RFID 复杂事件检测方法的研究和改进

刘海龙[1],李战怀[1],陈 群[1],娄 颖[1,2]
LIU Hai-long[1],LI Zhan-huai[1],CHEN Qun[1],LOU Ying[1,2]

1.西北工业大学 计算机学院,西安 710072
2.河南科技大学 电子信息学院,河南 洛阳 471003
1.School of Computer Science,Northwestern Polytechnical University,Xi'an 710072,China
2.Electronic and Information Engineering College,Henan University of Science and Technology,Luoyang,Henan 471003,China
E-mail:liuhailong@nwpu.edu.cn

LIU Hai-long,LI Zhan-huai,CHEN Qun,et al.Study and improvement on RFID complex events detecting methods.Computer Engineering and Applications,2008,44(11):5-8.

Abstract: Complex Event Processing(CEP) technology is a new technology which is applied in processing large volume primitive events and picking out valuable semantic events.In comparison with traditional events,RFID events have the following characteristics:large volume,temporal & spatial,inaccurate etc.It is necessary to do some researches on these characteristics and build an extensible CEP framework for RFID events.In this paper,the current researches on describing models,processing framework,parameter context,detecting method of RFID CEP are summarized.This paper picks out three problems of SASE and brings out some enhancing methods.These methods are proved efficient by some experiments.
Key words: RFID;complex events;detection;enhancement

摘 要: 复杂事件处理(CEP)是一个新兴的技术领域,用于处理大量的简单事件,并从其中整理出有价值的事件。RFID 事件和传统事件相比具有海量、时间性和空间性、数据不准确等特征。针对 RFID 事件的相关特征进行研究,建立一个可扩展的基于规则的 RFID 复杂事件处理系统是非常必要的。从复杂事件的描述模型、处理框架、上下文语义、检测方法等方面总结了国内外 RFID 复杂事件处理的研究现状,分析了目前的研究存在的不足,并且针对 SASE 的复杂事件算法存在的三点不足进行了改进。实验证明相关的改进对简单数据集是有效的。
关键词: RFID;复杂事件;检测;改进
文章编号: 1002-8331(2008)11-0005-04 **文献标识码:** A **中图分类号:** TP315

RFID(Radio Frequency Identification)系统的原子事件所包含的语义信息非常有限。通过原子事件仅仅可以了解一些简单信息。而实际的应用系统更多关注货物的流通规律、业务逻辑等复杂信息。要获取这些信息,需要通过一定的规则组合多个原子事件构成一个复杂事件。RFID 应用系统通常将业务逻辑转化成复杂事件,通过检测复杂事件来检测相关的业务逻辑。复杂事件处理(CEP)是一个新兴技术领域,用于处理大量的简单事件,并从其中整理出有价值的事件。Gartner 公司曾预言,CEP 将在 5~10 年的时间内成为一种通用计算模式[4]。本文从复杂事件的描述模型、处理框架、上下文语义、检测方法等方面总结了国内外 RFID 复杂事件处理的研究现状,分析了目前研究存在的不足。本文针对 SASE[9,10]复杂事件检测算法的三点不足进行了改进,并且基于简单数据集对改进的效果进行了验证。

1 相关定义

定义 1 原子事件:读写器和标签之间的一次数据交互被定义成一个原子事件。原子事件用下面的三元组来表示:<RID,OID,Timestamp>[3]。其中 RID 为阅读器标识,OID 为电子标签标识,Timestamp 为事件发生的时间戳。

定义 2 复杂事件:是由原子事件或者复杂事件按照一定的运算规则形成的事件。复杂事件用下面的二元组来表示:<Element、Rule>。其中 Element 是复杂事件的组成元素,Element={原子事件,复杂事件}。Rule 为运算规则。

2 RFID 事件描述模型

Snoop[8]是基于中基于主动数据库提出的一套复杂事件管理语言。原子事件或者复杂事件可以通过运算转换成新的复杂事件。通过 Snoop 的相关操作符可以将原子事件组合成复杂事

