一种前置非操作的复杂事件处理方法

杨　晟,杨　颖

(广西大学 计算机与电子信息学院,广西 南宁 530004)

摘　要:针对许多基于无线射频识别的商业应用对复杂事件处理具有严格的实时性要求,而目前存在的复杂事件处理技术仍然存在处理效率不高、实时性差等问题,提出了一种前置非操作的复杂事件处理方法。该方法在构建非确定性有限自动机时加入消极事件对应的状态,用于把非操作前置到序列构建和扫描中,通过前置非操作来动态地去除序列扫描过程中不满足查询要求的原子事件,使得在平滑窗口内的事件数量大大减少,以此减少序列构建过程中的中间结果规模;并减少序列构建时回溯搜索的代价,以提高复杂事件处理的吞吐量。仿真实验结果表明,使用前置非操作方法后,复杂事件处理的吞吐量最高可提升约 20.2 倍,而中间结果规模在最好的情况下减少为原来的 1/20。因此使用前置非操作方法可以有效地提高复杂事件处理的性能。

关键词:无线射频设别;复杂事件处理;前置;非操作

中图分类号:TP391.7　　**文献标识码**:A　　**文章编号**:1673-629X(2015)11-0156-07

doi:10.3969/j.issn.1673-629X.2015.11.031

A Method of Complex Event Processing of Pushing Negation into Sequence Scan and Construction

YANG Sheng,YANG Ying

(School of Computer,Electronics and Information,Guangxi University,Nanning 530004,China)

Abstract:A new CEP (Complex Event Processing) method that pushes negation operation into SSC (Sequence Scan and Construction) to address changes that a large number of commercial applications based on RFID (Radio Frequency IDentification) have time-critical requirements in the processing CEP while the existing CEP technology still has the disadvantages of inefficiency and poor real-time capacity. The method adds negative status in the NFA (Non-deterministic Finite Automate) so as to push negation into SSC,which can remove the inappropriate events dynamically in process of sequence scan. Consequently,it reduces the intermediate result sizes. Besides,it cuts down the cost of traversing back along AIS in the process of Sequence Construction (SC) to improve the throughput. The results of simulation experiments show that at maximum case,the proposal shows 20.2 times higher throughout than the traditional method SASE, and the intermediate result sizes reduces to 1/20. So it's concluded that the method that pushes negation into SSC can significantly improve the performance in CEP.

Key words:RFID;CEP;pushing;negation

0　引　言

复杂事件处理(Complex Event Processing,CEP)是对由无线射频识别(Radio Frequency IDentification,RFID)技术产生的原始数据进行检测和转换,以便进行更高级应用的过程。

CEP 目前已成为许多企业信息系统中至关重要的部分。它被广泛应用在金融服务、医疗管理、物流配送、供应链管理[1-2]以及航空航天管理中。随着 RFID 在各商业领域的广泛应用[3],RFID 传感器产生的数据正在急剧增长,因而实时高效的复杂事件处理技术越来越成为众人关注的热点。

美国加州大学伯克利分校开发的 SASE[4]系统是一种采用非确定有限自动机(Non-deterministic Finite Automate,NFA)变体模型来进行复杂事件处理的典型系统。SASE 通过在 NFA 中引入活动实例栈(Active Instance Stack,AIS),以及把选择操作和窗口约束前置到序列扫描和构建 SSC(Sequence Scan and Construction)中,大大提高了 CEP 的性能。在 CEP 过程中,把

收稿日期:2015-03-07　　**修回日期**:2015-06-11　　**网络出版时间**:2015-11-04

基金项目:广西自然科学基金项目(2013GXNSFAA019344)

作者简介:杨　晟(1988-),男,硕士,研究方向为计算机网络与并行分布式计算;杨　颖,博士,教授,研究方向为数据库技术、数据挖掘技术。

网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20151104.0953.086.html

不允许出现在查询结果中的事件称为消极(Negative)事件,而对 Negative 事件的操作称为非(Negation)操作。但是 SASE 并没有介绍在出现 Negative 事件的情况下,如何实现将 Negation 操作前置 SSC 中加快事件处理。最近,关于 CEP 的研究不断被提出。陈琳等提出了一种多维度的 RFID 复杂事件处理方法[5],使用 R-tree 处理多维属性。陈远等提出了一种在 NFA 中引入可靠性约束检测的 CEP 方法[6],实现直接在不可靠的 RFID 数据流上进行实时清洗和复杂事件处理。文献[7-8]提出了基于分布式的 CEP 算法来提高查询处理的速度。Liu 等设计了基于树的图来构建 CEP 事件检测引擎[9]。Bok 等提出了一种使用最小约束条件的 CEP 方法[10],使用基于网格的查询索引和位图结构来检测复杂事件是否满足最小约束条件。Naotaka 等提出了一种将多个具有相同 RIP(the most Recent Instance in the Previous stack)域的事件进行链接聚合,以便在回溯搜索过程中执行批处理的 CEP 方法[11]。汤新提出了一种新的 CEP 引擎 Ceper[12],基于模式匹配树执行复杂事件处理。阳建坤等在传统的发布/订阅系统中引入复杂事件处理技术,提出了一种基于 CEP 的发布/订阅中间件[13]。但这些方法都没有较好地实现对 Negative 事件的处理。Wang 等提出了一种人工合成“伪事件”,主动执行 Negation 操作的方法[14],但其并不能解决时序和数值约束问题,也没有实现在 NFA 上提升 CEP 的性能。

