DOI：10.3901/JME.2011.12.191

面向数控系统的数据流反馈调度框架*

秦承刚[1,2]　于　东[2]　吴文江[2]　韩文业[3]
(1. 中国科学院研究生院　北京　100039；
2. 中国科学院沈阳计算技术研究所　沈阳　110004；
3. 沈阳高精数控技术有限公司　沈阳　110004)

摘要：提出一个数据流反馈调度框架(Feedback scheduling framework for data flow，FSF-DF)，该调度框架可以预防因数据流中断而导致全软件数控系统加工时间延长与加工质量降低的现象。FSF-DF 在数据流上的缓存中设置溢出警戒线，并实时地监测缓存中的数据量是否超出溢出警戒线，以此判断数据流是否有发生数据中断的危险。在数据流将要中断时，调整相关任务的执行频率和优先级，以维持数控系统的稳定性。利用马尔可夫链等统计方法降低执行频率的调整次数，使得 FSF-DF 具有较低的系统开销。通过建立包括监视器、控制器与基本调度器在内的动态反馈调度框架，使得任务执行频率与优先级的调整规则能够有效地集成在实时操作系统中。在实时操作系统 RTAI 中实现了 FSF-DF，并验证它的有效性。试验结果表明，数据流反馈调度框架可以有效地预防数控系统中的数据流中断现象，能够提高数控系统的整体性能与加工速度。

关键词：反馈调度　数控数据流　全软件数控系统　实时操作系统　数据饥饿
中图分类号：TG156

Feedback Scheduling Scheme to Data Flow for Computer Numerical Control System

QIN Chenggang[1,2]　YU Dong[2]　WU Wenjiang[2]　HAN Wenye[3]
(1. Graduate School, Chinese Academy of Sciences, Beijing 100039;
2. Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110004;
3. Shenyang Golding NC Technology Co., Ltd., Shenyang 110004)

Abstract: The interrupt of data flow in the full software computer numerical control (CNC) system will extend the machining time and degrade the machining accuracy. To prevent the interrupt from happening, a new feedback real-time scheduling scheme FSF-DF is proposed. FSF-DF can adjust the period of tasks in CNC system according to the system state. FSF-DF sets an overflow warning line in the buffer on the data flow, and carries out real time monitoring of the data quantity so as to determine whether it goes beyond the overflow warning line, thereby identifying whether there is a hazard of data interrupt. When the data flow is about to interrupt, FSF-DF adjusts the execution frequency and priority of related tasks, so as to maintain the stability of CNC system. Statistical methods such as Markov chain are used to reduce the adjustment times of execution frequency, so that FSF-DF may have a low system overhead. The scheduling scheme, including monitor, controller and basic scheduler, is used to integrate the adjusting rules into real-time operating system. The FSF-DF is implemented in RTAI real-time operating system, and is verified for its effectiveness. Test result shows that the FSF-DF can efficiently prevent the data flow interrupt and improve the whole performance and machining speed of CNC system.

Key words: Feedback scheduling　Data flow in computer numerical control (CNC)　Full software CNC system　Real-time operating system　Data starvation

0　前言

全软件数控系统的软件结构中，存在一条自顶向下逐层处理数据的数据流。数据流上的周期任务实现了数控系统的核心功能。上游任务的输出是下游任务的输入，各个任务通过共享缓存交换数据。数据流式的软件结构降低了任务间的耦合程度，但实时性较高的下游任务有可能因为缓存下溢而出现数据饥饿现象。本文将此现象称为数据流中断。在这种情况下，数控系统会中止连续的高速加工，从

* 国家科技重大专项(2009ZX04009-013)和国家重点基础研究发展计划(973 计划，2011CB302400)资助项目。20100714 收到初稿，20110226 收到修改稿

而延长了加工时间，增大了加工误差。对数据流中断的预防可以提高数控系统的性能。

目前，关于数据流软件结构的研究主要集中在缓存容量与任务集可调度性的静态分析上[1]。静态的分析方法对系统状态要求较为苛刻，不能自适应系统状态的变化。而数控系统是一个典型的动态不可测系统，任务的数据消费速度、执行时间，偶发任务的执行时机都是不确定因素。当静态分析所要求的约束条件被违反时，系统的行为与状态便不可预测。特别是系统过载或是某些任务的数据消费速度的突然增加，可能导致数据流中断。而且，静态分析方法在分配系统资源时过于悲观，增加了系统成本。动态反馈调度算法能够根据系统状态在线地调整任务的调度参数，提高了动态不可测系统的稳定性[2]。文献[3-6]探讨了在实时控制系统中使用动态反馈调度的方法，并逐步完善了控制系统与实时调度协同设计的理论。文献[7-8]将现代鲁棒控制理论应用到实时控制系统的调度中，进一步增强了系统的稳定性。但是这些反馈调度算法没有考虑到任务间的制约关系，以及基于数据流的软件结构。将其运用在数控系统中时，可能会加剧数据流中断现象。

