CEStream：一种复杂事件流处理语言

王亦雄[1] 廖湖声[2] 孔祥翾[1] 高红雨[1] 苏 航[1]
(北京工业大学计算机学院 北京 100124)[1] (北京工业大学软件学院 北京 100124)[2]

摘 要 复杂事件处理是支持大数据处理的流式计算平台的核心技术之一。CEStream语言作为一种新型的事件流处理语言，支持分布式环境下的复杂事件处理。该语言以XML等层次化数据为数据模型，为复杂事件检测提供了一种正规树模式匹配功能，并且支持结构连接和正规式匹配。同时，针对分布式的多个事件流，其能够将各个事件源模式匹配的结果按照时间顺序再一次进行正规式模式匹配，满足多源组合型复杂事件的检测需求，具有较强的事件处理能力。为了实现CEStream语言，研制了一个基于流数据处理集群和远端查询代理的执行引擎系统。该系统通过远程查询代理实现基于正规树模式的事件检测，通过流处理集群完成多源组合型复杂事件处理。实验表明，该系统实现了CEStream语言，有效地限制了各个节点之间的通信量，充分利用了集群的计算能力，整体性能能够满足应用需求。

关键词 复杂事件处理，计算机语言，流式计算
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.030

CEStream: A Complex Event Stream Processing Language

WANG Yi-xiong[1] LIAO Hu-sheng[2] KONG Xiang-xuan[1] GAO Hong-yu[1] SU Hang[1]
(College of Computer Science, Beijing University of Technology, Beijing 100124, China)[1]
(School of Software Engineering, Beijing University of Technology, Beijing 100124, China)[2]

Abstract Complex event processing is one of the core technologies of stream processing platform which supports big data processing. A new style event stream processing language, named CEStream, supports complex event processing in distributed environment. This language takes hierarchical data as data model, such as XML, and also provides a feature for complex event detecting, which is named regular tree pattern matching. This feature supports structural connection and regular expression matching. Moreover, aiming at distributed multi-sources event stream, CEStream provides another feature that matches each pattern matching result once again by regular expression which describes time sequence for each result. This feature supports the demand of multi-sources composite complex event detecting, and has good ability of event processing. To implement CEStream language, a process engine system was developed based on stream data processing cluster and remote querying agent. This system provides a feature which uses remote querying agent to detect events by regular tree pattern, and provides another feature which uses stream processing cluster to process multi-sources composite complex events. It's demonstrated by experiments that this system has implemented CEStream language, has effectively made constraint on communication flows between each node in cluster, has capitalized on ability of cluster computing, and it supports the demand of comprehensive performance.

Keywords Complex event processing, Computer language, Stream computing

1 引言

在大数据时代[1,3]，来自实时监控、实时交易、社交网络等各种实时系统的不断增长的海量数据处理需求，使得流式计算(Stream Computing)的研究得到重视。作为主要的流式计算[4]技术之一，复杂事件处理技术(Complex Event Processing)针对持续的事件流，采用通过提取符合特定模式的事件序列进行实时处理的工作方式，能够适应海量数据处理中低时延、高吞吐量的需求。然而，随着云计算技术的普及，实时应用系统需要处理流速变化、位置分散的异构数据源，而事件驱动方式符合分布式系统常用的异步处理方式，因此采用复杂事件处理技术来实现分布式环境下的流式计算成为必然的选择。

目前，复杂事件处理[5]的研究主要聚焦于两个问题：丰富的事件检测功能和高效的事件处理技术。现有的Sase+和Esper[5]等系统提供了复杂事件检测语言，支持基于克林闭包和正规式匹配的事件检测。Raced系统[6]提供了高效的流速自适应机制，通过集群计算实现高效处理。XSeq[2]基于可见

到稿日期：2015-11-30 返修日期：2016-02-27 本文受北京市自然科学基金项目(4122011)，国家自然科学基金青年基金项目(61202074)资助。
王亦雄 硕士生，主要研究方向为XML数据库技术，E-mail：466144425@qq.com；廖湖声 教授，主要研究方向为软件自动化方法、编译技术、XML数据处理、程序分析与程序变换，E-mail：liaohs@bjut.edu.cn(通信作者)。

