一种数据流上截止期敏感的滑动窗口处理策略*)

武珊珊 谷 峪 吕雁飞 于 戈
(东北大学信息科学与工程学院 沈阳 110004)

摘 要 在数据流上定义了截止期作为连续查询的实时约束,并建立了基于截止期的滑动窗口处理模型;提出了一种基于预测的截止期敏感的滑动窗口处理策略;在多滑动窗口查询处理环境中,提出了一种(k,m)截止期约束机制,在提高查询实时性的前提下,尽可能的满足不同查询对结果丢失率的不同约束。实验结果表明该处理策略能够有效提高数据流上滑动窗口查询的截止期满足率。

关键词 数据流,滑动窗口,截止期,查询处理

A Deadline-Sensitive Sliding Window Processing Strategy over Data Streams

WU Shan-Shan GU Yu LU Yan-Fei YU Ge
(School of Information Science and Engineering, Northeastern University, Shenyang 110004)

Abstract Deadline is defined as a real-time constraint of continuous query over data stream, and a deadline-based sliding window processing model is built. Also, a deadline-sensitive processing strategy based on prediction is proposed. Under the circumstance of multi-query, a (k,m) deadline constraint is dedicated to meet different result-loss-ratio constraints on different queries with the premise of improvement of real-time performance. Experimental results show that this strategy effectively improves the deadline satisfaction ratio of sliding window processing over data streams.

Keywords Data stream, Sliding window, Deadline, Query processing

1 引言

随着信息处理在通信、工业生产、经济信息处理等领域的广泛应用,数据已不仅仅拘泥于文件、数据库等传统的静态形式,一种连续、无界、不定速度的数据流已经出现在越来越多的应用领域,如:网络监控、网络流量管理、入侵检测、无线传感器网络、生产线管理、股市信息分析等等。数据流快速、连续、实时的在线到达,要求查询连续不断地执行并将结果以流的形式提交给用户,以满足在线数据处理的应用需求。目前,数据流相关研究已成为数据库领域一个热点课题,并已出现了一些初具规模的原型系统,如:Brandies,MIT 和 Brown 等几所大学联合设计开发的 Aurora 系统[1],Wisconsin 大学的 NiagaraCQ 系统[2],Stanford 大学的 STREAM 系统[3],UC Berkeley 分校的 TelegraphCQ 系统[4]等。

由于数据流连续不断地在线到达,应用往往要求数据流管理系统提供实时的查询服务,即在规定的时间里完成对流中数据的查询处理,过长的响应延迟往往会破坏数据的时间一致性,带来某些不确定性。在一些特殊应用领域,超过某一阈值的延时会导致系统资源浪费,甚至可能引发灾难性的后果。目前,数据流上实时查询处理的研究工作大都是尽最大可能降低平均响应延迟[5~8],而没有对响应延迟上限进行约束,为此我们借鉴实时数据库中事务截止期的概念[9],在数据流上定义了连续查询的截止期,并提出了一种以提高查询结果截止期满足率为目标的窗口实时查询处理策略。本文主要贡献如下:(1)提出了截止期的概念用于描述流式应用对数据流查询处理的实时性约束,并建立了截止期敏感的滑动窗口模型;(2)提出了一种滑动窗口的截止期满足性预测算法,同时给出了支持该算法的带反馈控制的滑动窗口查询处理框架;(3)在扩展到多查询处理环境时,针对不同查询对结果丢失率有不同约束的情况,提出了一种(k,m)截止期约束机制,使得多查询的调度在提高截止期满足率的前提下,尽可能地满足不同查询对结果丢失率的不同约束。

2 截止期敏感的滑动窗口模型

2.1 连续查询的截止期

不失一般性,假设本文采用的时间戳定义在非负整数域。对任意时间点 t 到达数据流的输入元组来说,理想情况下期望系统能够立即给出相应的查询结果,故把输入元组的到达时刻 t 作为相应查询结果的建立时间,记作 t_{create}。但在实际处理过程中,输入要经过一定的处理时间才能获得相应的查询结果,我们把查询结果的实际输出时间称作结果的释放时间,记作 $t_{release}$。故响应延时可用 $t_{release}-t_{create}$ 来表示,于是定义数据流上连续查询的截止期如下。

