维普资讯 http://www.cqvip.com

采用抢占阈值调度的具有释放抖动和特定释放偏移的最大响应时间计算方法*)

杨玉海[1,2] 宾雪莲[2] 余胜生[1] 周敬利[1]

(华中科技大学计算机学院 武汉 430074)[1] (空军雷达学院指挥自动化系 武汉 430019)[2]

摘 要 当采用抢占阈值调度时,如果任务具有释放抖动并且对释放偏移有特定要求,任务最大响应时间的计算就很复杂。通过将对响应时间有影响的任务实例划分为4个集合,分别分析得出达到最大响应时间的各种条件,从而进一步得到具有释放抖动和特定释放偏移的周期任务最大响应时间的计算方法。试验结果表明:这种方法的运行时间要远低于采用模拟运行方法时的运行时间。

关键词 抢占阈值调度,释放抖动,释放偏移,最大响应时间,实时系统

Worst-Case Response Time Computation for the Preemptive Threshold Scheduling Periodic Tasks with Release Jitter and Offsets

YANG Yu-Hai[1,2] BIN Xue-Lian[2] YU Sheng-Sheng[1] ZHOU Jing-Li[1]

(School of Computer Science, Huazhong University of Science and Technology, Wuhan 430074)[1]

(Airforce Army Radar Academy, Wuhan 430019)[2]

Abstract It is very complex to compute the worst-case response time for the preemptive threshold scheduling periodic tasks with release jitter and offsets. The task instances, which will impact the response time of a task instance, have been divided into four sets. Then the conditions for its worst-case response time have been found. Thereby the method to compute the worst-case response time of a periodic task with release jitter and offset is proposed. Simulation results show that running time when using this method is much shorter than that using the simulation running method.

Keywords Preemptive threshold scheduling, Release jitter, Release offsets, Worst-case response time

1 引言

目前,一般采用模拟运行的方法对具有特定释放偏移的任务集合进行可调度性分析。模拟运行的方法是在任务的释放时刻、计算时间、截止期限、周期等特征参数已知的情况下,模拟任务的执行情况。当任务没有经历释放抖动时(即到达时刻和释放时刻相等),只需模拟任务在整个超周期中的执行情况。模拟起始时,首先根据在就绪队列中的各个任务实例的优先级,确定出要运行的任务实例,并模拟执行该任务实例。在调度时刻(调度时刻指一个新的任务实例的释放时刻或者一个任务实例的完成时刻),根据任务实例的优先级决定在就绪队列中应该先执行的任务实例。当运行到超周期时,模拟结束,并可从中得到任务的最大响应时间和最小响应时间。如果任务被低优先级任务堵塞,则在最大响应时间中加上最大被堵塞的时间。当任务实例经历释放抖动时,以上模拟运行的方法并不能得到任务的最大响应时间,需要模拟每个任务实例在各种情况下的执行情况,从中求出每个任务实例的最大响应时间,然后再求出任务的最大响应时间。因此导致模拟时间较长。

由于采用模拟运行方法的运行时间是比较大的。当有任务经历释放抖动时,该方法就不再适用了[1]。因此需要研究具有释放抖动和特定释放偏移的周期任务最大响应时间的计算方法。目前已经有研究人员针对采用固定优先级抢占式调度策略给出了最大响应时间的计算方法[2~5],但还没有针对抢占阈值调度策略的具有释放抖动和特定释放偏移的周期任务最大响应时间的计算方法。本文将给出采用抢占阈值调度具有释放抖动和特定释放偏移的周期任务的最大响应时间的计算方法。

当任务对释放偏移有特定要求时,任务最大响应时间的计算很复杂。其原因在于:很难确定任务的关键时刻。对于释放偏移有特定要求的任务,关键时刻不是该任务与所有高优先级任务同时释放的时刻[6~9],关键时刻与在超周期中任务的释放偏移相关。为了简化分析,当 $O_i > T_i$ 时,令 $O_i = O_i \bmod T_i$。

