软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(2):481−494 [doi: 10.13328/j.cnki.jos.005312]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

多核环境下基于图模型的实时规则调度方法*

王娟娟[1,2], 乔 颖[1], 熊金泉[3], 王宏安[1]

[1](中国科学院 软件研究所,北京 100190)

[2](中国科学院大学,北京 100049)

[3](南昌师范学院 数学与计算机科学系,江西 南昌 330032)

通讯作者: 王娟娟, E-mail: wjuanj89@126.com

摘 要: 安全攸关反应式系统的核心要求是:必须在指定时间期限内完成对外部事件的检测和目标事件的响应,否则会产生灾难性的后果.随着安全攸关反应式系统对智能化需求的日益增加,将规则推理应用于这类系统成为必然趋势.规则调度是保证规则推理硬实时约束的关键.为此,提出了一种基于图模型的实时规则调度方法(graph-based real-time rule scheduling,简称 GBRRS).该方法对基于事件图的实时规则推理过程进行建模,提出了基于图的端到端推理任务模型,并给出了端到端推理任务的调度算法,保证了规则调度的安全性.采用模拟实验对 GBRRS 方法进行了验证,实验结果表明,与 DM-EDF 方法(通过直接映射把规则上的推理操作转成推理任务后,用全局 EDF 算法对其进行调度的方法)相比,GBRRS 方法在规则调度成功率上平均高出 13%~15%,且在规则集的平均负载较高时,仍保持着 80%以上的调度成功率.

关键词: 多核;安全攸关;实时推理;规则推理;规则调度

中图法分类号: TP316

中文引用格式: 王娟娟,乔颖,熊金泉,王宏安.多核环境下基于图模型的实时规则调度方法.软件学报,2019,30(2):481−494. http://www.jos.org.cn/1000-9825/5312.htm

英文引用格式: Wang JJ, Qiao Y, Xiong JQ, Wang HA. Method for graph-based real-time rule scheduling in multi-core environment. Ruan Jian Xue Bao/Journal of Software, 2019,30(2):481−494 (in Chinese). http://www.jos.org.cn/1000-9825/5312.htm

Method for Graph-based Real-time Rule Scheduling in Multi-core Environment

WANG Juan-Juan[1,2], QIAO Ying[1], XIONG Jin-Quan[3], WANG Hong-An[1]

[1](Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](University of Chinese Academy of Sciences, Beijing 100049, China)

[3](Department of Mathematics and Computer Science, Nanchang Normal University, Nanchang 330032, China)

Abstract: Safety-critical systems detect external events, match the targeted event patterns, and give timely responding actions; otherwise catastrophic results will be incurred. With the increasing demand for intelligence in the safety-critical systems, applying rule-based reasoning to these systems has become an inevitable trend. Besides, rule scheduling is the key to assure hard real-time constraints within rule-based reasoning solutions. In this study, a solution to the multi-core rule scheduling problem, named GBRRS (graph-based real-time rule scheduling), was proposed. With the real-time rule reasoning process analyzed, how rules in safety-critical systems can be modeled as tasks using the graph mapping is described first, and the graph-based end-to-end reasoning task model, E2ERTG, is proposed. Then, a multi-core scheduling algorithm, GBRRS, is presented to guarantee each rule's deadline via the control of the reasoning task's deadline. Simulation- based experiments have been conducted to evaluate the performance of GBRRS. The result shows that GBRRS

* 基金项目: 国家自然科学基金(61562063)
Foundation item: National Natural Science Foundation of China (61562063)
收稿时间: 2017-01-21; 修改时间: 2017-03-30, 2017-05-02; 采用时间: 2017-05-30; jos 在线出版时间: 2017-07-12
CNKI 网络优先出版: 2017-07-12 15:35:16, http://kns.cnki.net/kcms/detail/11.2560.TP.20170712.1535.012.html

remains a rule success ratio above 80% even with relatively high workload of the rule set and is superior to DM-EDF by average 13%~15% in terms of rule success ratio.

Key words: multi-core; safety-critical; real-time reasoning; rule reasoning; rule scheduling

实时反应式系统[1]被广泛应用在医疗、工业、军事、通信、交通等安全攸关领域.这类系统有主动行为,其运行受外部事件驱动.系统通过一系列传感器采集外部环境数据,对连续不断的事件流进行监视,从中识别出需要关注的场景(该场景是由一系列事件的发生所喻示的),实时地对识别出的场景做出响应.随着实时反应式系统应用环境的日趋复杂,人们对这类系统智能化的需求也越来越高,这就要求复杂反应式系统必须具有强大的推理能力,能够根据外部环境中发生的事件,自动决策如何对这些事件进行响应.因此,将规则推理系统应用于实时反应式系统就成为必然趋势.

实时反应式系统通常是安全攸关系统,具有硬实时约束.即当某些需要关注的事件发生后,系统必须在给定截止期内完成相应的动作,对这些事件进行响应;否则会产生严重后果.这就要求规则推理需要具有时间约束.学者们为此提出了实时推理方法,如迭代推理[2](如 Anytime 算法)、多重方法推理[3](如 Design-to-Time 算法)和渐进式推理[4,5](如 GREAT 算法和 PRIMES 算法).这些方法为整个推理过程定义了截止期约束,通过对推理运行时间与推理结果质量进行折中来满足这个截止期约束.此外,学者们还通过改进传统规则匹配算法 RETE 缩短了 OPS5 产生式系统的匹配时间,从而降低了整个专家系统的响应时间[6,7].同时,李想等人[8,9]提出了估算实时规则推理程序最大响应时间的方法.李娜等人[10,11]针对实时感知数据流,为复杂事件检测的整个过程引入了截止期,并提出启发式调度算法对检测过程进行调度,缩短了复杂事件检测的平均时间.由于上述研究忽略了各规则不同的紧迫度与资源需求,很可能会在资源受限的情况下获得不可接受的推理结果质量.

为此,我们在前期工作中提出了一种新的实时推理模式[12]:实时规则推理.在实时规则推理中,我们为每条规则(规则的形式为:IF ⟨*EVENT PATTERN*⟩ THEN ⟨*ACTION*⟩.其语义为:若⟨*EVENT PATTERN*⟩定义的目标事件发生,则执行⟨*ACTION*⟩定义的动作)引入了截止期(规则截止期为该规则响应时间的上界(规则响应时间是指从与规则相关的所有原子事件实例到达系统开始到响应动作执行完毕为止的时间间隔)),并提出了软实时规则调度算法[13,14].但这些方法只是尽可能地使规则在其截止期前执行完毕,无法预测实时规则推理对资源的动态需求,也就难以保证各规则的截止期,从而无法满足安全攸关系统的硬实时约束.规则调度是保证规则截止期的关键.因此,本文将对该问题进行研究.具体来说,规则调度问题(rule scheduling problem,简称 RSP)是找到一种实时调度策略,能够合理安排各规则的匹配与执行顺序,使得每条规则的响应时间不超过其截止期.

由于处理数据规模的不断增大和计算复杂度的持续上升,安全攸关系统需要运行在计算能力更强、更有弹性的多核环境下.多核计算具有更强的并行处理能力、更高的计算密度,同时具有更低的时钟频率,减少散热和功耗,因此,研究在多核环境下安全攸关系统的规则调度问题十分必要(多核环境是指“单机多核”,也就是说,安全攸关系统运行在单机上,只有 1 个处理器,但该处理器包括了多个处理器核).为了解决多核环境下安全攸关系统中的规则调度问题,本文提出了一种基于图模型的实时规则调度(graph-based real-time rule scheduling,简称 GBRRS)方法.该方法首先对基于事件图的推理过程进行建模,将规则调度问题转化成基于图的端到端推理任务调度.并给出了多核环境下端到端推理任务的调度算法,一方面通过调度端到端推理任务来合理安排规则匹配与执行的顺序;另一方面,通过提出准入控制算法,对推理过程所需资源进行预测,以保证每一条触发规则都在其截止期之前完成(若规则⟨*EVENT PATTERN*⟩部分所定义的目标事件发生,则该规则被称为触发规则),从而保障了多核环境下规则推理的实时性,进而保证了安全攸关系统的硬实时约束.