文中提出了一种将 Negation 操作前置到 SSC 中的方法。通过动态去除 AIS 中不满足查询要求的原子事件,减少复杂事件处理的中间结果规模,同时减少序列构建时回溯搜索的代价,提高复杂事件处理的吞吐量。

1　基本概念

RFID 技术是一种自动对标签数据进行识别和获取的技术[15]。为了跟踪及监控特定的对象,人们将 RFID 标签附着在特定物品上,国际标准委员会制定的电子产品代码(Electronic Product Code,EPC)为每个物理对象定义了特定的识别码,使得每个附着在物理对象上的 RFID 标签携带有全球唯一的身份标识。由 RFID 阅读器负责读取物理对象所附标签信息,经解码后向主机返回数据[16]。

RFID 事件分为原子事件[17]和复杂事件两种。一个原子事件是由 RFID 阅读器扫描到物理对象的标签时所产生的信息,由阅读器身份标识、标签身份标识以及阅读器扫描到目标标签时的时间戳组成。其基本形式为<r,o,t>。其中,r 表示阅读器 ID;o 表示物理对象的 EPC;t 表示阅读器 r 阅读到物理对象 o 时的时间戳,时间戳标识了不同事件发生的先后顺序,这些与事件相关的数据被称为事件的属性[3]。而复杂事件是由多个原子事件按照一定的语义规则复合而成。通常使用大写字母来表示每种事件的类型(例如 E),而使用对应的小写字母来表示每种事件类型的实例(例如 e)。RFID 阅读器动态产生一系列数据代表着某些原子事件的发生,而原子事件中所包含的语义信息是非常有限的。在现实应用中,需要对动态产生的基本事件进行过滤、关联及匹配等操作,发现更高级的复杂事件[18],以获得更为有用的信息进行相关的智能应用。

以商场里的偷窃管理为例。假定在商店的货架、收银台和商场出口处布置有三种类型的 RFID 阅读器,商场内的每件商品都附有 RFID 标签,如果商品进入某个阅读器的扫描范围,阅读器将读取商品标签,并产生一个原子事件。由此,阅读器将产生三种不同类型的原子事件,假设这三种事件的类型分别为 SHELF_READING、COUNT_READING 和 EXIT_READING。当一个商品的标签被货架上的阅读器扫描到(商品被排放上货架),但没有在收银柜台被扫描到(没有到收银台结账),却随后被出口处的阅读器扫描到(该商品被企图带出商店),此时终端应用程序将立即触发警报,通知商场保安有人企图将商品非法带出。

CEP 查询的描述语言可以表示为如下形式:

EVENT　<事件模式>
WHERE　<约束条件>
WITHIN　<平滑窗口大小>

其中,EVENT 子句用于描述需要匹配查询的事件模式。例如在上文商场偷窃管理的例子中,EVENT 子句可以表示为:SEQ (SHELF_READING x, !(COUNT_READING y), EXIT_READING z)。其中,x,y,z 分别表示三种事件类型对应的事件实例,SEQ 是一种顺序结构,指定 x,y,z 三种事件实例发生的先后顺序,即检测是否出现一个 SHELF_READING 事件后,没有出现 COUNT_READING 类型事件的情况下就发生了 EXIT_READING 事件。其中 COUNT_READING 类型事件即为 Negative 事件,使用符号“!”来表示。相应的,对于不带符号“!”的事件,称之为积极(Positive)事件[19]。在本例中,Negation 操作负责检测在 SHELF_READING 类型事件与 EXIT_READING 类型事件之间是否存在一个 COUNT_READING 类型事件。

WHERE 子句用于描述 EVENT 子句中出现的事件实例所要满足的条件限制。例如在上面的例子中,WHERE 子句可以表示为:$x.id = y.id \wedge y.id = z.id$(也可以简单表示为[id])。其中,$x$,$y$,$z$ 分别为 EVENT 子句中三种事件类型所对应的事件实例;id 表

示事件实例的 EPC 代码。该 WHERE 子句要求 x,y,z 具有同样的 id,即表示同一件商品被货架上的阅读器扫描到,在没有被收银台的阅读器扫描到的情况下,就被出口处的阅读器扫描到。

