·应用技术研究 ·

基于 RT-UML模型的实时系统可调度性分析

高军礼[1],李 迪[2],郑时雄[2]

(1. 广东工业大学 自动化学院,广东 广州 510006; 2. 华南理工大学 机械工程学院,广东 广州 510640)

摘 要:简要回顾实时系统的开发现状,对实时系统可调度性相关理论进行探讨. 提出一种基于实时统一建模语言对实时系统可调度性进行分析的方法. 通过将系统 RT-UML模型中实时任务的相关数量信息提取出来,在相应分析工具中进行可调度性分析,分析结果自动反馈到模型中去,实现了对实时系统可调度性进行系统实现前的离线分析.

关键词:实时统一建模语言;可调度性分析;实时系统

中图分类号: TP342. 3; TP311. 52 **文献标识码**: A **文章编号**: 1003-0972(2006)03-0349-04

近几年的统计资料表明,在美国 52% 的软件项目至少超出预算的 189%,31% 的软件项目中途被取消[1];而在欧洲有 34% 的实时嵌入式软件在开发过程中已被迫取消,72. 7% 的软件在产品开发完成时已超出预算的 50%.

究其原因主要是:(a)、软件本身没有明显的平均经常成本. 所有的成本都包含在软件的整个生命周期内,因而人们往往更加注重降低显而易见的硬件的平均经常成本,而软件成本却相应增加. 这对软件成本占有更大比重的实时系统来说势必会超过预算,以至影响到软件的正常开发;(b)、实时软件的开发者在使用最流行的软件工程方法学时,总要比其他类型软件的开发者滞后. 目前强耦合的过程式模块开发还是主流,仍旧以程序代码为中心[2]. 尽管有的也采用了面向对象建模技术,但由于模型是不可执行的、模型与代码之间不能动态关联等原因,以至于尽管在对程序代码不断完善的同时也不断的更新模型,最终还是会造成模型与程序代码之间的不一致,甚至最后放弃了模型的更新. (c)、实时系统的功能和性能之间的研究与应用存在相互独立或脱节的事实[3]. 实时系统的正确性不仅依赖于计算的逻辑结果即功能方面,而且还依赖于计算结果产生的时间即性能方面 (如可调度性). 二者能够有机地结合起来显得尤为重要.

1 实时统一建模语言

统一建模语言 (UML)自 1995年被对象管理组织 (OMG)确定为工业标准建模语言以来已被学术界、软件开发者和软件开发商所广泛接受,并成功应用于商业软件的开发. 但 UML1. x以前的版本在语义精确性方面存在着明显的不足,以至于难以由 UML模型生成产品质量级的代码、UML模型的可执行也难以实现. 另外,没有在 UML模型中建模有关实时系统实时信息的标准方法. 正是这些缺陷阻碍了其在实时系统方面的应用. 不过最新的 UML2. 0有效地解决了这些问题. 首先 OMG采纳了实时 UML特征文件包 (RT-UML Profile)作为对 UML在实时系统建模方面的扩展. 该特征文件包定义了在 UML模型中建模实时系统实时信息的标准方法,提供了和系统软硬软件资源相关、时间相关的量化信息即服务质量 (QoS)的建模基础,增强了 UML在实时系统建模方面的能力[4]. 再者,UML2. 0语义的精确性也做了很大改进,使得 UML模型精确到能由模型直接生成产品质量级代码甚至是整个应用程序. 因此,对于实时系统的功能方面,可以采用模型驱动开发方法创建可执行的 UML模型,在模型层测试所开发实时系统的功能,具体实现过程参见文献[5]. 而对于实时系统的性能方面,可以通过在 UML模型中建模实时系统相关的 QoS进行系统实现前的离线分析. 本文将就实时系统可调度性分析进行深入的探讨.