下推自动机(Visual Pushdown Automata)，支持基于正规式匹配的高效的半结构化数据流处理，但不支持跨网络的分布式处理。Apache S4 和 Apache Storm 系统[5]支持基于集群的流式计算，计算能力强，但编程接口简单，事件检测功能薄弱。

针对上述问题，本文提出一种分布式复杂事件处理语言 CEStream，为分布式环境下实时应用系统的开发提供了一个软件开发工具。本文主要贡献如下：

(1)提出了一种复杂事件流处理语言 CEStream，其以层次化数据为数据模型，提供了一种被称作正规树模式匹配的复杂事件检测功能，同时支持时间序列的正规式匹配和层次化数据的结构约束。

(2)面向基于多事件源的事件检测需求，提出并实现了一种基于自动机的复杂事件检测方法，其能够捕获来自不同事件源、符合特定正规式模式的组合型事件。

(3)针对上述组合型事件的检测过程，提出了一种查询分解方法，实现了基于流数据处理集群的复杂事件并行检测。

(4)设计并实现了一个 CEStream 复杂事件处理引擎系统，并通过实验表明该系统实现了 CEStream 语言，能够完成正规树模式匹配和多源组合型事件检测等特色功能，达到了低延时和高吞吐量的设计目标。

2　复杂事件流处理语言 CEStream

针对来自分布式网络的流数据，本文提出一种复杂事件流处理语言 CEStream。CEStream 是一种声明型语言，以层次化的流数据作为事件源，提供了一种基于正规树模式的复杂事件检测和多源复杂事件处理功能，能够针对来自不同事件流的复杂事件按照特定时序识别事件之间的关系，对多个事件流的检测和识别进行并行处理。

CEStream 语言中的正规树模式是一种正规式和树结构的混合模型。它支持层次化数据的结构约束及值谓词约束，同时采用正规表达式的形式来描述事件流中连续事件之间的约束关系。对于不同来源的事件，CEStream 语言提供了基于正规式的事件关系识别机制，并允许提取事件属性组成新的事件流，进而支持事件流的变换、归并和分解。下面通过几个实用案例说明其语言功能。

2.1　单事件源识别案例：火情检测

火情检测是复杂事件流处理中的一项典型应用，事件处理引擎接收指定设备检测到的温度数据信息，根据预先制定好的规则进行分析，并发出火情警告通知。其中，数据源是温度记录仪内部的数据记录，采用 XML 格式描述。

流数据检测规则为：若 30s 内连续出现 3 次以上温度检测值大于 80°的事件，则发出一条警告信息(含当前时间)。实现这个检测规则的 CEStream 程序描述如表 1 所列。

表 1　案例 1 的程序描述

案例 1
(1)create pattern $tp (/temp{$v, $t}) where tag($v)='value' and tag($t)='time' and $v>80;
(2)in $ts 'http://temperature.org/stream/';
(3)out $s0;
(4)create stream $s0 {⟨alert(⟨time($ts.$t)/⟩)/⟩} from ($ts match $tp)* where count($tp)>3 within time(30)

该 CEStream 程序包含了正规树模式定义(create pattern)、输入绑定(in)、输出绑定(out)和事件流定义(create stream) 4 种语句。语句(1)定义了一个正规树模式 $tp，该模式匹配事件中的 temp 节点要求温度大于 80°，并将子节点中的时间信息 time 与温度信息 value 依次绑定在变量 $t 与 $v 上。语句(2)和语句(3)定义了该程序的输入流与输出流。语句(4)定义了事件流 $s0，用克林闭包($ts match $tp)* 结合 where 子句和 within 子句的方式，检测 30s 内连续出现 3 次以上温度事件源 $ts 与正规树模式 $tp 的匹配结果，提取其中的时间信息 $ts. $t 构造成 XML 片段(⟨alert/⟩节点)并输出。于是，对于满足这个正规树模式匹配的每个事件组，该程序都会构造一个 alert 节点，从而形成输出事件流。

