ISSN1000-9825 Journal of Software 软件学报 2000,11(1):95~103

分布式实时系统中的预测调度算法*

许建峰 朱晴波 胡宁 谢立

(南京大学计算机软件新技术国家重点实验室 南京 210093)

E-mail: zqb@dislab.nju.edu.cn

摘要 对于分布式实时系统中的周期性任务,人们提出了一系列静态分配调度算法,有效地解决了各种特定条件下的任务分配和调度问题.这些算法的主要特点是,它们均要求被调度任务的特征参数为已知条件.然而在很多实时系统中,周期性任务的运行时间或任务数量常常是一些具有一定规律的随机过程,因而上述静态算法的效能将受到限制.在分析了特定应用背景中的处理流程之后,抽象得到两类随机任务模型,针对这两类模型介绍了在分布式实时系统中已经得到应用的静态分配调度算法SAA(static allocation algorithms),进而提出了多任务分配调度的预测算法PAA(predicting allocation algorithm).它根据周期性任务执行时间或子任务数量的统计特性,实现任务参量的合理预测和多任务的动态调度,以提高系统的实时性能.仿真结果表明,对于两类任务模型,PAA算法与SAA算法相比,在任务完成时间、负载均衡度、系统响应时间及任务夭折率等多方面均有显著改善.

关键词 分布式实时系统,周期性任务,分配调度算法,预测.

中图法分类号 TP316

在实时多任务系统中,人们既关注事件(event)或任务(task)之间逻辑关系的正确性,也关注系统能否在指定时间内响应各种事件,完成相应的处理任务,这就是系统的实时性问题.当构成系统的硬件设备性能完全确定后,影响系统实时性的便是软件设计中所采取的各种策略和算法.对单机系统而言,设计实现合适的任务调度(scheduling)算法是提高系统实时性能的关键所在;对于分布式系统,由于存在多个处理节点(processing node,简称PN),除了每个PN上的调度算法以外,各个任务在多个PN上的分配(assigning)方法也会在很大程度上影响实时性,所以在分布式实时系统中,多任务的分配调度(allocating)问题一直受到人们的关注.然而,即便是一些最简单的情形,多任务的最优化分配调度问题也是NP难题[1].因此,在研究过程中,常常需要作出一系列假设,进而获得这些假设条件下的优化算法.

一般地,实时系统中的任务可以分为周期性的(periodic)和非周期性的(aperiodic)两大类.对于周期性任务,人们考虑最多、也最为成熟的便是静态算法(static algorithm)[2~10].在一系列静态算法中,有的着重于使整个系统取得最小的计算与通信负载[5];有的力求使处理机之间的负载相对均衡[6,7];有的则以使各个PN中的最大计算时间取得最小值为目标[8~10].这类静态算法的主要问题是,它们均要求被调度任务的特征参数,如运行时间、所含子任务数量、彼此之间的前驱和后继关系、通信关系及通信信息量等为已知,当这些参数未知或变化范围较大时,这类算法便难以达到目标.对此人们开始引入进化算法,其主要思想是,在实时任务对资源需求呈较大幅度变化的条件下,根据实际需求的连续性,逐次递推出每一周期各个任务对资源需求的估计值,以此作为任务调度的依据,这方面的工作已经取得了一些有效的结果[11].目前算法中的不足之处是递推函数的建立有较大的随意性,因而估计值与实际参数之间的偏差难以控制,故不适用于实时性要求比较高的系统.

---

* 本文研究得到国家863高科技项目基金(No.863-308-19-01,863-308-19-02)资助.作者许建峰,1961年生,研究员,主要研究领域为分布式系统,软件工程.朱晴波,1976年生,硕士生,主要研究领域为分布式计算,并行处理.胡宁,1973年生,硕士生,主要研究领域为分布式系统.谢立,1942年生,教授,博士生导师,主要研究领域为分布与并行系统.

本文通讯联系人:许建峰,南京 210093,南京大学计算机软件新技术国家重点实验室

本文1998-11-25收到原稿,1999-02-02收到修改稿

在很多实时系统中,周期性任务在运行时间或子任务数量上常常呈现出具有一定规律的随机性,本文分析了特定应用背景中的处理流程,抽象得到两类随机任务模型,针对这两类模型介绍了在分布式实时系统中得到应用的静态分配调度算法SAA(static allocation algorithm),进而提出了多任务分配调度的预测算法PAA(predicting allocation algorithm).它根据周期性任务执行时间的统计特性,实现多任务的动态调度,以解决随机任务的实时性问题.我们给出了在一些条件下SAA与PAA的算法仿真结果及性能分析,最后给出结论,并提出进一步工作的方向.

