单位代码：10293　密　级：

南京邮电大学

硕士学位论文

论文题目：分布式数据处理系统的研究与应用

学　　号	1010041032
姓　　名	朱沙沙
导　　师	洪龙
学科专业	计算机系统结构
研究方向	并行/分布式计算
申请学位类别	工学硕士
论文提交日期	2013 年 03 月

Research and implementation of distributed data processing system

Thesis Submitted to Nanjing University of Posts and Telecommunications for the Degree of

Master of Engineering

By

Shasha Zhu

Supervisor: Prof. Long Hong

March 2013

南京邮电大学学位论文原创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得南京邮电大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

本人学位论文及涉及相关资料若有不实，愿意承担一切相关的法律责任。

研究生签名：_____ 日期：_____

南京邮电大学学位论文使用授权声明

本人授权南京邮电大学可以保留并向国家有关部门或机构送交论文的复印件和电子文档；允许论文被查阅和借阅；可以将学位论文的全部或部分内容编入有关数据库进行检索；可以采用影印、缩印或扫描等复制手段保存、汇编本学位论文。本文电子文档的内容和纸质论文的内容相一致。论文的公布(包括刊登)授权南京邮电大学研究生院办理。

涉密学位论文在解密后适用本授权书。

研究生签名：_____ 导师签名：_____ 日期：_____

摘要

随着网络技术的普遍应用，数据的产生日益加快，如何有效地处理数据已成为业内关注的热点问题。传统的数据处理系统以静态的有限数据集为对象，通过调用静态的查询脚本就可获得所需的查询结果。当今，数据以流行式逐渐向连续、快速、时变方向转变等，传统数据库应用环境的弊端逐渐显露出来，已经无法满足现代化工作的需求。在新的应用背景下，人们对数据库系统的功能和性能有了更高的要求，同时也对数据处理的自动化和智能化提出了更高的要求。

论文首先分析了国内外分布式数据处理系统的研究现状，然后设计了一种典型的分布式数据处理系统方案，并讨论了该系统中的两个关键技术，最终将其应用在邮包传送系统项目中。

论文研究了分布式数据处理中的两个关键技术：查询和降载。与传统的静态查询不同，数据源不间断地进入系统，连续查询始终保持在执行查询并输出查询结果的状态。笔者分析了连续查询的特点和内存空间的分配情况，设计了一种基于贪心策略的动态滑动窗口查询算法，提高了连续查询处理的效率。

降载是针对数据流速不规则引起的负载波动而实施的方法，数据流的到达速率通常是不可预测的且具有很高的突发性，当输入速率超过系统处理能力时，系统会发生过载并且导致系统性能的恶化，本文设计的一种基于蚁群算法的智能降载算法可以寻找出符合条件的最优路径，使得超出负荷的数据可以分流到其它计算节点进行处理，该算法还可以将闲置的节点计算能力充分利用。

关键词：分布式系统，数据流，连续查询，降载

Abstract

With the universal application of network and accelerating rate of data generation, it has become an important topic on how to process data effectively. Traditional data processing system takes static finite dataset as its processing object and obtains the result by calling a static query script. Nowadays, with data becoming continuous, rapid and time-varying, the shortcomings of traditional data processing system gradually appears which makes it unable to meet the needs of modernization work. In the new background of application, higher functionality and performance of database system is required as well as the automation and intellectualization of data processing.

The thesis first analyzes the research status of distributed data processing system at home and abroad. Then, the thesis designs a distributed data processing system scheme and discusses its two key technologies. Finally, the scheme is applied in a packet transmission system.

Two key technologies of distributed data processing are studied in this thesis: query and load shedding. Unlike traditional static query, continuous queries always execute the query and output query results while data sources uninterruptedly access to the system. I study the characteristics of continuous queries and analyze the allocation of memory space, then design a greedy strategy based on dynamic sliding window query algorithm to improve the efficiency of continuous query processing.

Load shedding is a method implemented for load fluctuations caused by irregular data flow rate. The arrival rate of the data stream is usually unpredictable. When the input rate exceeds system capacity; system overload occurs and causes deterioration of the performance of the system. This thesis designed a intelligent algorithm based on ant, which meet the conditions to find out the optimal path and makes full uses of the idle compute nodes.

Key words: distributed system, data stream, continuous query, load shedding

目录

第一章 绪论

1.1 分布式数据处理概述

1.1.1 分布式系统

对于用户来讲，分布式系统与其他应用系统没有区别，都是一个独立的系统，但从底层物理设备和结构体系来看，分布式系统是一个庞大、复杂的体系，由若干个独立的计算机组合而成[1]。从硬件角度来看，分布式系统的机器是独立的，各不同的计算机可以分布在世界的各个角落；从软件角度来看，分布式系统与用户的接口是单一的，用户无法直接感受到操作系统的分散。分布式系统的内部结构、通信方式和功能实现对用户是透明的，这是分布是系统的最大特点。此外，虽然分布式系统由若干个独立的计算机组合而成，但是当解决某一个具体的问题时，各部分能够实现高效、统一的任务指令，随时随地都能与用户进行交互，既提高了系统性能，又提升了用户体验。

可访问性、透明性、开放性和可扩展性是分布式系统的四个主要特性。

(1) 可访问性。实现方便、快捷的资源共享，让用户更方便的访问远程资源是分布式系统的根本目的。共享的资源类型是多种多样的，计算机、扫描机、服务器、存储器、网络等各种设备都可以作为资源被其他用户共享和使用。

(2) 透明性。分布式计算机的透明性是指，最终为用户呈现出来的界面是单一的，用户无法看到系统复杂的底层结构和资源分配情况。透明的类型主要有访问、位置、复制和故障透明四类。系统的透明性提高了系统的兼容性和可移植性，将底层复杂的体系结构隐藏起来，提升了用户体验。

(3) 开放性。计算机系统的开放性是决定系统能否以不同的方式被扩展和重新实现的特征。分布式系统的开放性则主要取决于新的资源共享服务能被增加和供多种客户程序使用的程度。

(4) 可伸缩性。分布式系统可在不同的规模下有效且高效地运转。如果资源数量和用户数量激增，系统仍能保持有效性。

1.1.2 分布式数据处理系统的研究现状

近几年，随着数字化信息技术的发展，分布式数据流处理技术迅猛发展，大量专家学者和研究机构致力于分布式数据流处理技术的研究，学术界和产业界充分认识到分布式数据流处理技术具有广阔的应用前景和发展空间。文献[2]深入研究了如何提取和检测数据流环境中的异常模式，提出了检测和预测异常模式随时间变化趋势的算法，并构建了度量框架和相应算法准确提取出异常模式。文献[3-5]深入研究了采用连续查询的方法处理数据流。文献[6]在共享滑动窗口查询操作的基础上，融入了降载策略和调度优化方法，提出了均匀降载和小窗口准确降载策略，有效提高了 burst 环境下的查询吞吐率，有效解决了系统超载的问题。

北京大学、哈尔滨工业大学走在了分布式数据处理系统的前沿，引领了分布式数据处理技术的发展潮流。北京大学数据库实验室研制并推出了原型阿尔戈斯(Argus)系统，具有很强的兼容性和可移植性，既能作为处理数据流的通用系统，也可以移植到其他操作系统上提供数据库的服务。与此同时，他们基于该系统开发了一套流查询语言，与结构化查询语言十分相似，能够轻松实现查询数据流的功能。国内外大量科研机构和专家学者开始了分布式数据处理系统的研究，创建了与数据流管理相关的体系结构和系统模型，用以满足各大企业对新型数据管理的需求。

斯坦福大学研发了一套通用的数据库管理系统 STREAM(Stanford Stream Data Manager)[7]。该系统将数据流的概念引入其中，以关系数据库系统为基础，不仅保留了传统的数据库查询功能，还增加了查询数据流的功能。斯坦福大学在 STREAM 系统上所做的工作主要包括：建立全新的数据模型[8]；扩充了 SQL 语言，建立了适合数据流查询的连续查询语言 (Continuous Query Language，CQL)；实现了原型系统并对实现方法进行了理论研究和实践，包括调度算子、管理文件和时钟[9]、队列缓存[10]、分流负载等；在查询处理和查询等方面提出了许多独到的见解和方法；也对数据流的信息统计做了许多研究；最后还对建立分布式的数据流管理系统进行了探索。

Aurora 系统[11]是由麻省理工和布朗大学联合研制并开发的，是一个专门处理数据流监控的新型数据处理系统。Aurora 系统体系结构简单、功能强大，能够实现三种应用的处理，即：实时监控应用；大量档案和历史数据管理型应用，不适用于随机序列存储的文档；跨度应用，专门处理当前数据或历史记录。Aurora 系统以一个超大型的触发器网络为核心。每一个数据流向图代表一个触发器，图中的结点对应于 boxes-Aurora 系统的 Bullt-in 操作，该操作共有七种模式。在 Aurora 系统的触发器网络中，存储了大量应用

管理器创建的触发器，这些触发器管理和监控 Aurora 系统的数据流。Aurora 系统优化了触发器网络的运行和编译过程。运行过程中出现的资源超载现象，能够被 Aurora 系统及时检测出来，并根据实际情况和具体需要制定相应的策略减压负荷。Aurora*系统是 Aurora 系统的改进，能够实现分布式 Aurora 系统的升级，具有更高的适用性和可测量性，提高了分布式数据流处理的效率。