本文第 1 节对面向安全攸关反应式系统的实时规则推理系统和基于事件图的实时规则推理过程加以描述.第 2 节给出多核环境下基于图模型的实时规则调度方法.第 3 节的模拟实验对 GBRRS 方法的性能进行验证.第 4 节对全文进行总结,介绍下一步的研究工作.

1 实时规则推理系统

本节将描述面向安全攸关反应式系统的实时规则推理系统,重点介绍其系统结构与实时规则推理引擎所采用的实时规则推理过程.

1.1 实时规则推理系统结构

面向安全攸关反应式系统的实时规则推理系统为 $S=(R,E,I)$,其系统结构[15]如图 1 所示.其中,R 是规则库,存放可描述专家知识的规则集 $R=\{R_1,R_2,\ldots,R_n\}$,每条规则 $R_i(1\leqslant i\leqslant n)$具有截止期 D_i;E 是由外部环境产生并进入实时规则推理系统的原子事件实例所形成的事件流,$E=\{e_1,e_2,\ldots,e_k,\ldots\}$.事件检测器将事件流中被规则 $R_i(1\leqslant i\leqslant n)$涉及的原子事件实例输入实时规则推理引擎 I.

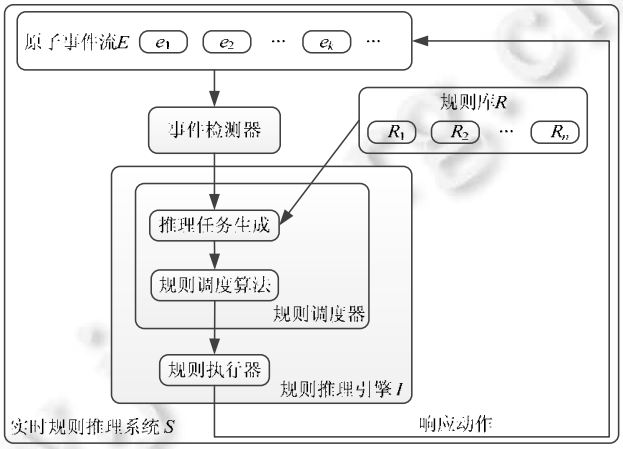


Fig.1 Real-time rule reasoning system

图 1 实时规则推理系统

实时规则推理引擎 I 根据输入的事件流,从规则库中匹配出相应的规则(规则匹配:给定规则集 $R=\{R_1,\ldots,R_n\}$,根据到达系统的原子事件流,对 R 中每条规则 R_i,判断其所定义的目标事件是否发生),并在其截止期前执行该规则所定义的响应动作指令(简称为规则执行),其核心包括规则调度器与规则执行器.规则调度器运行相应的规则调度方法将基于事件图的实时规则推理过程转化为一组并发的实时推理任务,并对这组推理任务进行调度;规则执行器负责执行被调度器选中的推理任务.这样,实时规则推理引擎即可通过合理安排规则匹配与执行的顺序来保证规则的截止期,使安全攸关反应式系统可以在截止期前对其所关注的场景做出响应.本文将对规则调度器所运行的规则调度方法进行研究.第 1.2 节将介绍基于事件图的实时规则推理过程.

1.2 基于事件图的实时规则推理过程

在基于事件图的推理过程[16]中,规则集 $R=\{R_1,R_2,\ldots,R_n\}$可用有向无环图(directed acyclic graph,简称 DAG)来描述.该有向无环图称为规则图 G_R.$G_R=(V,A,E)$,其中,V 表示事件节点(包括原子事件和复合事件),A 表示响应动作指令的节点(简称为动作节点),E 表示节点间时序关系(方向代表了时序先后)的有向边.

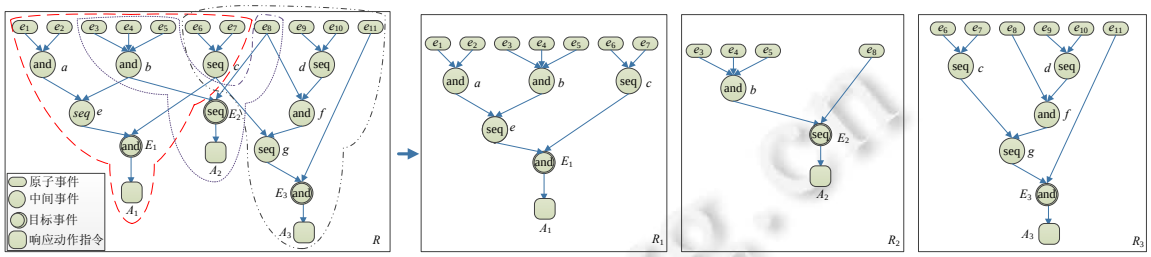
若存在规则集 $R=\{R_1,R_2,R_3\}$,其中,

- $R_1:(((e_1\wedge e_2)\rightarrow(e_3\wedge e_4\wedge e_5))\wedge(e_6\rightarrow e_7))\rightarrow A_1$;
- $R_2:((e_3\wedge e_4\wedge e_5)\rightarrow e_8)\rightarrow A_2$;
- $R_3:(((e_6\rightarrow e_7)\rightarrow(e_8\wedge(e_9\rightarrow e_{10})))\wedge e_{11})\rightarrow A_3$,

则 R 所对应的规则图 G_R 如图 2(a)所示,其中,没有直接前驱的节点是源节点,表示原子事件(如 $e_1,\ldots,e_{11}$),也称原子节点;没有直接后继的节点是终节点,表示响应动作指令(如 A_1、A_2、A_3),也称动作节点;其他节点为中间节点,表示复合事件;原子节点和中间节点都表示事件,也称事件节点;动作节点的父节点表示目标事件(如 E_1、E_2、E_3),

称为目标节点.

在规则图 G_R 中,事件节点的入度表示了组成该节点的事件个数.如图 2(a),事件节点 $b=e_3\wedge e_4\wedge e_5$,即 e_3~e_5 通过事件复合形成了 b,则 b 的入度是 3;事件节点 $e=a\rightarrow b$,则 e 的入度是 2.目标节点以外的事件节点,其出度表示了该节点参与生成复合事件的次数.若 A 是 G_R 中的动作节点,且表示规则 $R_i(1\leqslant i\leqslant n)$所定义的动作,那么规则 R_i 的规则子图是 A 可达的所有节点(包括 A 在内)组成的子图,如图 2(b)~图 2(d)所示.规则的高度是其规则子图从原子节点到动作节点的最大层数,如图 2 所示,R_1、R_2 和 R_3 的高度分别为 5、4 和 6.



(a) (b) (c) (d)

Fig.2 Rule graph

图 2 规则图

基于事件图的推理过程是以到达系统的原子事件实例为依据,通过对规则图的搜索及规则子图的匹配来完成的.该推理过程的具体步骤如下.

- 步骤 1:原子事件实例到达系统后按照时间先后被送往规则图中的原子节点.
- 步骤 2:原子节点接收原子事件实例后,将对其事件类型进行匹配,而后将匹配结果送往其所有直接后继的事件节点.
- 步骤 3:目标节点以外的事件节点在接收到送来的事件实例后,将根据事件复合的算法[17−19]处理这些事件实例并进行匹配,以判断当前节点的事件实例是否生成:若匹配成功,将生成新的事件实例,并把该实例送往其直接后继的事件节点;若匹配失败,则转到步骤 2 接收新的原子事件实例进行匹配.
- 步骤 4:若当前的事件节点是目标节点,且该节点的事件实例复合并匹配成功,则将生成的事件实例送往动作节点;若匹配失败,则转到步骤 2 接收新的原子事件实例进行匹配.
- 步骤 5:如果动作节点成功接收到目标节点送来的事件实例,则输出响应动作指令,表明以该目标节点为终节点的规则子图已匹配完成;否则,转到步骤 2 接收新的原子事件实例进行匹配.
- 步骤 6:重复步骤 2~步骤 5,直至所有的原子事件实例都被处理完毕.