2 采用抢占阈值调度的具有释放抖动和特定释放偏移的周期任务最大响应时间分析

考虑一个采用抢占式调度策略的单处理机系统。任务集 $\Gamma=\{\tau_i \mid 1 \leqslant i \leqslant n\}$ 中所有任务都为周期任务,任务之间相互独立。任务 τ_i 由〈周期 T_i、最大计算时间 C_i^{max}、最小计算时间 C_i^{min}、释放偏移 O_i、最大释放抖动 J_i、截止期限 D_i、可抢占优先级 v_i、不可抢占优先级 ρ_i〉表示。任务 τ_i 的截止期限为任意截止期限。任务 τ_i 的请求释放抖动限制在[0, J_i]中。任务 τ_i 第 k 次请求表示为 τ_{ik},设下标从0开始。任务 τ_i 的第1次请求 τ_{i0} 到达时刻 $a_{i0}=O_i$。以 r_{ik} 表示任务实例 τ_{ik} 的释放时刻。

由抢占阈值调度策略可知,不可抢占优先级低于 v_i 的请

*)本文受国家自然科学基金资助(项目标号:60073003)。**杨玉海** 博士研究生,讲师,主要从事实时系统、网络技术、智能化高速存储系统等方面的研究。

维普资讯 http://www.cqvip.com

求对 τ_{ik} 的响应时间没有任何影响。因此以下只考虑不可抢占优先级高于等于 v_i 的请求。

对 τ_{ik} 的响应时间有影响的任务实例分为 4 个集合。

1)Set 0(S_0):可抢占优先级高于等于 v_i、在 r_{ik} 之前到达且最迟释放时刻小于 τ_{ik} 的释放时刻 r_{ik} 的任务实例。

2)Set 1(S_1):可抢占优先级高于等于 v_i、在 r_{ik} 之前到达且最迟释放时刻大于 τ_{ik} 的释放时刻 r_{ik} 的任务实例。

3)Set 2(S_2):可抢占优先级高于 v_i、在 r_{ik} 时(或 r_{ik} 之后)到达的任务实例。

4)Set 3(S_3):可抢占优先级低于 v_i,而不可抢占优先级高于等于 v_i 的任务实例。

定理 1 当满足以下条件时,τ_{ik} 的响应时间最大。

1)τ_{ik} 经历最大释放抖动,τ_{ik} 以及所有属于 S_0、S_1、S_2 的任务实例都以最大计算时间执行。

2)所有属于 S_0 的任务实例以最迟释放时刻释放。

3)所有属于 S_1 的任务实例经历的释放抖动时间,使得它们的释放时刻与 τ_{ik} 的释放时刻相等。

4)所有属于 S_2 的任务实例的释放抖动为 0。

5)S_3 中计算时间最大的任务堵塞 τ_{ik} 的执行。

证明:(条件 1)要使得 τ_{ik} 的响应时间最大,条件 1 显然成立。

(条件 2)设任务实例 $\tau_{jm}\in S_0$ 没有经历最大释放抖动。推迟 τ_{jm} 的释放时刻,将会推迟 τ_{ik} 执行。因此当以最迟释放时刻释放 τ_{jm} 时,τ_{jm} 对 τ_{ik} 干扰的可能性也达到最大。

(条件 3)设任务实例 $\tau_{jm}\in S_1$ 经历的释放抖动时间,使得 τ_{jm} 正好在时刻 r_{ik} 释放。这时,τ_{jm} 对 τ_{ik} 的干扰达到最大。减少 τ_{jm} 的释放时刻或者增加 τ_{jm} 的释放时刻,只会减少 τ_{jm} 对 τ_{ik} 的干扰。

