# 分布式数据流关系查询技术研究

王春凯 孟小峰
(中国人民大学信息学院 北京 100872)

**摘 要** 随着在线分析连续数据流的需求日益增多,用于实时处理海量、易变数据的数据流管理系统由此产生.大数据时代下,随着开放式处理平台的发展,为处理大规模且多样化的数据流,出现了若干分布式流处理系统,如 S4、Storm、Spark Streaming 等.然而,为提升处理系统的易用性和处理能力,需要在其之上构建具有抽象查询语言的关系查询系统,以构筑完整的分布式数据流管理系统.如何设计并实现高效易用的关系查询系统是一个亟待解决的问题.文中首先概述了分布式数据流查询处理的典型应用、数据特征和实现目标.进而,提出了分布式数据流关系查询系统的基础架构,并基于此架构深入分析了用户自定义函数查询、查询优化、驱动方式、编译技术、算子管理、调度管理和并行管理等关键技术.然后,对比分析了 SPL、StreamingSQL、Squall 和 DBToaster 这 4 种具有代表性的查询系统实例.最后,指明了该技术在优化技术、执行策略、实时精准查询和复杂查询分析等方面所面临的挑战和今后的研究工作.

**关键词** 大数据;数据流;流处理系统;流查询系统;关系查询技术
**中图法分类号** TP311 **DOI 号** 10.11897/SP.J.1016.2016.00080

# Relational Query Techniques for Distributed Data Stream: A Survey

WANG Chun-Kai MENG Xiao-Feng
(*School of Information, Renmin University of China, Beijing* 100872)

**Abstract** The applications that require online processing continuous data stream are increasing. Data stream management systems which are used to deal with massive and variable data in real time have been produced. With the development of open processing platforms in the ear of big data, a number of distributed data stream processing systems have emerged for dealing with large scale and diverse data stream, such as S4, Storm, Spark Streaming, etc. However, we should construct relational query systems which have abstract query language on basis of the processing systems for improving the ease of use and processing capability of them, so as to build complete distributed data stream management systems. How to design and realize the high efficiency and easy-to-use query systems is a great challenge. In this survey, we first provide an overview of typical applications, data characteristics and achieve goals of distributed data stream query processing. Furthermore, we propose the framework of distributed data stream relational query systems. Based on the framework, we analyze the key techniques in several aspects: UDF query, query optimization, query-driven approaches, compiling techniques, operator management, scheduling management and parallel management. Then, there is the comparison of representative query systems including SPL, StreamingSQL, Squall and DBToaster. Finally, some new challenges are put forward, including optimization technique, execution strategy, real-time precise query and complex query analysis.

**Keywords** big data; data stream; stream processing system; stream query system; relational query technique

收稿日期:2015-01-29;在线出版日期:2015-10-16.本课题得到国家自然科学基金(61379050,91224008)、国家"八六三"高技术研究发展计划项目基金(2013AA013204)、高等学校博士学科点专项科研基金(20130004130001)、中国人民大学科学研究基金(11XNL010)资助.
**王春凯**,男,1981 年生,博士研究生,主要研究方向为分布式数据流管理.E-mail:chunkai_wang@163.com.**孟小峰**,男,1964 年生,博士,教授,博士生导师,主要研究领域为网络数据管理、云数据管理、移动数据管理、社会计算、闪存数据库系统、隐私保护.

# 1 引　言

近年来，随着社交网络、物联网和移动互联网等领域的快速发展，数据量呈爆炸式增长，大数据时代已真正到来[1-3]. 在这些应用中，数据变化的速度越来越快，需要处理和响应的时间越来越短，对大数据3V特性[4]中的高速性(velocity)处理便显得非常重要. 因此，数据流的实时分析和流式处理成为当今热点研究领域之一.

21世纪初，学术界开始对数据流管理系统(Data Stream Management System，DSMS)展开研究工作并推出了原型系统，如Aurora[5]、TelegraphCQ[6]和STREAM[7]等. 研究者认为，数据流的产生一般通过外部资源获取(如传感器网络等)，而非人为主动地发出数据，因此，数据流管理系统属于系统主动-用户被动(system-active human-passive)[5]的管理模式. 此外，数据流管理系统处理的数据重点是按照时间排序的内存数据，这要求系统具有序列化的访问模式和实时的响应时间. 并且，数据库管理系统(DataBase Management System，DBMS)一般提供即席(ad-hoc)查询模式，而数据流管理系统往往针对的是连续查询(Continuous Query，CQ). 表1对数据库管理系统和数据流管理系统进行了对比分析.

**表1　数据库管理系统和数据流管理系统对比分析**

| 对比指标 | 系统管理模式 | 数据状态 | 访问模式 | 响应时间 | 查询模式 |
|---|---|---|---|---|---|
| 数据库管理系统 | 用户主动-系统被动 | 历史数据 | 随机访问 | 非实时响应 | 即席查询 |
| 数据流管理系统 | 系统主动-用户被动 | 时间序列数据 | 序列访问 | 实时响应 | 连续查询 |

大数据时代下，出现了使用MapReduce[8]编程框架的分布式数据库管理系统，文献[9]对使用MapReduce框架的关系查询处理语言、实现的算子以及系统实例等整个分布式数据管理系统的生态圈进行了详实的综述和分析. 为满足数据流海量和高速等特点的流式处理，工业界和学术界相继出现了大量的分布式数据流处理系统，如Yahoo!的S4[10]、Twitter的Storm①[11]和伯克利大学的Spark Streaming②等. 目前国内外对分布式数据流处理的相关综述文章包括：文献[12]对大数据流处理的关键技术和流行的系统实例进行了分析；文献[13]着重关注数据流处理系统的实现技术；文献[14]把分布式流处理系统作为一种实时处理的内存数据管理系统进行了说明. 然而，目前的分布式数据流处理系统对数据的查询操作需要用户编码来实现，系统本身未提供相应的查询语言. 因此，为提高处理系统的易用性和处理能力，增强应用程序的可重用性和移植性，需在处理系统之上实现具有抽象查询语言的关系查询系统③，以构筑完整的分布式数据流管理系统. 本文重点分析查询系统中关于关系查询技术④的相关研究工作.

本文第2节总体概述分布式数据流查询处理的典型应用、数据特征和实现目标；第3节提出分布式数据流查询技术的基础架构，并基于此架构深入分析用户自定义函数查询、查询优化、驱动方式、编译技术、算子管理、调度管理和并行管理等关键技术；第4节对比分析具有代表性的数据流查询系统；第5节指明针对优化技术、执行策略、实时精准查询和复杂查询分析等方面所面临的挑战和今后的研究工作.

# 2 分布式数据流查询处理概述

分布式数据流处理系统具有查询响应时间短、查询精度高、可靠性强、易于扩展和使用灵活等特点. 然而，数据流往往具有监控周期长、输入速率不稳定等特征. 在处理系统之上，如何构建高效、易用的查询系统是满足用户查询需求的关键所在.

本节列举了分布式数据流查询处理的典型应用，分析了查询处理中分布式数据流的特征，并阐述了查询处理的相关目标.

## 2.1 分布式数据流查询处理的典型应用

分布式数据流处理系统的应用范围较为广泛. 如微博、Twitter等社交网站提供实时搜索博文的服务；移动互联网领域提出的实时监控移动宽带业务的需求；金融领域针对高频交易的实时分析软件；以及电子商务领域的实时推荐系统等.

随着这些应用需求的不断扩展，出现了相关的查询系统. 如洛桑联邦理工学院数据实验室基于Storm

① http://storm.apache.org/
② https://spark.apache.org/streaming/
③ 下文将关系查询系统简称为查询系统.
④ 下文将关系查询技术简称为查询技术.

推出了针对实时查询处理的 Squall① 系统;文献[15]为提升 Squall 连接操作的扩展性并降低数据的冗余度,推出了 BiStream 系统;以及腾讯公司基于 Storm 设计的实时数据流推荐系统 TencentRec[16] 等.

### 2.2 分布式数据流的特征

为确保分布式数据流的实时处理,需要对数据流的传输和模型进行说明.

(1)数据流传输.为保证实时、完整且稳定地将数据流传输到处理系统,一般可通过消息队列和网络 Socket 传输等方法完成,以保证将数据发送到每台物理节点,为数据处理提供保障.利用消息队列的方式进行数据采集和传输是较为常用的一种方法,常见的消息队列产品有 Facebook 的 Scribe②、LinkedIn 的 Kafka③ 和 Cloudera 的 Flume④ 等.

(2)数据流模型.在查询处理过程中,由于数据流的来源不同,需要针对不同的数据源制定不同的数据样式.一般来讲,通用的数据流管理系统支持关系型数据模型,数据定义语言是基于关系型的原子类型,便于以属性和元组的形式划分和发送数据;针对特殊领域的数据流管理系统,可根据领域数据的特点设计基于对象类型的复合数据类型.如针对传感器网络的 COUGAR[17] 和管理网络数据传输的 Tribeca[18] 是利用复合数据类型构建的系统.

### 2.3 分布式数据流查询处理的目标

将分布式数据流查询处理集成到处理系统之上,可便于用户灵活地调用处理系统的函数库,提高处理系统的易用性.分布式数据流查询处理需要实现的目标可总结为以下几点:

(1)声明式查询语言.类似于基于 Hadoop⑤ 构建的 Hive[19] 中设计 HiveQL 的思想,在数据流处理系统之上设计的查询系统需支持类似 SQL 的声明式描述语言,并将查询计划自动编译成处理系统的工作任务.利用查询系统提供的算子库可降低用户使用处理系统的难度,提升使用效率.

(2)查询的自适应性.为应对分布式数据流传输速率不稳定和数据动态分布等特点,查询系统应具有自适应性.可根据查询算子处理数据流的时间变化、查询谓词选择率的变化以及数据流传输速率的变化,实现查询算子的动态调度和自适应地查询处理[20].

(3)查询的可扩展性.为快速响应查询请求,查询系统需支持查询算子的可扩展性.设计完备的可扩展策略,可降低分布式处理的开销,提高查询性能.可扩展性策略一般涉及到两个方面:① 伸缩性.可根据处理数据流数量的变化,动态增减处理节点;② 有效性.尽量提高新增处理节点的使用效率,充分利用集群的整体资源.

