前正确执行的前提下，为关键任务的主版本提供足够大的运行空间，使尽量多的关键任务能运行完主版本。为此，采用基于最后机会原则的预分配思想，在调度过程中采用“资源回收”和“资源检查”策略。

预分配是为任务的每次运行预留处理器时间$b_i$，预留的处理器资源为任务的副版本服务，使主版本出错或系统负载过大时，任务能在截止时间前运行完副版本。预分配时采用了最后机会(Last Chance)原则[5]。让副版本在能满足截止时间要求的最迟时机开始执行(副版本的最迟开始执行时间简称LTSEB)。若主版本在LTSEB时刻仍未完成，则放弃主版本而执行副版本。检查运行主版本的资源要求，满足资源需求时调度主版，否则放弃主版的运行。

### 2.2　算法调度过程及举例

基于调度模型，“最迟预分配容错实时调度算法”(LP_FT)的基本思想是：调度开始前，调度算法为$\tilde{A}_{pd}=\{\tilde{A}_{ft} \cup \tilde{A}_{nft}\}$中所有任务的副版本预分配处理器资源，生成Plist并得到各任务实例的LTSEB。在每个$T_{sp}$内运行相应任务的副版本。当系统时间到达了该$T_{sp}$的开始时间，利用预分配的时间段运行副版本。关键任务在其周期边界就绪，进入关键任务就绪队列。在除预分配之外的时间段，算法仅调度通过了资源检查的最高优先级任务。若有非周期任务到达，则进入$\tilde{A}_{ap}$集合。处理器空闲时，按照FCFS调度$\tilde{A}_{ap}$中的任务。处理机状态变迁过程如图1所示，初始状态为$S_{\text{free}\mid\text{apeirod}}$。

下面以实例说明调度过程。任务集为$\tau_{ft}=\{\tau_1, \tau_2\}$和$\tau_{nft}=\{\tau_3\}$，非周期任务为$\tau_4$，且有：

$$\tau_1 : T_1=6, p_1=2, b_1=1, P_{\text{RI}_1}=1$$

$$\tau_2 : T_2=8, p_2=3, b_2=2, P_{\text{RI}_2}=2$$

$$\tau_3 : T_1=12, b_3=2$$

$$\tau_4 : A_4=3, C_4=2$$

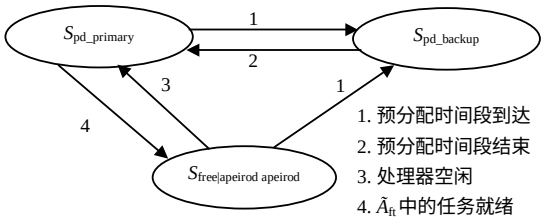


图1　处理机状态变迁图

首先将$\tilde{A}_{pd}$中的实时任务按周期长度非减排序，并在$[T,0]$反向区域内采用RM算法[6]进行逆向调度分析。即按照短周期任务优先为超周期内的各任务实例分配$T_{sp}$。由于逆向调度中的最早完成时间，恰好是正向调度中的最迟开始时间，所以$I_{ij}$.LTSEB随着$T_{sp}$的分配也随之产生。用$B_{ij}$表示为任务实例$I_{ij}$分配的$T_{sp}$，生成的$P_{list}$表为$[5,6,\tau_1]\rightarrow[6,8,\tau_2]\rightarrow[9,11,\tau_3]\rightarrow\cdots\rightarrow[23,24,\tau_1]$。各任务实例副版本的最迟开始执行时间为$I_{11}$.LTSEB=5、$I_{31}$.LTSEB=9等。

当任务出错概率较低时，由于$\tau_2$的优先级高于$\tau_1$，因此$\tau_2$优先执行。任务的主版本运行正确，则撤销为副版本分配的$T_{sp}$；否则，利用$T_{sp}$执行副版本。在某$T_{sp}$被撤销后，$P_{list}$表可能被调整，如$B_{12}$的撤销引起$B_{31}$推迟一个时间单位(从$[9,11,\tau_3]$变为$[10,12,\tau_3]$)。仅当处理器空闲时，运行非周期任务$\tau_4$，当有周期任务需要执行时，$\tau_4$被抢占。当任务出错概率较高时，任务实例$I_{23}$和$I_{14}$在LTSEB时刻仍未执行完主版本，则放弃主版本转而执行副版本。