2 实时系统可调度性分析相关理论

实时系统可以被刻画成一系列任务集. 所谓任务是指一个具有独立功能的无限循环的程序段的一次运行活动,是实时内核调度的单位,具有动态性、并发性和异步独立性[6],多表现为一个独立的线程. 任务包含可执行代码、数据、堆栈和程序执行的上下文环境等内容. 按照任务到达情况的可预测性,分为周期性和非周期性两种类型. 本文所讨论的任务为周期性任务. 任务的特性可以通过优先级、周期、执行时间、就绪时间、最后期限等参数进行描述. 对一个任务要进行可调度性分析,至少要指定其周期、执行时间和最后期限. 一个任务通常具有 3个状态:就绪 (Ready)、阻塞 (Blocked)和运行 (Running)状态,其状态转

收稿日期: 2005-11-16;**修订日期**: 2006-03-07

基金项目:广东省重点科技攻关资助项目 (2002C1020407; 2003A1040703);广东工业大学博士基金资助项目 (053037)

作者简介:高军礼 (1973-),男,河南尉氏人,讲师,博士,主要从事计算机控制技术研究.

换过程如图 1所示 .就绪状态指该任务准备就绪但还未执行,因为一个更高优先级的任务正在执行;阻塞状态指该任务已请求一个还不能使用的资源,或已经请求等待某些事件的发生,或其自身要延迟一段时间时所处的状态,有时该状态还包含悬挂 (Pended)和延迟 (Delayed)子状态;运行状态指该任务优先级最高并正在运行 .

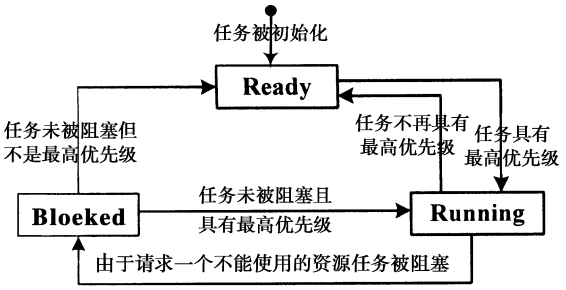

图 1 实时任务的状态图

Fig 1 The statechart of real-time tasks

调度算法是用以确定所要调度的任务在某一特定时刻执行的规则集 .依据调度顺序产生的时机和方式,分为静态和动态两种 .对于一种调度算法,如果任务的优先级是事先确定的则称为静态调度算法,或固定优先级调度算法;如果任务的优先级可以根据任务请求而改变,则称为动态调度算法 .静态调度算法在任务集的执行之前就已产生一个静态调度表,在运行时根据该调度表决定就绪任务队列中即将执行的任务[7].这类调度算法假设系统实时任务的特性 (如周期、执行时间、最后期限等)是已知的,可调度性分析是离线进行的,最典型的算法就是单调速率调度 (RMS)算法;动态调度算法是在运行期间才决定选择哪个就绪任务来执行,它所根据的是目前已处于就绪态的各个任务的相关属性 (如期限、松弛时间等),以此来决定当前的调度任务序列,典型的算法就是期限最近者优先调度 (EDF)算法 .一个系统是否可调度,即是指该系统的任务集中的每个任务能否在各自的最后期限内完成 .

2.1 单调速率调度 (RMS)

单调速率调度算法由 Liu和 Layland于 1973年提出,是首先为实时系统开发的用于可抢占的实时周期性任务的静态调度算法,已成为其他静态调度算法的应用研究基础 .

RMS算法可调度性判定的充分条件:

定理 1 设一任务集 $S = \{\tau_1, \tau_2, \cdots, \tau_n\}$,各任务的周期为 $T_1, T_2, \cdots, T_n$,执行时间为 $c_1, c_2, \cdots, c_n$,任务集 S的处理器利用率为 U,S可被 RMS调度,如果

$$U = \sum_{i=1}^{n} (c_i / T_i) \leq n(2^{1/n} - 1). \qquad (1)$$

Liu和 Layland在文献 [8]中证明了该算法是最优的,即对于在任何其他静态优先级算法下可调度的任务集,在 RMS算法下也是可调度的 .

