基于云计算的流数据集成与服务

王桂玲[1] 韩燕波[1] 张仲妹[1),2)] 朱美玲[1),2)]

[1)](北方工业大学云计算研究中心大规模流数据集成与分析技术北京市重点实验室 北京 100144)

[2)](天津大学计算机科学与技术学院 天津 300072)

摘 要 当前,大数据的管理和处理是云基础设施的重点用武之地,而服务是落实云计算环境中各类资源及能力交付和使用模式的主要方式.随着感知设备的普及,系统规模急剧扩张,数据多元异构复杂性提升,流数据并发数量及速度剧增,传统的流数据系统在处理能力、可扩展性、容错性等方面面临瓶颈问题,而云计算技术依靠其良好的可伸缩性、数据的并行化处理能力、对服务使用模式的支持、容错性等特点,可作为流数据管理与处理的基础.基于云计算对来自不同类型设备的大规模流数据进行集成、处理及服务化正是文中关注的焦点所在.从应用需求出发,该文对大规模流数据集成和实时处理及服务的概念框架、集成方法、流数据查询处理、定制化服务、可伸缩性保障和可靠性保障以及相关评测基准等要点进行了剖析,归纳了大规模流数据的集成与服务研究面临的挑战,探讨了云计算环境下求解相关问题的思路.

关键词 流数据;云服务;数据服务;云数据集成

中图法分类号 TP391 **DOI 号** 10.11897/SP.J.1016.2017.00107

Cloud-Based Integration and Service of Streaming Data

WANG Gui-Ling[1)] HAN Yan-Bo[1)] ZHANG Zhong-Mei[1),2)] ZHU Mei-Ling[1),2)]

[1)](*Beijing Key Laboratory on Integration and Analysis of Large-scale Stream Data, Cloud Computing Research Center, North China University of Technology, Beijing* 100144)

[2)](*Computer Science and Technology Institute, Tianjin University, Tianjin* 300072)

Abstract Big data management and processing are a good place where a Cloud infrastructure shows power. Service offers an important way to implement the delivery and usage model for various Cloud-based resources and abilities. With the prevalence of large-scale sensor data, the scale of system has expanded dramatically, the complexity of the multivariate and heterogeneous data has escalated, and the concurrency and frequency of streaming data has increased a lot. These bring great challenges to the traditional streaming data system in addressing issues in areas such as processing efficiency, scalability and fault tolerance. The Cloud computing techniques can be the foundation to manage and process large scale streaming data because of their high scalability, parallel processing capabilities, support of service delivery model and fault tolerance. The issue of streaming data integration and services based on Cloud computing is the focus of this paper. The paper starts with the application requirements of large-scale streaming data processing and integration, presents a framework for streaming integration, discusses the state-of-the-art key technologies including streaming data query, service customizations, mechanisms for scalability

收稿日期:2015-08-18;在线出版日期:2015-10-30.本课题得到北京市属高等学校创新团队建设与教师职业发展计划项目(IDHT20130502)、北京市自然基金重点项目(4131001)、北京市教育委员会科技计划重点项目(KZ201310009009)资助.**王桂玲**,女,1978 年生,博士,副研究员,中国计算机学会(CCF)会员,主要研究方向为服务计算、面向服务的数据集成、大规模数据流处理等. E-mail: wangguiling@ict.ac.cn. **韩燕波**,男,1962 年生,博士,教授,中国计算机学会(CCF)高级会员,主要研究领域为互联网计算与云计算、服务互操作与服务组合、可靠分布式系统、大规模流数据集成与分析等. **张仲妹**,女,1990 年生,博士研究生,主要研究方向为服务计算、流数据集成与分析. **朱美玲**,女,1987 年生,博士研究生,主要研究方向为服务计算、流数据集成和分析.

and reliability, evaluation metrics and benchmark etc. The uses of Cloud computing for large-scale streaming data integration and service provision are scrutinized, and challenges and future trends are summarized.

Keywords streaming data; cloud service; data service; cloud data integration

1 引 言

网络化进程、规模效应和学科交融推动了云计算、大数据、物联网、社会计算等新发展,催生了计算机科学与技术一次又一次的变革和进步.近年来,随着 IPv6、传感和移动设备、物联网、泛在计算等的不断发展,来自不同行业领域、体现物理世界、人类社会生产生活等实际状态的感知类数据日益增长.据 Gartner 预测,未来 3 年,传感和移动设备将更深入延伸至我们的日常生活,导致数据爆发①.而如何充分利用这些数据为人类提供更智能、贴切的服务是当前研究面临的重要挑战.相对于以人为中心生成的传统互联网数据,这类数据面向物理世界,来自不同类型的设备,涉及物理世界和人类社会生产生活的方方面面,具有实时到达、持续不间断、到达速度快等特征,常被归类到"流数据(streaming data)".

国外咨询机构对企业信息化的调查也显示,70%的企业存在流数据实时处理的需求[1].除传统互联网领域外,大规模流数据集成与服务还广泛出现在智能交通、电信、物流、水利、工业监控等各个应用领域.例如,实时交通监管是流数据集成和服务的一类重要应用领域.传统大多数交通管理系统对于交通数据是基于静态数据进行离线计算处理的,即将一定时间段内采集到的数据集中存储和查询处理.这种处理方式无法满足交通数据实时查询和计算的处理需求.采用流数据处理技术是解决这一问题的新途径.我国大部分城市均在主要道路上部署车辆识别传感器,以一个大型城市的车辆识别交通数据为例,车牌识别传感器为成千上万个点,每个点的高峰采样频率为 1 条记录/秒,则每秒将产生成千上万条车辆识别数据,一年车辆识别数据记录数超过百亿条(来自我们曾实施的实际项目).基于对车牌识别流数据及历史数据的集成处理和分析,可以为交通管理部门提供各类实时交通管理服务,例如道路流量计算、伴随车辆发现、套牌车检测、黑名单白名单实时匹配、旅行时间计算等.国外例如爱尔兰首都都柏林市政采用流数据处理系统 IBM System S[2]监控全市上千个公交车的全球定位系统(Global Positioning System,GPS)信号,为 120 万市民提供服务,包括呈现实时交通信息(例如车流量、路段拥堵情况、最快旅行时间等)、在拥堵时为市民实时推荐最佳路线等.

在学术界,流数据的理论与技术研究也已经有十几年的历史.早期对流数据处理的研究主要包括流数据管理系统、实时数据库以及复杂事件处理系统中对事件流处理的研究等.近年来,随着感知类数据的爆发及新技术的发展,各类应用领域对流数据集成与服务提出了更高的要求,包括高吞吐低延迟的处理能力、能够动态适应流数据规模和速度、良好的容错性、支持服务使用模式等,为流数据研究提出了新的挑战,大规模流数据处理的研究成为学术界近年来重点探索的领域之一[3-6].

云基础设施是进行大数据处理的主要基础之一,而服务则是落实云计算环境中各类资源及能力交付和使用模式的主要方式.本文从云计算和服务的视角,对现有基于云计算的大规模流数据集成和服务研究资料进行归纳和总结.第 2 节给出基本概念定义以及基于云计算的流数据集成和服务的概念框架;在此基础上,第 3 节对云计算环境下的大规模流数据集成与服务的相关问题进行分析,包括流数据集成、流数据查询操作的实现及优化、流数据定制化服务、可伸缩性保障技术、可靠性保障技术以及相关评价指标和测试基准等;最后,第 4 节简要阐述基于云计算的流数据集成与服务面临的新挑战.

2 概念框架

2.1 流数据相关的基本概念

流数据的概念来源于数据库领域,一般将流数据看作无边界的、瞬时的数据项序列,同源流数据中的数据项都具有相同的模式.具体的,流数据指带有

① Gartner Says by 2017, Mobile Users Will Provide Personalized Data Streams to More Than 100 Apps and Services Every Day. http://www.gartner.com/newsroom/id/2654115, 2014

时间戳、按照到达时间的序列表示的数据. 流数据一般用时间序列(time series)模型来表示，例如，令 t 表示任一时间戳，a_t 表示在该时间戳到达的数据，流数据可以表示成 $\{\cdots,a_{t-1},a_t,a_{t+1},\cdots\}$[7]. 针对不同的流数据计算需求，常用的还有收银机(cash register)模型和十字转门(turnstile)模型等[8]. 流数据常采用元组、键值对(key-value)记录等几种不同的数据格式.

一个流数据集成与处理作业可以看作是由执行数据操作的“算子(operator)”或“处理单元(processing element)”集合互相之间通过数据“流(stream)”连接而成的有向无环图(Directed Acyclic Graph，DAG)形成. 如图 1 所示，算子在获取输入数据流后触发相关操作，算子的操作包括数据转换、处理及产生输出数据. 数据源接收算子负责接收来自多个数据源的数据，并将它们发送给下一个数据处理算子. 目的接收算子负责将流数据处理的结果递送到外部的应用或服务.

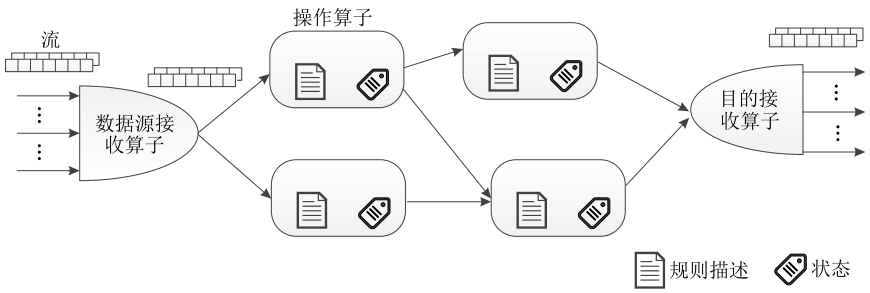

图 1　流数据集成与处理的抽象模型

根据算子是否需要在多个触发执行过程对状态进行维护，可将其分为“无状态(stateless)”算子及“有状态(stateful)”算子两类；根据算子在产生执行结果时是否需要预先读取整个流数据，可将其分为阻塞式算子及非阻塞式算子两类[9]. 在每一个数据处理算子中执行一些数据操作，包括数据转换、处理及产生输出数据等，可用流数据处理规则表达.

流数据处理的规则有两类，一类是转换规则(transformation rules)，一类是检测规则(detection rules)[9]. 转换规则实质上定义了一个由基本的算子运算符构成的执行计划. 这些运算符描述了算子所执行的包括过滤、连接、聚集等在内的数据操作. 执行计划可由用户自定义或由系统预定义. 系统预定义转换规则可用声明式语言和命令式语言来表达. 检测规则由条件和活动两部分构成，在事件流处理系统中尤为常见.

由于流数据是无边界的连续数据项，“窗口(window)”被用来对流数据算子执行时考虑的流数据进行限制. 流数据的窗口定义了算子执行过程中需要考虑哪一部分输入流数据. 现有系统定义的窗口可以分为时间窗口和计数窗口两类. 时间窗口的边界定义为一个时间函数，计数窗口边界使用窗口中包括的数据项数目进行定义. 根据窗口边界移动方式的不同，常见的窗口有滑动窗口和翻滚式窗口(tumbling-window)等[10-11].