由此可见，CEStream 语言提供的正规树模式定义借用了 XPath 语言描述双亲子女关系的结构约束关系(如：/temp/)和谓词约束(如：$v>80)，也允许采用正规式规定兄弟之间的顺序关系(如：$v, $t)。同时，在事件流定义中，允许采用正规式算子克林闭包规定多个模式匹配的顺序，如：($ts match $tp)*，并使用 count 和 time 等算子来补充匹配条件，利用形如⟨alert(…)/⟩的算子来构造输出事件流。

2.2　多数据源组合事件识别案例：盗窃账户行为检测

盗窃账户行为检测的数据来源是网银系统的后台日志数据流。如果将用户的不同操作记录看作不同的事件流，则可通过检测每个用户动作的时序关系分析出可能存在盗窃账户风险的操作行为。

假定事件识别规则为：对于同一个用户，60s 内接连发生用户登录，修改密码，至少一次转账或支付的事件即判定有盗窃账户的风险。实现这个检测规则的 CEStream 程序描述如表 2 所列。

表 2　案例 2 的程序描述

案例 2
(1)create pattern $p1 (/loginerr{$n, $t}) where tag($t)='time' and tag($n)='name';
(2)create pattern $p2 (/login{$n, $t}) where tag($t)='time' and tag($n)='name';
(3)create pattern $p3 (/change{$n, $t}) where tag($t)='time' and tag($n)='name';
(4)create pattern $p4 (/transfer{$n, $t}) where tag($t)='time' and tag($n)='name';
(5)create pattern $p5 (/pay{$n, $t}) where tag($t)='time' and tag($n)='name';
(6)create pattern $p6 (/account{$n, $t}) where tag($t)='time' and tag($n)='name';
(7)in $s1 'http://account.org/loginerr/stream/';
(8)in $s2 'http://account.org/login/stream/';
(9)in $s3 'http://account.org/change/stream/';
(10)in $s4 'http://account.org/transfer/stream/';
(11)in $s5 'http://account.org/pay/stream/';
(12)out $s0;
(13) create stream $s6 {⟨account(⟨name($s2.$n)/⟩, ⟨time($s2.$t)/⟩)/⟩} from ($s1 match $p1)*, $s2 match $p2, $s3 match $p3 join $s1.$n, $s2.$n, $s3.$n where count($p1)=0 or count($p1)>2 within time(60);
(14) create stream $s0 {⟨alert(⟨name($s6.$n)/⟩, ⟨time($s6.$t)/⟩)/⟩} from $s6 match $p6, ($s4 match $p4 | $s5 match $p5)* join $s6.$n, $s4.$n, $s5.$n where count($p4)>0 or count($p5)>0 within time(60);

其中,语句(1)—语句(6)定义了 6 个正规树模式,分别用于匹配事件流中的 loginerr 节点、login 节点、change 节点、transfer 节点、pay 节点以及 account 节点。匹配中将用户信息 name 与时间信息 time 分别绑定在变量 $n 与变量 $t 上。语句(7)—语句(12)分别指定了本案例的输入事件流和输出事件流。

语句(13)对登录异常事件源 $s1、登录事件源 $s2 和修改密码事件源 $s3 这 3 个输入事件流分别采用 $p1,$p2 和 $p3 进行匹配,根据 3 个模式的匹配结果来构造事件流 $s6 以表示可疑登录事件。在 from 子句采用正规式中的连接算子',',析取算子'|'和克林闭包算子'*'来描述各个事件发生的时序关系,检测出 60s 内同一用户连续匹配 2 次以上或 0 次登录异常数据 $s1、用户登录数据 $s2 和修改密码数据 $s3 的情况,将匹配结果按照 join 子句中 $s1.$n,$s2.$n 和 $s3.$n 的变量值进行连接,并提取用户登录数据中的时间信息 $s1.$t 与用户信息 $s1.$n 构造成 XML 片段(account 节点),以作为事件流 $s6 的数据项输出。

语句(14)对可疑登录事件流 $s6、转账事件流 $s4 和付款事件流 $s5 分别采用模式 $p6,$p4 和 $p5 进行匹配,根据 3 个匹配结果构造出事件流 $s0。按照 from 子句中正规式描述的时序,每个可疑登录事件后出现的转账和付款情况被检测出来,形成〈alert〉警告信息,组成事件流 $s0。