WITHIN 子句用于描述事件发生的有效时间期限,即平滑窗口。复杂事件处理经常为需要检测的一系列事件设置一个平滑窗口,表示匹配查询的原子事件必须在规定时间内发生完成。例如在本例中,超市的营业时间持续 12 h,则可以设置平滑窗口大小为 12 h,表示 EVENT 子句中需要检测的事件(Negative 事件除外)必须在 12 h 内全部发生完成,否则视为无效。

2　一种前置非操作的复杂事件处理方法

2.1　复杂事件处理的过程模型

CEP 的主要任务是根据给定的查询要求扫描输入数据流,检测出满足特定要求的原子事件组合。这些原子事件将被转换为有一定语义的复杂事件,作为更高级应用的输入。

对于阅读器产生的输入事件流,CEP 将首先执行两个操作:

(1)对事件流进行扫描,检测出所需要的原子事件,同时过滤掉其他事件,该过程被称为序列扫描(Sequence Scan,SS)。

(2)将这些原子事件构建成复杂事件序列,该过程被称为序列构建(Sequence Construction,SC)。

序列扫描和序列构建总是一起使用,形成一个组合操作,并称为 SSC。

在 CEP 过程中,一个非常有效的方法是引入 NFA 来表示查询中复杂事件的结构。在传统方法中,首先从查询模型中提取出所有的积极事件类型,组成一个查询子序列。例如对于上文的例子,查询模型为 SEQ(SHELF_READING, ! COUNT_READING, EXIT_READING),从中提取出的积极事件类型形成的查询子序列为(SHELF_READING, EXIT_READING)。接着为对应的子序列创建 NFA,将该子序列中的事件一一映射为 NFA 中的状态。则该 NFA 包含“0”、“1”、“2”三个状态,“0”表示起始状态;在状态“0”时,若扫描到一个 SHELF_READING 类型事件时,将立即由状态“0”跃迁到状态“1”;若接着又扫描到一个 EXIT_READING 类型事件,则继续跃迁到状态“2”。跃迁时,不同状态之间的原子事件使用引用指针进行关联[6]。进入状态“2”时,表示当前已经完成了一个匹配过程,该状态被称为“可接受状态”。

现在假设在某一场景中布置有五种不同的阅读器,每一种阅读器扫描到一个附有 RFID 标签的物品时,将产生一个原子事件,则该场景包含五种类型的原子事件。假设这五种事件的类型分别为 A,B,C,D,E。当产生某个事件实例时,该事件实例将作为原始数据传输到 CEP 设备上进行处理。假定在某一时间段内,CEP 设备扫描到一组输入原子事件流,如图 1 所示。

事件流	a_1	c_2	b_3	a_4	d_5	b_6	d_7	c_8	d_9	b_{10}	d_{11}	d_{12}	b_{13}	e_{14}	c_{15}	b_{16}	d_{17}	b_{18}	e_{19}	d_{20}
$attr_1$值	1	1	1	2	1	2	1	2	2	1	2	1	2	1	2	2	2	1	2	1
$attr_2$值	1	1	1	1	2	1	1	1	1	1	2	2	1	2	1	1	1	1	2	1

图 1　原子事件流

在该输入事件流中,小写字母 a,b,c,d,e 分别表示五种事件类型的事件实例。每一个事件实例右下角的数字代表其对应的时间戳。除时间戳外,每一个事件实例都包含了两个属性 $attr_1$ 和 $attr_2$,分别被赋予了不同的值。

假设需要对输入的事件流执行如下 Q_1 所示的查询,根据 Q_1 检测出对应的复杂事件,以便用于后续的高级应用。

Q_1:EVENT　SEQ(A a, B b, !(C c), D d)

WHERE[$attr_1$] ∧ d. $attr_2$ = ‘1’

WITHIN20

该查询中包含三条子句,其中 EVENT 子句包含四种不同的事件类型,SEQ 为 sequence 的缩写,SEQ 结构指明了对原子事件发生的先后顺序的要求,即 A,B,D 必须依次发生。! C 表示不允许在 B 类事件和 D 类事件之间发生任何 C 类事件,即 c 为 Negation 事件。WHERE 子句指明了对事件的条件约束。其中,[$attr_1$] 表示所要检测的原子事件实例 a,b,c,d 的属性值 $attr_1$ 必须相等,即要求 $a.attr_1 = b.attr_1 \wedge b.attr_1 = c.attr_1 \wedge c.attr_1 = d.attr_1$。而 $z.attr_2 = 1$ 表示事件实例 z 的属性值 $attr_2$ 必须为 1。为了方便表达和操作,使用事件数量来表示平滑窗口 W 的大小。如在查询 Q_1 中,窗口大小为 20,表示每个平滑窗口内容纳 20 个事件。

对于上文所示的输入数据流执行查询 Q_1,SASE 算法[4]在 SSC 过程中将产生 8 个结果:

$<a_1,b_3,d_7>$, $<a_1,b_3,d_{20}>$, $<a_1,b_{10},d_{20}>$, $<a_1,b_{18},d_{20}>$, $<a_4,b_6,d_9>$, $<a_4,b_6,d_{17}>$, $<a_4,b_3,d_{17}>$, $<a_4,b_{16},d_{17}>$

但这 8 个结果并没有达到最终的查询要求,称为“中间结果”。

为了获得需要的复杂事件,SASE 还需对以上 8 个中间结果进行 Negation 操作。Negation 操作包含两个部分:

(1)检测是否有某个 C 类型的 Negative 事件在特定的事件间隔出现。

(2)如果存在,则判断该事件是否满足 WHERE 子句中的约束条件。

任何满足以上两个条件的 C 类型事件将进一步对中间结果进行过滤。

对以上事件序列实例执行 Negation 操作后，得到最终需要的复杂事件为：

$<a_1, b_3, d_7>$，$<a_4, b_{18}, d_{20}>$，$<a_4, b_{13}, d_{17}>$，$<a_4, b_{16}, d_{17}>$

2.2　PNCEP：前置非操作的复杂事件处理方法

SASE 虽然经过一系列改进，减少了 SSC 过程的中间结果数量，提升了事件处理的吞吐量，但并没有论述如何实现将 Negation 操作前置到 SSC 中。文中在 SASE 的基础上，提出将 Negation 操作前置到 SSC 中的具体方法，并把它命名为 PNCEP（Pushing Negation into sequence scan and construction in Complex Event Processing）。

由于前置 Negation 操作可以在序列扫描过程中动态去除 AIS 中不满足查询要求的原子事件，因此 PNCEP 减少了 CEP 的中间结果规模。同时，由于前置 Negation 操作动态地减少了 AIS 中的事件数量，大大减少在序列构建时进行回溯搜索的代价，因此 PNCEP 也能提高 CEP 的吞吐量。下面以上文所示的事件输入流和查询 Q_1 为例，一步一步介绍 PNCEP 方法。

步骤 1：对于查询 Q_1，首先为其创建 NFA，如图 2 所示。

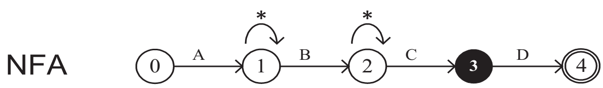

图 2　为事件序列 SEQ(A,B,！C,D)创建的 NFA

其中，状态“4”为可接受状态，采用两个同心圆来表示。在状态“1”和状态“2”的上方有一个由“ * ”标识的自环，表示若扫描的事件不能引发跃迁，则回到原来的状态。与传统方法不同的是，PNCEP 在构建 NFA 时，并不去除 EVENT 子句中的 Negative 事件，而直接针对事件序列（A，B，！C，D）构建 NFA。

从图 2 可以看到，在 NFA 中除了包含积极事件类型 A、B、D 所引发的跃迁状态外，还包括 Negative 事件类型 C 所引发的跃迁状态，将其命名为 Negative 状态。为了醒目地表示 Negative 状态，PNCEP 使用一个背景为黑色，字体为白色的圆来表示。

步骤 2：对于每一个 NFA 的状态，除初始状态外，都在其下创建一个 AIS，用于存储触发状态跃迁的事件。

在每一个 AIS 中，从上到下，存储的为同一类型的事件，在每个事件的前面有一个用括号括起来的指针域，称为 RIP 字段。设置 RIP 域的目的是方便在序列构建过程中进行回溯搜索。但与 SASE 中 NFA 的跃迁操作不同，PNCEP 的跃迁操作如算法 1 所示：假设 S_1，S_2 为 NFA 中任意两个相邻的状态，其中 S_2 为 S_1 的下一状态。假定 Negative 状态不会出现在 NFA 的开始（指状态‘1’）和末尾。

算法 1：PNCEP 的跃迁操作处理过程。

1：扫描输入事件流
2：扫描到一个事件 e
3：IF（e 与引发状态跃迁的事件类型都不相同）
4：GOTO 第 1 行
5：END IF
6：e 与引发状态 S_1 跃迁到状态 S_2 的事件类型相同
7：IF（S_1 不为 Negative 状态）
8：IF（S_1 为初始状态）
9：将 e 加入到状态 S_2 下的 AIS 中。设置 e 的 RIP 域为空
10：GOTO 第 1 行
11：END IF
12：IF（S_1 的 AIS 不为空）
13：将 e 加入到状态 S_2 下的 AIS 中。设置 e 的 RIP 域为 S_1 中具有最大时间戳的事件
14：GOTO 第 1 行
15：END IF
16：GOTO 第 1 行
17：END IF
18：IF（S_1 前一状态的 AIS 不为空）
19：将 e 加入到状态 S_2 下的 AIS 中。设置 e 的 RIP 域为 S_1 前一状态中具有最大时间戳的事件
20：GOTO 第 1 行
21：END IF
22：GOTO 第 1 行

