Software Engineering and Applications 软件工程与应用, 2018, 7(2), 75-83
Published Online April 2018 in Hans. http://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2018.72009

Hans汉斯

Research on Query Plan Generation and Optimization in Multi-Source Complex Event Detection

Chang Su, Husheng Liao, Hongyu Gao, Jiawei Zhang

Faculty of Information Technology, Beijing University of Technology, Beijing
Email: su_chang@yeah.net, liaohs@bjut.edu.cn, hy_gao@bjut.edu.cn, zhangjiawei217@163.com

Received: Apr. 1st, 2018; accepted: Apr. 11th, 2018; published: Apr. 19th, 2018

Abstract

Complex event processing technology is a common method of streaming data query. With the development of data diversification, the complexity of query increases dramatically, and its processing performance is greatly challenged. In order to improve the processing efficiency of multi-source complex event query, this paper proposes a new query decomposition scheme to improve the utilization rate of cluster resources. For the matching of event sources, we combine the filtering in query with the filtering in event source matching to design a special optimization program. Experimental results show that the above method can effectively improve the efficiency of complex event query.

Keywords

Complex Event Processing, Pattern Match, Query Optimization

多源复杂事件检测中查询计划生成与优化技术的研究

苏　畅，廖湖声，高红雨，张嘉伟

北京工业大学信息学部，北京
Email: su_chang@yeah.net, liaohs@bjut.edu.cn, hy_gao@bjut.edu.cn, zhangjiawei217@163.com

收稿日期：2018年4月1日；录用日期：2018年4月11日；发布日期：2018年4月19日

摘　要

复杂事件处理技术是流数据查询的常见手段，随着数据多元化的发展，查询的复杂度急剧提升，其处理

文章引用: 苏畅, 廖湖声, 高红雨, 张嘉伟. 多源复杂事件检测中查询计划生成与优化技术的研究[J]. 软件工程与应用, 2018, 7(2): 75-83. DOI: 10.12677/sea.2018.72009

性能面临极大地挑战。为了提高多源复杂事件查询的处理效率，本文提出一种新的查询分解方案，提高集群资源利用率;针对事件源匹配,将查询中的筛选与事件源匹配中的筛选相结合设计了专用优化方案。实验证明以上方法能够有效地提高复杂事件查询效率。

关键词

复杂事件处理，模式匹配，查询优化

1. 引言

近年来，大数据相关领域飞速发展，其应用系统单位时间内所处理的数据呈爆炸式增长，产生的数据也规模随之扩大。诸如安全、医疗、金融、交通等应用对系统的实时响应时间有较高的要求[1]。传统的“存储－查询”和“发布－订阅”的方式无法满足诸如欺诈识别、病情监测、金融交易，交通路况等应用系统对流数据实时处理的需求[2]。复杂事件处理(Complex Event Processing, CEP)技术能够很好地解决此类问题[3] [4] [5]。

复杂事件处理以事件驱动(Event Driven)的方式来处理海量数据，能够对蕴含在事件间的逻辑关系进行模式匹配，生成具有一定抽象层次、符合业务需求的高级事件。数据多元化的迅猛发展，使得复杂事件处理系统数据源出现分散的情况[2]。根据这一特点，有研究者将分布式系统与复杂事件处理系统相结合，提高了系统对多事件源查询的处理能力[6]。随着系统查询功能的丰富，查询代价也在随之增加，系统的处理性能方面也面临着巨大的挑战[7] [8]。

复杂事件处理系统中影响运行效率的主要两个方面分别是事件源的匹配方法以及查询的执行策略[9]。本文的主要贡献：

1) 提出一种新的查询分解方案，充分利用集群资源，提高系统处理效率；
2) 将事件源匹配与查询筛选相结合设计专用优化方案，减少通信开销；
3) 设计查询计划构造方法，便于系统进行合理优化。