1　任务模型

现代信息处理系统如雷达、超声诊断仪和CT等,常常需要面对具有以下特性的信号序列:周期性——由于传感器周期性地采集外部信号,在信息处理中可以认为信号的到达是等周期的,或可近似为等周期的,我们称这种周期为帧(frame);随机性——由于外部环境的变化和测量噪声的存在,各种信号无论在数量上还是在属性上都具有很大的随机性;相关性——单帧信息难以完整地反映信号的属性,但连续数帧内的信号表现出较强的相关性.

对这类信号进行的处理,不能仅仅孤立地对当前帧进行,而必须综合考虑到前后各帧信号数据的特征,才能确定它们的具体属性,这就是在信息处理系统中广为采用的相关处理.例如,在雷达信号和数据处理中,多对象、多层次的相关处理同时存在,使整个信息处理过程可被视为一系列相关处理的组合.下面以雷达目标检测过程为例来描述相关处理的主要特征.

1.1　检测原理

一般地,在提取真实目标信息的过程中,设第i帧输入信号为S_i,它包含若干组超过门限的量测记录,即

$$S_i=\{R_{ij}\mid j=1,2,\ldots,n_i\},\quad i=1,2,3,\ldots;$$

其中R_{ij}是一个量测记录,含有距离、速度、方位、俯仰及幅度等参量;n_i为第i帧输入的量测记录数,它在0与某一上限之间随机变化,故序列$\{S_i\}$为一随机过程.由于信号采样方式的限制以及信号中可能含有噪声、杂波和真实目标等多种信息,对于信号属性的判断必须根据下述检测原理进行.

检测原理　R_{ij}为目标数据的充分必要条件是:除第i帧信号外,在与第i帧序号之差不超过K的$2K$帧信号中,存在不小于$N(\leqslant K)$的L帧信号,其中的L个量测记录$R_{i_1j_1},R_{i_2j_2},\ldots,R_{i_Lj_L}$满足

$$D(R_{ij},R_{i_kj_k})<G,\quad k=1,2,\ldots,L.$$

这里,K和N均为正整数,D为相关判别函数,G为相关门限,这些参量均由具体的物理模型确定.

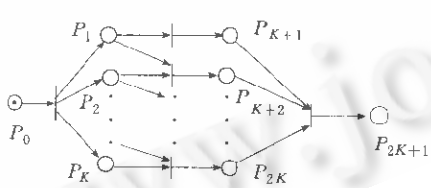

Fig.1　Petri net model of moving windows correlation processing
图1　滑窗相关处理的Petri网模型

根据检测原理,S_i中的每个量测记录都必须与$S_{i-K},\ldots,S_{i-1},S_{i+1},\ldots,S_{i+K}$中的各个量测记录进行相关处理,才能获得其真实属性.在实际处理过程中,对第i帧信号S_i,我们视前K帧信号为一个完整的相关窗,将S_i中每一个量测记录与前K帧信号中每一个量测记录进行相关判别处理.由于相关窗总是随着新的帧信号的输入而不断后移,所以我们就能够实现上述S_i与前后个$2K$帧信号相关的要求,这样的相关过程称为滑窗相关处理,其Petri网模型如图1所示.

此Petri图中各个位置的具体涵义如下.p_0:当前帧信号到达;p_i:当前帧信号与前$K-i+1$帧信号的相关处理$(0<i\leqslant K)$;p_j:前$K-j-2$帧信号取代前$K-j+1$帧信号$(K<j<2K)$;p_{2K}:当前帧信号取代前一帧信号;p_{2K+1}:等待下一帧信号.

根据上述处理过程及模型的描述,我们可以得出滑窗相关处理的主要特征如下.

(1) 若将S_i与前K帧信号的相关处理视为一个完整的任务T_i,则由于S_i的到达是周期性的,所以T_i也是周期性的任务.

(2) 由于 S_i 与 S_j 的相关处理包含了 $n_i \times n_j$ 次彼此之间先后顺序无关的相关函数的计算与判别,所以,若将每一次计算与判别定义为一个子任务,则对于当前帧为 S_i、滑窗宽度为 K 的相关处理任务 T_i,它包含的独立子任务数为

$$n_i(n_{i-1}+n_{i-2}+\ldots+n_{i-K}).$$

为保证此表达式在边界情形下的正确性,我们假定

$$n_0=n_{-1}=\ldots=n_{-K+1}=0.$$

(3) 对于任一 S_i,其所含记录数 n_i 随系统所处环境的变化而变化,因此是一个在一定范围内变化的随机数,所以 T_i 所包含的子任务数是随机的,或者完成 T_i 所需的时间是随机的.

(4) 外部环境具有一定的连续性,因而相邻数帧内的量测记录数也具有较强的相关性,故相邻若干帧上的相关任务 T_i 的运行时间或所含子任务数之间也具有相关性.

