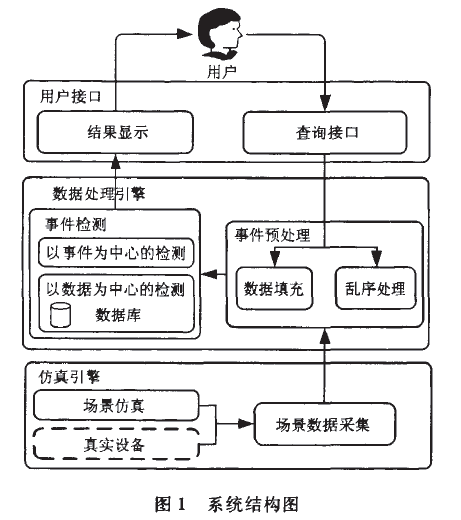
1. **国内研究方法**

谷峪, 胡小龙, 李传文,等. REvent:一种面向RFID应用的复杂事件处理系统[J]. 计算机研究与发展, 2010, 47(z1).

**guyu@ise．neu．edu．cn**

REvent的系统框架图

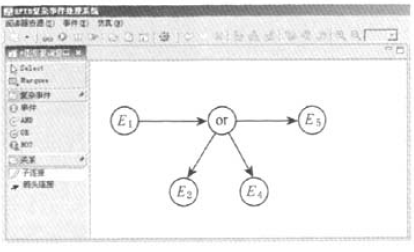


涉及到的关键技术：

数据预处理技术

事件检测技术

评语：从系统界面上能看出他们是用的java开发的一个小程序，从上面的系统结构图我们也可以看出他们的处理系统主要还是以人为中心，智能化处理程度还是偏低。



陈皓, 李瑜, 虎嵩林,等. 基于S4框架的并行复杂事件处理系统[J]. 通信学报, 2012(s1):165-169.

关键技术：

基于操作符的负载分流优化方法

当面对高速数据流时，如果能将负载均匀地分

流到集群的各个节点，并行地处理事件流，则可以

增加系统的吞吐量。然而复杂事件处理的特殊性在

于事件往往不是孤立地存在，在处理中需要根据事

件流上下文或其他事件流进行决策，单纯地拆分数

据流到不同节点，会影响事件间的关系，导致处理

出错。

S4（simple scalable streaming system）是一个去中心的、分布式的、可扩展的流式处理系统。类似于Storm 的消息分组策略，S4 框架通过对流事件的关键字进行散列计算，得到处理事件的节点编号，再将事件发送到对应节点进行处理。通过修改S4框架的事件分流机制，基于操作符的不同，使用不同的分流策略，实现了并行的通用CEP 系统。

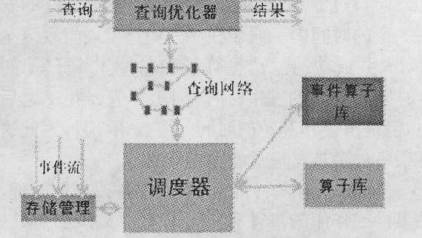
魏永超, 陈立军. 数据流上复杂事件处理系统Eagle的设计与实现[C]// 中国数据库学术会议. 2008.

涉及到的关键技术：

事件代数

事件操作符

聚集操作符  
系统架构：



实验环境：Inter酷睿E4400双核 2.00GHz Windows XP

根据文章资料可以推断数据库是用的Oracle，主要使用java语言开发的

1. **国外研究方法**

Brenna L, Demers A, Gehrke J, et al. Cayuga: a high-performance event processing engine[C]// Proceedings of the 2007 ACM SIGMOD international conference on Management of data. ACM, 2007:1100-1102.

不断增加的数据量作为高速事件流到达。库存报价数据，网络流量数据，在传感器网络中收集的数据，RFID流只是这种流的几个示例。此外，随着网络上的动态内容（博客，新闻，拍卖）的数量迅速增长，网络正在成为许多有趣和复杂的事件流的来源。诸如医疗保健，环境监测，供应链管理和合规性检查等新兴企业级应用以及个性化新闻馈送和信息聚合等消费者应用都需要过滤，关联和聚合各个事件。所得到的复杂事件查询超出了发布/订阅，因为查询涉及多个事件之间的相关，而不是对单个事件的简单谓词。在康奈尔，我们建立了Cayuga系统，一个用于复杂事件处理的高性能系统[1,2]。 Cayuga将用于组成状态查询的简单查询语言与基于具有缓冲区的非确定性有状态自动机的可扩展查询处理引擎组合。 Cayuga的一个重要特征是它不仅可以扩展流中事件的到达速率，而且还可以扩展查询数量;可扩展性的后一维对于消费者应用尤其重要，其中数百万用户可以具有向系统注册的查询。在本演示中，我们将展示Cayuga的应用程序，用于监控来自Web Feed聚合器的事件流，用于个性化新闻聚合和通知。我们首先给出Cayuga系统的高级概述，介绍一些我们将在演示中显示的Cayuga查询（第2节）。然后我们详细描述实际演示（第3节）。

在Cayuga中，每个事件流都有一个固定的关系模式，并且流中的事件被视为关系元组。

每个事件具有两个时间戳，开始时间和检测时间，对事件可以具有非零但有限的持续时间的事实进行建模。 事件按其检测时间的顺序串行化; 具有相同检测时间的事件被认为同时发生，并且不能保证它们的序列化顺序。 几个有趣的理论问题导致了这种设计。

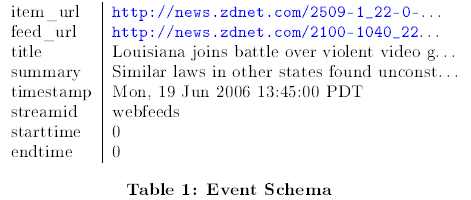
Cayuga查询语言是从事件代数派生的[1]; 它是代数运算符的简单映射一个SQL语法，在精神上类似于复杂事件SASE [4]中的语言。 每个查询具有以下形式：

SELECT <> attributes

FROM <> algebra\_expression

PUBLISH <> output\_stream

SELECT子句指定输出流模式中的属性，FROM子句指定Cayuga事件模式，PUBLISH子句给出输出流的名称。 事件模式可以用三个不同的运算符构建。 FILTER {？}运算符从满足谓词？的输入流中选择那些事件。



演示查询1：在演示中，我们将通过以下查询来说明FILTER运算符，其中包含由整个流中的Google新闻发布的所有新闻项目：webfeeds？ 具有表1中所示的模式。

SELECT \* FROM

FILTER {feed\_url ='http：//news.google.com/'}（webfeeds）

PUBLISH google\_news\_items

一个强大的结构允许我们随时间关联事件是排序运算符NEXT {？}。 当应用于两个输入流S1和S2作为S1 NEXT {α} S2时，运算符将来自S1的每个事件与S2中满足谓词α的下一个事件组合。 并且发生在S1中的事件的检测时间之后。

演示查询2.此查询包含某些网站发布的所有新闻项目，其后是一篇关于Google新闻的文章。 请注意，此查询将Query 1的输出作为其输入流之一。 用户定义函数包含执行子字符串匹配; $ 1和$ 2指的是NEXT运算符的两个输入流。

SELECT $ 2.summary，$ 1.item\_url

FROM（webfeeds）

NEXT {contains（$ 2.item\_url，$ 1.item\_url）= 1}（google\_news\_items）PUBLISH re？ed\_by\_google\_news