2. 动机

复杂事件处理系统是基于某种类似数据库查询语言来进行查询的，这种查询语言属于声明型高级语言。使用者在描述一个具体查询时，不会考虑查询的执行效率。因此系统需要对查询任务进行分析，考虑查询任务的不同执行策略，从中选取最优策略进行执行。在此结合查询案例 Q，阐明当前存在的问题：

```
create stream $S0 {<alert($S1.$v) />}        // 输出结果的构造
from $S1 match $M1, $S2 match $P2, ($S1 match $M2, $S3 match $P3, $S2 match $P2)
temporal meet        // 时序关系
where $S1.$v < 50 and $S2.$v > 100        // 谓词筛选
within time(60)        // 时间窗口
```

from 子句描述多源复杂事件检测表达式，表达式中对于不同事件源的匹配可以并行处理。因此，其

对应的查询计划如图 1 所示，图中“()”中内容表示当前操作的时序关系。对于各事件源所依赖的正规树模式[10]定义如下所示，该方法是一种面向半结构化数据流的事件约束模式，能够按照正规式的方式来描述事件间的时序约束关系以及结构约束关系。

```
create pattern $M1 (/buy {$n, $v})      // 匹配的模式名称
where tag($n) = ‘name’ and tag($v) = ‘value’ and $v < 100      // 谓词筛选
create pattern $M2 (/buy {$n, $v})
where tag($n) = ‘name’ and tag($v) = ‘value’ and $v > 10
create pattern $P2 (/buy {$n, $v})
where tag($n) = ‘name’ and tag($v) = ‘value’
create pattern $P3 (/buy {$n, $v})
where tag($n) = ‘name’ and tag($v) = ‘value’
```

通过以上案例可以看出，当且仅当查询中的筛选条件作用于某个事件源时，可以结合其所依赖的正规树模式中的谓词筛选进行优化。合理的将查询中的筛选前置与模式匹配中，可以有效的减少系统内部的通信开销，从而提升系统的查询效率。同时也存在对于同一事件源匹配不同模式的情况，例如事件源 S1 所依赖的正规树模式就有两种 M1 和 M2，在同一系统中对于相同事件源做多次模式匹配显然对性能有所损耗，因此，需要对此方面进行改进。

3. 优化方法

为了便于以上问题的优化，本文将查询进行分解，并针对以上发现的问题，提出以下几种措施进行优化。将查询任务进行合理分解，利用分布式集群来处理多源数据流的匹配，调整集群的部署策略来提高系统性能；采用一些常用的优化方法对本文中的查询计划进行优化；结合系统的特性，考虑正规树模式匹配的特点，对模式采取一些专用优化方案。

3.1. 分布式查询优化

表 1 描述了可分解部分 S 和不可分解部分 R 对应的文法。根据该文法，本文提出一种新的分解方案，对于局部无法分解的部分，构建自动机来进行事件检测；对于可分解部分，则根据其所执行的具体操作

