计算机科学 2005Vol. 32№. 4

数据流管理技术

刘学军[1,2] 徐宏炳[1] 董逸生[1] 王永利[1] 钱江波[1]

(东南大学计算机科学与技术系 南京 210096)[1] (南京工业大学信息科学与工程学院 南京 210009)[2]

摘 要 最近,人们已经广泛认识到:在某些新的应用领域中,把数据看作瞬时的数据流比看作持久的关系更为适合。本文首先分析了传统数据库管理系统处理数据流的局限性,然后分析了三个典型的数据流管理系统的基本实现技术,讨论了当前数据流管理技术的研究现状和今后的研究方向,最后,给出了一个数据流管理原型系统的体系结构。

关键词 数据流,连续查询,资源管理,系统调度,数据挖掘

Data Stream Management Techniques

LIU Xue-Jun[1,2] XU Hong-Bing[1] DONG Yi-Sheng[1] WANG Yong-Li[1] QIAN Jiang-Bo[1]

(Department of Computer Science and Technology, Southeast University, Nanjing 210096)[1]

(College of Information Science and Engineering, Nanjing University of Technology, Nanjing 210009)[2]

Abstract Recently a new class of data-intensive applications has become widely recognized: it is appropriate to treat high-speed time-changing data in some application as transient data streams rather than as persistent relations. This paper analyzes the limitation of the traditional Database Management System in aspect of dealing with data streams firstly, then the basic technology of three representative Data Stream Management Systems is mainly introduced. Furthermore, this paper reviews the recent studies for data stream techniques, including the research directions and research emphases of data stream techniques. Lastly the overview architecture of our Data Stream Manage Prototype System is given briefly.

Keywords Data stream, Continuous queries, Resource management, System scheduling, Data mining

1 引言

随着 Internet 技术和传感器网络的发展,在越来越多的实际应用中,需要处理大量、连续、快速、随时间变化的数据。这些数据连续到达,频繁变化,到达的速度也可能突然发生变化,数据的更新通常以插入为主,数据量事前是不确定的,也可能是无限的,我们把这种数据称为数据流。通常,数据流按照固定的次序,只能被读取一次或几次。与传统数据库不同,数据流中的数据是以一种流的形式源源不断地进入系统,不再是永久的关系形式,数据无法在全部保存之后再进行处理。因此,许多已有的数据库技术难以应用于数据流,这就需要发展新的数据处理技术。

数据流已经出现在许多应用中,主要应用领域可以概括如下:传感器网络、电信通话记录、气象监测与分析、移动物体位置跟踪、股票分析、邮件过滤、网络监控与安全、网络日志分析、生产制造过程控制等。

目前,国外已经在这方面开展了大量的研究工作,取得了一定的成果,并且开发了一些数据流管理系统原型。近几年来,著名的国际会议 VLDB 和 ACM SIGMOD 等都将数据流作为一个重点问题来讨论,数据流管理技术正成为国际数据处理领域的一个热点问题。研究数据流管理技术不仅有重要的理论意义,而且,有着巨大的应用前景,已经引起了越来越多的学者的重视。但是,国内在这方面的研究刚刚起步,还很少有相关的研究报道。

传统的数据库管理系统在处理数据流时,表现出极大的局限性,主要体现在:

1) 通常的数据流应用中(如网络监控),当反常的行为出现时,需要自动报警(或触发某种动作),这是一种数据库主动、人被动的模式(简称 DAHP 模式)。而传统的数据库管理系统本质上是为处理事务型数据而设计的,是一种数据库被动、人主动的模式(简称 HADP 模式)。

2)数据流的查询是连续查询,即当新数据到达时所有当前活动的查询被执行,而查询一经注册,连续执行,直至取消为止。数据流中的查询静态存储在数据库中,数据是动态的。而传统关系数据库中数据是静态的,用户使用动态的查询语句去查静态的数据,这两种形式是截然不同的。由此可见,传统关系数据库本质上不支持连续查询。另外,传统数据库是随机获取数据的,而数据流通常是顺序获取的,随机存取代价昂贵。

3)传统的数据库管理系统假定数据是同步的,对数据的查询得到的是准确的结果。而数据流通常是异步到达的,经常需要在不完整的信息下,提供近似的回答。另一方面,数据流具有无限连续性,随着时间的推移,数据的容量可以认为是无限大的,受到有限资源的限制,数据无法在全部保存之后再进行处理,这时,也需要提供近似的查询技术。显然,传统的数据库管理系统并没有提供内建的功能来支持近似查询操作。

4)在数据流的许多应用中(如传感器网络),需要大量的触发器。而在传统的数据库管理系统中,触发器和报警器是处

刘学军 博士研究生,主要研究兴趣为数据流管理、数据挖掘、进化计算等。徐宏炳 教授,研究生导师,研究领域包括数据库、语义 Web 和信息处理与安全等。董逸生 教授,博士生导师,研究领域包括数据库、信息系统和软件工程等。王永利 博士研究生,主要研究兴趣为数据流管理、信息系统等。钱江波 博士研究生,主要研究兴趣为数据流管理、数据挖掘等。