对于上文中的输入数据流，首先依据输入事件的属性 $attr_1$ 的值将事件流进行分组，形成分组活动实例栈（Partition Active Instance Stack，PAIS），由于输入数据流中的所有事件的 $attr_1$ 取值为 1 或 2，因此 PAIS 中共包含两个分组，每个分组内的事件实例的 $attr_1$ 值都相同。

在序列扫描的过程中，一旦某个事件 e 进入“可接受状态”下的 AIS 中，序列构建机制将立即被激活。序列构建的过程为：以 e 为起点，沿着 RIP 域进行回溯，搜索整个 AIS，直到到达最左边的 AIS 为止。与 SASE 不同，在 PNCEP 算法中，并不是所有 RIP 域都指向前一状态的 AIS。当遇到 Negative 状态时，RIP 域将绕过 Negative 状态的 AIS，而指向更前的 AIS。在回溯操作之前，Negative 状态下的 AIS 必须为空，因为任何 Negative 事件的存在将会导致所构建的序列不满足要求。后面的文章中将详细描述如何使得回溯之前 Negative 状态下的 AIS 为空。

当扫描到 d_7 时，NFA 进入可接受状态。此时序列构建即被激活，以 d_7 为起点，通过 RIP 域进行回溯搜

索。在回溯搜索的过程中,检测所遇到的事件是否满足 WHERE 子句中的约束条件,由于采用了 PAIS,使得每个分组 AIS 中所有事件的 $attr_1$ 值都相同,因此同样只需检验“$d.attr_2$ = ‘1’”。通过回溯算法,可以构建事件序列<a_1,b_3,d_7>,如图 3 所示。

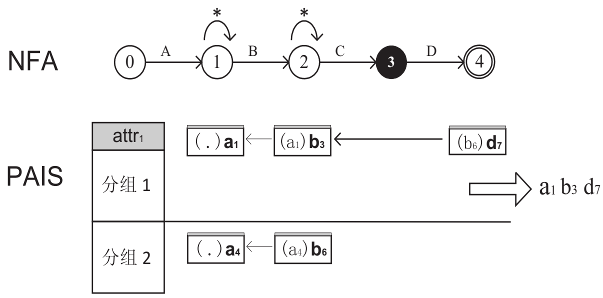

图 3　扫描到事件 d_7 时的 NFA

接着将可接受状态下的 AIS 中,触发序列构建的事件删除。

步骤 3:继续序列扫描过程,当扫描到一个 Negative 事件时,即将其加入到状态 3 对应的 AIS 中。

图 4 展示了扫描到事件 c_8 时的 NFA,由于 c_8 为一个 Negative 事件,且其属性 $attr_1$ 值为 2,因此被加入到“分组‘2’”的 Negative 状态下的 AIS 中。

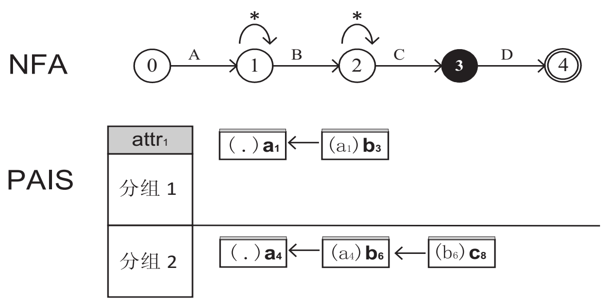

图 4　扫描到事件 c_8 时的 NFA

在序列扫描的过程中,一旦出现某个 C 类事件实例,则立即判断该事件是否满足 WHERE 子句的约束条件。若满足,则将其加入到 Negative 状态下的 AIS 中。接着根据其 RIP 域移除前一状态 AIS 中的事件,随后再移除该 Negative 事件。

由于 c_8 满足 WHERE 子句中所有的约束件,且 c_8 的 RIP 域为 b_6,因此 b_6 所在的 AIS 中,所有发生在 b_6 之前的事件(也包括 b_6)必定无效,因此可以从 AIS 中去除 b_6。去除 b_6 后,在当前分组的 AIS 中,将不再有任何 B 类事件发生在事件 c_8 之前,此时可以移除 c_8。

步骤 4:重复步骤 2 和 3,实现对数据流的处理。图 5 展示了当扫描到事件 d_{20} 时的 NFA,以及通过回溯搜索产生的复杂事件结果。

此时 NFA 通过在 AIS 中采用回溯搜索算法,得到三个复杂事件结果:<a_4,b_{18},d_{20}>,<a_4,b_{13},d_{17}>,<a_4,b_{16},d_{17}>。加上在图 3 中创建的复杂事件结果<a_1,b_3,d_7>,共产生四个结果。由于将 Negative 操作下推到 SSC 中,PNCEP 算法产生的中间结果数量显著减少,而 SASE 在 SSC 过程中则产生了 8 个中间结果。