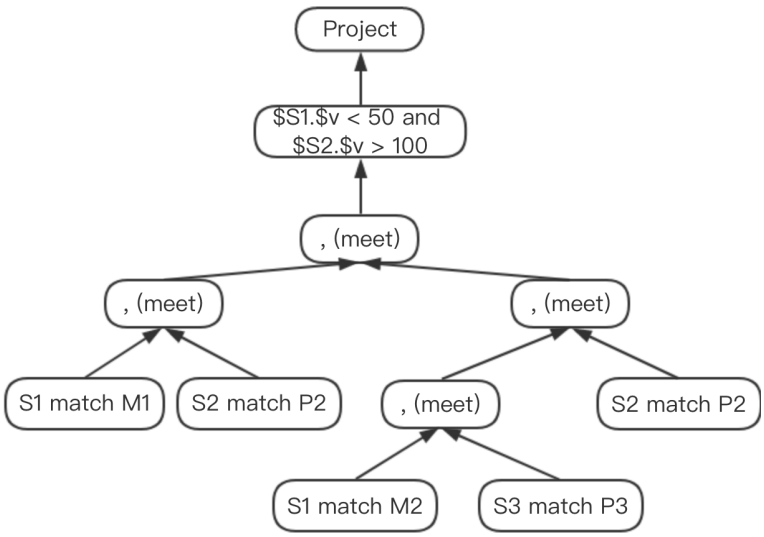


Figure 1. Query plan of Q
图 1. 案例 Q 的查询计划

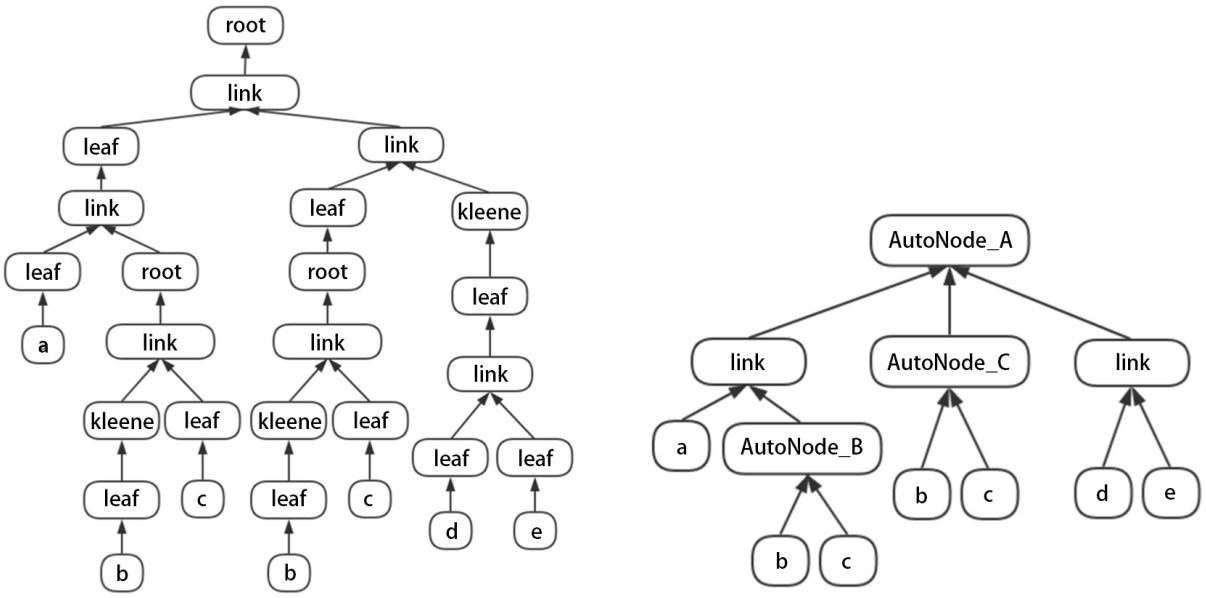
构造相应的查询算子。

对于事件模型(a, (b*, c)), ((b*, c), (d, e)*)，将该模型根据表 1 中的文法可确定存在不可分解的部分。根据分解算法，构造出查询计划如图 2(a)所示。其中表达式 b*，c 为一个不可分解的子任务，此任务的输出与 a 事件的连接操作作为一个可分解的部分，与 d，e 的克林操作又作为一个不可分解部分。查询计划构造算法中，将此类的不可分解的子任务用 root 节点来标识子树的顶部，用 leaf 节点来标识子树的叶子叶节点，这样有利于提取不可分解部分，将其交给自动机处理单元进行处理。

对于该事件模型，经过分解算法处理，一共包含三个不可分解部分，即需要构造三个自动机。这三个自动机的关系具体如图 2(b)所示。AutoNode_A 作为不可分解部分构造的自动机，其拥有三个输入事件源。其中事件源 a 与自动机 AutoNode_B 进行连接操作后的输出结果作为 AutoNode_A 的一个输入事件源；自动机 AutoNode_C 的输出结果作为 AutoNode_A 的一个输入事件源；最后一个输入事件源是一个简单的连接操作。这样可以将原来不可分解的事件检测表达式进行拆分，并应能够合理分配集群的资源，从