1.2 任务模型

根据上述特征,我们将考虑以下两类包含 N 个 PN 的分布式同构系统中的任务模型.

Ⅰ类模型——系统以 T 为周期,每帧需要完成 J 个相互独立的任务 $T_1,T_2,\ldots,T_J$,在此我们不考虑通信及其他资源问题,故假设每个 T_i 在任意一个 PN 上的执行时间是相等的,都为一随机过程 $\{C_i(k)\}$,即 T_i 在第 k 帧内的执行时间为 $C_i(k)$,它仅在第 k 周期结束时才成为已知,并且相邻或相近帧上的 $C_i(k)$ 之间具有相关性($i=1,2,\ldots,J$).

Ⅱ类模型—— 系统以 T 为周期,每个帧需要完成 J 个相互独立的任务 $T_1,T_2,\ldots,T_J$,其中每个 T_i 又包含 $n_i(k)$ 个子任务 $t_{i_1},t_{i_2},\ldots,t_{in_i(k)}$($i=1,2,\ldots,J$),这里 $n_i(k)$ 是第 k 帧内 T_i 中的子任务数,它在一定范围内随机变化,仅在第 k 周期结束时才成为已知,故 $\{n_i(k)\}$ 为一随机过程,且相邻或相近帧内的 $n_i(k)$ 具有相关性. 同样地,由于不考虑通信及其他资源问题,我们假设每个 T_i 中的所有子任务在任意一个 PN 上的执行时间都是相同的,设为 D_i($i=1,2,\ldots,J$).

2 静态算法 SAA

在分布式系统中进行任务分配调度的主要准则是:每一帧内系统的任务完成时间以及对外部事件的总响应时间应尽可能取最小值. 由于任务完成时间取决于系统中最后完成所有任务的 PN 的执行时间,所以使其取最小的目标可以通过任务在各个 PN 上的合理分配来实现. 另一方面,我们可以将每一任务的完成视为对外部事件的一次响应,定义各个任务的等待时间之和为系统的总响应时间,当任务在系统中的分配确定以后,各个 PN 上进行合理调度的主要目的便是减小响应时间. 因此,分布式实时系统中周期性任务的分配调度分为两个步骤:首先是确定每一帧内任务在各个 PN 上的分配策略,其次是确定各个 PN 对分配在其上的各个任务的调度策略.

如果在上述两类任务模型中排除随机因素,即Ⅰ类模型中任务的执行时间及Ⅱ类模型中各任务所含子任务数均为常数,所得到的就是分布式系统中最为简单的多任务情形,此时可以根据已知参数预先确定分配和调度策略,这就是静态的 SAA 算法. 下面,我们分别描述两类模型的 SAA 算法,它是我们引入 PAA 算法的基础.

2.1 基于Ⅰ类模型的静态算法 SAA(Ⅰ)

由于 N 和 J 均为常数,所以,对于Ⅰ类模型中的任务分配问题,我们总能通过一些方法(如穷举法)找到使系统取得最佳任务完成时间的分配方式,但这已被证明是一个 NP 完备问题[12],因而对于实时系统的动态调度没有借鉴意义. 这里介绍一种易于实现的近似分配算法.

不失一般性,假设任务完成时间序列 $\{C_i\}$ 是单调递减的,这样,近似分配算法可以利用数学归纳法来描述:(i) 将任务 T_1 分配到 PN_1 上;(ii) 设任务 $T_1,\ldots,T_{i-1}$ 均已分配到各个 PN 上,并且此时系统中任务负载最小,且序号最小的节点是 PN_m;(iii) 将任务 T_i 分配到 PN_m 上,$i=2,3,\ldots,J$.

此近似算法不是分配问题的最佳解法,但是它具有以下性质[12]:(1) 当 $J\leqslant 2N$ 时,近似算法给出的就是最佳分配;(2) 在一般情况下,如果最佳算法得到的系统完成时间为 b^*,近似算法得到的系统完成时间为 b,则 $(b-b^*)/b^*\leqslant\frac{1}{3}-\frac{1}{3N}$;(3) 此近似算法的复杂度为 $O(J\log_2 J)$.

由于存在上述特性,此算法已经在工程实践中得到充分应用.在SAA算法中,我们即利用它来实现多任务的分配.

对于各个PN上的任务调度问题,我们首先定义一个任务在一帧中的响应时间为从本帧开始到任务完成时的时间片,一个PN的任务响应时间即为其上所有任务的响应时间之和.显然,当分布式系统中所有任务均被分配到各个PN以后,如果每个PN都得到最小响应时间,则整个系统也获得最小响应时间.由于我们假设各个任务都是独立的,因此它们之间没有预先设置的优先级限制,我们确定如下调度原则:设T_i和T_j是分配于同一PN的两个任务,则(i)当$C_i<C_j$时,任务T_i的优先级高于T_j的优先级;(ii)当$C_i=C_j$且$i<j$时,任务T_i的优先级高于T_j的优先级.