本文设计了一种针对数据流结构的反馈调度框架(Feedback scheduling framework for data flow，FSF-DF)。该框架根据缓存状态调整任务的执行速率和优先级，预防了因缓存下溢而出现的数据流中断。将该框架应用在数控系统中时，能够提高数控系统的加工速度和加工质量。

1 数控系统中的数据流模型

图 1 是数控系统的软件结构模型，该模型将数控加工代码转换为控制伺服电动机的位置指令。解释器任务将加工代码处理为数控系统内部使用的微代码。粗插补任务根据微代码计算出各个运动轴在当前插补周期中运动的距离。加减速控制任务优化粗插补的结果，平滑机床运动。精插补模块给出各个轴在一个控制周期内应运动的距离。位置控制模块根据精插补模块给出的参考位置点和伺服电动机的实际位置点，利用 PID 控制算法计算出发送给伺服的位置指令[9]。由于精插补模块与位置控制模块在时序上需要严格的匹配，通常将它们实现为同一个实时任务，本文中称其为位置控制任务。此外，数控系统还存在若干不在该模型中的任务，比如实时刀轨显示任务，系统日志任务，管理界面等。其他的全软件数控系统可能在功能划分上与本文的模型有所不同，但基本都具有类似的数据流结构。

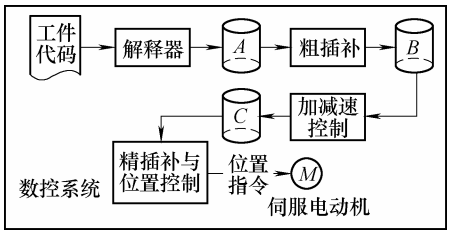

图 1 数控系统中的数据流模型

粗插补任务的数据消费速度是不定的。加工线段的长度以及对加工速度的修调，都会改变该任务的数据消费速度。而且，在实时系统中，任务的调度抖动与处理器负载有直接关系。系统过载会加剧任务的调度抖动[10]，从而造成缓存的数据消费速率与生产速率的变化。当某个缓存的数据消费速率与生产速率长期不匹配时，就有发生下溢的危险。

缓存 A、B、C 中的任何一个出现下溢都会破坏数控加工的连续性。缓存 A 的下溢使得速度规划出现断点，刀具会有一个减速到 0，然后再加速的过程。这不仅延长了加工时间，电动机的加减速还会带来随动误差的抖动，最终增大了加工误差。缓存 B 与 C 的下溢，使得位置控制任务无法发出新的位置点，造成加工过程的停顿，也延长了加工时间。如果下溢过于频繁，甚至有可能出现电动机抖动。为了确保数控加工的速度和精度，必须预防数据流中断的发生。

2 数据流中断的预防方法

如第 1 节所述，缓存的数据生产速率与消费速率是不断变化的。当消费速率大于生产速率时，有可能发生下溢。反之，有可能发生上溢。数控系统中，生产者任务每个周期仅生产一条数据，缓存的数据生产率即为生产者任务的执行频率。在缓存将要下溢时，通过增加生产者任务的执行频率，能够避免数据流中断。在缓存将要上溢时，可以降低生产者任务的执行频率，将处理器带宽分配给其他任务，以改善数控系统的整体性能。通过在缓存中设置上溢和下溢警戒线，可以得知缓存是否有发生溢出的危险。如果有可能发生溢出，则及时地调整生产者任务的执行频率。在数据流已中断时，可以利用优先级继承机制减少中断的时间。

2.1 任务模型与缓存模型

将数据流上的周期任务用 5 元组表示为 $\tau_i=(p_i^n,p_i,T_i^n,T_i,e_i)$。其中 p_i^n 为任务的名义优先级，是在系统的设计阶段分配给任务的优先级。在数据流上，各个任务的重要性自上而下依次增加。位置

控制任务位于数据流的尾端，名义优先级最高。p_i 为任务的当前优先级。T_i^n 为任务的名义周期，是在系统设计阶段为任务分配的周期。T_i 为任务的当前周期，$T_i^n \leqslant T_i$。e_i 为任务的在最坏情况下无中断执行所花的时间(Worst case execution time，WCET)。

数据流上的缓存 B，分别为 C、W_{cur}、W_h、W_l、$R_c^{\max}$、$R_c^{\min}$ 与 R_p。其中，C 为缓存的容量，W_{cur} 为缓存当前的数据量。W_h 为上溢警戒线，W_l 为下溢警戒线。缓存模型如图 2 所示。消费者任务在单位时间内消费的数据量称为缓存的数据消费率 R_c，生产者任务在单位时间内生产的数据量称为缓存的数据生产率 R_p。$R_c^{\max}$ 为该缓存的最大数据消费率，$R_c^{\min}$ 为最小数据消费率。粗插补任务有可能很多个周期都不需消费数据，可以认为缓存 A 的 $R_c^{\min}$ 等于 0。而在最坏的情况下，该任务每个周期都要消耗一条数据，所以缓存 A 的 $R_c^{\max}$ 为 $1/T_{rp}$，T_{rp} 为粗插补任务的周期。加减速控制任务与位置控制任务每个周期都只消耗一条数据。因此对缓存 B 与 C 而言，$R_c^{\max} = R_c^{\min} = 1/T_c$。$T_c$ 为消费者任务的周期。各个缓存的 R_p 可以根据生产者任务的周期来确定，即 R_p=1/T_p。其中 T_p 为生产者任务的周期。

