基于非抢占 RMS的分布式控制系统中实时任务调度算法

刘 怀, 黄建新, 史国生

(南京师范大学 电气与自动化工程学院, 江苏 南京 210042)

[摘要] 分布式控制系统是一种应用极为广泛的分布式实时系统, 如何将系统中的任务分配到各个处理器上并保证它们的时限是系统关键技术之一. 对于系统中每一个处理器上的任务采用非抢占 RMS算法调度, 该算法是一种最优的静态任务调度算法, 在单处理器调度算法的基础上, 结合启发式任务分配算法, 提出了一种分布式控制系统的调度算法. 该算法是一种静态任务分配算法, 算法的开销小、网络负载量低; 同时在任务分配时考虑了各个处理器的负载均衡. 仿真结果表明了算法的有效性.

[关键词] 分布式控制系统, 异构系统, 启发式, 调度算法

[中图分类号] TP316 [文献标识码] B [文章编号] 1672-1292(2005)02-0010-04

Scheduling Algorithm for Real-time Tasks in Distributed Control System based on RMS

LIU Huai, HUANG Jianxin, SHI Guosheng

(School of Electrical and Automation Engineering, Nanjing Normal University, Jiangsu Nanjing 210042, China)

Abstract Distributed control system (DCS) is one kind of the widely used distributed real-time systems. How to assign tasks of system to processors and guarantee their deadlines is one of the key techniques in DCS. For every processor in DCS, RMS algorithm is an optimal static tasks of scheduling algorithms. Based on the combination of heuristic algorithm for assigning tasks and the scheduling algorithm for uniprocessor, the paper proposes a novel scheduling algorithm for DCS. The algorithm is a static algorithm for assigning tasks and its offset is small and it lowers the load of network. At the same time, the algorithm can equilibrate the load of every processor. The results of simulation show that the algorithm is effective.

Key words distributed control system, heterogeneous system, heuristic, scheduling algorithm

0 引言

随着工业生产的发展、科学技术的进步和各种系统复杂性的提高, 简单的控制系统很难满足实际应用的要求, 因此分布式控制系统已越来越多地应用于各种实际控制领域, 如工业控制系统、武器防御控制系统、飞行控制系统及电站控制系统等. 然而在一个控制系统中同时存在有多种实时任务[1], 如何确定这些任务何时在何处理器上执行就显得尤为重要. Mok等人[2]指出, 对于实时多处理器系统, 并不存在最优的任务调度算法. 这使得我们只能采用启发式调度算法来解决这类问题.

目前, 实时分布式系统的调度算法多是以启发式搜索算法为基础, 如近似算法[3]、节约算法[2]、集中式任务调度方案[4]等, 均是在系统运行过程, 当某个任务实例到达后, 按照一定的指标搜索可执行该任务实例的处理器. 这种算法存在两个问题, 一是算法的开销较大, 二是系统中信息交换频繁. 另外, 这些算法没有结合单一处理器的调度算法, 而多数是设计了集中任务调度器[2,4], 如果任务调度器出现故障, 整个系统将可能瘫痪. 基于此, 本文针对存在有多种任务的分布式控制系统设计了一种静态调度算法, 算法的开销小, 并与单处理器的调度算法相结合.

收稿日期: 2004-05-27.
基金项目: 南京师范大学科研基金资助项目(2003KZXXGQ2B88)和南京师范大学青年基金资助项目(2004111XQNBQ41).
作者简介: 刘怀(1971-), 博士后, 副教授, 主要从事实时控制系统和综合自动化系统等方面的教学与研究. E-mail: liuhua@njnu.edu.cn

1 系统模型

典型的控制系统如图 1所示[5], 系统以一个固定的频率通过 A/D转换器从物理过程获得测量数据, 经控制器进行控制计算得出控制量, 再通过 D/A转换器将控制信号送到被控对象. 它需要完成数据采集、控制计算、状态更新、控制输出、报警、人机交互、显示更新等功能. 这些任务中有些是周期性的, 如数据采集、控制计算、状态更新、控制输出等; 有些是非周期的, 如报警、人机交互等. 显然图 1所表示的只是控制系统中的一个控制回路, 分布式控制系统需要在多个处理器上实现多个控制回路. 当控制系统设计完成后, 每一个周期性任务的执行时间和周期都是确定的. 这类任务是处理器所要处理的主要任务, 本文主要研究这类任务在异构分布式系统中的调度问题.