于次要位置的，由于不能衡量每个表上的多个发生过的触发器而无法实现大数量的触发器。

5)数据流的无限流动性使得数据无法在全部保存之后再进行处理，对同一数据的可访问次数也是有限的，因此，需要以数据的本源形式——流的形式实时在线处理数据。数据流是源源不断的，所以对数据流中的每条记录，处理时间必须很短，实现高速数据处理自然成为数据流系统的关键。所有这些，都对传统的数据库管理系统提出了挑战。

上述的局限性，使得传统的数据库管理系统很难处理数据流，因此，数据流管理系统应运而生。目前，最典型的原型系统主要有 stanford 大学的 Stanford Stream Data Manager(简称 STREAM)、Berkeley 的 Telegraph 系统及 Brandeis University、Brown University 和 M. I. T. 联合开发的 Aurora 系统。其它数据流系统主要包括 Tapestry、Tribeca 、Niagara、COUGAR、ATLaS、Fjords、hancock 等，见文[8～16]。

本文首先简要地分析了 STREAM 系统、Aurora 系统和 Telegraph 系统的 Psoup 查询处理器的实现技术，讨论了数据流管理技术研究现状和今后的研究方向，最后给出了一个数据流原型系统的总体结构。

2 典型原型系统基本实现技术分析

2.1 STREAM 基本实现技术

STREAM 系统是由 Stanford 大学开发的，其目标是建立一个既可以处理流速非常高的数据，又能处理成千上万连续查询的通用数据流管理系统[1,2]。在数据的流速和查询负载超出可获得资源的情况下，系统优化内部配置，可以为连续的查询提供相对准确的近似结果。系统内部的多查询优化策略、有效的资源分配算法和灵活的调度策略保证了系统的高性能。有限资源和结果近似性的自动平衡是系统最重要的关键技术。

STREAM 系统中，查询首先被注册，随着新数据的到来，查询不停地被执行。CQL (Continuous Query Language)是 STREAM 系统的标准查询语言，它既支持传统的关系操作，又支持流操作。CQL 扩展了 SQL 语言，主要表现在 From 子句。From 子句包含一个可选的滑动窗口和一个取样子句。滑动窗口由分割子句(Partition by)、窗口大小和可选的过滤谓词组成。分割子句把数据分为几组，在窗口内计算每组的值，然后，合并结果，类似于标准 SQL 的 Group by。窗口大小由 row 或 range 确定，如：range 15 minutes preceding。取样子句定义了取样的百分比，如：sample(2)。下面是两个查询的实例：①Select Count (*) From Requests S [Partition By S. client_id Rows 10 Preceding Where S. domain = 'seu'];② Select Count(*)From Requests S [Range 1 Day Preceding] Where S. domain = 'seu'。

对于无限的数据流，STREAM 系统通过滑动窗口将其转化为有限的关系元组集合。系统又定义了一些新的运算符，如 Istream(插入流) 和 Dstream(删除流)，通过它们将关系元组转化为流，为用户提供连续的查询结果。

STREAM 系统中，每一个查询都有一个独立的查询计划。查询计划由三种不同的结构组成：查询运算符(query operators)、操作队列(inter-operator queues)、大纲(synopses)。一般来说，队列是内存中用来存储数据的一块缓冲区，有最大容量限制，因此，当数据量超出缓冲区的最大容量时，旧数据被丢弃，以便容纳新数据。队列可以为多个运算符提供数据源。在传统的关系数据库中，两个关系的连接操作需要多次扫描关系表，而数据流具有瞬时性，数据不停地被更新，为了实现上述功能，STREAM 系统定义了大纲数据结构，用于暂时存储数据。当然，大纲不仅仅用于连接操作。大纲与队列的重要区别是：大纲是为某种操作服务的，容量的大小也通常根据操作的需要而决定。

下面举一个查询的例子，如图 1。R、S 和 T 为三个数据流，o_1 和 o_3 为连接运算符，o_2 为投影运算符，S_1、S_2、S_3 为大纲，q_1、q_2、q_3、q_4 为队列。数据流 R、S 和 T 进入各自的队列 q_1、q_2、q_3，暂时存储。O_1 从 q_1 和 q_2 两个队列中取得数据，分别放入大纲 S_1、S_2，实现连接操作。操作后的数据存入队列 q_3，队列 q_3 为 O_2 和 O_3 提供数据源。O_2 从队列 q_3 中取得数据，处理后得到结果 Q_1。O_3 从队列 q_3 和 q_4 中取得数据，分别放入大纲 S_3、S_4，实现连接操作，结果为 Q_2。为合理有效地利用有限的资源，多查询优化和资源的共享需要重点考虑。例如，在上述查询过程中，q_3 为 O_2 和 O_3 所共享，合理地安排 O_2 和 O_3 的执行速度，可以减少 q_3 和 q_4 所占用的存储空间。