2.3 语言设计

下面给出 CEStream 语言的核心语法,其中{ }表示 1 次或多次,[ ]表示 0 次或 1 次。

(1)事件检测模式

CEStream 语言提供了一种叫作正规树模式的事件检测模式,其综合了基于正规式的连续事件时序关系检测功能和 XPath 表达式的结构约束与谓词筛选功能。其语法描述如表 3 所列。

表 3 正规树模式定义语法

正规树模式定义语法
pd::=create pattern ID '(' *re* ')' [where *logic*]
re::=*re*[',' *re*] | *re*['|' *re*] | *re* '*' | '(' *re* ')' | ID | *path*['{' *re* '}']
path::=*step*[*path*]
step::='/' *nodetest*['[' *predicate* ']'] | '//' *nodetest*['[' *predicate* ']']

正规树模式定义描述中包含模式名(ID)、正规表达式、逻辑式 3 大部分。正规表达式(*re*)在连接、克林闭包和析取算子的基础上扩展了 XPath 路径检测表达式,具有路径检测(*path*)、节点检测(*nodetest*)和谓词筛选(*predicate*)等功能,还可以利用 where 子句描述连续事件之间的约束关系,将事件属性值绑定到分量名。逻辑表达式(*logic*)针对分量名表述的事件属性,结合 CEStream 提供的内置函数(如 tag,prev,count)来描述检测条件。

(2)事件流定义语法

事件流定义语句用于描述如何从一个或多个输入事件流构造出新的事件流,支持通过正规式匹配从若干个事件流中获得信息,按照给定的构造式及时序关系构造出新的事件流。构造时可以以'模式名.分量名'的形式来访问事件源中的数据。其语法描述如表 4 所列。

表 4 事件流定义语法

事件流定义语法
sd::=create stream ID '=' *constructor* from *pair* { ',' *pair* } [where *logic*] [within *win*]
sd::=create stream ID '{' *constructor* '}' from *rp* [join *var* {',' *var*}] [where *logic*] [within *win*]
pair::=ID match ID
rp::=*rp*[',' *rp*] | *rp*['|' *rp*] | *rp* '*' | '(' *rp* ')' | *pair*
var::=ID '.' ID

事件流定义描述(*sd*)中包含事件流名、构造式、事件源、连接式、逻辑式及窗口几大部分。构造式(*constructor*)利用正规树模式内部定义的分量名提取每个匹配结果的值,根据用户描述的结构构造出 XML 片段并输出。match 算子作用于事件流名和模式名,提供了基于模式的事件检测功能。from 子句提供了多源组合事件检测的识别功能。多源事件检测表达式(*rp*)提供了基于正规式的组合事件检测功能,利用连接、析取、克林闭包等算子来检查事件之间的时序关系。连接式(join)用于指定多个事件源之间的连接条件。within 子句约束匹配数据集的窗口范围,包括时间、事件数与条件范围,窗口之外的事件不参与到事件流匹配的过程中。

3 CEStream 复杂事件流处理引擎

3.1 系统简介

CEStream 复杂事件流处理引擎系统实现了 CEStream 语言。该系统是一个基于流数据处理集群的针对分布式 XML 数据源并支持多数据源事件流查询的处理引擎系统。系统针对 CEStream 查询描述进行编译处理与结构优化,将查询任务分解并分配到流数据处理集群中的各个节点与事件检测代理服务器上进行处理,最终根据用户给出的构造式将查询结果进行包装后以 XML 事件流的形式输出。

3.2 系统架构设计

CEStream 分布式复杂事件流处理引擎系统包含一个客户端、一个 CEP 服务器、多个事件检测代理及一个流数据处理集群。图 1 示出了 CEStream 系统架构图。

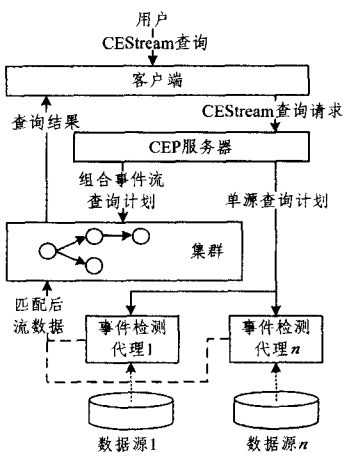

图 1 系统架构图

系统中，客户端负责发出 CEStream 查询请求，并接收查询结果。

CEP 服务器负责将用户给出的符合 CEStream 语法标准的查询请求进行编译，分解后生成若干个单源事件处理任务、若干个多源事件处理任务和一个事件流处理拓扑图。这个拓扑图描述各种事件处理任务之间的依赖关系。随后，CEP 服务器将每个单源事件处理任务委托给不同的事件检测代理，同时将事件流处理拓扑图发送给流数据处理集群。