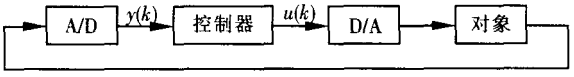


图 1 控制系统结构图

另外, 本文中控制系统中的控制器指的是数字控制器, 如单片机、PLC、IPC及各种集散控制系统中的控制单元等. 在同一个分布式控制系统中, 数字式控制器可能包含其中的一种或几种; 即使是一种数字控制器, 如果型号不同, 它们的性能也会不同; 且对于控制系统中不同的部分可能会采用不同的控制器, 所以控制系统是一个异构分布式系统.

基于上述分析我们给出分布式控制系统中的任务模型和处理器模型:

定义 **1** 集合 $S_p = \{\tau_{p1}, \tau_{p2}, \cdots \tau_{pn}\}$, 其中 $\tau_{pi} \in S_p$ 是一个四元组, $\tau_{pi} = (C_i, P_i, D_i, Pr)$, C_i、P_i、D_i 和 Pr分别为任务 τ_{pi}的执行时间、周期、时限和所分配的处理器.

定义 **2** 异构分布式控制系统被描述为一个处理机的集合: $\Omega = \{Pr_1, Pr_2, \cdots, Pr_k\}$, $k \geqslant 2$, $Pr_i = (\rho_i, u_i)$, 其中 ρ_i和 u_i分别表示处理器 P_{ri}的性能和分配到该处理器上的任务的最大可能利用率.

显然, 对于一个实时任务来说, 在异构分布式控制系统中不同的处理器上的执行时间不同, 为此我们引入两个执行时间向量如下:

定义 **3** 在异构分布式控制系统中, 同一个任务在不同的处理器上的执行时间不同, 定义 $C_i = [C(i, 1), C(i, 2), \cdots, C(i, k)]$为任务 τ_{pi}的执行时间向量, 其中 $C(i, j)$为任务 τ_{pi}在处理器 Pr_j上的执行时间.

定义 **4** 处理器的处理能力表示处理器执行任务速度快慢, 处理能力越高任务在该处理器上执行越快. 在一个异构分布式系统中, 将一个处理器的处理能力作为标准, 其处理能力定为 1, 其他处理器的处理能力指一个任务在标准处理器上执行时间与该任务在该处理器上执行时间的比值, 即处理器 P_{ri}的处理能力为

$$\rho_i = \frac{C(j, \mathrm{normal})}{C(j, i)} \tag{1}$$

式中, $C(j, \mathrm{normal})$为任务 τ_{pj} 在标准处理器上的执行时间.

由定义 3和定义 4可得如下关系

$$\frac{\rho_q}{\rho_j} = \frac{C(i, j)}{C(i, q)} \tag{2}$$

即处理器的处理能力与任务在该处理器上的执行时间成反比.

定义 **5** 任务是可调度的是指在任何情况下该任务的实例都能满足其时限, 任务集是可调度的是指任务集中所有任务都是可调度的.

为了设计和分析调度算法的方便, 我们做如下假设:

(1) 所有任务之间相互独立;

(2) 周期性任务的时限等于周期;

(3) 处理器之间相互独立.

2 调度算法

分布式系统的调度算法是要确定任务何时在何处理器上执行. 整个调度算法分为两个部分: 任务分配算法和处理器上的任务调度算法. 任务分配算法用于确定执行该任务的处理器, 而处理器上的任务调度算法决定的任务执行时间. 下面针对控制系统的特点, 设计处理器上任务的调度算法和任务分配算法.

2.1 处理器上的任务调度算法

处理器上的任务调度算法是指处理器用于调度分配到其上任务的算法. 单处理器的调度算法研究得比较成熟, 如 RMS、EDF、DMS等[6,7]. 由于 RMS是一种最优的静态调度算法, 其开销比较小, 且易于处理任务之间一定的约束关系, 因此本文采用非抢占 RMS调度算法, 即每一个处理器上的任务采用 RMS设置任务的优先级, 而高优先级任务不能抢占正在运行的低优先级任务.