定义 1(截止期) 连续查询 Q 所能容忍的响应延时上限被称为连续查询 Q 的截止期,记作 D_Q。对连续查询 Q 的任何输出结果来说:

- 如果 $t_{release}-t_{create}\leqslant D_Q$,则称该结果满足查询截止期,记作 DSAT;
- 否则 $t_{release}-t_{create}>D_Q$,则称该结果错失查询截止期,记作 DMISS。

有些应用具有软实时性,数据应该在其截止期内完成处理,但超过截止期的查询结果还有一定的意义(尽管意义不断

*)基金项目:国家自然科学基金项目(编号:60473073,60503036);霍英东青年基金优选课题资助(104027)。武珊珊 博士研究生,研究方向:数据流理论与技术;于 戈 教授,博士生导师,CCF 会员,研究方向:数据库理论与技术,数据挖掘,分布式数据管理,数据流。

下降),故即使到达其截止期也不必立即夭折数据处理。而有些应用具有硬实时性,数据必须在其截止期内完成处理,否则将带来灾难性的后果,故到达截止期还未完成处理的数据必须夭折丢弃。还有一类应用具有固实时性,即数据处理一旦到达截止期,其结果的价值立即降为零,此后固定为零(也不会为负),因而白白浪费了系统资源。本文讨论数据流上有固实时约束的滑动窗口处理。

截止期满足率(Deadline Satisfaction Ratio)是实时应用中一个重要的性能评测指标,定义如公式(1)所示。其中,#DMISS和#DSAT分别表示错失截止期和满足截止期的数据个数。

$$DSR=\frac{\#DSAT}{\#DSAT+\#DMISS}\times 100\% \quad (1)$$

2.2 滑动窗口模型

数据流连续、无限的特性使得连续查询无法在整个数据流上进行聚集、连接等阻塞操作,于是滑动窗口作为一种近似查询技术被广泛应用在数据流的查询处理中[10]。滑动窗口的实质是:用窗口限定数据流上的局部范围,然后通过窗口的滑动,用窗口内动态的局部数据来代替整个数据流。假设 $0<H\leqslant Z$,对于数据流 S 上大小为 Z、跳数为 H 的滑动窗口 W 来说,W 中总是包含最近 Z 时间单位内到达数据流 S 的元组,且窗口每 H 单位时间向前滑动一次。若窗口在时间点 t 滑动,则称窗口实例 W_t 建立。我们把 W_t 上查询结果的建立时间和释放时间分别记作 $t_{create}(W_t)$ 和 $t_{release}(W_t)$,有 $t_{create}(W_t)=t$。于是,对滑动窗口 W 上截止期为 D_Q 的连续查询 Q 来说,任一窗口实例 W_t:

- 如果 $t_{release}(W_t)-(t_{create}(W_t)\leqslant D_Q$,则称 W_t 满足查询截止期,即 W_t 是 DSAT 的;
- 否则 $t_{release}(W_t)-t_{create}(W_t)>D_Q$,则称 W_t 错失查询截止期,即 W_t 是 DMISS 的。