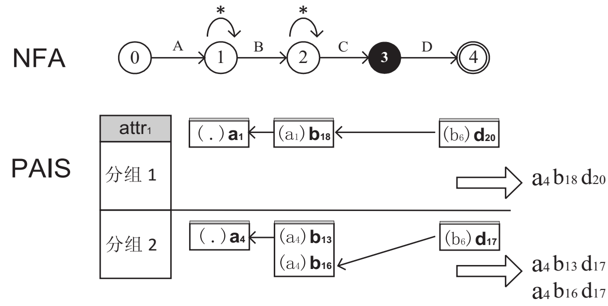

图 5　扫描到 d_{20} 时的 NFA 以及所产生的事件序列

假定 Negative 状态不会出现在 NFA 的开始(指状态‘1’)和末尾,算法 2 描述了 PNCEP 的复杂事件处理过程。假设任意两个相邻的状态 S_1,S_2,其中,S_2 为 S_1 的下一状态。假设任意一个事件 e_0。

算法 2:PNCEP 的复杂事件处理过程。

1:扫描输入事件流

2:扫描到一个事件 e

3:IF(事件 e 不能引发跃迁)

4:GOTO 第 1 行

5:END IF

6:e 引发状态 S_1 跃迁到状态 S_2

7:根据算法 1 将事件 e 加入到状态 S_2 下的 AIS 中,并设置 e 的 RIP 指向事件 e_0

8:IF(S_2 为 Negative 状态)

9:移除 e_0 所在的 AIS 中所有发生在 e_0 前事件以及 e_0

10:将 e 从 AIS 中移除

11:GOTO 第 1 行

12:END IF

13:IF(S_2 为可接受状态)

14:以 e 为起点,沿着 RIP 域进行回溯,进行序列构建

15:将 e 从 AIS 中删除

16:GOTO 第 1 行

17:END IF

18:将进入平滑窗口外的事件从 AIS 中移除

19:GOTO 第 1 行

由以上分析可知,引入 Negative 状态的作用是动态地移除前一状态中不满足查询要求的事件实例。Negative 事件加入到 Negative 状态下的 AIS 中后,则立即移除前一状态中 AIS 内的相应事件,随后 Negative 事件亦立即被移去。Negative 事件只是在 AIS 中存在很短的一瞬间,因此在大多数情况下 Negative 状态的 AIS 为空(包括序列构建时)。

3　性能评估

为了确定文中提出算法的优越性,将 PNCEP 与 SASE 两种算法进行吞吐量和中间结果的比较。

实验环境为:AMD 速龙 II 代 2.3 GHz 处理器,4 GB 主存,操作系统 Windows XP professional,运行平台 J2SE 1.8。

为了对本系统进行测试,编写了事件生成器,以产生一系列事件。在实验中,定义了 5 种基本事件类型,通过随机生成算法,每种事件的生成概率都为 20%。并编写了两个事件处理器,分别模拟 PNCEP 和 SASE 的复杂事件处理过程。同时创建了一个查询生成器,所构建的查询的 EVENT 子句为 SEQ($E_1,E_2,\cdots,E_L$)。其中,$E_1,E_2,\cdots,E_L$ 代表事件类型,L 代表查询中的事件长度。使用事件数量来表示平滑窗口 W 的大小。

在实验中,将查询长度 L 设置为 5,Negative 事件数量设置为 1,将窗口大小 W 设置为从 500 到 900 变化。

图 6 展示了实验结果中两种算法的吞吐量的比较,其中横轴表示窗口大小,纵轴表示处理原子事件的吞吐量。

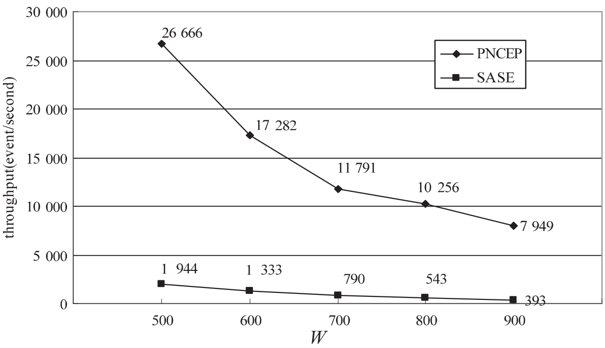

图 6　变更窗口 W 时两种算法的吞吐量比较

从图中可以看到,随着窗口大小的增加,两种算法的吞吐量都出现明显的下降,但总体上看,PNCEP 算法的吞吐量远高于传统的 SASE 方法。当平滑窗口大小为 500 时,PNCEP 性能提升最小,约为 SASE 的 13.7 倍。当窗口大小为 900 时,PNCEP 性能提升最大,约为 SASE 的 20.2 倍。主要原因是随着平滑窗口大小的增加,窗口内所能容纳的事件数量增加,导致在回溯搜索中搜索路径的增加,增加了序列构建的代价。因此,虽然两种算法的吞吐量都有不同程度的下降,但 PNCEP 算法能在序列扫描过程中,通过前置 Negation 操作动态地去除窗口内不满足查询要求的事件,减少了回溯搜索的路径数量,在一定程度上减少了序列构建的代价。