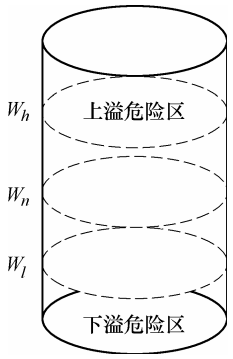

图 2　缓存模型

2.2　警戒线的分配方法

定义缓存的数据量变化率 ΔR=R_p–R_c。当 ΔR>0 时，缓存中的数据量正在增加。ΔR<0 时，缓存中的数据量正在减少。设 b_i 与 b_{i+1} 分别为时刻 t_i 与 t_{i+1} 时，缓存中的数据量。那么在时间段[t_i,t_{i+1}]中，缓存中数据的变化量为

$$\Delta b = b_{i+1} - b_i = (R_p - R_c)(t_{i+1} - t_i) = \Delta R \Delta T \qquad (1)$$

通过设置溢出警戒线可以提前判断缓存是否将发生溢出，从而为调整操作留取时间。因此，可以通过数据变化速度的最大值和调整操作需要的时间 ΔT 来确定溢出警戒线的值。ΔT 应该包括调度框架的监测周期和生产者任务响应调整的时间。监测周期 T_s 是指检查缓存数据量的时间间隔。如果缓存中的数据量在一次检查之后，立刻超出了 W_l 或 W_h，那么直到下次监测才能发现这个情况。这时可能已经发生了溢出。因此 ΔT 必须包括一个监测周期。生产者任务的周期被调整后，需要等到下个周期才能执行。如图 3 所示，假设数据流上的所有任务都能在其时限之前执行完毕，生产者任务两次输出之间的最大时间间隔应该为 $2T_p^n - e_p$。因此

$$\Delta T = 2T_p^n - e_p + T_s \qquad (2)$$

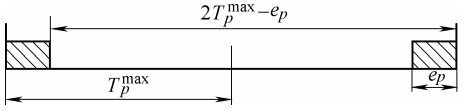

图 3　相邻 2 次数据生产之间的最大时间间隔

在 ΔR<0 时，数据量的最大变化率为 $R_c^{\max} - R_p$。在 ΔR>0 时，数据量的最大变化率为 $R_p - R_c^{\min}$。定理 1 给出了为缓存分配警戒线的方法。

定理 1：为了预防缓存溢出，缓存的高低警戒线应满足如下条件

$$W_l \geqslant (R_c^{\max} - R_p)(2T_p^n - e_p + T_s) \qquad (3)$$

$$W_h \leqslant C - (R_p - R_c^{\min})(2T_p^n - e_p + T_s) \qquad (4)$$

2.3　生产者任务周期的调整算法

缓存可以消除数据生产与数据消费的速率差异，只要缓存的数据量在区间[W_l,W_h]内，便无须调整生产者任务的执行速率。而且，在 ΔR>0 时，即使缓存中的数据量低于 W_l，也无须调整，因为缓存中的数据量已经在增加了，不存在下溢的危险。同样，在数据量高于 W_h 并且 $\Delta R < 0$ 时，也不存在上溢的危险，因为数据量已经在减少了。因此，调整生产者任务执行速率的时机可以概括为规则 1。

(1) 如果缓存中的数据量低于 W_l，并且 ΔR<0，需要增加生产者任务的执行速率。

(2) 如果缓存中的数据量高于 W_h，并且 ΔR>0，需要降低生产者任务的执行速率。

执行速率调整的首要目标是预防缓存溢出。在缓存将要溢出时，数据生产率至少应该变化 ΔR，才能适应消费者任务的需求。但是，实时系统对调度开销的敏感性要求在预防缓存溢出之外，还应减少未来的调整次数。ΔR 的变化具有阶段性和重复性，在保持一个较为稳定的阶段之后，才会出现大的跃变。而且，变化模式可能会重复出现多次。而对生产速率的调整通常发生在 ΔR 出现跃变时。利用 ΔR 的统计特性，可以尽量保证在 ΔR 没有发生跃变时，不用对生产者的执行频率进行调整。

当 ΔR 在一个监测周期内的变化超过了正整数 δ 时，便认为 ΔR 发生了跃变。并令 Δt_c 为两次跃变

之间的时间间隔。使用 K 个连续的 Δt_c 的数学期望 Δt_c^{next}，便可以估计出 ΔR 下次跃变的时间。将 K 称作采样窗口。那么调整的目标至少应是，在 Δt_c^{next} 之前，将缓存的数据量保持在安全范围内。