# 3 算法的正确性和可调度性分析

## 3.1 算法的正确性

对于有容错需求的实时任务，应保证任务执行的时间正确性和执行结果的逻辑正确性；对于无容错需求的实时任务，只需保证其时间正确性。要证明LP_FT的正确性，只需证明满足如下性质。

性质 4 $\forall\tau_i\in\tilde{A}_{nft}$，$\tau_i$的每次执行均能在截止时间到来前完成。

性质 5 $\forall\tau_i\in\tilde{A}_{ft}$，$\tau_i$的每次执行在主版本无错和出错时均能在截止时间到来前获得正确结果。

性质 6 $\forall\tau_i\in\tilde{A}_{pd}$，$\tau_i$的运行不会导致其他任务的运行超越截止时间。

**定理 1** 如果LP_FT算法为实时任务的每个实例成功地预分配了$T_{sp}$，并且处理器严格地利用$T_{sp}$执行相应任务的副版本，则性质4、5和6均满足。

证明：在调度开始前，LP_FT算法进行处理器资源的预分配，假设已经为$\tilde{A}_{pd}$中实时任务的每个实例成功地预分配了$T_{sp}$。$\forall\tau_i\in\tilde{A}_{nft}$，当$T_{sp}$在开始时刻达时，系统立刻调度$\tau_i$连续运行$T_{sp}$.strat–$T_{sp}$.end个时间单位。因此，$I_{ij}$能执行完成。此外，由性质1可知完成时间$y_k\le jT_i$，则执行满足时限要求。性质4满足。

$\forall\tau_i\in\tilde{A}_{ft}$，若$\tau_i$在$I_{ij}$.LTSEB时刻前执行完主版本并运行正确，则为$I_{ij}$分配的$T_{sp}$撤销。由于$I_{ij}$.LTSEB$\le jT_i$，$\tau_i$在截止时间到来前获得正确的运行结果。若$\tau_i$在$I_{ij}$.LTSEB时刻前未执行完主版本或者执行完主版本但运行出错，则为$I_{ij}$分配的$T_{sp}$不会撤销，则与性质4的证明同理，性质5满足。

$\forall\tau_i\in\tilde{A}_{pd}$，由于$\tau_i$的主版本只能在预分配之外的时间段中运行，且预分配时间段之间无交叉关系，所以$\tau_i$的副版本只能在为其分配的$T_{sp}$中运行，不会导致其他任务超出截止时间，性质6满足。

## 3.2 可调度性分析

**定义 7** 系统的主版本负载$U_{lp}$(Load of Primary)和系统的副版本负载 (Load of Backup) $U_{lb}$分别表示以$p_i$和$b_i$计算的处理器利用率之和。

若任务集的处理器利用率为$U\le n(2^{1/n}-1)$[6]，则可由RMS调度。由于预分配时仅考虑任务的副版本，所以用$U_{lb}$替换式中的$U$，便可得出LP_FT算法的可调度条件。

**定理 2** 若周期实时任务集$\tilde{A}_{pd}$满足下列条件，则必可由LP_FT算法调度：

$$U_{lb}=\sum_{i=1}^{n+m}(b_i/T_i)\le(n+m)(2^{1/(n+m)}-1)$$

定理2是LP_FT算法的充分可调度条件。事实上，不满足上述定理的实时任务集仍旧可能由LP_FT调度。仿照文献[7]提出的RM算法的充分必要可调度条件，可得LP_FT的充分必要可调度条件。

**定理 3** 若周期实时任务集 $\tilde{A}_{pd}$ 满足条件，则该任务集可采用 LP_FT 算法调度，即：

$$\begin{cases}\forall i:1\le i\le n+m,\\ \min\limits_{(k,l)\in R_i}\left[\sum\limits_{j=1}^{i-1}\dfrac{b_j}{T_j}\dfrac{T_j}{lT_k}\left\lfloor\dfrac{lT_k}{T_j}\right\rfloor+\dfrac{b_i}{lT_k}\right]\le 1\end{cases}$$

式中 $R_i=\left(\left|(k,l)\right|1\le k\le i,l=1,2,\cdots,\left\lfloor\dfrac{T_i}{T_k}\right\rfloor\right)$。