可以看出,规则图 G_R 中每个节点上的推理操作是接收一次事件实例,对其进行匹配并向其直接后继节点传输;每条规则上的推理操作是对规则图 G_R 中相应规则子图的一次匹配.

2 多核环境下基于图模型的实时规则调度方法

本节将重点描述如何解决多核环境下规则调度问题的难点:(1) 如何对基于事件图的推理过程对规则调度问题进行建模,即如何将实时规则推理过程转化成基于图的端到端推理任务;(2) 如何为推理任务合理安排执行顺序,并提出准入控制算法,对推理过程中所需资源进行预测,保证每条规则一旦被处理都能在其截止期前完成,从而保障了安全攸关系统中规则的实时性.

为此,本节将给出一种基于图模型的实时规则调度方法.该方法首先通过图映射方法为规则调度建立基于图的端到端推理任务模型,并提出多核环境下的推理任务调度算法,从而解决多核环境下的规则调度问题.

2.1 实时规则推理过程建模

实现规则上的推理操作与推理任务之间的映射有两种方法:一是直接映射,简称 DM;二是图映射(graph-

based mapping),简称 GM.

在 DM 方法中,把每条规则 $R_i(R_i\in R)$上的推理操作直接映射成推理任务 τ_i,规则集的推理过程就转化成推理任务集 $\tau=\{\tau_1,\ldots,\tau_n\}$,$\tau_i=(r_i,C_i,D_i)$.其中,$\tau_i$ 的参数如下.

r_i 是 τ_i 的就绪时间,即 R_i 相关联的所有原子事件实例均到达系统的时间;C_i 是 τ_i 的执行时间,其数值可通过规则匹配的最坏执行时间分析技术来获得[20];D_i 是 τ_i 的相对截止期,等于 R_i 的相对截止期(可由应用给定).

此外,s_i 是 τ_i 开始执行的时间;$C_i(t)$是 t 时刻 τ_i 的剩余执行时间;f_i 是 τ_i 的完成时间,$f_i=\min\{t|C_i(t)=0\}$;RS_i 是 τ_i 的响应时间,$RS_i=f_i-r_i$,$RS_i\leqslant D_i$;d_i 是 τ_i 的绝对截止期,$d_i=r_i+D_i$,$f_i\leqslant d_i$.

从第 1.2 节可知,基于事件图的实时规则推理过程是动态的.一条规则的匹配有可能因为其规则子图上的任一节点匹配不成功而终止,其运行时间无法事先预测.而 DM 方法中,只能使用规则匹配的最坏执行时间作为推理任务的执行时间,这过度估计了推理任务所需资源,从而降低了推理任务的调度成功率,导致了规则调度成功率的下降.为了更为精确地对动态的实时规则推理过程进行描述,本文提出了 GM 方法,将实时规则推理过程转化为一组基于图的端到端推理任务(end-to-end reasoning task graph,简称 E2ERTG).

在 GM 方法中,把每条规则 $R_i(R_i\in R)$上的推理操作映射成一个端到端推理任务 τ_i,把规则子图中每个节点上的推理操作定义成 τ_i 的子任务$\{\tau_{i,1},\ldots,\tau_{i,j},\ldots\}$,则规则集的推理过程就转化成推理任务集 τ.如图 3 所示,R_1~R_3所对应的端到端推理任务为 τ_1~τ_3,其中,$\tau_1=\{\tau_{1,1},\ldots,\tau_{1,13}\}$,$\tau_2=\{\tau_{2,1},\ldots,\tau_{2,7}\}$,$\tau_3=\{\tau_{3,1},\ldots,\tau_{3,12}\}$.

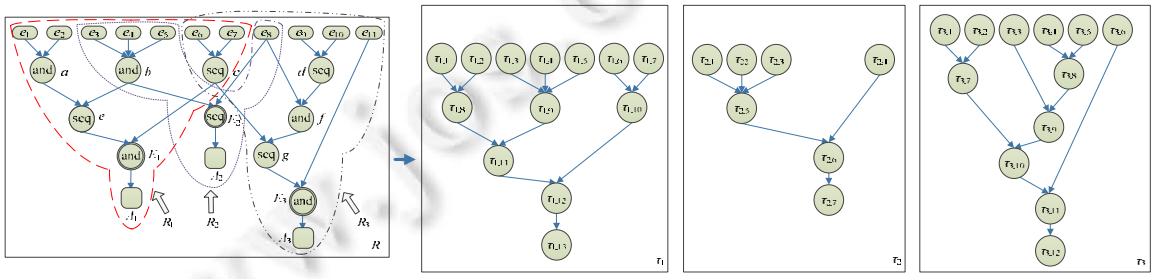


Fig.3 Graph-based mapping

图 3 图映射

推理任务 $\tau_i=(D_i,\{\tau_{i,j}|1\leqslant j\leqslant n_i\},G_i)$有以下参数:$D_i$ 是 τ_i 的相对截止期;$\{\tau_{i,j}|1\leqslant j\leqslant n_i\}$是 τ_i 的子任务集合;G_i 是 τ_i 的子任务图,是个有向无环图,代表了 n_i 个子任务之间的偏序关系.

在 G_i 中,若存在有向边从 $\tau_{i,u}$ 指向 $\tau_{i,v}$,则 $\tau_{i,v}$ 是 $\tau_{i,u}$ 的直接后继,$\tau_{i,u}$ 是 $\tau_{i,v}$ 的直接前驱.$\tau_{i,j}$ 的直接前驱集合记为 $preddt(\tau_{i,j})$,其直接后继集合记为 $succdt(\tau_{i,j})$.若存在 $\tau_{i,u}$ 到 $\tau_{i,v}$ 的路径,则 $\tau_{i,v}$ 称为 $\tau_{i,u}$ 的后继,$\tau_{i,u}$ 称为 $\tau_{i,v}$ 的前驱.$\tau_{i,j}$ 的前驱集合记为 $pred(\tau_{i,j})$,其后继集合记为 $succ(\tau_{i,j})$.在 G_i 中,没有直接前驱的节点是源节点,其代表的子任务称为源任务;没有直接后继的节点是终节点,其代表的子任务称为终任务;其余节点是中间节点,其代表的子任务称为中间任务.

值得注意的是,不同端到端推理任务的子任务可能执行相同的操作.如图 3 所示,τ_1 的子任务 $\tau_{1,3}$ 和 τ_2 的子任务 $\tau_{2,1}$ 都执行的是对事件 e_3 的推理操作;τ_1 的子任务 $\tau_{1,9}$ 和 τ_2 的子任务 $\tau_{2,5}$ 均执行的是对事件 b 的推理操作.

定义 1. 设 $\tau_i,\tau_{i+1},\ldots,\tau_j$ 为一组端到端推理任务,若其子任务 $\tau_{i,p},\tau_{i+1,l},\ldots,\tau_{j,q}$ 所执行的推理操作相同且任务参数也相同,则称 $\tau_{i,p},\tau_{i+1,l},\ldots,\tau_{j,q}$ 互为关联的子任务.

显然,根据第 1.2 节的实时规则推理过程,在一组互为关联的子任务中,若其中任一子任务执行完毕并获得计算结果,则其余子任务便可获得相同的执行结果.因此,在调度中只需为其中任意一个子任务分配处理器资源即可.为此,本文引入关联节点来表示互为关联的所有子任务,形成了推理任务集 τ 的推理任务图 G_τ,如图 4 所示.

定义 2(关联子任务集 Ψ). 设 V 是 G_τ 中的关联节点集合(节点个数是 c),$V_r(1\leqslant r\leqslant c)$是 V 中的关联节点,则 $\psi(r)$是 V_r 所表示的推理子任务集合,且 $\Psi=\bigcup_{1\leqslant r\leqslant 8}\psi(r)$.