根据 ΔR 未来的跃变方向来确定执行频率的调整量，能够进一步降低调整次数。考虑以下情况：在某次缓存的数据量低于 W_l 并且 $\Delta R<0$ 时，提高了生产者的执行频率，但是提高的幅度小于 2δ。如果 ΔR 在下次跃变时减小，必然会出现 $\Delta R<0$ 的情况。此后，缓存中的数据量将会降低。那么，如果能够使缓存中的数据量在下次跃变时达到 W_h，就可以推迟再次调整的时间。若是生产速率的增加幅度大于 2δ，即使 ΔR 再次跃变时仍然减小，也不会有 $\Delta R<0$ 的情况。在这种情况下，便不能使数据量在 Δt_c^{next} 后达到 W_h，否则在跃变后，需要立即进行调整。如果 ΔR 在下次跃变时增大，可以使数据量在 Δt_c^{next} 结束时仅到达 W_l。

K 阶马尔可夫模型可以根据已经发生的 K 个状态预测当前状态。利用已经发生的跃变方向作为马尔可夫模型的训练集，能够得到在 K 次跃变已经发生的情况下，两种跃变方向的条件概率。

使用跃变状态表记录已经发生的跃变方向。

定义 1：ΔR 的跃变序列 μ 记录了 ΔR 从系统启动到当前时刻，每次跃变时的变化方向。μ 是一个二值序列，序列中元素的定义为

$$\mu(k)=\begin{cases}1 & \text{在}\Delta t_c^k\text{开始时，}\Delta R\text{增加}\\ 0 & \text{在}\Delta t_c^k\text{开始时，}\Delta R\text{减小}\end{cases} \tag{5}$$

μ^K 是跃变序列 μ 尾部的 K 个元素组成的子序列。跃变状态表 $S_{\text{T}}=\{s,s_0,s_1\}$ 用来统计所有长度为 K 的序列发生的次数。表中的每条记录均对应一个长度为 K 的序列，字段 s 记录了该序列的出现的次数；s_1 记录了在该序列之后跃变方向为 1 的次数；s_2 记录了该序列之后跃变方向为 0 的次数。由于表中的元素个数为 2^K，K 的取值不能过大。调度程序在 ΔR 跃变时，根据当前的跃变方向更新跃变状态表。在跃变序列 μ^K 之后，跃变方向为 θ 的条件概率为

$$P(\theta\,|\,\mu^K)=\frac{\mu^K s\{\theta\}}{\mu^K s}\qquad \theta=0,1 \tag{6}$$

式中，$P(\theta\,|\,\mu^K)$ 可以作为 K 阶马尔可夫模型的转移概率。$\mu^K s\{\theta\}$ 与 $\mu^K s$ 的值可以从状态表中查到。

如果概率 $P(\theta\,|\,\mu^K)$ 较低，那么预测错误的可能性比较大。错误的预测会造成对生产速率的调节力度过大或者过低，从而导致调整次数增加。为了降低预测错误的可能性，为跃变方向的条件概率 $P(\theta\,|\,\mu^K)$ 设置一个可信度 P_{ref}。只在 $P(\theta\,|\,\mu^K)>P_{ref}$ 时，才认为预测是可信的。当 $P(1\,|\,\mu^K)$ 和 $P(0\,|\,\mu^K)$ 都小于 P_{ref} 时，可以将调整目标设定成为在 Δt_c^{next} 结束后，缓存中的数据量应达到缓存容量的一半。根据以上叙述，可以得到数据生产率的调整策略。调整后的数据生产率应该满足如下关系

$$R_p^{\text{new}}=\frac{W_{\text{obj}}-W_{\text{cur}}}{\Delta t_c^{\text{next}}}+R_c \tag{7}$$

式中，W_{obj} 为缓存的数据量 Δt_c^{next} 之后的预期值，W_{cur} 为当前的数据量。对 W_{obj} 的选取需要遵循规则 2。

(1) 规则 1 中的情况 1 成立，并且 $P(0\,|\,\mu^K)>P_{\text{ref}}$。如果 $(W_h-W_{\text{cur}})/\Delta t_c^{\text{next}}<\delta$，那么 $W_{\text{obj}}=W_h$。否则 $W_{\text{obj}}=W_n$。

(2) 规则 1 中的情况 1 成立，并且 $P(1\,|\,\mu^K)>P_{\text{ref}}$，那么 $W_{\text{obj}}=W_l$。

(3) 规则 1 中的情况 2 成立，并且 $P(0\,|\,\mu^K)>P_{\text{ref}}$，那么 $W_{\text{obj}}=W_h$。

(4) 规则 1 中的情况 2 成立，并且 $P(1\,|\,\mu^K)>P_{\text{ref}}$，如果 $(W_{\text{cur}}-W_l)/\Delta t_c^{\text{next}}<\delta$，那么 $W_{\text{obj}}=W_l$。否则 $W_{\text{obj}}=W_n$。

(5) 规则 1 成立，并且 $P(0\,|\,\mu^K)<P_{\text{ref}}$，$P(1\,|\,\mu^K)<P_{\text{ref}}$，那么 $W_{\text{obj}}=W_n$。