定理 设分配到处理器 P_{rj}上的任务集为 $S_{pj} = \{\tau_{p1}, \tau_{p2}, \cdots \tau_{pn_j}\}$, 在采用 RMS调度算法时, 任务集 S_{pj}可调度的充分条件是

$$\sum_{q\leqslant n_j}\frac{C(q,\text{normal})}{P_q}+\frac{C_{\max}}{P_{\min}}\leqslant\theta_j n_j(2^{1/n_j}-1). \tag{3}$$

其中 $C_{\max}=\max_{i<q\leqslant n_j}\left(\frac{C(q,\text{normal})}{\theta_j}\right)$,

$P_{\min}=\min_{i<q\leqslant n_j}(P_q)$.

证明　根据 Baker[7] 的研究可知, 对于处理器 P_j上任务可调度的条件为:

$$\sum_{q\leqslant i}\frac{C(q,j)}{P_q}+\frac{B_i}{P_i}\leqslant i(2^{1/i}-1),\quad\forall i,\ 1\leqslant i\leqslant n_j$$

式中, B_i为比任务 τ_{pi}优先级低的阻塞时间, $B_i=\max_{i<q\leqslant n_j}(C(q,j))$. 根据定义 4可得

$C(q,j)=\frac{C(q,\text{normal})}{\theta_j}$, 带入上式可得:

$$\sum_{q\leqslant i}\frac{C(q,\text{normal})}{\theta_j P_q}+\frac{\max_{i<q\leqslant n_j}(C(q,\text{normal}))}{\theta_j P_i}\leqslant i(2^{1/i}-1),\quad\forall i,\ 1\leqslant i\leqslant n_j,$$

令 $C_{\max}=\max_{i<q\leqslant n_j}\left(\frac{C(q,\text{normal})}{\theta_j}\right)$, 则:

$$\sum_{q\leqslant i}\frac{C(q,\text{normal})}{P_q}+\frac{C_{\max}}{P_i}\leqslant\theta_j i(2^{1/i}-1), \tag{4}$$

$\forall i,\ 1\leqslant i\leqslant n_j$,

如果任务满足下式:

$$\sum_{q\leqslant i}\frac{C(q,\text{normal})}{P_q}+\frac{C_{\max}}{P_{\min}}\leqslant\theta_j i(2^{1/i}-1), \tag{5}$$

$\forall i,\ 1\leqslant i\leqslant n_j$,

则任务必满足式(4), 式中 $P_{\min}=\min_{i<q\leqslant n_j}(P_q)$. 显然式不等号左边是 i的增函数, 而右边是 i的减函数(证明略). 所以只要当 $i=n_j$时满足式(5), 对于任意 $1\leqslant i\leqslant n_j$的 i都满足式(5). 证毕.

2.2　任务的分配方法

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[9], 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)[8]方法, 并考虑各处理器负载尽可能平衡. 基于此给出任务分配算法:

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(包括周期性任务和非周期任务)的参数; 处理器集合 Ω, 初始化各处理器的参数 $Pr_j.u_j=0$

(2) 将任务 $\tau_{pi}(1\leqslant i\leqslant n)$按如下步骤分配到处理器集 Ω中的处理器上.

(3) 如果存在处理器 Pr_j满足 $Pr_j.u_j=\min_{Pr_q\in\Omega}\{Pr_q.u_j\}$, 并令 $Pr_j.u_j=Pr_j.u_j+\frac{C_p(i,\text{normal})}{\theta_j P_i}$, $n_j=n_j+1$, 若该处理器 $Pr_j\cdot u_j+\frac{C_{\max}}{P_{\min}}>\theta_j n_j(2^{1/n_j}-1)$, 则 Sch_Success=失败, 转到(5), 否则将任务 τ_{pi}分配到处理器 Pr_j, 当 $i<n$时, 令 $i=i+1$并重复(3), 否则转到(4).

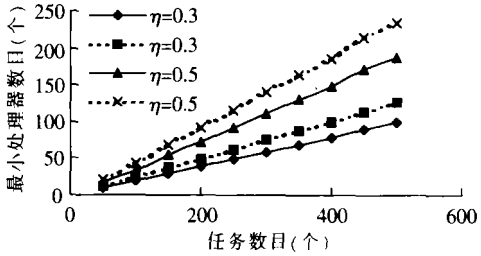
(4) 所有任务分配完成, Sch_Success=成功.

(5) 算法结束.

说明　此任务分配算法是静态算法, 它在系统运行前将周期性任务和非周期任务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 RMS算法动态设置. 因此算法的开销比较小, 同时系统中信息交换不太频繁, 各处理器的自治能力较强.