基金项目: 国家自然科学基金(the National Natural Science Foundation of China under Grant No.60720106001);西北工业大学青年教师创新基金。
作者简介: 刘海龙(1980-),男,博士研究生,讲师,主要研究领域为数据管理技术、网络软件技术;李战怀(1961-),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库理论与技术;陈群(1976-),男,博士,教授,主要研究领域 RFID 数据管理、XML;娄颖(1980-),男,博士研究生,主要研究领域为数据管理技术。
收稿日期: 2007-11-21 **修回日期:** 2008-01-15

件，也可以将复杂事件组合成复杂事件。例如：$X=(E_1\Delta E_2;E_3;E_2\Delta E_4)$，是由<复杂事件；原子事件；复杂事件>组成的复杂事件。文献[3]在 snoop 的基础上中提出了针对 RFID 原子事件和复杂事件的形式化描述方法，并且继承了 snoop 的采用事件树来描述复杂事件的方法。

和 snoop 相比，文献[3]中的运算符针对 RFID 事件的特点追加了滑动窗口相关的操作。例如：$E=WITHIN(E_1\wedge E_2,10\ s)$。只有当 E_1 和 E_2 的实例中满足如下条件：$interval\ (e_1,e_2)\leqslant 10$ s 才会生成事件 E。文献[3]中构成复杂事件的对象可以是原子事件也可以是复杂事件。例如：$E=WITHIN(E_1\wedge E_2,10\ s)$是由复杂事件追加了滑动窗口限制的条件下生成的一个复杂事件。

SASE 作为一种支持复杂事件处理查询的表达语言，是目前 RFID 复杂事件处理中较为先进的方法。SASE 具有的特点：支持自由的非（¬）操作（Negative Operation）、支持参数化谓词、支持滑动窗口限制。

3 RFID 复杂事件检测

3.1 复杂事件分析框架

文献[5]提出了一个包含 7 个模型的通用复杂事件分析框架。这些模型从不同方面刻画了基于事件观察与通知的分布式应用中的主要活动和相互关联，它们之间的关系如图 1 所示。文献[13]中总结了这 7 个模型之间的关系，给出了框架的图形化描述（如图 1），并且指出可将事件观察或复杂事件检测抽象地看作是对系统接收的事件历史进行的模式匹配过程。

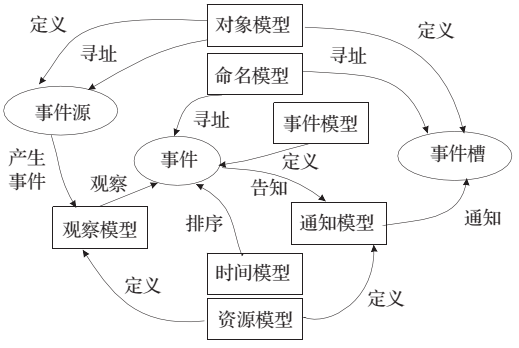

图 1 复杂事件的通用分析框架

3.2 复杂事件的检测上下文语义

文献[8]中提出了 Recent、Chronicle、Continuous 和 Cumulative 四种检测复杂事件的上下文语义。其中只有 Chronicle 适用于 RFID 复杂事件检测。因为 RFID 应用系统中多个阅读器同时获取数据，并且这些数据会同时被处理，所以复杂事件通常会互相交叠。这种情况下使用 Recent、Continuous、Cumulative 检测语义会导致检测错误。

在 Chronicle 语义中，选取构成复杂事件的最旧的初始事件作为该复杂事件的初始事件；选取构成复杂事件的最旧的终止事件作为该复杂事件的终止事件。例如：使用该语义从序列 $\{e_1^1,e_1^2,e_2^1,e_3^1,e_2^2,e_4^1,e_3^2,e_4^2\}$ 中检测复杂事件 $X=(E_1\Delta E_2;E_3;E_2\Delta E_4)$ 的结果为：$e_1^1e_2^1e_3^1e_2^1e_4^1$。