此时,公式(1)中#DSAT和#DMISS分别表示满足截止期和错失截止期的窗口实例个数。

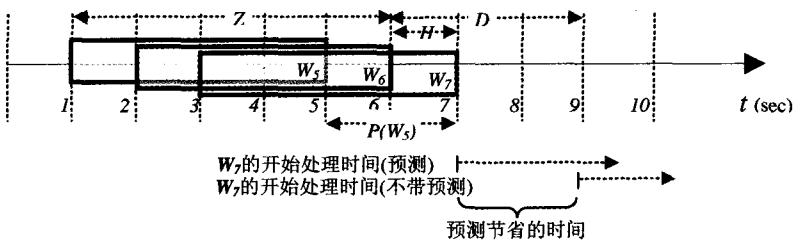


图1 截止期敏感的滑动窗口模型

以图1中的滑动窗口为例,窗口大小为4秒,跳数为1秒,查询截止期为3秒。在第5秒结束时 W_5 建立,假设此时 W_4 已处理完毕,则 W_5 可即刻开始处理直到第7秒处理完毕,有 $t_{release}(W_5)-t_{create}(W_5)=7-5\leqslant 3$,故称 W_5 是 DSAT 的。

本文采用增量计算的方法维护滑动窗口,与前一窗口实例 W_{t-H} 相比,W_t 中加入了最近 H 单位时间内到达数据流的元组(记作 W_t-W_{t-H}),而 W_{t-H} 中最老的 H 单位时间的元组(记作 $W_{t-H}-W_t$)因滑出窗口而失效。显然,W_t 的查询结果可以在 W_{t-H} 的查询结果基础上,通过追加上新元组和失效元组所产生的查询结果(分别用 $output^+()$ 和 $output^-()$ 表示)来获得,如公式(2)所示。值得指出的是,由于窗口滑动引起的新元组加入和旧元组失效对总查询结果集都可能增加或删去元组。

$$Output(W_t)=Output(W_{t-H})+Output^+(W_t-W_{t-H})+Output^-(W_{t-H}-W_t),t=Z+i\cdot H,i=1,2\cdots \quad (2)$$

从窗口的处理过程来看,任意窗口实例 W_t 的计算代价除窗口维护代价 P_m 外,还有对新元组和失效元组的计算代价,分别用 $P^+(W_t-W_{t-H})$ 和 $P^-(W_{t-H}-W_t)$ 表示。于是有窗口实例计算代价公式(3)。

$$P(W_t)=P^+(W_t-W_{t-H})+P^-(W_{t-H}-W_t)+P_m,t=Z+i\cdot H,i=1,2\cdots \quad (3)$$

对于带截止期的滑动窗口查询来说,查询处理的目标就是尽可能地降低每个窗口实例的响应延时,使其在截止期内完成处理。而在实际的处理系统中,窗口实例往往不可能在建立时刻起就获得处理机。通常,在单查询处理系统中,处理机可能还未完成前面窗口实例的处理;而在多查询处理系统中,处理机还可能正在被其他查询处理占用。因此,可以通过去掉窗口实例在系统中的等待时间($t_{current}-t_{create}(W_t)$)来修正绝对截止期,以获得该窗口实例的剩余处理时间。这样,我们给出过期因子的定义如下:

$$\delta(W_t)=\frac{P(W_t)}{D(W_t)}=\frac{P(W_t)}{D-(t_{current})-t_{create}(W_t))},where\ t\geqslant Z \quad (4)$$

- $0<\delta(W_t)\leqslant 1$:表示 W_t 满足截止期,即 W_t 是 DSAT 的;
- $\delta(W_t)>1$:表示 W_t 错失截止期,即 W_t 是 DMISS 的。

3 基于预测的滑动窗口查询处理策略

3.1 截止期满足性预测算法

固实时应用中,错失截止期的查询结果通常会由于对系统输出做出零贡献而被丢弃。如果在未处理过期窗口前能够预测出其截止期不满足性,则可实施降载以避免计算资源的浪费,并把有效的处理时间尽量贡献给能够满足截止期的窗口实例,进而提高截止期满足率。

