基于自平衡的非周期任务动态实时调度算法*)

陈旭东 朱清新 廖 勇 熊光泽

(电子科技大学计算机科学与工程学院实时系统研究室 成都 610054)

摘 要 为了更有效地进行开放和不可预测系统的载荷管理,提出了一种基于自平衡的动态实时调度模型——DRSSR(Dynamic Real-time Scheduling based on Self-Regulation)。DRSSR把许可控制和QoS降级相结合,采用反馈控制技术来确保系统的性能、消除干扰和提高系统吞吐率。建立了DRSSR的数学模型,并分析了其稳态性能和瞬态性能。最后,一组基于实时操作系统CRTOS-Ⅱ的实验表明,DRSSR在不确定实时环境中具有良好的性能,并且响应快、实现简单。

关键词 实时系统,动态调度,非周期,自平衡,反馈控制

A Dynamic Real-Time Scheduling Algorithm for Aperiodic Tasks Based on Self-Regulation

CHEN Xu-Dong ZHU Qing-Xin LIAO Yong XIONG Guang-Ze

(Real-Time Systems Lab, School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu 610054)

Abstract A category of performance-critical real-time systems executing in open and unpredictable environment have been rapidly growing. This paper proposes a DRSSR (Dynamic Real-time Scheduling based on Self-Regulation) model to deal with such an unpredictable environment. The DRSSR integrates admission control with QoS degradation, which is able to dynamically determine the QoS of incoming tasks, meet their timeliness requirements, eliminate the disturbance and improve the system throughput. Then, this paper analyses the DRSSR's transient and steady state performance. Finally, a set of experiments, based on the real-time operating system CRTOS-Ⅱ, demonstrate that the DRSSR performs well under dynamical real-time environment and response quickly to the unpredictable variations.

Keywords Real-time systems, Dynamic scheduling, Aperiodic, Self-regulation, Feedback control

越来越多的开放和不可预测系统都要求满足实时性能要求,如航空电子、舰载计算、在线贸易、柔性制造和C4I等[1,25]。如果不能确保其实时性能,就可能造成各种损失,如顾客流失、金融损失、不可靠甚至任务失败等。对于这些应用而言,请求的到达率是非周期性的,对资源的需求是随机不可预知的[2],可能出现过载、系统利用率低等现象。为了更有效地解决这些问题,提出了一种针对非周期任务的反馈控制实时调度模型,称之为DRSSR (Dynamic Real-time Scheduling based on Self-Regulation,基于自平衡的动态实时调度)。

在上述动态系统中,一方面请求的非周期性到达可能导致系统过载(Overload),而另一方面WCET (Worst Case Execution Time)的假设可能导致系统利用率低下。为此,DRSSR引入了"自平衡"的机制,自适应系统载荷的动态变化,同时又利用反馈机制来确保系统性能稳定在某设定值、消除干扰和提高系统利用率。

1 相关工作

自1973年Liu和Layland[3]提出RM(Rate Monotonic)调度算法以来,实时应用越来越广泛,其研究重点也从静态调度过渡到动态调度、从可预测环境过渡到不可预测环境、从周期性任务过渡到非周期性任务。经典的实时调度算法假设任务参数是事先知道的,在任务运行前可以通过精心设计来保障其性能和时间约束(Time Constraints)。常见的调度算法RM[3,4]和EDF(Earliest Deadline First)[3,5]等都属于此类。

然而,对于航空电子、舰载计算、在线贸易等开放和不可预测系统而言,任务的到达时间和参数均无法预知,这为实时调度理论带来了新的问题和挑战。许可控制(Admission Control)是一个最常用的解决方法[1]。但是许可控制会降低系统利用率,浪费宝贵的计算资源。QoS降级(QoS Degradation)是另外一个可行办法[2]。类似的方法还有资源预留技术(Resource Reservation)[6]和非精确计算(Imprecise Computation)[7,8]等。Buttazzo等提出了一种弹簧调度算法(Elastic Scheduling)来进行载荷管理[9,10]。Pedro等人提出了一种基于降级的启发式算法,来适应载荷的动态变化和处理过载[11]。