虽然NEXT允许我们关联两个事件，但是在许多情况下，我们需要遍历已知数量的事件，直到满足停止条件。 此功能由FOLD操作员提供。 直观地，FOLD是NEXT运算符的泛化，因为它查找包括两个或更多事件的模式。 我们将使用下一个例子来描述FOLD。

演示查询3.此查询发送通知

每当一个iPod在新闻中受欢迎（即，有

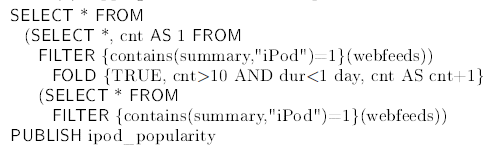
至少十篇文章在一段时间谈论iPod

持续时间）。 FOLD运算符包含三个表达式：

（1）迭代的条件

（2）迭代的停止条件

（3）迭代步骤之间的映射。



Cayuga在时代处理事件;在历元期间，处理具有相同检测时间的所有事件。查询被实现为具有如下工作的缓冲器和自循环的非确定有限状态自动机。自动机中的每个状态都分配有固定的关系模式。每个边缘，例如在状态P和Q之间，由三元组hS，α，fi标记，其中S标识输入流; ？是模式（P）×模式（S）的谓词;和f，模式映射是将模式（P）×模式（S）转换为模式（Q）的函数。 NFA操作如下。假设NFA实例处于具有存储数据x的状态P（注意x符合模式（P））。让事件e在流S上到达，使得满足α（x，e）。然后，机器非确定地转变到状态Q，并且存储的数据变为f（x，e）。 Cayuga支持重新订阅，一个类似于关系数据库系统中的查询计划的概念：来自一个查询的输出事件流可以用作一个或多个其他查询的输入流。重新订阅启用非常复杂的事件模式查询，并且它显着扩展了查询语言的表达性。该系统的更多细节可以在我们最近的出版物中找到[1,2]。

3演示概要

在本节中，我们提供了如何演示上面显示的查询的概述，以Demo Query 2为运行示例。 我们的演示包括以下三个组件：（i）基于Web的前端，供用户输入查询; （ii）轨迹可视化器，其显示Cayuga引擎的内部状态和基于自动机的查询表示; 和（iii）显示查询评价结果的输出可视化器。 在演示期间，我们将对来自Web Feed聚合器的事件流运行查询。 让我们现在通过查询2的执行来呈现上述每个组件。该查询是针对包含来自于2006年6月和10月之间记录的418个频道的225365个源项目的事件流来处理的。

4 提交Cayuga查询

基于Web的前端正在自定义的Python Web服务器上运行，基于AJAX的控件用于异步通信和浏览器中的用户友好界面。 Cayuga有一个基于Web的前端，用户可以输入持久性查询并使用正在运行的Cayuga引擎注册它们。为了方便起见，用户可以从下拉菜单中为所述查询语言选择预定义的模板。然后，他们可以根据自己的需要修改它们，或从头开始写一个。由于Cayuga查询可以轻松跨越多行，因此我们提供的多行输入框比通常在基于Web的查询界面中看到的多。图1中选择进行编辑的查询是显示查询2的查询前端的屏幕截图。请注意，此版本的查询是内联的，没有重新订阅。在提交查询时，查询将被编译并且在Cayuga之外。用户将收到一条带有查询ID或错误消息的消息，查询未编译，并且必须在更改后重新提交。用户可以选择批处理多个查询，然后再按EXECUTE向Cayuga注册查询。可以通过按清除来清除批次。

跟踪Visualizer

Cayuga可以输出其内部状态在事件之间如何变化的连续跟踪。此跟踪将写入文件，其中包含查询自动机的描述，后跟事件和状态实例。我们的可视化器读取跟踪文件，并使用基于Java Swing的GUI来显示事件如何匹配查询自动机中的谓词。

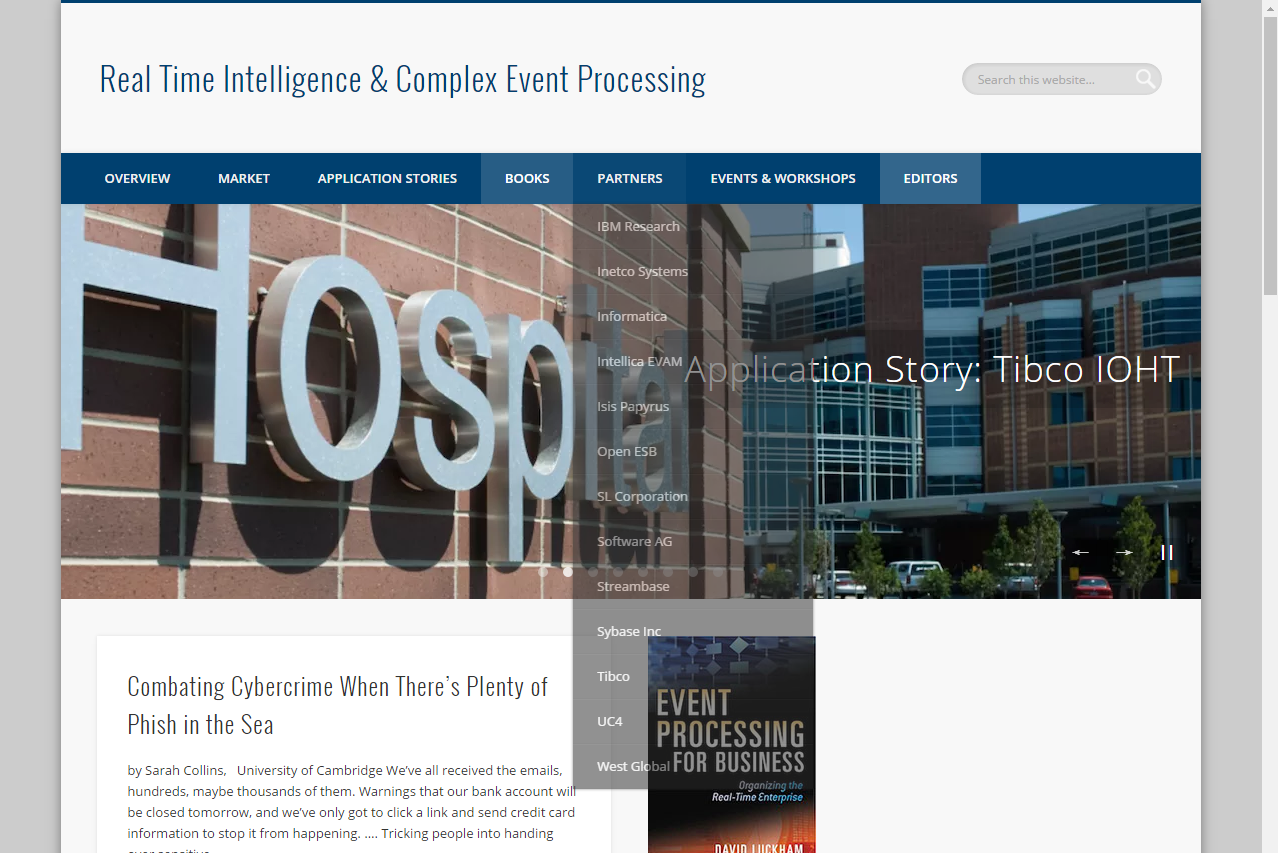
为了精确地查看加载的查询如何对输入流中的事件起作用，用户可以选择向前播放，在任何给定点暂停，或者逐步进行跟踪。这允许用户详细调查任何单个事件如何影响查询自动机的内部状态。在图2和图3中的可视化器的工具栏上看到的控件是（从左到右的顺序），用于放大和缩小自动机，调整与工具栏的垂直距离，以及用于回放。输入和输出流在动画中被命名和给定不同的颜色。在任何时候，如果用户点击自动机中的一个状态，将弹出一个文本框，以显示处于此状态的实例。单击边缘将显示该边缘上的谓词。图2示出了具有用其谓词注释的过渡边缘的查询2的自动机。在图3中，自动机已经处理了一些事件。流到目前为止产生了状态1的7个实例，状态2的4个实例。右边的文本框显示状态2的内容，向下滚动以查看最后一个实例达到这个状态。从文字中可以看出，http://news.zdnet.com在3月12日（星期三）06:00发布的故事是由http://news.google.com在当天的13:00链接到的。