Table 1. System resulting data of standard experiment
表 1. 多源组合事件的文法描述

可分解部分	不可分解部分
$S \rightarrow S_1$	$R \rightarrow S$ “,” R_1
$S \rightarrow S$ “|” S_1	$R \rightarrow S$ “|” R_1
$S_1 \rightarrow S_2$	$R \rightarrow R_1$
$S_1 \rightarrow S_1$ “,” S_2	$R_1 \rightarrow R_2$
$S_2 \rightarrow$ ID	$R_1 \rightarrow R_2$ “|” R_1
$S_2 \rightarrow$ “(” S “)”	$R_2 \rightarrow S$ “*”
$S_2 \rightarrow R$	$R_2 \rightarrow S$ “*”“,” R_2



(a) (b)

Figure 2. Query plan (a) and topology structure (b)
图 2. 查询计划(a)与拓扑结构(b)

而提高系统查询效率。

3.2. 专用优化

CEStream 复杂事件处理系统采用正规树模式来进行事件流的匹配。为了提高系统的查询效率，考虑扩大正规树模式匹配的应用范围，提出两种优化方法。现结合图 1 中的查询计划进行阐述。

3.2.1. 模式合并

为了合理利用集群中各个节点的资源，可以将查询计划中具有相关性的节点进行合并，使其共用一个集群节点资源，减少冗余节点有利于节省系统资源。例如，对于筛选条件仅作用于某一个模式时，可以将该筛选条件与此模式中的筛选条件进行合并，不仅减少计算节点的占用，还能提前将事件流中的无用事件剔除，从而减少系统内部的通信开销。对上述案例进行常规优化操作后，对于事件源 S1 和 S2 匹配后的事件流的谓词筛选与其所匹配的模式进行合并，构造新的模式 P2’，同理构造 M1’和 M2’，合并后的查询计划如图 3 所示。

```
create pattern $P2’ (/buy($n, $v))
where tag($n) = ‘name’ and tag($v) = ‘value’ and $S2.$v > 100
同理构造 M1’和 M2’：
create pattern $M1’ (/buy($n, $v))
where tag($n) = ‘name’ and tag($v) = ‘value’ and $v < 50
create pattern $M2’ (/buy($n, $v))
where tag($n) = ‘name’ and tag($v) = ‘value’ and $v > 10 and $v < 50
```

3.2.2. 公共模式提取

事件检测表达式中可能存在同一事件源匹配不同模式的情况，为了进一步降低系统内部的通信开销，可以将相似的模式匹配进行整理，提取出公共匹配部分，使公共部分的匹配结果达到一次匹配多次使用的效果，从而提升节点利用率。对于上述案例，事件流 S1 所匹配的模式 M1’，M2’求交集，提取公共模式 M，并修改查询计划，如图 4 所示。