如图 4 所示,V_r={e_3,e_4,e_5,b,e_6,e_7,c,e_8},其中,$\psi(1)$={$\tau_{1,3}$,$\tau_{2,1}$},$\psi(2)$={$\tau_{1,4}$,$\tau_{2,2}$},$\psi(3)$={$\tau_{1,5}$,$\tau_{2,3}$},$\psi(4)$={$\tau_{1,9}$,$\tau_{2,5}$},$\psi(5)$={$\tau_{1,6}$,$\tau_{3,1}$},$\psi(6)$={$\tau_{1,7}$,$\tau_{3,2}$},$\psi(7)$={$\tau_{1,10}$,$\tau_{3,7}$},$\psi(8)$={$\tau_{3,3}$,$\tau_{2,4}$},则

$$\Psi=\bigcup_{1\leqslant r\leqslant 8}\psi(r)=\{\tau_{1,3},\tau_{1,4},\tau_{1,5},\tau_{1,6},\tau_{1,7},\tau_{1,9},\tau_{1,10},\tau_{2,1},\tau_{2,2},\tau_{2,3},\tau_{2,4},\tau_{2,5},\tau_{3,1},\tau_{3,2},\tau_{3,3},\tau_{3,7}\}.$$

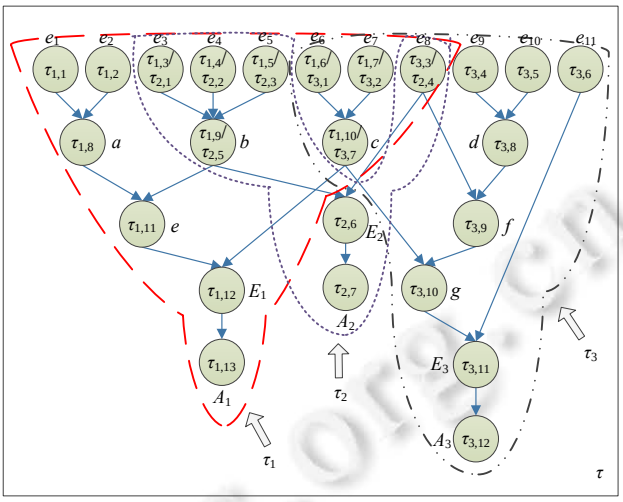


Fig.4 Reasoning task graph

图 4 推理任务图

在推理任务图 G_τ中,每个推理任务$\forall\tau_i(i\geqslant 1)$及其子任务$\tau_{i,k},k\in\{1,2,\ldots,j\}$的参数见表 1.其中,$s_i$,$f_i$,$D_i$,$d_i$ 和 $C_i(t)$可参见 DM 方法中的相关定义.$s_{i,k}$,$f_{i,k}$,$d_{i,k}$,$C_{i,k}(t)$的定义与 s_i,f_i,d_i 和 $C_i(t)$的定义类似.在此不再赘述.

Table 1 Parameters of tasks and sub-tasks

表 1 任务及子任务参数

	就绪时间	开始执行时间	完成时间	相对截止期	绝对截止期	最坏执行时间	剩余执行时间
τ_i	r_i	s_i	f_i	D_i	d_i	C_i	$C_i(t)$
$\tau_{i,k}$	($r_{i,k}$)	$s_{i,k}$	$f_{i,k}$	/	($d_{i,k}$)	$C_{i,k}$	$C_{i,k}(t)$

若$\tau_{i,k}$ 是源任务,则 $r_{i,k}$ 是原子事件实例到达其相应源节点的时间(即原子事件实例到达系统的时间);否则,$r_{i,k}$ 是其所有直接前驱任务的最晚完成时间,即 $r_{i,k}=\max\{f_{i,j}\},\forall\tau_{i,j}\in preddt(\tau_{i,k})$.$r_i$ 是所有源任务的最晚就绪时间,即 $r_i=\max\{r_{i,k}\},\tau_{i,k}$是源任务.$C_{i,k}$是$\tau_{i,k}$在 G_τ中相应节点上推理操作的执行时间;C_i是τ_i的最坏执行时间,即τ_i所有推理子任务的最坏执行时间之和: $C_i=\sum C_{i,j}$. 若$\tau_{i,k}$为终任务,则有 $d_{i,k}=d_i$,$f_{i,k}=f_i$.

显然,通过 GM 方法建立基于图的端到端推理任务模型 E2ERTG,可将多核环境下的规则调度问题 RSP 转化为基于图的端到端推理任务调度问题,即:

给定 m 个处理器核 $P=\{P_1,P_2,\ldots,P_m\}$及推理任务集$\tau=\{\tau_1,\tau_2,\ldots,\tau_n\}$,其中,$\tau_i=(D_i,\{\tau_{i,j}|1\leqslant j\leqslant n_i\},G_i)$;任意子任务$\tau_{i,j}(\tau_{i,j}\in\tau_i,\tau_i\in\tau)$可被分配到任意的处理器核 $P_i(P_i\in P)$上执行,并允许其在处理器核之间迁移.找到一种实时调度方法,使$\forall\tau_i(\tau_i\in\tau)$,$f_i\leqslant d_i,i\in\{1,2,\ldots,n\}$;其中,$f_i$是端到端推理任务$\tau_i$的完成时间,$d_i$是$\tau_i$的绝对截止期.

为解决上述问题,下一节将给出基于图的端到端推理任务(E2ERTG)的调度算法,通过保证所有的推理任务都在其截止期之前完成执行,来保证规则的截止期.

2.2 推理任务调度算法

多核环境下,对基于图的端到端推理任务进行调度的关键点是:(1) 如何实现推理子任务之间的同步,使其满足子任务间的时序约束;(2) 如何分配推理子任务的全局动态优先级;(3) 如何对新到达的推理子任务进行准入控制,以保证所有准入的推理任务一旦执行都将在其截止期前完成.为此,我们将提出基于图的端到端推理任务调度算法(end-to-end reasoning task graph-based scheduling,简称 ERTGS).

2.2.1 同步协议

同步协议的目的是为了保证推理任务图中子任务之间由有向边指定的时序关系,其策略为:除源任务外的子任务就绪时间等于其直接前继任务的最晚完成时间.即

$$r_{i,q}=\max\{f_{i,p}\},\ \forall \tau_{i,p}<\tau_{i,q} \tag{1}$$

图 5 是实现同步协议的伪代码.对于推理任务图中除终任务外的其他子任务,将其就绪时间初始化为无穷大的整数值;邻接矩阵 *adj_G* 存储的是推理任务图中子任务之间的时序关系.从邻接矩阵 *adj_G* 获取所有子任务的直接后继和直接前驱任务列表(参见图 5 的第 3 行和第 5 行).当任意一个推理子任务完成时,通知该子任务的所有直接后继任务更新其直接前驱任务的完成情况(参见图 5 的第 7 行和第 12 行).$\tau_{i,s}$ 的所有直接前驱任务均完成时,根据公式(1)更新 $\tau_{i,s}$ 的就绪时间(参见图 5 的第 9 行和第 13 行).

```
1:  procedure assprec(adj_G)
2:    if a task τ_{i,j} that completed is not the member of ActionNodes
3:      store the non-zero column to be the succdt elements;
4:      for each member τ_{i,s} of succdt(τ_{i,j})
5:        store the non-zero row to be the preddt elements;
6:        if τ_{i,s} is not the member of ActionNodes
7:          preddtcompletedflag(τ_{i,s})=true;
8:          if allpreddtcompletedflag=true
9:            r_{i,s}=f_{i,j};
10:         end
11:       else
12:         preddtcompletedflag(τ_{i,s})=true;
13:         r_{i,s}=f_{i,j};
14:       end
15:     end
16:   end
17: end procedure
```

Fig.5 Synchronization protocol

图 5 同步协议

2.2.2 优先级分配

ERTGS 根据紧迫度和影响度为每个就绪的推理子任务 $\tau_{i,k}$ 分配全局的动态优先级.