(条件 4)设任务实例 $\tau_{jm}\in S_2$ 没有经历释放抖动。增加 τ_{jm} 的释放抖动,将会减少 τ_{jm} 对 τ_{ik} 干扰的可能性,因此当 τ_{jm} 经历的释放抖动为 0 时,τ_{jm} 对 τ_{ik} 干扰的可能性达到最大。

(条件 5)由 S_3 的定义可知,S_3 与 $lhp(i)$($lhp(i)$为任务的抢占优先级低于 v_i,不可抢占优先级高于或等于 v_i 的任务集合)相等。任务 τ_i 最多只能被 $lhp(i)$中的一个任务堵塞。因此要使 τ_{ik} 的响应时间最大,S_3 中计算时间最大的任务应堵塞 τ_{ik} 的执行。证毕。

以 $S_i(k)$表示任务 τ_{ik} 开始执行时刻:即 τ_{ik} 的忙周期开始时刻,忙周期的优先级为 ρ_i。

由于采用抢占阈值调度策略,在 τ_{ik} 开始执行之后,τ_{ik} 的优先级由 v_i 提高到 ρ_i,因此要计算 τ_{ik} 的响应时间,需先求出 $S_i(k)$,然后再求出 τ_{ik} 的响应时间 $R_i(k)$。

将 S_2 又分为两个子集。

· S_{21}:抢占优先级高于 v_i、在 τ_{ik} 的开始执行之前到达的任务实例。

· S_{22}:抢占优先级高于 ρ_i、在 τ_{ik} 的开始执行之后到达的任务实例。

令 $I_0(\tau_{ik})$、$I_1(\tau_{ik})$、$I_{21}(\tau_{ik})$表示 S_0、S_{11} 和 S_{21} 中的任务实例对 τ_{ik} 的干扰时间。$I_{22}(\tau_{ik})$表示以 $S_i(k)$为起点的优先级为 ρ_i 的忙周期长度。

由于在 $S_i(k)$之前,没有执行 τ_{ik},因此在 $S_i(k)$之前,所有抢占优先级高于 v_i 的任务实例都可以抢占 τ_{ik} 的执行,且 S_0、S_1、S_{21} 中任务实例都已执行完。由于确定 S_3 中任务实例是否堵塞 τ_{ik} 是非常复杂的,为了简化分析,假设 S_3 中计算时间最大的任务堵塞 τ_{ik} 的时间。因此 $S_i(k)$的计算公式为:

$$S_i(k)= r_{ik}+I_0(\tau_{ik})+I_1(\tau_{ik})+I_{21}(\tau_{ik})+\max_{\tau_j\in S_3}C_j$$

以下将介绍求 $I_0(\tau_{ik})$、$I_1(\tau_{ik})$、$I_{21}(\tau_{ik})$的方法。

设 τ_j 为可抢占优先级高于 v_i 的任务,τ_{jp} 为在时刻 r_{ik} 时(或 r_{ik} 之后)到达 τ_j 的第一个请求。令 $\phi_{ij,k}$表示时刻 a_{jp} 与 r_{ik} 差值,可知 $0\leqslant\phi_{ij,k}<T_j$;$\delta_{ij,k}$表示 r_{ik} 与在 S_0 中 τ_j 的最后一个任务实例的释放时刻之差;$r_{ik}-t$ 表示在 r_{ik} 之后结束、忙周期优先级为 v_i 的忙周期的开始时刻,φ_j 表示在该忙周期中释放的 τ_j 的第一个请求的释放时刻与时刻 $r_{ik}-t$ 之差,可知 $0\leqslant\varphi_j<T_j$。图 1 为 τ_j 对 τ_{ik} 干扰时间示意图。