美国加州大学伯克利分校正在研制专门用于处理连续数据流的系统-TelegraphCQ[12]系统。该系统能够查询大量的连续流，能够满足大量高速变化的数据流的查询需求。有一个适应性数据流处理系统[13-15]在项目初期建立起来，基于 Java 操作平台。接下来的研究他们决定使用公开源码的 PostgreSQL[16]系统。有许多系统能够连续的查询监控分布于网络上的持久性数据，如：OpenCQ[17]系统，以增量视图维护的方法为基础；NiagaraCQ[18]系统，将分组连续查询技术融入到查询过程中，极大提高了查询效率，缩短了查询时间；计算机网络上的 Web 站点。Tapestry[19]系统也使用了连续查询，以内容为基础过滤了电子邮件和电子公告信息数据库。为了有效执行查询求值操作，该系统的查询语言使用了部分 SQL 语言，有效提高了数据查询效率。Alert 系统[20]采用“事件-条件-动作”触发器机制，以传统 SQL 数据库为基础，实现连续查询只增型的活跃库表。Sfilter 过滤系统采用基于内容的过滤模式，充分考虑了用户的需求，连续查询功能采用 Xpath 语言来实现，能够有效过滤 XML 文档。COUGAR[21]用 ADTs 模拟传感器模型，时间序列是该系统的输入源，是一个典型的传感器数据库。Gigascope[22]在数据源中加入了查询操作，其监控结构基于分布式网络体系。StatStream[23]能够对几个数据流进行统计，是一个在线流监控系统。

1.1.3 分布式计算

随着信息化技术的飞速发展，计算机已经成为人们生活工作不可或缺的重要组成部分。如何合理使用计算机成为人们首要解决的问题。目前，计算机的利用率并不高，没有充分发挥中央处理器的潜力，大部分电脑仍处于闲置状态。尤其是家用计算机，花费在“等待”上的资源远远超过其他事务。即便计算机处于被使用状态，等待输入、等待响应等“等待”仍然耗费了系统的大量时间和空间资源。对于这些复杂的系统，可以将其划分为若干个计算片段，计算用服务端和客户端就是在这种背景下研发出来的。分布式计算是指，将一个庞大的计算任务经过服务器的处理划分为若干个小任务，然后为计算机网络中的计算机分别分配一些小任务，通过并行处理提高处理效率，最后综合并整理计

算数据，得到最后的计算结果。

分布式计算是指信息不只分布在一个软件或计算机上，而是分布于多个软件上，可以用多台或一台计算机同时运行若干个软件，通过网络实现信息的共享。与其他算法相比，分布式算法有明显的优势：第一、共享资源更加方便。第二、能够实现计算负载的平衡，用多台计算机同时处理任务。第三、可以根据实际需要合理选择适当的计算机运行该程序。计算机分布式计算的灵魂是平衡负载和共享资源。

随着现代教育体系的不断完善，课题学科的涵盖范围更广，并进行了更加准确和精细的学科划分。但是，几乎所有学科都需要计算大量数据。天文学为了分析太空脉冲、星球位置等，需要计算机处理大量数据；生物学为了研究蛋白质的体系结构和折叠过程，需要计算机模拟大量数据；药物学为了研制非典、肝炎、禽流感等药物，需要计算机分析大量数据；数学家为了获得更精准的圆周率和计算结果，需要计算机计算大量数据；经济学家为了全面统筹和分析经济发展状况，为宏观调控和战略决策提供更准确、全面的参考依据，需要计算机进行大量的数据统计和分析。因此，计算已经深入到科学的每个角落。分布式计算具有高效、快捷、准确的优势，越来越多的专家学者加入到分布式计算的研究行列之中。

1.2 分布式数据处理系统中的相关问题

1.2.1 数据流

数据流的理论概念是数据序列，呈现的特点是连续、无限制且变化与时间有关。设 a_t 为具体的数据，那么数据流具有的结构形式为{…，a_{t-1},a_t,a_{t+1},…}，其中 t 为时间。与数据库模型相比，数据流模型的数据量不受限制、实时性强且连续，而且数据只能进行一次性处理，没有保存，如果再次进行数据处理，相对困难。数据流模型强调实时性，可以一次性处理大量的数据。

数据库模型对数据的处理可以多次进行，因为数据可以保存在存储介质中，使用时通过 DML 操纵。应用数据库模型处理数据流非常困难，首先要将数据存储，然后才可以使用 DML 查询结果[24]。而实际中的数据流是高速度到达且数量非常大，需要实时处理，如果还要等着存储后再查询，大量的数据流因为来不及存储而丢失。传统的数据库技术不适合处理数据流。

1.2.2 数据流概要构建

构建数据流概要目前具体有如下几种方法：

（1）抽样(sampling)方法

体现概率的数据流组成的数据结构是用抽样方法来实现的。以小部分的数据表示全体数据。这小部分的数据可以采用两种方法获取，一是均匀抽样，数据流中每个数据的概率都是相同的，被抽取的可能性均等，小部分数据表现的是整体的平均性，水库抽样方法属于此类；二是偏倚抽样，每个数据的抽取概率不同，重点的数据几率大，抽取的小部分数据所组成的样本具有某种特性，以计数抽样方法为代表。

（2）直方图(histogram)方法

数据结构采用数据流的轮廓来表示是直方图的特点。数据集在坐标轴上被划分成多个小数据集，类似一个个的桶，不同的桶用不同的数字来表示，外观上形成一个整体轮廓，这样的图形表示直观。很多商业模式多用直观图来表示。直方图还有很多分类，适用的范围也各不相同。

（3）小波(wavelet)方法

数字信号处理技术生成的数据流结构就是应用小波分析方法得到的。小波分析是以模拟量为输入，经过傅立叶变换，形成小波参数，这些小波参数具有模拟量的能量，用它们就可以还原最初的输入信号。模拟量的分析方法在数据库领域得到广泛的应用，像估算选择率等。数据库应用的小波类型较多，哈尔小波是相对简单的，文献[25]就是利用小波技术将数据流划分成若干小数据集形成直方图，哈尔小波算法就是将整个数据流经过转换，产生系列的小波，保留一些高能量，从而将大量的数据流近似为模拟数据集。文献[26]对该技术进行改进，实现了数据的增删改操作，但是这两种技术对误差的范围具有不确定性，数据流的模拟量分析还存在瓶颈。

（4）Hash 方法

Hash 方法的应用是将数据流通过哈希函数，进行演变，缩小范围，这是数据结构形成的常用方法，大范围通过映射转为小范围，产生概要数据结构。如 Bloom Filter 等。

（5）基于滑动窗口模型的方法

一种产生数据流查询近似结果的技术就是通过滑动窗口在最近数据而不是在整个历史数据上进行查询评估。例如，仅用最近一个星期的数据来产生查询结果，这个星期以前的数据将会被丢弃。

在数据流上实施滑动窗口是近似的一个很自然的方法。它是定义良好且容易理解的，更重要的是，它着重于最近的数据，在大多数现实应用中，相对于历史数据，最近数据更加重要：如果一个人想实时地搞清楚网络流量模式，或电话记录或交易记录，或科学传感数据，那么观察最近数据比观察陈旧的历史数据更具有参考价值。 事实上，很多类似的应用中，滑动窗口并不是作为由于计算整个历史数据的不可行性而引入的一种近似技术，而是用户查询的一种查询语义。

1.2.3 连续查询

数据流处理系统的最大特点体现在它的查询方式上，主要表现为连续查询，因为大量的数据流不停息地进入，需要系统不断进行查询，产生新的结果。这种查询方式对系统的要求较高[27]。

传统数据库的查询是一次性的，用户所看到的查询结果基于数据集快照。而连续查询是以时间为变化单位的，不同的时间反映不同的查询结果，将动态的数据流变化按时间进行查询得到结果。它可以实现连续求值，对目前的数据流给出查询结果，此结果也许被存储也许被更新。传统数据库的一次性查询不能应对大量的、连续的数据流，查询结果不能实现随时间而变化，因此连续查询成为数据流处理的主要方式，它可以长期运行，不受查询次数的影响。

1.2.4 降载技术

数据流的特性使之在传输过程中出现波动性，当大量高速的数据流到来时，突发性的情况经常发生，为了保证系统的平衡运行，会对数据流采取一些限制。如果数据流的功率加大超过系统的实际承载能力时，系统处理不了所有到达的数据，影响到下一批数据流的处理，为此必须进行卸载，也就是应用降载技术，去除超出系统的负载。

目前的降载技术研究大体可以分为两类，针对流数据处理的方式，有语义降载和随机降载，还有一些研究资料中提出自适应性降载。它们的处理原则存在差别，但最终的目的是减轻系统负载。

随机降载的原理是随机地丢弃元组，那些丢弃的元组存在一定的比例，从整体上可以保持系统的相对稳定性，减少系统的超负载，但是丢弃的元组损失了一部分语义，影响了语义的完整性。随机降载的不可控是最大的缺陷。

语义降载与随机降载不同，丢弃元组时不是随机性的，它采用过滤技术，判断出哪

些元组不重要，然后再选择丢弃，比较常用的算法有牛奶算法和葡萄酒算法。牛奶算法强调新数据的重要性，出现超负载时旧数据将被先丢弃；葡萄酒算法则不同，它强调旧数据的重要性，超载时新数据将被先处理丢弃。

1.3 论文的研究背景和意义

科技的进步和网络技术的日新月异，增加了数据处理的难度，大量的实时数据流不断地冲击着应用系统的处理极限，要求应用系统能够做出及时的反应，并且还要提高准确性，这些挑战对于传统的数据库技术相当棘手。DBMS[28]的适用范围有限，处理静态的小规模数据结构还可以，面对大型的网络数据流，DBMS 体现出来的弊端较为明显，它没有能力应对动态的数据流，更不可能形成随时间而变化的查询结果，它在大规模的网络应用环境中的作用受到限制，存在技术性障碍。所以开发新的数据模型是目前网络应用系统发展的关键环节。分布式数据模型引起研究者的关注，它是专门的数据流结构模型，可以将产生的数据结构应用于分布式环境。