有效的资源管理是数据流管理系统的关键技术之一。STREAM 系统中，重点关注内存的管理，主要采用了两种技术：根据统计信息和查询注册信息，对大纲进行压缩；采用优化的调度策略，降低队列所占用的存储空间。在数据的流速和查询负载超出可获得资源的情况下，可以为连续的查询提供相对准确的近似结果是 STREAM 系统的一个重要特点。那么，在有限的资源下，如何获得最佳的查询结果呢？STREAM 系统主要采用了静态近似策略和动态近似策略。动态近似是更具有挑战性的课题，比静态近似具有更多的优越性。STREAM 系统中最主要的资源消耗是大纲和队列。因此，主要的动态近似方法有：采用柱状图、小波压缩等技术实现大纲的压缩；采用动态取样、数据流丢弃等技术减少队列所占用的存储空间。有限资源和结果近似性的自动平衡是 STREAM 系统最重要的关键技术。

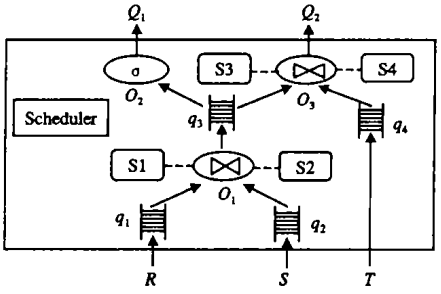

图 1 一个查询的例子[2]

应该指出，STREAM 系统还有许多不完善之处，它是一个基于关系模型的集中式数据库系统，而对许多数据流的应用，分布式处理更为重要，因此，Stanford 大学正计划把 STREAM 升级为分布式系统。查询计划的形成策略还有待进一步研究，资源分配算法还不能很好地支持近似回答，已经设计了查询语言，而目前仅仅实现了一个子集，系统如何适应数据流速和数据流量猝然变化的情况，所有这些，都是今后的研究重点和研究方向。

2.2 Psoup 基本实现技术

Psoup 是 Berkeley 的 Telegraph 系统的查询处理器[3,4]。查询首先要被注册，注册后，返回给用户一个句柄，接下来，不停地执行查询请求。查询语句的一般模式是：SELECT select-list FROM from- list WHERE conjoined boolean factors BEGIN begin time END end time。

查询被分为两部分：SELECT-FROM-WHERE 子句和 BEGIN-END 子句。SELECT-FROM-WHERE 子句为标准查

询子句(SQC),存储在数据结构 query steM 中。满足 SQC 的元组记录在 Result Structure 中。BEGIN-END 表示了一个滑动窗口,该子句存储在一个独立的结构 WindowsTable 中。新数据流进入系统后,被分配一个全局的元组 ID(称为 tupleID)和物理时间戳(称为 physicalID),然后,插入到数据结构 Data steM 中。每一个数据流有一个 Data steM。新数据元组主动检索 query steM,满足 SQC 的元组记录到 Result Structure 中,Result Structure 中的元组包含元组的 tupleID 和 physicalID。新数据元组也可以检索其它 Data steM,实现连接等操作。注册一个查询后,被分配一个唯一的 ID 号(queryID),标准查询子句插入 query steM 中。新查询主动检索 Data steM,满足条件的元组记录到 Result Structure 中。由此可见,Psoup 将数据和查询都作为流来处理。最终查询结果的获得是根据 BEGIN-END 子句检索 Result Structure,获取 tupleID 后,根据 tupleID 从 Data steM 中得到元组。查询的执行过程如图 2 所示。

由上述分析可以看出,Data SteM、Query SteM 和 Result Structure 是 Psoup 中最重要的三种数据结构。受篇幅限制,我们仅做简要的介绍。每个数据流有一个 Data SteM 数据结构,用于存储和索引数据流,Data SteM 采用基于树的索引结构,每个属性建一棵树。整个系统有一个 Query SteM 结构,用于存储和索引查询,树结构用于单属性索引,多属性索引采用链表结构。Result Structure 用于记录结果元数据,这些元数据是用来表示满足 SQC 的元组,常用的结构有两种:①二维表结构,每个 From-list 有一个独立的二维表结构,行根据元组物理时间戳来排序,列根据查询 ID 来排序;②每个查询带有一个链表,包含所有满足查询条件的元组。

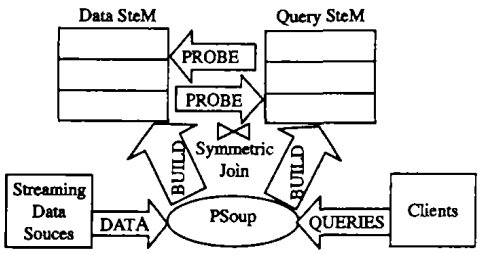

图 2 查询的执行[4]

从 Psoup 的基本实现技术可以看出,Psoup 将数据和查询都作为流来处理,因此,新数据可以主动检索已注册的旧查询,新查询也可以检索已存在的旧数据,而其他数据管理系统要么将数据作为流,要么将查询作为流,不能将两者同时作为流来处理。Psoup 将查询和结果的传送分开处理,查询在服务器端不停地被执行,结果记录在 Result Structure 中,客户端通过 Result Structure 获得结果,因此,客户端短时离线也不会影响查询结果。

目前,Psoup 仅支持内存操作,它的今后研究方向是支持磁盘上的流存储和研究物化视图与滑动窗口之间的关系。

2.3 Aurora 基本实现技术

Aurora 是由 Brandeis University、Brown University 和 M.I.T. 联合开发的数据流管理系统,它支持大数量的触发器,因此,又被称为触发器网络[5~7]。