C. Lu等人认为单纯使用上述技术都是一种开环(Open-Loop)的方式[2]。这些方法都需要反复设计、调试和测试,不能充分利用计算资源,不具有通用性[2],因此有必要引进反馈控制技术(Feedback Control)。Steere等构造了一个反馈调度器来分配CPU[12],Abeni等提出了一种基于预留技术的反馈调度方法[13],Cervin等建立了用于数字控制系统(Digital Control Systems)的反馈调度器[14],Abdelzaher等人描述了基于许可控制的反馈调度方法[15],C. Lu等人构造了任务执行时间未知的反馈调度方法[2,16]并扩展到了端到端系统(End-to-End System)[17]。

国内的研究力量还主要停留在对经典调度算法的改进,只有少量关于动态调度的研究,而且仅限于在控制方法上的改进,如:李允等[21,22]将回馈控制应用到普适计算中,实现了普适计算终端的自适应资源分配;童立靖等[23]提出了基于PID回馈控制的分时调度算法以自适应地分配CPU带宽;文

*)基金项目:国家"十五"预研项目(41315040106)、国家863计划资助项目(2003AA1Z2210)。陈旭东 博士生,主要研究领域为:实时调度、最优控制、自适应控制;朱清新 博士生导师、教授,主要研究领域为:最优控制、最优搜索;廖 勇 博士生,主要研究领域为:实时调度、分布式实时计算;熊光泽 博士生导师、教授,主要研究领域为:实时计算、分布式实时系统、普适计算、高可信赖系统等。

[18,19]分别将自适应控制技术与模糊控制技术应用到实时系统中。

反馈调度算法的关键是如何刻画计算系统的行为。DRSSR从实时计算系统本身出发,采用一种模型化实时系统的新方法,把许可控制和QoS降级相结合,运用反馈控制技术来确保系统的性能和提高系统的吞吐率。

2 任务模型

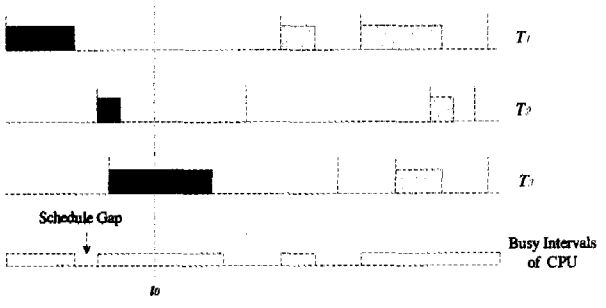

图1 结构化非周期任务模型和瞬态利用率

为了叙述的完整性,首先介绍Abdelzaher等[25]提出的瞬态利用率。瞬态利用率基于一种结构化的非周期任务模型(Acyclic model),利用该模型,文[24]推导出了其可调度性上限。然后,Abdelzaher等证明该模型等价于一般的非周期任务模型(由于该证明不是本文的重点,所以不予叙述),这样上述可调度性上限便可直接应用于一般非周期任务。

一个结构化非周期任务由一系列连续实例组成,每个实例有自己的到达时间A_i、执行时间C_i、相对截止时间D_i和决定截止时间d_i,不同实例的参数彼此独立。一个实例的到达时间等于上一个实例的绝对截止时间,即$A_i=d_{i-1}$(如图1)。图1中有3个任务(T_1, T_2和T_3),矩形表示实例的执行时间(值得注意的是,实例的执行时间可以为0)。

根据结构化非周期任务模型,令$M(t)$表示已到达系统、但截止时间尚未结束的任务实例所组成的集合,则在任意时刻t,$M(t)=\{T_i \mid A_i \leqslant t \leqslant A_i+D_i\}$。图1中黑色矩形表示$t_0$时刻的$M(t)$,最下面一行表示CPU的忙期和闲期。显然,在某个闲期之前到达的实例,对在该闲期之后到达的实例的可调度性不会有任何影响。为此,Abdelzaher定义了"当前实例(current invocations)集",$N(t)\subset M(t)$,$N(t)$只包括$M(t)$中的并且在某个闲期之前到达的实例。比如,在时刻t_0,只有两个实例(T_2和T_3)包括在$N(t_0)$中(排除了T_1),这是因为在T_1所在的忙期和T_2、T_3所在的忙期之间存在着一个闲期。由此,瞬态利用率$U_s(t)$定义为当前实例集中实例的利用率之和:

$$U_s(t)=\sum_{T_i\in N(t)} C_i/D_i$$

根据$U_s(t)$,Abdelzaher推导出了非周期实时任务在Deadline Monotonic (DM)调度策略下的可调度上限(58.6%),该上限可以用于许可控制,以确保非任务的截止时间。此外,根据上定义,某个CPU上的瞬态利用率是小于等于实测利用率的,因为前者考虑了任务截止时间,而后者却没有。Abdelzaher[24]指出,用瞬态利用率可调度上限作为许可控制不会造成CPU的太多浪费。文中实验显示,在用该上限58.6%作为到达任务的许可控制时,CPU的实测利用率几乎接近100%。因此,这一重要性质为我们后面的控制器设计提供了条件和依据。为方便讨论DRSSR,瞬态利用率在下面有时也用术语"队列长度(Queue Length)"表示。