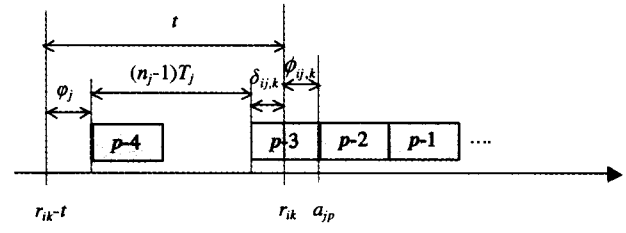

图 1 τ_j 对 τ_{ik} 的干扰时间示意图

2.1 计算 $I_1(\tau_{ik})$,$I_{21}(\tau_{ik})$

由于 τ_{jp} 为在时刻 r_{ik} 时(或 r_{ik} 之后)到达的 τ_j 的第一个请求,因此 $\tau_{jp}\in S_{21}$。从而 $a_{jp}=r_{jp}=p*T_j+O_j$。可以得到

$$p*T_j+O_j=r_{ik}+\phi_{ij,k} \tag{1}$$

由于 $0\leqslant\phi_{ij,k}<T_j$,因此 $0\leqslant p*T_j+O_j-r_{ik}<T_j$。由于 p 为整数,因此可得到

$$\rho=\left\lceil\frac{r_{ik}-O_j}{T_j}\right\rceil \tag{2}$$

将公式 2 代入公式 1 可以得到:

$$\phi_{ik,j}=\left\lceil\frac{r_{ik}-O_j}{T_j}\right\rceil*T_j-(r_{ik}-O_j) \tag{3}$$

由于允许任务的最大释放抖动大于其周期,因此在 S_1 中可能包含该任务的多个请求。设 $N_{ik,j}$为 τ_j 在 S_1 中的请求个数。由于 $\tau_{i(p-N_{ik,j})}$ 为 S_1 中 τ_j 的第一个任务实例,而 $\tau_{i(p-N_{ik,j}-1)}$为 S_0 中的任务实例。因此可以得到以下两个公式:

$$a_{jp}-N_{ik,j}*T_j+J_j\geqslant r_{ik}$$

$$a_{jp}-(N_{ik,j}+1)*T_j+J_j<r_{ik}$$

从而可以得到:

$\phi_{ij,k}-N_{ik,j}*T_j+J_j\geqslant 0$ 并且 $\phi_{ij,k}-(N_{ik,j}+1)*T_j+J_j<0$

由这两个公式可以推出:

$$N_{ik,j}=\left\lfloor\frac{\phi_{ik,j}+J_j}{T_j}\right\rfloor_0 \tag{4}$$

因此 $I_1(\tau_{ik})$为:

$$I_1(\tau_{ik})=\sum_{\tau_j\in hpv(v_i)}N_{ik,j}*C_j^{\max} \tag{5}$$

设 $r_{ik}+W_{ik,1}{}^C$ 为 S_{21}中任务实例的完成时刻。则,

$$I_{21}(t_{ik})\sum_{\tau_j\in hpv(v_i)}\left\lceil\frac{W_{ik,1}^C-\phi_{ik,j}}{T_j}\right\rceil*C_j^{\max} \tag{6}$$

其中,$S_i(k)=r_{ik}+W_{ik,1}{}^C$

2.2 计算 $I_0(\tau_{ik})$

在以 $r_{ik}-t$ 为起点、忙周期优先级为 v_i 的任务实例可能影响 $S_i(k)$。显然,只有当该忙周期的长度大于 t 时,S_0 中的任务实例才可能推迟 τ_{ik} 的执行。

在 r_{ik} 之前释放的 S_0 中最后一个任务实例为 $\tau_{i(p-N_{ik,j}-1)}$,根据定理 1 可知:

维普资讯 http://www.cqvip.com

$$r_{i(p-N_{ik,j}-1)}=r_{ik}+\phi_{ik,j}-(N_{ik,j}+1)*T_j+J_j \quad (7)$$

由于$\delta_{ij,k}$为r_{ik}与$\tau_{i(p-N_{ik,j}-1)}$的释放时刻之差，因此$\delta_{ik,j}=r_{ik}-r_{i(p-N_{ik,j}-1)}$。将公式(7)代入得到：

$$\delta_{ik,j}=(N_{ik,j}+1)*T_j-(\phi_{ij,k}+J_j)$$

由于$a \bmod b=a-b*\lceil a/b \rceil$，因此

$$\delta_{ik,j}=T_j-(\phi_{ij,k}+J_j)\bmod T_j \quad (8)$$

当$t<\delta_{ik,j}$时，显然$[r_{ik}-t, r_{ik}]$中没有τ_j的请求释放，因此不会影响$S_i(k)$。可以看出当$t>\delta_{ik,j}$时，在时间段$[r_{ik}-t, r_{ik}]$中至少有一个τ_j的请求释放。

令$n_{ik,j}$表示S_0中任务τ_j释放的请求个数，则$[r_{ik}-t, r_{ik}]$的长度可以表示为：

$$t=\varphi_j+(n_{ik,j}-1)*T_j+\delta_{ik,j} \quad (9)$$

由于$0\leqslant\varphi_j<T_j$，由公式(9)得到：

$$n_{ik,j}(t)=\left\lfloor\frac{t-\delta_{ik,j}}{T_j}\right\rfloor+1 \quad (10)$$

因此，S_0中任务实例产生的工作负载$W_p(\tau_{ik}, t)$为：

$$W_p(\tau_{ik},t)=\sum_{\tau_j\in hpv(v_i)} n_{ik,j}(t)*C_j^{\max} \quad (11)$$

令$\Delta_{ik}(t)$表示在$[r_{ik}-t, r_{ik}]$中释放的可抢占优先级高于v_i的任务实例在r_{ik}之后还需的CPU时间。若在$[r_{ik}-t, r_{ik}]$中释放的可抢占优先级高于v_i的任务实例在时刻r_{ik}（或在r_{ik}之前）完成，则不会对$S_i(k)$造成影响，因此

$$\Delta_{ik}(t)=\max(W_p(\tau_{ik},t)-t,0) \quad (12)$$

由上可知$\Delta_{ik}(t)$的大小与t相关，因此需要根据t的取值，求出$\Delta_{ik}(t)$。显然t小于等于忙周期优先级为v_i的忙周期的最大长度L_i。L_i的计算方法请参见文[6]。

计算$W_p(\tau_{ik}, t)$需要通过以下迭代方法：$W_p^{n+1}=\sum_{\tau_j\in hpv(v_i)} n_{ik,j}(W_p^n)*C_j^{\max}$。由于在$r_{ik}$之后，$r_{ik}$至少要执行$C_i^{max}$的时间，因此$W_p{}^0(\tau_{ik},t)=L_i-C_i{}^{\max}$。迭代的计算结果记为$W_p^*$。显然$t$小于$W_p^*$。

由于$r_{ik}-t$为忙周期的起始时刻，因此至少有一个属于S_0的任务实例的释放时刻等于$r_{ik}-t$。在S_0中的τ_j的任务实例的释放时刻为：$r_{ik}-\delta_{ik,j}-l*T_j$，其中$l$为$S_0$中$\tau_j$的请求个数。因此需检查在$S_0$中是否存在任务实例的释放时刻与$r_{ik}-t$相等。在$S_0$中$\tau_j$的任务实例个数最大$l_j{}^{\max}$：$l_j{}^{\max}=\left\lfloor\frac{W_p^*-\delta_{ik,j}}{T_j}\right\rfloor$。

综上可知，公式(12)又可写为：

$$I_0(\tau_{ik})=\max_{t\in\Lambda(\tau_{ik})}\Delta_{ik}(t)\quad \Lambda(\tau_{ik})=\{\delta_{ik,j}+l*T_j \mid j=1,\cdots,i;l=0,\cdots,l_j{}^{\max}\} \quad (13)$$

