doi:10.3772/j.issn.1002-0470.2013.09.006

嵌入式系统流水线资源管理模型①

林 军②*** 倪 宏③** 孙 鹏** 张 辉**

(*中国科学院研究生院 北京 100190)

(**中国科学院声学研究所国家网络新媒体工程技术研究中心 北京 100190)

摘 要 针对嵌入式系统多任务多资源分配问题,提出了一种采用数据流水线的资源管理模型和基于模糊控制规则的自适应作业调度算法。该模型建立一组资源管理服务节点,每个服务节点管理一种类型资源,不同服务节点以流水线形式顺序处理任务作业。当多个不同任务的作业进入同一服务节点,该节点执行自适应作业调度算法进行多任务资源分配。该算法以保证不同任务作业实时性为目标,采用当前任务队列长度为输入,基于模糊控制规则自适应调整任务队列带宽。实验结果表明该模型使任务作业处理速度得到约1.4 倍流水线加速提升,并能自适应分配资源,确保多任务作业实时性。

关键词 流水线,资源管理,模糊控制,嵌入式系统,作业调度

0 引 言

嵌入式智能设备中存在多种同一时刻只能提供给一个任务使用的设备资源,系统如何管理分配这些资源以保证用户体验一直是研究热点。为满足智能系统用户的体验需求,本研究针对嵌入式系统多任务多种独占资源管理的问题进行了探索,提出了采用数据流水线的资源管理模型和基于模糊控制规则的自适应作业调度算法,并进行了资源管理模型和作业调度算法的仿真实验。

1 相关工作

按传统的多任务多资源管理设计,任务通过资源临界区访问设备资源,当一个任务试图获取被其他任务占用的资源时会被阻塞,从而不能确保任务实时性,于是研究人员提出了各种资源访问控制协议以解决这一问题。优先级继承协议(priority inheritance protocol,PIP)[1]通过优先级继承来解决高优先级任务所需资源被低优先级任务占用时的优先级倒置问题。优先级冲顶协议(priority ceiling protocol,PCP)[2]在解决优先级倒置基础上还能阻止死锁和链式阻塞的形成,保证任务至多被一个临界区阻塞。Baker 提出了栈资源协议(stack resource policy,SRP)[3],该协议对 PCP 进行了精炼,它为每个任务扩展一个抢占级,只有任务的抢占级高于其它任务时才能发生抢占。SRP 可用于多资源动态优先级调度。文献[4]在 SRP 的基础上实现了三种同步协议并配合多层次调度框架支持组件资源同步。Marko[5]等提出了最小化资源占有时间策略以取代传统的非抢占式资源共享方法。相对这些全局的资源访问控制方法,通过对每种类型资源进行独立管理并且配合作业调度算法进行资源分配,能避开这些需要复杂协议解决的资源共享问题。

多任务资源分配依赖于对任务作业的调度,常用的实时任务调度策略有 EDF 算法[6]、RM 算法[7]等,这些基于优先级的任务调度算法提供严格的硬实时调度,在系统过载时会产生低优先级任务饥饿和系统性能下降的问题,而在一些混合实时系统中,如果能够保证非实时和软实时任务执行的公平性[8],效果更好。文献[9]提出了一种混合系统的自适应调度算法,它基于模糊控制进行资源调整。比例共享调度算法[10]是一种基于 CPU 使用比例的

① 863 计划重点资助项目(2011AA01A102),国家科技支撑计划(2011BAH16B03)和中国科学院战略性先导科技专项(XDA06010302)资助项目。

② 男,1986 年生,博士生;研究方向:嵌入式操作系统;联系人,E-mail:linj@dsp.ac.cn

③ 通讯作者,E-mail:nih@dsp.ac.cn

(收稿日期:2012-09-07)