图 7 展示了实验结果中两种算法所产生的中间结果数量的比较,其中横轴表示窗口大小,纵轴表示每扫描到 1 000 个原子事件时所产生的中间结果数量。

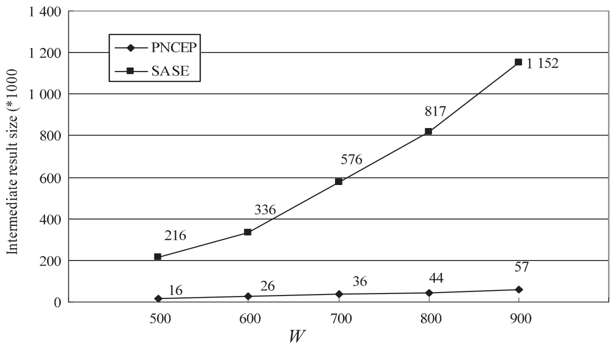

图 7　变更窗口 W 时两种算法的中间结果比较

从图中可以看到,随着窗口大小的增加,两种算法所产生的中间结果数量都明显增加,但总体上看,PNCEP 算法每 1 000 个原子事件所产生的中间结果数量远低于传统的 SASE 方法。当平滑窗口大小为 500 时,PNCEP 所产生的中间结果数量约为 SASE 的 1/13。当窗口大小为 900 时,PNCEP 所产生的中间结果数量约为 SASE 的 1/20。主要原因是随着平滑窗口大小的增加,在序列扫描过程中 AIS 所能容纳的原子事件数量随即增加。由这些原子事件所能构建的复合事件序列迅速增加,因此导致两种算法所产生的中间结果数量都迅速增加。但 PNCEP 算法能在序列扫描过程中,提前去除掉不满足查询要求的原子事件,减少了序列构建过程中的中间结果数量,减轻了对内存的占用。

4　结束语

文中提出了一种将 Negation 操作前置到 SSC 中,以提高复杂事件处理吞吐量,并减少中间结果规模的 PNCEP 算法,并详细介绍了如何实现该方法以提升复杂事件处理性能。实验结果显示,新算法的吞吐量高于传统的 SASE 方法,而中间结果规模低于 SASE,在复杂事件处理上具有比较明显的优势。但文中没有涉及到两种出现消极事件的情况,即“非开始”(如 SEQ(! A,B))和“非结束”(如 SEQ(A,! B))。对于“非开始”和“非结束”的处理将作为未来的工作进行进一步研究。

参考文献:

[1] Sheng B,Li Q,Mao W. Efficient continuous scanning in RFID systems[C]//Proceedings of international conference on computer communications. San Diego:IEEE,2010:1-9.

[2] Yang L,Han J S,Qi Y,et al. Season:shelving interference and joint identification in large-scale RFID systems[C]//Proc of IEEE INFOCOM. [s. l.]:IEEE,2011:3092-3100.

[3] 江连峰,赵佳宝. 复杂事件处理技术及其应用综述[J]. 软件,2014,35(2):188-192.

[4] Wu E,Diao Y,Rizvi S. SASE:high-performance complex e-

vent processing over streams[C]//Proceedings of ACM conference on management of data. Chicago: ACM, 2006: 407-418.

[5] 陈　琳,彭商濂,尹方鸣,等. 多维度的 RFID 复杂事件处理优化算法研究[J]. 计算机仿真,2009,26(8):360-364.

[6] 陈　远,李战怀,陈　群. 不可靠 RFID 数据上的复杂事件处理研究[J]. 计算机应用研究,2009,26(7):2537-2539.

[7] Tao K,Zhu Y L,Hu K Y,et al. A novel distributed complex event processing for RFID application[C]//Proceedings of third international conference on convergence & hybrid information technology. [s. l.]:IEEE,2008:1113-1117.

[8] Wang Yongheng, Yang Shenghong. High-performance complex event processing for large-scale RFID applications[C]//Proceedings of 2nd international conference on signal processing systems. [s. l.]:[s. n.],2010:127-131.

[9] Liu Y,Wang D. Complex event processing engine for large volume of RFID data[C]//Proceedings of second international workshop on education technology and computer science. [s. l.]:[s. n.],2010:429-432.