3.3 RFID 事件的检测模式

传统的基于图形的事件处理系统检测复杂事件时一般采用自底向上的方式：原子事件的发生被注入叶节点，随之上溯导致父复杂事件的发生。但是这种方式对 RFID 事件并不适用。因为许多 RFID 事件是非自发的事件，它们并不能检测到自己的发生，除非被触发（例如：复杂事件¬ E）。文献[3]中针对上述问题基于事件树提出了三种通用的 RFID 事件检测模式：Push（↑），Pull（↓），Mixed（↕），分别用来检测自发事件，非自发事件和有时间限制的自发事件。

3.4 复杂事件的检测方法及中间件

复杂事件的检测在主动数据库中已经做了大量的研究，常用的方法有如下几种：即基于 Petri 网[14,15]、基于树[3,8]、基于图[16]和基于自动机[10,17]的检测方法。文献[13]分别从复杂事件模型、时间模型、检测模型三个方面评价了上述三种方法的优缺点。

最近出现一些事件处理器，如 CompAS[16]提出一个全息方法来过滤原始事件和识别复杂事件，它专注于处理传统事件，并不能针对 RFID 事件采用一些优化方法。HiFi[18]则沿着树结构的网络在不同时态和地理粒度层级上聚合事件。Siemens RFID Software[10,19]使用基于事件的框架将原子事件转化成有意义的复杂事件。它的事件处理是基于一个图的运算模型。HiFi 系统和 Siemens RFID Software 的表达能力均比 SASE[9]逊色，并且在处理大量的数据时并不能提供很好的优化技术。另外 HiFi[18]则对完全事件的支持有限。但是 SASE 也存在两个缺陷：（1）只能将原子事件转化成复杂事件，不能把复杂事件转化成复杂事件；（2）相关的研究是基于原子事件是有序的假设。

目前关于复杂事件的研究还存在一些问题，如文献[8]所做关于要求非事件必须拥有一个初始事件和一个终止事件的假设以及文献[3]必须要借助伪事件（Pseudo Event）才能检测非自发事件，都存在局限性。文献[5]中也没有给出非事件的主动规则的实现方法。

3.5 SASE 的复杂事件的检测方法存在的问题

基于 SASE 的事件处理系统采用有限自动机（NFA）和堆栈来检测复杂事件。其中针对两个显著的问题点——大的中间结果和大的滑动窗口采用了一些优化策略：Pushing Predicates Down（PPD）和 Pushing Window Down（PWD）。优化后的处理方法与 TelegraphCQ[22]相比在吞吐量和内存需求上有了较大的改善。但是上述优化策略还存在如下不足：

（1）PPD 只能应用于参数域有限制的事件序列，而通常处理的 RFID 事件数量比较大，这种情况下通过为不同的变量建立不同的堆栈来处理参数限制的方法就不再适用。

（2）RFID 应用场景中通常某一过程产生的事件比较多，而其后续过程产生的事件比较少。此时使用 SASE 检测复杂事件，会进行多次无效的检测。例如：检测图 2 中的复杂事件 SEQ（factory，truck，warehouse），每检测一个 truck 类事件都需要检测 factory 类事件中是否有相关的事件，而实际上 67%的检测是不必要的。

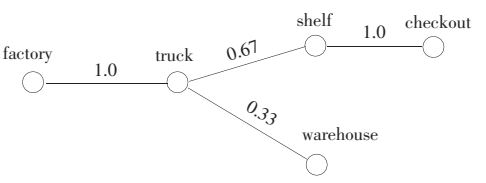

图 2 一个典型的商品供应链

（3）SASE 在检测包含非事件的复杂事件时（例如：SEQ（A，¬ B，C）），将检测的非事件（b_i）保存在堆栈中，当检测到其后续事件时（c_i）判断其相关的事件是否在非事件的堆栈中出现，如