共享式调度算法,系统中的任务按照其权重比来分享 CPU 带宽。比例共享资源的特点会使得所有任务按比例变慢,通常采用动态权重调整来解决这一问题。文献[11]提出了基于流水线模型的软实时任务 QoS 保障调度方法,提出以实时调度策略为基础,保障任务 QoS 为目标的控制设计。文献[12]则进一步结合应用层次和资源层次的 QoS 需求提出双层反馈控制保证任务实时性和应用的服务质量。

本文结合智能操作系统多任务多资源分配场景,提出了采用数据流水线的资源管理模型以及采用模糊控制的自适应作业调度算法,以满足智能系统用户的体验需求。

2　流水线资源管理模型

2.1　任务集及相关概念

开放式系统包含 n 个任务,用任务集 $\{\tau^i \mid 1 \leqslant i \leqslant n\}$ 表示,任务 τ^i 由一系列的作业组成,第 k 个作业 J_k^i 到达时刻记为 r_k^i,完成时刻 f_k^i,执行时间 c_k^i,作业实际完成时间 $m_k^i = f_k^i - r_k^i$。若任务为实时任务,d_k^i 为作业绝对截止期,D^i 为相对截止期,$d_k^i = r_k^i + D^i$。若 $r_k^i = r_1^i + (k-1)T_i$,r_1^i 为初始相位,T_i 是它的周期,则 τ^i 为周期任务,否则为非周期任务。若 $f_k^i > d_k^i$,即 $m_k^i > D^i$,则 τ^i 错过截止时间。大多数多媒体应用为软实时任务,任务适当错过截止期不会造成重大的系统错误。

任务间存在资源约束,设共享资源集合为 $\{R_1, R_2, \cdots, R_r\}$,集合为广义的资源集,包括存储、通信、实际硬件资源及其他抽象资源。任务使用资源临界区来获取共享资源,资源在同一时刻只能提供给一个任务使用,称为独占资源。如果任务 τ^i 进入 R_k 临界区,则 τ^i 使用资源 R_k,不失一般性,将 R_k 表示 R_k 临界区。$\xi^i(R_k)$ 表示任务 τ^i 占用临界区 R_k 的执行时间。

从资源使用的角度,将作业 J_k^i 在时序上分成 r 个连续的子作业序列 $\{J_{k,1}^i, J_{k,2}^i, \cdots, J_{k,r}^i\}$,每个子作业只使用一种类型资源。

任务 τ^i 赋予权重系数 $\omega_i (\in \mathbf{R}_+)$ 表示任务的相对重要性,任务越重要,权重系数也就越大。权重系数的大小,可以通过多种方式确定,如可以直接使用任务的优先级,或者由用户在系统中根据自己的偏好设置。当存在多个权重因素时,权重系数可以使用层次分析法[13]等方法计算获得。

2.2　传统多任务多资源管理

如图 1 所示,传统设计中多任务互斥争夺独占资源。为简单起见,令 $r = 2$。任务 τ^i 和 τ^j 因为访问同一个临界区 R_k 造成处理延迟:若 $r_k^j < r_p^i$,则 τ^j 会被 τ^i 阻塞直到作业 J_p^i 释放资源。τ^j 至少需要等待 $\xi^i(R_k)$ 时间才能继续执行,当更多并发任务竞争临界区时,任务实时性不可保证。文献[3]采用 EDF 算法[6]和栈资源协议(SRP)结合的作业调度方式能保证一个任务至多被另一个任务阻塞,阻塞时间 $t \leqslant \xi^i(R_k)$。在非实时操作系统中,如标准 Linux 系统,没有机制保证任务的实时性。任务数目增多或者某个任务爆发大量短周期作业序列都会加重资源竞争,造成大量任务作业错过截止期,影响用户体验。