实际中的数据流大多来自于远程数据源，分布式的特点突出，所以分布式数据流的研究对于分布式应用系统相当重要。本文的研究重点定为数据查询和降载处理。

本文研究的最终目的是将分布式数据处理系统中的核心问题优化后应用在一个具体的系统中，使得该系统的处理数据的效率更高。

要想在海量数据中准确、快速的寻找到答案，需要耗费大量的时间、空间资源，对系统的性能也有更高的要求。虽然外部存储大量数据集的技术已经发展的比较成熟，并在各大领域得到广泛应用，但它不支持连续查询，而且查询效率低，并不适用于数据流应用，无法达到实时性的要求。为了实现高速处理大规模数据，往往要求系统的响应时间短、处理速度快。在进行数据处理时，系统仍然在进行工作，仍有大量数据输入进来。为了达到实时性的要求，必须尽量缩短处理时间，提高响应效率。如果处理速度不够，会有大量数据堆积，造成系统拥塞或停滞。因此，设计一种分布式数据处理系统，提高查询处理速度和系统的负载优化是一件具有现实意义的事情。

1.4 论文的研究工作和安排

全文共分为六个章节。

第一章简要介绍了分布式处理系统和分布式处理系统中的若干关键技术概念，对分布式系统、分布式计算以及国内外研究现状进行了介绍，重点介绍了数据流技术相关的

概念，比如构建数据流概要、连续查询和降载策略，并简要介绍了文章结构。

第二章在对分布式数据处理系统分析的基础上，给出了其总体设计。并对系统中的重点功能模块进行了详细分析。

第三章和第四章主要分析了分布式数据处理系统中的两个关键技术问题：连续查询和降载。第三章针对数据流的特点，在滑动窗口构建概要数据结构的基础上，设计了基于贪心策略的动态滑动窗口查询算法并实现。第四章主要分析了降载算法的核心问题，设计了基于蚁群的降载算法并实现。

第五章针对一个具体系统-邮包传送系统，将蚁群降载算法应用到该系统的终端数据查询功能中，并验证该降载算法的有效性。

第二章 分布式数据处理系统的总体设计

本章讨论了系统的设计，并且对系统主要模块的功能进行了说明。

2.1 分布式数据处理系统

在功能和性能上相比传统功能的数据库系统，分布式数据处理系统(Distributed Data Stream Management System, DDSMS)与传统 DBMS 是相似的，在功能上 DDSMS 允许所有的数据或部分数据通过连续数据流的方式传达。

2.1.1 DDSMS 和 DBMS 的区别

如果我们把数据集看作一个特殊的数据流，那么可以把 DDSMS 定义为一个传统数据库系统的扩展。下面我们先对 DDSMS 和 DBMS 进行下归纳比较。

传统 DBMS 与 DDSMS 在功能和性能方面的几种差异：

(1) 基本的计算模型不相符。传统的数据库管理系统假定 DBMS 被动地存储数据单元，而用户主动发起查询等操作，这是个用户主动，DBMS 被动的模型。而 DDSMS 从外部数据源获取数据，当系统检测到符合查询条件的数据时将数据返回给用户，这是个 DDSMS 主动，用户被动的模型。

(2) DBMS 的查询是精确的查询，目前还没有 DBMS 提供内建的功能支持近似查询。而 DDSMS 由于数据量巨大并且快速变化，在很多时候只能提供近似的查询结果。

(3) DBMS 提供的是一次查询，一次查询获得查询结果，而 DDSMS 是连续查询，只要用户注册了一个查询，并且没有注销这个查询，那么这个查询将一直有效，DDSMS 向用户不断地返回查询结果。

(4) DBMS 通常不考虑与事务相关联的时间和空间的限制，其调度与处理决策不考虑数据的各种时间特性，其系统的设计指标并不强调实时性和查询服务质量的自适应性，而实时性和自适应性正好是数据流应用所必需的。

2.1.2 DDSMS 的需求分析

DDSMS 所处理的是一种随时间变化的数据信息序列，也就是数据流，它的特点是：

连续的、潜在的、无限的、快速的，而且传统的 DDSMS 在实际处理过程中，这种数据序列具有到达顺序不可控、数据的速率不稳定、数据量巨大等特点。这些特点使得设计一个 DDSMS 需要具有以下的功能：

(1) 由于物理存储空间的限制和处理效率的要求，对数据流进行在线处理时，一般只扫描数据一遍；

(2) 在一定的时间内，能够对数据进行排序，使无序变为有序；

(3) 对用户而言，传统的 DDSMS 的程序设计使用户对数据的查询具有很好的实时性；

(4) 传统的 DDSMS 在处理数据中，当遇到数据流的数据量巨大已超过系统的承载能力时，随机或者有选择地清除一些数据以缓解系统数据的膨胀；

(5) 传统的 DDSMS 对异常数据的处理也提出了一些要求，首先要迅速，同时要合乎实时的要求；

(6) 及时的数据用户的接口能够为用户提供方便的数据信息查询。

2.1.3 DDSMS 的系统结构

图 2.1 对 DDSMS 提供了一个可供参考的抽象系统结构。

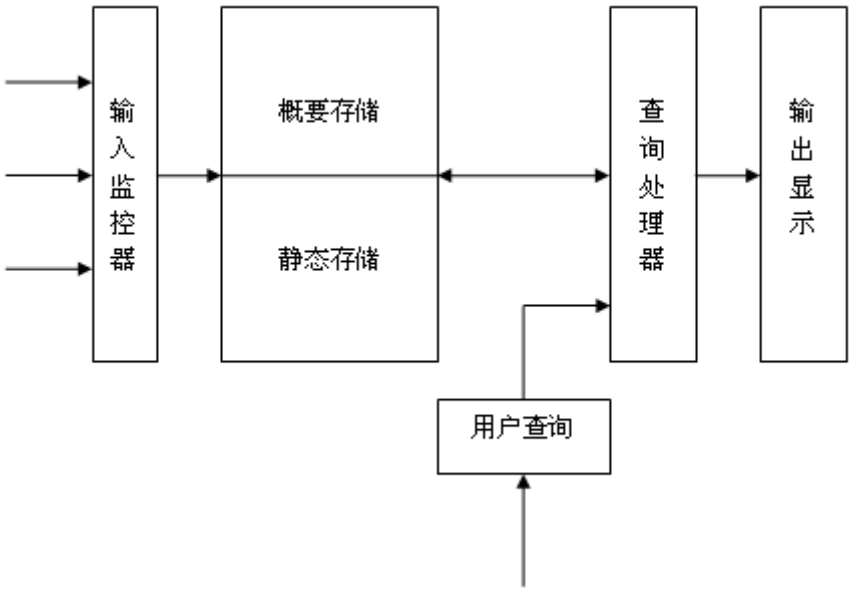

图 2.1 DDSMS 的结构

通过取样的方法控制数据输入的流量可以输入监听器。查询库可以处理共享，它存储系统的连续查询，连续查询已经在系统中注册。窗口查询的临时工作区，关于每个数据源的物理位置等静态存储这三个部分构成数据存储。在当前数据流状态上，既可以一

次查询也可以连续查询。输入的监控器和查询的处理器互相联系，其结果存储在临时缓存中或通过流输给用户，而且通过变化数据输入速率可以对查询计划进行优化。

这个系统可以分为下面两个部分:

(1) 服务器方面：服务器访问接口可以处理客户和服务器相互的所有命令和数据，服务器访问的接口被称作是外界和服务器的纽带。服务器等待连接用户，控制器监听特定的端口访问接口，通过用户给的命令、处理结果或者数据流，最终返回查询的结果。

(2) 终端接口：终端的接口是为用户操作提供的接口，屏蔽了其中的作用过程，控制命令和查询接口构成了终端的接口，DLL 在终端中是终端接口模块。

2.2 主要功能模块说明

接下来对各模块的主要功能进行简单的描述。

2.2.1 数据监控模块

数据监控器的功能主要有两点：

(1) 根据数据的特征，构建概要数据。数据流是一个实时的、连续的、潜在无界的、有序的数据项的序列。由于数据流速率的变化是无法预测的，某一时刻到达的数据量可能会超过系统的计算能力(根据 CPU 周期和主存的大小)，所以需要一个输入监视器在需要的时候销毁一些元组。一般采用抽样、直方图或者小波的方法构建概要数据。

(2) 数据处理节点的负载均衡。降载的问题在传统数据库系统中是可以忽视的：第一，传统的数据库 DBMS 对数据进行的是静态存储方式；第二，对一般数据库的查询没有 QOS 方面的需求；第三，传统的数据库 DBMS 对数据的查询是一次性的。然而，当系统的处理能力无法负荷突发流量时，如果不进行及时处理，那么整个系统的吞吐量和响应时间就会逐步恶化，便会导致系统发生拥塞，要解决流数据流速的不稳定的问题，DDSMS 在数据查询问题上即将要接受巨大的考验。当系统发生拥塞时，根据需要本文采用语义降载，而输入监视器需适当时销毁一些元组，语义降载的含义是通过用户理解流处理语义，根据需要丢弃元组。通常情况下，可以将不重要的元组丢弃，尽量减小系统功能和性能受丢弃元组的影响。在丢弃语义时，要以有用信息为依据。火警检测可以通过温度流的检测来实现，此时可以丢弃低温数据，因为有用信息是高温数据。降载算法与前后文语义是息息相关的，选择操作符的功能与语义降载很相似，几乎不会影响最终结果的准确性。

2.2.2 存储模块

对于 DDSMS 性能方面，DDSMS 是具有现实意义的。在对查询进行处理时，待处理的数据是储存在内存的工作区中。在其操作过程中，系统为了存储流入的数据，必须产生查询窗口，即为查询分配相应的工作区。元数据与一般的关系在磁盘中保留。就像数据字典功能，元数据包含元数据。由于需要近似处理，用来存数据流的大概信息，还需要为其中的数据流设定存储区域。