其中，$\Delta_{ik}(t)=\max(W_p(\tau_{ik},t)-t,0)$

2.3 计算$I_{22}(\tau_{ik})$

在$S_i(k)$之后，τ_{ik}的优先级为不可抢占优先级ρ_i，因此在$S_i(k)$之后，对τ_{ik}的响应时间造成影响的只有S_{22}中的任务实例。

设在S_{22}中τ_j（τ_j的可抢占优先级高于ρ_i）的实例个数为

$$m_{ik,j}=\left\lceil\frac{S_i(k)+I_{22}(\tau_{ik})-O_j}{T_j}\right\rceil-\left\lceil\frac{S_i(k)-O_j}{T_j}\right\rceil。$$

因此，

$$I_{22}(\tau_{ik})=C_i^{\max}+\sum_{\tau_j\in hpv(\rho_i)} m_{ik,j}*C_j^{\max}=C_i^{\max}+\sum_{\tau_j\in hpv(\rho_i)}\left(\left\lceil\frac{S_i(k)+I_{22}(\tau_{ik})-O_j}{T_j}\right\rceil-\left\lceil\frac{S_i(k)-O_j}{T_j}\right\rceil\right)*C_j^{\max} \quad (14)$$

2.4 计算τ_{ik}的最大响应时间

根据以上推导可以得到τ_{ik}请求的最大响应时间的计算公式。

$$R_i(k)=S_i(k)+I_{22}(\tau_{ik})-r_{ik} \quad (15)$$

$$S_i(k)=\max_{\tau_j\in S_3}C_j+I_0(\tau_{ik})+I_1(\tau_{ik})+I_{21}(\tau_{ik})+r_{ik} \quad (16)$$

$$r_{ik}=a_{ik}+J_i=k*T_i+O_i+J_i \quad (17)$$

最大响应时间$R_i=\max\{R_i(k) \mid 1\leqslant k\leqslant\lceil 2H/T_i\rceil\}$。$H$为任务集中任务周期的最小公倍数。

3 性能比较

为了对本文所给方法的性能进行评价，进行了两组实验。

实验1：采用文[2]中实验1的数据生成办法，即任务满足以下条件：

1. 任务个数服从[2,15]的均匀分布；
2. 任务的周期为集合[5,15,30,60,100]中的任意一个；
3. 任务的截止期限服从[0,2*周期]的均匀分布；
4. 释放偏移服从[0,周期)的均匀分布；
5. 最大释放抖动服从[0,周期)的均匀分布；
6. 任务的计算时间能够使得任务集合的资源利用率为0.9。

实验2：为了分析当任务的周期变化比较大时，本节所给方法的性能，采用文[2]中实验2的数据生成办法，即任务满足以下条件：

1. 任务个数服从[10,50]的均匀分布；
2. 任务的周期为集合[5,10,15,20,30,50,60,80,100]中的任意一个；
3. 任务的截止期限服从[0,2*周期]的均匀分布；
4. 释放偏移服从[0,周期)的均匀分布；
5. 最大释放抖动服从[0,周期]的均匀分布；
6. 任务的计算时间能够使得任务集合的资源利用率为0.9。

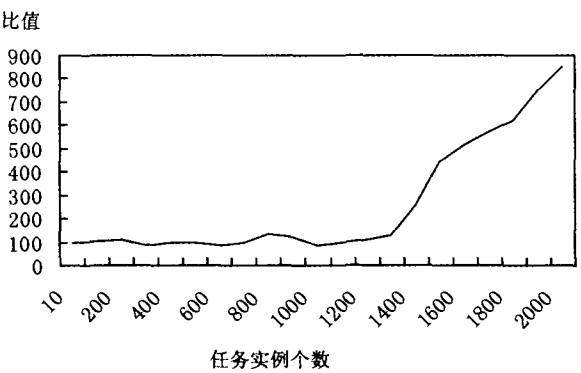

图2 模拟运行方法与新方法的运行时间比值分布图(实验1)