3　计算机模拟

为验证算法的有效性, 选择多组包含有 N个周期性任务的实时任务集, 按照本文算法进行调度. 我们将处理能力最小的处理器定为标准处理器, 其他处理器的处理能力满足[1, 1.5]上均匀分布. 设周期性任务的周期都服从[100, 500]ms上的均匀分布, 任务在标准处理器上的执行时间为在区间$(0,\eta P_{pi})$上均匀分布两种情况, η取为 0.3和 0.5两种情况, 求任务所需的最小处理数目(实际处理器数目(n_r)和折算为标准处理器数目(n_n)), 同一组数据重复 10次取均值, 结果如图 2所示; 求得任务的总负荷 $U=\sum_{i=1}^{n}\frac{C(i,\text{normal})}{P_i}$和处理器数目 M(折合成标准处理器数目)比值, 同样同一组数据重复 10次取均值, 所得结果如图 3所示.



虚线为折合成标准处理器的数目 n_n; 实线为实际分配的处理器数目 n_r

图 2　处理器数目与任务个数之间的关系

由图 2可以看出, 最小处理器数目随任务数的增加而增加, 因为任务数增加就需要更多的处理器来保证所有任务在其时限前完成. 在任务数固定的情况下, 任务执行越长系统所需要承担的工作量就

越大, 就需要更多的处理器在任务时限前完成这些工作量. 同时实际分配的最小处理器数目小于折合成标准处理器的数目, 这是因为, 除标准处理器外, 其他处理器的处理能力都大于等于 1, 使得实际处理器数目需要量减少.

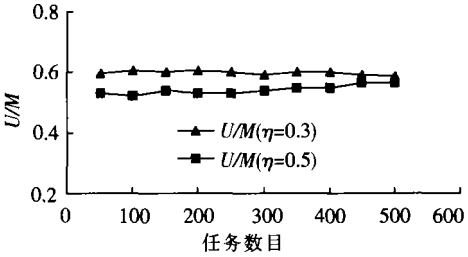


图 3 U/M 与任务数目之间的关系

由图 3可以看出, 处理器的 U /M 与任务的多少基本上没有关系, 只与任务的执行时间和任务周期的比值有关系, 即与任务的利用率有关, 任务本身的利用率越大, U /M 越小, 说明处理器的利用率越低, 这是因为为了满足任务的可调度性条件, 将任务分配到处理器上时, 达不到处理器在 RMS调度下的极限利用率.

从仿真结果可以看出, 本文给出的分布式控制系统调度算法是有效的.

4 结论

分布式控制系统是一种异构实时系统, 其上的周期性任务是处理器所要处理的主要任务, 它们的执行情况决定着系统的性能. 本文在分析分布式控制系统中周期性任务特性的基础上, 设计了任务调度算法, 将任务分配算法和单处理器的调度算法结合起来. 其中单个处理器上的任务调度算法采用了非抢占 RMS调度算法, 而任务分配算法采用了首次适用的启发式调度算法. 这种算法的开销小, 系统中信息交换少, 仿真结果表明了任务的有效性.

[参考文献]

[1] 刘怀, 朱广宇, 费树岷. 控制系统中实时任务的多优先级带宽调度算法[J]. 控制与决策, 2002, 17(2): 212-214.

[2] 乔颖, 王宏安, 戴国忠. 一种新的实时多处理器系统的动态调度算法[J]. 软件学报, 2002, 13(1): 51-58.

[3] Ramamritham K, Stankovic A, Shiah P F. Efficient scheduling algorithms for real-time multiprocessor systems [J]. IEEE Transactions on Parallel and Distributed Systems, 1990, 1(2): 184-194.

[4] 王堃, 乔颖, 王宏安, 等. 实时异构系统的动态调度算法研究[J]. 计算机研究与发展, 2002, 39(6): 725-732.

[5] Cervin A. Improved scheduling of control tasks[A]. In Proceedings of the 11th Euromicro Conference on Real-Time Systems[C]. York: IEEE Computer Society Press, 1999. 4-10.

[6] Liu C L, Layland J W. Scheduling algorithms for multiprogrammming in hard real-time environment[J]. Journal of Association for Computing Machinery, 1973, 20(1): 46-61.

[7] Baker T. Stack-based scheduling of priority real-time processes[J]. Real-Time Systems, 1991, 3(1): 67-79.

[8] 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.

[9] 秦啸, 韩宗芬, 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.

[责任编辑: 严海琳]