设 $\tau_{i,k}$ 的紧迫度为 $urgency(\tau_{i,k})$,其计算方法如下:

$$urgency(\tau_{i,k})=\begin{cases}1/d_i, & \tau_{i,k}\in\tau_i,\tau_{i,k}\notin\psi\\ 1\Big/\min\limits_{\tau_{j,l}\in\tau_j,\tau_{j,l}\notin\psi(r)}(d_j), & \tau_{i,k}\in\tau_i,\tau_{i,k}\notin\psi(r),\psi(r)\subset\psi\end{cases} \tag{2}$$

由公式(2)可见:对于推理任务图 G_τ 中非关联节点所代表的推理子任务 $\tau_{i,k}$,$urgency(\tau_{i,k})$ 是其所属推理任务 τ_i 绝对截止期 d_i 的倒数;对于 G_τ 中关联节点 V_r 所代表的任意 $\tau_{i,k}$,$urgency(\tau_{i,k})$ 是 $\psi(r)$ 中任意 $\tau_{j,l}$ 所属推理任务 τ_j 绝对截止期的最小值之倒数($\psi(r)$是 V_r 上所有互相关联的推理子任务集合).推理子任务的紧迫度越大,其优先级也越高.

当紧迫度相同时,按其影响度来决定优先级高低.影响度越大,优先级越高.设 $\tau_{i,k}$ 的影响度为 $effect(\tau_{i,k})$,对于 G_τ 中非关联节点所代表的 $\tau_{i,k}$,$effect(\tau_{i,k})$ 是 $\tau_{i,k}$ 在 G_τ 中的出度;对于 G_τ 中关联节点 V_r 所代表的任意 $\tau_{i,k}$,$effect(\tau_{i,k})$ 是 $\psi(r)$ 中推理子任务的个数.$effect(\tau_{i,k})$ 的计算方法如下.

$$effect(\tau_{i,k})=\max\{cnt(succdt(\tau_{i,v}))|\forall\tau_{i,v}\in succ(\tau_{i,k})\cup\{\tau_{i,k}\}\} \tag{3}$$

其中,$cnt(A)$ 是集合 A 中的元素个数.$effect(\tau_{i,k})$ 取值为 $\tau_{i,k}$ 出度和 $succ(\tau_{i,k})$ 出度的最大值.

就绪的推理子任务根据公式(2)、公式(3)分配优先级,按照递减的顺序形成优先级队列.若处理器核出现空闲,则将队首的推理子任务从队列中移除,并分配到该处理器核上执行.若推理子任务执行完毕或者有新的推理任务被准入系统(准入控制的具体方法可参见下小节),则根据公式(2)、公式(3)重新分配各就绪推理子任务的优先级,并更新优先级队列.重复上述操作,直至优先级队列为空.

2.2.3 准入控制

当新的推理任务就绪时,ERTGS 将对其进行准入控制来保证所有推理任务一旦被准入系统,必将在截止期

前完成,即所有被准入的推理任务必定会被成功地调度,从而保证了规则集进行调度的安全性.

定义 3(*t* 时刻的活动任务). 若 τ_a 已就绪,但在 t 时刻未完成且未到其截止期,则 τ_a 是 t 时刻的活动任务.

定义 4(*t* 时刻的活动任务集). 在 t 时刻,所有的活动任务所组成的集合是 t 时刻的活动任务集.

定义 5. 基于图的端到端推理任务集 $\tau=\{\tau_i|1\leqslant i\leqslant n\}$ 在推理任务调度算法 ERTGS 下是可调度的任务集,当且仅当对于 $\forall\tau_i(\tau_i\in\tau)$ 均有 $f_i\leqslant d_i$,f_i 是 τ_i 的完成时间,d_i 是 τ_i 的绝对截止期.

定理 1. 基于图的端到端推理任务集 $\tau=\{\tau_i|1\leqslant i\leqslant n\}$ 在推理任务调度算法 ERTGS 下是可调度的任务集,当且仅当对于 $\forall\tau_{i,j}(i\in[1,n],j\in[1,n_i])$ 均有 $f_{i,j}\leqslant d_i$,$f_{i,j}$ 是 $\tau_{i,j}$ 的完成时间.

证明:如果对于 $\forall\tau_{i,j}(i\in[1,n],j\in[1,n_i])$ 均有 $f_{i,j}\leqslant d_i$,则根据第 2.1 节中所述的推理任务模型,显然有 $f_{i,ni}=f_i$;因此,$\forall\tau_i(\tau_i\in\tau)$ 均有 $f_i\leqslant d_i$,根据定义 3 可知,基于图的端到端推理任务集 τ 在 ERTGS 下是可调度的任务集.

如果基于图的端到端推理任务集 τ 在 ERTGS 下是可调度的任务集,则根据定义 3,对于 $\forall\tau_i(\tau_i\in\tau)$ 均有 $f_i\leqslant d_i$.根据第 2.1 节中所述的推理任务模型,显然有 $f_{i,j}\leqslant f_i$,那么 $f_{i,j}\leqslant d_i$.

综上所述,基于图的端到端推理任务集 $\tau=\{\tau_i|1\leqslant i\leqslant n\}$ 在推理任务调度算法 ERTGS 下是可调度的任务集,当且仅当对于 $\forall\tau_{i,j}(i\in[1,n],j\in[1,n_i])$ 均有 $f_{i,j}\leqslant d_i$,$f_{i,j}$ 是 $\tau_{i,j}$ 的完成时间. □

假设 $\tau(t)=\{\tau_1,\tau_2,\ldots,\tau_u\}$ 是 t 时刻可调度的活动任务集,如果 t 时刻新任务 τ_r 就绪,则准入控制操作如下.

根据定理 1 判断:若 $\{\tau_1,\tau_2,\ldots,\tau_u\}\cup\{\tau_r\}$ 为可调度任务集,则 τ_r 被准入;否则,τ_r 被拒绝.

在此,τ_i 的完成时间可如下计算:

不妨设 $\{\tau_1,\tau_2,\ldots,\tau_v\}$ 是按当前优先级递减排序的活动任务集,处理器核个数为 m,那么,

- 如果 $v<m$,则

$$f_i=t+C_i(t),\forall i\in[1,v] \tag{4}$$

- 如果 $v\geqslant m$,则

$$f_i=\begin{cases}t+C_i(t), & \forall i\in[1,m]\\ \min\limits_{i-m\leqslant q<i}\{f_q\}+C_i(t), & \forall i\in(m,v]\end{cases} \tag{5}$$

其中,公式(5)的迭代计算方法可参见文献[21],在此不再赘述.

ERTGS 的准入控制伪代码如图 6 所示.在此,假设 τ 为可调度的活动任务集,τ_{new} 为就绪的新任务.

```
1:  procedure admissionControl(τ_new)
2:      τ'=τ+τ_new;
3:      flag=isSchedulableTaskset(τ');   /*判断τ'是否可调度任务集*/
4:      if flag is true
5:          admit τ_new;
6:          updateFinishTime(τ');   /*准入后需更新系统中任务集的完成时间情况*/
7:      else
8:          reject τ_new;
9:  end
10: end procedure
11:
12: function isSchedulableTaskset(τ)
13:     calFinishTime(τ);   /*根据公式(4)和公式(5)计算推理任务的完成时间*/
14:     scheflag=true;
14:     for each task τ_i in τ
15:         if f_i>d_i
16:             scheflag=false;   /*根据定理 1 判断τ是否可调度任务集*/
17:         end
18:     end
19:     return scheflag
20: end function
```