见证输出

当生成见证事件（即查询结果）时，Cayuga当前将其转储到Web前端正在拖尾的文件中。 然后将这些文件放入一个BUYER中，其中频繁轮询AJAX函数可以获得它们并且在终端用户的web浏览器的HTML页面中显示，并按查询排序。 图4是屏幕截图，显示了查询2出现的第一个两个见证。

结论

Cayuga是一个成熟的事件处理系统，有趣处理模型。 我们相信它的功能演示将对数据库社区感兴趣。



读文献的时候看到这个网站<http://www.complexevents.com/>，这个应该是一个比较成熟的东西了

# An Overview of Event Processing Software

A large variety of commercial and open source event processing software is available to architects and developers who are building event processing applications. These are sometimes called event processing platforms, complex-event processing (CEP) systems, event stream processing (ESP) systems, or distributed stream computing platforms (DSCPs).

Some examples are:

1. Apache Samza
2. Apache Spark
3. Apache Storm
4. Codehaus/EsperTech’s Esper, Nesper
5. DataTorrent RTS (Real-time Streaming)
6. FeedZai Pulse
7. FujitsuInterstage Big Data Complex Event Processing Server
8. Hitachi uCosminexus Stream Data Platform
9. IBM InfoSphere Streams
10. IBM Operational Decision Manager (ODM)
11. Informatica RulePoint
12. LG CNS’ EventPro
13. Microsoft StreamInsight
14. OneMarketData OneTick CEP
15. Oracle Event Processor
16. RedHat Drools Fusion/JBoss Enterprise BRMS
17. SAP Event Stream Processor
18. SAS DataFlux
19. ScaleOut Software
20. SQLStream s-Server
21. Software AG Apama Event Processing Platform
22. Tibco BusinessEvents
23. Tibco StreamBase
24. Vitria Technology Operational Intelligence Analytic Server
25. WS02 CEP Server

These subsystems offer a wide range of different features, but all perform CEP in a technical sense. CEP (or ESP) is a technique in which incoming data about what is happening (event data) is processed more or less as it arrives to generate higher-level, more-useful, summary information (complex events). Event processing platforms have built-in capabilities for filtering incoming data, storing windows of event data, computing aggregates and detecting patterns. In formal terminology, CEP software is any computer program that can generate, read, discard or perform calculations on complex events. A complex event is an abstraction of one or more base (input) events. Complex events may signify threats or opportunities that require a response from the business. One complex event may be the result of calculations performed on a few or on millions of base events from one or more event sources.

种类繁多的商业和开源事件处理软件可供建筑师和开发商谁正在构建事件处理应用。这些有时被称为事件处理平台，复杂事件处理（CEP）系统，事件流处理（ESP）系统，或分布式流计算平台（的DSCP）。

一些例子是：

1. 阿帕奇Samza
2. 阿帕奇星火
3. 阿帕奇风暴
4. Codehaus / EsperTech的蛾，Nesper
5. DataTorrent RTS（实时流）
6. FeedZai脉冲
7. FujitsuInterstage大数据复杂事件处理服务器
8. 日立uCosminexus流数据平台
9. IBM InfoSphere Streams的
10. IBM经营决策管理器（ODM）
11. Informatica的RulePoint
12. LG CNS'EventPro
13. 微软的StreamInsight
14. OneMarketData OneTick CEP
15. Oracle事件处理器
16. RedHat的Drools的融合/ JBoss企业BRMS
17. SAP事件流处理器
18. SAS DataFlux的
19. ScaleOut软件
20. 的SQLstream S-服务器
21. Software AG公司Apama事件处理平台
22. TIBCO BusinessEvents
23. TIBCO StreamBase
24. Vitria技术运营智能分析服务器
25. WS02 CEP服务器

这些子系统提供范围广泛的不同特点，但都在一个技术意义上进行CEP。CEP（或ESP）是其中关于正在发生的事情（事件数据）的输入数据被处理或多或少即立即产生更高级别的，更是有用的，摘要信息（复杂事件）的技术。事件处理平台具有内置的功能，用于滤波输入数据，存储事件数据的窗口，计算聚集体并检测图案。在正式的术语，CEP软件是可以产生，阅读，丢弃或复杂事件进行计算的计算机程序。复杂事件是一个或多个基站（输入）事件的抽象。复杂事件可能意味着需要从企业的响应威胁或机遇。一个复杂的事件可以是在一些或上百万碱事件从一个或多个事件源执行的计算的结果。