系统结构如图 3 所示。系统的最基本结构是 box 和 arrow,每个 box 是一个处理单元,arrow 反映了数据的流动方向和处理单元处理数据的先后次序。查询是通过图形化的用户界面,以 box 和 arrow 形式实现的。用户也可以用扩展的 SQL 语句实现查询,但是,扩展的 SQL 将被系统编译成 box 和 arrow 形式。这是 Aurora 与其它数据流管理系统的显著不同。

Aurora 系统定义了专用的运算符,以完成各种查询需要,主要包括 Filter、Drop、Union、Wsort、Tumble、Windowed、Slide、Latch、Resample、Map、GroupBy 和 Join 等。Filter,对输入流过滤;Union,将多个输入流合并为一个输出流;Wsort,缓冲所有输入流,并按属性排序输出;GroupBy,按属性分组;Join,连接操作;Windowed,截取一段数据流,即固定时间窗口;Slide,滑动窗口;Resample,再取样;Map,映射;Tumble 和 Latch 运算符比较复杂,可参见文[6]。

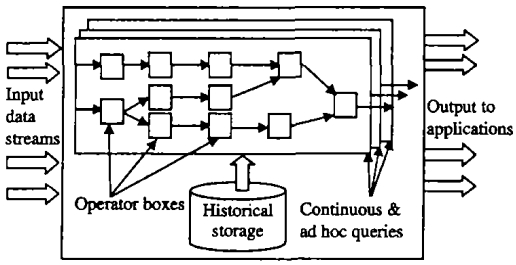

图 3 Aurora 系统结构[6]

Aurora 系统支持三种查询模式:连续查询(实时处理)、视图和 ad 查询,表现为三层结构,如图 4。$S1$、$S2$ 和 $S3$ 为元组队列,b_1 至 b_9 为处理单元 box,app 表示应用,QoS 是服务质量,图中的实心圆点代表连接点。第一层结构为连续查询,该层的数据被实时处理后,不保存。服务质量规范(QoS specification)决定资源如何分配给各个处理单元。通常,一个用户查询被分解为若干个 box,box 由连接点处增加或删除,连接点可以将数据暂存一段时间。第二层结构是视图,是在调度器的控制下物化或部分物化的视图。通过视图可以加快查询的速度,服务质量规范说明各个视图的重要程度。第三层结构为 ad 查询,ad 查询可以在任意时刻附加到连接点,实现对历史数据的查询。

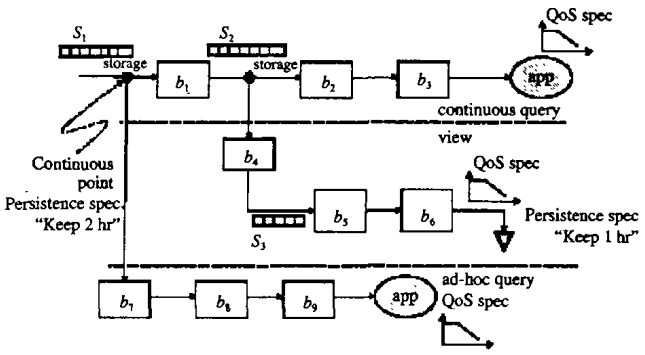

图 4 Aurora 查询结构[5]

系统总体结构如图 5,主要包括路由器、调度器、存储管理器、服务质量监控器、负载均衡器和 box 处理器。来自外部数据源或其它 box 的输出数据进入路由器,路由器继续将数据或者发送给外部应用程序,或者存到存储管理器中相应的队列。存储管理器负责维护 box 队列和管理缓存。调度器按照一定的策略选择 box,并将其传送给多线程 box 处理器。box 处理器执行相应的操作,然后输出元组给路由器。调度器接着选择下一个 box,循环往复。服务质量监控器连续不断地监控系统的执行性能,在监测到过载状态和系统性能低下时激活负载均衡器。负载均衡器分流负载或丢弃部分元组,保证系统性能达到可接受的水平。服务质量(QoS)由应用管理员根据实际应用的要求来确定。

归纳起来，Aurora 系统具有如下特点：1)DAHP 模式(DBMS-Active,Human-Passive)；2)适宜处理时间序列信息，便于获取数据的新值和旧值；3)触发器居于主导地位，能够有效实现大数量的触发器；4)能够根据负荷和资源的情况，实现精确查询或近似查询；5)具有实时性，可在线处理数据。因此，Aurora 系统的主要应用环境是：处理的对象是数据流；需要大量的触发器；有较高的实时性要求；以非精确的数据处理为主或近似查询应用为主。

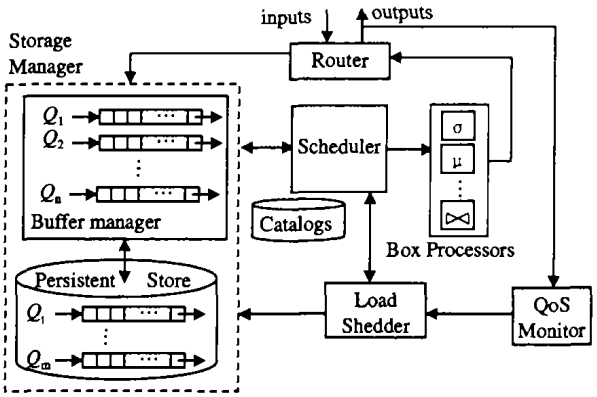

图 5 Aurora 系统总体结构[7]