2.2　基于云计算的流数据集成和服务的框架

近年来随着各应用领域感知类数据的爆发、系统规模上升、数据的多元异构复杂性的提升、流数据并发数量及速度的剧增，从处理能力、可伸缩性、容错性、使用模式等各个方面对传统流数据处理提出了新的挑战. 为了应对这些挑战，出现了新的流数据处理技术框架及系统. 本文所讨论的流数据集成与服务从多个不同的数据来源接收流数据，基于云计算基础设施进行并行处理，并对它们进行过滤、关联以及聚合等处理. 在此基础上，将流数据操作以 Web 应用程序编程接口(Application Programming Interface，API)方式提供给开发者进行定制化开发，降低开发成本. 图 2 展示了基于云计算的流数据集成与服务的一般性框架. 该框架主要分为云基础设施层、云数据管理层、云服务层以及性质保障等四个核心组成部分. 其中，云基础设施层提供对 CPU、网络和存储等云基础设施资源的虚拟化抽象、资源监控、负载管理、资源部署、存储管理和安全等功能. 云数据管理层通过缓冲区接收来自不同消息系统、数据源的流数据，进入流数据处理引擎进行实时处理. 云计算环境下的流数据处理引擎采用分布式的部署架构及并行处理模式. 基于流数据处理引擎，云数据管理层提供特定的流数据查询操作接口，例如基本查询操作、聚集查询操作、连接查询操作以及高级查询操作等. 云服务层基于云数据管理层提供的流数

据操作接口，提供流数据服务的建模、发现、编程、提供及托管功能，可面向不同类型的应用提供定制化的服务. 例如，支持统计监控视图的连续更新、支持事件通知应用以及实时的数据分析应用. 为了更好地支持应用和服务的并发请求、保障服务的质量，流数据查询处理操作的结果可通过消息队列将处理结果分发到应用和服务中，也往往分发到数据库中进行持久化，再以服务的方式对外提供数据访问接口.

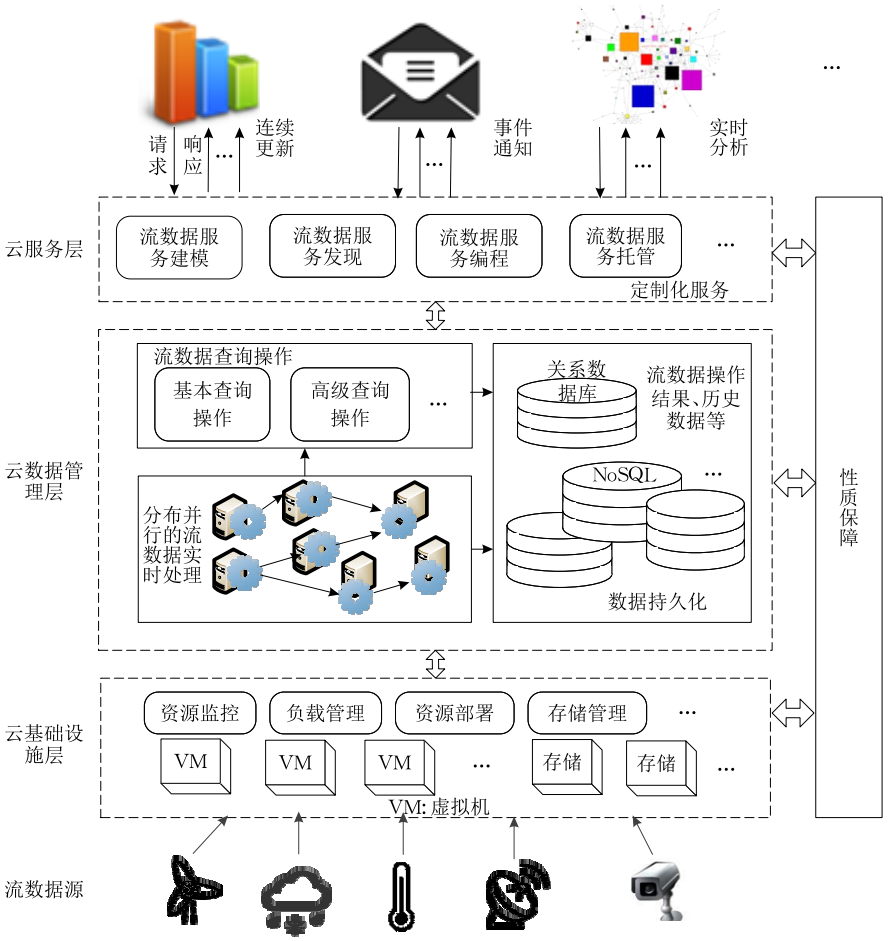

图 2 基于云计算的流数据集成与服务框架

此外，在大规模的云计算环境中，机器故障时有发生，系统访问量巨大且是动态变化的，因而流数据服务的可靠性保障、可伸缩性保障等能力的重要性非常突出.

云计算环境已经成为数据存储、管理和计算广泛使用的平台，虽然研究者对传统流数据处理系统的特点已有不同程度的归纳[12-13]，但对基于云计算的大规模流数据集成和服务的研究才刚刚开始. 文献[13]将流数据处理系统的发展划分为三代. 第一代流数据处理系统通常为单机版并且功能有限；第二代流数据处理系统为分布式体系结构，开始具备良好的容错性、支持适应性的查询等特征；第三代流数据处理系统是由云计算技术促成的，其典型特征是高可伸缩性及容错能力. 本文总结归纳云计算环境下的流数据集成与服务有以下几个特点：

(1) 支持服务交付和使用模式，支持多租户共享，支持“即取即用”(pay-as-you-go)的服务使用模式. 以服务的模式提供各类资源及能力，是云计算的显著特点之一. 在云计算环境下，流数据的各类操作可以以服务的方式交付使用. 同时，支持服务交付模式还意味着支持多个租户共享流数据处理的设施，并按照“即取即用”的模式使用.

(2) 实时性，高吞吐低延迟数据的处理能力. 因为每秒有大量(一般至少 1 万条记录以上)需要处理的数据，所以响应时间延迟需在秒甚至毫秒以内.

(3) 动态可伸缩性，能够适应负载变化. 动态可伸缩是云计算的显著特点之一. 在云计算环境中，应用和服务在理论上可以做到随意伸缩，即应用和服务所占用的资源可以在负载均衡的前提下，随着负载的上升或降低而增加或减少，从而保证在不同的负载下仍能获得一致的性能.

(4) 容错性，能够处理有缺陷的数据，在出现故

障时仍能够保持正常运行，并且保障容错的开销较小. 流数据源时有延迟、丢失和无序的情况发生，此外，云计算环境基于大规模中低端服务器组建，机器的故障在云计算环境中几乎是必然事件或“常态”，而非简单的“异常”. 因此，可靠性保障是流数据服务的一项基本要求.

可以从不同角度分析基于云计算的流数据集成与服务与传统流数据处理的区别. 从体系结构角度来说，基于云计算的流数据集成与服务的基础设施层的软硬件资源(或虚拟化软硬件资源)以分布式共享的形式存在. 为应对流数据规模和速率的变化，基于云计算的流数据集成与服务应具有动态伸缩和负载均衡的能力. 从数据处理模式的角度来说，为支持高吞吐流数据的集成与处理，基于云计算的流数据集成与服务应支持数据在多个节点上的并行化处理. 此外，基于云计算的流数据集成与服务支持服务交付和使用模式，对流数据处理延迟及吞吐量的要求较高. 在容错性方面，基于云计算的流数据集成与服务将机器故障看作“常态”进行处理，能够有效应对. 表 1 对二者进行了对比. 值得指出的是，虽然分布式计算环境流数据处理系统的一些技术也可以在云计算环境中适用，但本文仅着重介绍与分析那些基于云计算的流数据集成与服务区别于传统流数据处理的技术.

表 1　传统流数据处理系统与基于云计算的流数据集成与服务的对比

比较项	基于云计算的流数据集成与服务	传统流数据处理系统
体系结构	分布式，强调系统的动态伸缩	集中式
数据处理模式	多个异构数据源并发；数据分布式并行化处理	单一数据源；串行、单机并行等数据处理方式
交付与使用模式	支持即取即用的服务使用模式，支持多租户共享	传统软件部署和使用模式
实时性及吞吐量	高吞吐、低延迟	吞吐能力和延时性能相对较弱
可伸缩性	随负载变化动态伸缩	可扩展能力受计算和内存资源的限制
容错性	能够有效应对机器故障	将机器故障作为异常进行处理

3　相关问题分析

基于云计算的流数据集成与服务以云计算基础设施为基础，在合适的流数据处理引擎等工具的辅助下，对流数据源进行汇聚和集成，利用合适的流数据查询处理技术，对它们进行过滤、关联以及聚合等处理，并对外提供一系列流数据定制化服务. 在此，基于云计算的流数据集成与服务涉及到多方面的问题，如大规模流数据集成、流数据查询操作、服务模型、可伸缩性、可靠性等非功能属性的保障技术等方面，本文从集成与服务的视角将其中的关键问题进行归纳总结. 而对于从数据管理与分析、安全可信等角度出发的其他问题，如流数据查询语言、流数据分析与挖掘、流数据处理系统的安全性等[4-6,14]，本文不进行赘述.

3.1　流数据集成

流数据的集成问题大致可以分为几个方面：多个流数据源的集成、流处理引擎间的集成以及流处理引擎和传统数据库的集成[15]. 多个流数据源的集成是指把不同来源、格式、特点性质的流数据在逻辑上或物理上有机地集中. 流处理引擎间的集成是指流处理引擎实例之间的集成，以便综合利用多个流处理引擎的能力. 流处理引擎和传统数据库的集成是指将流处理引擎和传统数据库有机组织在一起，主要针对混合查询需求，即不仅涉及动态的流数据，还需要访问存储于传统数据库中的静态历史数据的查询需求. 下面分别对它们进行介绍.

云计算环境下用户的查询请求一般会涉及到多个流数据源，并且这些数据源常具有不同的数据模型和访问接口. 同样，对于流处理引擎的输出，也存在流处理的结果输出到多个不同目的系统中的情况. 对于这类问题的常见解决方法是提供适用于不同数据类型的适配器. 例如，Coral8① 和 StreamBase[16] 等流处理引擎就提供了大约 20 多个输入/输出适配器或 SDK，来连接不同的消息系统、数据种子(data feed)以及呈现数据处理结果的仪表板等. 这种基于适配器的集成方法虽然需要一些人工操作，不够灵活，但却是支持流数据输入/输出集成的最直接方法.

基于中介模式与数据源模式之间的语义映射关系利用视图回答查询(answering queries using views)是一种很普遍的数据集成方法. 这种方法同样可用于多个流数据源的集成系统中. 由于多个流数据源的模式各不相同，采用基于中介模式的集成方法，重写后的查询可能非常复杂. 为了克服此缺点，MDQ(Mapping Data to Queries)直接将到来的流数据的格式和模式映射到连续查询上[17]，并将查

① Engine C. Coral8. Inc.. http://www.coral8.com

询重写推迟到运行时. MDQ 技术对异构模式流数据的灵活集成有较大意义.