在缓存存在溢出的危险时，根据定理二计算缓存生产者的新周期。

定理 2：调整后的生产者任务的周期应该满足关系

$$T_p^{\text{new}}=\frac{T_p\Delta t_c^{\text{next}}}{T_p(W_{\text{obj}}-W_{\text{cur}})+(1-T_p\Delta R)\Delta t_c^{\text{next}}} \tag{8}$$

2.4 优先级调整方法

在数控系统的数据流结构中，缓存消费者任务的优先级均比生产者任务高。当缓存下溢时，生产者任务可能会因为优先级较低而得不到立即执行。特别是当系统中存在优先级高于生产者而低于消费者的任务时，生产者任务有可能因被抢占而长期处于等待状态。如图 4a 所示，任务 τ_1 的优先级高于生产者任务 τ_p，低于消费者任务 τ_c。由于被 τ_1 抢占，τ_p 直到接近时限时才输出数据，而消费者任务已空运转了多个周期。若是能减少生产者任务的等待时间，便可以减少消费者任务数据饥饿的时间。优先级逆转是指高优先级的任务因共享资源被低优先级的任务持有而被迫等待的现象。可以将消费者任务的数据饥饿类比成优先级逆转现象。实时操作系统

通常使用资源访问控制协议来抑制优先级逆转。优先级继承协议是一种简单的资源访问控制协议，它虽不能消除优先级逆转现象，但是可以减少高优先级任务等待的时间[11]。参考优先级继承协议，本文在数据流模型上使用规则三对生产者任务的优先级进行调整，以减少缓存下溢的时间。

规则 3 如下所述。

在某个缓存下溢时，便将生产者任务的优先级提升至消费者任务的优先级。缓存不空时，再恢复该任务的优先级。

当缓存下溢时，规则 3 减少了其他任务对生产者任务的干扰，使它能够尽快地得以执行。规则 3 具有传递性，若图 1 中的缓存 A 与缓存 B 同时下溢，可以将解释器任务的优先级提升至加减速控制任务的优先级。如图 4b 所示，在消费者任务 τ_c 执行时发生了缓存下溢，因此将生产者任务 τ_p 的优先级提升至 τ_c 的优先级。此后，τ_1 无法抢占 τ_p，减少了 τ_c 数据饥饿的时间。考虑到位置控制任务的高实时性，不能有任何任务的优先级与其相等，否则便会干扰到位置控制任务的时间行为。因为，对于优先级相等的任务，多数实时操作系统采用时间片轮转的调度策略。优先级与位置控制任务相同的实时任务会延长位置控制任务的执行时间，从而增加了控制延迟，降低了数控系统的加工质量。因此规则 3 不适用数控软件模型中的缓存 C。

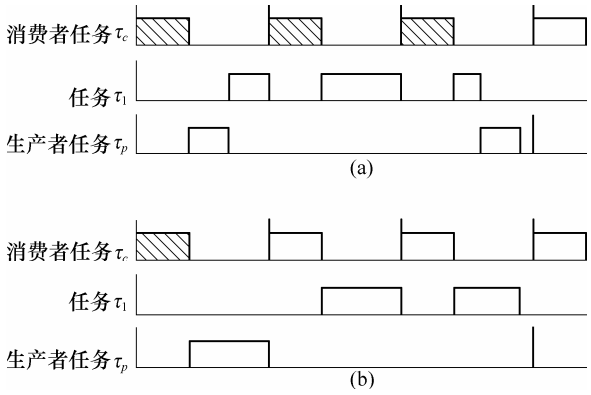

图 4　通过优先级调整减少消费者任务数据饥饿的时间

3　反馈调度框架

3.1　反馈调度框架的基本结构

如图 5 所示，调度框架由监视器，控制器以及基本调度器组成。监视器负责监测所有缓存的数据量和数据量的变化率。以此为依据计算出 ΔR 每次发生跃变的时间间隔，更新系统中保存的 K 个最新的 Δt_c 的值。并根据跃变方向更新跃变状态表。控制器负责调整各个任务的周期与优先级。首先，比较各缓存的数据量是否超出了警戒线，然后利用规则 2 和定理 2 计算出各周期任务的新周期。如果出现缓存下溢，便根据规则 3 调整该缓存生产者任务的优先级。

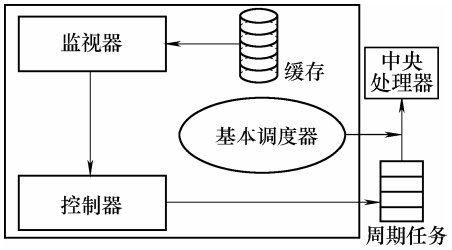

图 5　调度框架

反馈调度在传统的实时调度算法之上增加了监测与调整的框架。因此，反馈调度中包括一个执行传统实时调度算法的基本调度器。常用的实时调度算法可以分为静态优先级算法与动态优先级算法。

虽然静态优先级调度算法的处理器利用率较低，但是可预测性优于动态优先级调度算法。由于数控系统具有较为固定的软件结构，本文采用了固定优先级的静态调度方法。

3.2　负载均衡

对一个实时调度算法而言，如果所有实时任务的时限约束都能得到满足，便称该任务集是可调度的。任务集的系统负载不大于实时调度算法的可调度利用率是可调度的充分条件。当任务集的系统负载超出了调度算法的可调度利用率时，便认为系统处在过载状态，过载的系统无法保证实时任务的时限约束，也不能稳定地提供正常功能。因此在反馈调度框架 FSF-DF 中，必须保证任务集的系统负载低于基本实时调度算法的可调度利用率。