构造公共模式 M：

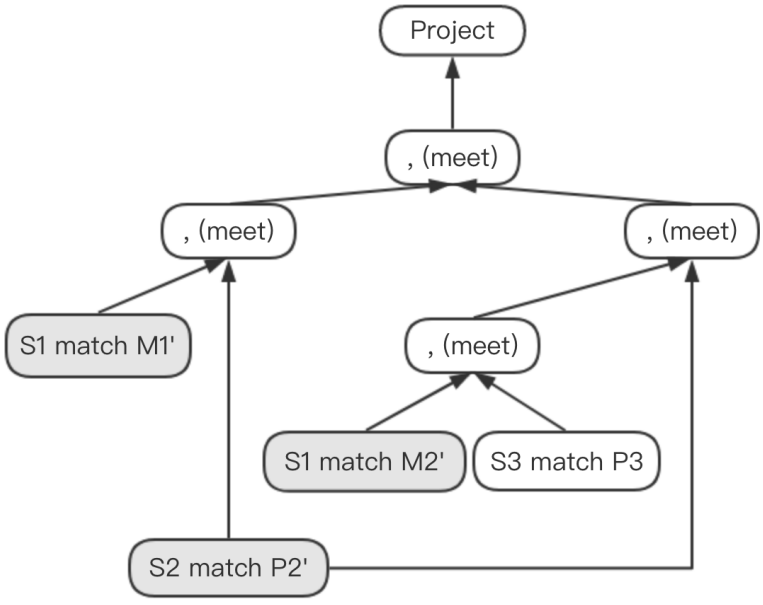


Figure 3. Pattern merged query plan
图 3. 模式合并后的查询计划

create pattern $M (/buy($n, $v))
where tag($n) = ‘name’ and tag($v) = ‘value’ and $S1.$v < 50

4. 实验结果与分析

本章针对本文提出的复杂事件处理系统的优化方法进行性能测试。测试案例采用 XML 文本作为系统的输入源，针对查询分解和模式优化根据系统吞吐量以及响应时间进行测试，采用控制变量的方法进行多次测试。测试案例中所涉及到的模式定义如表 2 所示，事件检测表达式以及其约束条件如表 3 所示。

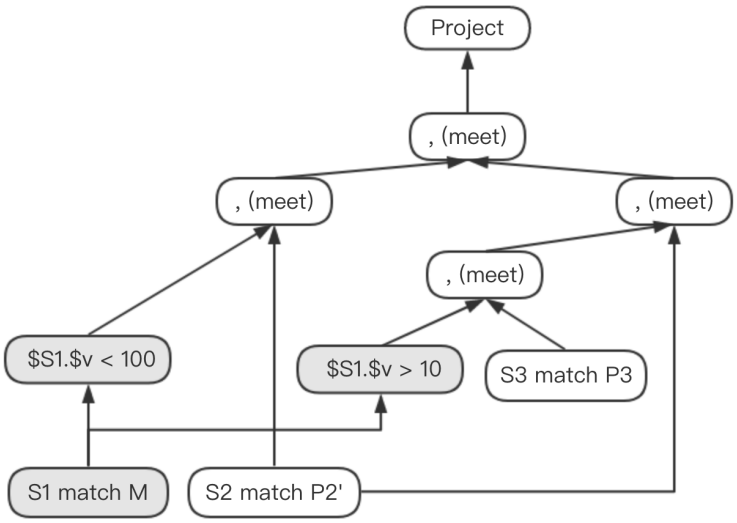


Figure 4. Common pattern of query plan
图 4. 公共模式构造后的查询计划

Table 2. Test case—pattern define
表 2. 测试案例——模式定义

名称	模式定义
P1	create pattern $P1 (/buy{$n, $v}) where tag($n) = ‘number’ and tag($v) = ‘value’ and $v > 1000
P2	create pattern $P2 (/sell{$n, $v}) where tag($n) = ‘number’ and tag($v) = ‘value’ and $v < 50
P3	create pattern $P3 (/warning{$n, $v}) where tag($n) = ‘number’ and tag($v) = ‘value’
P2	create pattern $P4 (/buy{$n, $v}) where tag($n) = ‘number’ and tag($v) = ‘value’ and $v < 100

Table 3. Test case—event detection expression
表 3. 测试案例——事件检测表达式

案例名称	事件检测表达式	约束条件
Q1.1	s1-p1, s2-p2*, s3-p3,s4-p4*	无
Q1.2	s1-p1 | (s2-p2*, s3-p3*)	无
Q1.3	s1-p1, (s2-p2*, s3-p3)*	无
Q2.1	s1-p1, s2-p2, s3-p3	$S1.$v > 900
Q2.2	s1-p1, s2-p2*, s3-p3	$S2.$v > 40
Q3.1	s1-p1, s2-p2, s1-p1, s3-p3	无
Q3.2	s1-p1, s2-p2, s1-p4, s3-p3	无