事件处理平台是丰盛的很多：

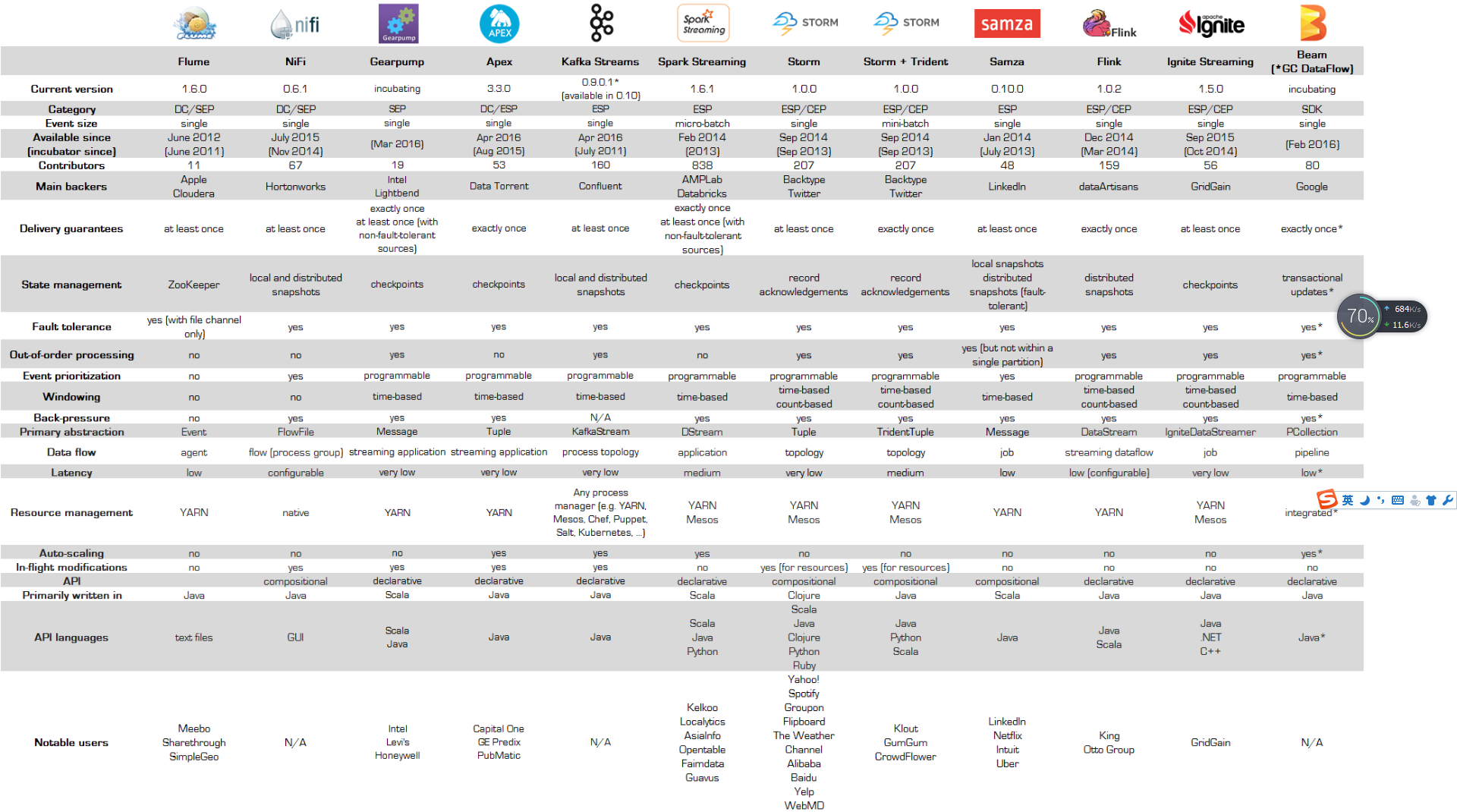
* 建立，被广泛使用的商用产品，如IBM的ODM和InfoSphere Streams的，SAP的ESP（以前的Sybase埃勒瑞），Software AG的Apama，TIBCO的业务活动和StreamBase，和其他人，是多方面的开发和运行软件套件，包括适配器与整合事件源，开发和测试工具，仪表板和报警工具和管理工具。
* 其他事件处理平台相结合，与功能，如查询，报表，互动分析，警报或​​关键绩效指标的工具，这是在运行智能应用专门针对。例子包括FeedZai脉冲，的SQLstream S-Server和Vitria Technology的作战情报。
* 新兴的DSCP，如Apache Samza，Spark和风暴，是没有全面的本地CEP分析功能和相关配件的通用平台。他们是高度可扩展的和可扩展的，因此开发人员可以添加逻辑来处理多种流处理应用程序。
* 这些所描述的DSCP，与蛾/ Nesper和RedHat的Drools的融合/ JBoss企业BRMS（两者均为CEP系统）一起，是开源产品。
* DataTorrent的RTS和IBM的InfoSphere流被划分为的DSCP或CEP系统，根据上下文和作者的观点。
* 有些事件处理平台捆绑在一起的规则引擎。例子包括IBM的ODM，RedHat的Drools的融合/ JBoss企业BRMS和TIBCO业务活动。

CEP使用量迅速增长，因为CEP，从技术意义上说，是让在实时或近实时事件流信息的唯一途径。该系统具有处理事件数据或多或少到达使得相应的操作可以快速地作出。在这篇文章中列出的子系统是通用开发和运行时工具，用于开发人员构建自定义，事件处理应用程序，而无需重新实现核心算法处理事件流。然而，这些子系统是唯一的几种方式来获得的CEP的功能之一，事实上，该实施CEP逻辑大多数应用程序不使用这些子系统。

最CEP是作为较大产品的一部分获得。公司收购打包应用程序或订阅已在幕后嵌入式CEP一个SaaS服务。公司正在购买这种情况，要求事件处理的溶液中，它可能没有意识到正在使用的CEP。例如，供应链可视化产品; 安全性信息，甚至管理（SIEM）产品; 某些类型的欺诈检测和治理，风险和法规遵从（GRC）的产品; 系统和网络监控系统; 业务活动监控（BAM）工具; 和许多其他类别的软件实现CEP逻辑的一些或大或小数额。在少数情况下，这些产品或SaaS产品的开发人员已经利用上面列出，以减少他们必须编写的代码量的通用事件处理平台。但在大多数情况下，开发人员实现的在新的代码事件处理算法的一个专门的子集，以满足他们的应用程序的目的。

一些用户企业都写有CEP逻辑的自定义应用程序，而不是利用一个现成的现成事件处理平台。这是20世纪90年代和21世纪初尤为常见上面列出的子系统广泛使用之前，一些开发商仍然选择写性能或成本的原因自己CEP逻辑。例如，大型银行及相关金融服务公司已经做了前端办公系统的资本市场有自己的嵌入式CEP逻辑交易。

每个公司都有它已经使用了幕后CEP实现一定量的应用程序或SaaS解决方案的地方。越来越多的公司需要的那种这里列出来支持高要求，高吞吐量，低延迟的应用中，事件处理逻辑是主要的业务问题，事件处理子系统。在这两种情况下，你越了解CEP，更好的将您的应用程序体系结构。