在 $D>H$ 的前提下,对窗口实例 Wt 处理过程中的任一时间点 τ 来说,必存在非负整数 k,使得 $k*H\leqslant\tau-t<(k+1)*H$,其中 $\tau\in[t,t+D)$。换句话说,在 W_t 的处理到时间点 τ 时,已经有 k 个后继窗口实例建立。这样,我们就有可能对这 k 个窗口进行截止期满足性预测。以系统中仅有一个查询的情况来看,如图1中 W_5 在第7秒结束时处理完毕,此时 W_6 和 W_7 已经建立,因此我们可以对它们进行截止期满足性的预测。如果预测 W_6 不能满足截止期并对其降载,则可从第8秒起处理 W_7(忽略预测计算代价)。否则,只有当 W_6 处理到其截止期(第9秒结束)才发现错失截止期,这样 W_7 的开始

处理时间就被推迟至第 10 秒，而第 8、9 秒的有效处理时间就白白浪费在了无效的 W_6 的处理中了。综上，在 $D>H$ 的可预测条件下，截止期满足性预测能够指导有效的处理时间尽量贡献给能够满足截止期的窗口，从而提高查询的截止期满足率。

分析公式(3)所示的窗口处理代价，窗口维护代价 P_m 是可以通过统计计算获得的常量。而能够影响 $P^+(W_t-W_{t-H})$ 和 $P^-(W_{t-H}-W_t)$ 取值的有三个因素：(1)新加入到窗口中的元组个数和失效元组的个数，分别用 $|W_t-W_{t-H}|$ 和 $|W_{t-H}-W_t|$ 表示；(2)窗口处理函数的计算复杂性；(3)各操作的物理实现算法。实际上，查询计划一旦确定，因素(2)和(3)对 $P^+(W_t-W_{t-H})$ 和 $P^-(W_{t-H}-W_t)$ 取值的影响就已经确定，可以通过新到元组和失效元组的单位处理代价来反映，分别用 P_0^+ 和 P_0^- 来表示。在查询执行的过程中，监视器定期对新到元组和失效元组的处理代价进行统计，以获得 P_0^+ 和 P_0^-，并将其传给预测器。预测器根据输入数据的元数据信息和单位处理代价等信息按公式(5)预测窗口处理代价，进而预测出窗口的截止期满足性。窗口截止期满足性预测算法如表 1 所示。

表 1　窗口截止期满足性预测算法

输入：$|W_t-W_{t-H}|$，$|W_{t-H}-W_t|$，P_0^+，P_0^-
输出：窗口的截止期满足性预测结果。

```
1. if(W_k is built up){
2. P̂(W_k)=(P_0^+ * |W_k-W_{k-H}|+P_0^- * |W_{k-H}-W_k|) * ρ+P_m;
3. D̂(W_k)=D-(t_current-t_build(W_k));
4. δ̂(W_k)=P̂(W_k)/D̂(W_k);
5. if (δ̂(W_k)>1){
6.       predict W_k is DMISS;
7.       shed load;
8.       }
9. else predict W_k is DSAT;
10. }
11. Make prediction adaptation;
```

$$\hat{P}(W_t)=(P_0^+ * |W_t-W_{t-H}|+P_0^- * |W_{t-H}-W_t|) * \rho + P_m, t=Z+i \cdot H, i=1,2\cdots \quad (5)$$

为了提高算法的健壮性和适应性，我们对算法进行基于控制反馈的动态调整。窗口的处理代价是由运行时统计信息和输入的元数据估算的，显然运行时统计信息是误差的主要来源。当对窗口的处理代价估计过大时，窗口易被预测错失截止期，而有过多的窗口被降载，在这种情况下，DSR 会急剧下降并且降载器的降载率(LSR)将明显增大；相反，如果对窗口的处理代价估计过小，那些实际上不能在截止期内完成处理的窗口很可能被预测满足截止期，在这种情况下，有效的处理时间就容易浪费在不能满足截止期的窗口处理中，同样也会导致 DSR 的显著下降并且预测错误率(PFR)将明显增大。综上，DSR 的显著下降说明对窗口处理代价的估计有偏差，可通过适当调整公式(5)中的 ρ 因子的值(初始置为 1)进行修正。