显然,根据排队原则,这样的调度准则能够使各个PN达到最小响应时间.

2.2 基于Ⅱ类模型的静态算法SAA(Ⅱ)

在考虑Ⅱ类模型的分配问题时,我们仍然可以假设序列$\{D_i\}$是单调递减的,由于各个任务中的子任务均可单独分配,故可将它们视为Ⅰ类模型中的任务,此时,运用对于Ⅰ类模型的SAA算法,即可以获得优化的近似算法.但对于每个PN内部的调度问题,我们却不能直接应用Ⅰ类模型的方法,这是因为对Ⅰ类模型而言,系统的总响应时间并不直接取决于PN上各个子任务的响应时间,而是由各个任务的响应时间决定,因此必须采用下面的调度策略:

对于任务T_i,我们假设其中分配到各个PN上的子任务数分别为$m_{i1},m_{i2},\ldots,m_{iN}$,于是,令

$$E_i=D_i\max_{1\le j\le N}m_{ij},\quad i=1,2,\ldots,J,$$

不失一般性,仍然假设$\{E_i\}$为递减序列.这样,对于任意一个PN,上的任意两子任务t_{il}和t_{jm},我们有如下调度准则:(i)当$i<j$时,t_{il}的优先级高于t_{jm};(ii)当$i=j$且$l<m$时,t_{il}的优先级高于t_{jm}.

显然,这样的准则能够使完成时间相对较短的任务获得相对较高的优先级.根据排队原则,在此情形下系统能够达到最佳响应时间.

3 预测算法PAA

3.1 滤波-预测方程

在以下讨论中,"参量"一词对于Ⅰ类模型而言就是任务的执行时间,对于Ⅱ类模型而言就是任务中的子任务数.当这两类模型中出现随机因素时,若要采用SAA算法,则我们只能以随机值的上限来替代算法中需要确定的参量,采用这样的方法,往往由于上限值在很多帧中可能远大于实际值,因而根据其确定的分配调度方案远非最佳方案.为了保证所有任务都能在1帧内完成,不得不要求各个PN具有比实际所需高很多的处理能力.同时,由于分配策略的限制,即使系统工作在不饱和状态,也难以进行任务的进一步扩充.此外,如果PN的处理能力受到限制,必须允许一部分任务在下一帧开始时中断执行,这样也会由于任务分配的不均衡而导致过高的夭折率.鉴于这方面的问题,有必要寻求更加理想的分配调度算法.对此我们注意到任务模型随机性的起因主要包括以下两个部分.

(1)稳定性的影响.外部环境系统的稳定变化会导致计算需求的变化,以雷达系统为例,面向自由空间与面向复杂背景,其计算上的需求将相去甚远,然而这样的变化是稳定而有规律的,因而在一定程度上是可以预知的.

(2)随机性的影响.偶然因素的存在常常会导致外部输入的摄动,这样的摄动便造成计算需求的变化,仍以雷达为例,系统内部噪声及外部点状杂波出现都会增加计算需求,并且这样的摄动完全是随机的、白色化的,所以是无法预知的.

可见,如果在系统工作过程中我们能够采用适当的方法,减小上述两类因素的影响,便能比较准确地估计出各帧的任务参量,获得理想的分配调度策略.在这一方面,滤波理论为我们提供了有力的手段.线性自回归滤波、维纳滤波、加权最小二乘滤波和卡尔曼滤波等方法能够适应多种类型的动态和噪声模型,提高系统的估计精度[13].尽管这些方法首先是针对运动学或动力学问题而提出的,但其应用早已超出这两方面的范畴,这里提出

的PAA算法即基于 α-β-γ 滤波方法,下面给出它的基本表达式:

状态方程 $$X(k+1)=\Phi X(k)+Gw(k),$$

其中 $X(k)$ 是所考虑参量在第 k 帧中的零阶、一阶及二阶导数构成的一维数组,$w(k)$ 为零均值的高斯白噪声序列,

$$\Phi=\begin{bmatrix}1 & T & T^2/2\\ 0 & 1 & T\\ 0 & 0 & 1\end{bmatrix},\qquad G=\begin{bmatrix}T^3/6\\ T^2/2\\ T\end{bmatrix}.$$

量测方程 $$y(k)=HX(k)+v(k),$$

其中 $y(k)$ 为所考虑参量在第 k 帧的实际采样值,$H(k)=[1\ 0\ 0]$,$v(k)$ 是均值为 0 的高斯量测噪声,其方差为 σ.