事件检测代理是一组分布式的代理服务器，根据 CEP 服务器给定的单源事件处理任务中的事件源地址与正规树模式，负责从指定 URL 读取事件源并进行正规树模式匹配与筛选，将结果以流数据的形式发送给流数据处理集群。

流数据处理集群为每个事件处理任务分配处理单元，并按照事件流处理拓扑图建立各个处理单元的依赖关系。各个处理单元按照规定的拓扑结构接收事件检测代理传送过来的匹配结果，将其包装成数据元组并使之沿拓扑路径依次接受处理，通过一种自动机模块进行组合型事件的正规式匹配，最终构造出 XML 事件流的形式发送给拓扑结构规定的处理单元或客户端。

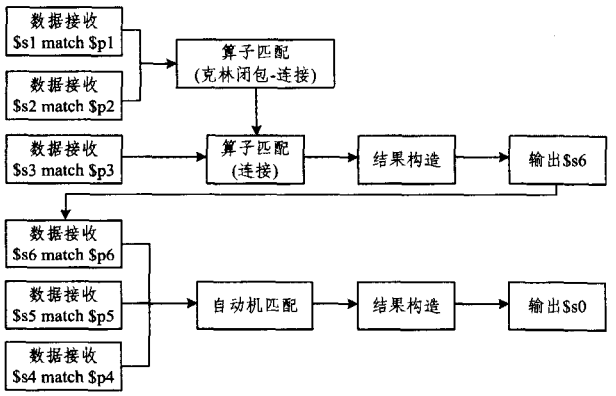

图 2　集群拓扑图

根据 2.2 节案例所创建的流数据处理集群的拓扑结构，图 2 示出了若干个处理单元及处理单元间的拓扑关系。该案例中包含两个事件流查询计划。其中 $s6 对应的查询计划可以分解为一个结果构造任务以及克林闭包-连接操作和连接操作这两个多源事件处理任务，所以根据这两个算子类型可以构造出对应的两个算子匹配单元。$s0 对应的查询计划无法分解，所以使用一个自动机匹配单元和一个结果构造单元进行组合型事件匹配和输出事件流的构造。

4　实验

根据事件流中事件源数量的不同，存在单事件源查询和多数据源查询两种情况，实验中分别应用不同的 XML 数据集测试 CEStream 分布式复杂事件流处理引擎系统每分钟可处理的事件数量与不同数据源流速下的平均响应时间。本实验以网站服务器中各种 XML 格式的操作日志为数据源。实验数据的统计不包含读取事件源与输出结果事件至文件的时间。本节的所有测试中，软件环境为 JDK1.7，硬件环境为 6 个节点的集群环境，每个节点配置为 Intel Xeon E5-2650 2.6GHz，20GB 内存。

图 3 展示了表 5 所列的 11 种不同的查询案例的吞吐量。图 4 与图 5 展示了不同检测类型中典型案例的响应时间以及多源匹配使用分解算法前后的响应时间对比。测试中针对用例 Q3.1－Q3.6（见表 5），检测了使用分解算法前后的吞吐量，各个案例分解后的吞吐量与分解前相比平均提高了 29.8%。经过分析得知，性能主要消耗在事件检测代理与流数据处理集群的数据传输以及内部各个处理单元的数据传输上。本系统已经基本满足毫秒级的响应时间及每分钟数百万的吞吐能力，从目前复杂时间流处理的应用场景来看，可以满足大部分应用的需求。