式(9)给出了静态优先级实时调度算法的可调度利用率。当任务集的处理器利用率在满足式(10)时，该任务集才具有可调度性。

$$U = n\left(2^{1/n} - 1\right) \tag{9}$$

$$\sum_{i=0}^{n-1} \frac{e_i}{T_i} \leqslant U \tag{10}$$

式中，n 为任务集的任务数量。如图 1 所示，数控系统中的周期性硬实时任务共有 4 个，可调度利用率 U 应为 75.7%。在根据式(8)为某个任务分配了新周期之后，需要判断在这样一个周期下，任务集是否还具有可调度性。如果任务集的处理器利用率超出了 U，需要重新计算被调整任务的周期。假设在调整之前，任务集的处理器利用率为 U_{old}，那么被

调整任务可增加的处理器带宽最多为 $U–U_{\text{old}}$，因此任务周期的最小值 $T_p^{\min}$ 可以通过式(11)得出。如果 $T_p^{\text{new}} < T_p^{\min}$，便令 $T_p^{\text{new}} = T_p^{\min}$。

$$T_p^{\min} = \frac{e_p}{U - U_{\text{old}}} \tag{11}$$

数控系统是一个混合任务系统[12]。除了数据流上的硬实时任务外，数控系统中还存在很多非实时任务[13]，比如实时刀轨显示任务，系统日志任务等。很多实时操作系统都提供了多级调度框架，采用前后台的方式调度混合任务集。在硬实时任务执行的间隙执行非实时任务。硬实时负载的增加，会抢占非实时任务的计算带宽。从而影响到它们的服务质量，比如实时刀轨显示任务的图形质量。

3.3 控制器算法

控制器是整个调度框架的核心，综合以上对数据流上各任务的周期及优先级的调整规则，可以得到控制器的算法框架。

算法 1 如下所述。

矢量 $\boldsymbol{B}$ 为描述缓存数据结构的数组，大小为 N，N 为数据流上缓存的个数；T_{ctrl} 为控制器的执行周期。

```
Dataflow_Controller()
{
    for(i=0; i<N; i++)
    {
根据缓存 B[i]中数据的变化量与 T_ctrl 计算 ΔR;
        if(ΔR>δ_d)
        {//ΔR 发生了跃变
            更新 Δt_c^next ;
            更新跃变状态表;
        }
        if((B[i].W_cur > B[i].W_h && ΔR>0) ||
(B[i].W_cur < B[i].W_l && ΔR < 0))
        {//缓存 B[i]将要发生上溢或下溢
            根据规则二计算 W_obj;
            根据式(8)计算缓存 B[i]的生产者分配新周期 T_p^New ;
            根据式(11)计算 T_p^Min ;
            if( T_p^new < T_p^min )
                T_p^new = T_p^min ;
            if( T_p^new > T_p^n )//生产者任务的新周期大于它的名义周期
                T_p^new = T_p^n ;
            根据式(2)与式(3)更新缓存 B[i]的数据生产率与溢出警戒线;
            if(i!=0)
                更新缓存 B[i–1]的数据消费率与溢出警戒线;
        }
        if(B[i].W_cur == 0 && B[i]的消费者不是位置控制任务)
        {//缓存 B[i]发生了下溢
            令 B[i]生产者的优先级等于 B[i]消费者的优先级;
        }
        if(B[i].W_cur != 0 && p_p != p_p^n)  //缓存 B[i]不空，并且生产者任务的优先级不等于它的名义优先级
        {
            p_p = p_p^n; //将生产者任务的优先级恢复为它的名义优先级
        }
    }
}
```

该算法的时间复杂度为 $O(N)$，仅给实时调度带来了有限的开销。

4 试验

本文在全软件数控系统中采用了开源实时操作系统 RTAI。RTAI 是基于 Linux 操作系统的实时扩展，支持硬实时调度和多级调度框架。本文在 RTAI 的基础上，实现了数据流反馈调度框架 FSF-DF。首先，根据需要扩展了 RTAI 的缓存模型和任务模型。其次，通过一个具有最高优先级的实时任务实现了监视器和控制器，该任务使用“算法 1”调节相关任务的周期和优先级。试验平台的处理器为 Pentium E2200，主频为 2.20 GHz，内存为 2 GB。在试验过程中，采用了一个小线段密集的工件模型。加工此模型时，粗插补任务的数据消费率存在较大的变化。

试验中数控软件的结构基本如图 1 所示。数据流上共包含 4 个周期任务与 3 个共享缓存。表 1 给出了流水线上各周期任务的参数设置。这些任务均为硬实时任务，其中粗插补任务，加减速任务，位置控制任务运行在内核空间。解释器任务利用 RTAI 提供的 LXRT(Linux real-time)机制，运行在用户空间。硬实时任务集的处理器利用率为 46.04%。由于