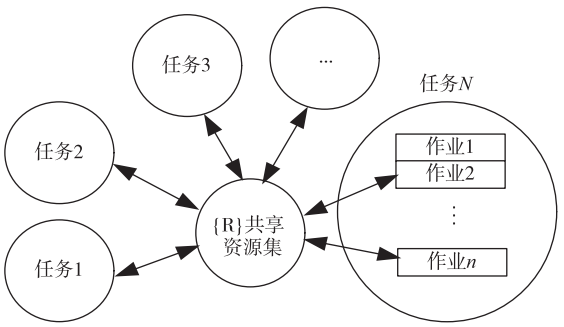

图 1　传统多任务资源管理

2.3　流水线资源管理模型

如图 2 所示,本文给出一种采用数据流水线[11]的资源管理模型。该模型为 r 个独占资源组建独立资源管理服务节点,节点间通过单向缓存两两连接,协作完成任务作业。对于 τ^i,作业 J_k^i 划分成 r 个子作业 $\{J_{k,p}^i \mid 1 \leqslant p \leqslant r\}$,子作业 $J_{k,p}^i$ 进入第 p 个节点

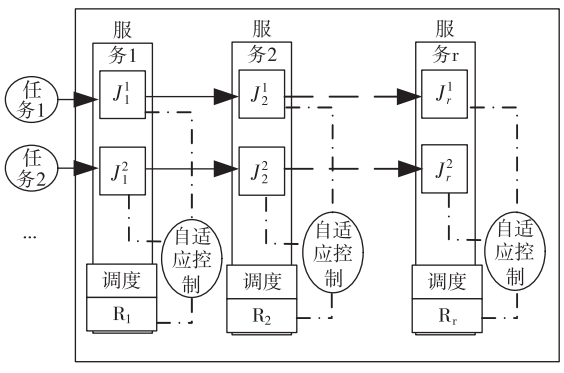

图 2　流水线资源管理模型

输入缓存,处理完毕后激活子作业 $J^i_{k,p+1}$ 以流水线形式进入第 $p+1$ 个节点。假设任务由有限的作业序列组成,作业个数 N_l,每个子作业的处理时间为 $\{\Delta t^i_{k,p} \mid 1 \leqslant p \leqslant r\}$,为简单起见,假设最后一个子作业处理时间 $\Delta t^i_{k,r}$ 最大。节点并行工作,单位时间完成的工作量增加,流水线吞吐率 $T_c = \frac{N_l}{(\sum_{p=1}^{r}\Delta t^i_{k,p} + (N_l-1)\max\{\Delta t^i_{k,p}\})}$。特别当作业序列长度 $N_l \geqslant r$ 时,流水线的最大吞吐率 $T_c^{\max} = \frac{1}{\max\{\Delta t^i_{k,p}\}} = 1/\Delta t^i_{k,r}$。完成同一个任务,不使用流水线和使用流水线完成时间比,称为加速比。r 段流水线模型的加速比 $S = \frac{N_l\sum_{p=1}^{r}\Delta t^i_{k,p}}{\sum_{p=1}^{r}\Delta t^i_{k,p} + (N_l-1)\max\{\Delta t^i_{k,p}\}}$。当 $N_l \geqslant r$ 时,$S \approx r$。所以理想状态,采用流水线模型,每个独立的任务被系统处理速度将提高 r 倍。在实际操作系统中,考虑到服务节点的执行依赖于操作系统对节点的进程调度,实际加速比与理论加速比会存在偏差。系统使用的存储资源也会增加,因为这是一个空间换时间的过程。

流水线资源管理模型同时改变了资源分配方式,使用同一种类型独占资源的多任务子作业串行进入资源管理服务节点,通过作业调度算法获得资源。模型阻止了任务链式阻塞,避免使用复杂的资源访问控制协议。

3 自适应作业调度算法

服务节点的作业调度需要解决混合任务的不可抢占调度问题,作业能够使用资源的时间本质上取决于作业分到的 CPU 带宽。本文采用一种基于模糊规则的自适应比例共享调度算法进行子作业调度。