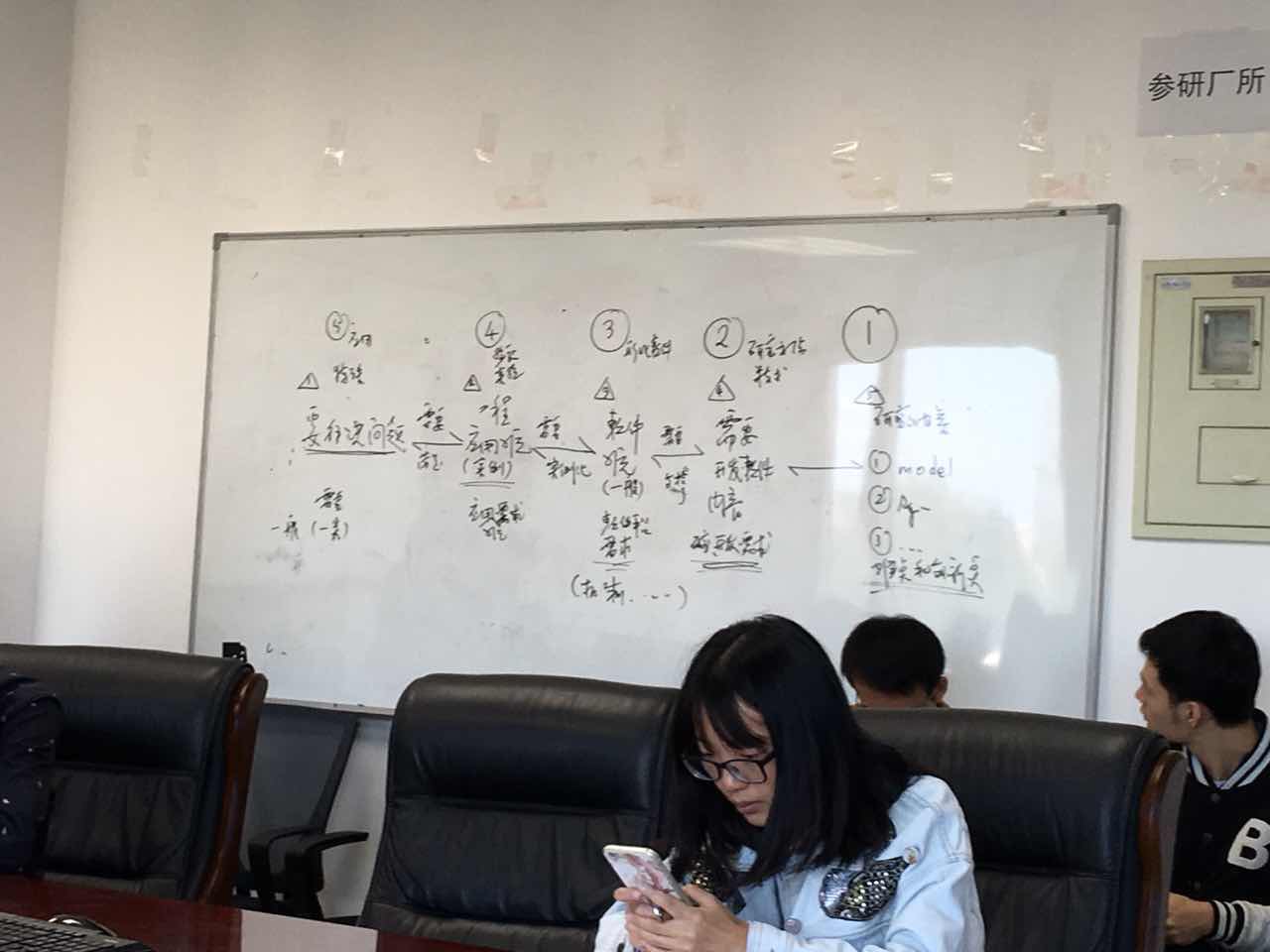
Akdere M, etintemel, U&#, Tatbul N. Plan-based complex event detection across distributed sources[J]. Proceedings of the Vldb Endowment, 2008, 1(1):66-77.

系统架构

我们系统中的主要组件是事件源和基本节点（图1）。源生成事件;例如网络监控应用中的路由器和防火墙以及灾难监控应用中的温度传感器是示例。源具有本地存储，允许它们临时记录感兴趣的事件。可以查询这些日志，并在必要时获取事件。在实践中，一些事件源可能不具有任何本地存储或者是自主的并且在我们的控制之外（例如，web上的RSS源）。在这种情况下，我们依赖代表它们提供这些功能的代理节点。因此，当引用原始事件源或其代理时，我们使用术语源。基站负责生成和执行CED计划。计划执行涉及与事件源的协调，因为事件根据需要从基地传输。因此，我们的系统将数据收集的拉和范例结合起来，以避免纯基于推送的系统的缺点。 CED计划我们努力通过仔细选择哪些来源将传输什么事件来努力减少对基站的网络流量。

**尊敬的杨老师：**

您好，这是我结合10月8号杨老师对我们的指点，还有10月12号杨老师对我提的一点问题进行了总结，下面就是本周汇报的主体内容：



**要解决的问题**

复杂事件的处理：比如发生了某个，或者一连串的有关联的事件，mes系统做出相应的响应

**实例**

简单事件处理实例：某批次货物交货期快到了或者要到了，但是生产车间的反馈回来的生产数量还不够，这时候系统应该对管理者发送警告，同时计算出比较优化的生产计划（需要排产的接口~），保证按期完工。

复杂事件处理实例： 某个加工处，加工人员已经就位，周转箱与该加工处所需物料正在运向该处的途中，设备处于正常状态，则系统应当提示在加工现场的工人准备接收物料

问题核心：复杂事件的处理操作

**工程应用（实例）**

比如我们在某个工厂，如MES系统某些加工工厂应用，对生产中的产生的各种事件进行保存，处理，然后发布对应的动作。

**软件（一般情况）**

不针对某一任何工厂，应当具有通用性，应当具有一般性，并且能根据企业的需求进行快速构建(通用化的事件定义软件)。

**需要开发软件内容**

这个过几天我会以开发的说明书的形式呈现。

**研究的内容（难点、模型、算法）**

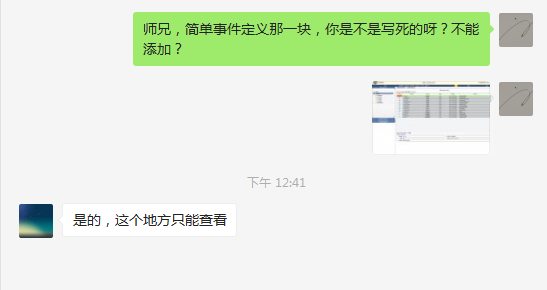
1.难点：

多事件同时发生的调度问题，比如某工厂有1000多个机床，工人加上管理员估算3000人，那么当工厂满负荷运转的时候，产生的数据量是相当庞大的。那么我们在处理的数据的时候应该借用一个高效的处理引擎，如：Drools、OpenRules、OpenLexicon这类。

2.算法？？？？

<这一块目前我还没有想好会用到什么类型的算法>

10月9号我也咨询了一下晨晖师兄，他简单事件那一块确实也没有写出来，根据目前对师兄做的东西的认知，他已经把工厂内发生事件的大类规划好了，即有任务类型事件、设备类型事件、物料类型事件、人员类型事件和周转箱类型事件这五大类，系统可以对这五大类做出修改，但是没有办法多定义一个大类。



下面是我根据10月12号与您的沟通大致整理的：

**事件有哪些：**

目前的事件类型我想还是基本继承师兄的定义，然后在上面扩展。

1. 简单事件

|  |  |  |
| --- | --- | --- |
| 事件类型 | 子事件命名 | 子事件语义 |
| 任务类型事件  （TaskEvent） | TaskDispatch | 任务派工 |
| TaskReady | 任务就绪 |
| TaskStart | 任务开始 |
| TaskCompleteFirst | 完成首件 |
| TaskCancel | 取消任务 |
| TaskComplete | 完成任务 |
| TaskInterrupt | 任务中断 |
| 设备类型事件  （DeviceEvent） | DeviceRun | 设备运行中 |
| DeviceFree | 设备空闲 |
| DeviceFault | 设备故障 |
| DeviceMaintain | 设备维修中 |
| 物料类型事件  （MaterialEvent） | MaterialApply | 物料申请 |
| MaterialSign | 物料签收 |
| MaterialCheck | 物料检验 |
| 人员类型事件  （EmployeeEvent） | EmployeeReady | 人员就绪 |
| EmployeeAbsence | 人员未到位 |
| 周转箱类型事件  （BoxEvent） | BoxAddition | 周转箱进入 |
| BoxCurrent | 周转箱存在 |
| BoxStop | 周转箱停留 |
| BoxLeave | 周转箱离开 |