对许多数据流的应用，其本质上是分布式的，因此，开发分布式的数据流管理系统是十分必要的。Aurora* 是以 Aurora 为基础的分布式数据流管理系统。受篇幅限制，我们不做进一步的介绍，更详细的情况请参见文[6]。

3 数据流管理技术研究热点和发展趋势

目前，数据流技术的研究热点主要集中在连续查询语言设计、查询优化、资源管理(主要是内存)、近似查询操作、流系统的调度、数据流管理系统的开发和针对数据流的数据挖掘算法研究。

传统的查询语言不适宜处理数据流，因此，设计、开发针对数据流的连续查询语言成为数据流技术的一个重点内容。目前的主要实现技术有两种：SQL 语言的扩展和设计专用的流处理运算符。STREAM 和 Psoup 是通过扩展 SQL 语句来实现数据流查询的，而在具体技术细节上又有所不同。STREAM 系统对 SQL 语句的扩展主要表现在 From 子句，From 子句包含一个可选的滑动窗口和一个取样子句。Psoup 则通过增加一个 BEGIN-END 子句来实现扩展，而且，SELECT-FROM-WHERE 子句和 BEGIN-END 子句分开存储。Upson Hall Cornell 大学的 COUGAR 系统[10]和 IBM 公司等开发的 ATLaS 系统[8](Aggregate & Table Language and System)也是通过扩展 SQL 语言实现对数据流的处理的。COUGAR 系统采用面向对象技术，通过定义抽象数据类型和抽象数据类型函数来实现 SQL 语言的扩展。ATLaS 系统允许用户定义集合运算(UDAs)和表函数，支持面向数据流的计算模型，ATLaS 的基本扩展机制可以为许多应用领域带来益处，如 OLAP，递归查询，数据挖掘等。Tapetry[13]是为过滤 Internet 上的 email 和新闻信息而开发的，它通过将 SQL 语句转化为与之对应的增量查询来实现流数据的处理。Aurora 系统的处理技术与它们截然不同，它是通过专门设计的运算符来实现查询的，为了方便使用，用户也可以输入扩展的 SQL 语句，但是，最终将被编译成系统专用的运算符。不难看出，通过对 SQL 语言的扩展来实现连续查询语言是目前的主流方向，这是因为 SQL 语言不仅功能强大，而且已经成为数据库的标准语言。

在过去的二十年中，查询优化一直是数据库领域的一个重要研究课题。现代查询优化技术大多数是基于代价估算的优化，这对于传统的数据库管理系统是很有效的。文[22,25]等指出传统的代价估算在数据流的应用中难以实现，这就需要发展新的查询优化技术，因此，他们提出了基于速度的查询优化方法，其目标是获得最大的查询输出率，并给出了一个优化框架结构。STREAM 系统的查询优化侧重于减少内存的消耗，因此，多查询优化和资源的共享被重点考虑。合理调度查询计划，减少队列所占用的存储空间和实现队列的共享都是行之有效的方法。Aurora 系统主要以处理元组的执行时间作为评价指标。在 Aurora 系统中，大量的 box、高速的数据流速使得传统的优化方法不再适用。Aurora 系统将整个网络(系统)划分为若干个子网，每个子网单独优化，这样，某个子网优化时，不会影响其余部分的执行和元组的输出，主要策略有：插入投影操作，预先除去不必要的属性；合并 box 的操作；调整 box 的执行顺序等。对于 ad 查询，既需要处理连接点处存储的历史数据，又要处理新进入系统的数据，因此构建两个子网，一个用于历史数据的查询优化，另一个用于新数据的查询优化。各个子网周期地执行优化操作，从而实现整个系统的查询优化。数据流的查询优化对提高查询效率具有显著的作用，因此它仍然是今后数据流技术的一个重点研究内容。

通常数据流系统具有较高的实时性要求，因此多数数据流系统是基于主存的。数据流是无限的，而系统的内存资源是有限的，所以有效的内存管理和内存分配策略自然成为数据流技术的一个重要研究内容。在 STREAM 系统中，采用优化的调度策略降低队列所占用存储空间和采用适当的算法对大纲进行压缩是两个重要的方法。目前，大纲压缩的常用方法有：随机采样、草图技术、柱状图、小波压缩等。文[23]研究了采用 k-约束算法来减少内存消耗的方法。在 Aurora 系统中，服务质量(QoS)决定着内存资源的分配，当系统内存资源无法满足需要时，激活负载均衡器，负载均衡器分流负载或丢弃数据元组以降低系统对内存的需求。有效地降低内存资源的消耗而又尽可能减少服务质量的下降是数据流系统的一个努力目标。