3.2　带反馈的查询处理框架

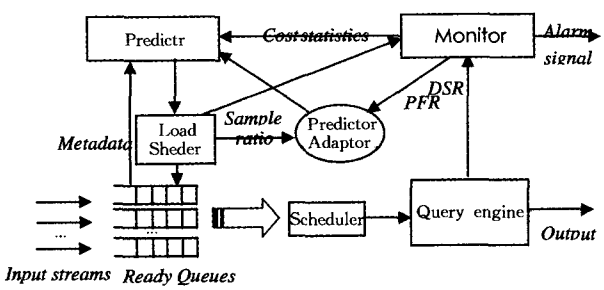


图 2　带反馈的窗口处理框架

如图 2 所示，已建立的窗口实例排队等待处理，调度器将查询引擎分配给不同的窗口查询进行查询处理，查询结果不断输出；监视器监视 DSR 和 PFR 的变化情况，同时获取窗口处理代价的统计值，并将其传给预测器；预测器根据输入元数据和监控器的信息预测窗口的截止期满足性；Load Shedder 对预测错失截止期的窗口实施降载；为了提高处理系统的适应性，预测适配器周期性地根据 DSR、PFR 和 LSR 的变化调整预测器中的 ρ 因子；此外，若 DSR 急剧恶化或者窗口丢弃过多，系统会发出报警信号通知用户系统严重过载以不能适应查询要求。

表 2　优先级定义

优先级
过期因子预测值越大越优先；
过期因子相同*，约束因子越小越优先；
过期因子相同、约束因子为零，窗口分母越大越优先；
过期因子相同、非零约束因子相等，窗口分子越大小越优先；
其他：先来先服务。

* 因为过期因子是一种统计计算的数值，可取小数点后一位的近似值进行比较。

4　支持多查询的(k,m)截止期约束机制

支持复杂应用的数据流系统中通常注册有大量的连续查询，如何在多查询间合理分配处理机关系到各个查询获得的有效处理时间，从而会影响到不同查询的截止期满足率。虽然可以通过丢弃过期窗口实例进行降载，但是对大多数数据流应用来说，通常对结果的丢失率有一定的上限约束，否则即使给出近似结果也很难反映实际应用中的数据状态变化情况。为此，我们定义了 (k,m) 截止期约束机制如下。

定义 2　(k,m) 截止期约束机制：若窗口查询 Q_i 上有 (k_i,m_i) 截止期约束，则 Q_i 的约束因子 C_i 定义如下：$C_i=k_i/m_i$。称 k_i 为窗口分子，m_i 为窗口分母。如果连续 m_i 个窗口实例中不满足截止期的窗口实例个数不超过 k_i 个，则称该查询满足 (k_i,m_i) 截止期约束，反之，称该查询违反 (k_i,m_i) 截止期约束。

为了保证 (k,m) 截止期约束机制，在多查询竞争处理机时，按如表 2 给出的优先级定义进行处理机分配。系统为每个查询动态维护各自的截止期约束因子 $C'_i=k'_i/m'_i$，初始时 $C'_i=C_i$。在查询处理过程中，当查询 Q_i 的某个窗口实例错失截止期，需要提高查询 Q_i 的约束因子 C_i 来反应其紧急程度的提高；相反地，当 Q_i 的某个窗口实例满足截止期，需要降低 C_i。也就是说，流上的约束因子随着时间而动态调整，根据其他流上的窗口实例或者同一个流上前面窗口实例的是否满足截止期而变化。约束因子调整算法见表 3。这种优先级调度策略，最大程度上保证了窗口的丢失率的约束，也尽可能地提高了截止期满足率。