定理 1是一个充分条件,对于一个任务集 S,若 n个任务的处理器利用率

$$U \leq n(2^{1/n} - 1),$$

则这 n个任务一定可调度;如果 $U \geq 1$则这 n个任务一定不可调度;如果

$$n(2^{1/n} - 1) < U \leq 1,$$

则定理 1并不能判定这 n个任务是否可调度 .

RMS算法可调度性判定的充要条件:

定理 2 设一任务集 $S = \{\tau_1, \tau_2, \cdots, \tau_n\}$,各任务的周期为 $T_1, T_2, \cdots, T_n$,且 $T_n > T_{n-1} > T_1$,执行时间为 $c_1, c_2, \cdots, c_n$,S可被 RMS调度,当且仅当对每一任务 τ_i: $\forall i, 1 \leq i \leq n$,

$$\min_{(k,l) \in R_i} \sum_{j=1}^{i} c_j \frac{1}{lT_k} \left\lceil \frac{lT_k}{T_j} \right\rceil = \min_{(k,l) \in R_i} \sum_{j=1}^{i} U_j \frac{T_j}{lT_k} \left\lceil \frac{lT_k}{T_j} \right\rceil \leq 1, \qquad (2)$$

其中,$R_i = \{(k, l) \mid 1 \leq k \leq i, l = 1, \cdots, \lfloor T_i / T_k \rfloor\}$.

由于在系统分析、设计阶段,任务的执行时间 (CPU执行时间)一般是估计值,首选定理 1进行可调度性判定 .如果不能满足最坏情况上限,再使用其他的调度算法 (如定理 2)做进一步的判定 .

2.2 期限最近者优先调度 (EDF)

期限最近者优先调度算法,以最后期限的顺序指定优先级,优先级最高的任务是距离最后期限最近的任务 .因此在每一个任务结束时,余下任务的优先级须重新计算 .同 RMS算法较低的处理器利用率 (上限 69%)相比,EDF可以达到 100%的处理器利用率,并且如果 EDF系统过载或错过了最后期限,在错过期限之前,它将以 100%的负荷运行 .

EDF算法可调度性判定的充要条件:

定理 3 设一任务集 $S = \{\tau_1, \tau_2, \cdots, \tau_n\}$,各任务的周期分为 $T_1, T_2, \cdots, T_n$,执行时间为 $c_1, c_2, \cdots, c_n$,S可被 EDF调度,当且仅当

$$U = \sum_{i=1}^{n} (c_i / T_i) \leq 1. \qquad (3)$$

EDF作为一种动态优先级调度算法是最优的,即对于在任何其他动态优先级算法下可调度的任务集,在 EDF算法下也是可调度的 .

2.3 RMS和 EDF比较

RMS:调度决策运行开销小,可预测性强,算法实现简单 .该算法适合于问题需求确定,并且运行中不会有较大变化的情况 .但是当处理器负载增加的时候,优先级较低的任务偶尔会超时,灵活性相对差些,不适合动态变化的,或不可预测环境下的多任务调度 .CPU利用率虽然较低,但它可以保证所有进程的最后期限都可满足 .权威的《实时系统》杂志 1997年第 3期的一份调查报告表明,当时所占市场份额比例较高的 50个实时操作系统中有 41个都支持这种算法 .本文所采用的即是这种算法 .事实证明,这

种固定优先级的做法可满足许多场合下实时多任务的调度.