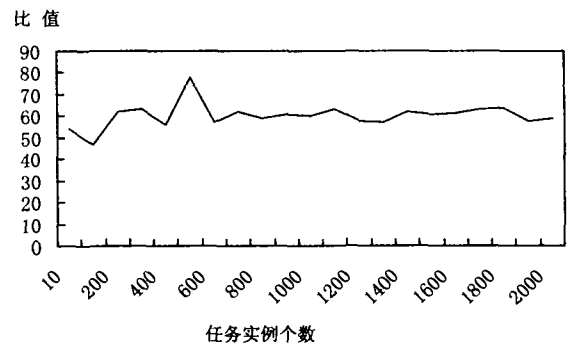

图3 模拟运行方法与新方法的运行时间比值分布图(实验2)

(下转第154页)

维普资讯 http://www.cqvip.com

表 2 三种不同方法计算结果比较

需求量	路段	H-J	EDO	GA
100	y1	1.25	1.31	1.33
	y2	1.20	1.19	1.22
	y3	0.00	0.06	0.00
	y4	0.95	0.94	0.97
	y5	1.10	1.06	1.09
	目标值	1200.61	1200.64	1200.58
	分配次数	11	8	5400
150	y1	5.95	5.98	6.04
	y2	5.65	5.52	5.47
	y3	0.00	0.02	0.00
	y4	4.60	4.61	4.65
	y5	5.20	5.27	5.26
	目标值	3156.38	3156.24	3156.21
	分配次数	14	11	7500
200	y1	13.00	12.85	13.03
	y2	11.75	12.02	11.68
	y3	0.00	0.02	0.01
	y4	10.25	10.33	10.30
	y5	11.75	11.77	11.75
	目标值	7086.21	7086.45	7086.15
	分配次数	20	12	4200
300	y1	28.44	28.11	28.44
	y2	25.75	26.03	25.74
	y3	0.00	0.01	0.00
	y4	23.44	23.39	23.44
	y5	26.56	26.58	26.54
	目标值	21209.91	21210.54	21209.91
	分配次数	28	14	7710

结束语 笔者提出的算法具有良好的搜索性能，对于处理有界连续变量的规划问题具有普遍适用性。对于求解连续交通网络设计问题，算法也具有一定实用价值。但从算例可以看出，算法需要较多的交通分配次数，对于大型路网来说，势必消耗大量的 CPU 时间。因此，有必要对算法进行更进一步的改进。可以从两个方面着手解决交通分配的问题，一是寻求更快速的交通分配算法；二是在遗传算法中尽量减少交通分配的次数，可以通过构造一个标准（例如采用类似隐枚举法的标准），仅对满足一定条件的方案进行交通分配，不满足条件的方案的目标函数依照“满足一定条件的方案比不满足条件的目标值好”的原则，参照给出目标值。由于篇幅限制，笔者将另文给出解决办法。

参 考 文 献

1 刘灿齐. 现代交通规划学[M]. 北京：人民交通出版社，2001

2 袁亚湘. 非线性规划数值方法[M]. 上海：上海科技出版社，1993

3 Suwansirikul C, Friesz, Tobin R L. Equilibrium Decomposed Optimization: A Heuristic for the Continuous Equilibrium Network Design Problem. Transportation Science, 1987, 21:254～263

（上接第 57 页）

以运行时间作为性能评价的标准。由于本文所给方法和模拟运行的方法的运行时间都与在超周期中所有任务的请求个数相关，因此以任务集合中的任务实例个数进行分类。实验中共生成 20 个集合类。集合类 1 包括所有任务实例个数小于等于 10 的集合；集合类 2 包括所有任务实例个数大于 10 且小于等于 100 的集合；集合类 3 包括所有任务实例个数大于 100 且小于等于 200 的集合；…；集合类 20 包括所有任务实例个数大于 1900 且小于等于 2000 的集合。每个集合类中有 20 个任务集。