滤波-预测方程

$$\hat{X}(k/k)=\hat{X}(k/k-1)+K[y(k)-H\hat{X}(k/k-1)],$$
$$\hat{X}(k+1/K)=\Phi\hat{X}(k/k),$$
$$K=[\alpha,\beta/T,\gamma/T^2]^T.$$

这里,k 为帧序号,$\hat{X}(k/k)$ 为第 k 帧参量的平滑值,$\hat{X}(k/k-1)$ 为第 $k-1$ 帧完成时对第 k 帧参量的预测值.

从状态方程可知,这里考虑的主要是二阶变化模型,更高阶的分量以零均值的白噪声来表示.由于在很多系统模型中三阶以上的分量可以忽略,所以采用这样的滤波器可以满足现实世界中的大量应用需求.量测方程中的零均值高斯噪声表明了量测噪声的性质,其方差是选择滤波增益 α,β,γ 参数的主要依据.α 是介于 0 和 1 之间的小数,一般在 α 确定以后,可以根据最优关系式

$$\beta=2(2-\alpha)-4\sqrt{1-\alpha},$$
$$\gamma=\beta^2/\alpha$$

来确定参数 α 和 γ.由于 α-β-γ 滤波器的主要特点是,在滤波增益确定以后,实时过程中不需要再考虑系统的运动及量测模型,因此各帧中的计算负担很小,这使得我们将其应用于存在大量独立任务的分布式系统成为可能.

3.2 预测算法 PAA

一般地,分布式系统中的动态调度既可以是分布的,也可以是集中的.在分布式调度中,各 PN 均参加调度,当新任务到达某一个 PN 时,它可以根据自身的资源状态确定能否接纳该任务,在不能接纳时可将其推荐到相邻 PN 上.在集中式调度中,仅有一个 PN 进行任务管理,新任务首先进入此 PN,由其在确定合适的分配调度策略后将任务分配至各个 PN 上执行.PAA 算法采用集中调度机制,即系统中存在一个主控 PN,当一个工作周期中的各个任务完成时,该 PN 能够获得各任务的实际参量,据此进行对下一周期该任务参数的滤波预测,然后根据所有任务的预测参数进行任务的重新分配和调度.为简化模型,本文暂不考虑分布式系统中通信问题对实时性能的影响.

由滤波-预测方程可知,各帧的滤波预测依赖于前一帧的预测结果,故在系统开始运行时,我们必须根据先验知识以及最初几个帧内的实际参数获得所需的预测数据,所以 PAA 算法分为初始阶段和正常阶段两个部分.

(1) 初始阶段

在系统开始工作后的前 3 帧中,以任务参量的均值作为参量估值,通过 SAA 算法,实现所有任务在系统中的分配调度.同时,在此 3 帧中,我们可以得到实际的任务参量 $C(1)$,$C(2)$ 和 $C(3)$(如果在此 3 帧中出现任务夭折,则我们以任务参量的上限代替实际值),于是,通过多次两点外推法,可以得到第 4 帧滤波所需的预测参量:

$$\hat{C}''(4/3)=[C(1)+C(3)-2C(2)]/T^2,$$
$$C'(4/3)=[C(3)-C(1)]/(2T)+C''(4/3)T,$$
$$C(4/3)=C(3)+C'(4/3)T+C''(4/3)T^2/2.$$

(2) 正常阶段

在获得上述预测值以后,在从第 4 帧开始的每一帧中,我们利用所列的滤波-预测方程,都能够得到下一帧任务参量的预测估值.显然,对于不同的应用模型,采用这样的方法得到的估计精度是不一样的.此精度对一些

模型可以通过数学推导得到,也可以通过经验参数得到,或者在实际调度过程中通过实时统计获得其近似值.在此我们设利用 α-β-γ 方法的估计误差上限为 M,于是,

$$\hat{C}(k/k-1)+M$$

即为任务参量在第 k 帧中的估计上限值.之所以需要获得此估计上限值,是因为在处理资源足够的条件下,依据参量上限实现任务分配可以保证各个任务在下一帧开始时得以完成,这常常是实时系统中最为重要的准则.所以,获得各个任务的估计上限后,即可根据 SAA 算法的原则实现多任务的分配和调度.

由于滤波-预测方程只包含 6 个二次多项式的计算,所以 PAA 算法的复杂度仍然与 SAA 算法相当.

4 算法仿真与结果分析

SAA 算法是工程实践中得到充分应用的分配调度算法,PAA 算法是以其为基础进行的修正与改进.我们在多处理机模拟系统中实现了 SAA 和 PAA 算法,并针对 I 类模型和 II 类模型分别利用 Monte Carlo 方法进行了算法仿真.对于 I 类模型,仿真基于 10 个 PN 上的 40 个周期性应用任务;对于 II 类模型,仿真基于 10 个 PN 上的 10 个周期性任务,每个任务参量均以基值、正弦波和随机噪声三者组合的形式变化,各个参量的基值、振幅、周期和初始相位均随机选取,仿真过程中我们关注的主要目标函数是有以下几个.