果出现则不构成有效序列,否则构成有效序列。当非事件比较多的情况下,这种检测方法需要占用大量的内存。

(4)PWD 中的技术主要是尽可能早的使用滑动窗口的限制以减少产生序列的数量。其中没有提供一种有效并且自动的方法来摈弃已经在滑动窗口之外事件。

3.6　基于 SASE 的复杂事件的检测方法优化

PPD 中使用堆栈来处理大量的数据时存在较大的局限性,而使用 Hash 表则可以提高存储和查询的效率。为此,针对 SASE 子事件分布不均匀和含有非事件的复杂事件检测方法的改善都是基于有限自动机(NFA)和 Hash 表来实现的。使用 B+树(如图 3)可以灵活的摒除滑动窗口之外的数据,从而减少中间数据的数量。

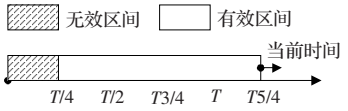

图 3　B+树可以灵活定义无效区间长度

文中将重点对 3.5 节中的前三个问题点进行讨论,基于 B+树的改善策略将在后续的文章中讨论。

下面的算法中要构成复杂事件 SEQ(A,…,F)必须要满足如下条件:(1)a_1.OID=…=f_1.OID;(2)a_1.Timestamp<…<f_1.Timestamp;(3)f_1.Timestamp−a_1.Timestamp≤滑动窗口大小。

3.6.1　子事件分布不均匀时复杂事件检测策略

当子事件分布不均匀时,可以根据业务特征指定哪一类事件需要暂缓处理。在扫描到这些特殊事件时,先直接存放到相关的 Hash 表中。在处理其后继事件时,再检测该类事件及其前驱事件 Hash 中是否有满足条件的事件。这样可以避免大量的无效检测。具体算法如下:

(1)检测复杂事件

```
读取配置文件,生成相关 NFA;
While(! end())
{
  从输入事件流中取得一个事件 e;
  If(NFA(e)! =0)continue;//该事件为无关事件
  if(e 是特殊处理//e 是起始事件)保存在 Hash 表中;
  else{
       if(e 的前驱事件 Hash 表中包含相关事件){
         if(e 是终结事件)输出复杂事件;
         else  保存在 Hash 表中;
       }
    }
}//end  while
```

(2)输出复杂事件

```
pCurrentEvent=终结事件指针;
While(pCurrentEvent! =NULL)
{
  Push_stack(pCurrentEvent);//压入输出结果堆栈
  if(pCurrentEvent 前驱事件是特殊处理事件)
  {
    If(pCurrentEvent 前驱事件的前驱 Hash 表中包含相关事件)
    {
      Push_stack(pCurrentEvent 前驱事件指针);
      pCurrentEvent=pCurrentEvent 前驱事件指针;
    }
    else
     return;
    }
      pCurrentEvent=pCurrentEvent 前驱事件指针;
  }//end  while
While(输出结果堆栈非空)
{
  Pop_Stack()
  输出复杂事件;
}
```

3.6.2　含有非事件的复杂事件检测策略

当检测到非事件时,可以直接检测其前驱事件的 Hash 表中是否含有相关的事件。如果含有,则将前驱事件 Hash 表中相关的事件删除。这样不需要保存非事件相关的 Hash 表,同其前驱事件 Hash 表中和非事件相关的事件也不需要保存,在非事件较多时可以节省大量的内存。具体算法如下:

```
读取配置文件,生成相关 NFA;
While(! end())
{
  从输入事件流中取得一个事件 e;
If(NFA(e)! =0)continue;//该事件为无关事件
if(e 是非事件)
{
  将 e 所有前驱事件类中的关联事件从 Hash 表中删除;
  Continue;
}
else
{
  if(e 为起始事件//e 前驱事件 Hash 表中包含相关事件)
  {
    if(e 终结事件)
      输出复杂事件;
    else
      保存在 Hash 表中;
    }
  }
}//end  while
```