除了流数据的模式异构带来的挑战之外,与传统的数据集成不同,流数据难以预先存储,数据也不属于某一个流处理引擎,并且通过网络连续不断地到达,这使得待集成的数据源的不可预测性、不可靠性更加突出.因此,基于云计算的流数据集成对适应流数据负载变化的可伸缩性保障技术以及针对系统故障的可靠性保障技术具有更高的要求,本文将在 3.4 节和 3.5 节对它们进行详细分析.

对多个流数据源集成的最终目的是为了方便进一步的查询和分析,尤其是那些涉及到多个数据源分析和挖掘的应用.例如在文献[18]中,为了分析交通流与空气质量之间的相关性,涉及到交通流、人的移动性和气象数据等多种不同类型的数据;在文献[19]中,为了分析道路规划的合理性,涉及到交通流、道路结构等多种类型的数据.区别于传统先集成再分析的技术路线,多数据源的协同计算也是很多大数据分析应用采用的手段[20],这是流数据集成技术研究的一种新趋势.

流处理引擎间的集成可分为两类:同构流处理引擎间集成以及异构流处理引擎间集成.云计算环境下同构流处理引擎间集成的本质是利用分布、并行的流处理技术来提升流处理系统的可用性、可伸缩性,其关键技术见本文在 3.2 节和 3.4 节中对云计算环境下的流数据查询操作及可伸缩性保障的归纳和分析.异构流处理引擎间的集成在应对跨组织和地域的大规模流处理应用时非常有用.当前流处理引擎在语法及执行行为方面均存在多方面的异构问题,使得这种集成方式和传统的数据库集成有了很大不同.文献[21]提出一个形式化的模型,可用于描述和分析流处理引擎的语法和行为.这对增强异构流处理引擎之间互操作性的研究进行了理论上的铺垫,但对云计算环境下流处理引擎语法和行为的形式化描述与分析还是一个待研究的问题.

异构流处理引擎间的集成用于应对跨组织和地域的大规模流数据协同应用系统中不同参与方流数据处理引擎间的互操作,也常用于融合批处理、流处理及混合处理等不同的计算模式,以发挥异构数据处理引擎的优点、进行流数据处理的优化.文献[22]以窗口聚集计算为例的相关实验表明,流数据源窗口计算的窗口时间范围、滑动大小、数据速率等,都会影响不同计算模式下流数据处理的延迟.这为集成多种计算模式的优点带来了困难.文献[22]解决这一问题的思路是通过预测不同参数下流数据处理的性能进行计算模式的选取与切换,但其仅针对窗口聚集查询,无法适用于数据速率、窗口时间范围等参数动态变化的情况.

此外,面向常见的混合查询需求,即不仅涉及动态的流数据,还需要访问传统数据库中的静态历史数据的情形,需要将流处理引擎和传统数据库进行集成,例如提供一个统一的查询接口来支持针对流数据源和静态关系数据库数据源的混合查询. MaxStream[23]是针对此问题的一个典型工作,其目标是无缝集成多个自治、异构的流数据处理引擎以及传统数据库,对外提供基于结构化查询语言的声明式查询接口和共用的 API. MaxStream 的实现原理如图 3 所示,在客户端和关系数据库、流处理引擎之间增加了联邦层,联邦层对流数据连续查询请求进行解析、查询结果进行转换,数据代理则负责所有控制消息的交换以及数据的转发. MaxStream 可适用于跨组织和地域的分布式计算环境中,但在云计算环境下,如何针对动态变化的负载,对跨流处理引擎和传统数据库的连续查询进行功能及非功能属性方面的优化,还有待进一步的研究.

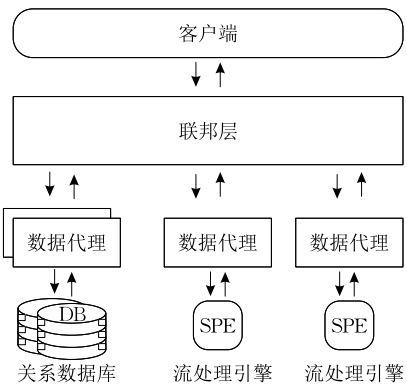

图 3 MaxStream:流处理引擎与传统数据库的集成[23]

3.2 基于云计算的流数据查询操作

流数据查询操作是实现定制化服务的基础,其相关工作可以分为两个方面.一方面是单个查询操作的实现及优化,例如连接查询、聚集查询等操作的实现及优化;另一方面是互相连接的多个流数据操作算子的执行及优化.针对包含多个流数据操作算子的优化方法,文献[24]进行了总结,如算子重新排序、算子去重等优化方法,本文不进行赘述.与传统数据库一样,流数据的基本查询操作也可分为选择、投影、连接(join)和聚集(aggregation)等操作.其中,选择和投影操作相对简单,而连接操作和聚集操作是两种较复杂、耗时的查询操作.在传统数据库研

究中，连接和聚集查询一直是热点研究内容，在流数据研究中，由于大规模流数据的实时性、不间断等固有特性以及云计算环境下分布并行处理的特点，为聚集操作和连接操作的实现和优化方法带来一些新的问题.除流数据基本查询操作之外，还有一些应用较广泛的流数据高级查询操作，例如流数据上的skyline查询、K最近邻(K-Nearest Neighbor，KNN)查询、关键字查询、相似查询等.下面针对它们在云计算环境下的实现及优化方法进行解析.

3.2.1　基于云计算的流数据连接查询操作

流数据在理论上是无限的，因而流数据上的查询一般被定义为“滑动窗口连续查询”.面向连接操作的滑动窗口连续查询(以下简称“滑动窗口连接查询”)可用于关联不同的流数据源，例如关联多个移动对象生成的数据等.流数据的连接查询算法可以分为两个或多个流数据之间的连接查询以及流数据和静态数据之间的连接查询两类.传统的对称哈希连接算法(Symmetric Hash Join，SHJ)扩展后可以支持滑动窗口连接查询[25].针对多个流数据源的连接查询，文献[26]提出进行多重连接的MJoin算法.针对流数据和静态数据之间的连接查询问题，索引循环嵌套连接(Index-Nested-Loop Join，INLJ)算法和MESHJOIN是两种基本的算法[27].在分布式和云计算环境中，连接查询主要通过数据分区技术实现①，文献[28]提出一种可用于云计算环境下的基于数据并行划分技术的流数据滑动连接查询方法.该方法可细粒度地动态调整每个节点上的窗口分片大小，以应对运行时流数据速率的变化问题.

真实系统中的流数据连接查询往往还需要针对具体情况灵活应对.例如，Photon从不同数据中心的GFS(Google File System)读入查询日志和点击日志后根据查询ID进行连接操作.由于Photon中查询请求数据流根据查询的时间戳基本有序到达，但点击流有天然的延迟，不是按照查询请求的时间戳顺序到达的，因此，Photon持久化保存最近N天内已处理过的点击ID的集合.N的选择取决于资源消耗和丢失数据影响面之间的权衡，系统对延迟到达N天之前的数据进行丢弃处理[29].

3.2.2　基于云计算的流数据聚集查询操作

根据分配型、代数型和整体型聚集函数作用在滑动窗口新增元组和消失元组的特点，可以将流数据上的聚集函数分为非整体型(指对窗口的新增元组和消失元组都是分配型或代数型的，例如sum、count等聚集函数)、半整体型(指对窗口的新增元组为分配型或代数型，但对消失元组不是分配型或代数型的，例如max、min、top-k等聚集函数)以及整体型(指对新增元组既不是分配型又不是代数型的聚集函数，如求中位数)等三类聚集函数[30].基于云计算分布并行的计算模式，流数据上的滑动窗口聚集操作有几种优化方法[22]：

(1)并行划分.基本的滑动窗口划分方法以窗口为单位将其划分到多个节点上执行，这种方法虽然简单，但由于连续的滑动窗口之间存在元组重叠的情况，导致同一元组划分到多个节点上重复处理.基于批量窗口的划分方法将多个窗口的元组作为一个单元划分到节点上进行处理，同一分片中的元组不需重复处理，减少了计算开销和空间开销.由于划分代价和计算代价都会随重叠元组数目提升而提升，因此，当窗口太大、流数据的到达速度太高时，基本窗口和批量窗口的并行处理方法不具有较好的可扩展性.此外，还可以与层次型处理方法结合，将窗口划分为子窗口后，再划分到多个节点上执行.

(2)增量式处理.增量式处理方法只处理相邻窗口中不同的元组.增量式处理只对非整体型聚集函数有意义，非增量式处理可应用于任何类型的聚集函数.增量式处理可以减少待处理的元组，加快元组处理效率，但可能造成处理浪费.

(3)层次型处理.层次型处理实质上是将窗口聚集函数进行两轮计算：将窗口分成几个不重叠的子窗口，先在子窗口上进行聚集函数计算，然后再进行整体上的聚集计算.在第二轮计算时，可以采用增量式和非增量式处理方式.层次型处理可对不同子窗口的数据进行并行处理，从而进一步提升处理效率.文献[31]将滑动窗口划分为互不相交但间隔相等的格(pane)，称为“基于格的窗口聚集查询”方法.基于格的划分方法以格为单元进行划分，将节点以“环”的拓扑结构组织，将格以轮询方法循环分派到环中的下一个相邻节点上，从而使得“基于格的窗口聚集查询”可在分布式的云计算环境中工作.基于格的划分方法消除了元组的重复处理，因此其划分代价不会随窗口重叠元组数目的提升而提升，而吞吐率也不会随之下降[32].划分并行由于数据划分、网络通信、并行任务的执行等处理会带来一些开销.

半整体型聚集函数由于不需要专门对消失元组进行处理，因此层次型处理的第二轮计算可基于子

① Ilya Katsov. In-Stream Big Data Processing. https://highlyscalable.wordpress.com/2013/08/20/in-stream-big-data-processing/, 2013

窗口的处理结果进行,节省了计算开销和空间开销.整体型聚集函数虽然无法基于子窗口的处理结果节省开销,但也减少了重叠窗口聚集查询的计算开销和空间开销.

文献[24]将多种计算模式优化选取方法用于聚集函数的优化,此方法已经在3.1节中进行了介绍,主要思想是利用不同种类流处理引擎的优点、融合批处理、流处理及混合处理等不同计算模式,从中选取优化的计算模式.该工作尚未考虑基于格的窗口聚集查询,还有进一步的研究空间.

此外,当前滑动窗口聚集查询方法的优化目标大多为减少计算开销和空间开销,在分布的云计算环境中,如何减少网络通信代价,如何在各种优化指标之间进行权衡,都是需要考虑的问题.文献[33]提出了优化目标为减少源节点和目的接收节点之间通信代价的△SPE算法,其核心思想是找到源节点发送到目的接收节点的最小聚集集合.但这方面的研究还很初步.表2对本节介绍的几种典型工作进行了对比.