表 3　约束因子调整算法

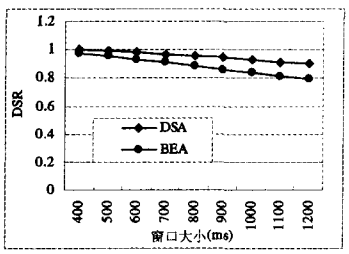
输入：m'_i,k'_i
输入：调整后的 m'_i,k'_i

Q_i 的窗口实例 DSAT 时	Q_i 的窗口实例 DMISS 时

if($m'_i>k'_i$) then $m'_i=m'_i-1$;
else if($m'_i=k'_i$)and ($k'_i>0$) then
$k'_i=k'_i-1;m'_i=m'_i-1;$
if($m'_i=k'_i=0$)then
$m'_i=m_i;k'_i=k_i;$
ifQ_i 违后(k,m)约束 then
$m'_i=m_i;k'_i=k_i;$Alert();
	if($k'_j>0$)then
	$k'_j=k'_j-1;m'_j=m'_j-1;$
	if($m'_j=k'_j=0$) then $k'_j=k_j;$
	$m'_j=m_j;$
	else if($k'_j=0$)and($m_j>0$) the
	$m'_j=m'_j+1;$
	报告 Q_i 违反(k,m)约束

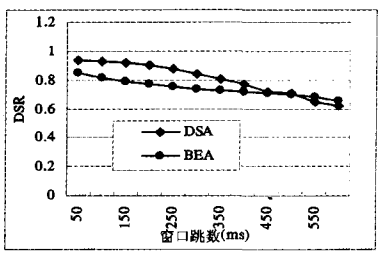
5 实验与性能分析

Internet Traffic Archive 数据集的 LBL-PKT-4 文档保存了某一小时内 Berkeley 实验室与外界广域网间的所有通信记录。'LBL-PKT-4'中包含 863，000 条 TCP 数据包的记录，每条记录中包括时间戳和五个整型属性的值：源 Ip、目的 Ip、源端口、目的端口以及包大小。在此数据集基础上，我们在一定范围内随机生成每个数据包的处理代价。为了保证实验公平性，所有的实验都是在同一物化的随机数据上进行的。实验环境：PentiumIV2. 4GHz，512M DDRAM，Windows 2000 Server Edition。



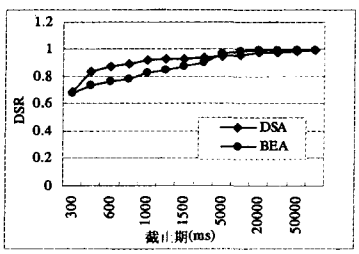
(Hop=300ms Deadline=500ms

图 3 窗口大小对 DSR 的影响



(Size=1500ms Hop=300ms)

图 4 窗口跳数对 DSR 的影响



(Size=1500ms Hop=300ms)

图 5 截止期大小对 DSR 的影响

在单查询处理环境下，我们分别改变窗口的大小、跳数以及查询截止期来考察截止期满足性预测算法(DSA)对截止期满足率的影响。为了评价我们提出的截止期敏感的处理方法，在相同的数据集和实验环境下实现了尽最大努力的处理方法(BEA)。在尽最大努力的处理方法中，如果窗口能够在其截止期内完成处理，则窗口计算结果有效；反之，处理每个窗口直到其截止期才发现不能满足截止期约束，则放弃对该窗口的处理而进行后继窗口处理。实验结果如图 3～5 所示，明显可见，与尽最大努力的算法相比，截止期满足性预测算法对截止期满足率有明显的提高。随着窗口大小和窗口跳数的增大，窗口处理代价随之增大，在相同截止期的前提下，必然导致截止期满足率的下降。而随着查询截止期的增大，查询处理的实时性要求降低，能够满足截止期的窗口个数增多必然使得截止期满足率的提高。