**2．动作有哪些：**

动作的类型，大概根据师兄的分类，有如下类型：

|  |  |
| --- | --- |
| 动作策略 | 说明 |
| 调度事件同步 | 在现有MES的生产计划调度模块中，预定义有调度系统所关注的事件类型，这些事件会影响对车间任务调度的结果，主要包括设备故障、新增任务、任务完成等事件。调度系统需要实时同步这些事件以获得一个更加符合实际生产情况的调度方案。因此该动作策略就是为了实现调度事件的实时同步，当匹配成功的事件符合调度事件表中预定义的类型时，快速完成同步任务。 |
| 异常信息记录 | 生产过程的进行不可能是一直按照生产计划进行，随着任务的执行会产生许多异常情况，如任务的中断、周转箱滞留等，因此作为管理车间的MES系统需要及时了解到这些异常信息以采取相应的处理方式。因此该策略实现异常事件的实时记录。 |
| 消息通知预警 | 该策略实现事件消息的实时通知和预警，主要配合移动终端使用，当有一些重要事件消息时会主动在终端上推送消息，提醒现场工人，如有新的任务派工、任务进度慢等。 |
| 关键状态更新 | 该策略主要实现生产过程中的关键状态信息的实时更新，如任务状态、设备状态、加工记录等。这样MES其它功能模块可以实时获取这些状态，如调度系统可以实时同步加工任务情况，实现对生产过程的动态优化。 |
| 复杂事件日志 | 该策略主要实现事件处理过程中实时记录检测成功的复杂事件模式实例，以支持MES对这些事件进行统计分析等业务。 |

**如何去驱动：**

采用基于事件—条件—动作（Event—Condition—Action，ECA）规则来实现，一条ECA复杂事件处理规则可以用以下形式表示。

EVENT [complex event pattern]

IF [condition]

DO [action]

其中complex event pattern代表要匹配的复杂事件模式，IF表示要满足的条件，如果二者匹配满足则执行action定义的动作。具体规则的定义需要结合具体的业务逻辑，由ECA规则可知一条复杂事件处理规则的定义主要包括三个部分，即事件、条件和动作。

**资料整理**

Drools规则引擎：

**1.1 现状**

在很多行业应用中比如银行、保险领域，业务规则往往非常复杂，并且规则处于不断更新变化中，而现有很多系统做法基本上都是将业务规则绑定在程序代码中。

**1.2 问题**

主要存在的问题有以下几个方面：

1） 当业务规则变更时，对应的代码也得跟着更改，每次即使是小的变更都需要经历开发、测试验证上线等过程，变更成本比较大。

2） 长时间系统变得越来越难以维护。

3） 开发团队一般是由一个熟悉业务的BA（业务分析人员）和若干个熟悉技术的开发人员组成，开发人员对业务规则的把握能力远不及BA，但实际上却承担了将业务规则准确无误实现的重任。

4） 系统僵化，新需求插入困难。

5） 新需求上线周期较长。

**1.3 解决方案**

能否让我们的业务系统更灵活一点呢？

思路：将业务规则从技术实现中提取出来，实现技术和业务分离，开发人员处理 技术、业务分析人员定义业务规则，各自做自己所擅长的事情。

方案：目前已经有比较成熟的开源产品支持，这就是本文所要介绍的Drools，我们将业务规则定义在Database或者BRMS(Business Rule Management System)中，通过管理DB或者BRMS实现业务逻辑的动态改变。

**1.4 适用情景**

什么时候应该使用规则引擎？

虽然规则引擎能解决我们的许多问题，但我们还需要认真考虑一下规则引擎对我

们的项目本身是否是合适的。需要关注的点有：

Ø 我的应用程序有多复杂?

对于那些只是把数据从[**数据库**](http://lib.csdn.net/base/mysql)中传入传出，并不做更多事情的应用程序，最好不要使用规则引擎。但是，当在[**Java**](http://lib.csdn.net/base/javaee)中有一定量的商业逻辑处理的话，可以考虑Drools的使用。这是因为很多应用随着时间的推移越来越复杂，而Drools可以让你更轻松应对这一切。

Ø 我的应用的生命周期有多久？

如果我们应用的生命周期很短，也没有必要使用Drools，使用规则引擎将会在中长期得到好处。

Ø 我的应用需要改变吗？

这个答案一般情况下是肯定的，“这世界唯一不变的只有变化”，我们需求也是这样的，无论是在开发过程中或是在开发完成以后，Drools能从频繁变化的需求中获得好处。

**2 什么是规则引擎**

规则引擎是基于规则的专家系统的核心部分，主要由三部分组成：规则库(Knowledge base)+Working Memory(Fact base)+推理机(规则引擎)，规则引擎根据既定事实和知识库按照一定的[**算法**](http://lib.csdn.net/base/datastructure)执行推理逻辑得到正确的结果。

**3 Drools简介**

Drools 是一个基于Charles Forgy's的RETE算法的，易于访问企业策略、易于调整以及易于管理的开源业务规则引擎，符合业内标准，速度快、效率高。

业务分析师人员或审核人员可以利用它轻松查看业务规则，从而检验是否已编码的规则执行了所需的业务规则。

**4 竞争产品比较**

与Drools功能类似的同类开源产品主要有：OpenRules、OpenLexicon等，商业产品功能比较强也比较贵，这里不做比较，主要差别如下表：

|  |  |  |  |
| --- | --- | --- | --- |
|  | Drools | OpenRules | OpenLexicon |

|  |  |  |  |
| --- | --- | --- | --- |
| 规则表示方法 | 支持以下四种：  The Rule Language  Domain Specific Language Decision Tables  XML Rule Language | Decision Table | 无规则语言，通过Web界面配置规则，并保存在数据库中 |
| **规则算法** | RETE算法 | 不详 | 不祥 |
| **规则开发** | Rule IDE(Eclipse插件)  Excel  XML | Excel | 基于Web的配置界面 |
| 规则知识库 | 支持XML、Excel、BRMS | Excel | Database |
| **易用性** | 通过Rule IDE，可以方便地编辑DRL和DSL文件 | 业务人员直接使用Decision Table估计还是有一定的难度 | 不太好用 |
| **开放性** | 非常开放 | 不开放 | 不开放 |
| **可拓展性** | 通过与J2EE技术结合，具有很强的拓展性。 | 不太好 | 很不好 |
| **JSR-94标准** | 支持 | 支持 | 不支持 |
| **成熟度** | 高 | 较低 | 比较新，不成熟 |
| **厂商背景** | JBoss | 小厂商 | 小厂商 |

此致

敬礼

文波

2016/10/17

尊敬的杨老师：

您好，这是我本周的学习情况和生活情况，希望杨老师能够给予宝贵的意见。

## 生活情况

本周还是延续着比较好的学习状态，每天基本延续着早起的习惯（ps，除了今天9点才起来，大概是昨晚睡觉中途被冷醒了的缘故）。

## 学习情况——文献的阅读

首先对上次汇报杨老师问为什么学逆向工程那一点进行简答的解释：逆向建模生成javabean的方法是在看晨晖师兄的那一块代码中，关于和数据持久层的建立利用Hibernate来操作看到的，此方面能较快的从数据库生成代码，对开发起到一定的帮助（ps晨晖师兄是正着来的，先建javabean，再到数据库的映射，可以直接省去建javabean这一步，直接由数据库生成javabean）。

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, 1(3).这篇文章分析了RFID数据的特点，讨论了RFID数据清洗方法、以数据为中心的检测技术、以事件为中心的检测技术、以及复杂事件处理系统（典型的有美国加州大学伯克利分校的的原型系统SASE、Cornell大学开发的Cayuga系统、Texas大学Arlington分校开发的EStream系统，以及Dartmouth学院开发的PQS系统。Ps：事件处理机制我还没有详细的看）