表 5　CEStream 测试案例

案例名	事件检测表达式	检测类型	吞吐量/事件数/s	
			分解前	分解后
Q1.1	$s1 match $p1	单源检测	72000	—
Q2.1	($s1 match $p1)*,$s2 match $p2,($s3 match $p3)*	多源不可分解	25800	—
Q2.2	($s1 match $p1 | $s2 match $p2)*,($s3 match $p3)*	多源不可分解	24200	—
Q2.3	($s1 match $p1,$s2 match $p2)*,($s3 match $p3)*	多源不可分解	23230	—
Q2.4	($s1 match $p1,$s2 match $p2,$s3 match $p3)*	多源不可分解	24300	—
Q3.1	$s1 match $p1 | $s2 match $p2 | $s3 match $p3 | $s4 match $p4	多源可分解	31200	41030
Q3.2	$s1 match $p1,$s2 match $p2,$s3 match $p3,$s4 match $p4	多源可分解	30100	42100
Q3.3	$s1 match $p1,$s2 match $p2,($s3 match $p3 | $s4 match $p4)	多源可分解	30150	40890
Q3.4	$s1 match $p1 \| ($s2 match $p2,$s3 match $p3,$s4 match $p4)	多源可分解	29300	38910
Q3.5	($s1 match $p1 | $s2 match $p2),$s3 match $p3,$s4 match $p4	多源可分解	27810	36070
Q3.6	($s1 match $p1 | $s2 match $p2)*,($s3 match $p3 | $s4 match $p4)	多源可分解	28400	32010

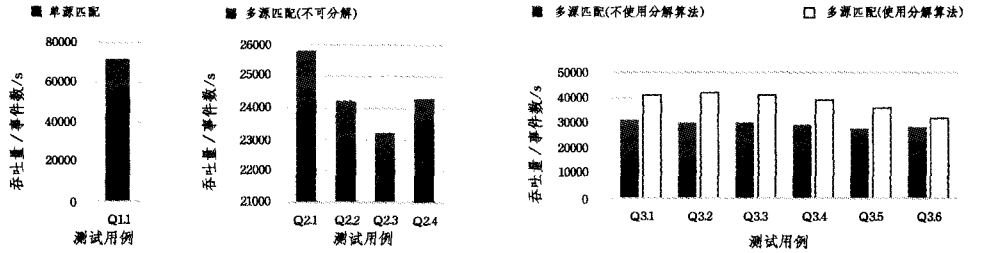

图 3　各案例的吞吐量对比

（下转第 164 页）

者被确定为第一个发现特定漏洞的人，该漏洞需要被证实为真实的。确定该漏洞被初始发现时的重要性比例通过$(T_1-T_0)/(T_3-T_0)$进行计算，若该比例较高，则说明此发现十分有用，因为在很短的时间内就能够累积一大笔资金。很明显，部分用户群将会从修复漏洞中获得重要的实用程序。

同样，如果花费的时间段T_2-T_1较长，则往往表明修复意义重大，并且漏洞修复者花费了相当大的精力。同样，时间段T_3-T_2越长，修复就越健壮。因此，测试者越早发现其中的缺陷，越能从中获得更大的份额。

结束语 本文将众包任务的分解完成工作抽象为一个市场平台，引入交易成本理论，设计了众包模式下的新的并行管理模型，然后分析了设计原则，并用数学方法进行符号化定义、成本分析与托管以及买卖双方均衡点的设置，最后以“漏洞修复”为案例讲解了模型的应用。模型具有一定的创新性，有助于实现供需双方资源管理与优化分配，提高开发效率。“代理”监管的有效性以及模型中的效率量化是下一步要研究的内容。

参 考 文 献

[1] ZHANG Z Q, et al. Research on quality control strategy and evaluation algorithm of [J]. Chinese Journal of Computers, 2013, 36(8): 1636-1649. (in Chinese)
张志强，等. 众包质量控制策略及评估算法研究[J]. 计算机学报，2013，36(8)：1636-1649.

[2] ITWORLD B A. Finding freelance jobs: 6 sites for talented techies [OL]. http://www.itworld.com/article/2826288/careers/finding-freelance-jobs-6-sites-for-talented-techies.html.

[3] http://baike.so.com/doc/6347317-6560942.html

[4] BUYYA R, ABRAMSON D, GIDDY J, et al. Economic models for resource management and scheduling in grid computing [J]. Concurrency & Computation, 2002, 14(13-15): 1507-1542.