在传统关系数据库中，数据集相对固定，每次查询都得到精确的结果。而数据流系统中，由于数据流的无限性和系统资源的有限性，经常出现资源不足的情况，因此系统能够提供近似查询操作就十分必要了，这也是数据流技术中的一个重要的研究课题。STREAM 系统在资源不足的情况下，提出了两种近似操作技术：静态近似和动态近似。系统也采用了相应的算法，能够在有限的资源下，获得尽可能精确的查询结果。Aurora 系统能够自动平衡负荷和资源，根据负荷和资源的情况，实现精确查询或近似查询。取样、柱状图、小波和草图技术是常用的数据近似方法[20,21]。文[22]等研究了数据流连接中的查询操作，得出了一些重要的结论。有效的近似查询操作是数据流系统成功的关键，因此也将成为今后的一个研究重点。

流系统的调度直接关系到查询计划的执行和资源的分配，是数据流系统的核心问题之一。目前，数据流系统常用的调度算法有：环调度(Round-Robin)、先进先出(FIFO)、贪心调度(Greedy)、链调度(CHAIN)等。初始的 STREAM 系统主要采用 Round-Robin 调度算法和贪心调度算法，最主要目标是减少内存的消耗，但是，面对数据流速和流量突然变化的情况缺乏自适应性。最近，STREAM 系统发展了 CHAIN 调度算法，可以最小化不可预知的队列尺寸，适合突发式数据流到达模式[30]。在 Aurora 系统中，若干个 box 组合成一个 super-

box 作为调度和执行的基本单元[7]。系统采用两级调度，第一级调度决定哪些 superbox 被调度，第二级调度决定 superbox 内部 box 的执行次序。第一级调度有静态调度和动态调度两种，目前以静态调度为主，可以通过各种调度算法(如 Round-Robin)来实现。第二级调度采用三种调度策略：基于最大吞吐量的调度、基于最小延迟的调度和基于最小内存消耗的调度。在 Aurora 系统中，由于大规模、高动态的系统特征和调度的粒度等方面因素，寻找最优的调度方法是十分困难的，因此启发式方法常常是很有效的。Telegraph 是基于 Eddy 的调度。Eddy[29]使用的调度规则类似于 STREAM 系统中的贪心调度，不过，Eddy 调度的是元组，而不是调度运算符的执行。数据进入 Eddy，Eddy 发送元组给运算符，运算符作为一个独立的线程而执行，其结果再返回给 Eddy，元组被所有运算符处理后，Eddy 将其输出。Eddy 的目的是最大化吞吐量，不像 STREAM 关心的是队列的尺寸。

目前的数据流系统多数是集中式系统，而且专注于某一方面，缺乏通用性，限制多，使得应用范围狭窄。开发通用的、面向对象的分布式数据流管理系统将是今后数据流管理系统的发展方向。数据流具有突发性，新一代数据流管理系统应当能够自适应环境的变化，特别是猝然变化的数据流速和流量，因此自适应的数据流管理系统也是未来的发展方向。

最近，面向数据流的数据挖掘已经引起了人们广泛的关注。通常，数据流按照固定的次序，只能被读取一次或几次，因此与传统的数据挖掘算法不同，评价数据流的挖掘算法好坏的一个重要指标是：一次或有限几次的数据扫描次数就可以得到较好的结果。文[18]提出了适合数据流的决策树算法。文[19]则提出了数据流的聚类算法。文[17]采用改进的离散傅里叶算法研究了多个数据流的相关性问题。文[31,32]研究了频繁项集求解问题。美国电报电话公司设计了一种专门语言---hancock[11]，能够从大量的交易数据流中提取特征数据。文[24]采用数据立方技术研究了数据流的多维在线统计分析问题。面向数据流的数据挖掘技术尽管刚刚起步，但是，我们有理由相信它将成为数据挖掘领域的一个新热点。

4 一个数据流管理原型系统的体系结构

数据流的流动性和无限性以及计算机资源的有限性使得提高数据处理速度成为数据流管理系统的关键。在深入分析国外已有的数据流原型系统的基础上，我们提出了采用硬件预处理的并行数据流管理系统的体系结构。本系统不局限于通过查询优化、系统调度等方法来提高数据流的处理速度，而是考虑采用一种全新的体系结构来提高数据处理速度。并行数据流管理系统体系结构如图 6 所示。整个系统分为前端预处理和后端数据引擎。在数据引擎端，带有时间参数的连续查询经查询语法分析器分析后保存在注册查询缓存中，连续查询被解析和分解为多个操作算子，由并行调度器生成动态查询计划，依照查询计划执行策略将某些操作算子组成查询因子，经查询输出控制器下载到前端硬件查询预处理模块。在前端，对进入系统的数据流进行预处理(主要是数据过滤、数据压缩和数据加处理标记)，采用软硬件协同方式以提高数据处理速度。处理之后的数据流经通信网络发往后端数据引擎，经过数据流划分控制器，在并行调度器的控制下，根据用户查询请求所得出的调度策略动态划分数据流元组，按照查询的种类，所涉及的属性采用动态范围划分或散列划分策略将带有时间戳的元组分配到多个缓存中，在并行查询执行器的控制下由各自的处理器并行地完成查询处理。数据缓存借助两种类型的队列实现 Push(流数据)和 Pull(传统静态数据)操作，实现新数据对旧查询、新查询对旧数据的双重检测，各自查询结果经并行查询结果合成模块合成，输出最终查询结果。