曹科宁, 王永恒, 李仁发,等. 面向物联网的分布式上下文敏感复杂事件处理方法[J]. 计算机研究与发展, 2013, 50(6):1163-1176.这篇文章提出一种高效的面向物联网的分布式上下文敏感复杂事件处理架构和方法。该方法使用模糊本体进行上下文建模（好像乔老师那边的学生经常提到本体，我查了一查本体的相关概念，在后文中以专业词汇的方式出现），以支持事件的不确定性及模糊事件查询问题。文中指出，物联网产生的数据具有大数据的特征，同时也指出于传统的Hadoop/MapReduce的框架不能完全适用于处理物联网实时数据（但是我这里想的，我们不是需要做一些智能的提醒业务，提醒业务就可以基于过去的数据进行推理和判断，就像网易云音乐这类的目前运用机器学习来分析用户偏好类似，我们也可以分析工人偏好，或者加工偏好这类，来确定推送，推送可以由pad+手环结合的方式）ps文中的上下文敏感复杂事件的处理方法暂时略读了一遍，暂时还没有看明白。下面摘录了文章中有关一些方法的综述性内容，希望对将来相关方法的研究能提供一些参考性意见：

## 上下文敏感事件处理

上下文模型在上下文系统的应用和开发中具有重要作用，在上下文的相关研究中提出了各种上下文表示模型，包括键一值模型、面向对象的模型和基于本体的模型．一般认为本体是事件上下文表示的最佳模型，但传统的本体无法处理不确定的知识这限制了其在不确定事件处理中的应用，因此近年出现了一些模糊本体模型和推理方面的研究：Almeida等人提出了一种面向智能环境的基于模糊逻辑的模糊模型；Singh等人提出了一种模糊集成本体模型(FOIM)，试图把模糊逻辑集成到本体设计结构中；Zhang等人提出了面向分布式模糊推理分解的分布式模糊推理Petri网模型；Cai等人研究了模糊本体中的隶属性和典型性问题，并提出了一种面向推荐系统的新型模糊本体模型．上下文感知在异构环境中具有重要作用．

上下文敏感系统面临的挑战主要是实时地对于用户上下文作出合适的决策．用智能的方式处理上下文数据称为上下文推理．Lee等人提出了基于相似性的上下文推理，定义了上下文模型之间的相似性。．该方法不需要初始的上下文信息，从而减少了推理过程的复杂性．除此之外，该方法使用基于相似性分布式推理的Mahalanobis间隔加入了上下文模型的关联，寻找一种最佳的上下文模型，并且减少两个上下文模型之间的模糊来增加推理的可信度．

近年来也出现了一些上下文敏感事件处理的研究工作：Helmer等人描述了基于上下文的事件处理框架，并总结了目前支持常用的事件处理系统的上下文；Ashish等人提出了一种基于本体的上下文敏感复杂事件处理方法；Zhou等人提出了一种非精确语义复杂事件处理框架，在复杂事件处理引擎中加入语义知识，实现非确定复杂事件模式检测，在Teymourian的研究中，加入了本体和声明性规则，优化了智能事件处理机．但目前的文章大多讨论上下文敏感复杂事件处理的思路和框架，而缺少把上下文引入复杂事件处理算法的细节．

## 复杂事件处理

复杂事件处理即检测不断产生的数据流，根据每个事件发生的集合／序列识别复杂事件，然后对检测到的状态作出回应．复杂事件处理主要有4步：

1)从大量数据中获取原始事件；

2)根据具体规则检测关联性事件或聚合事件，用事件操作符创建有意义事件；

3)处理原始事件或复杂事件，获取这些事件的时间、因果关系、层次关系及其他语义关系；

4)为了确保事件信息发送到用户，向应用层作出回应．

在主动数据库及RFID的研究中，复杂事件处理大都采用固定的数据结构．“等人采用了基于树结构的复杂事件处理方法，并通过事件分组的方法对算法进行优化．Wang等人使用有向图在RFID事件流中进行复杂事件处理．Jin等人采用定时Petri网(timed Petri-net，TPN)在RFID数据流中检测复杂事件．SASE是一种基于查询规划的高性能复杂事件处理方法，SEQ事件检测作为SASE的主要部分，采用有限状态机(NFA)和活动实例栈(AIS)．近年来有一些针对SASE的改进算法，Agrawal等人提出了改进的有限状态机机(NFA6)模型来提供更丰富的查询能力，如使用skip—till—next—match和skip—till—any—match等查询指令，Zhang等人对SASE的模型进行了改进来支持非确定时标．传统的基于集中式的复杂事件处理结构当客户增多时，这种体系结构需要更多的带宽和更强的计算能力，同时由于网络拥塞和单点失效等问题，这种结构也很难达到良好的稳定性和可伸缩性，而当前很多物联网应用本身就是分布式的，必须采取分布式的事件处理方法．Akdere等人提出了面向分布式资源的CEP方法，Ku等人提出了一种面向RFID应用的分布式CEP方法．

下面几篇英文的文章，是这几篇文章的参考文献，我还没有看，主要是讲RFID和复杂事件的，我是否应该多花些时间去研读这些文章，还是接着晨晖学长做的东西先试着做一做复杂事件的处理？希望杨老师能给予一点指导性的意见。

A Novel Distributed Complex Event Processing for RFID Application

Accelerating sequence event detection through condensed compositionpdf

Bridging Physical and Virtual WorldsComplex Event

Plan based complex event detection across distributed sources

此致

敬礼

文波

2016/9/18

## 概念性词汇的学习

本体论（ontology）是哲学概念，它是研究存在的本质的哲学问题。近几十年里，这个词被应用到计算机界，并在人工智能、计算机语言以及数据库理论中起到越来越重要的作用。

然而，到目前为止，对于本体论，还没有统一的定义和固定的应用领域。斯坦福大学的Gruber给出的定义得到了许多同行的认可，即本体论是对概念化的精确描述（Gruber，1995），本体论用于描述事物的本质。

在实现上，本体论是概念化的详细说明，一个ontology往往就是一个正式的词汇表，其核心作用就在于定义某一领域或领域内的专业词汇以及他们之间的关系。这一系列的基本概念如同工程一座大厦的基石，为交流各方提供了一个统一的认识。在这一系列概念的支持下，知识的搜索、积累和共享的效率将大大提高，真正意义上的知识重用和共享也成为可能。就此意义而言，Web语言XML（Extensible Markup Language，可扩展标识语言）就是本体理论的一项典型应用；.xml文档就是一个标签化的词汇表。

本体论可以分为四种类型：领域、通用、应用和表示。领域本体包含着特定类型领域（如电子、机械、医药、教学）等的相关知识，或者是某个学科、某门课程中的相关知识；通用本体则覆盖了若干个领域，通常也称为核心本体；应用本体包含特定领域建模所需的全部知识；表示本体不只局限于某个特定的领域，还提供了用于描述事物的实体，如“框架本体”，其中定义了框架、槽的概念。

可见，本体论的建立具有一定的层次性，在教学领域而言，如果说某门课程中的概念、术语及其关系看成是特定的应用本体，那么所有课程中的共同的概念和特征则具有一定的通用性。