[5] CHE Z B, XING Y. Research on Composable Cost Model of Cloud Resource Economics Management Structure [J]. Computer engineering and Science, 2011, 33(8): 185-189. (in Chinese)
车战斌，邢颖. 云资源经济学管理架构中组合成本模型研究[J]. 计算机工程与科学，2011，33(8)：185-189.

[6] TRUONG H L, BRUNNER P, NAE V, et al. Dipas: A distributed performance analysis service for grid service-based workflow [J]. Future Generation Comp. Syst., 2009, 25(4): 385-398.

[7] GEIMER M, SHENDE S, MALONY A D, et al. A generic and Configurable Source-code Instrumentation Component[M]// Allen G, Nabrzyski J, Seidel E, et al., eds. ICCS (2), Vol. 5545 of Lecture Notes in Computer Science. Springer, 2009: 696-705.

(上接第 143 页)

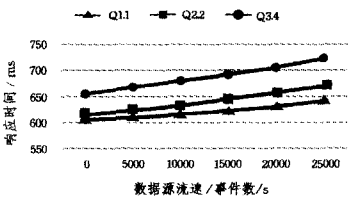

图 4 各案例的响应时间对比

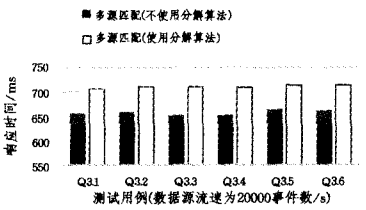

图 5 多源匹配使用分解算法前后的响应时间对比

结束语 本文提出了一种新式的语言 CEStream 和与之相对应的查询引擎，并针对分布式环境进行了优化，有效地控制了系统各个节点之间的通信量，充分利用集群的计算能力，提高了整体性能。CEStream 语言的设计与实现仍需要继续研究与发展。在语言设计中可以考虑引入更多实用的语言功能。

参 考 文 献

[1] ZHANG Q, CHENG L, BOUTABA R. Cloud computing: state-of-the-art and research challenges [J]. Journal of Internet Services and Applications, 2010, 1(11): 7-18.

[2] ZENG K, YANG M, MOZAFARI B, et al. Complex pattern matching in complex structures: The XSeq approach [C]// 2013 IEEE 29th International Conference on Data Engineering (ICDE). IEEE, 2013: 1328-1331.

[3] BUGHIN J, CHUI M, MANYIKA J. Clouds, big data, and smart assets: Ten tech-enabled business trends to watch [J]. McKinsey Quarterly, 2010, 56: 75-86.

[4] BAI Y, THAKKAR H, WANG H, et al. A data stream language and system designed for power and extensibility [C]// CIKM. ACM, 2006.

[5] CUGOLA G, MARGARA A. Processing flows of information: From data stream to complex event processing [J]. ACM Computing Surveys (CSUR), 2012, 44(3): 1-62.

[6] CUGOLA G, MARGARA A. RACED: an adaptive middleware for complex event detection [C]// Proceedings of the 8th International Workshop on Adaptive and Reflective Middleware. ACM, 2009: 5.

[7] Peng F, Chawathe S S. Xpath queries on streaming data [C]// SIGMOD. 2003.

[8] DIAO Y, IMMERMAN N, GYLLSTROM D. Sase+: An agile language for kleene closure over event streams [OL]. [2012-12-23]. http://archive.systems.ethz.ch/www/dbis.ethz.ch/education/ws0708/adv_top_infsyst/papers/sase_tr07.pdf.

[9] ABADI D, CARNEY D, CETINTEMEL U, et al. Aurora: a data stream management system [C]// SIGMOD. ACM, 2003.

[10] RASANEN O, KAKOUROS S. Modeling Dependencies in Multiple Parallel Data Streams with Hyper dimensional Computing [J]. Gnal Rong Lr, 2014, 21: 899-903.

[11] AGUADO J, MENDLER M. Towards Strategies for Data Flow Programming [C]// Preproceedings of the 22nd Symposium on Implementation and Application of Functional Languages (IFL 2010). 2010: 372-473.

[12] KIEBURTZ R B. Codata and comonads in Haskell [J/OL]. http://core.ac.uk/display/24270653.