表 2 模式定义中的 tag 操作可以将事件的具体属性绑定到变量上，便于系统对查询的描述。在 where 子句中对所匹配的模式进行相应的约束。

案例 Q1 是原不可分解现在可分解的情况的对比，其吞吐量对比结果如图 5 所示，响应时间对比结果如图 6 所示。案例 Q2 是针对筛选移动或模式合并的对比，其吞吐量对比结果如图 7 所示，响应时间对比结果如图 8 所示。案例 Q3 是对于涉及公共表达式提取或公共模式构建的对比，其吞吐量对比结果如图 9 所示，响应时间对比结果如图 10 所示。

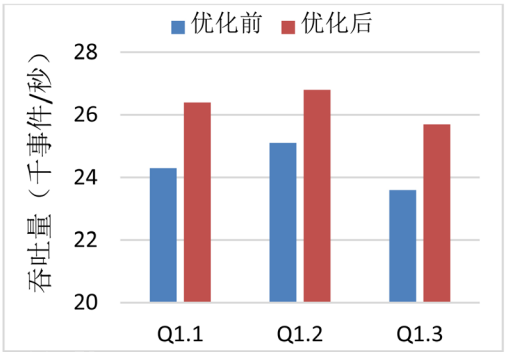


Figure 5. Throughput comparison of before and after decomposition optimization
图 5. 分解优化前后吞吐量对比图

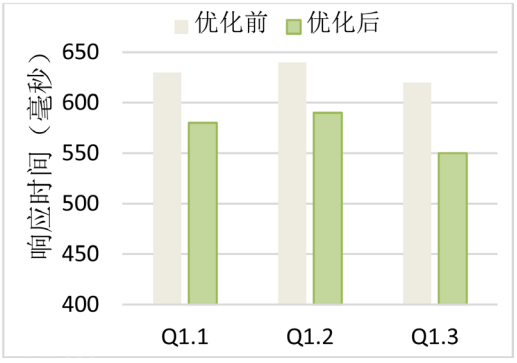


Figure 6. Response time comparison of before and after decomposition optimization
图 6. 分解优化前后响应时间对比图

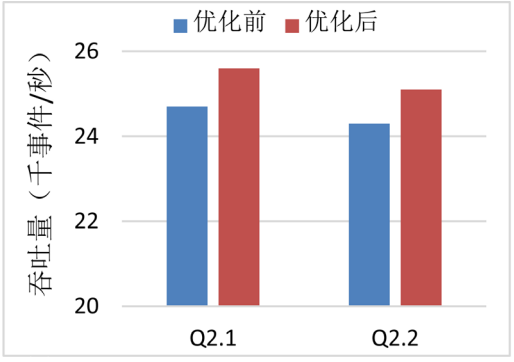


Figure 7. Throughput comparison of before and after pattern merged optimization
图 7. 模式合并优化前后吞吐量对比图

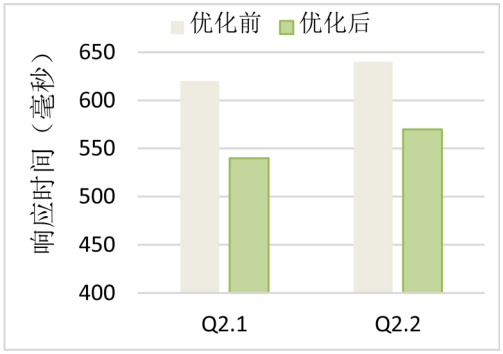


Figure 8. Response time comparison of before and after pattern merged optimization
图 8. 模式合并优化前后响应时间对比图