完成时间系数——设在第 n 帧中最后完成所有任务的 PN 执行时间为 P_n,则我们定义

$$KB_n=P_n/T$$

为第 n 帧内的完成时间系数.显然,其值愈小,系统的潜力愈大.

负载均衡系数——设在第 n 帧中首先完成所有任务的 PN 执行时间为 Q_n,则我们定义

$$KB_n=(P_n-Q_n)/T$$

为第 n 帧内的负载均衡系数.直观上,KP_n 与 KB_n 具有相近的意义,我们将通过算法仿真对此进行验证.

响应时间系数——设在第 n 个周期中任务 T_i 的完成时间为 $W_n(i)$,则我们定义

$$KR_n=\sum_{i=1}^{J}W_n(i)\Big/(JNT)$$

为第 n 帧内的响应时间系数.系统调度的一个重要原则就是获得最小的 KR_n.

任务夭折率——由于运算能力的限制,多处理机系统往往并不总能保证所有任务均能在截止时间之前完成,特别是当所有任务均出现最坏情况时,某些任务可能在工作周期结束时夭折.我们定义第 n 帧中的夭折任务数与总任务数之比为任务夭折率 KD_n.

下面的一系列图形列出了主要的仿真结果.

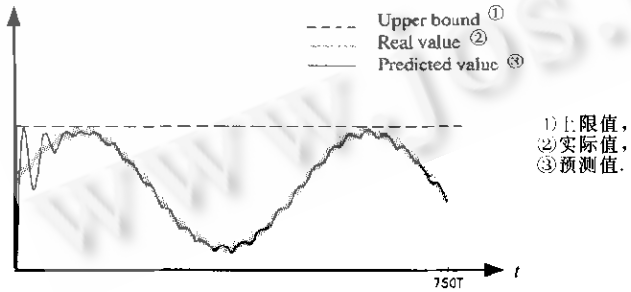

Fig. 2 Parameter value and predicting results of application tasks

图 2 应用任务参量实际值和预测效果

图 2 描绘了系统开始工作后的 750 帧内某一任务参量的实际值、预测值和任务上限值,由于假设任务参量按照正弦规律变化,所以参量实际值在一正弦曲线附近摄动,摄动的范围与方差的大小成正比,此时若采用参量上限来进行任务的分配和调度显然不尽合理.从预测效果来看,由于系统从初态进入稳态需要经过一段时间的积累,在开始数十帧内,预测值与实际值之间有较大的误差.进入稳态以后,预测曲线与实际曲线基本吻合,因为我们所考虑的是长时间工作的系统,因而系统初态出现的误差基本可以忽略,故这样的预测效果可以满足实际调度的要求.由于采用二阶预测方程作用于正弦变化的曲线仍能获得比较理想的结果,可以预见,当任务参量变化的复杂度不是很高时,这样的预测仍可以保持良好的效果.

图 3~10 分别是针对 I 类模型和 II 类模型进行 750 帧仿真所得到的任务完成时间、负载均衡度、响应时间系数和任务夭折率曲线,多次反复仿真的结果与图中所示基本相同.可见,PAA 算法与 SAA 算法在各项指标上均

有显著改善.值得指出的是,测试任务夭折率时所设的调度周期小于测试其他3类指标所使用的周期,这样,运行过程中会出现一部分任务的夭折.对于I类模型,SAA算法的夭折率维持在40%左右,PAA算法则由于初态过程中预测误差较大,因而任务夭折率也较高,但当系统进入稳态后,预测误差达到很低的水平,此时任务夭折率基本为0.对于II类模型,SAA算法的任务夭折率介于25%～35%之间,PAA算法的结果则低于20%,比SAA降低了40%左右.之所以不能降低至零值附近,主要是因为子任务数量的预测也会出现一定误差,当实际值高于预测值时,就会出现高出部分的任务夭折.这个问题可以通过充分运用预测误差上限参数,使算法具有一定的容差能力来解决.