2.2.3 查询模块

在处理查询接口提交的数据时，查询模块起到关键性的作用。不仅有数据流上的连续查询，同时也涵盖了传统关系上的查询、插入、删除和修改等。

用户为获取一个查询可以通过向系统提交一个注册申请，以方便与 DDSMS 进行相互交流。之后用户会接受到一个系统发来的句柄，以便于用户使用这个句柄在今后的任意时间来查询结果，当然没有用处时用户也可随时注销以往相关查询，以保证信息的安全性。接下来我们用一个具体的事例来简述一下：

SELECT *

FROM S[RANGE 6 minutes]

WHERE S.a=10

DDSMS 处理方式如图 2.2：

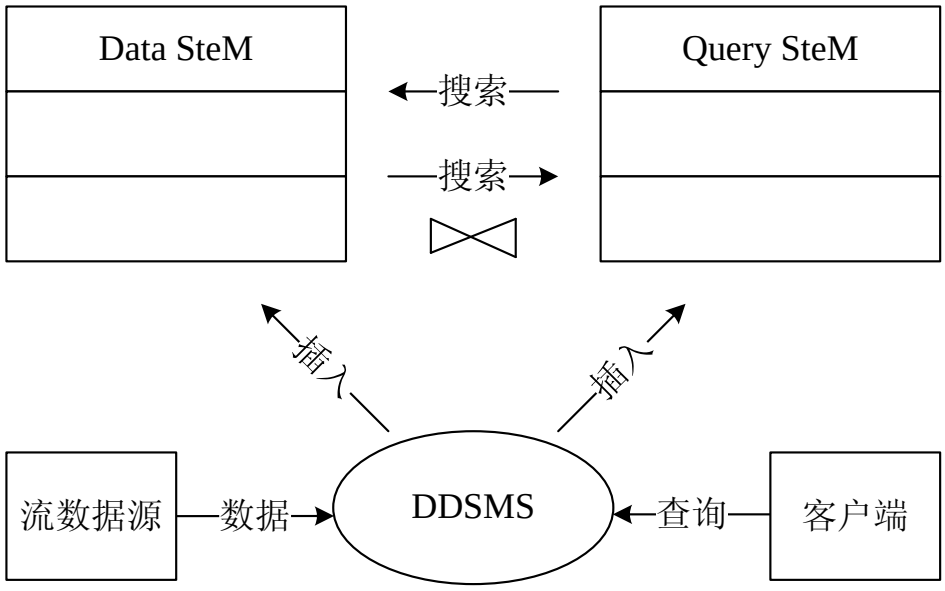

图 2.2 查询模块结构图

在模块内部，DDSMS 把查询流的执行和数据流数据看作两个数据流的连接。如图 2.2 所示。我们把这个处理方式叫做查询-数据连接。

当客户端第一次注册一个查询的时候，查询被插入到 Query Stem 中，然后用它来搜索 Data Stem。这种“新”查询对“旧”数据的方法就是 DDSMS 执行参考历史数据的查询方式。同样的，当一个新数据元素到达的时候，被插入到 Data Stem 中，用它来搜索 Query Stem。这种“新”数据对“旧”查询的应用就是 DDSMS 支持连续查询的方法。两种情况下，搜索的结果物化在一个结果结构(Resultant Stem)中。当调用一个查询的时候，根据输入的窗口，对查询结果进行物化生成结果输出。

2.3 本章小结

本章分析和总结了 DDSMS 和 DBMS 的不同，根据 DDSMS 的分布式的特点设计出了系统的总体结构，然后对系统中的主要功能模块进行了解释说明。

第三章 基于动态滑动窗口的数据流查询算法的设计

查询处理是 DDSMS 中研究较早也是成果较多的内容。针对数据流的查询方式是连续查询，其需求是如何使查询处理结果随着数据的不断到达而不断地产生。因为内存空间是有限的，而数据流资源却是无限的，内存空间的有限性限制了数据流中数据的存储，所以常采用的方法是在内存之中，为数据流系统开辟一个滑动的窗口，即 sliding window，用这个滑动窗口构建一些概要数据，用来保存新近时间内需要处理的数据流信息，方便实时的查询要求。

当今，常采用的一种方法是利用滑动窗口内的数据来实现对数据流信息的连续查询，并且已经取得了一定的研究结果[29-31]。这些研究都是基于固定的滑动窗口大小这一假设，而且查询过程中所涉及的信息(简称为查询范围)只是限制在滑动窗口内部的数据信息。但是，这种方法在实际中是不成立的，因为这一假设在现实应用中很难达到。基于此，本章根据连续查询的特点和数据流流速的不稳定性，提出并设计出了动态调整滑动窗口规模的算法。

3.1 基于固定滑动窗口的数据流查询

3.1.1 滑动窗口及其分类

所谓滑动窗口，指的就是在数据流上面设定的一个独立的区间，这个区间的数据就只有新近得到的数据信息。新数据不断到来，窗口相应的向前移，将旧有数据替换为新近数据，因而可以把滑动窗口看作是部分数据流信息的快照。滑动窗口的定义很好理解，它的重要性在于它看重新近的数据信息，实际之中，新近数据显然比历史数据更有意义：若想实时掌握电话记录、网络流量的模式或者传感数据，甚至交易记录，在这些应用之中，滑动窗口是查询语义，是用户查询中的一种，并非作为因为计算历史信息不可行性引入的一种相似技术。

滑动窗口根据其特点可将其数据流模型分为多种，常见的三种为：快照、标记滑动模型[32]。

（1）快照模型

把初始时间设为 S，结束的时间设定为 E，(S 和 E 都为固定的时间点)，快照模式即

是处理从 S 到 E 的数据信息。比如如果要查询从上午八点到上午十二点的数据，就可以把数据窗口设为[8，12]。采用快照模式有利于查询历史数据。

(2) 标记模型

把初始时间设为 S(固定的时间)，当前的时间设定为 T，标记模式所处理的数据就是 S 到 T 的数据信息。比如查询从上午八点到目前的信息(设定当前是上午十二点)，就可以把数据窗口设为[8，now]。采用标记模式可以实时的处理新数据，并且随着时间推移，处理的信息越来越多。

(3) 滑动窗口模型

基于滑动窗口的数据流模型如图 3.1 所示。设定当前时间为 T,滑动窗口模型处理的信息就是从 T 之前的某段时间到 T 时刻的数据信息。比如若查询此前时刻之前三分钟到达的信息，就可以将数据窗口定为[T-180，T]，单位默认是秒。滑动窗口模型不仅仅可以处理新数据，而且可以处理历史数据，时间推移之后，窗口不断向前，窗口中数据的大小一般是不变的。

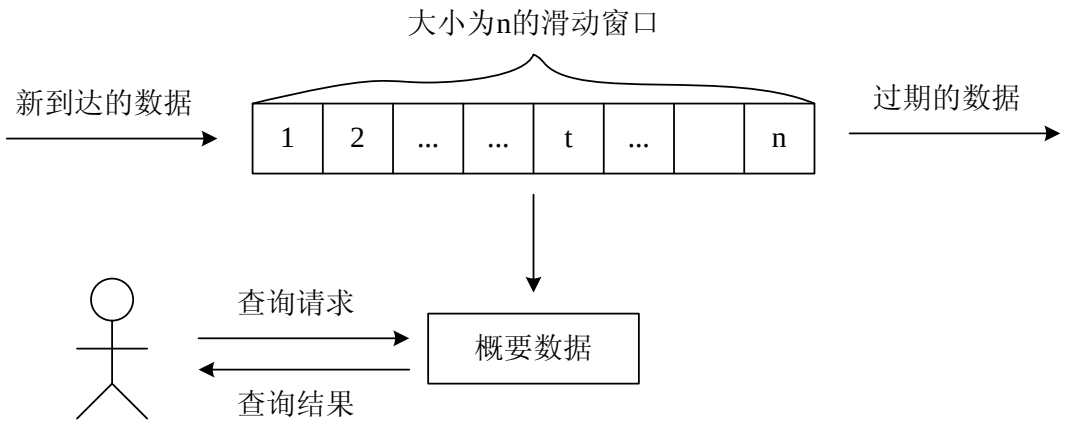

图 3.1 基于滑动窗口的数据流模型

3.1.2 基于固定滑动窗口的数据流连接查询

数据流具有无限、连续的特点，而分布式数据流除了具有此类特点外，还具有数据分散且处理数据多路的特点，因此，DDSMS 多采用多连接查询这种查询方式。

多连接查询有两种主要的实现机制，连接树和多连接算子。考虑到数据流具有无限性的特点，因此，本文中所采用的是滑动窗口这种技术。下图中显示了对三条来自不同方向的数据流进行查询的实现机制，其中，图 3.2(a)显示的是多连接算子的方式，而图 3.2(b)显示的是连接树的方式。

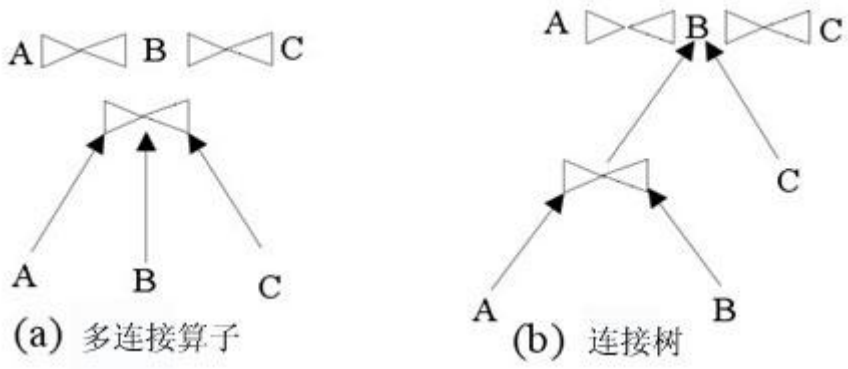

图 3.2　数据流连接查询

在多连接算子中，每个数据流对应一个独特的滑动窗口来保存收到的没有到期的元组，这种模式不保存连接过程中的中间信息。在多连接算子中的每一个数据流，均对应一个探测的序列，假定 A 数据流具有 B、C 的探测顺序，那么当从 A 数据流接收一个新元组 a 时，这个元组先探测滑动窗口 W_B，生成 a_B 结果集，这一结果集在不保存的情况下直接探测滑动窗口 W_C，最后再把连接后的结果作为最终的结果并输出。