EDF:处理器利用率可达 100%,算法比较灵活,能够应对变化的运行环境,适合于任务不断生成,且在任务生成之前,其特性并不太清楚的动态实时系统.但是,相对于静态优先级算法,由于任务的优先级不确定,不易协调相互关联的多个任务.算法实现比 RMS复杂得多,调度决策的运行开销相对较大,且在系统发生瞬间过载时,难以保证某些关键实时任务的时间约束要求,很难诊断出即将出现过载的可能性.当系统负载相对较低时,该算法非常有效,但在系统负载较重时,系统性能会急剧下降,引起大量任务错过其最后期限,甚至可能导致 CPU执行时间大量花费在调度决策上.

3 计算机数控系统可调度性分析

本节将以计算机数控系统 (CNC) 作为实时系统基于 UML模型进行可调度性分析的一个实例.CNC作为一种典型的实时系统,对其多任务进行调度的目的就是得到一个多任务的最佳执行顺序,当它们按照此顺序执行时能满足它们各自的时间约束条件,使得每个实时任务能够在其最后期限内完成.具体做法如下:首先使用 UML实时特征文件包中的构造型 (Stereotype)、标签值 (Tag Value) 将所要进行可调度性分析的多任务实时信息 (QoS) 在 CNC UML模型中标识出来.然后通过 Visual Basic Application将这些信息提取出来,在实时可调度性分析工具 RapidRMA 中进行分析.分析结果则自动反馈至 CNC UML模型中,并为开发者反馈指导性的建议,以对所开发实时系统进行修正,其过程如图 2所示.

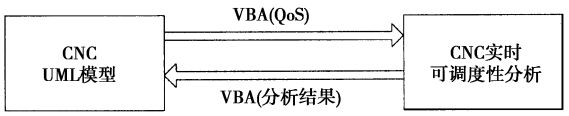

图 2 CNC UML模型实时可调度性分析过程

Fig. 2 The schedulability analysis process of CNC UML models

该方法的应用可以在系统分析、设计、实现等多个阶段对实时系统可调度性进行分析.而模型驱动开发方法可以实现在 UML模型层对待开发实时系统的功能进行测试[5],两者相结合可以有效解决使用传统开发方法开发实时系统时所存在功能和性能相脱节的问题.

图 3是 CNC自动操作模型测试时用以解释 G-code的解释器对象和用以协调多个运动轴的轴组对象间的交互过程,这两个对象分别对应着 CNC的 interpreter task, axisGroup task. 在进行可调度性分析前首先在图 3中将每个任务的相关实时信息 (QoS) 标识出来.以解释器任务为例:该任务包含 evContiStart() 事件、interpret() 子动作和 outData(…) 子动作.outData(…) 的执行时间为 3 ms,其标识为构造型《Local: : SAAction》,标签值为 RTduration = (3, ‘ms’);与此类似 interpret () 的执行时间为 4 ms, evContiStart() 的执行时间为 2 ms,此外还需要标识出解释器任务的期限 100 ms: SAAbsDeadline = (100, ‘ms’);该任务作为一个单独线程的名称: SAHost = Thread1;该任务执行的处理器 (节点) 载体: SAExecutionEngine = X86;该任务的执行周期 100 ms: SAOccurrencePattern = (100, ‘ms’).标识完后启动 RapidRMA 工具并进行相应的设定之后,就可以对 CNC系统实时多任务进行可调度性分析.分析结果会自动反馈至 UML模型中,如图 3所示.由于《SAAction》RTduration是指包含子动作序列的总计执行时间,因此解释器任务的执行时间为 evContiStart事件、interpreter() 子动作和 outDate(…) 子动作的执行时间之和: 2 + 4 + 3 = 9 ms;解释器任务的最坏情况执行时间为 42 ms: SAWorstCase = (42, ‘ms’); 被阻塞时间为 14 ms: SABlocking = (14, ‘ms’); 任务的优先级为 253: SAPriority = 253,该优先级为映射到目标平台下实时操作系统的优先级,其范围和升/降序含义可以在可调度性分析前视具体情况进行设定.该任务的可调度性为真:《SATrigger》SASchedulable = True,说明该任务能够在其最后期限内完成.同样对 CNC系统的其它实时任务进行可调度性分析即可确定所开发的实时系统是否可调度.