Fig.6 Admission control

图 6 准入控制

若处理器核的个数为 m,实时规则的个数为 N,则端到端推理任务的类型也有 N 种,不妨设所需调度的推理

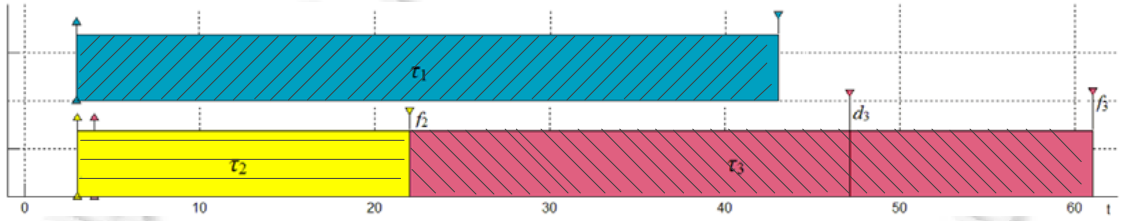
任务实例个数是 n(实时规则可被触发多次而产生多个规则推理任务的实例),基于图的端到端推理任务模型下,推理子任务实例个数总和为 w,那么 DM-EDF 方法时序约束控制的复杂度是 $O(1)$(DM-EDF 方法是用 DM 方法把规则上的推理操作映射成推理任务后用全局 EDF 算法[22]对其进行调度),GBRRS 方法虽然引入图模型,其时序约束控制的复杂度仍是 $O(1)$;DM-EDF 方法优先级排序的复杂度是 $O(n\times\log_2 n)$,GBRRS 方法优先级排序的复杂度是 $O(w\times\log_2 w)$;DM-EDF 方法准入控制的复杂度是 $O(m\times n^2)$,GBRRS 方法准入控制的复杂度是 $O(m\times w^2)$.于是,DM-EDF 方法的复杂度是 $O(1)+O(n\times\log_2 n)+O(m\times n^2)$,而 GBRRS 方法的复杂度是 $O(1)+O(w\times\log_2 w)+O(m\times w^2)$.在此,由于处理器核的个数事先给定,$m$ 为常数;同时,由于规则也是事先给定的,规则图的结构及其所包含的节点数也是固定的,对于每个基于图的端到端推理任务实例来说,其子任务实例数都是它的常数倍,因此,w 最多是 n 的常数倍,引入图后规则调度方法的复杂度并没有本质提高(引入图后规则调度方法能提高 10%以上的规则调度成功率,而这种程度的成功率提高在实际应用中的意义是比较大的,因此还是值得引入图来对推理任务进行建模).

2.3 调度举例

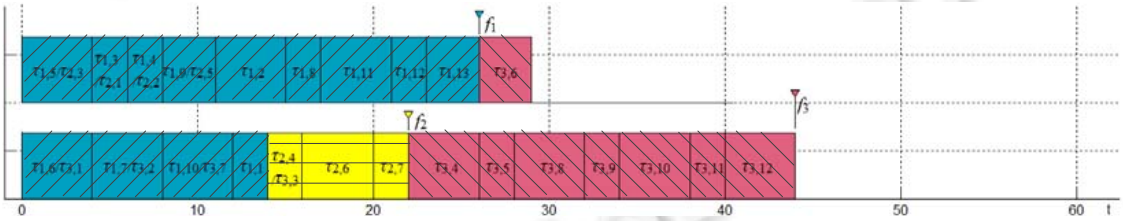
设处理器核的个数 m=2,用 DM-EDF 方法和 GBRRS 方法来调度图 2 所示的规则集 R(R={R_1,R_2,R_3}).

DM-EDF 方法将 R 转化成τ={τ_1(3,40,42),τ_2(3,19,43),τ_3(4,39,43)}进行调度,τ中推理任务的执行顺序如图 7(a)所示.其中,τ_3 因为完成时刻(为 61)错过其绝对截止期(为 47)而被拒绝进入系统,即其相应的规则 R_3 在 DM-EDF 方法调度下被系统拒绝执行规则匹配操作.

GBRRS 方法则将 R 转化为推理任务图 G_R 如图 4 所示.根据第 2.1 节中基于图的端到端推理任务模型,可利用任务及其子任务的部分参数(表 2 中的非粗体数值)初始化它们的其余参数(表 2 中的粗体数值)并根据第 2.2 节中的推理任务调度算法对其进行调度,τ中推理任务的执行顺序如图 7(b)所示.



(a) DM-EDF



(b) GBRRS

Fig.7 DM-EDF and GBRRS scheduling examples

图 7 DM-EDF 和 GBRRS 方法的调度实例

从图 7(a)和图 7(b)可以反推出规则 R_1(正斜线),R_2(横线)和 R_3(反斜线)的执行顺序,由此可见,规则集 R 用 DM-EDF 方法不能被调度成功,而用 GBRRS 方法则能够被调度成功.具体来说,图 7 的调度结果显示,R_1 和 R_3 的完成时间在 DM-EDF 方法下比 GBRRS 方法下都提前了 17 个时间单元.这是因为 GBRRS 方法具有以下两个优点.

- 首先,GBRRS 方法的调度单元是推理子任务 $\tau_{i,j}$,能够更充分地利用处理器的空闲资源.具体来说,

  (1) GBRRS 方法允许已就绪的子任务 $\tau_{i,j}$ 在 τ_i 就绪前开始执行,更好地利用了处理器的空闲时间,减

少了τ_i的等待时间.如图 7(b)中,$\tau_{1,5}$和$\tau_{1,6}$可利用τ_1就绪前在 t=1 到 t=3 的时间段占用处理器运行.

(2) GBRRS 方法可利用推理任务图的特点,将具有相同直接前继的多个推理子任务并行地占用处理器,提前了τ_i的完成时间.如图 7(b)中,$\tau_{1,2}$和$\tau_{1,1}$可并行执行,$\tau_{3,6}$和$\tau_{3,5}$也可并行执行,则τ_1和τ_3的完成时间分别提前了 14 个和 3 个时间单元.

- 其次,GBRRS 方法可以及时地反映推理任务τ_i执行时间的动态变化,从而反映规则对处理器资源需求的动态变化,提高了规则调度成功率.比如,GBRRS 方法通过引入关联子任务集来减少关联子任务的重复执行,从而提前了推理任务的完成时间.如图 7(b)中,子任务$\tau_{1,5}$执行完成,互为关联的$\tau_{2,3}$也执行完成.因此,τ_2和τ_3分别减少了 11 个和 16 个时间单元的重复执行,其完成时间也被提前.

Table 2 Parameter initialization of tasks and sub-tasks

表 2 任务和子任务的参数初始化

名称	就绪时间	相对截止期	绝对截止期	紧迫程度	影响度	最坏执行时间
τ_1	**3**	42	**45**	–	–	**40**
τ_2	**3**	43	**46**	–	–	**19**
τ_3	**4**	43	**47**	–	–	**39**
$\tau_{1,1}$	0	–	**45**	**1/45**	**1**	2
$\tau_{1,2}$	0	–	**45**	**1/45**	**1**	4
$\tau_{1,3}$/ $\tau_{2,1}$	2	–	**min(45,46)=45**	–	**2**	2
$\tau_{1,4}$/ $\tau_{2,2}$	3	–	**min(45,46)=45**	–	**2**	2
$\tau_{1,5}$/ $\tau_{2,3}$	0	–	**min(45,46)=45**	**1/45**	**2**	4
$\tau_{1,6}$/ $\tau_{3,1}$	0	–	**min(45,47)=45**	**1/45**	**2**	4
$\tau_{1,7}$/ $\tau_{3,2}$	1	–	**min(45,47)=45**	–	**2**	4
$\tau_{1,8}$	**Inf**	–	**45**	–	**1**	2
$\tau_{1,9}$/ $\tau_{2,5}$	**Inf**	–	**min(45,46)=45**	–	**2**	3
$\tau_{1,10}$/ $\tau_{3,7}$	**Inf**	–	**min(45,47)=45**	–	**2**	4
$\tau_{1,11}$	**Inf**	–	**45**	–	**1**	4
$\tau_{1,12}$	**Inf**	–	**45**	–	**1**	2
$\tau_{1,13}$	**Inf**	–	**45**	–	**1**	3
$\tau_{2,4}$/ $\tau_{3,3}$	0	–	**min(46,47)=46**	**1/46**	**2**	2
$\tau_{2,6}$	**Inf**	–	**46**	–	**1**	4
$\tau_{2,7}$	**Inf**	–	**46**	–	**1**	2
$\tau_{3,4}$	0	–	**47**	**1/47**	**1**	4
$\tau_{3,5}$	0	–	**47**	**1/47**	**1**	2
$\tau_{3,6}$	4	–	**47**	–	**1**	3
$\tau_{3,8}$	**Inf**	–	**47**	–	**1**	4
$\tau_{3,9}$	**Inf**	–	**47**	–	**1**	2
$\tau_{3,10}$	**Inf**	–	**47**	–	**1**	4
$\tau_{3,11}$	**Inf**	–	**47**	–	**1**	2
$\tau_{3,12}$	**Inf**	–	**47**	–	**1**	4

3 模拟实验

本节将通过模拟实验来验证 GBRRS 方法的性能,并将其与 DM-EDF 方法进行对比.