(4)查询的容错性.查询系统的查询过程中,一般存在两种数据丢失的现象.一种是执行过程中算子状态信息的数据丢失,这种数据的丢失往往通过复制算子的状态信息进行容错处理;另外一种是分布式环境下网络传输过程中的数据丢失,这种情况一般通过上游备份[21] 和主动备份[22] 等策略进行容错处理.

(5)查询的实时性.由于数据流具有不稳定性和易变性等特点,需要实时获取查询过程的中间结果.因此可根据用户接受的查询延迟时间,设定批量处理数据的大小[23],将查询结果分批返回给用户;或者可引入过程调用等技术,将运行结果实时返回给用户,如在 Storm 中引入的分布式远程过程调用(Distributed Remote Procedure Call,DRPC)技术.

## 3 分布式数据流查询技术研究

新型的分布式数据流处理系统,提供开放的编程接口,可为用户提供丰富的计算资源.然而,如何高效、便捷地使用处理系统提供的计算资源,是设计查询系统的关键所在.本节首先提出分布式数据流查询系统的基础架构,然后重点分析用户自定义函数(User Defined Function,UDF)查询、查询优化、查询驱动方式、编译技术、算子管理、调度管理和并行管理等关键的查询技术.

### 3.1 分布式数据流查询系统架构

分布式数据流处理系统具有高扩展性、高可用性、负载均衡和容错控制等特性.因此,在这种开放式处理系统之上构建的分布式数据流查询系统不同于传统的集中式系统.这要求系统在查询优化、调度管理和并行管理等方面要有满足分布式特征的架构体系.我们归纳总结了分布式数据流查询系统的技术要点,并设计了基础架构.如图 1 所示,该架构被分为 5 个方面.

① https://github.com/epfldata/squall
② https://github.com/facebookarchive/scribe
③ http://kafka.apache.org/
④ http://flume.apache.org/
⑤ http://wiki.apache.org/hadoop

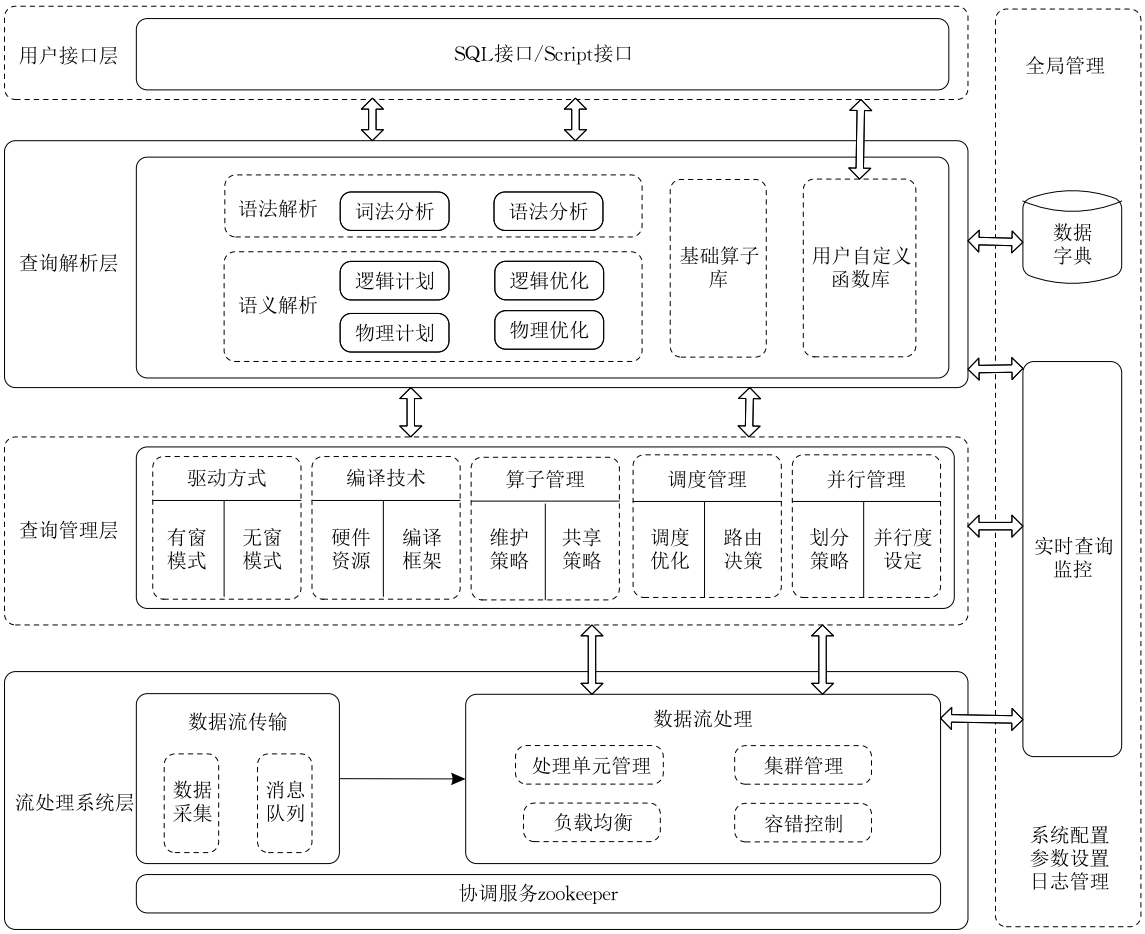


图 1　分布式数据流查询系统架构

(1)用户接口层. 该层负责接收用户的查询请求,提供统一的查询接口. 即制定相应的 SQL 查询标准,并支持用户自定义函数(UDF)的访问模式和复杂查询过程的脚本模式,以及提供图形化的访问界面等.

(2)查询解析层. 根据用户接口层提交的查询请求,进行语法和语义解析. 完成词法和语法的解析任务,生成相应的逻辑查询计划和物理查询计划并对其进行优化. 该层还提供基于查询处理所需的基础算子库,以及根据用户需求构建的用户自定义函数库.

(3)查询管理层. 控制查询系统的整体流程,是提升查询性能的关键一层. 我们将该层分为 5 个模块:① 查询驱动方式. 根据查询请求制定不同的查询驱动方式(如滑动窗口和无窗口的驱动方式);② 编译技术. 根据新型的硬件资源特征设计的查询编译框架及其优化技术;③ 算子管理. 针对分布式环境下有向无环图(Directed Acyclic Graph,DAG)的处理方式,需根据系统性能和算子状态信息,设计查询系统的部署和维护策略,以及针对多查询处理的应用场景需引入算子共享机制;④ 调度管理. 为满足自适应性的查询需求,对查询算子和数据流需进行实时的调度优化和动态的路由决策;⑤ 并行管理. 在分布式环境下,实现针对算子和数据流的划分策略,以及为提升处理系统的性能,动态设置各个处理节点的并行度.

(4)流处理系统层. 完成多数据源的传输和查询任务的执行. 处理层一般利用 ZooKeeper[24] 进行协调服务. 其中,数据流传输往往通过消息队列的方式实现(详见 2.2 节). 数据流处理系统完成执行查询任务的核心功能,负责管理集群和各个处理单元,以及系统底层的负载均衡和容错控制等.

(5)全局管理模块. 负责全局数据词典的维护和更新,并实现全局查询处理的实时监控功能,以及完成整个数据流查询系统的配置信息和日志管理等任务.

### 3.2　分布式数据流查询处理关键技术研究

通过对分布式数据流查询系统架构的分层说明,我们可将流查询系统涉及到的关键技术归纳

为:关注于支持复杂任务分析的用户自定义函数(UDF)查询、查询解析过程中的查询优化、驱动方式、编译技术、算子管理、调度管理以及针对分布式shared-nothing处理框架的并行管理等.

3.2.1 用户自定义函数查询

用户自定义函数(UDF)是数据库管理系统可扩展的高级功能,允许用户通过创建存储方法将复杂SQL逻辑封装成函数,便于用户使用.用户自定义函数(UDF)可发挥其灵活和易用的特性,充分提高查询系统的查询效率和处理能力.

用户自定义函数(UDF)一般有3种类型:标量UDF、源UDF和聚集UDF.标量UDF一般利用返回值为确定类型的标量函数定义,用于操作数据库中的标量类型值(如获取子字符串SUBSTR的标量函数).源UDF是基于已存在函数的扩展,一般可通过系统内置函数或已定义的UDF表示.除了可利用标量函数外,还可针对数据表的列进行扩展操作(如SELECT语句中求最大值MAX的列操作).聚集UDF,又称为用户自定义聚集函数(User Defined Aggregation,UDA),是在UDF的基础上增加了数据分析和挖掘的功能.UDA往往用于决策支持系统,被认为是数据库高级应用的关键所在.UDA作用范围较广,除了可对标量类型和列进行操作外,还可操作整个数据表的任意列.如IBM DB2上实现的高级语言SADL[25],可支持对数据表的不同类型的聚集操作.表2总结了各种UDF的应用范围.

**表2 UDF应用范围**

| | 标量 | 列 | 表 |
|---|---|---|---|
| 标量UDF | √ | — | — |
| 源UDF | √ | √ | — |
| UDA | √ | √ | √ |

在分布式数据流查询系统中,针对数据流动态、易失和非确定等特点,文献[26]提出了在非确定数据集上支持UDF的通用框架,并通过基于高斯过程[27]的学习方法,计算输出UDF结果的近似分布情况,确保查询结果的错误区间.文献[28]基于MapReduce编程模型,利用数据重排的方式为数据划分做预处理,进一步优化了UDF的执行效率.文献[29]构建了UDA的执行框架,重点涉及到关于UDA的查询优化、查询重写和视图维护等关键流程.文献[30]解除了UDF中包含强制性构建函数(如条件分支、循环等)的相关性,允许以面向集合的方式执行查询.通过将嵌套查询重写成平面查询的方法,生成对应的UDF树,并利用转换规则对其进行优化.

通常支持UDF的查询系统需要构建支持UDF的查询分析器和函数库,再根据UDF的查询处理流程生成对应的查询计划和执行计划.