3.7　实验结果及分析

和传统事件流处理[20,21]一样,实验中将使用两个标准来衡量算法的性能:吞吐量和内存需求。实验基于  来产生相关的数据,具体数据生成规则如表 1、表 2。

表 1　数据生成规则、相关参数及测试环境 1

A 类事件个数	n_A=1 000 000
事件概率分布	$p_{A\to B}$=0.9,$p_{B\to D}$=[0.1,0.6],$p_{C\to D}$=0.1
事件时间分布/s	t_A=[0,3 600],$t_{A\to B}$=$t_{A\to C}$=$t_{B\to D}$=$t_{C\to D}$[0,300]
滑动窗口/s	100~300
Hash 表大小	1 000
测试环境	Pentium®4 CPU 2.8 GHz 2.79 GHz,0.99 G,DDR

通过图 4 的实验结果可以看出,对于复杂事件 SEQ(A,B,D)的检测,在 $p_{B\to D}$ 不超过 0.5 时,改进后算法的吞吐量明显要优于改进前。随着 $p_{B\to D}$ 的增加,改进前后的算法吞吐量的差异会随之降低。因为随着 $p_{B\to D}$ 增加,对产生复杂事件来说无效的检测会随之减少,此时改进前后算法的差异也会随之减少。另

外随着滑动窗口的增加，有效的复杂事件的数目会随之增加，此时进前和改进后的算法的吞吐量都会随之降低(如图5)。通过图6可以看出，在检测SEQ(A,¬B,D)时，因为B类事件相关的A类事件被从Hash表中删除，改进后的算法占用的内存明显要少于改进前的算法。随着$p_{A\to B}$增加，非事件B的数量随之增加，A类事件的Hash表中的被删除的数据也会随之增加，占用的内存会随之减少，这种优势会更加明显。图7是$p_{A\to B}$=0.5是改善前后的内存使用状况对比，通过该图可以看出改进后占用的内存明显要少于改进前。

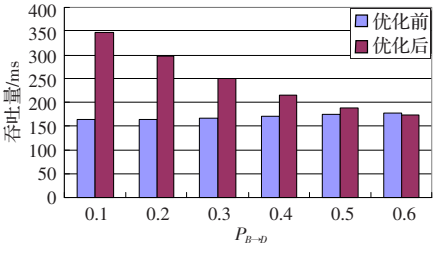

图4 SEQ(A,B,D)(3.6.1 节改进效果)

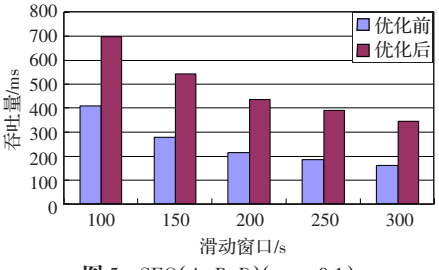

图5 SEQ(A,B,D)($p_{B\to D}$=0.1)

表2 数据生成规则、相关参数及测试环境2

A类事件个数	n_A=100 000
事件概率分布	$p_{A\to B}$=[0.5,0.9],$p_{A\to C}$=0.1,$p_{B\to D}$=$p_{C\to D}$=0.5
事件时间分布	t_A=[0,3 600],$t_{A\to B}$=$t_{A\to C}$=$t_{B\to D}$=$t_{C\to D}$[0,300]
滑动窗口/s	600
Hash表大小/s	1 000
测试环境	Pentium ® 4 CPU 2.8 GHz 2.79 GHz,0.99 G,DDR

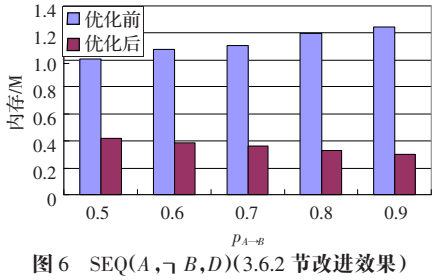

图6 SEQ(A,¬B,D)(3.6.2 节改进效果)