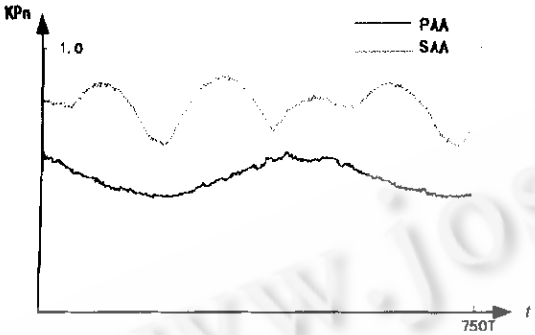

Fig. 3 Task run time for type I
图3 I类模型任务完成时间

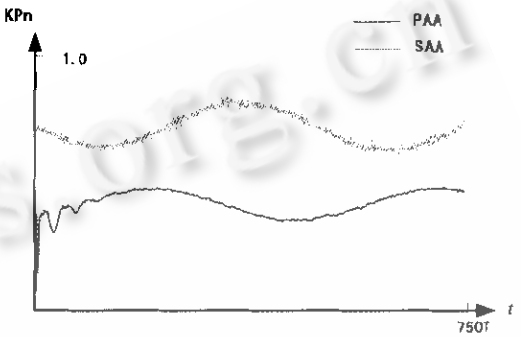

Fig. 4 Task run time for type II
图4 II类模型任务完成时间

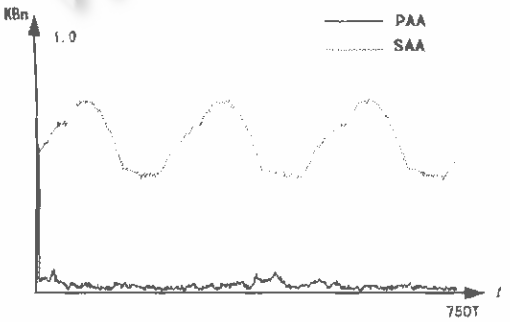

Fig. 5 Performance of load balancing for type I
图5 I类模型负载均衡度

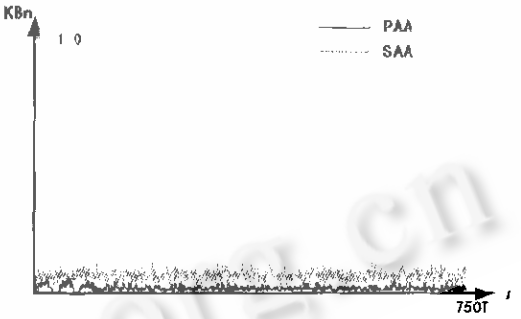

Fig. 6 Performance of load balancing for type II
图6 II类模型负载均衡度

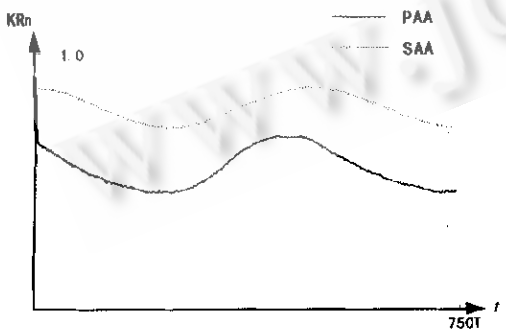

Fig. 7 Task response time for type I
图7 I类模型任务响应时间

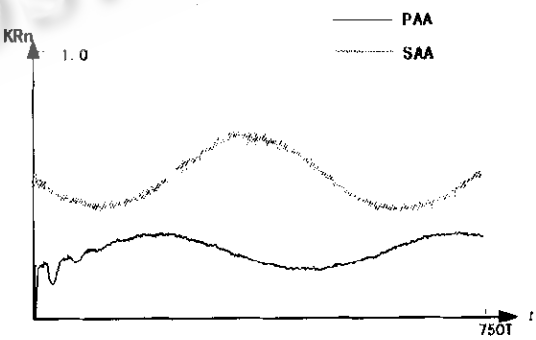

Fig. 8 Task response time for type II
图8 II类模型任务响应时间

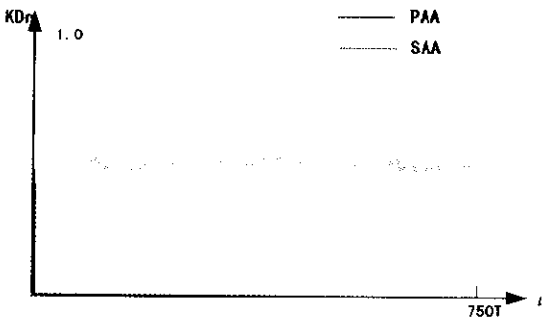

Fig. 9　Task time out ratio for type I
图 9　I类模型任务夭折率

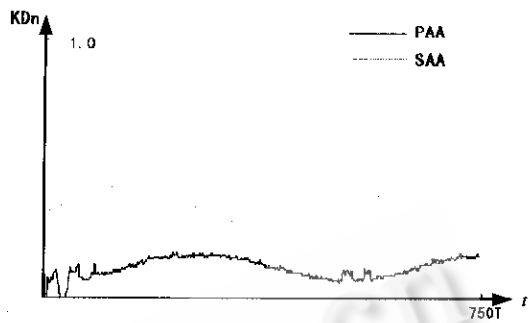

Fig. 10　Task time out ratio for type II
图 10　II类模型任务夭折率

5　结束语

分布式实时系统中很多未知因素会影响到系统的性能,如果能够比较准确地对这些未知量进行预测估计,便可以有效地进行任务分配和调度.我们提出的算法便是这方面的一个尝试,初步的仿真结果说明,采用这样的方法可以得到令人满意的效果.由于篇幅的限制,本文尚留有不少问题值得进一步研究,例如,任务模型与滤波参数之间的关系、初始状态下的分配调度准则等.特别需要指出的是,本文仅考虑了任务参量对多任务分配调度的影响,尚未涉及到系统实现中的另一重要因素,即任务间的通信问题.事实上,很多分布式实时系统中的通信周期、信息量及时延等参量均可以描述为一组随机过程,我们也能够对其建立合理的预测估计机制,从而在分配调度算法中综合考虑任务参量和通信参量,这样的算法必定更为完善,更具应用前景.

参考文献

1　Garey M R, Johnson D S. Strong NP—completeness results: motivation, examples, and implications. Journal of the Association for Computing Machinery, 1978,25(3):499～508

2　Peng D—P, Shin K G. Modeling of concurrent task execution in a distributed system for real-time control. IEEE Transactions on Computers, 1987,36(4):500～516

3　Ramamritham K. Allocation and scheduling of precedence-related periodic tasks. IEEE Transactions on Parallel and Distributed Systems, 1995,6(4):412～420

4　C-J, Shin K G. Allocation of periodic task modules with precedence and deadline constraints in distributed real-time systems. IEEE Transactions on Computers, 1997,46(12):1338～1355

5　Houstis C E. Module allocation of real-time applications for distributed systems. IEEE Transactions on Software Engineering, 1990,16(7):699～709

6　Tantawi A N, Towsley D. Optimal static load balancing in distributed computer systems. Journal of the ACM, 1985,(32):445～465

7　Chou T C K, Abraham J A. Load balancing in distributed systems. IEEE Transactions on Software Engineering, 1982,8(7):401～422

8　Shen C C, Tsai W H. A graph matching approach to optimal task assignment in distributed computing systems using a minimax criterion. IEEE Transactions on Computers, 1985,34(3):197～203

9　Chu W W, Lan L M T. Task allocation and precedence relations for distributed real-time systems. IEEE Transactions on Computers, 1987,36(6):667～679