3.2.2 查询优化

查询优化是查询处理的永恒主题,针对数据流易变的特点,查询系统需要快速响应用户提交的查询请求.一般的优化方法分为编译时的静态优化和运行时的动态优化,现归纳如下:

(1)静态优化

静态优化可分为逻辑优化和物理优化两种.

① 逻辑优化.一般采用基于启发式的优化规则,比如优先做选择和投影、剪枝优化、连接条件下推等规则.在分布式环境下,数据流查询系统一般均支持有向无环图(DAG)的处理模型.此时,基于规则的逻辑优化可将规则转换器传入到有向无环图(DAG)的拓扑结构中顺序执行,通过查询重写得到优化后的执行计划.

② 物理优化.从逻辑计划可以派生出多个物理计划,物理优化器则对每个物理计划进行代价估计,从而选择出最小估计代价的物理查询计划,这种代价估计往往基于数据分布的元数据信息.在分布式环境下,需要利用调度器协同考虑各节点的计算能力.在流应用场景中,数据分布的元数据信息可以根据历史数据的信息进行估计(如算子的选择率、数据流速等).基于这些统计信息,查询优化器可以估计中间结果的大小.另外,基于分布式集群中各节点的资源配置情况,估算出不同算子在不同节点上的执行代价.

(2)动态优化.数据流是源源不断输入的,数据分布的信息难以预测和准确估计.由于最优查询计划的代价是基于查询提交时获取的统计信息来估算的,但这些统计信息在查询执行过程中往往会发生改变,很可能导致"最优"查询计划的执行代价变得很高.因此,在查询提交时由优化器选中的查询计划很难保证在整个查询执行过程中都是最优的.这就需要在查询执行的过程中,动态自适应地优化查询执行计划.

表3对比分析了不同的查询优化技术.关于静态优化的代表技术,康奈尔大学数据库研究组设计了基于规则的多查询优化框架RUMOR[31].该框架针对多查询需要共享状态和计算的特点,扩展了基于规则的查询和基于查询计划的处理模型.通过将

多个流扩展并映射成一个信道(channel)的方法，利用信道表述多查询的优化技术，并基于自动机的事件监控系统[32]向基于关系的查询系统进行转换，将查询系统和事件监控系统集成到 RUMOR 中. 文献[33]提到的分布式多模型优化技术综合考虑了输出率和计算代价两个方面，并设计了在分布式环境下分解查询计划的算法，进一步提升了物理优化的效果. 由于数据流不易确定表的基数，文献[34] 针对选择、投影、连接等不同算子构建基于速率的代价模型，并以最大输出速率为优化目标利用启发式规则生成优化框架. 关于动态优化的代表技术，文献[35]在辅助系统上模拟原始数据流，用于验证优化的查询计划. 在保证原始查询计划可正常执行的基础上，仅将已验证的有效查询计划迁移到主系统. 文献[36]针对多个连续查询设计了基于共享的自适应优化方案 A-SEGO . 通过设置共享连接算子的代价边界，可从不同的优化策略中生成全局执行计划，在最优执行计划和执行优化的处理时间上寻求平衡点.

**表 3　查询优化技术**

| 优化策略 | | 代表技术 | 优化方式 |
|---|---|---|---|
| 静态优化 | 逻辑优化 | RUMOR[31] | 基于规则转换的多查询优化框架 |
| | 物理优化 | 分布式多模型优化[33] | 基于输出率和计算代价的代价模型优化 |
| | | 基于速率的优化[34] | 产生输出速率最大化的查询计划 |
| 动态优化 | | 基于模拟的优化[35] | 通过辅助模拟系统优化查询计划 |
| | | A-SEGO[36] | 设置共享连接算子的代价边界进行二次优化 |

3.2.3　驱动方式

根据不同区间的数据流连续查询，可将查询驱动方式分为滑动窗口类型和无窗口类型两种，如图 2 所示.

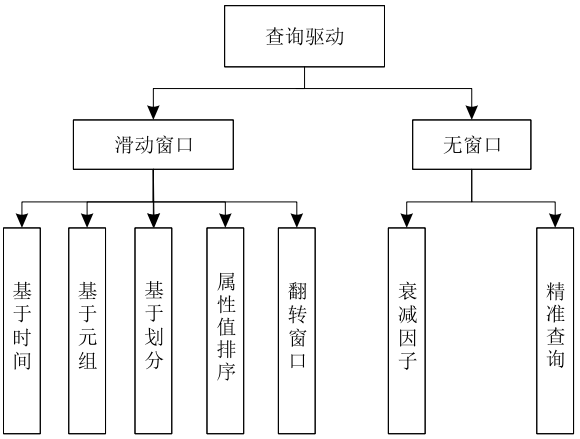


图 2　查询驱动分类

滑动窗口将无限的数据流切分成若干有限子数据流，每次的查询处理仅针对当前滑动窗口内的子数据流. 一般可根据用户设定的时间间隔或窗口内元组数量设定滑动窗口的大小. 除此之外，CQL[37]引入了基于划分的滑动窗口、序列数据库 SEQ[38]设计了基于属性值排序的划分窗口、在 Aurora[5]系统中定义了翻转窗口等.

随着数据流挖掘算法的深入研究和分布式查询处理能力的不断提高，人们需要对窗口外的历史数据进行分析，而基于窗口的驱动方式不能满足这样的要求. 因此，需要引入无窗口的处理模式. 在无窗口模式下，对全部历史(full-history)数据有两种处理方法. 一种是引入衰减因子的概念，即数据流中越陈旧的数据对整个分析模型的影响越小. 此类处理方法往往应用于数据流的挖掘算法中[39-41]. 衰减因子还可分为多项式衰减[42]、指数衰减因子[43]和前向衰减因子[44]等，以说明历史数据对分析模型的影响程度. 另一种是随着分布式数据流处理系统的不断完善，以及实时监控等需求的发展，需要对全部历史数据给出实时精准的查询结果. Squall① 和 BiStream[15]对实时到来的数据流进行路由选择和可扩展的连接查询，保证查询的实时性和正确性；DBToaster[45]基于敏捷视图(agile view)[46]的方法，通过递归编译的思想对数据流进行增量处理，以获得全部历史数据的查询结果.

可以看出，对不同的查询需求和查询系统的处理能力，需要在控制管理阶段选择不同的查询驱动方式.

3.2.4　编译技术

为满足流查询系统的实时快速响应，在考虑查询优化和调度优化的同时，需结合先进的硬件设施和编译技术进一步提高查询效率.

对于以计算为瓶颈的复杂分析任务(如 UDF 的决策支持等)，需要应对较高的查询工作负载. 这需要纵向扩展查询系统的处理能力，尽量提高 CPU 的处理效率，降低指令周期的处理时间. 文献[47]基于 MonetDB 系统设计了一种基于向量处理模型的 X100 查询系统. 运用块处理方式提高缓存的随机读

① https://github.com/epfldata/squall

写效率;并且为了进一步提高 CPU 的吞吐量,X100 可将整个表达式编译成向量原语,无需单独编译各个函数.文献[48]利用与语言无关的 LLVM[49] 编译框架,利用将查询转换成机器代码的方法进行查询优化.并以数据为中心,将待处理的数据推向不同算子.通过利用单指令多数据(Single Instruction Multiple Data,SIMD) CPU 寄存器实现的数据扫描方法[50],可在同一时刻处理多条数据,从而大大提高查询的并行处理能力.

除了将硬件设施和编译框架结合外,还可集成优化策略和编译技术.针对流查询处理均是在内存中进行操作的特点,文献[51]利用基于动态划分的部分分解存储模型(partially decomposed storage model)[52] 和及时(just-in-time)编译的方法,在节省内存带宽的同时,尽量不影响 CPU 的处理能力.文献[53]设计的 LegoBase 查询系统,利用支持运行时编译和代码生成工具的 LMS[54] 框架,实现了高级语言和生成式程序结合的优化方案.Ahmad 和 Koch 设计的 DBToaster 系统[45] 在前端利用 LLVM[49] 编译框架生成 LLVM 程序,在后端利用 LMS[54] 框架递归编译增量的敏捷视图,生成相应的 C++ 或 Scala 函数,实现对数据流的高效处理.

3.2.5 算子管理

为满足用户的不同需求,在分布式环境下,需要对算子库中的算子进行高效的管理,以提升查询的效率,并确保查询的正确性.

为实时获取最新的查询结果,满足用户 pay-as-you-go 的计算模型,文献[55]在处理包含状态信息的算子时(如连接、去重等算子),设计了具有检查、备份、恢复和划分等功能的算子模型,以保证查询系统具有自动扩展虚拟机数目和快速容错的能力.

在多查询的应用场景下,CQL[37] 通过概要(synopses)存储中间状态结果,用于共享算子的处理.SBON[56] 算法利用网络感知的算子部署,自动检测不同查询的数据共享和算子共享,提供多查询优化机制.文献[57]设计的共享感知的中间件技术,可共享多查询情况下的算子和中间结果,自适应地调整网络间的查询部署,提升查询效率.

在分布式数据流查询系统中,针对连接算子的管理最为复杂.实现可扩展且实时的连接操作是一个具有挑战性的研究问题.如表 4 所示,我们总结了目前流行的分布式数据流在线连接算子的连接类型、连接模型和处理方式等相关特性.