在多查询环境下，实验模拟系统中注册 9 个计算代价相同的滑动窗口查询，可以分成 1～3，4～6，7～9 三组，各组内的查询具有相同的截止期约束，且组间截止期逐个增大；各组内分别采用三个逐次递减的截止期约束因子 C_i，分别为 1/4，1/6 和 1/8。实验对基于(k,m)约束因子的调度算法和 EDF 算法进行比较，结果如图 6 所示。EDF 算法仅仅以查询截止期的静态优先级进行调度，故在截止期相同的各组查询内满足截止期的窗口个数基本相同，而随着截止期的增大，优先级降低导致满足截止期的窗口个数减少。而基于(k，m)约束因子的调度算法在考虑截止期的前提下，还把查询对结果丢失率的约束作为调度优先级确定的考虑因素，故在截止期相同的各查询分组内，随着 C_i 的减小满足截止期的窗口个数逐渐增加，即较好地体现出了查询对结果丢失率的不同约束。

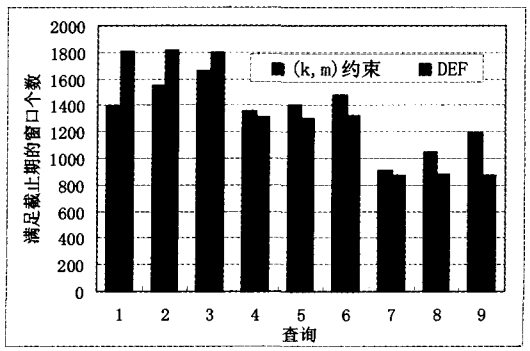


图 6 多查询中(k,m)约束机制对截止期满足性的影响

结束语 本文介绍了一种数据流上截止期敏感的滑动窗口处理策略。通过对截止期敏感的滑动窗口处理建模分析，提出了一种滑动窗口截止期满足性预测算法，并给出了支持该算法的带反馈控制的滑动窗口查询处理框架。在多查询处理环境中，采用一种(k,m)截止期约束机制来满足不同查询对结果丢失率有不同约束的情况。实验结果表明，这是数据流上滑动窗口实时处理中一种有效可行的处理策略。

参 考 文 献

1 Abadi D J, Carney D, Cetintemel U, et al. Aurora: A New Model and Architecture for Data Stream Management [J]. The VLDB Journal, 2003, 12(2): 120～139

2 Chen J, DeWitt D J, Tian F, et al. NiagaraCQ: a scalable continuous query system for internet databases [A]. In: Proceedings of the 2000 ACM SIGMOD international conference on management of data [C]. Dallas, May 2000. 379～390

3 Motwani R, Widom J, Arasu A, et al. Query Processing, Resource Management, and Approximation in a Data Stream Management System [A]. In: CIDR Conference [C], Asilomar CA, January 2003. 245～256

4 Chandrasekaran S, Cooper O, Deshpande A, et al. TelegraphCQ: Continuous Dataflow Processing for an Uncertain World [A]. In: CIDR Conference [C], Asilomar CA, January 2003. 269～280

5 Carney D, Cetintemel U, Rasin A, et al. Operator Scheduling in a Data Stream Systems [A]. In: Proceeding of the 29th VLDB Conf. [C], Berlin, Germany, Setp. 2003

6 Babu S, Shivnath, Motwani R, Munagala K, et al. Adaptive Ordering of Pipelined Stream Filters [A]. In: Proc. of ACM Intl. Conference on Management of Data [C], Paris France, 2004. 407～418

7 Avnur R, Hellerstein J M. Eddies: Continuously Adaptive Query Processing [A]. In: Proceedings of the ACM SIGMOD [C], Dallas TX, May 2000. 261～272

8 Babcock B, Babu S, Datar M, et al. Chain: Operator Scheduling for Memory Minimization in Data Stream Systems [A]. In: Proceedings of the ACM SIGMOD Int. Conf. On Management of Data [C], San Diego, CA, 2003. 253～264

9 Haritsa J, Livny M, Carey M. Earliest Deadline Scheduling for Real-Time Database Systems [A]. In: Proc. of the 12th IEEE Real-Time Systems Symp. [C], Los Alamitos, 1991. 232～243

10 Babcock B, Babu S, Datar M, et al. Models and Issues in Data Stream Systems [A]. In: Proc. ACM Symp. on Principles of Database Systems [C], Madison Wisconsin, 2002. 1～16