系统后端的数据引擎主要负责连续查询的语法分析、并行查询计划生成、将预处理过的元组划分到不同的数据缓存，对并行处理得到的查询结果进行最后组装分发及负载均衡和服务质量监控等功能。系统前端的查询预处理层接收并行数据引擎端的查询请求，对多路采集来的模拟量或开关量进行本地预处理或部分计算，形成分布式查询代理，将预处理结果送交至后端查询引擎。

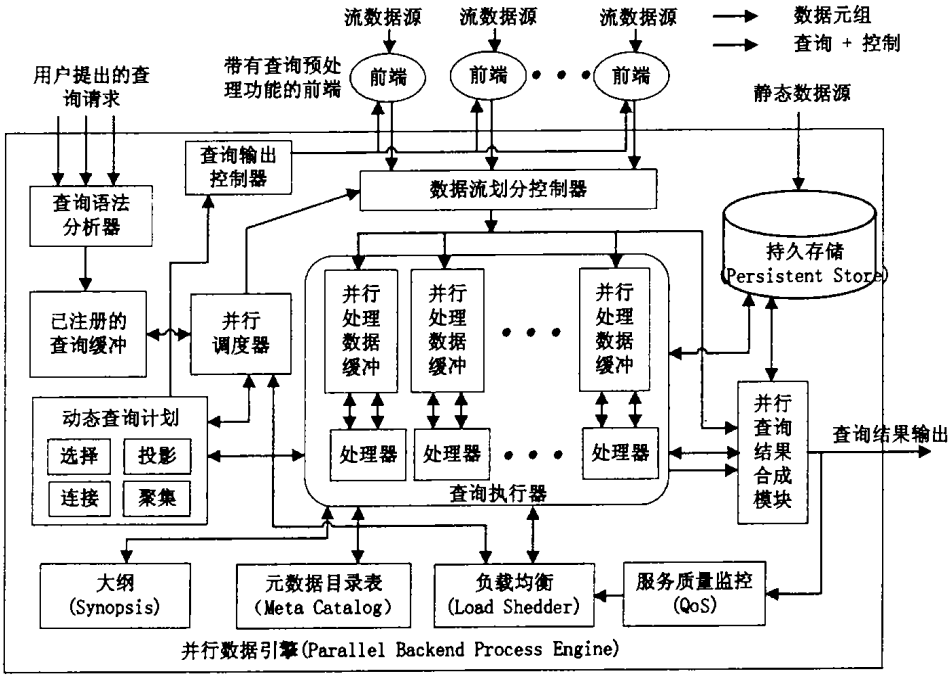

图 6 并行数据流管理系统体系结构

结束语 数据流管理技术正在赢得越来越多的研究者关注，并逐渐成为一个热点问题，然而这是一个极其年轻并在不断发展的领域。目前，国外在这一领域的研究十分活跃，国际

(下转第 41 页)

ence, 1990. 391～407
13 Kamvar S D, Haveliwala T H, Golub G H. Adaptive Methods for the Computation of PageRank. In:Linear Algebra and its Applications, Special Issue on the Numerical Solution of Markov Chains, 2003
14 Han J, Kamber M. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2001
15 Kleinberg J. Authoritative sources in a hyperlinked environment. In:ACM-SIAM Symposium on Discrete Algorithms, 1998
16 Li L Z, Shang Y, Zhang W. Improvement of HITS-based algorithms. on web documents. In:Proc. of the eleventh intl. conf. on World Wide Web, 2002. 527～535
17 Page L, Brin S, Motwani R, Winograd T. The PageRank citation ranking: Bringing order to the web. Stanford Digital Libraries Working Paper, 1998
18 Salton G, McGill M J. Introduction to Modern Information Retrieval. McGraw-Hill, 1983
19 Srivastava J, Cooley R, Deshpande M, et al. Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. In 2000 ACM SIGKDD, Jan. 2000
20 Yuwono B, Lee D L. Search and Ranking Algorithms for Locating Resources on the World Wide Web. In:Proc. of the Twelfth Intl. Conf. on Data Engineering, 1996. 164～171
21 朱炜，王超，李俊，潘金贵. Web超链分析的算法研究. 计算机科学, 2003,30(9)
22 Alexa Internet, Inc. http://www.alexa.com, 1996-2004

(上接第10页)

会议 VLDB 和 ACM SIGMOD 等都将数据流作为一个热点问题来讨论，而我国在这方面的研究才刚刚起步。本文简要分析了目前三个典型原型系统的基本实现技术，并将这个领域目前的研究热点和今后的发展趋势介绍给广大计算机工作者，希望我国有更多的感兴趣的同行加入到这一领域的研究行列中来，以提高我国在这一领域的整体研究水平。本文也给出了一个基于硬件预处理的并行数据流管理原型系统的体系结构。