# 4 模拟实验与分析

实验针对实时周期任务和非周期任务研究LP_FT的调度性能。

**定义 8** 主版本执行完成率(Percentage of Completion Perform，PCP)表示关键任务主版本执行完成数占关键任务总数的比率，是衡量算法性能的重要参数，设为$W_{PCP}$。(1) 确定任务周期和计算时间的取值区域$V_R$、$U_{lp}$和$U_{lb}$。(2) 确定集合$\tilde{A}_{ft}$和$\tilde{A}_{nft}$中实时任务的个数$N(\tilde{A}_{ft})$和$N(\tilde{A}_{nft})$。容错率(Fault-Tolerant Percentage，FTP)是集合$\tilde{A}_{ft}$中实时任务的个数占总个数的比率，即$R_{FTP}=N(\tilde{A}_{ft})/N(\tilde{A}_{pd})=N(\tilde{A}_{ft})/(N(\tilde{A}_{ft})+N(\tilde{A}_{nft}))$。(3) 按照设定的参数随机生成$\tilde{A}_{ft}$和$\tilde{A}_{nft}$中的实时任务。实时任务的周期和计算时间

概率密度函数$P\{c=x\}=1/V_R$，$x\in[1, V_R]$。(4) 确定关键任务的出错概率$F_P$。(5) 使用LP_FT模拟调度生成的任务集，根据调度结果计算$W_{PCP}$。(6) 按照上述步骤，进行相互独立的1 000组实验，并得到$W_{PCP}$的平均值。

实验针对实时周期任务和非周期任务研究LP_FT的调度性能。

(1) 实验的参数为$V_R=40$，$U_{lb}=0.3$，$R_{FTP}=0.8$。$F_P$分别取为0.02、0.05和0.1，得到的模拟结果表明：随着$U_{lp}$的增大，$W_{PCP}$减小。因为$U_{lp}$越大，任务主版本的资源需求越大，能完成主版本的任务越少；$R_{FTP}$越小，$W_{PCP}$越大。任务的出错概率越低，主版本运行正确的概率越大，回收的资源越多，其他任务主版本的执行时间越多，$W_{PCP}$越大。

(2) 实验的参数为$V_R=40$；$U_{lp}=0.8$；$R_{FTP}=0.8$。$F_P$分别取为0.02、0.05和0.10，得到的模拟结果表明随着$U_{lb}$的逐渐减小，在同等条件下，$W_{PCP}$呈上升趋势。因为预留的处理器资源减少，可以用来执行主版本的时间就相对增加，主版本运行完成的机会也随之增加。

另外进行了两组试验，分别验证系统的副版本负载$U_{lb}$与$W_{PCP}$之间的关系以及参数$W_{PCP}$与非周期任务的平均响应时间之间的关系。结果表明，非周期任务的计算时间越长，平均响应时间也越大。平均响应时间随着计算时间的$V_R$的增大而增加。此外，系统的副版本负载越低，平均响应时间越小。因为任务副版本执行时间越短，留给非周期任务处理的时间也越多，所以响应时间越小。

## 5　结 束 语

本文提出了可应用于高可靠强实时多任务系统的容错实时调度算法LP_FT，证明了算法的正确性，给出了可调度条件，并针对周期和非周期任务分别建立了性能模拟模型，对LP_FT算法进行了性能分析。下一步的研究工作包括：(1) LP_FT采用RM调度的思想来逆向分配处理器资源，为放宽算法的可调度条件改进模型；(2) 研究将提出一种适合于非周期实时任务的实时调度算法。

## 参 考 文 献

[1] CPISTIAN H. Understanding fault-tolerant distributed systems[J]. Communications of the ACM, 1991, 34(2): 56-78.

[2] SHIN K G, KOOB G. Fault-tolerant in real-time systems[J]. IEEE Real-Time Systems Newsletter, 1991, 7(3): 28-34.

[3] THUEL S, STROSNIDER J. Enhancing fault tolerant of real-time systems through time redundancy[C]// Foundation of Dependable Computing: System Implement. Kluwer: [s.n.], 1994.

[4] LIN K J, NATARAJAN S, LIU J W S. Imprecise results: Utilizing partial computations in real-time systems[C]// Proc.Real-Time Systems Symp. San Jose: [s.n.], 1987.

[5] CHETTO H, CHETTO M. Some results of the earliest deadline scheduling algorithm[J]. IEEE Trans., Software Eng., 1989, 15(10): 1261-1269.

[6] LIU C L, LAYLAND J W. Scheduling algorithm for multiprogramming in a hard-real-time Environment[J]. Journal of the ACM, 1973, 20(1): 40-61.

[7] LEHOCZKY J, SHA L, DING Y. The rate-monotonic scheduling algorithm: exact characterization and average case behaviour[C]// Proceeding IEEE Real-Time System Symposium. Santa Monica: [s.n.], 1989.

编　辑　熊思亮