图 2 与图 3 是模拟运行方法与采用本文提出新方法的运行时间比值分布图，图中比值表示模拟运行方法的运行时间除以新方法的运行时间。从图 2、图 3 中可以看出在任务实例个数的变化过程中，新方法的运行时间始终小于模拟运行方法的运行时间。

小结 本文针对抢占阈值调度策略，给出了具有释放抖动和特定释放偏移的周期任务最大响应时间的计算方法。本文提出的新方法是对强实时任务的可调度性分析方法的补充和完善，对判断强实时周期任务的可调度性具有重要意义。

参 考 文 献

1 宾雪莲，杨玉海，金士尧. 基于 EDF 抢占式调度的周期任务最小响应时间分析. 计算机科学，2004，31(9)：114～116

2 Redell O, Törngren M. Caculating exact worst-case response times for static priority scheduled tasks with offsets and jitter. In: Proceedings of the 8th real-time and embedded technology and applications symposium. RTAS'02, San Jose. CA. USA. IEEE computer society press, 2002. 164～172

3 Bate I, Burns A. Schedulability analysis of fixed priority real-time systems with offsets. PEUROMICRO-RTS'97. IEEE computer society press, 97

4 Redell O. Response time analysis for implementation of distributed control systems:[Phd thesis]. Royal institue of technology, KTH, Sweden, 2003

5 Goosesens J, Devillers R. The non-optimality of the monotonic priority assignments for hard real-time offset free systems. Real-time System, 1997, 13(2):107～126

6 杨玉海，宾雪莲，金士尧. 基于抢占阈值调度的周期任务最小响应时间分析. 计算机应用研究，2004，21(11)：41～44

7 宾雪莲，金士尧，杨玉海. 周期多帧任务模型的响应时间分析. 计算机工程与科学，2003，25(6)：104～108

8 宾雪莲，实时任务调度分析：[博士论文]. 长沙：国防科技大学研究生院，2004

9 宾雪莲，杨玉海，金士尧. 一种基于分组与适当选取策略的实时多处理器的动态调度算法. 计算机学报，2006，29(1)：81～91

（上接第 125 页）

4 He H. Automatic Integration of Web Search Interfaces with WISE-Integrator. WWW2004, New York, NY, 2004

5 Wu W S, Yu C. An Interactive Clustering-based Approach to Integrating Source Query Interfaces on the Deep Web. SIGMOD2004, Paris, France, 2004

6 Nambiar U, Kambhampati S. Answering Imprecise Queries over Autonomous Web Databases. ICDE, 2006

7 Zhang W, Li J Z. Processing Probabilistic Range Query over Imprecise Data Based on Quality of Result. LNCS3842, 2006. 441～449

8 Bilke A. Schema Matching using Duplicates. ICDE2005, Tokyo, Japan, 2005

9 Wang J Y. Instance-based Schema Matching for Web Databases by Domain-specific Query Probing. VLDB2004, Toronto, Canada, 2004

10 He B. Statistical Schema Matching across Web Query Interfaces. SIGMOD2003, San Diego, California, 2003

11 Anjomshoaa A. The Design and Implementation of Grid Database Services in OGSA-DAI. In: Proceedings of UK e-Science All Hands Meeting, Nottingham, EPSRC, 2003

12 Paques H, Liu L, Pu C. Distributed Query Adaptation and Its Trade-offs. SAC 2003, Melbourne, Florida, USA, 2003

13 Urhan T. Dynamic Pipeline Scheduling for Improving Interactive Query Performance. VLDB2001, Rome, Italy, 2001

14 Naughton J, DeWitt D, Maier D. The Niagara Internet Query System. VLDB2000, Cairo, Egypt, 2000

15 Ives Z. Adaptive Query Processing for Internet Applications. IEEE Data Engineering Bulletin, 2000, 23(2):19～26