与多连接算子方法不同，连接树方式缓存并保存了的中间结果(连接过程中生成)。如图 3.2(b)所示，A 数据流和 B 数据流连接查询，产生的中间结果保存在滑动窗口 W_{AB} 内，当 C 数据流收到新的元组之后，无需再分别探测对滑动窗口 W_A 和 W_B，只要新元组探测滑动窗口 W_{AB}，探测之后的匹配结果就可以作为查询结果使用，因此采用这种方式减小了探测的代价。虽然仍需更新滑动窗口 W_{AB}，但是对于整个查询过程来说仍然节省了不少时间。

3.2 滑动窗口动态调整算法设计

数据流是大规模、快速、实时、变化的，实际应用中的滑动窗口模型经常是规模固定的，但是现实之中，其数据常常是跟随时间变化的，因此，对于滑动窗口的设计实在成为了一个困扰的问题。本节设计一种基于贪心策略的动态滑动窗口算法，使得在查询处理多路数据流的时候效率更高更稳定。

3.2.1 贪心算法概述

为了使得滑动窗口更加符合现实中的动态要求，特采用贪心策略来优化此问题。

贪心法是一种求解最优化问题的算法设计策略[33]。贪心法是通过分布决策的方法来求解问题的。贪心法在求解问题的每一步上都做出某种决策，产生 N-元组解的一个分

量。贪心法要求根据题意，选定一种最优量度标准，作为选择当前分量值的依据。这种在贪心法每一步上用做决策依据的选择准则被称为最优度量标准或者叫做贪心准则，也称贪心选择性质。这种量度标准通常只考虑局部最优性。

在初始状态下，解向量 solution=ϕ，其中未包含任何分量。使用最优度量标准，一次选择一个分量，逐步形成解向量$(x_0,x_1,\ldots\ldots,x_{n-1})$。算法执行过程中生成的向量$(x_0,x_1,\ldots\ldots,x_k)$，k<n，称为部分解向量，也成为部分向量。

在根据最优度量标准选择分量的过程中，还需要使用一个可行解判定函数。设$(x_0,x_1,\ldots\ldots,x_{k-1})$是贪心法已经生成的部分解，根据最优度量标准，算法当前选取解向量的第 k(k<n)个分量为 x_k，此时需使用可行解判定函数来判断，在添加新的分量 x_k 后所形成部分解$(x_0,x_1,\ldots\ldots,x_k)$是否违反可行解约束条件。若不违反，则$(x_0,x_1,\ldots\ldots,x_k)$构成新的部分解，并继续选择下一个分量；否则，需另选一个值做为分量 x_k。

贪心法之所以被称为是贪心的，是因为它希望每一步决策都是正确的，即要求在算法的每一步上，仅根据最优度量标准选择分量，并只需要保证形成的部分解不违反约束条件，最终得到的 N-元组不仅是可行解，而且必定是最优解。但是事实上，最优度量标准一般并不从整体考虑，它只是在某种意义上的局部最优选择，也就是说，每一步决策只是在当前看来是最优的，因此，贪心法不能保证对所有问题都得到整体最优解。但是对于许多小问题，如图的最小代价生成树问题、单源最短路径问题等，利用经过精心设计的选择准则，确实能够生成整个过程的最优解。除此之外，其他状况下，就算采用贪心算法不能够得到整个过程的最优解，也可以得到其近似值。

贪心算法的基本框架描述如下：

```
SolutionType Greedy(SType a[],int n)
{
   Solution Type solution=φ;
   for(int i=0;i<n;i++){
      SType x=Select(a);
      If(Feasiable(solution,x))
         Solution=Union(solution,x);
}
   Return solution;
}
```

SolutionType 是问题解 solution 的类型，Stype 是分量 x 的类型。函数 Select 的功能是按照某种最优量度标准，从 a[]中选择一个值作为下一个考察的分量值。Feasible 是布尔函数，它判定由 Select 选择的 x 是否可以添加到部分解向量中。Union 将 x 与已形

成的部分解向量合并，产生新的部分解向量。对于给定的应用问题，如果能够具体实现 Select、Feasible 和 Union，便得到了该问题的贪心算法。

3.2.2 可变尺寸的滑动窗口

DDSMS 系统中，对于特殊情况，没有界限的输入流显然需要没有界限的内存，这很明显是不切合实际情况的。因此问题就在于在数据的处理过程中采用较为灵活实用的窗口策略，用来限制单个输入流在运行过程中储存的元组数目。滑动窗口即是在单个数据流上面独特设定的区间，这个区间仅包含最新到来的那些数据。新数据不断到来，相应的窗口就往前移，旧数据就被新数据所替换。所以，我们可以将滑动窗口看成是数据流某一部分的过去型快照，与此同时，滑动窗口还可以满足数据流的两方面要求——实时性和无界性。

正是基于滑动窗口是数据流某一部分的过去型快照这一点，本文采用以时间为标准的滑动窗口。此种类型的滑动窗口要求窗口的大小保持恒定大小，并且根据它的更新方式的不同又分为两种：连续更新和周期更新。连续更新滑动窗口的定义如下：s[t-T，t]是某一个数据流上面的滑动窗口，其中，t 是这个数据流最新数据到达的时间。可是在许多 DDSMS 查询应用中，许多窗口设计都是基于某一恒定时间的，仅仅在时空的复杂度等方面取得了一定成果。这些技术成果往往都忽略了数据流在实际中的随时间变化的特性，当实际的数据分布情况随时间不断变化时，模型很难进行自适应的调整[34-35]。吉娜拉等[36-37]提出了一种变尺度的时间窗口——对数时间窗口。它的主要优点为采用了不相同力度的时间窗口，并且这一时间窗口的大小是由用户来指定，比如以年为单位标尺的粗粒度时间窗口以及以刻为单位标尺的细粒度时间窗口。横向对比，自然等长的时间窗口往往需要 35136 个窗口单位,而此种方法仅需 17 个单位。此种对数时间窗口是预先设定好的对数窗口，不能实时的根据数据流来改变窗口的大小，因此并非实际意义上的可变尺寸的动态滑动窗口。

可以分析到，固定滑动窗口大小的局限性如下：

(1) 由于内存资源有限,滑动窗口的规模也是有限，相应的滑动窗口也是有限的规模，只能储存无限数据流中的某一个子集。实际中，由于在数据流上进行连续查询时，其查询的范围也是不可以预先知道的，因此很难确保它仅仅涉及滑动窗口内的数据集。

(2) 在同一数据流上，不同的连续查询具有不同的查询范围，因此数据流的流速也不是恒定不变的，它跟随时间进行变化。因此，相应的滑动窗口规模也要实时进行调整，

以便适应数据流流速与连续查询的动态性变化。而如若滑动窗口大小规模保持恒定不变，就会出现一些问题：一定时间内，一部分滑动窗口有大量的空闲储存空间，而一部分滑动窗口由于其窗口太小而难以处理连续的查询。

可变尺寸的动态滑动可以自适应的根据数据分布变化情况和数据流流速变化情况来调整窗口的大小，以减少处理时间和内存空间消耗。

利用一个例子来解释对滑动窗口进行动态调整的重要性和必要性。设定数据流内部有两个滑动窗口，分别为 w_1、 w_2，而数据流的流速平均值均为 1byte/s，可供滑动窗口使用的内存总共 50bytes。分配策略为：两个滑动窗口都可以存储新近 25s 内的数据.。设 q_1、q_2 分别为 w_1、w_2 上的连续查询，q_1 查询的是新近 20s 内的数据，而 q_2 查询的是新近 15s 内的数据。很明显，两个滑动窗口都可以进行 q_1、q_2 的查询。目前有一个连续查询，假设为 q_3，在 w_2 上开始执行，它查询的范围为新近 30s 内的数据。如若 w_1、w_2 的大小是固定不变的，那么给 w_2 分配的内存太小，使得 q_3 的查询不能进行。但是 w_1 具有一定的空闲储存空间，可以储存 5s 的数据流信息。在滑动窗口所占内存资源恒定的情况下，如若将 w_1 上面空闲的内存空间让给 w_2 使用，就是说让 w_1 储存新近 20s 内的数据，而 w_2 储存新近 30s 内的数据，那么这两个滑动窗口能够支持全部的连续查询。上面这一事例可以看出，当连续查询启动、结束或者数据流的流速变化的时候，数据流系统内采取恒定的滑动窗口规模这一策略是不切实际的，它不能支持全部的连续查询。由此我们要推翻对数据流进行处理的技术中要求滑动窗口规模保持恒定这一条件，要找寻一种能够灵活改变滑动窗口规模的方法机制，更加有效的支持在数据流系统中的全部有效连续查询。

3.2.3 基于贪心策略的滑动窗口规模调整算法的设计

表 3.1 和表 3.2 中包含了本节用到的一些数学符号和概念。

表 3.1 数学符号

W	滑动窗口集	w	任一滑动窗口
DSize(w)	滑动窗口 w 中数据的大小	q	数据流上的连续查询
Rate(w)	流入 w 的数据流的流速	$Q_t(q)$	查询的时间宽度
W(w)	滑动窗口 w 的宽度	$Q_e(q)$	查询 q 的允许时间误差范围
U(w，q)=1	查询 q 是滑动窗口 w 的查询	U(w，q)=0	q 不是滑动窗口 w 的查询
M	所有滑动窗口所需内存总和	Min_T(w)	窗口最小有效宽度