3 系统模型

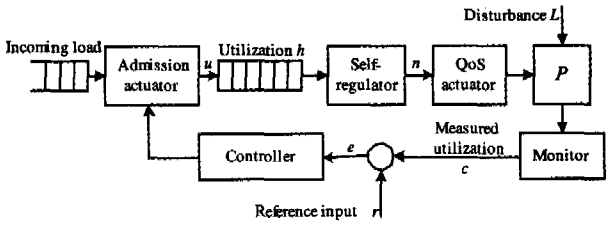

图2 DRSSR的体系结构图

DRSSR包括QoS执行器(QoS Actuator)、自平衡器(Self-Regulator)、处理器(P)、许可控制执行器(Admission Actuator)、反馈控制器(Controller)和监视器(Monitor)等(图2)。其中,QoS执行器负责调整任务的QoS。QoS自平衡器负责根据任务队长动态调整QoS级别因子(QoS Level Factor),反馈控制器负责计算应该输入多少请求载荷,其结果提交给许可控制执行器,负责载荷流量控制。监视器周期性地采集处理器的实际利用率(Measured Utilization)。

3.1 QoS执行器

类似于Tarek F. Abdelzaher的方法[15],QoS执行器有M个离散的服务级别,服务质量从低到高记为1,…, M, 0表示拒绝接受新的请求。另令n表示QoS级别因子,n是一个连续变量,$0\leqslant n\leqslant M$。n可写成$n=I+F$,I和F分别表示n的整数部分和分数部分。如果n是一个整数,它将唯一地决定所有请求的QoS级别;如果n是一个分数,请求的任务中,$(1-F)*100\%$的执行第I个QoS级别,$F*100\%$的执行第$I+1$个QoS级别。

执行器可以通过产生伪随机数的方法来分配到达任务的QoS。令r为[0,1]内的伪随机数,一旦有请求到达系统,就产生一个r。如果$r\leqslant F$,请求就以QoS级别$I+1$执行;否则,就以QoS级别I执行。$n=M$时,所有请求都被提供最好的服务质量;$n=0$时,所有请求都被拒绝。

这样定义的QoS级别可用来决定处理器的处理速度。QoS级别越高,提供的服务质量越好,处理器单位时间内处理的任务数越少;QoS级别越低,提供的服务质量越差,单位时间内处理的任务数越多。

3.2 自平衡

令h表示系统已经接受但正在等待执行的任务队列长度,其值用瞬态利用率来描述。u表示许可控制器允许进入系统的瞬态利用率,也称为控制输入(Control Input)(图2)。

h的变化情况可用下式来描述:

$$\frac{dh}{dt}=u-g(n) \qquad (1)$$

单调非增函数$g(n)$表示QoS级别因子对h的影响,反映处理速度对任务队列长度变化的影响。

由(1)式可见,不使用许可控制,请求率和处理器的处理速度相匹配时,系统的瞬态利用率可以恒定不变,任务集可确保被调度。但对动态系统而言,这是一个十分苛刻的条件。请求率过低时,任务队列长度总在减少并趋近于0,这时系统仍然是可调度的;但是,请求率过高时,任务队列长度会不断增大,以致于出现过载现象。这就是开放和不可预测环境的非自衡性(Non-Self-Regulation)。非自衡性使得系统不能自动确保其性能,因而不能长期不加干预。