参 考 文 献

1 Arasu A, Babu S, Widom J. An Abstract Semantics and Concrete Language for Continuous Queries over Streams and Relations: [Technical Report]. Nov. 2002. http://dbpubs. stanford. edu: 8090/pub/2002-57
2 Motwani R, et al. Query Processing, Approximation, and Resource Management in a Data Stream Management System. In: Proc. Conf. on Innovative Data Syst. Res, 2003. 245～256
3 Chandrasekaran S, Franklin M. Streaming Queries over Streaming Data. In: Intl. Conf. on Very Large Databases (VLDB), Hong Kong, 2002
4 Chandrasekaran S, et al. TelegraphCQ: Continuous Dataow Processing for an Uncertain World. In:Proc. Conf. on Innovative Data Syst. Res, 2003. 269～280
5 Carney D, et al. Monitoring streams-A New Class of Data Management Applications. In: Proc. Int. Conf. on Very Large Data Bases, 2002. 215～226
6 Cherniack M, et al. Scalable Distributed Stream Processing. In CIDR, Asilomar, CA. Jan. 2003
7 Carney D, et al. Operator Scheduling in a Data Stream Manager. In: Proc. of the 29th Intl. Conf. on Very Large Data Bases (VLDB'03), Berlin, Germany, Sep. 2003
8 Wang H, Zaniolo C. ATLaS: A Native Extension of SQL for Data Mining and Stream Computations, UCLA CS Dep. 2002
9 Naughton J, DeWitt D, Maier D. The Niagara Internet Query System. University of Wisconsin, 2002
10 Bonnet P, Gehrke J, Seshadri P. Towards Sensor Database Systems. In:Proc. Int. Conf. on Mobile Data Management, 2001. 3～14
11 Cortes C, Fisher K, Pregibon D, Rogers A. Hancock: a language for extracting signatures from data streams. In: ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, 2000. 9～17
12 Madden S, Franklin M J. Fjording the stream: An architecture for queries over streaming sensor data. In:Proc. of 18th Intl. Conf. on Data Engineering, 2002
13 Terry D, Goldberg D, Nichols D, Oki B. Continuous queries over append-only databases. In:Proc. of the 1992 ACM SIGMOD Intl. Conf. on Management of Data, June 1992. 321～330
14 Sullivan M, Heybey A. Tribeca: A System for Managing Large Databases of Network Traffic. In: Proc. of the USENIX Annual Technical Conf., New Orleans, LA, 1998
15 Madden S, Shah M, Hellerstein J, Raman V. Continuously adaptive continuous queries over streams. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, Madison, Wisconsin, May 2002. 49～60
16 Shah M, Hellerstein J, Chandrasekaran S, Franklin M. Flux: An Adaptive Partitioning Operator for Continuous Query Systems: [Technical Report CS-02-1205]. U. C. Berke-ley, 2002
17 Zhu Y, Shasha D. StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. In VLDB, 2002
18 Domingos P, Hulten G. Mining high-speed data streams. In: ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, 2000. 71～80
19 Guha S, Mishra N, Motwani R, O'Callaghan L. Clustering data streams. In: the Annual Symposium on Foundations of Computer Science, IEEE, 2000
20 Gilbert A C, Kotidis Y, Muthukrishnan S, Strauss M. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In VLDB, 2001. 79～88
21 Guha S, Koudas N, Shim K. Data streams and histograms. In: Proc. of Symposium on Theory of Computing, 2001. 471～475
22 Viglas S D, Naughton J F. Rate-based query optimization for streaming information sources. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, Madison, Wisconsin, May 2002. 37～48
23 Babu S, Widom J. Exploiting k-constraints to reduce memory overhead in continuous queries over data streams: [Technical report]. Stanford University Database Group, Nov. 2002. Available at: http://dbpubs. stanford. edu/pub/2002-52
24 Chen Y, Dong G, Han J, Wah BW, Wang J. Multi-Dimensional Regression Analysis of Time-Series Data Streams. In VLDB, 2002
25 Kang J, Naughton J F, Viglas S D. Evaluating Window Joins over Unbounded Streams. In ICDE, Feb. 2003
26 Arasu A, Babcock B, Babu S, McAlister J, Widom J. Characterizing memory requirements for queries over continuous data streams. In:Proc. 21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, May 2002. 221～232
27 Babcock B, Babu S, Datar M, Motwani R, Widom J. Models and issues in data stream systems. In:Proc. of the 2002 ACM Symp. on Principles of Database Systems, June 2002. 1～16
28 Dobra A, Gehrke J, Garofalakis M, Rastogi R. Processing complex aggregate queries over data streams. In: Proc. of the 2002 ACM SIGMOD Intl. Conf. on Manage-ment of Data, 2002. 61～72
29 Aanur R, Hellerstein J. Eddies: Continuously Adaptive Query Processing. In SIGMOD, 2000. 261～272
30 Babcock B, Babu S, Datar M, Motwani R. Chain: Operator Scheduling for Memory Minimization in Stream Systems. In:Proc. of the Intl. SIGMOD Conf. San Diego, CA, 2003
31 Manku G S, Motwani R. Approximate Frequency Counts over Streaming Data. In: Proc. of the 28th Intl. Conf. on Very Large Data Bases (VLDB 2002), Aug. 2002
32 Charikar M, Chen K, Farach-Colton M. Finding frequent items in data streams. In:Proc. of 29th Intl. Colloquium on Automata, Languages and Programming, 2002