**表 4 连接算子对比**

| 连接算子 | 连接类型 | 连接模型 | 处理方式 | 算子特点 |
|---|---|---|---|---|
| Photon[58] | 多数据流等值连接 | 多数据中心 | 非阻塞的元组处理 | 优点:利用中心协调器实现容错和扩展的连接操作<br>缺点:不支持 $\theta$ 连接操作 |
| D-Streams[59] | 多数据流 $\theta$ 连接 | RDD 转换 | 阻塞的微批次处理 | 优点:利用世系机制,实现快速容错降低连接时延<br>缺点:有窗口大小的限制,某些离散的 batch 存在丢失连接元组的现象,导致只能获得近似查询结果 |
| TimeStream[61] | 多数据流 $\theta$ 连接 | 弹性替代和依赖追踪 | 阻塞的批次处理、非阻塞的元组处理 | 优点:利用元组依赖性,确保连接可靠性<br>缺点:连接状态的备份导致过高的通信开销 |
| ATR/CTR[62] | 多数据流 $\theta$ 连接 | 一步/多步扩散 | 非阻塞的元组处理 | 优点:根据数据特征实时自适应调度<br>缺点:基于窗口语义的操作,有窗口尺寸的限制 |
| PSP[63] | 多数据流 $\theta$ 连接 | 流水线状态维护 | 非阻塞的元组处理 | 优点:通过时间切片,构造环结构,将连接算子拆分成若干的子算子进行操作<br>缺点:连接状态的备份导致过高的通信开销,有窗口大小的限制 |
| DYNAMIC[64] | 多数据流 $\theta$ 连接 | join-matrix | 非阻塞的元组处理 | 优点:设计动态划分样式,最小化输入装载因子<br>缺点:数据备份量大 |
| JB[15] | 多数据流 $\theta$ 连接 | join-biclique | 非阻塞的元组处理 | 优点:数据备份量小<br>缺点:需要人工干预数据分组的参数设置 |

Photon[58] 是谷歌公司针对连接网络查询的数据流和用户点击广告的数据流而设计的,支持两种数据流的等值连接操作,通过中心协调器实现多数据中心的容错和扩展连接.其利用向中心协调器注册查询事件的方法和多数据中心的分布式架构确保数据的完备性,并采用非阻塞的方式实时处理新到来的每条元组.但 Photon 主要处理事件 ID 的等值连接,不支持 $\theta$ 连接操作.

D-Streams[59] 是 Spark Streaming 定义的数据流操作对象,可支持多数据流的 $\theta$ 连接.通过转换(transformations)操作,可利用底层 Spark① 提供的 RDD[60] 机制确保查询处理的正确性和容错性.

① http://spark-project.org/

D-Streams 是将若干数据流元组以阻塞的方式构造成一系列的微批次(mini-batch)进行处理.这种处理方式也导致了某些离散的 batch 存在丢失连接元组对的现象,从而仅可获得近似的查询结果.

TimeStream[61]设计的弹性替代(resilient substitution)和依赖追踪(dependency tracking)机制,确保了数据流计算的可信赖性.针对任意数据流的 $\theta$ 连接操作,提供了 MapReduce 风格的批处理和非阻塞的元组处理两种方式.但 TimeStream 需要维护连接状态的依赖信息,通信代价较大.

ATR/CTR[62]是应对多个窗口数据流 $\theta$ 连接操作设计的自适应负载扩散算法.这两个算法可实现保留语义的元组路由的功能,实现细粒度的负载均衡和连接操作的正确性.通过主从数据流切换的一步(one-hop)扩散法或基于路由表的多步(multi-hop)扩散法,可支持数据流元组至少被执行一次的语义保障.但 ATR/CTR 需要控制计算重复结果的开销,且只能处理基于窗口模式的数据流连接操作.

PSP[63]设计了基于时间的算子状态划分环以支持多个窗口数据流 $\theta$ 连接操作.流水线状态划分技术提供了每个连接子操作的状态信息,将连接算子拆分成若干子连接算子进行操作.但维护分布式的连接状态信息增加了系统的通信开销,且该操作受限于窗口模式下的数据流.

DYNAMIC[64]算子支持多个数据流的 $\theta$ 连接操作,使用基于矩阵的连接模型 join-matrix,冗余存储每个子处理单元的数据,并采用无窗口模式下的非阻塞元组处理方式以实时获取正确的查询结果.DYNAMIC 算子设计的重组器(reshuffler)可动态设计划分样式,确保最小化的数据输入装载因子(input-load factor).但该算子存储的数据冗余度较大.

JB[15]算子利用基于完全二部图概念构建的 join-biclique 模型,将集群分成两个部分.JB 算子可支持多个数据流的 $\theta$ 连接操作,并支持无窗口模式下的非阻塞元组处理方式,还可根据 $\theta$ 连接操作的负载程度动态扩展处理单元的数量.相对于 DYNAMIC 算子使用的 join-matrix 模型来讲,JB 算子使用的 join-biclique 模型大幅降低了数据备份的冗余度,提高了资源利用率.但在数据分组时 JB 算子利用元组内容敏感性的混合路由策略,需要人工干预数据分组的参数设置.

### 3.2.6　调度管理

分布式数据流查询系统中,一般需要根据查询算子的缓冲队列长度、查询代价和选择率等方面考虑系统的调度策略.如表 5 所示,调度优化策略可分为基于算子的调度、基于数据的调度和混合调度策略等.

**表 5　调度策略技术**

| 调度策略 | 代表技术 | 调度方式 |
|---|---|---|
| 基于算子的调度策略 | Aurora[5] | 指定查询的 QoS 指标[65],动态确定算子的执行顺序 |
| | Eddy[66] | 通过路由调度,重排算子关系并更改查询计划 |
| | Chain[67] | 根据运行时最小化内存的需求,更改算子调度策略 |
| 基于数据的调度策略 | Juggle[68] | 根据关注元组内容信息,将已扫描的元组重新排序 |
| | Borealis[69] | 根据元组的 QoS 值和 QoS 梯度,确定元组的执行顺序 |
| 基于算子和数据的混合调度 | JuggleEddy[70] | 根据元组的重要性和算子的处理时延,重排序元组和算子关系 |

基于算子的调度策略往往通过在主控节点制定服务质量(QoS)指标[65]、路由调度算法[66]和监控内存消耗情况[67],以更改相应的算子执行顺序和对应的查询计划.基于数据的调度策略,以 Juggle[68]和 Borealis[69]为代表技术.Juggle 针对感兴趣的元组内容信息,设计了重排序(reorder)算子,将已扫描的元组重新排序.Borealis 在 Aurora[5]的基础上,计算每个盒处理器的平均服务质量(QoS)梯度值,用于重排元组的执行顺序.JuggleEddy[70]是基于算子和数据的混合调度策略的代表,其根据在线查询处理产生的部分查询结果,通过计算相关元组的重要性和对应算子的处理时延,利用 juggle 对同一数据流中的中间元组进行重新排序.并结合 Eddy 的原理,通过路由调度将对应算子进行重新排序,以达到调度优化的目的.

### 3.2.7　并行管理

为实时处理大量数据流,提高整个分布式系统的处理能力和吞吐量,需划分查询任务和设定处理单元的查询粒度.

针对数据流处理的并发性来讲,存在两种划分机制.一种是划分查询执行计划;另一种是划分输入的数据流.

文献[71]是基于 Aurora 系统实现的划分查询计划的分布式划分策略.但因为存在某些占用资源较多和耗时较长的算子,导致无法生成有效的执行

计划,因此划分查询执行计划的方法使用较少.较常见的解决方法是针对不同的应用需求,对某些负载过重的算子进行划分,如文献[64]设计的 DYNAMIC 算子,可对算子内部进行实时自适应的划分,确保最少的状态迁移开销.

基于输入数据流划分的策略是应用较多的划分方法,其一般可分为静态划分和动态划分两种.静态划分是指在查询编译时确定具体的划分码(如图 3(a)所示);动态划分是在查询执行过程中根据数据流的分布和关联情况动态调整划分码(如图 3(b)所示),减少了网络开销,提高了查询效率. AT&T 实验室首先提出了感知查询的划分方法[72]:针对具有聚集和连接操作的多查询来讲,根据每个查询的划分码,求出向上兼容的最大划分码集合.基于代价模型,通过转换规则将逻辑查询计划转换成最优的物理查询计划.文献[73]基于文献[72]设计的方法,提出了挖掘划分码间时态近似依赖性(Temporal Approximate Dependencies,TADs)的概念.为避免运行时的二次划分和最小化额外的通信开销,将满足 TADs 规则的范式进行规约,实现对划分码动态合并的优化处理.文献[74]设计了感知数据关联性的多查询优化器 CMR.在运行时,根据分析和判断每个数据流内部的倾斜一致性(skewed uniformity),动态划分各个数据流;再根据查询的聚集或连接码选取各个被划分子数据流的聚类特征,利用 BIRCH[75] 方法进行层次聚类,以获得最终的划分区间.

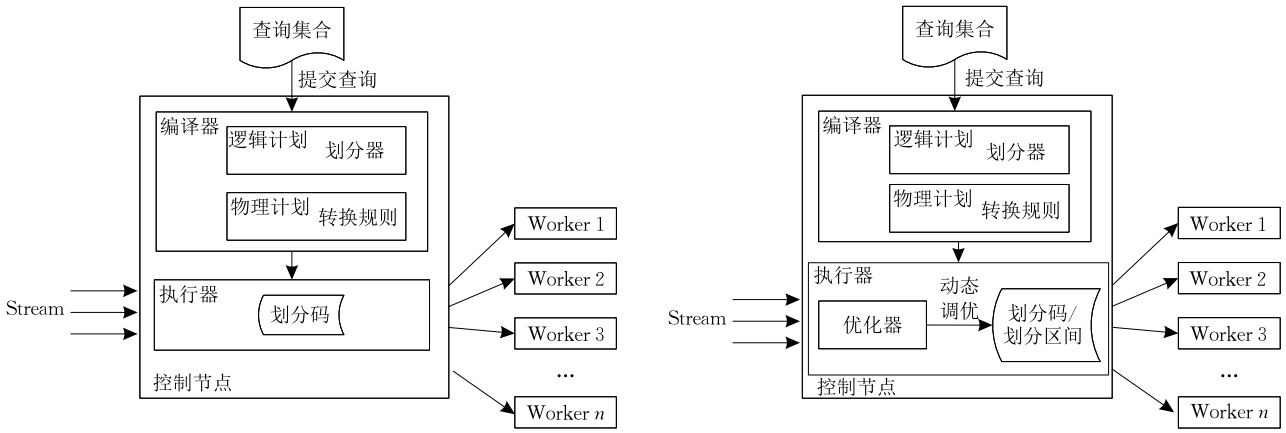


图 3 数据划分策略

划分后的数据通过处理系统计算时,根据数据量的多少和计算的复杂度,优化算子内部的并行度,并对各个处理节点的负载进行动态调整.文献[76]以 Storm 为实验平台,根据有向无环图(DAG)中不同 bolt 的处理复杂度,动态设定其并行处理粒度,以提升查询系统的性能.