可用软件使系统根据任务队列长度来调整QoS级别,称之为自平衡器。请求率过高时,任务队列长度变长,自平衡器

就降低 QoS 级别；反之，请求率降低时，任务队列长度变短，自平衡器就提高 QoS 级别。自平衡可用下面的单调非增函数来表示：$n=f(h)$，于是(1)式可变为：

$$\frac{dh}{dt}=u-g[f(h)] \tag{2}$$

一般情况下，f 和 g 均是非线形的[20]。令 f 和 g 具有如下形式：

$$g=\frac{k_1}{n},f=\frac{k_2}{h}$$

所以，(2)式可变为：

$$\frac{k_2}{k_1}\frac{dh}{dt}+h=\frac{k_2}{k_1}u \tag{3}$$

由(3)式可知，自平衡本身隐含了一个负反馈，因此系统能够根据请求载荷的变化使 h 从一个稳态过渡到另一个稳态。(3)式中 k_1 是系统固有属性，不可以调节；k_2 是自平衡器的参数，可以调节。令 $k=k_2/k_1$，(3)式变为

$$k\frac{dh}{dt}+h=ku \tag{4}$$

u 阶跃变化时，解(4)式可得

$$\Delta h=k\Delta u(1-e^{-t/k}) \tag{5}$$

(5)式描述了 h 的变化过程(图 3)。

图 3 中，k 叫做时间常数。当请求率(u)上升时，系统中等候执行的任务队列(h)变长，处理器开始降低 QoS 级别因子(n)。经过时间 k 后，h 变化了全程的 $1-e^{-1}$，即 63.2%，并依次继续下去，直到最终稳定。

3.3 计算系统及干扰

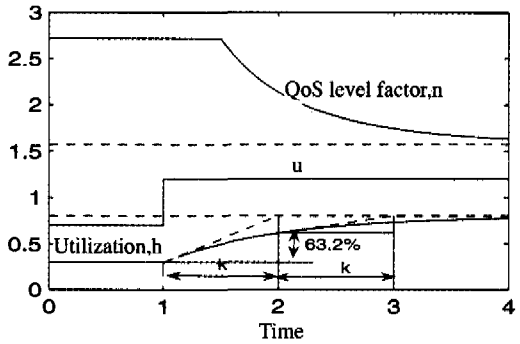

图 3 自平衡过程

由于估计执行时间和实际执行时间总是存在偏差，从而导致系统估计的瞬态利用率(h)和实际测量的瞬态利用率(c)不一致。为了完全确保系统的可调度性，传统的方法是使用 WCET 假设，C. Lu 和 Tarek F. Abdelzaher 是使用反馈控制[1,2][15~17]。反馈控制技术要求模型化这种不一致性。假设 G 表示实际测量的瞬态利用率和估计的瞬态利用率的比率，$G=c/h$。显然，G 是非线形时变参数[15]。可以通过反复实验的方法确定最坏情况下的 G[2,15]，也可采用自适应控制技术来在线辨识 G[18]。而 DRSSR 只确定一个适中可行的 G，把任何引起 G 变化的因数视为干扰(Disturbance)。

把 $h=c/G$ 代入(5)式，就可得到包含自平衡的计算系统的微分方程描述公式：

$$k\frac{dc}{dt}+c=kGu \tag{6}$$

式中，c 是实际测量的瞬态利用率，也叫系统输出(System Output)。

3.4 DRSSR 系统模型

自平衡可使系统稳定，并在一定范围内自适应载荷的动态变化和提高系统吞吐率，却不能保证任务集总是可以被调度。采用负反馈控制可以进行过载保护、抗干扰和进一步提高系统吞吐率。

因为自平衡的作用，使用比例控制器 K_P(Proportional Controller)足以控制计算系统的性能。对(6)式两边做 Laplace 变换，可得到计算系统的频域表示形式：

$$H_1=\frac{c(s)}{u(s)}=\frac{G}{s+1/k}$$

式中，H_1 是计算系统的传递函数，s 表示复变量。图 4 是 DRSSR 模型频域形式的系统框图。比例控制器的结果被输入到许可控制执行器，由其进行流量控制。r 是系统的参考输入，表示用户希望的系统性能。为了保证可调度性，这里 $r=U_d$，两边做 Laplace 变换，得 $r(s)=U_d/s$。

由图 4，系统开环传递函数为：

$$H_2=\frac{c(s)}{r(s)}=\frac{K_PG}{s(s+1/k)}$$

所以，带负反馈控制环的 DRSSR 可用下面的数学模型来描述：

$$c(s)=\frac{H_2}{1+H_2}r(s)=\frac{K_PG}{s^2+s/k+K_PG}r(s) \tag{7}$$

4 性能分析及参数设定

4.1 稳态性能

4.1.1 稳定性(Stability)