Ontology 这个哲学范畴，被人工智能界赋予了新的定义，从而被引入信息科学中。然而信息科学界对 Ontology 的理解也是逐步发展才走向成熟的。1991 年 Neches 等人最早给出 Ontology 在信息科学中的定义：“给出构成相关领域词汇的基本术语和关系，以及利用这些术语和关系构成的规定这些词汇外延规则的定义。”后来在信息系统、知识系统等领域，随着越来越多的人研究 Ontology，产生了不同的定义。1993 年 Gruber 定义 Ontology 为“概念模型的明确的规范说明”。1997 年 Borst 进一步完善为“共享概念模型的形式化规范说明”。Studer 等人对上述两个定义进行了深入研究，认为 Ontology 是共享概念模型的明确的形式化规范说明，这也是目前对 Ontology 概念的统一看法。

Studer 等人的 Ontology 定义包含四层含义：概念模型（Conceptualization）、明确（Explicit）、形式化（Formal）和共享（Share）。“概念模型”是指通过抽象出客观世界中一些现象（Phenomenon）的相关概念而得到的模型，其表示的含义独立于具体的环境状态；“明确”是指所使用的概念及使用这些概念的约束都有明确的定义；“形式化”是指 Ontology 是计算机可读的，也就是计算机可处理的；“共享”是指 Ontology 中体现的是共同认可的知识，反映的是相关领域中公认的概念集，它所针对的是团体而非个体。Ontology 的目标是捕获相关领域的知识，提供对该领域知识的共同理解，确定该领域内共同认可的词汇，并从不同层次的形式化模式上给出这些词汇（术语）和词汇之间相互关系的明确定义。

尽管定义有很多不同的方式，但是从内涵上来看，不同研究者对于 Ontology 的认识是统一的，都把它当作是领域（领域的范围可以是特定应用中，也可以是更广的范围。）内部不同主体（人、机器、软件系统等）之间进行交流（对话、互操作、共享等）的一种语义基础，即由 Ontology 提供一种共识。而且Ontology提供的这种共识更主要的是为机器服务，机器并不能像人类一样理解自然语言中表达的语义，目前的计算机也只能把文本看成字符串进行处理。因此，在计算机领域讨论 Ontology，就要讨论如何表达共识，也就是概念的形式化问题。

证据理论是Dempster于1967年首先提出，由他的学生Shafer于1976年进一步发展起来的一种不精确推理理论，也称为Dempster/Shafer 证据理论(D-S证据理论)，属于人工智能范畴，最早应用于[专家系统](http://baike.baidu.com/view/10875.htm)中，具有处理不确定信息的能力。作为一种不确定推理方法，证据理论的主要特点是：满足比[贝叶斯](http://baike.baidu.com/view/77778.htm)概率论更弱的条件；具有直接表达“不确定”和“不知道”的能力。

RFID 是Radio Frequency Identification 的缩写，即射频识别，是一种非接触式的自动识别技术，它通过射频信号自动识别目标对象，可快速地进行物品追踪和数据交换。识别工作无须人工干预，可工作于各种恶劣环境。RFID技术可识别高速运动物体并可同时识别多个标签，操作快捷方便。RFID 技术诞生于第二次世界大战期间，它是传统条码技术的继承者，又称为"电子标签"或“射频标签”。

最基本的RFID系统由三部分组成：

　　标签（Tag）：由耦合元件及芯片组成，每个标签具有唯一的电子编码，附着在物体上标识目标对象；

　　阅读器（Reader）：读取（有时还可以写入）标签信息的设备，可设计为手持式或固定式；

　　天线（Antenna）：在标签和读取器间传递射频信号。

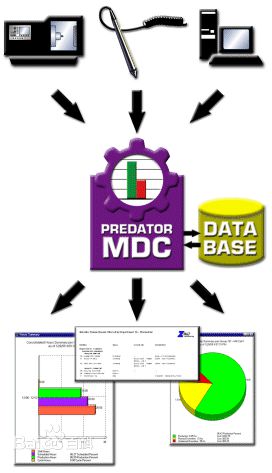
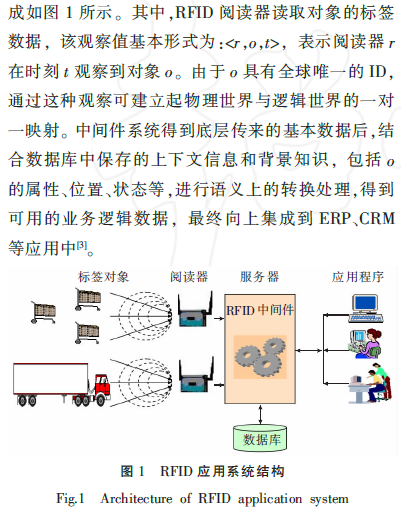
MDC (Manufacturing Data Collection & Status Management)是一套用来实时采集、并报表化和图表化车间的详细制造数据和过程的软硬件解决方案。[1]

MDC 通过多种灵活的方法获取生产现场的实时数据（包括设备、人员和生产任务等），将其存储在Access , SQL 和 Oracle 等数据库，并以精益制造(Lean Manufacturing)管理理念为基础，结合系统自带的近100种专用计算、分析和统计方法，以25,000多种报告和图表直观反映当前或过去某段时间的生产状况，帮助企业生产部门通过反馈信息做出科学和有效的决策。

在上世纪90年代初，盖勒普最早把MDC以精益制造管理理念及解决方案引入中国，基于全球20多年的技术沉淀和国内近14年的本地应用，真正助力中国离散制造企业的数字化制造集成生产管理落地。MDC通过多种灵活的方法获取生产现场的实时数据，结合近100种专用计算、分析和统计方法，直观反映当前或过去某段时间的生产状况，帮助企业生产部门通过反馈信息做出科学和有效的决策。[2]

强大的生产数据采集

从简单的开关机到复杂的模拟量和字符串，制造数据涵盖车间现场需求的各个方面。MDC通过多样化的数据采集手段，让数据的获取拥有最大的可能。MDC通过与数控系统、PLC系统、以及机床电控部分的集成，实现对机床数据采集部分的自动化执行，不需要操作人员的手动操作，这样既保证了数据的实时性，也减少了人工操作产生的失误，保证数据的真实和准确性。[3]

物联网(Internet Of Things)又称传感网，简要讲就是互联网从人向物的延伸。

“物联网”(Internet of Things)指的是将各种信息传感设备，如射频识别装置、红外感应器、全球定位系统、激光扫描器等种种装置与互联网结合起来而形成的一个巨大网络。其目的是让所有的物品都与网络连接在一起，方便识别和管理。

IoE是思科最近提出的，在IoE基础上更加进步的一个概念。我们正从今天的“物联网” (IoT:Internet of Things)走入“万物互联”(IoE:Internet of Everything)的时代，所有的东西将会获得语境感知，增强的处理能力和更好的感应能力。将人和信息加入到互联网中，你将会得到一个集合十亿甚至万亿连接的网络。这些连接创造了前所未有的机会并且赋予沉默的东西声音。

参考文献：

1. 生产数据及设备状态信息采集分析管理系统(MDC) ．盖勒普官网．2011/11/1
2. 生产设备状态采集，加快数字化工厂进程 ．e-works．2014/5/22
3. MDC：数控车间设备监控的“好助手” ．e-works．2014/9/29