表2 云计算环境中滑动窗口聚集查询操作实现方法对比

分类标准		概述	优点	缺点
并行划分	基本滑动窗口划分	单个窗口为一个并行划分单元	实现简单	同一元组可能划分到多个节点上重复处理
	批量滑动窗口划分	多个窗口为一个并行划分单元	同一分片中的元组不需重复处理	不太适合窗口太大、数据到达速度高的情况
增量式处理	增量式处理	只适用于非整体型聚集函数	减少待处理的元组,加快元组处理效率	有时会造成处理浪费
	非增量式处理	不对相邻窗口进行增量处理	实现简单	存在相同元组重复处理现象
层次型处理	基于格的层次型处理	以子窗口为并行划分单元	消除了元组的重复处理,可扩展性好	并行划分带来开销
其他	以△SPE算法为例	目标是减少源节点和sink节点之间通信代价	适合于网络带宽有限的情况	没有考虑计算和空间开销的优化

除流数据基本查询操作之外,流数据上的skyline查询、k最近邻查询、关键字查询等高级查询操作应用也较广泛,这类查询计算代价比较大,当数据规模大、速度高时,面临更大挑战.近年分布式环境中流数据高级查询的研究成为热点,而利用云计算环境提升流数据查询处理性能和效率的研究刚刚起步,其基本思想是采用划分的方法,将高速到达的大规模流数据进行有效划分,将其分配到各个并行计算节点上,利用各节点的并行来提高系统的性能.例如,文献[34]等通过交叉划分完整的滑动窗口的方法实现了流数据上分布并行的skyline查询;文献[35]在实现KNN查询操作时,使用哈希方法对流数据元组进行划分并行处理,同时设计了维护节点之间共享状态的策略.文献[36]面向流数据查询操作,提出了一个层次型的基于云计算的并行编程模型和框架.在分布并行的云计算环境中进行流数据高级查询操作还面临一系列难点问题.例如,数据划分到不同节点上之后,状态维护以及不同节点之间通信会带来开销,如何在开销及效率提升之间进行平衡;当负载变化时,如何动态保障系统的可伸缩性(此问题的详细讨论见本文3.4节).

此外,由于实际应用中产生的流数据往往是不确定或不精确的,在对这些不确定的流数据进行查询操作时,会涉及到流数据概率等复杂计算,使得针对大规模、高速到达的流数据进行查询面临极大的挑战.近年已有研究人员在如何利用云计算技术解决这些问题方面开展了一些工作[37-38].

3.3 流数据定制化服务

流数据定制化服务的目标是面向不同类型应用的需求,以服务方式提供对流数据连续查询、事件检测、实时分析等的能力,支持用户对大规模流数据集成和处理进行灵活的共享与定制.云计算环境下流数据定制化服务的现有工作可分为流数据服务建模、发现、编程、提供及托管等不同方面的工作.面向流数据的访问和获取,流数据定制化服务首先要考虑服务的抽象和建模问题.流数据的使用者还需要发现有哪些流数据需要访问以及数据的格式、语义、集成的方式等,流数据定制化服务向服务消费者提供API、表达语言、丰富的元数据等来满足这些需求,服务消费者可发送访问和查询的请求到服务提供者并从服务提供者获取数据,或进一步进行大粒度服务及应用的编程和构造.流数据服务的托管将用户对流数据进行管理、部署和运营的任务托管给云基础设施进行,并将流数据的采集、查询、分析和管理的能力在互联网上作为云数据服务[39]交付给用户使用,对用户提供"即取即用"的服务使用模式.

针对具有实时、持续不间断等特性的流数据,采用目前流行的网络服务(如Web服务)的方法和技术对其进行统一访问和查询存在天然的局限性.现有的服务抽象模型难以刻画对大规模流数据进行查询和集成的能力.例如,现存的服务模型(如基于Web服务接口描述语言(Web Services Description Language,WSDL)的服务模型[40])主要用来刻画与

其他分布式的组件进行交互的业务功能方面，不对数据源的数据模式进行显式的描述，将数据源抽象为一组带有输入/输出参数的操作接口.并且，这些操作接口是预先定义好的，只提供了对数据源有限的访问和查询功能，客户端无法针对数据源提交预定义功能外的定制查询请求.

针对服务模型存在的上述问题，国内外研究者在数据服务模型方面开展了不同程度的研究工作和实践[39,41-43]，包括“数据服务”(data service)以及“以数据为中心的 Web 服务”(data-centric Web Service)的工作.流数据服务的模型和数据服务模型类似，都将数据作为服务描述的“一等公民(first-class citizen)”.但由于大规模流数据具有规模大、实时、持续不间断等特性，使得二者有很大区别.表 3 对大规模流数据服务与传统数据服务从几个角度进行了初步比较.首先二者描述的数据源类型不同，其次，流数据服务模型还需屏蔽大规模流数据处理的复杂性，需要对流数据基本操作的算子进行定义，对流数据处理的规则进行描述.此外，相对于传统数据服务被动调用的方式以及较低的结果更新频率，大规模流数据服务的结果持续更新、频率较高，通常采用主动推送的调用方式.

表 3　大规模流数据服务模型与传统数据服务模型的比较

比较项	大规模流数据服务	传统数据服务
数据源类型	无边界的流数据；离散数据项，带有时间戳	静态数据；关系数据、网页数据、XML 数据等
主要功能	屏蔽大规模数据处理的复杂性	对数据源的只读访问和查询
调用方式	支持主动推送	被动调用
结果更新	结果持续更新，频率高	结果更新频率低
部署环境	部署在云存储、分布并行的流数据处理等云环境上	部署在传统数据库、服务及应用上

流数据服务的模型需要对服务的数据模型、操作、服务处理方法以及性质等进行刻画.针对这些问题，已有研究工作提出了流数据服务模型支持规模大、实时、持续不间断的流数据访问和查询等.Stream Feeds[44]建立了基于 RESTful 服务模型的流数据服务，将来自于传感器的流数据变为 Web 上可直接访问的资源.如图 4 所示，该服务模型支持将流数据更新实时、主动地推送(push)到客户端的调用模式(也兼容“拉(pull)”的服务调用模式)，在服务端提供基本的流数据过滤操作，支持将多个流数据融合成为一个新的流数据.同时，该服务模型支持对流数据及其历史数据的查询.在性能上，该服务模型也满足了流数据更新频率高、低延迟、数据量大等基本要求.与 Stream Feeds 的工作原理类似，Thing-REST[45]以及 Guinard 等人[46]的研究工作也为传感器数据等流数据建立了 RESTful 服务抽象.SOCRADES[47]则提供了 RESTful、基于简单对象访问协议(Simple Object Access Protocol，SOAP)和 WS-* 标准等两种流数据服务模型.针对特定领域流数据服务建模也有着相关研究，例如，针对时空轨迹流数据，Chu 等人[48]在传统实体-关系(Entity-Relation，ER)模型的基础上，提出了能够刻画时空变化及它们与移动对象之间关系的数据模型，并基于此数据模型提出交通流数据分析服务模型，能够捕捉交通流在不同情境下的变化.

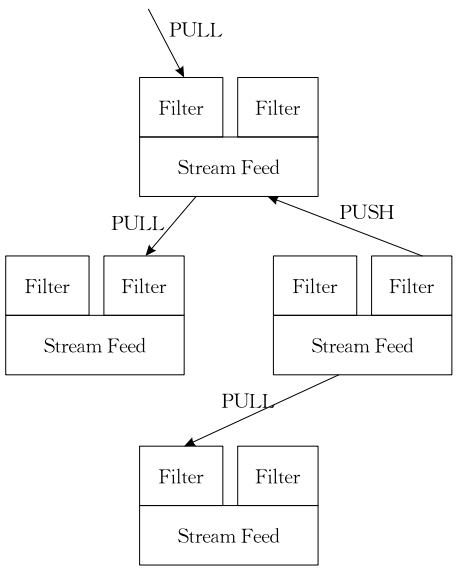

图 4　Stream Feeds：一种流数据服务抽象模型[44]

应该认识到，在数据源数据发送速度很高的情况下，并不需要将流数据源直接抽象为 HTTP 可访问的服务资源，将流数据查询操作提供为服务更有意义.此外，通过领域知识的重用面向领域进行流数据服务建模也是亟待研究的问题.

为了利用流数据构造具体的应用，研究者们提出了不同的编程模型与方法，从基于特定编程语言(例如 IBM 在分布式集群中进行流数据处理的编程语言 SPL[49])、基于程序库(例如 Spark Streaming[50])到基于 SQL 的流数据查询语言(例如 CQL[51])等.利用流数据服务构造应用的方法则延续服务计算的核心思想，采用面向服务的编程模型，在近年来传感器流数据应用的构造过程中得到广泛研究.其主要思想是将现实世界中的传感器流数据抽象为服务，并将其与传感器所在的周围情景信息及用户的情景信息相关联，从而实现灵活的智能服务，能够感知物

理世界并主动做出响应.与传统的服务不同,由于流数据服务大多是基于计算能力有限的传感器设备提供的,其数量多、变化快,因此在基于流数据服务的编程方法中,需尽可能减小服务执行和注册的开销,同时对根据情境信息动态发现服务、并主动按需向用户提供服务的能力具有更高的要求.在这方面,SOCRADES 提出了一种流数据服务发现、选取及按需提供的方法[47].Thing-REST 在流数据服务模型的基础上提出了一种 mashup 基本结构来生成智能 mashup 应用[45],但其表达能力未见评测.从上述内容可见,当前研究还比较初步,未来应对传统服务编程的方法加以改进,使其符合流数据服务自身的特点并满足流数据应用特有的需求.

在流数据服务的托管方面,亚马逊的 Kinesis 向用户提供了方便的客户端库以及 RESTful 服务编程接口来收集、分析由应用程序产生的数据流,用户无需自行运维和搭建流数据处理基础设施①.Google BigQuery② 以及 Google Predictive API③ 也是一类流数据定制化服务,用户无需搭建流数据处理系统,即可以每秒 10 万条记录的速率发送数据到指定接口并进行实时查询及分析.多租户共享的服务使用模式下,流数据规模和速度具有突出的动态变化性,因此,流数据服务托管面临的主要挑战是不可预测及动态变化的负载.解决此问题的关键是应用和服务的可伸缩性保障,使其能够灵活地伸缩,避免系统瓶颈并提高资源利用率.相关研究问题及现状的分析在下节进行介绍.

3.4 可伸缩性保障

可伸缩性(scalability)是云计算环境下支持多租户共享的应用或服务重要的度量指标.一个可伸缩的应用或服务意味着负载变化后,还能够在保障一定服务质量的前提下正常提供服务.