根据式(7)，系统的特征多项式(Characteristic Polynomial)为：$q(s)=s^2+s/k+K_PG$。显然，$k>0$，$K_P>0$，$G>0$。根据 Routh-Hurwitz 稳定性判据[20]，二阶系统特征多项式的各项系数为正时，系统是稳定的。

4.1.2 稳态误差(Steady-State Error)

稳态误差是指系统稳定后实测利用率和其期望值之差，表征系统的准确性。

由图 4，DRSSR 的误差为：

$$e(s)=\frac{1}{1+H_2}r(s)$$

根据终值定理(Final-Value Theorem)[20]，$r(s)=U_d/s$ 时，系统的稳态误差 e_s 为

$$e_s=\lim_{t\to\infty}e(t)=\lim_{s\to0}se(s)=\lim_{s\to0}s\frac{1}{1+H_2(s)}\frac{U_d}{s}=0$$

所以，当实时系统过载时，DRSSR 提供了良好的过载保护，确保任务的可调度性。

4.2 瞬态性能

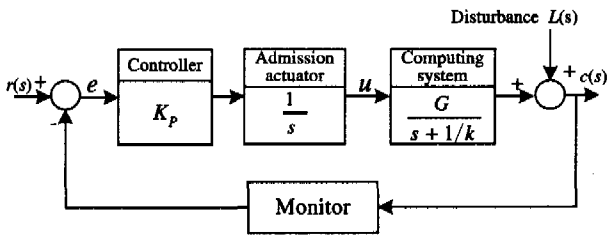

图 4 DRSSR 的反馈控制系统框图

瞬态性能是指当请求载荷变化时，DRSSR 开始工作直到系统重新稳定之前的响应情况。主要的瞬态性能指标有峰值时间(Peak Time)、峰值(Peak)、超调量(Percent Overshot)和调整时间(Settling Time)等。其中，峰值时间是指 DRSSR 开始工作直到其响应首次到达峰值的时间；峰值是指 DRSSR 响应的最大值；超调量是指峰值超出其稳定值的百分比。

二阶系统(7)式可变为如下标准形式

$$c(s)=\frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}r(s) \tag{8}$$

式中，$\omega_n^2=K_PG$，$1/k=2\zeta\omega_n$，ω_n 叫自然频率(Natural Frequen-

cy)，ζ叫阻尼率(Damping Ratio)，这两个参数将决定系统的瞬态性能。G是系统固有属性，可通过调整 K_P 和 k 的值来改变 DRSSR 的瞬态性能。$r(s)=U_d/s$ 时，

$$c(s)=\frac{\omega_n^2}{s(s^2+2\zeta\omega_n s+\omega_n^2)}U_d \tag{9}$$

对(9)式两边做 Laplace 逆变换，得到 DRSSR 的时域响应公式：

$$c(t)=U_d[1-(1/\beta)e^{-\zeta\omega_n t}\sin(\omega_n\beta t+\theta)] \tag{10}$$

式中，$\beta=\sqrt{1-\zeta^2}$，$\theta=\cos^{-1}\zeta$，$0<\zeta<1$。

由(10)式可知，系统的瞬态性能和其参数的关系往往是矛盾的。ζ越小，系统峰值时间越短，但是响应峰值越大，超调量变大。可根据(9)、(10)两式以及参数之间的变换关系，适当选取 K_P 和 k 的值来调节系统的瞬态响应。

K_P 越大，系统峰值时间越短，但是其振荡频率会加剧，同时响应峰值变大。k 越小，调整时间越短，但是峰值时间变大，同时响应峰值变大。

调整时间是另一个动态实时调度算法的关键性能。调整时间如果太长，系统将会长时间过载，造成损失。这里把调整时间定义为使实测利用率 $c(t)$保持在其稳定值的 2%以内所需的时间。由式(9)、(10)，可得到调整时间为 8k，即时间常数的 8 倍。通常情况下，采样周期在 1s 以内，时间常数可定为 1s，此时调整时间(8s)是可以接受的。说明 DRSSR 能迅速对过载做出反应。

4.3 干扰

干扰会影响系统的稳态误差。由图 4，系统误差对干扰的响应可用下面的数学模型来描述：

$$e(s)=-\frac{s^2+s/k}{s^2+s/k+K_PG}L(s)$$

根据终值定理，$L(s)=l_d/s$ 时，

$$\lim_{t\to\infty}e(t)=\lim_{s\to0}se(s)=-\lim_{s\to0}s\frac{s^2+s/k}{s^2+s/k+K_PG}\frac{l_d}{s}=-\frac{l_d}{K_PG}$$