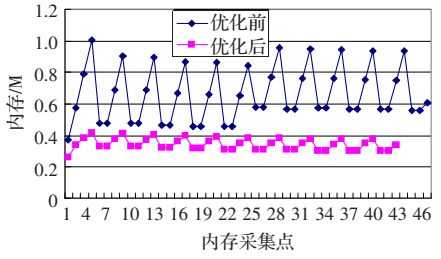

图7 SEQ(A,¬B,D)($p_{A\to B}$=0.5 时内存使用状况)

4 结束语

本文从复杂事件的描述模型、处理框架、上下文语义、检测方法等方面总结了国内外RFID复杂事件处理的研究现状，重点分析了SASE的复杂事件算法存在的不足，针对其中的部分不足进行了改善。实验证明相关的改进是有成效的。但是相关的实验是基于简单的数据集，针对复杂数据集(例如：(A,¬B,¬C,¬D,E))的改善效果还有待进一步的研究和验证。

参考文献：

[1] Stockman H.Communication by means of reflected power[C]//Proceedings of the IRE,1948:1196-1204.

[2] Landt J.Shrouds of time:the history of RFID[C]//the Association of Automatic Identification and Data Capture Technologies (AIM),2001.

[3] Wang F,Liu S,Liu P,et al.Bridging physical and virtual worlds:complex event processing for RFID data streams[C]//LNCS 3896:the 10th International Conference on EDBT'2006,2006:588-607.

[4] Palmer M.Seven principles of effective RFID data management[EB/OL].(2004).http://www.progress.com/realtime/docs/articles/7principles rfid mgmnt.pdf.

[5] Motakis I,Zaniolo C.Formal semantics for composite temporal events in active database rules[J].Journal of Systems Integration,1997,7(3/4):291-325.

[6] Gatziu S,Dirtrich K R.Detecting composite events in active databases using PetriNets[C]//Proc of the 4th International Workshop on Research Issues in Data Engineering:Active Database Systems,1994.

[7] 李战怀,聂艳明,陈群,等.RFID数据管理的研究进展[J].中国计算机学会通讯,2007(8).

[8] Chakravarthy S,Krishnaprasad V,Kim S.Composite events for active databases:semantic,contexts and detection[C]//VLDB'94,1994:606-617.

[9] Wu E,Diao Y,Rizvi S.High-Performance complex event processing over streams[C]//ACM SIGMOD,2006:407-418.

[10] Gyllstrom D,Wu E,Chae H J,et al.SASE:complex event processing over streams[C]//CIDR 2007,Asilomar,California,USA,2007:108-119.

[11] Moon M,Kim Y,Yeom K.Contextual events framework in RFID system[C]//Proceedings of the 3th International Conference on ITNG'06,2006:586-587.

[12] Rosenblum D S,Wolf A L.A framework for Internet-Scale event observation and notification[C]//Proceedings of the 6th European Software Engineering Conference/ACM SIGSOFT 5th Symposium on the Foundations of Software Engineering,1997.

[13] 张菊芳,魏峻.复合事件检测技术的综述与评价[J].计算机应用研究,2005,22(10):1-4.

[14] Gatsiu S,Dittrich K R.Events in an active objectoriented database system[C]//International Conference on Rules in Database Systems,1993:23-39.

[15] 左万利.复合时序事件及其基于Petri 网的检测[J].系统工程学报,2003,18(3):262-267.

[16] Hinze A.Efficient filtering of composite events[C]//Proceedings of the British National Database Conference,2003:207-225.

[17] Gehani N H,Jagadish H V,Shemueli O.Composite event specification in active databases:model and implementation[C]//VLDB,1992:327-338.

[18] Franklin M J,Jeffery S,Krishnamurthy S,et al.Design considerations for high fan-in systems:the HiFi approach[C]//CIDR,2005.

(下转25页)