表 3.2 相关数学概念

最小有效宽度	$\max\{Q_{t-e}(q)\|q\in Q\wedge U(w,q)=1\wedge w\in W\}$,其中 $Q_{t-e}(q)=Q_t(q)-Q_e(q)$}为滑动窗口 w 的最小有效时间宽度，记为 Min_T(w)。
累计误差	$\forall w\in W$ ， 滑动窗口 w 的累计误差 $E(w)=\sum_{q\in Q}P(w,q)*(Q_t(q)-W(w))$,其中当 U(w，q)=1 且 $W(w)<Q_t(q)$时，P(w,q)=1,否则 P(w,q)=0。 采用累计误差累衡量连续查询输出查询结果时产生的误差大小。
宽度增量	设 S_i 和 S_j 分别是第 i 和第 j 步滑动窗口规模调整后的瞬时描述，i<j.集合{$x_W\|w\in W$ 且 $x_W=W^{(S_j)}(w)-W^{(S_i)}(w)$}为宽度增量，记为Δ x。
最优滑动窗口和最优调整活动	设 S 是一个瞬时描述。满足 $G(u,W^{(S)}(u))=\max\{G(w,W^{(S)}(w))\|w\in W\}$的滑动窗口 u 称为 S 的最优滑动窗口。$L_u$ 中一定存在两个相邻收益点 au_i 和 au_{i+1} 使得 $au_i\le W^{(S)}(u)<au_{i+1}$ 成立，相应的调整活动 $I=(u,W^{(S)}(u),\Delta X_u)$称为 S 的最优调整活动,其中$\Delta X_u=au_{i+1}-W^{(S)}(u)$,称为 S 的最优增量.
最优调整活动的覆盖	设瞬时描述 S 的最优滑动窗口为 u,S 的最优调整活动 $I=(u,W^{(S)}(u),b)$.如果以 S 为基的宽度增量 X 满足下列条件之一: (1) $W^{(\Delta x)}(u)\ge b$ (2) $\forall v\in W$,$v\ne u$, $W^{(\Delta x)}(v)=0$,且 $W^{(\Delta x)}(u)\ge 0$ 则称 X 覆盖 I.

滑动窗口的规模调整目标是：使所有滑动窗口的累计误差总和降到最低。调整分为两步。

首先设置每个滑动窗口的宽度(w)成有效最小宽度，即 $W_Y(w)=Min\_T(w)$。

其次，将余下的内存空间逐步分布给每个滑动窗口，逐渐增加每个滑动窗口的宽度 $W_Y(w)$，以实现$\sum_{w\in W}E(w)$的最小化。

对于第二步的思想如下：

这一步骤又分为多步，每步都利用贪心算法给一个滑动的窗口增加可利用的内存空间，直到可利用的内存空间耗尽。经过调整，每一步都可以得到集合{$S_w|w\in W$ 并且 S_w 为 w 的宽度}，把这一集合叫做规模调整的一个瞬时描述，并把之记为 S。记 $W^{(s)}(w)$ 为 S 中 w (滑动窗口)的宽度。将三元组(w， $W^{(s)}(w)$，Δ x)表示为在 S 之后进行调整时，w 的宽度增加了Δ x(调整增量)。这一三元组成为 S 的一个调整活动，用 I 表示。

假设有瞬时描述 S,S 的调整活动 $I=(w,W^{(s)}(w),\Delta x)$，以及宽度增量Δ x,现定义如下：

(1) S±I 表示将滑动窗口 w 的宽度从原先的 $W^{(s)}(w)$增加(减少)到 $W^{(s)}(w)\pm\Delta x$ 后所形

成的瞬时描述。

(2) S+△x 表示的是{ W(s)(w)+ W(△x)(w)|w∈W},就是将滑动窗口(所有)的宽度从 W(s)(w)增加到 W(s)(w)+ W(△x)(w)后所形成的瞬时描述。

(3) $D(S_0, \Delta x)$表示的是从 S_0 到 $S_0+\Delta x$ 这一变化过程中滑动窗口减少的误差总和。

(4) $\Delta x + I$ 表示的是 w(滑动窗口)的宽度增量从 $W^{(\Delta x)}(w)$变化为 $W^{(\Delta x)}(w)+\Delta x$。

由分析可以看出，滑动窗口的规模调整可以看成是如下所示的优化：输出应为宽度增量Δx，$\Delta x=\{\Delta x_w | w\in W\}$，其中$\Delta x$ 应满足如下条件:

(1) $D(S_0,\Delta x)=\max\{D(S_0, \Delta A)\}$,其中$\Delta A$ 为任意的一个宽度增量;

(2) $\Sigma_{w\in W}(W^{(S_0)}(w)+ W^{(\Delta x)}(w))\leq M$.

式中 $S_0=\{Min\_T(w)|w\in W\}$，它表示瞬时描述。

设 S 是任何一次调整滑动窗口规模时的瞬时描述值，贪心算法主要进行下列调整：先选定 S 的滑动窗口中的最优 u，然后通过执行 S 的调整活动，增加 u 的宽度和 S 的最优增量。利用以下三个引理来证明问题(2)具有贪心选择性和优化子结构。

引理 1[38].把滑动窗口 u 设为初始的瞬时描述 S_0 的最优滑动窗口,并且把 S_0 的最优调整活动设为 $I_0=(u, W^{(S_0)}(u),b)$，同时宽度增量 A 是问题(2)以 S_0 为基的某一个优化解，那么 I_0 被 A 覆盖。

证明：如若 $W^{(A)}(u)$大于等于 b，那么根据最优调整活动覆盖定义中的条件(1)，A 覆盖了 I_0。如果 $W^{(A)}(u)<b$，若$\forall v\in W, v\neq u, W^{(A)}(v)=0$，则由最优调整活动的覆盖定义条件(2)，A 覆盖了 I_0。如果 $W^{(A)}(u)<b$，且"$\forall v\in W, v\neq u, W^{(A)}(v)=0$"不成立，则必$\exists v\in W, v\neq u, W^{(A)}(v)>0$,构造调整活动 $I_v=(v, W^{(S_0+A)}(v)-\Delta X_v, \Delta X_v)$和 $I_u=(u, W^{(S_0+A)}(u)-\Delta X_u, \Delta X_u)$，$\Delta X_u$ 和 ΔX_v 满足：$\Delta X_v < W^{(A)}(v)$ 并且 $\Delta X_u * DSize(u) * Rate(u)$ 等于 $\Delta X_v * DSize(v) * Rate(v)$等于$\Delta M(\Delta M>0)$。

令 $A-I_v+I_u=B$，那么 B 也是一个宽度增量(以 S_0 为基)。

$D(S_0, B)=D(S_0, A)-\Delta M\times G(v, W^{(S_0+A)}(v)-\Delta X_v)+\Delta M\times G(u, W^{(S_0+A)}(u))= D(S_0, A)+\Delta M*G(u, W^{(S_0+A)}(u))-G(\Delta M*G(v, W^{(S_0+A)}(v)-\Delta X_v))$。

显然，$W^{(S_0+A)}(v)-\Delta X_v=W^{(S_0)}(v)+ W^{(A)}(v)-\Delta X_v > W^{(S_0)}(v)$。

由于滑动窗口 u 为 S_0 的最优滑动窗口且$W^{(A)}(u)<b$，因此 $G(u, W^{(S_0+A)}(u))=G(u, W^{(S_0)}(u))$。由调整收益的定义可知，$G(v, W^{(S_0+A)}(v)-\Delta X_v) <= G(v, W^{(S_0)}(v)) <= G(u, W^{(S_0)}(u))= G(u, W^{(S_0+A)}(u))$成立，从而可得 $D(S_0, B)>=D(S_0, A)$。于是 B 也是一个问题(2)的一个优化解(以 S_0 为基)，并且 $W^{(B)}(u)>W^{(A)}(u)$。假如 B 仍然没有覆盖 I_0，那么就把 B 作为新

出发点，连续重复以上过程，直到构造出一个可以覆盖 I_0 的优化解。

引理 2[38].把滑动窗口 u 设为初始的瞬时描述 S_0 的最优滑动窗口，并且把 S_0 的最优调整活动设为 $I_0=(u, W^{(S_0)}(u),b)$，同时宽度增量 A 是问题(2)以 S_0 为基的某一个优化解，I_0 被 A 覆盖。那么 $A'=A-I_0$(宽度增量)是问题(2)以 $S_1=(S_0+I_0)$为基的优化解。

证明：显然，A′是问题(2)以 S_1 为基的解。下面需要证明 A′是优化解。如果 A′不是最优解，那么假设存在一个优化解，设为 B′，$D(S_1, B') > D(S_1, A')$。将宽度增量设为 $B= B'+I_0$,由于 S_0 的最优调整活动为 I_0，那么 B 是一个问题(2)以 S_0 为基的优化解。因为 $A=A' + I_0$，$D(S_0,B)=D(S_0,I_0)+D(S_1, B') > D(S_0,I_0)+D(S_1,A')=D(S_0,A)$，跟 A 是问题(2)以 S_0 为基的优化解矛盾，所以 A′是以 $S_1=S_0+I_0$ 为基的问题(2)的优化解。

以上两个引理证明了问题(2)有优化子结构，以下定理 1 证明了贪心选择性。

定理 1[38]**.** $\forall i\geq 0$,设 I_i 为 S_i(瞬时描述)的最优调整活动,并且 $S_{i+1}=S_i+I_i$。假如 A (宽度增量)是以 S_0 为基的优化解,并且覆盖 I_0,那么存在 k(正整数),满足 $A=\sum_{j=0}^{k} I_j$ 。