流数据处理领域对于可伸缩性的研究可分为集中式和分布式两类.在集中式环境下,受计算和内存资源的限制,可以通过算子重新排序(reordering)、负载降载(load shedding)以及延期处理(deferred processing)等方法来提供可伸缩性[52].由于流数据的到达速率和用户需求的多变性以及单个服务器计算资源的有限性,单个服务器往往无法完成大规模流数据的处理.很多研究开始考虑在云环境上处理流数据[53-54],利用云基础设施基于虚拟化技术动态管理和扩展节点的能力,为处理任务按需分配计算资源,应对流数据规模和速率的剧增.当前面向大规模流数据的流数据处理系统或中间件,如 S4[55]、Storm④、StreamCloud[56]、Esc[57]、MillWheel[58]、Spark Streaming[50]、SEEP[59]、ChronoStream[60]、Samza⑤ 以及 TimeStream[61] 等都建立在可动态管理和扩展节点的云基础设施之上.算子放置技术与数据并行化处理技术是保障基于云计算的流数据服务可伸缩性的关键技术,其中,数据并行化被认为是基于云计算的流数据处理系统的主要特征[13].下面结合学术界及工业界的实际系统及研究对其进行分析.

流数据处理 DAG 中的逻辑算子可能根据其资源的需求情况部署在多个节点上,图 5 是一个云计算环境下处理任务(算子)的部署示意图,其中一个虚拟节点上可注册多个资源容器,一个逻辑算子可部署到多个资源容器中.图中虚线框中的逻辑算子 v_6 部署在多个虚拟机节点 n_1 和 n_2 中.针对由多个处理任务(算子)组成的流数据处理网络,在云计算环境下,随着计算节点的增加,主要通过在多个节点上平衡算子的分布来提供伸缩性,其关键问题是如何进行算子的放置(或布局),以及如何在不同的节点上进行负载均衡.这就是"算子放置(operator placement)"问题.算子放置问题是指将处理任务的一系列算子布局在一系列的节点上执行,并在满足一定约束条件的情况下达到优化目标.

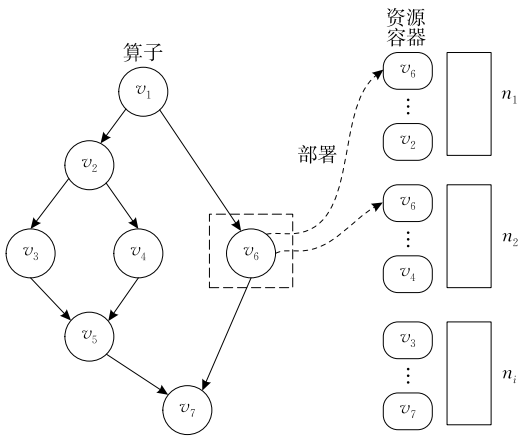

图 5 云计算环境中算子的部署[60]

Lakshmanan 等人[62] 将现有的算子放置算法从体系结构、算法结构、优化目标、算子层次上的操作、动态重配置等 5 个维度进行了分析和比较.在体系结构上,分为主从式、分散式以及二者混合等情况,当前,大多数云环境下的流数据处理系统或中间件

① Amazon Kinesis. http://aws.amazon.com/kinesis, 2014
② Google BigQuery. https://cloud.google.com/bigquery, 2015
③ Prediction API. https://cloud.google.com/prediction, 2015
④ Twitter. Storm (v0.9.0). https://github.com/nathanmarz/storm, 2015
⑤ Samza. http://samza.incubator.apache.org/, 2015

采用主从式体系结构进行资源的调度来平衡算子的分布.然而 S4 采用了分散式结构,由各对等节点互相协调实现资源调度分配.在算法结构上,有中心式算法(基于全局状态信息进行决策)、分散式算法(基于本地资源和负载信息进行决策)两种.在优化目标上,可分为计算负载、延迟、带宽或以上几个指标的混合等几种情况.在算子层次的操作上,可分析是否采取算子结果重用、复制等机制来提升处理效率.为适应网络、数据或处理流程等的变化,需要对算子放置进行动态的重新配置,可分为离线重配置以及基于当前工作负载进行在线重配置两种情况.除 Lakshmanan 等人提到的工作之外,最近出现的关于算子放置算法的研究包括 SODA[63]、SQPR[64]、面向 Storm 的适应性调度机制[65]以及 Twitter 公司的流处理系统 Heron[66]等,其主要进展在算子动态重配置方面.当前,越来越多云计算环境下的流数据处理系统支持在线的算子重配置,从而可根据网络、数据或处理流程等的变化灵活地扩展资源、调整资源分配,避免系统瓶颈并提高资源利用率.因此,下面着重从该维度对它们进行分析.

SODA[63]以满足根据任务优先级进行区分以及最大化资源利用率作为优化目标,通过约束优化模型及启发式混合算法求解资源分配的问题.在动态重配置方面,SODA 采取周期性的动态调度策略,可进行运行时的适应性算子重配置,并在每个调度周期将问题分解为多个阶段,每个阶段对应一个优化模块,从而提升了问题的求解效率.与 SODA 不同,SQPR[64]在新的任务提交时,只对那些与新提交的任务存在资源共享关系的算子资源进行重新分配,因而减少了算子重配置的开销.文献[65]中 Storm 的优化目标是减少节点间的网络流量,在动态重配置方面,其在线调度机制考虑运行时的资源约束,采用约束优化模型及启发式混合算法来实现.值得指出的是,Storm 对每个 worker(Storm 中的工作进程,负责执行作业的一个子集)公平对待、分配同样的资源,但事实上每个 worker 所需资源并不相同,因此存在为算子超量申请资源的问题.Twitter 最近在 Storm 基础上研发了 Heron[66],一个 worker 只运行一个算子,可以对每个算子占用的资源进行不同的细粒度配置,提高了资源利用率.这样也会带来新的问题,即每个作业中 worker 数目增多将会导致通信端口资源不足.为此,Heron 对 Storm 的体系结构进行了改进,worker 之间的数据通信不再是点对点进行,而是通过本地的流数据管理器(Stream Manager,SM)统一路由.这使得系统在提高资源利用率的同时,可伸缩性不会受到大的影响.

针对云计算环境下流数据的高吞吐特性,当每秒需要处理的流数据规模较大时,即使将算子部署在多个节点上执行,节点上的单个算子也将成为数据处理的瓶颈.因此,在算子内部进行数据并行化(intraoperator parallelism)是解决此问题的另一关键技术.算子内部的数据并行化将持续到达的流数据划分为并行的子数据流,并利用并行化的算子来处理相应的子数据流,可在负载变化时提高流数据处理的吞吐量.数据并行化一般按照并行化是否能够在运行时进行调整分为静态并行(statically parallelism)和动态并行(dynamically parallelism)[59]两类.其中,静态并行在应用构建时设置,而动态并行在此基础上还可以在应用运行时动态进行调整.

如表 4 所示,静态并行需要开发人员在部署应用时了解所有算子执行需要的资源,这依赖于流数据的到达速率以及数据的分布情况.目前较为流行的流数据处理系统 S4、Samza 和 Spark Streaming 等都支持静态并行(注:表 4 中支持动态并行的流数据处理系统也同时支持静态并行).文献[67]还提出了一种代价模型来评价静态并行的数据划分带来的开销.静态并行不能够根据随时变化的流数据到达速率以及算子执行的负载情况动态地修改并行度(parallelism degree)[68].

流数据处理的动态并行一般通过某种监控机制,能够实时发现负载过高或处在瓶颈期的算子,并根据系统的性能和算子的性质制定并行策略,动态地调整并行度.文献[35]提出了一种一般化的动态并行模型——split-(process*)-merge 模型,该模型将处于瓶颈的算子的输入数据流利用 split 操作进行划分,将输出的数据流结果利用 merge 操作进行合并.制定并行策略时需要考虑的问题主要有算子是否适合并行化、最优的并行化级别是什么以及采用哪种并行化技术(例如轮询技术、哈希技术等).

针对流数据处理的无状态算子和有状态算子,有一些动态数据并行化的工作展开.文献[69-71]采取了动态并行度的概念,但是仅仅针对无状态的算子,或者不考虑算子的状态,没有涉及有状态算子的正确性.如表 4 所示,Storm、Esc 等云计算环境下的流数据处理中间件支持针对无状态算子的动态并行化处理.值得指出的是,Storm 虽然支持运行过程中调整并行度,但需要开发人员或者管理人员的介入.

表 4 基于数据并行化的可伸缩性保障方法对比

实现方法		技术方案	优缺点	系统
静态并行		基于静态分析在提交任务前制定并行策略	优点：执行过程中无额外开销 缺点：无法在运行中调整并行度	S4，Samza，Spark Streaming
动态并行	无状态算子	基于 split-(process*)-merge 并行模型	优点：可在运行中动态调整并行度 缺点：相对于静态并行化方法存在额外的开销	Esc，Storm
	有状态算子	基于状态共享、状态迁移等手段保障操作正确性	优点：可在运行中动态调整有状态算子的并行度 缺点：保障状态正确性带来额外开销	StreamCloud，MillWheel，SEEP，ChronoStream，TimeStream

有状态的算子不能够简单地进行划分并行化和合并，还需保障状态的正确性，即能保障调整前后并行执行的正确性. 表 4 列出了支持有状态算子并行化处理的流数据处理系统或中间件，包括工业界的 TimeStream、MillWheel 以及学术界 StreamCloud、SEEP 等工作. 涉及有状态算子的并行化状态正确性的保障通常可以通过状态迁移、状态共享等方式实现. 状态迁移是指在为算子添加或删除节点的方式实现方案转换时，将算子的状态连同算子一同迁移[72]. 状态共享通常通过共享的内存来实现状态的共享，因此需要避免共享数据的竞态条件[24]. 如何减少保障正确性带来的额外开销是有状态算子并行化解决方案的约束条件. 本质上，状态迁移力求在负载均衡的前提下最小化迁移成本，而状态共享则要求在正确的共享内存管理和避免竞态条件（如通过保证数据不可变或同步访问）的前提下尽量减少内存占用[24,73]. 文献[68]进一步将有状态并行化问题理解为效益问题（profitability problem），即力求以尽可能小的代价（额外的资源消耗等）获取尽可能大的效益（正确性保障、高吞吐、低延迟等）. 解决这一问题仍面临较大挑战. TimeStream、StreamCloud 和 ChronoStream 等工作研究了透明的数据并行化编程模型或流数据处理表达方式，使数据并行化细节在语法和语义上对开发人员透明，以减轻开发人员的负担，这也是流数据处理并行化不容忽视的研究问题.

3.5 可靠性保障

可靠性保障是云计算环境下流数据服务及应用的一项基本需求[74-75].

导致流处理系统不可用的故障，按其影响可分为如下两类：(1) 失效-停止（fail-stop）故障，表现为节点/系统的数据处理功能停止. 这一类的故障出现后，除非存在外在（人工或者非人工的）干预，数据处理过程无法继续；(2) 瞬时失效（transient unavailability）故障，表现为节点的处理并不停止，但经常性或间歇性（如每 1 min 或每 10 s）出现不超过 10 s 的暂时停滞[76]. 瞬时失效故障源于资源竞争，计算资源（CPU、内存和带宽）没有被处理任务合理的分配. 这类故障可在被竞争的资源释放后自动恢复. 自 2009 年开始，以 IBM 的工作为代表[76-77]，这类研究逐步兴起.