## 4 分布式数据流查询系统介绍

为提升分布式数据流处理系统的简洁性、易用性和扩展性等特点,工业界和学术界相继推出了支持不同处理系统的查询系统.本节选择有代表性的分布式查询系统实例 SPL[77]、StreamingSQL①、Squall② 和 DBToaster[45] 进行对比分析.分析内容包括底层处理系统的简介,查询系统的实现原理,以及查询系统的相关特征等.

### 4.1 SPL 查询系统

SPL[77] 是由 IBM 的声明式系统 SPADE[78] 演化成的商业软件,构建在支持处理高速数据流的 InfoSphere Streams③ 之上.作为商业产品的代表,SPL 不仅具有丰富的系统内置算子,而且可较好支持用户自定义函数(UDF)的功能.

#### 4.1.1 InfoSphere Streams 简介

InfoSphere Streams 广泛应用于电信、金融和医疗等多个领域,可快速提取、分析和关联来自多个数据源的信息,并可处理结构化、半结构化和非结构化的不同数据类型. InfoSphere Streams 根据多个连续查询构建数据流图(data-flow graph),用于支持高效的分布式数据流处理框架.

#### 4.1.2 SPL 实现原理

SPL 支持关系型数据流,可定义嵌套的元组类型和带环的拓扑结构.利用算子和数据流的概念,SPL 为每个应用程序产生一个数据流图,图中的每

① https://github.com/Intel-bigdata/spark-streamingsql
② https://github.com/epfldata/squall
③ http://www.ibm.com/software/data/infosphere/streams/

个顶点对应于后台的算子代码. 算子具有声明式语义信息，便于用户描述具体的应用需求. SPL 提供用本地语言(C++或 Java)定义的原始算子(包括系统内置算子和用户自定义算子)，以及包含可重用数据流子图的复合算子[79]. SPL 利用非阻塞的元组处理以及基于时间和元组的滑动窗口，实现了窗口内部数据的实时处理.

图 4[77]说明了SPL处理原始算子的流程. 每个原始算子由算子模式(operator model)和算子代码生成器(operator code generator)两部分组成. 其中，算子模式是一个描述算子到 SPL 编译器(SPL Complier)的 XML 文档；算子代码生成器负责为特定算子生成相应代码. 在编译应用源码(application source code)时，SPL 会产生多个算子调用，每个算子调用均需要检查算子模式并产生相应的算子实例模式(operator instance model)，并利用算子代码生成器产生算子实例(operator instance). 在运行时，InfoSphere Streams 根据已定义的应用模式(application model)触发算子并产生数据流图，利用流平台(streaming platform) InfoSphere Streams 的处理单元(Processing Elements，PEs)对数据流进行分析处理. 每个算子实例是一个转换器，根据流平台的运行环境处理输入数据项流(stream of input data items)，并产生输出数据项流(stream of output data items).

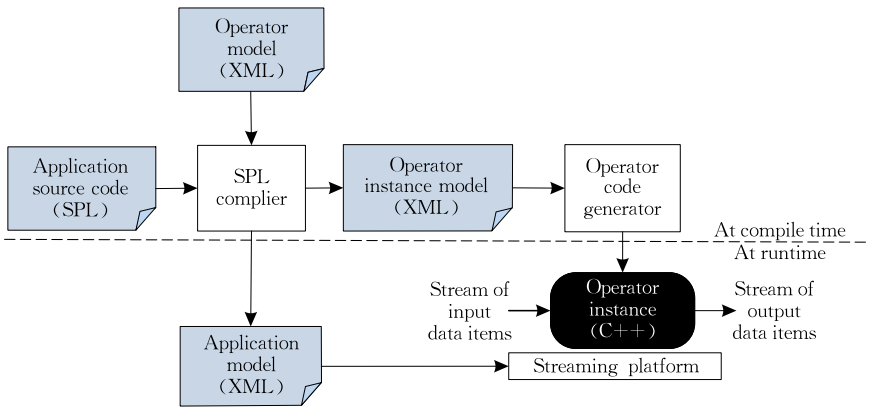


图 4　算子处理流程

4.1.3　小　结

SPL 查询系统具有的特征包括：(1) SPL 通过编程式语言描述数据流处理过程，不受限于数据流图的描述形式和中间语言的转换；(2) SPL 可生成便于组合的并行或分布式数据流图的中间语言，并支持用户自定义的数据类型和算子；(3) SPL 支持代码的错误检查，利用编译器搜集的信息进行优化，并支持 C++或 Java 代码的转换.

SPL 查询系统的不足之处在于：用户需要掌握 SPL 提供的不同算子进行编程实现，缺少抽象语义的描述性信息，不如 SQL 类型的查询系统易于使用.

## 4.2　StreamingSQL 查询系统

StreamingSQL是Intel公司基于Spark Streaming① 和结构化数据查询框架 Catalyst② 实现的数据流查询系统.

4.2.1　Spark Streaming 简介

由于 Spark 只支持数据的批量处理模式而不支持流式处理模式，伯克利大学设计了 Spark Streaming 以满足对数据流的处理. Spark Streaming 定义了用于描述数据流计算的操作对象 D-Streams[59]，并将输入数据流切分成 Spark 可以处理的若干 RDD[60]. 这使得 Spark Streaming 对 D-Streams 的操作可转换为 Spark 对 RDD 的操作，从而实现了数据流的批量处理.

4.2.2　StreamingSQL 实现原理

StreamingSQL 使用类 SQL 声明式语言调度和操作 Spark Streaming 的任务，利用 Scala 语言实现系统的核心功能. StreamingSQL 的定义参照了 SQLstream 公司的 Streaming SQL 标准③，并对 HiveQL 进行了扩展. StreamingSQL 类似于操作数据库中的结构化数据来处理数据流，并构建支持连续查询的模块. StreamingSQL 支持基于时间的滑动窗口和无窗口两种驱动方式，分别利用阻塞的窗口处理和阻塞的微批次处理实现聚集和连接操作.

StreamingSQL 基于修改 Hive 的解析器设计自身语言并增加了关于数据流的语义操作；为易于和 Hive 集成，共享了 Hive 的 Metastore[80]. StreamingSQL 的设计复用了 Spark 原始的逻辑计划和相应的分析规则，通过构造包装器的方法进行

① https://spark.apache.org/streaming/
② http://spark-summit.org/wp-content/uploads/2013/10/J-Michael-Armburst-catalyst-spark-summit-dec-2013.pptx
③ http://www.sqlstream.com/docs/

物理计划树的转换. StreamingSQL 的处理流程如图 5 所示.

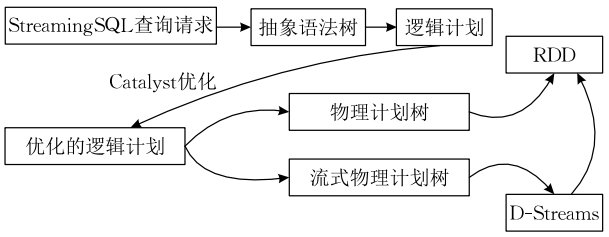


图 5 StreamingSQL 处理流程

StreamingSQL 利用 Hive 解析的抽象语法树(Abstract Syntax Tree,AST),生成对应的逻辑计划,并通过 Catalyst 进行优化,最终生成分别依赖于 RDD 和 D-Streams 的不同的物理计划树. StreamingSQL 通过从抽象语法树叶子节点的 D-Streams 中提取 RDD,可将流式物理计划树转换成 Spark 的物理计划树,以获取最终的查询结果.

4.2.3 小 结

StreamingSQL 查询系统具有的特征包括:(1)支持多种查询驱动方式;(2)可在同一个查询中进行表和数据流的交互式操作.

StreamingSQL 查询系统的不足之处在于:(1)不支持嵌套等复杂查询;(2)不支持基于元组的滑动窗口操作.

### 4.3 Squall 查询系统

Squall① 是由洛桑联邦理工学院数据实验室开发的基于 Storm②[11] 的分布式在线查询系统,可利用 SQL 查询语句实现对数据流的实时分析处理.

4.3.1 Storm 简介

Storm 是 Twitter 开源的实时数据流处理框架,具有简单的编程模型,可利用不同的通信协议支持多种编程语言,并通过 acker 机制实现可靠的消息处理. Storm 集群架构如图 6 所示. Storm 由一个 nimbus 节点和若干 supervisor 节点组成. 其中,nimbus 节点负责资源分配、任务调度和监控各个 supervisor 节点的运行情况;各个 supervisor 节点负责接受 nimbus 分配的任务和管理该节点下的各个工作进程 worker. 整个 Storm 集群利用 ZooKeeper[24] 提供分布式应用的协调管理.

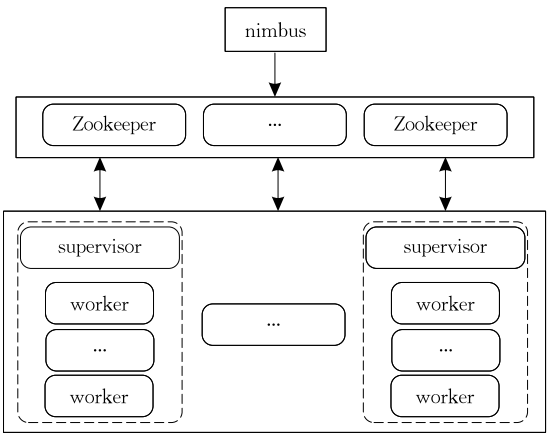


图 6 Storm 组成架构

4.3.2 Squall 实现原理

Squall 将 SQL 查询翻译成由执行算子构成的有向无环图(DAG)的查询计划,用 Java 语言编写了查询系统. 每个算子对应于 Storm 的一个组件(spout 或 bolt),并通过构建 Storm 的拓扑结构(topology)执行查询计划.