10　Chu C C, Leung K K. Module replication and assignment for real-time distributed processing systems. Proceedings of the IEEE, 1987,(75):547～562

11　Tzilla Elrad, Lin Jin-long. Evolving processes and evolution schedulers for concurrent scheduling controls and parallel evolutionary computation. In: Rolim J ed. Proceedings of the 12th International Parallel Processing Symposium and the 9th Symposium on Parallel and Distributed Processing. Berlin: Springer-Verlag, 1998. 270～278

12 Lu Kai-cheng. Combinatorial Mathematics Algorithm and Analysis. Beijing: Tsinghua University Press, 1983
(卢开澄.组合数学算法与分析.北京:清华大学出版社,1983)
13 Zhou Hong-ren, Jing Zhong-liang, Wang Pei-de. Tracking of Maneuvering Targets. Beijing: National Defense Industry Press, 1991
(周宏仁,敬忠良,王培德.机动目标跟踪.北京:国防工业出版社,1991)

Predicting Allocation Algorithm in Distributed Real-Time Systems

XU Jian-feng ZHU Qing-bo HU Ning XIE Li

(*State Key Laboratory for Novel Software Technology Nanjing University Nanjing* 210093)

Abstract For periodic tasks in a distributed real-timesystem, a number of static allocation algorithms have been developed which solve the problem of assigning and scheduling tasks effectively under some determined conditions. The principal limitation of these approaches is that the attributes of the tasks have to be known. Sometimes the execution time or the number of subtasks of a periodic task might be a stochastic process obeying some rule. In such cases, the performance of the static schemes will decrease greatly. According to the analysis of the processing in specific application fields, the authors model two types of random tasks in distributed real-time systems and introduce the static allocation algorithms (SAA) which have been applied in engineering for the two task models separately. On the basis of SAA, a predicting allocation algorithm (PAA) is presented for the assignment and the scheduling of multitasks in distributed systems. The proposed algorithm, depending on the statistic features of the task execution time or of the number of subtasks included in the tasks, can predict the task parameters reasonably and implement dynamic allocation of the tasks, so that the system can meet the timing requirements more efficiently. The results of the simulation of the two task models have shown that compared with SAA scheme, the performance of PAA is significantly better in task finishing time, load balancing, system response time, ratio of discarded tasks, etc.

Key words Distributed real-time system, periodic task, allocation algorithm, predicting