证明：利用归纳法证明。假如 k＝0，可以从引理 1 得知，命题是成立的。假设 k<n 的时候命题是成立的，当 k=n 时，由引理 2 知，$A=A' + I_0$，A′是以 $S_1=S_0+I_0$ 为基的优化解。由归纳假设，$A' =\sum_{j=0}^{n-1} I_j$，于是 $A= \sum_{j=0}^{n} I_j$ 。

根据上述引理和定理,我们可以给出如下的贪心算法思想：

(1) $\forall w \in W$,令 $W_Y(w)=Min\_T(w)$;

(2) 计算得到剩余可用内存空间 $M_R=M-\sum_{w\in W} Min\_T(w)*DSize(w)*Rate(w)$;

(3) $\forall w \in W$,产生滑动窗口 w 在区间 [Min_T(w),Max_T(w)]上的收益点序列 L_w;

(4) 将初始瞬时描述 S_0 设为$\{W_y(w)|w\in W\}$,初始宽度增量Δ X 设为 $\{X_w=0|w\in W\}$,i=0;

循环执行(5)-(8)：

(5) 寻找 S_i(瞬时描述)的 u(最优滑动窗口),最优调整活动 $I_i=(u,W^{(Si)}(u),b_i)$;

(6) 判断是否 $M_R\geq b_i*DSize(u)*Rate(u)$，若满足，则$\Delta x=\Delta x+I_i$ ；若不满足，则执行(8)。

(7) $W^{(\Delta x)}(u)=W^{(\Delta x)}(u)+M_R/(DSize(u)*Rate(u))$。

(8) $M_R=M_R-b_i*DSize(u)*Rate(u)$,$S_{i+1}=S_i+I_i$,i=i+1。判断是否 $M_R>0$，若满足，则执行(5)；

(9) $W_Y(w)=W_Y(w)+W^{(\Delta X)}(u)$; /* 最终输出的是每个滑动窗口的最优宽度。

以上算法中，第一步实现的是第一阶段的调整，从第四步到第九步是贪心求解过程

(问题 (2))。第十步最终输出结果-最优宽度(每一个滑动窗口)。第三步的时间复杂度为O(CqlogCq)。第五步到第九步最多能执行 Cq 次。第六步的时间复杂度为第(6)步的时间复杂性为O(d)。可以看出，时间复杂度方面，总共是O(max{d*Cq, CqlogCq}),其中，Cq表示的是连续查询的总次数，d 表示的是滑动窗口的个数。

3.3 本章小结

本章在分析了基于固定滑动窗口的查询操作的基础上，设计和实现了基于贪心策略的可以动态改变滑动窗口大小的查询算法，并证明了该算法的正确。

第四章 基于蚁群的降载算法设计

数据流系统的处理能力是有限的，当数据流的量和速度超出系统的阈值范围时，会造成系统超负载工作，系统将做出错误的判断和行为。为了避免系统超负载带来的负面影响，要正确处理多个流的系统资源分配的问题。降载技术主要解决元组抛弃的时间、地点和多少，即 When-Where-How much-Which，是解决系统超负载最行之有效的方法。要将降载操作符添加到查询计划中才能真正实现降载。随机降载和语义降载主要解决抛弃哪类元组的问题。随机降载即 Random Load Shedding，系统随机抛弃部分负载；语义降载(Semantic Load Shedding)，系统抛弃非谓词的元组，需要合理构建谓词结构。数据处理中的降载策略。

4.1 数据处理中的降载策略

4.1.1 基于偶图的连接降载策略

更新和输出部分共同构成了基于偶图的连接降载策略，分别实现插入和输出元组的功能。可以用偶图来形象的描述连接降载策略，解决从偶图中求出最大子偶图的问题，就能解决降载问题。文献[39]证明该问题是 NP 难题，没有标准答案。在线策略采用在线试探语义降载算法，建立了一个输入流的频率模型，假设完全已知数据流的分布，并建立优先级队列。系统过载时，依次丢弃低优先级的元组。该策略假设完全已知数据流的分布，但实际数据流波动很大，与实际情况有很大偏差。此外，算法仅考虑了等值连接，忽略了非等值连接，仅考虑了两个数据流连接，略了多数据流的连接，适用范围很窄。

4.1.2 双窗口连接降载策略

CPU 周期资源耗费最大的部分是流连接运算中的探测操作。该策略会将到达的部分元组插入窗口，并不对该部分进行探测。因此，这部分元组可以连接对应流到达的元组，从而使结果子集输出最大化。

探测、插入、过期是连接各新到达元组的基本流程。执行连接的元组存储于连接窗口中。窗口计数器是实现降载的重要组件，能够统计各元组产生的连接结果，辅助和连接窗口大小相同，对于连接结果产生较少的元组，会将其删除，从而实现了降载的目的。

每个流的到达率可以通过检测器统计出来，输入流的波动通过输入队列的处理能够变得平缓。降载在队列长度超出阈值时被激活。常见的降载策略有两种：一种是后端降载，不探测输出结果较少的元组；另一种是前端降载，元组是否插入到队列根据概率进行判定。

要频繁调整窗口大小，以减缓相对速率波动较大的两个流。该策略 CPU 周期耗费大，虽然使用二叉树索引方法，在一定程度上降低了计算复杂度，但执行效率仍然不高。

4.1.3 基于语义近似的连接降载策略

基于语义近似的连接降载策略[40]模型有两种，即集成和模块连接模型，这两种模型既有区别又有相似之处。相同之处是这两种模型都有存储器和输入队列，前者用于存放元组的连接，后者用于存放新到达的元组；区别是具有不同的组件集成度。队列模型并不完全了解连接存储器，过载时，删除哪个元组由各流自行决定，统计模块会影响和制约决策信息。不同的流丢弃元组的策略是不一样的。集成模型有机融合了连接存储器和队列，虽然无法共享中间队列，但能够为降载决策提供更好的参考依据。

该策略与基于偶图的策略有很大相似之处，用偶图问题建模该问题。Kurotowski 图组成了偶图，连接方式采用等值的方式。PROB 试探法建立了优先级，根据输出元组数和两个流到达率，会丢弃低优先级的元组。与 PROB 试探法相同，LIFE 试探法也采用了这种建立优先级的方法，但是该方法把更高的权重赋予剩余生命周期。

该策略是具有很强的适用性和通用性，资源耗费小，但是当需要执行连接的流过多时，由于需要数据流模式的先验知识，时间和空间复杂度会急剧增加。

4.1.4 基于重要性的连接降载策略

基于重要性的连接降载策略[41]主要考虑了两个方面的重要性，即查询源和数据流元组。近似连接结果的最大化是连接查询的根本目的。要根据元组的重要性确定降载策略，并综合考虑连接属性的分布特征，主要有三种降载策略。

调整每个元组的优先级是 SIMP 方法的重要性特征。DIMP-PROB 试探法新元组的重要性是匹配数和重要性的乘积，匹配概率随着时间的推移实时更新。优先级累加公式是 DGL 策略的重要性特征，该策略忽略了连续时间域问题，只将等值连接作为参考依据。DGL 和 DIM-PROB 算法的时间和空间复杂度很大，要始终实时计算优先级，当系统过载时会造成系统堵塞等状况。

4.1.5 基于联合的连接降载策略

基于联合的连接降载策略[42]有本地约束和全局约束两类，即：每个流需要降载的量和需要降载的总量。前者属于本地约束，后者属于全局约束。IS 策略忽略了不同流之间的联系，独立的降载裁决每个流，无法保证输出的结果是最大子集。

BLAS 算法采用降载排序的方式，即按照一定的关系顺序排列所有的流。需要探测降载顺序空间，并选择降载顺序，以获得最大子集。当有很多输入流时，BLAS 算法的计算开销很大，执行效率很低。MxLF 算法采用流损失率，即计算丢弃部分数据后的输出结果。MLAS 采用多轮算法，重复迭代进行多次计算。GAS 计算每个流的数据项输出率，遵循全局约束条件，并建立等级表，等级表按输出率排序，丢弃数据表最后的数据实现降载。多轮策略也适用于 GAS 算法。基于联合的策略采用多轮技术，能够输出更多结果，充分考虑了各流之间的联系，具有很高的空间和时间复杂度，且没有明确的参考标准合理设置轮数。

4.2 降载策略的优劣性比较

应用高度会影响降载策略的实施，多样性是降载策略最鲜明的特点，无法直接比较不同降载策略的性能。如表 4.1 所示，总结并分析了不同降载策略的特点。

表 4.1 常用降载算法的比较

降载策略	降载位置	丢弃方式	丢弃策略	降载机制
基于偶图	处理之前	PNI	语义	动态
语义近似	处理之前	CF	语义	动态
重要性	处理之前	CF	重要性	动态
双窗口	入队之前	INP	语义	自适应
基于联合	处理之前	CF	语义	自适应

下列特点是连接降载策略性能优良的主要表现：

(1) 额外开销小、降载策略实施效率高。

(2) 能够满足数据流波动的情况，具有很强的适应性。

(3) 符合 DDSMS 系统特征变动的要求。

(4) 系统在降载时有较高的稳定性。

(5) 能够满足大量数据流的需求。

(6) 适合多种连接查询方式。

4.3 蚁群算法

4.3.1 蚁群智能优化算法

蚁群算法是从蚁群的觅食行为中研究出来的。在蚁群觅食初期，其觅食行为没有规律，处于混沌模式，具有很强的随机性[43-45]，但是当蚁群寻找到食物以后，无规律的觅食行为会逐渐呈现出组织规律性，有食物的路径上的蚂蚁会越来越多。经过大量研究表明，在觅食的过程中，蚂蚁释放了有关食物源的信息素，信息素的浓度决定了觅食的蚂蚁的路径选择，蚂蚁更倾向于选择信息素浓度大的路径。通过该路径的蚂蚁越多，信息素的浓度就越高；信息素浓度越高，就会有更多的蚂蚁被吸引过来，不断循环往复，形成一种正反馈机制。这种自催化剂导致越来越多的蚂蚁最终选择有食物的路径[46]。