[10] Bok K S,Yeo M H,Lee B Y,et al. Efficient complex event processing over RFID streams[J]. International Journal of Distributed Sensor Networks,2012,2012:71-81.

[11] Nishimura N,Kawashima H,Kitagawa H. A high throughput complex event detection technique with bulk evaluation[C]//Proceedings of the 2013 eighth international conference on P2P,parallel,grid,cloud and internet computing. [s. l.]:[s. n.],2013:624-629.

[12] 汤　新. Ceper:一个高性能复杂事件处理引擎[J]. 电脑与信息技术,2013,21(4):32-37.

[13] 阳建坤,祖向荣. 一种基于 CEP 发布订阅中间件应用研究[J]. 中国科技信息,2014(24):103-104.

[14] Wang F S,Liu S R,Liu P,et al. Bridge physical and virtual worlds: complex event processing for RFID data streams[C]//Proc of EDBT. [s. l.]:[s. n.],2006:588-607.

[15] Fuhrer P,Guinard D,Liechti O. RFID:from concepts to concrete implementation[C]//Proceedings of the international conference on advances in the internet, processing, systems and interdisciplinary research. [s. l.]:[s. n.],2006:1-12.

[16] 彭小娟,刘世安,熊春如,等. 复杂事件处理在大规模 RFID 数据通信中的应用研究[J]. 化工自动化及仪表,2009,36(4):76-79.

[17] Chakravarthy S,Krishnaprasad V,Anwar E,et al. Composite events for active databases:semantics, contexts and detection[C]//Proceedings of international conference on very large data bases. [s. l.]:[s. n.],1994:606-617.

[18] 谷　峪,于　戈,张天成. RFID 复杂事件处理技术[J]. 计算机科学与探索,2007(3):255-267.

[19] Akdere M,Cetintemel U,Tatbul N. Plan-based complex event detection across distributed sources[C]//Proceedings of international conference on very large data bases. [s. l.]:[s. n.],2008:66-67.

(上接第 155 页)

数据的冗余;提出了最小包围盒二叉树算法,用以建立空间数据索引。该算法具有数据结构简单、树结构平衡、索引性能好等优点。实验最终结果表明,提出的管理机制可以有效管理海量遥感数据。

参考文献:

[1] Nativi S,Mazzetti P,Santoro M,et al. Big data challenges in building the global earth observation system of systems[J]. Environmental Modelling & Software,2015,68:1-26.

[2] Yang C,Goodchild M,Huang Q,et al. Spatial cloud computing:how can the geospatial sciences use and help shape cloud computing? [J]. International Journal of Digital Earth,2011,4(4):305-329.

[3] Chang F,Dean J,Ghemawat W C,et al. Bigtable:a distributed storage system for structured data[J]. ACM Transactions on Computer Systems,2008,26(2):1-14.

[4] 李国斌. 空间数据库技术[M]. 北京:电子工业出版社,2010.

[5] 黄　杰,刘仁义,刘　南,等. 海量遥感影像管理与可视化系统的研究与实现[J]. 浙江大学学报:理学版,2008,35(6):701-706.

[6] 陈金水,王　崟. 非结构化数据存储管理的实用化方法[J]. 计算机与现代化,2006(8):25-28.

[7] van Oosterom P. Scaleless topological data structures suitable for progressive transfer:the gap-face tree and gap-edge forest[J]. Cartography and Geographical Information Science,2005,32(4):331-346.

[8] 李东军,曾国荪. 一种基于四叉树的空间数据缓存策略[J]. 计算机工程与应用,2008,44(22):162-165.

[9] 唐立文,廖学军,汪荣峰. 基于四叉树的海量空间数据模型研究[J]. 装备指挥技术学院学报,2007,18(2):70-74.

[10] 王晨星. 海量遥感影像数据管理系统的设计与实现[D]. 北京:装备学院,2012.

[11] 李文琦. 数字地球中的非结构化数据存储与管理方案研究[D]. 北京:装备学院,2010.

[12] 邓红艳,武　芳,翟仁健,等. 一种用于空间数据多尺度表达的 R 树索引结构[J]. 计算机学报,2009,32(1):177-184.

[13] 艾廷华,帅　赟,李精忠. 基于形状相似性识别的空间查询[J]. 测绘学报,2009,38(4):356-362.

[14] 董　鹏,杨崇俊,芮小平,等. 一种基于改进四叉树的 GIS 空间选择查询算法—以 ESRI SHAPE 格式文件为例[J]. 计算机工程与应用,2013,39(13):58-61.

一种前置非操作的复杂事件处理方法

作者： 杨晟， 杨颖， YANG Sheng， YANG Ying
作者单位： 广西大学 计算机与电子信息学院,广西 南宁,530004
刊名： 计算机技术与发展ISTIC
英文刊名： Computer Technology and Development
年，卷(期)： 2015,25(11)