Squall 采用基于矩阵的连接模型 join-matrix 适应数据流不断增长的应用场景,支持无窗口驱动方式下全部历史数据的 $\theta$ 连接处理,并实现了基于非阻塞元组处理的 DYNAMIC 连接算子[64]. Squall 将连接操作分解成连接器(joiner)和重组器(reshuffler)两部分. 其中,连接器负责实际连接计算,重组器负责分发元组和路由元组. 重组器负责监控全部历史数据的统计信息和支持自适应调整的样式划分. 这样的组合设计确保了数据流连接算子的负载均衡,分摊了输入元组的处理代价,并利用重组器对内容的不敏感性来抵御数据倾斜的问题.

4.3.3 小 结

Squall 查询系统具有的特征包括:(1)Squall 实现了在线数据流连接算子,支持任意谓词的连接操作. 并且可根据数据的动态变化对算子进行连续调整和状态的重划分;(2)Squall 设计了带状态的在线连接算子,在缺乏数据流统计信息的情况下可自适应的处理连接操作;(3)Squall 提供了一种位置感知的迁移机制,确保了最小的状态迁移开销.

Squall 查询系统的不足之处在于:(1)Squall 目前只支持简单的连接查询操作,不支持 IN、BETWEEN、LIKE、EXISTS、HAVING 等 SQL 子句;(2)连接条件不支持 OR 表达式,也不支持嵌套查询和过程查询.

### 4.4 DBToaster 查询系统

DBToaster[45] 是洛桑联邦理工学院数据实验室开发的新型数据流查询系统,提供从 SQL 查询语句到生成本地代码的编译框架.

① https://github.com/epfldata/squall
② http://storm.apache.org/

4.4.1　处理系统简介

根据不同的业务需求，可将 DBToaster 部署在不同的处理系统上. 若需保证数据流的并行处理能力和处理效率，可使用 Storm 作为底层处理系统(详见参见 4.3.1 节)；若有大量联机分析处理(Online Analytical Processing，OLAP)的操作，可使用 Cumulus[81]作为底层处理系统. 本节简要介绍 Cumulus 处理系统.

Cumulus 是洛桑联邦理工学院和图卢兹大学联合开发的可信云系统，Cumulus 提出了一种新的拜占庭容错(Byzantine Fault Tolerance，BFT)[82]协议 CBFT，可保证机器之间独立且无感知的数据备份，为 DBToaster 的视图维护操作提供了可靠的信息传输和实时的精准聚集结果①.

4.4.2　DBToaster 实现原理

DBToaster 通过递归的思想将增量查询转换为 C++或 Scala 函数，利用敏捷视图(agile view)[46]保持实时更新，省略了多余的扫描和连接操作，并消除了解释查询计划的冗余开销. DBToaster 还可以将用于维护敏捷视图的程序嵌入到用户的应用程序中，根据支持数据流管理系统中物化视图[46]的机制，确保将更新视图的最新查询结果直接返回给用户. DBToaster 采用无窗口的非阻塞元组处理方式，实现全部历史数据的实时增量计算.

4.4.3　小　结

DBToaster 查询系统具有的特征包括：(1) DBToaster 通过 LLVM[49]和 LMS[54]编译技术直接生成目标代码，消除了解析器所需的时间成本，提高了查询性能；(2) DBToaster 通过物化视图的操作，快速处理大量更新的数据流，而且打破了基于窗口的数据流操作方法，从而可以访问内存中的全部历史数据；(3) 为了支持复杂的聚集操作，DBToaster 构建了支持增量处理的数据库环[83]. 用环作为设计查询计算的基础，可保证增量计算的封闭性，同时简化了处理机制；(4) DBToaster 利用有限差的方法，提出了小视图变换(viewlet transforms)的概念[84]，可将高阶的增量查询集合物化成视图，降低视图维护的代价. 并通过启发式和基于代价的优化框架，保证视图的频繁更新.

DBToaster 查询系统的不足之处在于：(1) 不支持排序(order by)和外连接(outer join)等查询操作；(2) 内存溢出时需进行内外存的交互处理，查询的响应时间增长；(3) 进行多查询处理时，会出现敏捷视图的存储冲突和版本不一致等问题.

### 4.5　查询系统对比分析

根据以上数据流查询系统的分析说明，可以看出：针对数据流处理的不同需求，需采用不同的查询系统和处理方法. 各个查询系统的特点分析如表 6 所示.

**表 6　分布式数据流查询系统对比分析**

| 对比指标 | SPL | StreamingSQL | Squall | DBToaster |
|---|---|---|---|---|
| 处理系统 | InfoSphere Streams | Spark | Storm | Storm/Cumulus |
| 抽象语言 | 编程式语言 | SQL 类型语言 | SQL 类型语言 | SQL 类型语言 |
| 处理方式 | 非阻塞的元组处理 | 阻塞的微批次处理 | 非阻塞的元组处理 | 非阻塞的元组处理 |
| 查询驱动方式 | 基于时间和元组的滑动窗口 | 基于时间的滑动窗口<br>无窗口 | 无窗口 | 无窗口 |
| 开发语言 | C++或 Java | Scala | Java | C++或 Scala |

## 5　挑战与展望

分布式数据流查询系统为满足查询速度快、查询精度高且功能完备等特点，依然存在着很多有价值的研究问题.

### 5.1　分布式数据流查询优化技术

为满足实时性的查询要求，分布式数据流查询处理的优化技术需不断完善. 相关技术可归纳为如下几点：

(1) 在查询优化的过程中，为适应数据流波动、有损、易失等特点，需制定流查询系统的优化规则，并协调规则的执行顺序，进一步完善基于规则的优化策略.

(2) 针对无窗口驱动方式的连续查询，由于数据分布和执行代价等信息的动态变化，查询系统需要实时调整查询计划. 例如，在 Shark 系统中引入部分有向无环图执行策略(Partial DAG Execution，PDE)[85]的概念，利用在查询处理中实时收集到的统计信息动态改变查询计划. 为满足自适应查询处理的需求，宾夕法尼亚大学的学者们设计了基于代价模型的二次优化框架[86]，可根据代价信息增量更新查询优化器. 如何在运行时自适应地寻求最小代价的查询计划，成为近年来一个热门的研究问题.

① http://webdam.inria.fr/wordpress/wp-content/uploads/2013/10/christoph.ppt

(3)针对用户提交的复杂查询(如多查询和嵌套查询等),对查询重写、算子共享等数据库管理系统中关注的优化问题,在数据流处理场景中也有很多研究工作要做.

### 5.2 分布式数据流查询执行策略

基于分布式环境下的数据流查询处理框架,流查询系统将各个查询处理计划以流水线的形式组织在一起.在运行时,需重点关注算子和数据的调度执行策略,存在的挑战性问题分析如下:

(1)执行需保持状态信息的算子时,要设计算子调度和执行的策略,以最低的代价存储和传输中间状态信息.

(2)针对用户提交的多查询请求,需引入数据流和查询划分码的相关性评测标准.根据不同的查询驱动方式和数据分布情况,制定综合的划分策略作为数据分组的处理依据.

(3)根据分布式集群的部署情况和用户提交的查询请求,随着数据流分布特征的不断变化,需设计出合理且高效的最优并行度分裂算法(包括数据流的划分和复制),尽可能提高查询系统的并行处理能力.

### 5.3 分布式数据流实时精准查询

分布式数据流处理系统在可伸缩性、容错性、状态一致性管理和数据细粒度处理等方面不断提升和发展,系统处理大量无损数据流的能力得以完善.因此,学者们将研究的重点从丢弃部分元组的降载[87]方式转向处理全部历史数据的实时精准查询.其中面临的研究点包括:分布式环境下的本地增量视图和全局增量视图的统一与维护;针对多查询应用场景下的视图一致性管理和共享算子的状态管理等.针对这些问题的研究,对提高系统的查询速度和查询精度均具有十分重要的意义.

### 5.4 分布式数据流复杂查询分析

为满足以用户自定义函数(UDF)、嵌套查询等为代表的复杂查询分析,分布式数据流查询系统需构建实时高效的以有向无环图(DAG)为拓扑结构的任务执行计划.因此,为了充分发挥系统性能,如何优化有向无环图(DAG)的层次划分,如何利用重写规则的模板推断语义等价的查询执行计划,是需要重点考虑的问题.另外,针对数据流的动态和不确定性,如何更好地设计在线计算模型以支持复杂查询的处理能力,也是值得研究的问题.并且,如何利用增量聚集算法和增量学习框架[88-89]支持无窗口驱动方式的用户自定义函数(UDF),以及如何优化同时处理多个用户自定义函数(UDF)的执行计划问题,也值得我们进一步深入研究.

### 5.5 其他研究点

对于制定流查询系统的相关标准和增强数据的处理能力方面,仍有许多其他的研究课题.例如,为应对不同模式定义的异构数据源,查询系统需根据用户需求的变化支持不同的查询请求.参照文献[90]的思想和方法,可制定统一的流查询系统标准和算子转换规则,以适应不同的数据流处理系统,使其具有更好的参照性、完备性和通用性.此外,为支持Lambda架构①,可设计同时进行批量处理和流式处理的查询计划适配器,通过统一的查询系统接口,不仅可以实时获取当前数据流的查询结果,而且可以从离线的历史数据中获取相应的查询结果.

## 6 结 论

随着大数据概念的兴起,分布式数据流处理技术已成为新型数据处理技术的一个重要分支.基于流处理系统的查询技术成为研究热点,受到了学术界和工业界的共同关注.查询技术需要对相关概念、理论、方法和体系进行深入研究.本文主要归纳总结了近年来分布式数据流查询技术的若干关键特性,并对比了4种有代表性的查询系统.同时指出了存在的挑战性问题.这些问题的研究还处于起步阶段,均没有较为成熟的、可以进行大规模实际应用的成果和方案.因此,这些课题具有重要的研究价值和意义.