3.1 作业调度模型

流水线资源管理模型将多资源的管理分散到各个资源管理服务节点,采用如 Cgroup[14] 的 Linux 资源隔离技术,节点可以接受不同的 CPU 带宽,其递增的取值集合为 $U = \{U_j \mid 0 \leqslant j \leqslant K\}$。第 i 个节点获得的带宽记为 $u_i \in U$。若操作系统分配给资源管理服务节点的总带宽不超过 C,则必有 $\sum_{i=1}^{r} u_i \leqslant C$。

考虑第 1 个服务节点。假设该节点同时服务 N_g 个任务队列,任务子作业以先进先出的方式进入各自队列。采用多级带宽预留的比例共享调度,任务队列 τ^i 分配的带宽记为 b_i,递增的离散带宽取值集合为 $B = \{B_j \mid 0 \leqslant j \leqslant M\}$,则 $b_i \in B, i \in [1, N_g]$ 且满足 $\sum_{i=1}^{N_g} b_i \leqslant U_K$。若子作业不可抢占的资源使用时间为 t,则作业实际完成时间 $m = t/b_i$。

当前 τ^i 队列长度记为 q_i,任务子作业资源使用时间的不确定性会引起 q_i 变化,进一步影响任务子作业的实际完成时间 m^i_k,其中 $m^i_k = f(q_i)$ 为单调递增函数。为了自适应保证任务的实时性,以队列的长度为输入基于模糊反馈调整任务队列的带宽。控制结构如图 3 所示,控制策略模块监视队列长度,并通过模糊控制提供队列带宽的参考输出;决策分配模块负责集中调整服务节点和任务队列带宽,有效利用系统资源。

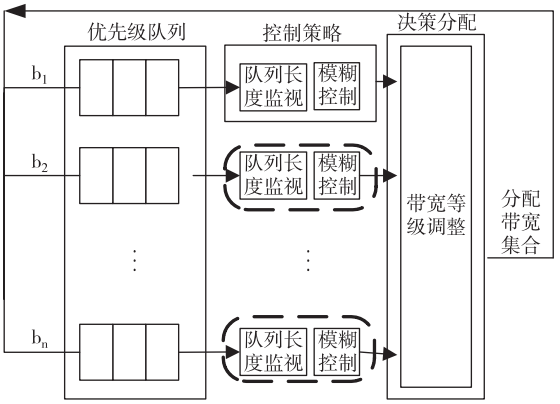

图 3 作业调度控制结构图

3.2 模糊控制规则

控制策略模块,输入变量为队列长度 q_i,输出为带宽参考输出 b_i,采样时刻为当前执行作业完成时刻。输入输出变量模糊化,用模糊集合表示。将 τ^i 队列长度区间 $[0, \infty)$ 划分成 $M+1$ 个子区间模糊集合 $\{Q_0, Q_1, \cdots, Q_M\}$,区间长度取自经验值。将 τ^i 队列参考输出依据集合 B 划分成 $M+1$ 个模糊集合 $\{\theta_0, \theta_1, \cdots, \theta_M\}$;将资源管理服务节点带宽依据集合 U 划分成 $K+1$ 个模糊集合 $\{S_0, S_1, \cdots, S_K\}$,隶属度函数选取三角函数,如图 4 所示。

基于队列长度模糊控制带宽输出规则描述为:IF $q_i$is$Q_j(j \in [0, M])$ THEN $b_i$is $\theta_j(j \in [0, M])$。

采用最大隶属度法对带宽输出模糊集合执行解模糊策略,取控制输出论域上使得隶属函数达到最大值的点作为解模糊的结果,该点即取值集合 B 中元素。因此生成控制规则如下:

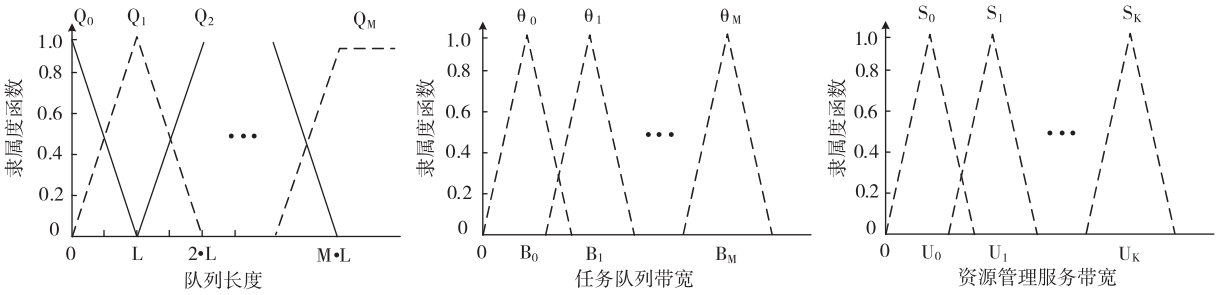

图 4　输入输出模糊集合隶属函数

$$b_i = \begin{cases} B_0, & 0 < q_i < 0.5L \\ B_j, & (j-0.5)L < q_i < (j+0.5)L \\ B_M, & (M-0.5)L < q_i \end{cases} \quad (1)$$

3.3　带宽调整规则

决策分配模块集中处理分散的任务队列带宽，带宽总和 $B_{sum} = \sum_{i=1}^{N_g} b_i$，判断 B_{sum} 模糊集合归属。若 B_{sum} 过大，超出资源管理服务结点最大带宽，则引入任务的准入和退出算法。若 B_{sum} 归属于集合 S_i，集合 S_i 的隶属函数最大值为节点带宽 U_i。记 $\Delta u = B_{sum} - U_i$，若 $\Delta u > 0$，则需要降低任务队列的带宽使得节点带宽取为 U_i。若 $\Delta u < 0$，则提升任务队列的带宽使得 $|\Delta u|$ 最小，最大化利用节点带宽。集合 S_i 隶属函数体现了作业获得服务质量水平和系统资源消耗水平的取舍。带宽调整算法优先保证权重系数大的任务队列。

带宽调整算法描述：

(1)任务加入/退出，重新计算带宽总和 B_{sum}。

(2)当前执行作业结束，获取当前各个任务队列的长度 q_i，根据公式(1)调整任务队列带宽 b_i，重新计算带宽总和 B_{sum}。

(3)若 B_{sum} 不属于任何集合 S_i，重复去除优先级最低的任务队列，直到其归属于集合 S_K。

(4)若 B_{sum} 归属于集合 S_i，记 $\Delta u = B_{sum} - U_i$。如果 $\Delta u > 0$ 执行子算法 1；如果 $\Delta u < 0$ 执行子算法 2；

子算法 1 伪代码描述

```
While( Δu > 0)
  //找出可调整带宽的最低优先级队列
  FindQL( taskQ)
  If( taskQ! = NULL) Then
      Δb  : = taskQ. B[ j] - taskQ. B[ j - 1] ;
      If( |Δu| > Δb ) Then
          j: = j - 1 ; Δu: = Δu - Δb //降低该队列带宽
          Else
          j: = j - 1 ; Break ;
      Endif
  Else
      Break ;
  Endif
End
```

子算法 2 伪代码描述

```
While( Δu < 0)
  //找出可调整带宽的最高优先级队列
  FindQH( taskQ)
  If( taskQ! = NULL) Then
      Δb  : = taskQ. B[ j + 1] - taskQ. B[ j] ;
      If( |Δu| > Δb ) Then
          j: = j + 1 ; Δu: = Δu + Δb //提升该队列带宽
      Else
          setFlag( taskQ) ;//不再调整该队列的带宽
      Endif
  Else
      Break ;
  Endif
End
```

4　仿真及实验

本文设计一组实验从作业平均完成时间，流水线加速比等方面验证流水线资源管理模型的性能，并对作业调度算法的自适应性进行评价。

4.1　流水线资源管理模型仿真

本小节采用多线程流水线模型(C ++ 实现)对流水线资源管理模型进行仿真，实验环境设置为 Intel Dual-core2. 1GHz，RAM 2GB，标准 Linux 系统，内核版本 2. 6. 37，使用 POSIX 时间函数 clock_getTime ( )对作业处理时间进行毫秒量级处理，周期任务作