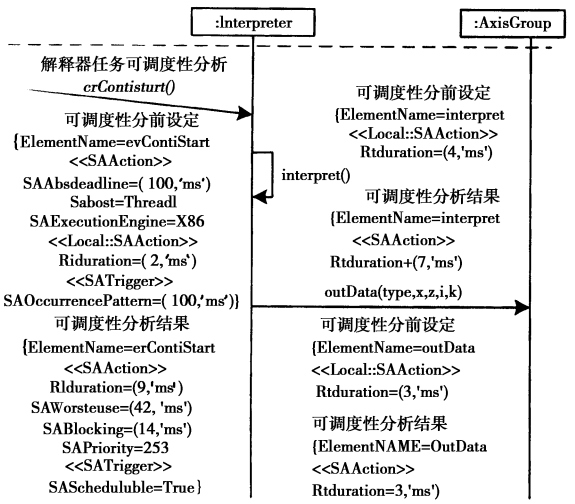

图 3 实时任务可调度性分析 UML模型序列图

Fig. 3 UML model sequence diagram of real-time task schedulability analysis

4 结束语

本文在探讨当前实时系统的开发现状和实时系统可调度性分析相关理论的基础之上,提出了一种基于实时统一

建模语言对实时系统的可调度性进行离线分析的方法 .并在典型的实时系统 ——计算机数控系统中进行了验证 .实践证明,本文所提出的方法结合我们在文献 [5]中提出的模型驱动开发方法可以有效地解决当前实时系统开发中所存在的功能和性能脱节的问题 .

参考文献:

[1] JEROME L KRASNER. *Reducing OEM development costs and enabling embedded design efficiencies using the UML2. 0*[R]. American Technology International, Inc. 2004.

[2] BRUCE POWEL DOUGLASS. *Doing hard time: developing real-time systems using UML, objects, frameworks, and patterns* [M]. Reading, MA: Addison-Wesley, 1999.

[3] PAULO MARTINS. *Integrating real-Time UML models with schedulability analysis*[EB]. Tri-Pacific Software Inc. http://www. tripac. com.

[4] BRAN SELIC. *Model-Driven development of real-time software using OMG standards*[C]// Proceedings of the Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC '03).

[5] 高军礼. 基于模型驱动开发方法的开放式结构计算机数控系统的研究[D]. 广州:华南理工大学博士学位论文, 2005. 6.

[6] QING LI, CAROLINE YAO. 嵌入式系统的实时概念[M]. 王安生,译. 北京:北京航空航天大学出版社, 2004: 6.

[7] 王永吉, 陈秋萍. 单调速率及其扩展算法的可调度性判定[J]. 软件学报, 2004, 15(6): 799-814.

[8] LIU C L, JAMES W Layland. *Scheduling algorithm for multiprogramming in a Hard-Real-Time environment*[J]. Journal of the Association for Computing Machinery(S 0004-5411), 1973, 20(1): 46-61.

Schedulability Analysis of Real-Time System Based on RT-UML Models

GAO Jun-li[1], LI Di[2], ZHENG Shi-xiong[2]

(1. Department of Automation, Guangdong Univ. of Technology, Gunagzhou 510006, China;
2. Department of Mechanical Engineering, South China Univ. of Technology, Gunagzhou 510006, China)

Abstract: The actuality of real-time system development is reviewed and the correlative theories for real-time system schedulability analysis are discussed. One method for real-time system schedulability analysis is presented based on real-time unified modeling language. We have realized the off-line real-time schedulability analysis before system implementation. Through distilling the correlative quantities information of real-time task in the system real-time UML models and analysing by the corresponding tool, the result can be fed back to the models automatically.

Key words: real-time unified modeling language; schedulability analysis; real-time system

责任编校:郭红建