## 参 考 文 献

[1] Li Guo-Jie, Cheng Xue-Qi. Research status and scientific thinking of big data. Bulletin of Chinese Academy of Sciences, 2012, 27(6): 647-657(in Chinese)
(李国杰,程学旗. 大数据研究:未来科技及经济社会发展的重大战略领域——大数据的研究现状与科学思考. 中国科学院院刊, 2012, 27(6): 647-657)

[2] Wang Yuan-Zhuo, Jin Xiao-Long, Cheng Xue-Qi. Network big data: Present and future. Chinese Journal of Computers, 2013, 36(6): 1125-1138(in Chinese)
(王元卓,靳小龙,程学旗. 网络大数据:现状与展望. 计算机学报, 2013, 36(6): 1125-1138)

[3] Meng Xiao-Feng, Ci Xiang. Big data management: Concepts, techniques and challenges. Journal of Computer Research and Development, 2013, 50(1): 146-169(in Chinese)

① http://lambda-architecture.net/

(孟小峰，慈祥. 大数据管理：概念、技术与挑战. 计算机研究与发展，2013，50(1)：146-169)

[4] Big data：Science in the petabyte era. Nature，2008，455(7209)：1-136

[5] Carney D，Cetintemel U，Cherniack M，et al. Monitoring streams—A new class of data management applications//Proceedings of the 28th International Conference on Very Large Data Bases (VLDB2002). Hong Kong，China，2002：215-226

[6] Chandrasekaran S，Cooper O，Deshpande A，et al. TelegraphCQ：Continuous dataflow processing for an uncertain world//Proceedings of the 1st Biennial Conference on Innovative Data Systems Research(CIDR 2003). Asilomar，USA，2003：269-280

[7] Arasu A，Babcock B，Babu S，et al. STREAM：The stanford stream data manager. IEEE Data Engineering Bulletin，2003，26(1)：19-26

[8] Dean J，Ghemawat S. MapReduce：Simplified data processing on large clusters//Proceedings of the 6th Symposium on Operating System Design and Implementation(OSDI 2004). San Francisco，USA，2004：137-150

[9] Li Feng，Ooi B C，Özsu M T，Wu S. Distributed data management using MapReduce. ACM Computing Surveys，2014，46(3)：31:1-31:42

[10] Neumeyer L，Robbins B，Nair A，Kesari A. S4：Distributed stream computing platform//Proceedings of the 2010 Industrial Conference on Data Mining Workshops (ICDM2010). Berlin，Germany，2010：170-177

[11] Toshniwal A，Taneja S，Shukla A，et al. Storm@Twitter//Proceedings of the 2014 International Conference on Management of Data (SIGMOD 2014). Snowbird，USA，2014：147-156

[12] Sun Da-Wei，Zhang Guang-Yan，Zheng Wei-Min. Big data stream computing：Technologies and instances. Journal of Software，2014，25(4)：839-862(in Chinese)
(孙大为，张广艳，郑纬民. 大数据流式计算：关键技术及系统实例. 软件学报，2014，25(4)：839-862)

[13] Cui Xing-Can，Yu Xiao-Hui，Liu Yang，Lv Zhao-Yang. Distributed stream processing：A survey. Journal of Computer Research and Development，2015，52(2)：318-332 (in Chinese)
(崔星灿，禹晓辉，刘洋，吕朝阳. 分布式流处理技术综述. 计算机研究与发展，2015，52(2)：318-332)

[14] Zhang H，Chen G，Ooi B C，et al. In-memory big data management and processing：A survey. IEEE Transactions on Knowledge and Data Engineering，2015，27(7)：1920-1948

[15] Lin Q，Ooi B C，Wang Z，Yu C. Scalable distributed stream join processing//Proceedings of the 2015 International Conference on Management of Data (SIGMOD 2015). Melbourne，Australia，2015：811-825

[16] Huang Y，Cui B，Zhang W，Xu Y. TencentRec：Real-time stream recommendation in practice//Proceedings of the 2015 International Conference on Management of Data (SIGMOD 2015). Melbourne，Australia，2015：227-238

[17] Bonnet P，Gehrke J，Seshadri P. Towards sensor database systems//Proceedings of the 2nd International Conference on Mobile Data Management(MDM2001). Hong Kong，China，2001：3-14

[18] Sullivan M，Heybey A. Tribeca：A system for managing large databases of network traffic//Proceedings of the USENIX Annual Technical Conference(NO 98). New Orleans，Louisiana，1998：13-24

[19] Thusoo A，Saram J S，Jain N，et al. Hive—A petabyte scale data warehouse using Hadoop//Proceedings of the 26th IEEE International Conference on Data Engineering (ICDE2010). California，USA，2010：996-1005

[20] Gounaris A，Tsamoura E，Manolopoulos Y. Adaptive query processing in distributed settings. Advanced Query Processing，2013：211-236

[21] Hwang J，Balazinska M，Rasin A，et al. High-availability algorithms for distributed stream processing//Proceedings of the 21th International Conference on Data Engineering (ICDE 2005). Tokyo，Japan，2005：779-790

[22] Balazinska M，Balakrishnan H，Madden S，Stonebraker M. Fault-tolerance in the Borealis distributed stream processing system//Proceedings of the 2005 International Conference on Management of Data (SIGMOD 2005). Baltimore，USA，2005：13-24

[23] Chandramouli B，Goldstein J，Barnett M，et al. Trill：A high-performance incremental query processor for diverse analytics//Proceedings of the 40th International Conference on Very Large Databases (VLDB2014). Hangzhou，China，2014：401-412

[24] Hunt P，Konar M，Junqueira F P，et al. ZooKeeper：Wait-free coordination for Internet-scale systems//Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference. Boston，USA，2010：11-11

[25] Wang H，Zaniolo C. User defined aggregates in object-relational systems//Proceedings of the 16th International Conference on Data Engineering (ICDE2000). San Diego，USA，2000：135-144

[26] Tran T，Diao Y，Sutton C A，Liu A. Supporting user-defined functions on uncertain data//Proceedings of the 39th International Conference on Very Large Databases(VLDB2013). Riva del Grada，Trento，2013：469-480

[27] O'Hagan A. Bayesian analysis of computer code outputs：A tutorial. Reliability Engineering & System Safety，2006，91(10-11)：1290-1300

[28] Zhang J，Zhou H，Chen R，et al. Optimizing data shuffling in data-parallel computation by understanding user-defined functions//Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation(NSDI2012). San Jose，USA，2012：295-308

[29] Cohen S. User-defined aggregate functions：Bridging theory and practice//Proceedings of the 2006 International Conference on Management of Data (SIGMOD2006). Chicago，USA，2006：49-60

[30] Simhadri V, Ramachandra K, Chaitanya A, et al. Decorrelation of user defined function invocations in queries//Proceedings of the 30th International Conference on Data Engineering (ICDE2014). Chicago, USA, 2014: 532-543

[31] Hong M, Riedwald M, Koch C, et al. Rule-based multi-query optimization//Proceedings of the 12th International Conference on Extending Database Technology (EDBT2009). Saint-Petersburg, Russia, 2009: 120-131

[32] Demers A J, Gehrke J, Panda B, et al. Cayuga: A general purpose event monitoring system//Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research (CIDR2007). Asilomar, USA, 2007: 412-422

[33] Liu Y, Plale B. Query optimization for distributed data streams//Proceedings of the 15th International Conference on Software Engineering and Data Engineering (SEDE2006). Los Angeles, USA, 2006: 259-266

[34] Viglas S, Naughton J F. Rate-based query optimization for streaming information sources//Proceedings of the 2002 International Conference on Management of Data (SIGMOD2002). Madison, USA, 2002: 37-48

[35] Heinz C, Krämer J, Riemenschneider T, Seeger B. Toward simulation-based optimization in data stream management systems//Proceedings of the 24th International Conference on Data Engineering (ICDE2008). Cancun, Mexico, 2008: 1580-1583

[36] Park H K, Lee W S. Adaptive optimization for multiple continuous queries. Data & Knowledge Engineering, 2012, 71(1): 29-46

[37] Arasu A, Babu S, Widom J. The CQL continuous query language: Semantic foundations and query execution. The VLDB Journal, 2006, 15(2): 121-142

[38] Seshadri P, Livny M, Ramakrishnan R. SEQ: A model for sequence databases//Proceedings of the 11th International Conference on Data Engineering (ICDE 1995). Taipei, China, 1995: 232-239

[39] Chen Y, Tu L. Density-based clustering for real-time stream data//Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2007). San Jose, USA, 2007: 133-142

[40] Zhong S. Efficient streaming text clustering. Neural Networks, 2005, 18(5-6): 790-798

[41] Chu W, Zinkevich M, Li L, et al. Unbiased online active learning in data streams//Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2011). San Diego, USA, 2011: 195-203

[42] Cohen E, Strauss M. Maintaining time-decaying stream aggregates//Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS2003). San Diego, USA, 2003: 223-233

[43] Gilbert A, Kotidis Y, Muthukrishnan S, Strauss M. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries//Proceedings of the 27th International Conference on Very Large Data Bases (VLDB2001). Roma, Italy, 2001: 79-88

[44] Cormode G, Shkapenyuk V, Srivastava D, Xu B. Forward decay: A practical time decay model for streaming systems//Proceedings of the 25th International Conference on Data Engineering (ICDE2009). Shanghai, China, 2009: 138-149

[45] Ahmad Y, Koch C. DBToaster: A SQL compiler for high-performance delta processing in main-memory databases//Proceedings of the 35th International Conference on Very Large Databases (VLDB2009). Lyon, France, 2009: 1566-1569

[46] Kennedy O, Ahmad Y, Koch C. DBToaster: Agile views in a dynamic data management system//Proceedings of the 5th Biennial Conference on Innovative Data Systems Research (CIDR2011). Asilomar, USA, 2011: 284-295

[47] Boncz P A, Zukowski M, Nes N. MonetDB/X100: Hyper-pipelining query execution//Proceedings of the 2nd Biennial Conference on Innovative Data Systems Research (CIDR2005). Asilomar, USA, 2005: 225-237

[48] Neumann T. Efficiently compiling efficient query plans for modern hardware//Proceedings of the 37th International Conference on Very Large Databases (VLDB2011). Seattle, USA, 2011: 539-550

[49] Lattner C, Adve V S. LLVM: A compilation framework for lifelong program analysis & transformation//Proceedings of the 2nd International Symposium on Code Generation and Optimization (CGO2004). San Jose, USA, 2004: 75-88