业执行时间为 1400ms，其中子作业 1 执行时间 400ms，子作业 2 执行时间 1000ms。图 5 展示了任务作业数增大时作业平均完成时间变化曲线。流水线模型作业的实际平均完成时间随着作业数的增加达到一个稳定的最小值，低于传统模型的作业实际平均完成时间。两种模型作业实际完成时间均比理论值要高，原因在于标准 Linux 为分时系统，线程的并发性依靠系统 CPU 调度，因此存在调度产生的时间损耗；同时跨线程的数据传递也会产生一定时间损耗。图 6 为流水线模型的加速比，实际的加速比和理论加速比契合度很高，在本试验中流水线能提供 1.4 倍左右的作业处理加速。

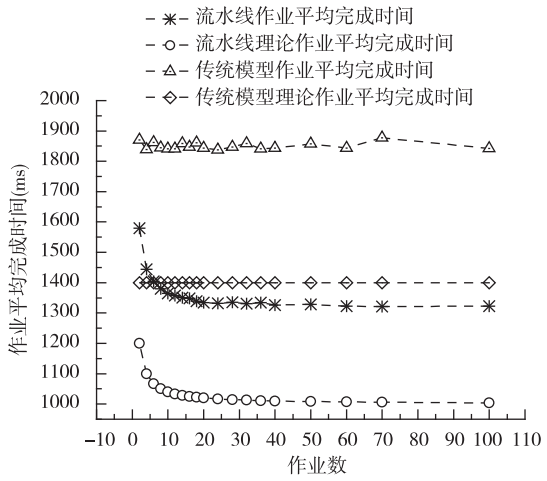

图 5　不同模型作业平均完成时间

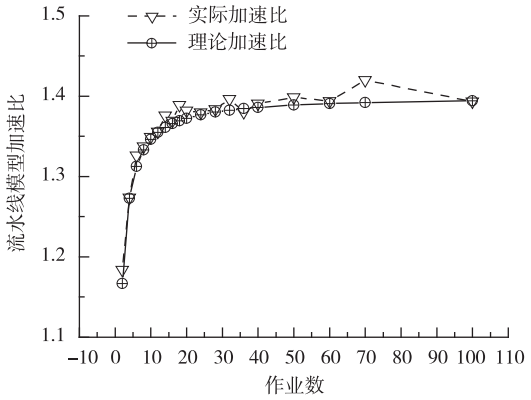

图 6　流水线加速比

4.2　基于模糊控制规则的自适应作业调度仿真

此小节对自适应作业调度算法进行仿真。服务节点采用分布式的自适应调度，因此只需对其中一个任务队列进行仿真。该任务参数设置如下：作业周期 $T=400$ms，作业执行时间取自[100ms，200ms]区间集合，作业数 100（为了图表的辨识度，图表只显示一部分作业），队列带宽取值集合为 $B=\{10\%,20\%,30\%,40\%,50\%,60\%\}$。基于模糊控制规则的自适应调度可以理解为一个携带出水阀门的水箱物理模型[15]，水箱高度为任务作业队列长度，作业加入为水箱注水，调整任务队列带宽为通过调节出水阀门的大小调整水箱高度。本节以作业的实际完成时间 $m_k^i = f_k^i - r_k^i$ 来衡量自适应作业调度的性能。图 7 为所有作业执行时间恒定为 100ms 时，即需求带宽为 25% 时，固定任务带宽和自适应调节队列带宽随作业加入的变化曲线。自适应调度算法动态调整带宽在 20% 至 30% 间变化以寻求注水速度和出水速度的平衡。图 8 为作业实际完成时间随作业加入的变化曲线。固定任务带宽为 10% 时，注水速度大于出水速度，水箱高度不断增加，作业的实际完成时间也累积增加。固定任务带宽为 30% 时，注水速度小于出水速度，能保持作业到来即被处理，作业实际完成时间最短，但是浪费了更多的资源，降低了系统的整体效用。而自适应资源调整