在云计算环境下，当系统扩展处理节点的数量时，也增加了因为某一节点的失效导致系统整体受到影响的风险[75]. 当前许多流数据处理系统，如 HOP[78] 和 S4，假设数据丢失和结果不准确是可以容忍的，其可靠性保障仅仅是重启故障的节点继续进行数据处理. 但是，这种假设在实际场景下并不总是恰当. 例如，基于流数据的交通流量和旅行时间的统计业务，计算中间结果作为必要的状态若无法恢复，势必造成恢复时间较长的滞留或恢复后结果较大的误差[79].

此外，由于存在内存、带宽等开销，可靠性保障在系统中体现为处理性能与保障效果的折中. 在高速高并发的应用场景下，流数据快速到达系统，一旦某一节点出现故障，不但影响甚至中断下游的数据接收和处理，同时导致未能传递的数据迅速积累，影响上游节点的处理过程，甚至产生连锁反应[80]. 为了保证在故障时无人工干预也可以快速地恢复数据处理，节点的被动副本、主动副本及上游备份等策略和技术在当前得到了广泛应用. 由于具体需求不同，相关策略和技术意味着不同的内存、带宽等开销，以及恢复延迟和恢复后结果的精确性. 例如，基于被动副本的故障恢复，实际是以增加延迟和资源开销换取系统可靠性[77].

如表 5 所示，流数据处理的可靠性保障通常包

表 5 云计算环境下的流数据服务可靠性保障方法对比

分类标准		特点	适用场景
运行时的备份方法	上游备份	上游节点保存发送的数据，实现简单、开销小，但恢复慢	高并发、高带宽负载的数据监控
	主动备份	备份副本完全冗余并独立运行，在故障时切换. 开销大、恢复快	高负载、快速恢复的应用
	被动备份	备份检查点，故障时由日志恢复. 开销小，实现复杂	低负载、精确恢复的应用
故障时的恢复目标	间隔恢复	保障故障恢复后节点能继续处理数据. 开销小，实现简单	精确性要求不高的应用，如基于传感器的网络监控等
	回滚恢复	保障故障恢复后数据不丢失	基于条件通知的应用，如火警、资产防盗等
	精确恢复	保障故障恢复后数据不丢失且不重复. 开销大、恢复慢	精确性要求较高的应用，如实时的金融分析和军事计算等

括运行时节点备份和故障时节点恢复两个阶段，可将系统的保障技术从这两个维度进行分类.

一方面，从运行时节点备份方法的角度来看，流数据服务的可靠性保障可以分为上游备份、主动备份和被动备份 3 类：(1) 上游备份在上游节点保存发送数据，优点在于实现简单，运行时开销小，但存在恢复延迟大的缺点[81]；(2) 主动备份的副本冗余独立运行，其优点在于故障后备份副本可以快速地切换继续处理，缺点在于系统开销要增长一倍以上. 另外，备份副本的状态同步[81]也是一个挑战，特别是当节点存在不确定性操作(如随机化抽样)的情况；(3) 被动备份是主副本的状态增量备份，优点在于运行时资源开销较小而且可以灵活折中保障的效果；缺点在于主副本故障时需要重建副本状态，恢复延迟大，实现也较复杂.

另一方面，从故障时恢复目标的角度来看，流数据服务的可靠性保障可以分为间隔恢复、回滚恢复和精确恢复 3 类. (1) 间隔恢复的恢复目标最弱，容忍节点恢复后的数据丢失. 这类保障机制在许多精确性要求不高的场景，如基于传感器的网络监控环境，有着广泛应用. 由于开销小和实现相对简单的优点，也应用于当前许多分布式数据流处理系统，如 HOP[78] 和 S4[55] 等；(2) 回滚恢复相比间隔恢复，保证了节点恢复后的数据不丢失. 它使得故障前后输出数据等价，但允许故障前系统已经处理的数据重复，故适用于基于条件通知的应用，如火警、资产防盗等系统[81]. 为达到这个目标，往往需要备份上游节点的输出数据，重建故障节点的状态；(3) 精确恢复相比回滚恢复，保证了节点恢复后的数据不丢失且不重复，适合应用在数据精确性要求较高的应用，如实时的金融分析和军事计算等. 为达到这个目标，需要在回滚恢复的基础上，从欲恢复的数据中移除故障前系统已经确定处理的数据. 相比而言，精确恢复的开销最大，恢复延迟最长.

3 种备份方法和恢复目标有着各自的优点、缺点和适用场景，恰好体现了不同的可靠性需求和对性能和开销的不同程度的折中. 在实际的系统中，由于存在多样的业务需求和大量处理节点，系统可能需要同时实现主动备份和被动备份[76-77]. 事实上，同样的备份方法可以实现不同的恢复目标；相同的恢复目标可以通过不同的备份方法实现. 例如，Borealis[82] 采用主动备份方法实现精确恢复目标；S4[55] 采用主动备份方法实现间隔恢复目标；Flume① 采用主动或被动的备份方法实现间隔或回滚恢复目标；CLASP[83] 采用主动或被动的备份方法实现精确恢复目标；Storm 采用上游备份方法实现回滚或精确恢复目标等.

3.6　评测基准

评测基准是可用于评测、比较不同系统性能的规范. 评测基准用于客观、全面反映具有类似功能的系统之间的差异. 相较于大数据处理的其他技术，关于基于云计算的流数据处理系统、应用和服务的评测基准方面还存在许多亟待研究之处.

传统的数据管理技术如数据库处理，其评测基准包含度量指标、模拟数据生成器、工作负载设定、审计等要素[84]，发展已相对成熟和稳定. 流数据处理由于各个要素普遍缺乏标准，评测基准还有其特殊性. 对于流数据处理，基准测试的需求在于下述方面. (1) 广泛的评估和比较流计算系统. 特别是，除了根据传统分布式系统的吞吐量和响应时间等指标评估性能之外，还需要评估可用性、可伸缩性等关键要素；(2) 评估不同数据或计算特征下的能力. 特别是，除了能够根据传统平稳运行或峰值指标评估系统能力，还需要针对动态变化的负载(例如突发数据暴增)评估其伸缩能力；(3) 产生不同的有代表性的负载. 特别是，与传统分布式系统相比，基于云计算的流数据处理系统由于面向多个不同类别的租户提供服务，业务需求差异大，缺乏统一的具有代表意义的数据，故难以设计普遍适用的相对公平、合理的负载.

LinearRoad Benchmark(简称 LRB)是传统流数据处理应用非常广泛的测试基准，它由 Aurora 和 STREAM 合作设计[85]. LRB 是一定规模的车辆在一个城市的多条高速公路上行驶时的位置、车速等数据，基于这些数据，可以进行计费通知、事故通知、旅行时间估计等查询. LRB 可用来测试不同系统执行这些查询时的性能.

近年来大数据的繁荣带来了相关评测基准的发展. HiBench[86] 针对 Hadoop 海量数据处理，其中包含了 7 个测试场景，使用诸多机器学习的算法作为基准程序. BigBench[87] 针对端对端的大数据处理，其中包含了丰富的业务用例，借鉴了 TPC-DS 的思想，使用了结构化和非结构化的数据产生数据负载. BigDataBench[88] 针对生产环境下的业务，包含 6 组真实数据集和 19 种测试场景. BerlinMOD[89] 针对

① Cloudera. Flume Documentation (v0.9.3). http://archive.cloudera.com/cdh/3/flume/, 2011

连续移动对象数据,给出了具有代表性的数据集,设计可扩展的查询语句作为基准程序,评估查询处理的能力.此外许多业界知名的流处理系统,各自完成了在自身适用场景下的专用评测,并给出了相关技术报告.例如,S4 给出了典型的基于点击事件的基准程序评估自身性能;Spark Streaming 通过典型的流数据 grep 运算和单词计数统计(word count)作为基准程序,不仅评估了自身性能还给出了扩展性和容错能力指标;微软的 TimeStream[61] 使用相异计数(distinct count)和推文(Tweets)分析作为基准程序,评估了自身的扩展性和容错能力.应该看到,云计算环境下针对流数据评测基准的工作还亟待深入开展相关研究.

4 挑战与展望

流数据处理虽然不是一个新的研究领域,但其内涵和外延已然发生变化.由于物联网和云计算的兴起和发展,遍布各行业的传感和移动设备将导致流数据大爆发,传统的流数据处理技术面临新的挑战.以交通应用领域为例,传统的流数据处理技术在面向多个区域和组织提供大规模实时交通监控及管理服务时,在支持多租户共享及方便的交付及使用模式、服务的实时性、可伸缩性、可靠性等方面,还有大量的问题有待进一步解决.当前,云计算环境下的流数据的集成与处理技术还处于起步阶段,尤其是从服务角度研究基于云计算的大规模流数据集成与服务的工作还比较少,未来研究工作及面临的主要挑战集中在以下几个方面:

(1)流数据服务模型.在前面的分析和介绍中可以看到,流数据与传统的数据有很大不同,流数据定制化服务在服务的抽象和建模、服务编程等方面还面临一些挑战.首先,现有服务抽象模型还不具备对大规模、实时、持续不间断、多样化和多变的流数据访问和查询进行刻画的能力.针对此问题,未来关于流数据服务抽象模型的研究一方面要在底层并行流数据计算框架基础上屏蔽大规模数据处理的复杂性,另一方面需要能够对流数据的连续查询和计算进行灵活的定制,从而方便服务的发现和使用.其次,由于流数据服务大多是基于计算能力有限的传感器设备提供的,其数量多、变化快,基于流数据服务的编程方法在减小服务注册及发现的开销、增强服务发现及编程的动态性、对情境信息变化的适应性等方面面临挑战.未来研究可在流数据服务模型的基础上,从流数据服务操作算子定义及执行优化等方面对传统服务编程的方法加以改进,使其符合流数据服务自身的特点并满足流数据应用特有的需求.

(2)基于云计算的流数据服务的性能优化问题.吞吐量、延迟、可伸缩性等是衡量大规模流数据服务的主要非功能属性,随着流数据的空前爆发,高吞吐低延迟的大规模流数据处理是当前普遍面临的挑战.针对大规模的流数据处理的吞吐量和延迟时间等实时性指标,首先,云计算环境下与特定流数据处理操作无关的优化方法是未来待研究的方向.例如,基于集群架构,将数据划分并行与其他多种优化手段相结合,动态优化选取不同的计算模式、不同的优化策略及算法.此外,针对大规模流数据特定操作(例如连接查询、聚集查询等),未来可研究特定的分布式并行查询优化方法.流数据处理还应该能够适应流数据规模和速度的动态变化,适应不同的负载.研究适应性的大规模流数据处理集群构造和负载均衡算法,特别是针对多租户共享流数据处理基础设施的情况,支持对不同类别的租户应用能够适应性地进行资源分配.将数据划分并行与可伸缩性保障技术结合起来,根据负载情况动态地进行流数据划分和并行执行,例如,动态决定划分数目、进行算子资源分配等.