静态优先级实时调度算法的可调度利用率为75.7%，存在很多富余处理器带宽。表 2 给出了各共享缓存的属性。

表 1　各周期任务的属性

任务	名义优先级 p_i^n	名义周期 T_i^n /ms	WCET e_i /ms
解释器	3	5	0.452
粗插补	2	1	0.161
加减速控制	1	1	0.073
位置控制	0	0.125	0.017

表 2　各缓存的属性

缓存	容量 C_1/个	下溢警戒线 W_l/个	上溢警戒线 W_h/个
A	100	95	305
B	50	12	38
C	10	2	8

FSF-DF 中存在 3 个参数 δ、K 与 P_{ref}。δ 的取值应使 Δt_c^{next} 的变化较为平稳，因为 Δt_c^{next} 的剧烈变化会影响到马尔可夫预测的效果。图 6 反映了 δ 对 Δt_c^{next} 的变化过程的影响。从图 6 可以看到，当 δ=30 时，Δt_c^{next} 的变化较为平稳。采样窗口 K 是用于统计的样本数量。K 的值越大，对 Δt_c^{next} 与跃变方向的预测就越准确，但同时也带来了较大的系统开销。P_{ref} 反映了高阶马尔科夫预测的可信度。P_{ref} 的值越大，对跃变方向预测错误的可能性就越低。本文在试验中，评价了 K 与 P_{ref} 对算法性能的影响。

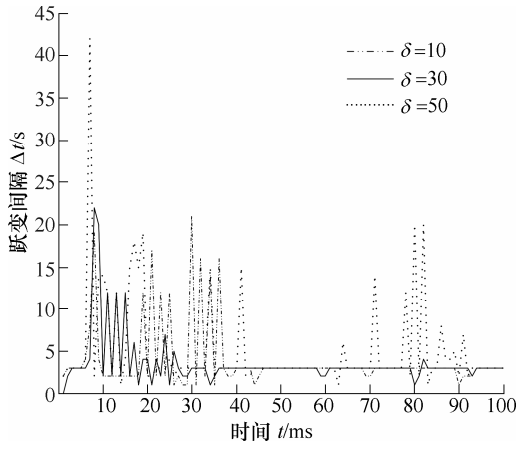

图 6　跃变时间间隔的变化图

未开启反馈调度时，过多的小线段使得粗插补任务的数据饥饿次数累计达到了 15 310 次，粗插补任务的等待又导致了下游任务的数据饥饿。在开启反馈调度以后，缓存的上溢和下溢得到了有效的控制。表 3～5 给出了测试结果。

从表 4 中可以看到，K 的值越大算法的效果就越好。因为样本数量的增加使得统计结果更加可信。但是，K 的增加会加大系统负载，从而加剧了实时任务的调度抖动。当 P_{ref} 过大的时候，规则 2 中的情况 1 到情况 4 成立的次数很少，从而出现很多对数据生产速率调整力度不够的情况，增加了调节次数。从试验结果中看到，P_{ref}=0.9 时效果最好。图 7 为 K=5，P_{ref}=0.9 时，缓存 *A* 中数据量的变化情况。从图 7 中的数据量变化曲线可以看到，当缓存有溢出的危险时，反馈调度框架能够及时地调节生产者任务的执行频率，有效地预防了溢出的发生。未开启反馈调度时，加工过程历时 85 min。开启反馈调度后，加工过程在 53 min 左右便完成了，大幅节省了加工时间。在试验过程中，系统负载的变化使得实时刀轨显示任务的图形质量出现波动。但是，这种现象只在小线段过于密集的时候才会出现，不会维持很长时间。

表 3　未开启反馈调度

缓存	调节次数 N/次	下溢次数 n_d/次	上溢次数 n_u/次
A	0	15 310	37
B	0	7 438	21
C	0	7 539	0

表 4　采样窗口 *K* 取不同的值时，各缓存的溢出情况

采样窗口 K/个	缓存	调节次数 N/次	下溢次数 n_d/次	上溢次数 n_u/次
3	*A*	28 860	82	0
	B	29 310	117	0
	C	31 247	132	0
4	*A*	23 147	81	0
	B	20 842	121	0
	C	20 143	116	0
5	*A*	13 426	63	0
	B	14 173	105	0
	C	16 395	109	0
6	*A*	15 712	71	0
	B	16 431	112	0
	C	17 933	131	0

表 5　在不同的 P_{ref} 值下，各缓存的溢出情况(K=5)

可信度 P_{ref}/%	缓存	调节次数 N/次	下溢次数 n_d/次	上溢次数 n_u/次
0.8	*A*	15 712	71	0
	B	17 651	112	0
	C	17 931	117	0
0.9	*A*	13 426	63	0
	B	14 173	105	0
	C	16 395	109	0
0.95	*A*	15 197	81	0
	B	15 203	113	0
	C	17 475	131	0