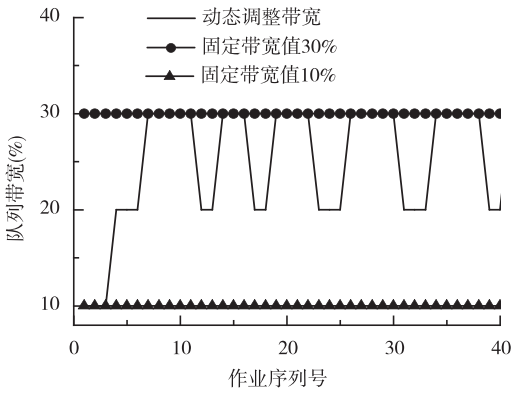

图 7　任务队列带宽变化图

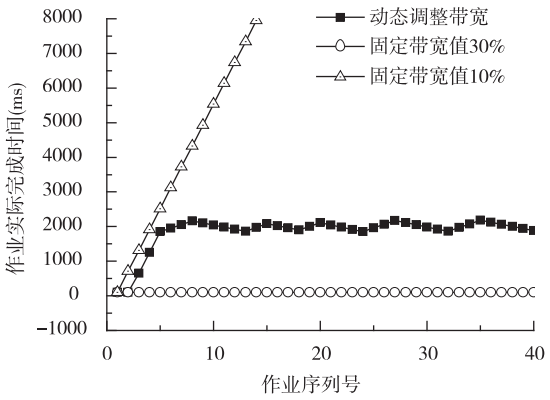

图 8　作业实际完成时间变化图

策略保证注水速度等于出水速度,控制作业实际完成时间稳定在适当范围。当作业的执行时间如图 9 在区间[100ms,200ms]内均匀分布出现时,如图 10 所示,自适应作业调度能体现其性能的优越性。固定任务带宽为 30% 时,作业完成时间不断累积,而自适应作业调度则保持作业完成时间稳定在小值上。

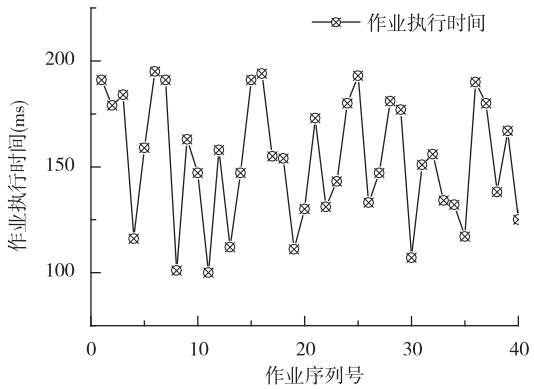

图 9　作业执行时间变化图

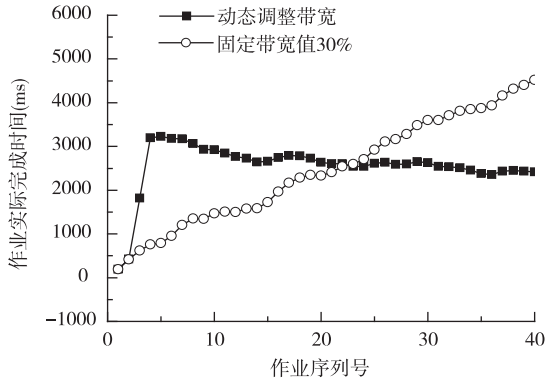

图 10　作业实际完成时间变化图

5　结　论

本文针对嵌入式系统多任务多种独占资源管理问题,提出采用数据流水线的资源管理模型和基于模糊控制规则的自适应作业调度算法。模型为每种资源划分独立的管理服务节点,节点协同合作为任务作业提供流水线加速处理。同时每个节点通过模糊控制规则自适应分配任务队列带宽保证作业执行的实时性。仿真实验展示了模型的流水线加速效果以及作业调度的自适应性。

