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

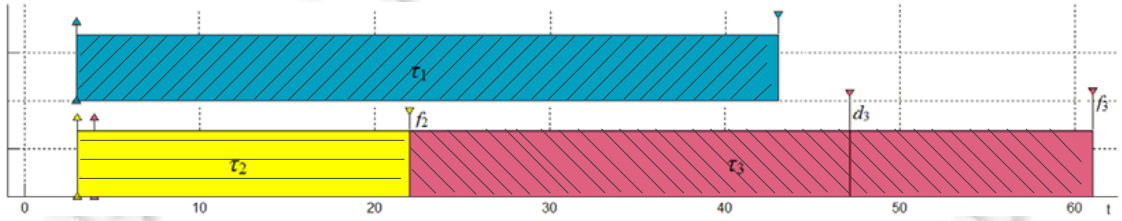
任务实例个数是 n(实时规则可被触发多次而产生多个规则推理任务的实例),基于图的端到端推理任务模型下,推理子任务实例个数总和为 w,那么 DM-EDF 方法时序约束控制的复杂度是 $O(1)$(DM-EDF 方法是用 DM 方法把规则上的推理操作映射成推理任务后用全局 EDF 算法[22]对其进行调度),GBRRS 方法虽然引入图模型,其时序约束控制的复杂度仍是 $O(1)$;DM-EDF 方法优先级排序的复杂度是 $O(n\times\log_2 n)$,GBRRS 方法优先级排序的复杂度是 $O(w\times\log_2 w)$;DM-EDF 方法准入控制的复杂度是 $O(m\times n^2)$,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

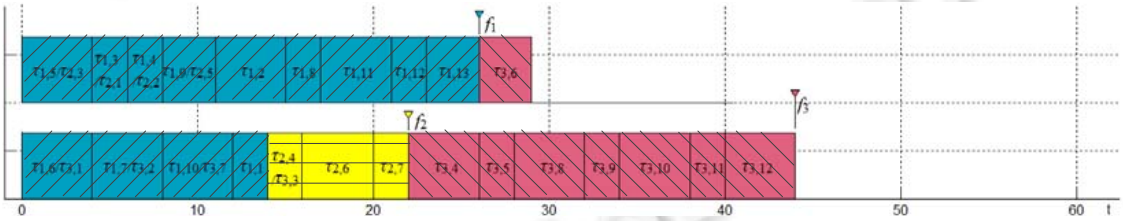
设处理器核的个数 m=2,用 DM-EDF 方法和 GBRRS 方法来调度图 2 所示的规则集 R(R={R_1,R_2,R_3}).

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.

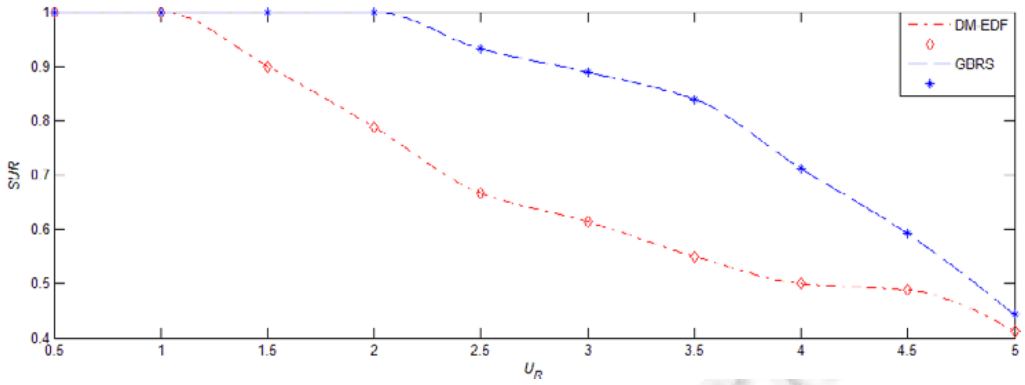


Fig.8 Average rule scheduling success ratio as a function of U_R when m=8

图 8 m=8 时随 U_R 变化的平均规则调度成功率

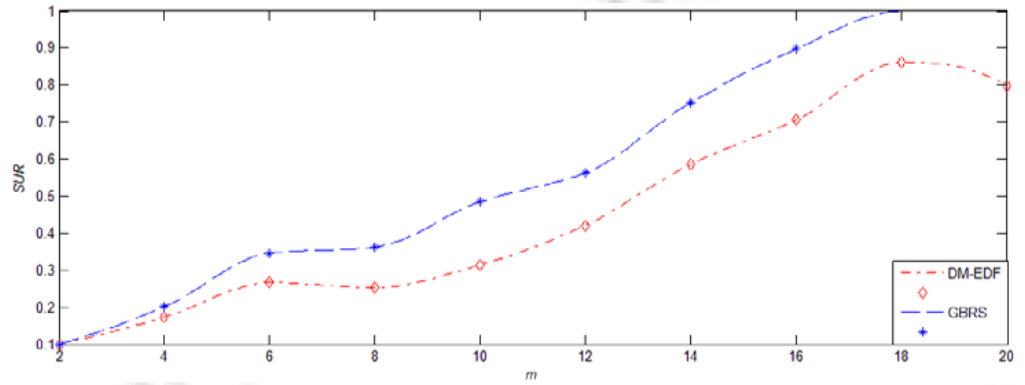


Fig.9 Average rule scheduling success ratio as a function of m when S_R=50

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.