所以，可调整 K_P 以减小干扰的影响。

4.4 采样周期

根据采样定理[20]，为了使 DRSSR 不失真和更加准确，采样周期必须要足够地小。但过小会增加系统开销，降低系统性能。此外，采样周期小到低于任务的截止时间时，反馈控制将失效，系统性能恶化。可以通过反复实验的方式来确定采样周期。下一节的仿真实验中，采样周期为 1s。

5 仿真试验

为了验证 DRSSR 的性能，本文基于实时操作系统 CRTOS-Ⅱ(即由电子科技大学实时系统实验室自主研发的高可信赖的实时操作系统)进行了一组仿真实验。实验中的任务是随机产生的，到达率服从 Poisson 分布。处理器有 3 个 QoS 级别，QoS 级别定义在任务的执行时间上。每个任务的第 3 个 QoS 级别，也就是最好的服务质量级别的执行时间在区间[300 500]ms 内随机产生；第 2 个和第 1 个 QoS 级别分别为第 3 个级别的执行时间的 1/3 和 1/5。实验中的各参数为，$k=1$，$G=0.5$，$K_P=1.87$，采样周期为 1s。底层采用 DM (Deadline Monotonic)调度算法[25]，根据该算法的非周期任务可调度性条件，只要瞬态利用率保持在 Ud (58.6%)以内，到达任务集便是可调度的。

图 5 是 DRSSR 的阶跃响应曲线，实验进行了 300s，图中绘出了非周期任务到达载荷(Incoming load：在任意时间段内到达的任务的执行时间之和除以该时间段的长度)和在该载荷下的实测瞬态利用率。由图可见，虽然到达载荷出现过载和波动(从 96%到 165%变化)，实测瞬态利用率仍能保持在 58.6%附近，而且摆动幅度不超过 0.1，调整时间不超过 10s。这样，DRSSR 就能从概率上确保接收任务的实时性，并使系统在动态环境下保持稳定。

第二个实验用于验证 DRSSR 能够提高系统吞吐量。首先，定义一个评价指标：效率，它指用回馈控制及 DRSSR 算法所能接收的任务数与不用回馈控制、许可控制而成功服务的任务数(满足端到端截止时间)的比值。图 6 绘出了本次实验 DRSSR 的效率曲线(实验中采用的参数同实验 1)，曲线上的每一个点都是 5 次实验结果的平均值，每次实验持续足够长的时间(300s)。从图 6 可以看出，随着到达载荷增加，效率曲线也开始逐渐上升，并最终收敛于某一特定值(效率增加到一定程度就不再增加了)。其原因是：在低载荷时，DRSSR 不会改变任务的 QoS，所有到达任务都将进入系统，并且其截止时间均得到满足，此时的情况与不采用 DRSSR 时是类似的。随着载荷增加，DRSSR 通过 QoS 降级接收更多的任务，相比之下，如果不用 DRSSR，任务的 QoS 不会降级，这样会造成错过截止时间的任务数增加、成功服务的任务数减少。因此，DRSSR 能有效提高系统吞吐量。

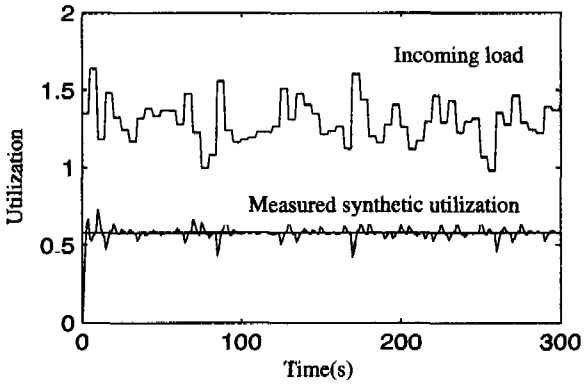

图 5 请求载荷和实测瞬态利用率

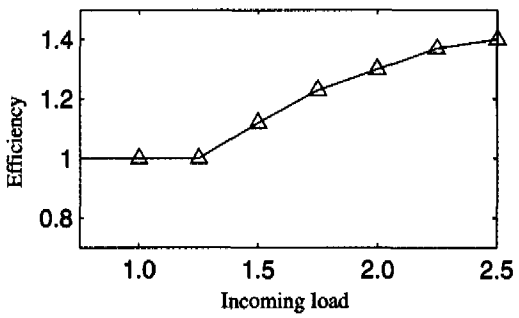

图 6 DRSSR 效率曲线

结论 本文提出了一种针对开放和不可预测实时环境的动态调度模型——DRSSR。在自平衡和负反馈控制的机制下，采用 QoS 降级和许可控制技术相结合的方法来实现过载保护、消除干扰和提高系统吞吐率。实验表明，DRSSR 在系统过载时能迅速响应、改善系统的性能，并且具有响应快、实现简单等优点和良好的稳态和瞬态性能。