(3)基于云计算的流数据服务的可靠性保障问题.由于流数据处理性能的提高与处理过程的可靠之间以及处理过程的实时性与保障机制产生的延迟之间存在天然的矛盾,使得大规模流数据处理的可靠性保障较传统分布式数据查询面临更大的挑战.特别是,如何适应云计算环境下多变的负载环境、支持可配置、满足多种需求的可靠性保障,是实现基于云计算的流数据服务的挑战和难点所在.未来可基于分布式的集群架构,重点研究如何根据业务需求、节点分布及负载环境的特性,综合利用不同的备份方法和恢复目标,降低可靠性保障过程的系统开销,应对可靠性保障的代价矛盾,并提高对动态负载环境的适应性.

(4)云计算环境下流数据的集成与服务的评测基准.针对流数据的评测基准存在如下挑战:一方面,流数据上的操作应用相关性强,普适性相对较弱,不可能设计覆盖所有应用或场景的全面的基准程序.那么,如何设计"合理"的基准程序是具有广泛意义的难题.另一方面,流数据是一种在线数据集,呈现时间具有不可逆的特征[5],被一次性消费而难

于重放.那么，如何灵活、按需配置流数据负载，使之适应更多场景的基准程序和系统，甚至模拟产生数据负载的暴增，评估系统的可用性和扩展性等非功能保障能力，也是极具挑战的难题.未来可面向交通等领域的流数据应用的共性需求，并针对可伸缩性、可用性等重要指标，模拟云计算环境下面向多租户应用的灵活、按需配置的流数据负载，研究领域相关的流数据基准程序及数据集.

(5)基于云计算的流数据集成与服务的研发与应用.由于上述挑战性问题的存在，研发流数据集成与服务及其应用仍然是一项具有挑战性的工作.北方工业大学大规模流数据集成与分析技术北京市重点实验室已经初步研发了海量交通感知数据实时处理平台软件系统，并且结合北京、深圳真实的城市道路车辆识别数据和违法车辆甄别及交通流计算等实际应用需求，进行了相关研究成果的应用验证，建立了海量交通感知数据实时处理测试基准程序、数据及用例集(数据量达100亿数据记录，用例包括违章车辆实时甄别布控、实时交通流分析、车辆数据高效查询分析等3类业务10余个用例).未来，将针对本文上述挑战性问题，进一步研发面向大规模流数据的云服务平台DeCloud.DeCloud拟构建在支持服务使用模式的云计算基础设施上，支持高吞吐、低延迟的流数据集成与处理，具有容错性和动态可伸缩的能力.其设计目标具体包括支持多种异构流数据源的接入，支持流数据的实时计算以及对历史数据的离线计算及查询，支持流数据的基本查询和部分高级查询操作，并提供对流数据丰富灵活、面向智能交通等行业多种类型应用的定制化服务等.

5　结束语

近年来，随着感知类系统深入延伸至人类的日常生活，新的数据在源源不断的产生，流数据空前爆发.在这种背景下，对响应时间要求比较高的流处理类大数据应用需求也越来越普遍，传统的流数据处理技术面临新的挑战.本文从如何利用云计算提升流数据处理能力以及如何利用服务提升流数据共享能力等视角出发，对近几年出现的基于云计算的大规模流数据集成与服务相关研究成果进行了归纳阐述，归纳了大规模流数据集成与服务的基本概念和一般框架.文中着重分析了大规模流数据集成、流数据查询操作、定制化服务、可伸缩性保障技术、可靠性保障技术以及相关评测基准等关键技术，还对基于云计算的大规模流数据集成与服务面临的新挑战进行了简要阐述，对我们的研发工作进行了展望.云计算环境下大规模流数据集成与服务的研究为我们带来若干亟待突破的课题，其研究成果可以直接应用于智能交通、电信、物流、互联网、物联网等多个领域，具有重大的研究价值和应用价值.

参　考　文　献

[1] Liu X, Iftikhar N, Xie X. Survey of real-time processing systems for big data//Proceedings of the 18th International Database Engineering & Applications Symposium ACM. New York, USA, 2014: 356-361

[2] Gedik B, Schneider S, Hirzel M, Wu K-L. IBM research report elastic scaling for data stream processing. USA: IBM, Technical Report 25401, 2013

[3] Cugola G, Margara A. Processing flows of information. ACM Computing Surveys, 2012, 44(3): 1-62

[4] 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)
(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)

[5] 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)

[6] Cui Xing-Can, Yu Xiao-Hui, Liu Yang, Lu Zhao-Yang. Survey of distributed stream processing technologies. 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)

[7] Jin Che-Qing, Qian Wei-Ning, Zhou Ao-Ying. Analysis and management of streaming data: A survey. Journal of Software, 2004, 15(8): 1172-1181(in Chinese)
(金澈清，钱卫宁，周傲英. 流数据分析与管理综述. 软件学报，2004，15(8)：1172-1181)

[8] Muthukrishnan S. Data Streams: Algorithms and Applications. Boston, USA: Now Publishers Inc, 2005

[9] Cugola G, Margara A. Processing flows of information: From data stream to complex event processing. ACM Computing Surveys, 2012, 44(3): 15

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

[11] Golab L, Özsu M T. Issues in data stream management. ACM Special Interest Group on Management of Data (SIGMOD) Record, 2003, 32(2): 5-14

[12] Stonebraker M, Cetintemel U, Zdonik S. The 8 requirements of real-time stream processing. ACM Special Interest Group on Management of Data (SIGMOD) Record, 2005, 34(4): 42-47

[13] Heinze T, Aniello L, Querzoni L, Jerzak Z. Tutorial: Cloud-based data stream processing//Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems. Mumbai, India, 2014: 238-245

[14] Ding Yan, Wang Huai-Min, Shi Pei-Chang, Wu Qing-Bo, Dai Hua-Dong, Fu Hong-Yi. Trusted cloud services. Chinese Journal of Computers, 2015, 38(1): 133-149(in Chinese)
(丁滟, 王怀民, 史佩昌, 吴庆波, 戴华东, 富弘毅. 可信云服务. 计算机学报, 2015, 38(1): 133-149)

[15] Tatbul N. Streaming data integration: Challenges and opportunities//Proceedings of the IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010). California, USA, 2010: 155-158

[16] Richard T, Steven Y, Rob M, David R. StreamBase LiveView. California, USA: StreamBase Inc, Technical Paper, 2012

[17] Hentschel M, Kossmann D, Florescu D, et al. Scalable data integration by mapping data to queries. ETH Zurich, Switzerland: Technical Report 633, 2009

[18] Zheng Y, Liu F, Hsieh H-P. U-Air: When urban air quality inference meets big data//Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining ACM. Chicago, USA, 2013: 1436-1444

[19] Yuan J, Zheng Y, Xie X. Discovering regions of different functions in a city using human mobility and POIs//Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Beijing, China, 2012: 186-194

[20] Zheng Y, Capra L, Wolfson O, Yang H. Urban Computing: Concepts, methodologies, and applications. ACM Transactions on Intelligent Systems and Technology, 2014, 5(3): 38

[21] Dindar N, Tatbul N, Miller R J, et al. Modeling the execution semantics of stream processing engines with SECRET. The VLDB Journal, 2013, 22(4): 421-446

[22] Lim H, Babu S. Execution and optimization of continuous windowed aggregation queries//Proceedings of the 2014 IEEE 30th International Conference on Data Engineering Workshops. Chicago, USA, 2014: 303-309

[23] Botan I, Cho Y, Derakhshan R, et al. A demonstration of the MaxStream federated stream processing system//Proceedings of the 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010). California, USA, 2010: 1093-1096

[24] Hirzel M, Soulé R, Schneider S, et al. A catalog of stream processing optimizations. ACM Computing Surveys, 2014, 46(4): 46

[25] Kim H G, Park Y H, Cho Y H, Kim M H. Time-slide window join over data streams. Journal of Intelligent Information Systems, 2014, 43(2): 323-347

[26] Viglas S D, Naughton J F, Burger J. Maximizing the output rate of multi-way join queries over streaming information sources//Proceedings of the 29th International Conference on Very Large Data Bases-Volume 29: VLDB Endowment. Berlin, Germany, 2003: 285-296

[27] Naeem M A. Efficient Joins to Process Stream Data[Ph. D. dissertation]. The University of Auckland, Auckland, 2012

[28] Chakraborty A, Singh A. Parallelizing windowed stream joins in a shared-nothing cluster//Proceedings of the 2013 IEEE International Conference on: Cluster Computing (CLUSTER). Indianapolis, USA, 2013: 1-5

[29] Ananthanarayanan R, Basker V, Das S, et al. Photon: Fault-tolerant and scalable joining of continuous data streams categories and subject descriptors//Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. New York, USA, 2013: 577-588

[30] Lim H, Babu S. Execution and optimization of continuous queries with cyclops//Proceedings of the 2013 International Conference on Management of Data-SIGMOD'13. New York, USA, 2013: 1069

[31] Li J, Maier D, Tufte K, et al. No pane, No gain: Efficient evaluation of sliding-window aggregates over data streams. ACM Special Interest Group on Management of Data (SIGMOD) Record, 2005, 34(1): 39-44

[32] Balkesen C, Tatbul N. Scalable data partitioning techniques for parallel sliding window processing over data streams//Proceedings of the International Workshop on Data Management for Sensor Networks (DMSN). Seattle, USA, 2011

[33] Jeon J H, Lee K Y, Kim M H. Communication-efficient processing of multiple continuous aggregate queries. Information Sciences, 2014, 284: 1-17

[34] Wang Guang-Dong, Wang Yi-Jie, Li Xiao-Yong, Wang Yuan. Parallel Skyline computation over uncertain data streams. Journal of Frontiers of Computer Science and Technology, 2012, 6(12): 1116-1125(in Chinese)
(王广东, 王意洁, 李小勇, 王媛. 不确定数据流上的并行Skyline查询算法. 计算机科学与探索. 2012, 6(12): 1116-1125)

[35] Wu S, Kumar V, Wu K-L, Ooi B C. Parallelizing stateful operators in a distributed stream processing system: How, should you and how much?//Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems. Berlin, Germany, 2012: 278-289

[36] Li Xiaoyong, Wang Yijie, Zhao Yu, Wang Yuan, Li Xiaoling. GPS: A general framework for parallel queries over data streams in cloud//Proceedings of the 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC). Zhangjiajie, China, 2013: 1139-1146

[37] Li X, Wang Y, Li X, Wang Y. Parallel skyline queries over uncertain data streams in cloud computing environments. International Journal of Web and Grid Services, 2014, 10(1): 24-53

[38] Li X, Wang Y, Li X, Wang Y. Parallelizing skyline queries over uncertain data streams with sliding window partitioning and grid index. Knowledge and Information Systems, 2014, 41(2): 277-309

[39] Carey M J, Onose N, Petropoulos M. Data services. Communications of the ACM, 2012, 55(6): 86-97

[40] Curbera F, Duftler M, Khalaf R, et al. Unraveling the Web services web: An introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing, 2002, 6(2): 86-93

[41] Dustdar S, Pichler R, Savenkov V, Truong H-L. Quality-aware service-oriented data integration: Requirements, state of the art and open challenges. ACM Special Interest Group on Management of Data (SIGMOD) Record, 2012, 41(1): 11-19