[50] Willhalm T, Popovici N, Boshmaf Y, et al. SIMD-Scan: Ultra-fast in-memory table scan using on-chip vector processing units//Proceedings of the 35th International Conference on Very Large Databases (VLDB2009). Lyon, France, 2009: 385-394

[51] Pirk H, Funke F, Grund M, et al. CPU and cache efficient management of memory-resident databases//Proceedings of the 29th IEEE International Conference on Data Engineering (ICDE2013). Brisbane, Australia, 2013: 14-25

[52] Grund M, Krüger J, Plattner H, et al. HYRISE—A main memory hybrid storage engine//Proceedings of the 36th International Conference on Very Large Databases (VLDB2010). Singapore, 2010: 105-116

[53] Klonatos Y, Koch C, Rompf T, Chafi H. Building efficient query engines in a high-level language//Proceedings of the 40th International Conference on Very Large Databases (VLDB2014). Hangzhou, China, 2014: 853-864

[54] Rompf T, Odersky M. Lightweight modular staging: A pragmatic approach to runtime code generation and compiled DSLs//Proceedings of the 9th International Conference on Generative Programming and Component Engineering (GPCE2010). Eindhoven, The Netherlands, 2010: 127-136

[55] Fernandez R C, Migliavacca M, Kalyvianaki E, Pietzuch P. Integrating scale out and fault tolerance in stream processing using operator state management//Proceedings of the 2013 International Conference on Management of Data (SIGMOD2013). New York, USA, 2013: 725-736

[56] Pietzuch P, Ledlie J, Shneidman J, et al. Network-aware operator placement for stream-processing systems//Proceedings

of the 22nd International Conference on Data Engineering (ICDE 2006). Atlanta, USA, 2006: 49-60

[57] Papaemmanouil O, Basu S, Banerjee S. Adaptive in-network query deployment for shared stream processing environments //Proceedings of the 24th International Conference on Data Engineering Workshops (ICDE 2008). Cancun, USA, 2008: 206-211

[58] Ananthanarayanan R, Basker V, Das S, et al. Photon: Fault-tolerant and scalable joining of continuous data streams //Proceedings of the 2013 International Conference on Management of Data (SIGMOD2013). New York, USA, 2013: 577-588

[59] Zaharia A, Das T, Li H, et al. Discretized streams: Fault-tolerant streaming computation at scale//Proceedings of the 24th Symposium on Operating Systems Principles (SOSP2013). Farmington, USA, 2013: 423-438

[60] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing//Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI2012). San Jose, USA, 2012: 15-28

[61] Qian Z, He Y, Su C, et al. TimeStream: Reliable stream computation in the cloud//Proceedings of the 8th Eurosys Conference (EuroSys13). Prague, Czech Republic, 2013: 1-14

[62] Gu X, Yu P S, Wang H. Adaptive load diffusion for multiway windowed stream joins//Proceedings of the 23rd International Conference on Data Engineering (ICDE 2007). Istanbul, Turkey, 2007: 146-155

[63] Wang S, Rundensteiner E. Scalable stream join processing with expensive predicates: Workload distribution and adaptation by time-slicing//Proceedings of the 12th International Conference on Extending Database Technology (EDBT2009). Saint-Petersburg, Russia, 2009: 299-310

[64] Elseidy M, Elguindy A, Vitorovic A, Koch C. Scalable and adaptive online joins//Proceedings of the 40th International Conference on Very Large Databases(VLDB2014). Hangzhou, China, 2014: 441-452

[65] Carney D, Cetintemel U, Rasin A, et al. Operator scheduling in a data stream manager//Proceedings of the 29th International Conference on Very Large Data Bases (VLDB2003). Berlin, Germany, 2003: 838-849

[66] Avnur R, Hellerstein J M. Eddies: Continuously adaptive query processing//Proceedings of the 2000 International Conference on Management of Data(SIGMOD2000). Dallas, Texas, 2000: 261-272

[67] Babcock B, Babu S, Datar M, Motwani R. Chain: Operator scheduling for memory minimization in data stream systems //Proceedings of the 2003 International Conference on Management of Data (SIGMOD2003). San Diego, USA, 2003: 253-264

[68] Raman V, Raman B, Hellerstein J M. Online dynamic reordering. The VLDB Journal, 2000, 9(3): 247-260

[69] Abadi D, Ahmad Y, Balazinska M, et al. The design of the borealis stream processing engine//Proceedings of the 2nd Biennial Conference on Innovative Data Systems Research (CIDR2005). Asilomar, USA, 2005: 277-289

[70] Raman V, Hellerstein J M. Partial results for online query processing//Proceedings of the 2002 International Conference on Management of Data (SIGMOD2002). Madison, USA, 2002: 275-286

[71] Cherniack M, Balakrishnan H, Balazinska M, et al. Scalable distributed stream processing//Proceedings of the 1st Biennial Conference on Innovative Data Systems Research (CIDR2003). Asilomar, USA, 2003: 257-268

[72] Johnson T, Muthukrishnan S M, Shkapenyuk V, Spatscheck O. Query-aware partitioning for monitoring massive network data streams//Proceedings of the 2008 International Conference on Management of Data(SIGMOD2008). Vancouver, Canada, 2008: 1135-1146

[73] Viel E, Ueda H. Data stream partitioning re-optimization based on runtime dependency mining//Proceedings of the 30th International Conference on Data Engineering Workshops (ICDE2014). Chicago, USA, 2014: 199-206

[74] Cao L, Rundensteiner E A. High performance stream query processing with correlation-aware partitioning//Proceedings of the 39th International Conference on Very Large Databases (VLDB2013). Riva del Grada, Trento, 2013: 265-276

[75] Zhang T, Ramakrishnan R, Livny M. BIRCH: An efficient data clustering method for very large databases//Proceedings of the 1996 International Conference on Management of Data (SIGMOD1996). Montreal, Canada, 1996: 103-114

[76] Sax M, Castellanos M, et al. Aeolus: An optimizer for distributed intra-node-parallel streaming systems//Proceedings of the 29th International Conference on Data Engineering (ICDE2013). Brisbane, Australia, 2013: 1280-1283

[77] Hirzel M, Andrade H, Gedik B, et al. IBM streams processing language: Analyzing big data in motion. IBM Journal of Research and Development, 2013, 57(3/4): 7

[78] Gedik B, Andrade H, Wu K, et al. SPADE: The system S declarative stream processing engine//Proceedings of the 2008 International Conference on Management of Data (SIGMOD2008). Vancouver, Canada, 2008: 1123-1134

[79] Hirzel M, Andrade H, Gedik B, et al. SPL stream processing language specification. New York: IBM Research Division T J. Watson Research Center, IBM Research Report: RC24897 (W0911-044), 2009

[80] Thusoo A, Saram J S, Jain N, et al. Hive—A petabyte scale data warehouse using Hadoop//Proceedings of the 26th IEEE International Conference on Data Engineering (ICDE2010). California, USA, 2010: 996-1005

[81] Guerraoui R, Yabandeh M, Shoker A, Bahsoun J P. Trustful cumulus clouds. Switzerland: EPFL, Technical Report: LPD-REPORT-2010-10, 2010

[82] Castro M, Liskov B. Practical byzantine fault tolerance and proactive recovery. ACM Transactions on Computer Systems, 2002, 20(4): 398-461

[83] Koch C. Incremental query evaluation in a ring of databases //Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS2010). Indianapolis, USA, 2010: 87-98

[84] Koch C, Ahmad Y, Kennedy O, et al. DBToaster: Higher-order delta processing for dynamic, frequently fresh views. The VLDB Journal, 2014, 23(2): 253-278

[85] Xin R S, Rosen J, Zaharia M, et al. Shark: SQL and rich analytics at scale//Proceedings of the 2013 International Conference on Management of Data (SIGMOD2013). New York, USA, 2013: 13-24

[86] Liu M, Ives Z G, Loo B T. Enabling incremental query re-optimization. University of Pennsylvania: Technical Report MS-CIS-11-11, 2011

[87] Tatbul N, Cetintemel U, Zdonik S B, et al. Load shedding in a data stream manager//Proceedings of the 29th International Conference on Very Large Data Bases (VLDB2003). Berlin, Germany, 2003: 309-320

[88] Gao W, Jin R, Zhu S, Zhou Z. One-pass AUC optimization //Proceedings of the 30th International Conference on Machine Learning (ICML2013). Atlanta, USA, 2013: 906-914

[89] He H, Chen S, Li K, Xu X. Incremental learning from stream data. IEEE Transactions on Neural Networks, 2011, 22(12): 1901-1914

[90] Jain N, Mishra S, Srinivasan A, et al. Towards a streaming SQL standard. Proceedings of the VLDB Endowment, 2008, 1(2): 1379-1390

**WANG Chun-Kai**, born in 1981, Ph. D. candidate. His main research interest is distributed data stream management.

**MENG Xiao-Feng**, born in 1964, Ph. D., professor, Ph. D. supervisor. His research interests include cloud data management, Web data management, flash-based databases, and privacy protection.

**Background**

Real-time processing of high-speed data stream has become one of the major areas of attention in industrial and academic circles. In recent years, there have been a number of distributed data stream processing systems represented by S4, Storm, Spark Streaming, etc. For improving the usability and processing capabilities of processing systems, we can build relational query systems on them. Query systems can define the SQL-like declarative languages to help users build the query requests and related applications.

This paper surveys the recent research work on distributed data stream relational query techniques. We first propose the framework of distributed data stream query systems. Based on the framework, we analyze the key techniques in several aspects: UDF query, query optimization, query-driven approaches, compiling techniques, operator management, scheduling management and parallel management. However, there are still many issues worthy to study. We point out the new challenges of query systems. We want to help researchers pay attention to the query techniques related issues that need to be addressed.

This work is partially supported by the grants from the National Natural Science Foundation of China (61379050, 91224008), the National High Technology Research and Development Program (863 Program) of China (2013AA013204), the Specialized Research Fund for the Doctoral Program of Higher Education (20130004130001), the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China (11XNL010).