3.1 模拟实验设置与性能指标

为了保障测试集的合理性(本文的任务是基于安全攸关系统中规则推理过程建模而产生的,而多核环境下基于图的端到端实时任务调度并没有被研究,因此,已有的调度研究中没有相关的Benchmark实例集可用),本文通过泊松分布模拟安全攸关系统中的原子事件实例到达情况,从而模拟出推理任务的生成,并通过设定规则事件匹配子图相关的参数来随机生成规则图,从而模拟安全攸关系统的规则.实验中涉及的参数见表 3.

Table 3 Parameters

表 3 参数

名称	描述
penum	原子节点的个数
indegreemax	事件节点入度的最大值
outdegreemax	事件节点出度的最大值
topdowndepthmax	规则子图高度的最大值
cminmax	节点上推理操作的执行时间范围
dminmax4rule	规则子图相对截止期的取值范围
totaload	规则集的总负载,即 $S_R=\sum(C_i/D_i)$
m	处理器核个数
aveload	规则集的平均负载,即 $U_R=S_R/m$

规则集的生成过程如下.

- 步骤 1:随机生成 *penum* 个原子节点,为每个节点随机选取出度(不超过 *outdegreemax*,规则子图中,每个节点的出度是该节点被用次数(即该节点上的事件被用于生成复合事件的次数)的最大值)、节点上推理操作的执行时间(范围为 *cminmax*)以及节点上原子事件实例的到达时间(满足泊松分布,其参数在 100~250 范围内),那么初始候选节点集为这些原子节点(候选节点是被用次数小于出度的事件节点).
- 步骤 2:随机选取当前规则子图的高度(不超过 *topdowndepthmax*),当前规则子图是当前即将生成的规则所对应规则子图.
- 步骤 3:随机选取当前事件节点的入度 *indegree*(事件节点的入度是复合生成该节点上事件的候选节点的个数,不超过 *indegreemax*),从当前的候选节点集中选取 *indegree* 个候选节点,通过随机的事件操作符复合生成当前节点上的事件,则当前事件节点生成完毕;更新当前规则子图的高度,事件节点的被用次数和当前的候选节点集.
- 步骤 4:重复步骤 3,直到当前规则子图的高度达到 *topdowndepth*−1(此时,目标节点已生成完毕)时,则生成该规则子图的动作节点.至此,该规则子图生成完毕,随机选取该规则子图的相对截止期(范围为 *dminmax*4*rule*),并更新规则集的总负载和平均负载.
- 步骤 5:重复步骤 2~步骤 4,直到规则集的负载达到设定值.

实验的性能指标是规则调度成功率:

$$SUR=N_{SR}/N_R \tag{6}$$

其中,N_{SR} 是在截止期前完成的规则个数,N_R 是被调度的规则总数.

3.2 实验结果分析

本节给出了规则调度成功率随规则集的平均负载 U_R 而变化的情况,如图 8(在保持 *m* 不变的前提下,通过 S_R 递增使 U_R 递增)和图 9(在保持 S_R 不变的前提下,通过 *m* 递增使 U_R 递减)所示.令 *penum*=1000,根据上一节所描述的方法来生成随机的规则集,并分别用 DM-EDF 方法和 GBRRS 方法对其进行调度.图 8 和图 9 中每个点的取值都是进行了 5 次实验取其平均值后的结果.

由图 8 可见,DM-EDF 方法和 GBRRS 方法下的规则调度成功率都随 U_R 的增加而逐渐减少.GBRRS 的规则调度成功率比 DM-EDF 的规则调度成功率平均高出 14.86%,这是因为 GBRRS 方法能够及时反映推理任务执行时间的动态变化,从而及时反映出规则对处理器资源需求的动态变化,减少了其响应时间,增加了规则调度成功率.当 $U_R\leqslant 100\%$时,两种方法的规则调度成功率都是 100%;但当 $U_R\geqslant 100\%$时,随 U_R 的增加,两者之间的差距先增大后减少,这是因为关联子任务集出现的概率随 U_R 的增加而增加,因此,GBRRS 方法的规则调度成功率下降较为平缓,两者差距逐渐增大;当 $U_R\geqslant 350\%$时,关联子任务集在 GBRRS 方法中所产生的优势由于处理器资源的紧缺而逐渐不再突出,因而两者差距又逐渐减少了.可以看出,在 U_R 较高,如 350%时,GBRRS 仍保持 80%以上的规则调度成功率.

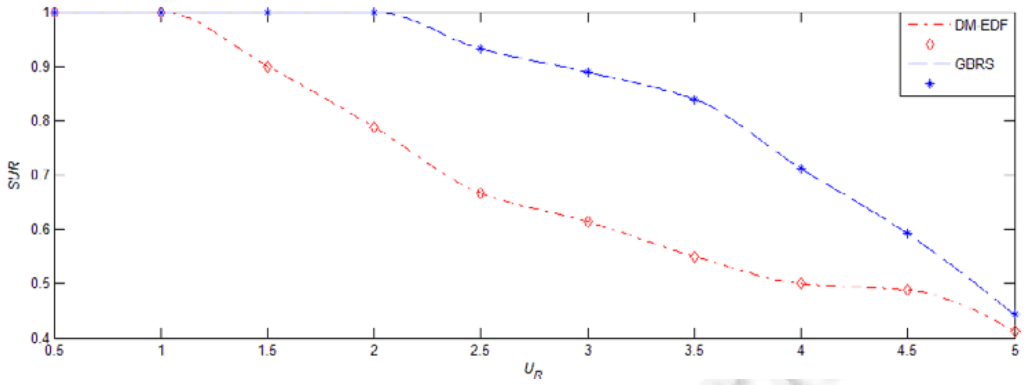


Fig.8 Average rule scheduling success ratio as a function of U_R when m=8

图 8 m=8 时随 U_R 变化的平均规则调度成功率

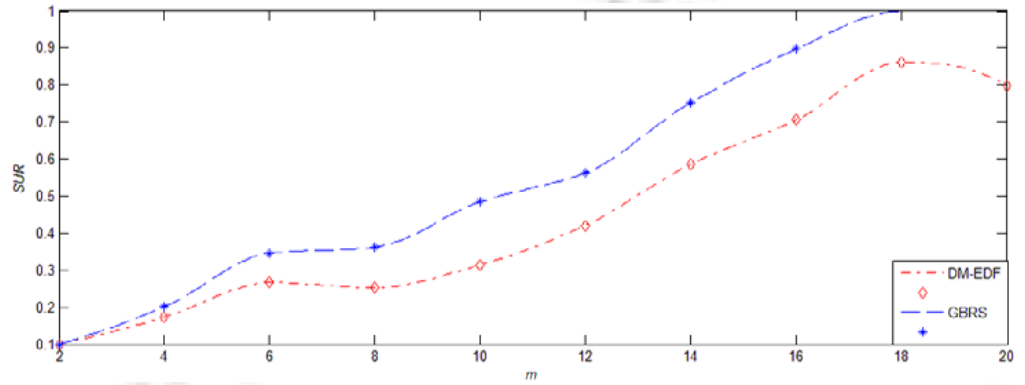


Fig.9 Average rule scheduling success ratio as a function of m when S_R=50

图 9 S_R=50 时随 m 变化的平均规则调度成功率

图 9 表明,DM-EDF 方法和 GBRRS 方法下的规则调度成功率都随 m 的增加而提高,也就是随 U_R 减少而提高.GBRRS 的规则调度成功率比 DM-EDF 的规则调度成功率平均高出 13.05%,这是因为 GBRRS 方法的调度单元是推理子任务 $\tau_{i,j}$,能够更充分地利用处理器的空闲资源.随着处理器核的个数增加,两者的差距逐渐增大.这是因为随着处理器核的个数增加,GBRRS 方法中推理任务并行占用处理器的几率增大,更容易被调度成功,因此,两者的差距逐渐增大.当 m≥18 时,GBRRS 可以成功调度规则集里的所有规则,成功率达到 100%.