[42] Wang Guiling, Yang Shaohua, Han Yanbo. Mashroom: End-user mashup programming using nested tables// Proceedings of the 18th International Conference on World Wide Web. Madrid, Spain, 2009: 861-870

[43] CCF Services Computing Committee. Data services in the cloud: Present and future//China Computer Federation. 2011 Technical Development Report of Chinese Computer Science. Beijing: China Machine Press, 2012: 108-136 (in Chinese)
(中国计算机学会服务计算专业委员会. 云计算中的数据服务：现状与趋势//中国计算机学会. 2011 中国计算机科学技术发展报告. 北京：机械工业出版社，2012：108-136)

[44] Dickerson R, Lu Jiakang, Lu Jian, Whitehouse K. Stream feeds — An abstraction for the world wide sensor web// Proceedings of the 1st International Conference on IOT 2008. Zurich, Switzerland, 2008: 360-375

[45] He J, Zhang Y, Huang G, Cao J. A smart Web service based on the context of things. ACM Transactions on Internet Technology, 2012, 11(3): 1-23

[46] Guinard D, Trifa V, Karnouskos S, et al. Interacting with the soa-based Internet of Things: Discovery, query, selection, and on-demand provisioning of web services. IEEE Transactions on Services Computing, 2010, 3(3): 223-235

[47] Guinard D, Trifa V, Pham T, Liechti O. Towards physical mashups in the web of things//Proceedings of the IEEE 6th International Conference on Networked Sensing Systems (INSS). Pittsburgh, USA, 2009: 1-4

[48] Chu V W, Wong R K, Liu W, et al. Traffic analysis as a service via a unified model//Proceedings of the 2014 IEEE International Conference on Services Computing (SCC). Anchorage, USA, 2014: 195-202

[49] 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): 7:1-7:11

[50] Zaharia M, Das T, Li H, et al. Discretized streams: A fault-tolerant model for scalable stream processing. Electrical Engineering and Computer Sciences, University of California at Berkeley, California: Technical Report UCB/EECS-2012-259, 2012

[51] Arasu A, Babu S, Widom J. CQL: A language for continuous queries over streams and relations//Proceedings of the 9th International Workshop on Database Programming Languages. Potsdam, Germany, 2004: 1-19

[52] Hummer W, Satzger B, Dustdar S. Elastic stream processing in the cloud. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2013, 3(5): 333-345

[53] Ishii A, Suzumura T. Elastic stream computing with clouds //Proceedings of the 2011 IEEE International Conference on Cloud Computing (CLOUD). Washington, USA, 2011: 195-202

[54] Kleiminger W, Kalyvianaki E, Pietzuch P. Balancing load in stream processing with the cloud//Proceedings of the 2011 IEEE 27th International Conference on Data Engineering Workshops (ICDEW). Hannover, Germany, 2011: 16-21

[55] Neumeyer L, Robbins B, Nair A, Kesari A. S4: Distributed stream computing platform//Proceedings of the 2010 IEEE International Conference on Data Mining Workshops (ICDMW). Sydney, Australia, 2010: 170-177

[56] Gulisano V, Jimenez-Peris R, Patino-Martinez M, et al. StreamCloud: An elastic and scalable data streaming system. IEEE Transactions on Parallel and Distributed Systems, 2012, 23(12): 2351-2365

[57] Satzger B, Hummer W, Leitner P, Dustdar S. Esc: Towards an elastic stream computing platform for the cloud//Proceedings of the 2011 IEEE International Conference on Cloud Computing (CLOUD). Washington, USA, 2011: 348-355

[58] Akidau T, Balikov A, Bekiroğlu K, et al. MillWheel: Fault-tolerant stream processing at internet scale. Proceedings of the VLDB Endowment, 2013, 6(11): 1033-1044

[59] Castro Fernandez R, 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. New York, USA, 2013: 725-736

[60] Wu Y, Tan K-L. ChronoStream: Elastic stateful stream computation in the cloud//Proceedings of the IEEE 31st International Conference on Data Engineering. Seoul, Korea, 2015: 723-734

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

[62] Lakshmanan G T, Li Y, Strom R. Placement strategies for Internet-scale data stream systems. IEEE Internet Computing, 2008, 12(6): 50-60

[63] Wolf J, Bansal N, Hildrum K, et al. SODA: An optimizing scheduler for large-scale stream-based distributed computer systems//Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware. Leuven, Belgium, 2008: 306-325

[64] Kalyvianaki E, Wiesemann W, Vu Q H, et al. SQPR: Stream query planning with reuse//Proceedings of the 2011 IEEE 27th International Conference on Data Engineering. Hannover, Germany, 2011: 840-851

[65] Aniello L, Baldoni R, Querzoni L. Adaptive online scheduling in storm//Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems (DEBS'13). Arlington, USA, 2013: 207-218

[66] Kulkarni S, Bhagat N, Fu M, et al. Twitter Heron: Stream processing at scale//Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. Melbourne, Australia, 2015: 239-250

[67] Zeitler E, Risch T. Massive scale-out of expensive continuous queries. Proceedings of the VLDB Endowment, 2011, 4(11): 1181-1188

[68] Gedik B, Schneider S, Hirzel M, Wu K-L. Elastic scaling for data stream processing. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(6): 1447-1463

[69] Collins R L, Carloni L P. Flexible filters: Load balancing through backpressure for stream programs//Proceedings of the 7th ACM International Conference on Embedded Software. Atlanta, USA, 2009: 205-214

[70] Schneider S, Andrade H, Gedik B, et al. Elastic scaling of data parallel operators in stream processing//Proceedings of the IEEE International Symposium on Parallel & Distributed Processing. Rome, Italy, 2009: 1-12

[71] Backman N, Fonseca R, Çetintemel U. Managing parallelism for stream processing in the cloud//Proceedings of the 1st International Workshop on Hot Topics in Cloud Data Processing. Bern, Switzerland, 2012: 1-5

[72] Ding J, Fu T Z, Ma R T, et al. Optimal operator state migration for elastic data stream processing. arXiv preprint arXiv:1501.03619, 2015

[73] Best M J, Mottishaw S, Mustard C, et al. Synchronization via scheduling: Techniques for efficiently managing shared state. ACM SIGPLAN Notices, 2011, 46(6): 640-652

[74] Repantis T, Kalogeraki V. Replica placement for high availability in distributed stream processing systems//Proceedings of the 2nd International Conference on Distributed Event-Based Systems. Rome, Italy, 2008: 181-192

[75] Hwang J-H, Xing Y, Cetintemel U, Zdonik S. A cooperative, self-configuring high-availability solution for stream processing //Proceedings of the IEEE 23rd International Conference on Data Engineering. Istanbul, Turkey, 2007: 176-185

[76] Zhang Z, Gu Y, Ye F, et al. A hybrid approach to high availability in stream processing systems//Proceedings of the IEEE 30th International Conference on Distributed Computing Systems (ICDCS). Genoa, Italy, 2010: 138-148

[77] Gu Y, Zhang Z, Ye F, et al. An empirical study of high availability in stream processing systems//Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware. Urbanna, USA, 2009: 1-9

[78] Condie T, Conway N, Alvaro P, et al. MapReduce online//Proceedings of NSDI 2010. San Jose, USA, 2010: 20-35

[79] Ding W, Han Y, Wang J, Zhao Z. Feature-based high availability mechanism for extreme aggregation tasks in real-time data stream processing. Journal of Internet Technology, 2013, 14(2): 327-340

[80] Shah M A, Hellerstein J M, Chandrasekaran S, Franklin M J. Flux: An adaptive partitioning operator for continuous query systems//Proceedings of the International Conference on Data Engineering (ICDE). Bangalore, India, 2003: 25-36

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

[82] Balazinska M, Balakrishnan H, Madden S R, Stonebraker M. Fault-tolerance in the Borealis distributed stream processing system. ACM Transactions on Database Systems, 2008, 33(1): 1-44

[83] Branson M, Douglis F, Fawcett B, et al. CLASP: Collaborating, autonomous stream processing systems//Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware. Newport Beach, USA, 2007: 348-367

[84] Jin Che-Qing, Qian Wei-Ning, Zhou Min-Qi, Zhou Ao-Ying. Benchmarking data management systems: From traditional database to emergent big data. Chinese Journal of Computers, 2014, 37(8): 1-18(in Chinese)
(金澈清, 钱卫宁, 周敏奇, 周傲英. 数据管理系统评测基准: 从传统数据库到新兴大数据. 计算机学报, 2014, 37(8): 1-18)

[85] Arasu A, Cherniack M, Galvez E, et al. Linear road: A stream data management benchmark//Proceedings of the 30th International Conference on Very Large Data Bases-Volume 30: VLDB Endowment. Toronto, Canada, 2004: 480-491

[86] Huang S, Huang J, Dai J, Xie T, Huang B. The HiBench benchmark suite: Characterization of the MapReduce-based data analysis//Proceedings of the IEEE 26th International Conference on Data Engineering Workshops (ICDEW). California, USA, 2010: 41-51

[87] Ghazal A, Rabl T, Hu M, et al. BigBench: Towards an industry standard benchmark for big data analytics//Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. Chicago, USA, 2013: 1197-1208

[88] Wang L, Zhan J, Luo C, et al. Bigdatabench: A big data benchmark suite from Internet services//Proceedings of the 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA), Orlando, USA, 2014: 488-499

[89] Düntgen C, Behr T, Güting R. BerlinMOD: A benchmark for moving object databases. The VLDB Journal, 2009, 18(6): 1335-1368

WANG Gui-Ling, born in 1978, Ph.D., associate professor. Her research interests include service computing, service-oriented data integration, large-scale stream data processing and integration.

HAN Yan-Bo, born in 1962, Ph. D., professor, Ph. D. supervisor. His current research interests include Internet and cloud computing, services interoperability and composition, dependable distributed systems, large-scale stream data integration and analysis.

ZHANG Zhong-Mei, born in 1990, Ph. D. candidate. Her research interests include services computing, stream data integration and analysis.

ZHU Mei-Ling, born in 1987, Ph. D. candidate. Her research interests include services computing, stream data integration and analysis.

Background

Data is the foundation of the computer application systems. Data management and processing is the focus of the Cloud infrastructure. With the coming of "The Age of Big Data" and the wider utilization of the Cloud infrastructure, the issue of stream data processing and integration from various censors has become a focus of study. However, many issues in this area still in its infancy, or even not addressed yet. This paper starts with the application requirements of large-scale stream data processing and integration, discusses the state of the art of the key technologies including stream data model, stream data integration, stream data service modeling, query processing and optimization, the scalability and reliability mechanism, evaluation metrics and benchmark etc. Finally some new challenges in the future are summarized.

This research was partially supported by the Beijing Natural Science Foundation (No. 4131001), Project of Construction of Innovative Teams and Teacher Career Development for Universities and Colleges Under Beijing Municipality (No. IDHT20130502), the Beijing Natural Science Foundation Program and Scientific Research Key Program of Beijing Municipal Commission of Education (No. KZ201310009009).