参 考 文 献

1 Stankovic J A, Lu C, Son S H, et al. The case for feedback control real-time scheduling. In: EuroMicro Conference on Real-Time Systems, York, UK, 1999. 11～12

2 Lu C, Stankovic J A, et al. Feedback control real-time scheduling: framework, modeling, and algorithms. Real-Time Systems Journal, 2002, 23(1/2): 85～126

3 Liu C L, Layland J W. Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of ACM, 1973, 20 (1): 46～61

(下转封三)

在进行系统设计时，采用程序与数据相分离、图形与属性相分离的原则，将监控数据与 GIS 的图形数据有机结合，利用 MapInfo 和 VC＋＋将监控系统与 GIS 系统进行无缝连接。其系统体系结构如图 1 所示。

4.2 系统主要功能模块

系统由以下功能模块组成：

(1)工作环境初始化模块：在程序安装时进行设置，完成系统工作环境的设置，如使用的数据库系统、工作图层的数量、监控程序的接口、需要在 GIS 中显示的内容等。

(2)监控数据显示模块：实现监控数据在 GIS 上的直观化显示，该模块能够直观显示监控系统的数据和属性。

(3)预警处理模块：负责对需要显示和处理的数据进行预警线的判断，通过数据比较将预警告知监控程序。

(4)GIS 图形处理模块：负责图形的数据采集、编辑、更新等功能。

(5)监控程序接口模块：负责启动监控线程，从而达到监控和显示数据同步的效果。

4.3 系统流程及主要功能模块的实现

(1)监控数据的 GIS 显示与查询功能模块：该模块是系统的核心模块，负责实现将监控点的检测数据直观的显示在 GIS 图形中，当选择一个监控点时，系统将根据系统设置在图形中漂浮显示指定数据。系统图形在显示中每一个监测点用一个圆点代表检测点的地理位置，显示时根据监控数据及监控点通讯情况显示不同的颜色，绿色表示一切正常，红色表示检测数据超标(高于上限或低于下限)，黄色表示检测点与监控中心通讯异常(包括通信异常、监控设备工作异常等)。

数据查询对任意监控点的相关内容按时间段、监控情况、超标情况等进行查询，并将查询结果以报表的形式进行存储和打印。

(2)GIS 图形处理模块：负责图形的数据采集、编辑、更新等功能。主要处理图形数据的采集输入、编辑更新更功能，借助 MapInfo 已有的处理手段将监控系统的监控地理范围内的地理信息输入存储到数据库等文件中，并且根据需要进行编辑修改和更新，供监控系统信息查询模块使用。

(3)预警处理模块：预警处理模块以线程的方式进行工作，它处于定时工作方式，根据监控系统的时间要求定时对监控数据进行检查，当检测数据超限(高于上限或低于下限)时设置相应的标志，其处理流程如图 2 所示。

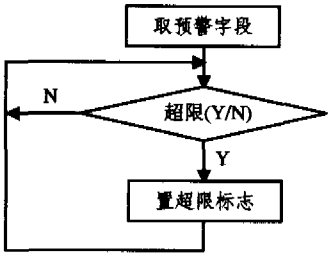

图 2 预警处理

4.4 系统安全

为了保证系统安全，通常从数据安全、访问口令安全等方面着手解决。在数据安全方面监控系统中通常利用数据库系统本身提供的安全措施进行监控系统的数据安全保障，根据系统数据和资金情况可以采用双机自动热备或手动数据备份方法进行。访问口令安全方面采用操作系统级和应用系统级两方面措施，在应用系统级安全性方面包括用户对系统的操作权限、用户对数据库的访问控制以及口令密码加密存放等措施。

结束语 该系统的研制为监控系统提供了一个通用的数据处理平台，通过对中间数据库的定义实现监控数据与 GIS 数据的接口的有机结合，使监控系统的开发更加轻松，达到数据处理重用的目的和效果。