综上,无论规则集的平均负载如何变化,GBRRS 方法都比 DM-EDF 方法在规则调度成功率上平均高出 13%~ 15%;而且 GBRRS 方法在规则集的平均负载较高(如 350%)时,仍保持着 80%以上的规则调度成功率.

4 结束语

为了解决多核环境下的规则调度问题,本文提出了一种基于图的规则调度方法(GBRRS).

- 首先,GBRRS 方法根据基于事件图的推理过程,用图映射方法对规则调度进行了建模,提出了基于图的端到端推理任务模型.
- 然后给出了推理任务的调度算法,通过调度这些推理任务来合理安排规则匹配与执行的顺序,从而保证了规则调度的安全性.
- 最后进行了模拟实验.实验结果表明,GBRRS 方法在规则调度成功率上优于 DM-EDF 方法,且在规则集的平均负载较高(如 350%)时,其规则调度成功率仍保持在 80%以上.

在下一步工作中,我们将考虑解决以下两个问题.

- 第一,在规则重要性不同的情况下,进行规则调度时,还需要考虑规则在不同重要性之间的切换以及在高重要性模式下截止期的优先满足.
- 第二,在异构的多处理器环境下为规则分配处理器核,还需要考虑不同处理器核上处理速率的差异性.

References:

[1] John C. Safety critical system: Challenges and directions. In: Proc. of the 24th Int'l Conf. on Software Engineering. New York: IEEE, 2002. 547−550. [doi: 10.1109/ICSE.2002.1007998]

[2] Dean T, Boddy M. An analysis of time-dependent planning. In: Proc. of the 7th National Conf. on Artificial Intelligence. New York: IEEE, 1988. 49−54.

[3] Garvey A, Lesser V. Design-to-time real-time scheduling. IEEE Trans. on Systems Man & Cybernetics, 1996,23(6):1491−1502. [doi: 10.1109/21.257749]

[4] Lesser V, Pavlin J, Durfee E. Approximate processing in real-time problem solving. AI Magazine, 1988,9(1):49−61.

[5] Mouaddib A, Charpillet F, Haton J. GREAT: A model of progressive reasoning for real-time systems. In: Proc. of the 6th Int'l Conf. on Tools with Artificial Intelligence. New York: IEEE, 1994. 521−527. [doi: 10.1109/TAI.1994.346447]

[6] Kang J, Cheng A. Shortening matching time in OPS5 production systems. IEEE Trans. on Software Engineering, 2004,30(7): 448−457. [doi: 10.1109/TSE.2004.32]

[7] Cheng A, Fujii S. Self-stabilizing real-time OPS5 production systems. IEEE Trans. on Knowledge and Data Engineering, 2004, 16(12):1543−1554. [doi: 10.1109/TKDE.2004.95]

[8] Li X, Qiao Y, Wang HA. A flexible event-condition-action rule processing mechanism based on a dynamically reconfigurable structure. In: Proc. of the 11th Int'l Conf. on Enterprise Information Systems. New York: IEEE, 2009. 328−329. [doi: 10.5220/0001987402910294]

[9] Li X, Qiao Y, Li X, Wang HA. Real-time ECA rule reasoning in active database. Journal of Computer Research and Development, 2010,47(1):250−258 (in Chinese with English abstract).

[10] Li N, Qiang G. Deadline-aware event scheduling for complex event processing systems. In: Proc. of the 14th Int'l Conf. on Intelligent Data Engineering and Automated Learning. New York: IEEE, 2013. 101−109. [doi: 10.1007/978-3-642-41278-3_13]

[11] Li N. Real-time sensor data stream processing with event-graph [Ph.D. Thesis]. Beijing: Institute of Automation, Chinese Academy of Sciences, 2014 (in Chinese with English abstract).

[12] Qiao Y, Zhong K, Wang HA, Li X. Developing event-condition-action rules in real-time active database. In: Proc. of the 2007 ACM Symp. on Applied Computing. New York: ACM Press, 2007. 511−516. [doi: 10.1145/1244002.1244120]

[13] Qiao Y, Li X, Wang HA, Zhong K. Real-time reasoning based on event-condition-action rules. In: Meersman R, Tari Z, Herrero P, eds. Proc. of the Move to Meaningful Internet Systems. Berlin: Springer-Verlag, 2008. 1−2. [doi: 10.1007/978-3-540-88875-8_1]

[14] Qiao Y, Han QN, Wang HA, Li X, Zhong K, Zhang KM, Liu J, Guo AX. RTRS: A novel real-time reasoning system based on active rules. ACM Applied Computing Review, 2013,13(2):66−76. [doi: 10.1145/2505420.2505426]

[15] Paschke A, Kozlenkov A. Rule-based event processing and reaction rules. In: Governatori G, Hall J, Paschke A, eds. Proc. of the Rule Interchange and Applications. Berlin: Springer-Verlag, 2009. 53−66. [doi: 10.1007/978-3-642-04985-9_8]

[16] Wang D, Rundensteiner E, Ellison RT, Wang H. Active complex event processing infrastructure: Monitoring and reacting to event streams. In: Proc. of the 29th Int'l Conf. on Data Engineering Workshops. New York: IEEE, 2011. 249−254. [doi: 10.1109/ICDEW.2011.5767635]

[17] Mei Y, Madden S. ZStream: A cost-based query processor for adaptively detecting composite events. In: Proc. of the SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2009. 193−206. [doi: 10.1145/1559845.1559867]

[18] Wu E, Diao Y, Rizvi S. High-performance complex event processing over stream. In: Proc. of the SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2006. 407−418. [doi: 10.1145/1142473.1142520]

[19] Xiao F, Aritsugi M, Wang Q, Zhang R. Efficient processing of multiple nested event pattern queries over multi-dimensional event streams based on a triaxial hierarchical model. Artificial Intelligence in Medicine, 2016,72(1):56−71. [doi: 10.1016/j.artmed.2016.08.002]

[20] Li X, Fan YS, Wang HA, Qiao Y. Estimation on worst-case execution time of real-time complex event processing. Journal of Computer Research and Development, 2012,49(10):2054−2065 (in Chinese with English abstract).

[21] Manabe Y, Aoyagi S. A feasibility decision algorithm for rate monotonic and deadline monotonic scheduling. Real-Time Systems, 1998,14(2):171−181. [doi: 10.1023/A:1007964900035]

[22] Li J, Luo Z, Ferry D, Agrawal K, Gill C, Lu CY. Global EDF scheduling for parallel real-time tasks. Real-Time Systems, 2015, 51(4): 395−439. [doi: 10.1007/s11241-014-9213-9]

附中文参考文献:

[9] 李想,乔颖,李欣,王宏安.主动数据库中的实时 ECA 实时规则推理算法.计算机研究与发展,2010,47(1):250−258.

[11] 李娜.基于事件图的实时感知数据流处理[博士学位论文].北京:中国科学院自动化研究所,2014.

[20] 李想,范玉顺,王宏安,乔颖.实时复杂事件处理的最坏响应时间估算.计算机研究与发展,2012,49(10):2054−2065.

王娟娟(1989－),女,福建莆田人,博士,主要研究领域为实时智能,实时调度.

熊金泉(1963－),男,教授,CCF 专业会员,主要研究领域为计算机图形图像处理,计算机辅助设计与可视化.

乔颖(1973－),女,博士,研究员,主要研究领域为实时智能,实时调度.

王宏安(1963－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为实时智能,人机交互.