本文只提供了资源管理模型和作业调度算法的仿真实验,下一步的工作重点将结合智能操作系统数字电视功能模块进行资源管理模型及作业调度算法的具体工程实施。

参考文献

[1] Sha L, Rajkumar R, Lehoczky J P. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on Computers*, 1990, 39(9): 1175-1185

[2] Goodenough J B, Sha L. The priority ceiling protocol: A method for minimizing the blocking of high priority Ada tasks. In: Proceedings of the second international workshop on Real-time Ada issues, New York, USA, 1988. 20-31

[3] Baker T P. A stack-based resource allocation policy for realtime processes. In: Proceedings of the IEEE Real-Time Systems Symposium, Lake Buena Vista, USA, 1990. 191-200

[4] Van Den Heuvel M, Bril R, Lukkien J. Transparent synchronization protocols for compositional real-time systems. *IEEE Transactions on Industrial Informatics*, 2012, 8(2): 322-336

[5] Bertogna M, Fisher N, Baruah S. Resource-sharing servers for open environments. *IEEE Transactions on Industrial Informatics*, 2009, 5(3): 202-219

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

[7] Lehoczky J, Sha L, Ding Y. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In: Proceedings of the IEEE Real Time Systems Symposium, Santa Monica, USA, 1989. 166-171.

[8] Abeni L, Palopoli L, Scordino C, et al. Resource reservation over general purpose applications. *IEEE Transactions on Industrial Informatics*, 2009, 5(1): 12-21

[9] 淮晓永,邹勇,李明树. 一种开放混合实时系统的开放自适应调度算法. 软件学报, 2004, 15(4): 487-496

[10] Stoica I, Abdel-Wahab H, Jeffay K, et al. A proportional share resource allocation algorithm for real-time, time-shared systems. In: Proceedings of the IEEE Real Time Systems Symposium, Los Alamitos, USA, 1996. 288-299

[11] Cucinotta T, Palopoli L. QoS Control for Pipelines of Tasks Using Multiple Resources. *IEEE Transactions on Computers*, 2010, 59(3): 416-430

[12] Cucinotta T, Palopoli L, Abeni L, et al. On the Integration of Application Level and Resource Level QoS Control for Real-Time Applications. *IEEE Transactions on Industrial Informatics*, 2010, 6(4): 479-491

[13] Saaty T L. The Analytic Hierarchy Process. New York, USA: McGraw-Hill, 1980. 107-205

[14] Hiroyu K. Cgroup and Memory Resource Controller. In: Japan Linux Symposium, Japan, 2008

[15] Stankovic J, Lu C, Son S, et al. The case for feedback control real-time scheduling. In: Proceedings of the 11th Euromicro Conference on Real Time Systems, York, England, 1999. 11-20

Pipelined resource management model for embedded systems

Lin Jun[*,**], Ni Hong[**], Sun Peng[**], Zhang Hui[**]

([*] Graduate University of Chinese Academy of Sciences, Beijing 100190)

([**] National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190)

Abstract

A pipelined resource management model and an adaptive job scheduling algorithm based on fuzzy control are proposed to deal with the multi-tasking resource allocation problem in embedded systems. The model creates a set of resources management service nodes which process task jobs by pipeline. Each node manages a type of resource and the adaptive job scheduling algorithm is used to allocation the multitask resources. This algorithm' target is to ensure the real-time characteristics of jobs. It takes the lengths of task queue as inputs, and adjusts the resources allocation by fuzzy control. The experimental result showed that the model improved about 1.4 times rate of job processing based on pipeline acceleration and ensured the real-time characteristics of jobs by effective adaptive adjustments.

Key Words: pipeline, resource management, fuzzy control, embedded system, job scheduling