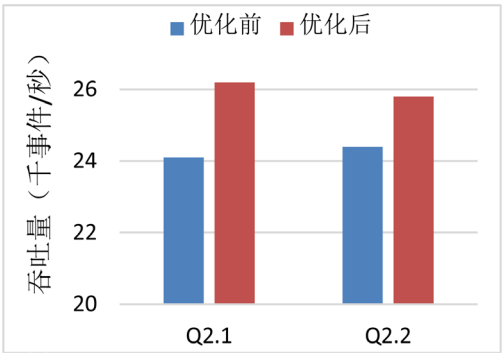


Figure 9. Throughput comparison of before and after common pattern extracted optimization
图 9. 公共模式提取前后吞吐量对比图

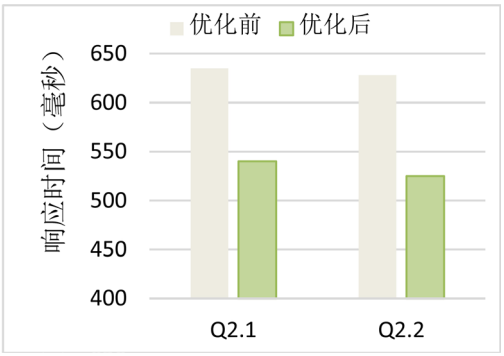


Figure 10. Response time comparison of before and after common pattern extracted optimization
图 10. 公共模式提取前后响应时间对比图

5. 结语

事件源的匹配方法以及查询的执行策略一定程度上决定了复杂事件处理的效率。本文中提到的查询分解方法可以较好的将来自不同事件源的数据进行合理的拆分，充分利用集群的并行计算能力；并且将对事件源的检测与实际查询相结合，极大程度上减少了系统中的通信开销。实验结果表明，通过以上优化方法可以提高系统单位时间的吞吐量，缩短系统的响应时间。

参考文献

[1] Cugola, G. and Margara, A. (2012) Processing Flows of Information: From Data Stream to Complex Event Processing. *ACM Computing Surveys* (*CSUR*), **44**, 15. https://doi.org/10.1145/2187671.2187677

[2] Luckham, D.C. (2002) The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Longman Publishing Co., Inc., United States.

[3] Cugola, G. and Margara, A. (2012) Complex Event Processing with t-Rex. *Journal of Systems and Software*, **85**, 1709-1728. https://doi.org/10.1016/j.jss.2012.03.056

[4] Cugola, G. and Margara, A. (2009) RACED: An Adaptive Middleware for Complex Event Detection. Proceedings of the 8th International Workshop on Adaptive and Reflective Middleware, Urbana Champaign, 1 December 2009, Article No. 5.

[5] Gyllstrom, D., Wu, E., Chae, H.J., *et al.* (2006) SASE: Complex Event Processing over Streams. http://cidrdb.org/cidr2007/slides/p46-yanlei-diao.pdf

[6] 王亦雄, 廖湖声, 孔祥翾, 等. CEStream: 一种复杂事件流处理语言[J]. 计算机科学, 2017, 44(4): 140-143.

[7] Wu, E., Diao, Y. and Rizvi, S. (2006) High-Performance Complex Event Processing over Streams. *Proceedings of the* 2006 *ACM SIGMOD International Conference on Management of Data*, Chicago, 27-29 June 2006, 407-418. https://doi.org/10.1145/1142473.1142520

[8] Agrawal, J., Diao, Y., Gyllstrom, D., *et al.* (2008) Efficient Pattern Matching over Event Streams. *Proceedings of the* 2008 *ACM SIGMOD International Conference on Management of Data*, Vancouver, Canada, 9-12 June 2008, 147-160. https://doi.org/10.1145/1376616.1376634

[9] 孟由, 栾钟治, 谢明, 钱德沛. 一种基于算子的可扩展复杂事件处理模型[J]. 软件学报, 2014, 25(11): 2715-2730.

[10] 路瑶. 一种基于正规树模式匹配的复杂事件检测方法[D]: [硕士学位论文]. 北京: 北京工业大学, 2016.