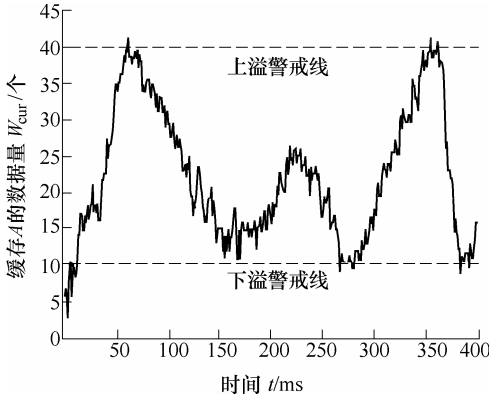

图 7 缓存 A 的数据量变化

4 结论

(1) 数控系统中的数据流中断会对加工时间和加工质量造成负面影响。可以利用本文提出的反馈调度框架 FSF-DF 预防数据流中断。FSF-DF 通过在缓存中设置溢出警戒线，有效地判断数据流是否存在中断的危险。当缓存有可能溢出时，通过调节生产者任务的周期使得缓存中的数据量向相反的方向变化。

(2) 反馈调度框架 FSF-DF 利用了统计方法及高阶马尔可夫模型，对缓存数据量变化率的跃变时间及跃变方向进行预测，减少了调整次数。将数控系统中已发生的事件作为高阶马尔可夫模型的训练集，使得该模型具有自适应能力，对不同的工件代码都能获得较好的预测效果。

(3) 通过在 RTAI 实时操作系统中的实现以及对全软件数控系统的研究分析表明：数据流反馈调度框架 FSF-DF 较好地预防了数据流的中断，提高了数控系统的加工速度。但是对算法参数的选取具有经验性，进一步的研究工作将提出一个可以根据调度效果自动调节算法参数的机制。

参 考 文 献

[1] JONATHAN P, SHUVRA S, BHATTACHARYYA, et al. Interface-based hierarchy for synchronous data-flow graphs[C]// IEEE Int. The Signal Processing Systems Conf., Oct. 7-9, 2009, Tampere, Finland. New York：IEEE Press, 2009：145-150.

[2] LU Chengyang, JOHN A, SANG H, et al. Feedback control real-time scheduling：Framework, modeling, and algorithms[J]. Real-time Systems, 2002, 23(1-2)：85-126.

[3] ANTON C, JOHAN E, BO B, et al. Feedback feedforward scheduling of control tasks[J]. Real-time Systems, 2002, 23(1-2)：25-53.

[4] MARTI P. Analysis and design of real-time control systems with varying control timing constraints [D]. Barcelona：Technical University of Catalonia, 2002.

[5] CERVIN A, HENRIKSSON D, LINCOLN B, et al. How does control timing affect performance[J]. IEEE Control Systems Magazine, 2003, 23(3)：16-30.

[6] PINGFANG Z, JIANYING X. Feedback scheduling for resource-constrained real-time control systems[C]// IEEE Computer Society. The 5th International Conference on Computer and Information Technology, Sept. 21-23, 2005, Shanghai, China. New York：IEEE, 2005：800-804.

[7] SIMON D, ROBERT D, SENAME O. Robust control / scheduling co-design：Application to robot control Technology[C]//IEEE Int. The 11th Real Time and Embedded Technology and Applications Symposium, March 7-10, 2005, San Francisco, CA, USA. New York：IEEE, 2005：118-127.

[8] BING D, CHUN R. FEL-H robust control real-time scheduling[J]. Software Engineering & Applications, 2009, 2：60-65.

[9] SUKHWAN S, SEONG-KYOON K, DAEHYUK C, et al. Theory and design of CNC systems[M]. London：Springer Press, 2008.

[10] BUTTAZZO G, LIPARI G, CACCAMO M, et al. Elastic Scheduling for flexible workload management[J]. IEEE Transaction on Computers, 2002, 51(3)：289-302.

[11] JANE W S L. 实时系统[M]. 北京：高等教育出版社, 2003.
JANE W S L. Real-time system[M]. Beijing：Higher Education Press, 2003.

[12] 李迪, 万加富, 叶峰, 等. 软数控系统混合任务两级调度策略[J]. 机械工程学报, 2008, 44(12)：157-162.
LI Di, WAN Jiafu, YE Feng, et al. Two-level hierarchical scheduling scheme of hybrid tasks for software CNC system[J]. Chinese Journal of Mechanical Engineering, 2008, 44(12)：157-162.

[13] 周刚, 邬义杰, 潘晓弘. 数控系统软件模块实时调度方法[J]. 机械工程学报, 2009, 45(1)：162-166.
ZHOU Gang, WU Yijie, PAN Xiaohong. Software modules real time scheduling framework in CNC system[J]. Journal of Mechanical Engineering, 2009, 45(1)：162-166.

作者简介：秦承刚(通信作者)，男，1983 年出生，博士研究生。主要研究方向为实时操作系统、信息物理融合系统。
E-mail：qinchenggang@sict.ac.cn
于东，男，1966 年出生，博士，研究员，博士研究生导师。主要研究方向为实时控制系统、数控技术。
E-mail：yudong@sict.ac.cn