参 考 文 献

1 胡来林. MapX 构件简介及其在开发 GIS 软件中的应用[J]. 微电脑运用，2002，18(5)：32～35
2 萨师煊，王珊. 数据库系统概论[M]. 高等教育出版社，2000
3 陈俊，宫鹏. 实用地理信息系统[M]. 北京科学出版社，1998
4 (美)Charlie Calvert, et al 著. 徐科，冯焱，吕志民，等译. C＋＋Builder 应用开发大全[M]. 清华大学出版社，1999
5 金辉，石敏. 系统集成的可维护性开发[J]. 计算机应用研究，2001，(7)：93～96

(上接第 290 页)

4 Klein M, Ralya T, Pollak B, et al. A practitioner's handbook for real-time analysis - guide to rate monotonic analysis for real-time systems. Kluwer Academic Publisher, 1993
5 Stankovic J A, Spuri M, Rammamritham K, et al. Deadline scheduling for real-time system - EDF and related algorithms. Kluwer Academic Publisher, 1998
6 Mercer C W, Savage S, Tokuda H. Processor capacity reserves: an abstraction for managing processor usage. In: WWOS, 1993. 129～134
7 Chung J Y, Liu J W, Lin K J. Scheduling periodic jobs that allows imprecise results. IEEE Transactions on Computers, 1990, 19(9): 1156～1173
8 Shin W K, Liu J W S. Algorithms for scheduling imprecise computations with timing constraints to minimize maximum error. IEEE Transactions on Computers, 1995, 44(3): 466～471
9 Buttazzo G C, Abeni L. Adaptive workload management through elastic scheduling. Real-Time systems, 2002, 23(1/2): 7～24
10 Buttazzo G C, Lipari G, Caccamo M, et al. Elastic scheduling for flexible workload management. IEEE Transactions on Computers, 2002, 51(3): 289～302
11 Mejia-Alvarez P, Melhem R, Mosse D, et al. An incremental server for scheduling overloaded real-time systems. IEEE Transactions on Computers, 2003, 52(10): 1347～1361
12 Steere D C, Goel A, Gruenberg J, et al. A feedback-driven proportion allocator for real-rate scheduling. In: OSDI, 1999. 145～158
13 Abeni L, Palopoli L, Lipari G, et al. Analysis of a reservation-based feedback scheduler RTSS, 2002. 71～80
14 Cervin A, Eker J, Bernhardsson B, et al. Feedback-feedforward scheduling of LOG-control tasks [J]. Real-Time Systems, 2002, 23(1/2): 25～53
15 Abdelzaher T F, Shin K G, Bhatti N. Performance guarantees for web server end-systems: a control theoretical approach. IEEE Transactions on Parallel and Distributed Systems, 2002, 13(1): 80～96
16 Lu C, Wang X, Gill C D. Feedback control real-time scheduling in ORB middleware. TRAS, 2003
17 Lu C, Wang X, Koutsoukos X, et al. Feedback utilization control in distributed real-time systems with end-to-end tasks. IEEE Transactions on Parallel and Distributed Systems, 2005, 15(6): 550～561
18 邹勇，淮晓永，李明树. 开放式实时系统中的自适应调度方法. 计算机学报，2004，27(1)：58～65
19 金宏，王宏安，傅勇，等. 模糊反馈控制实时调度算法. 软件学报，2004，15(6)：791～798
20 Dorf R C, Bishop R H. Modern Control System (Ninth Edition). Prentice Hall Inc, 2002
21 李允，熊光泽，罗蕾，等. 普及计算终端的自适应性技术研究. 电子学报，2002，30(8)：1121～1126
22 李允，罗蕾，熊光泽. 面向普适计算的自适应技术研究. 电子学报，2004，32(5)：740～744
23 童立靖，淮晓永，李明树. 一种基于 PID 反馈控制的分时调度算法. 计算机研究与发展，2004，41(1)：15～21
24 Abdelzaher T F, Sharma V, Lu Chenyang. A utilization bound for aperiodic tasks and priority driven scheduling. IEEE Transactions on Computers, 2004, 53(3): 334～350
25 LIAO Yong, CHEN Xu-dong, SANG Nan, et al. Optimal Reward Based Adaptive CPU Resource Allocation for Computing Devices in Pervasive Environment. Journal of Information and Computational Science, 2005, 2(1): 75～80