蚂蚁具有通过最短路径寻找蚁穴和食物的能力，蚁群算法就是以蚂蚁仿生学的研究结果为依据，又称为蚂蚁算法。如图 4.1 所示，蚁穴可以用节点 B 来表示，食物源可以用节点 F 来表示，一共有 B-C-F、B-D-F 和 B-E-F 三条路径。将若干只蚂蚁放到节点 B，让他们爬行到节点 F，蚂蚁选择这三条路径的概率是一样的。蚂蚁会释放一定数量的信息素遗留在经过的路径上，随着时间的推移，信息素会按照一定速率挥发，信息素强度大的道路会增加蚂蚁选择该路径的概率。

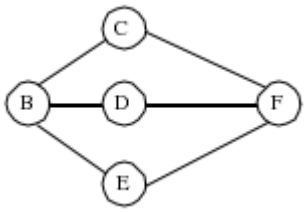

图 4.1 蚁群算法示意图

上图中有三条路径，即 B-C-F、B-D-F 和 B-E-F。其中，B-D-F 的路径最短，往返时间也最短。当蚂蚁选择 B-D-F 支路并按原路返回时，选择 B-C-F 和 B-E-F 的蚂蚁还没有返回，这就使其余两条支路上的信息素低于 B-D-F 上的信息素，增加了 B-D-F 支路被蚂蚁选中的概率。如此循环往复，信息素会在 B-D-F 支路上不断积累，B-C-F 和 B-E-F 支路上的信息素越来越少，从而得到最短路径 B-D-F，其余两条非最短路径逐渐消失。

4.3.2 蚁群智能优化算法的特点

为了解决特定的组合优化问题，提出了基于群体随机搜索的蚁群优化算法。蚁群算法的目标是建立一个可行解，使得该解的开销最小，并且解的序列式基于有效转移规则

的有效序列。

蚁群优化算法要求待解决的问题由一个图表示，且图由有限的节点和有限的边构成。每个节点代表解的一部分，每一条边表示从一个节点到另一个节点的转移。每条边都被赋予了一个开销值。蚁群优化算法的目标就是遍历该图来建立一条最短路径。所建立的路径代表一个构件序列-优化问题的一个解。路径是增量建立起来的，而每一个新节点的加入都经过了对部分路径的优化过程。

蚁群优化算法使用蚁群并行地对多个解进行搜索。在蚂蚁遍历图的过程中，每只蚂蚁增量地建立一个解。在此过程中，蚂蚁利用局部信息-信息素浓度和启发式信息，来决定下一个节点。与此同时，蚂蚁还修改它们所经路径上的信息素浓度，因此改变了环境，这些修改使得蚂蚁间可以间接地交换路径倾向性信息，从而达到合作目的。为了完成建立最优路径的任务，蚂蚁算法拥有如下特点：

(1) 蚂蚁有记忆功能来保存建立的路径信息。该记忆主要保证用于满足约束条件，比如每个节点只允许访问一次。该记忆还用于原路径返回，来释放信息素，增强对应边上的信息素浓度。

(2) 每只蚂蚁为每一个状态决定一些可选的邻接点。其中包括所有的从当前节点有效转移可到达的节点。之后蚂蚁进入一个新的状态。

(3) 每只蚂蚁都被分配一个初始状态，对应初始节点。

(4) 每只蚂蚁都有对应的一个或多个终止条件。终止条件包括：路径达到限定的最大节点数，找到了可接受的解，或者到达终节点。

(5) 每只蚂蚁根据概率转移规则从可选邻接点列表中选取下一节点。

(6) 每只蚂蚁都拥有修改所经历路径上的信息素浓度的能力。

4.3.3 基本蚁群算法的数学模型

所有边都在算法初始化的时候被赋予一定信息素。各条路径上的信息量决定了蚂蚁k(k=1,2,…,m)的转移方向和移动路径。蚂蚁 k 当前走过的节点用表 $tabu_k$(k=1,2,…,m)来记录。随着表 $tabu_k$ 的变化，集合做相应的动态调整。根据各条路径上的启发信息和信息量，计算状态转移概率。如式(4.1)所示，$P_{ij}^{k}(t)$表示蚂蚁 k 在 t 时刻从节点 i 到 j 的概率。

$$P_{ij}^{k}(t)=\begin{cases}\dfrac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum_{allowed_k}\tau_{is}^{\alpha}(t)\eta_{is}^{\beta}(t)}, & j\in allowed_k\\ 0, & otherwise\end{cases} \tag{4.1}$$

其中， $\tau_{ij}(t)$ 是信息素数量；allowed$_k$={V-tabu$_k$}，表示蚂蚁 k 下一步允许选择的节点； α 为信息启发因子； β 为期望启发式因子； $\eta_{ij}(t)$ 为启发函数，其表达式如下：

$$\eta_{ij}(t)=\frac{1}{d_{ij}} \tag{4.2}$$

为了避免启发信息被残留信息埋没，尽量减少冗余的残留信息素，要及时更新处理残留信息，可以在遍历 n 个节点或走完一步以后进行。具体过程如式(4.3)所示：

$$\tau_{ij}(t+n)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t) \tag{4.3}$$

在式(4.3)中， ρ 表示信息素挥发系数，1-ρ 表示信息素残留系数； $\Delta\tau_{ij}(t)$ 表示在本次循环中路径(i，j)上的信息素增量， $\Delta\tau_{ij}(t)$ 一般用以下模型来确定：

$$\Delta\tau_{ij}(t)=\begin{cases}Q/L+Q/(C_K-C_{BEST})，\text{蚂蚁k在本次循环中经过节点ij}\\ 0，\text{否则}\end{cases} \tag{4.4}$$

4.4 基于蚁群的降载算法设计

4.4.1 蚁群算法和分布式数据降载策略的关联

设 DDSMS 最大的数据处理负荷为 M，由于数据流的流速不稳定性，某些系统中的 DDSMS 的数据处理量超过 M，为了充分利用网络中的计算机节点的处理能力，特考虑当某一节点的数据流量超出该节点的处理能力时，可以快速寻找一条路径找到另外一个相对空闲的计算机节点来处理超出负荷 M 的数据流。所谓相对空闲的节点是指该节点目前所处理的数据流量和该节点所具有的最大处理负荷 M 有一定的差距。

经上节分析到，蚁群算法正是这样一种智能算法，可以寻找出符合条件的最优路径，使得超出负载的数据可以分流到其他计算节点进行处理，使得 DDSMS 的数据处理效率得到增强，还可以将闲置网络中的节点计算能力充分利用。

设某节点 DDSMS 的标准数据处理量为 M_0，该节点在某一刻所需处理的数据量为 M。

降载算法思想如下：

（1）若 $M<M_0$,说明未超出节点处理能力，由该节点中的 DDSMS 自动处理，无需降载处理。

（2）若 $M>M_0$，说明超出 DDSMS 处理负荷，需要降载处理。

（3）计算 $M_1=M-M_0$，并由蚁群算法寻找到最大空闲节点处理。此时如果该节点空余状态完全可以处理 M_1，则有该节点处理，否则计算未能处理的剩余任务 M_2，同样依照蚁群原理找出最优空闲节点，处理之。以此处理方式进行，直到 M_1 可以完全被网络中的其他节点承担处理。

4.4.2 基于蚁群优化后的降载算法设计

上面分析了蚁群算法原理和基于蚁群算法的降载策略，但是有一个问题未考虑。上述蚁群算法中，信息素最多的路径是蚂蚁通过的实际路径。如果最优路径恰好是这条路径，那么有数量庞大的蚂蚁从该路径经过，会迅速减少该路径上网络节点的能量，缩短整个网络的生命周期，甚至会堵塞该路径。根据蚁群算法的原理可知，蚁群算法很难直接找到全局最优解，容易陷入局部最优解，表现为其他节点无法处理完所需承担的数据任务，造成堵塞。如果改变网络拓扑结构，人工蚂蚁很难快速适应新的结构体系，无法找到更好的路径。因此，必须改进和优化蚁群算法，使所有路径都有数据流过，避免局部最优解的出现，均衡各节点的能量消耗。本文采取下列两种方法改进蚁群算法。

（1）根据公式(4.5)能够计算出概率 P_{ij}^k，表示第 K 只蚂蚁从节点 i 到 j 的概率。

$$P_{ij}^k=\begin{cases}\dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}d_{ij}E_j^r}{\sum_{allowed_k}\tau_{is}^{\alpha}\eta_{is}^{\beta}d_{is}^{\varphi}E_s^r}, j\in allowed_k\\ 0, otherwise\end{cases} \tag{4.5}$$

其中，$\tau_{ij}(t)$ 是信息素强度；$\eta_{ij}(t)$ 是从节点 i 到节点 j 的距离的倒数；d_{ij} 是节点 i 到节点 j 的时延的倒数；E 是蚂蚁 k 即将访问的下一个节点的剩余能量；$allowed_k$ 是蚂蚁 k 还未访问节点的集合；α 是表示信息素相对重要程度的参数，其值越大，该蚂蚁越倾向于选择其他蚂蚁经过的路径，蚂蚁之间的协作性越强；β 表示能见度的相对

重要性，反映了蚂蚁在运动过程中，启发信息在蚂蚁选择路径中的受重视程度，其值越大，该状态转移概率越接近贪心规则；γ是表示能量相对重要程度的参数。

将能量控制因子加入到式(4.5)中，使整个网络的生命周期增长，节点的能量消耗得到平衡，且保证了计算结果收敛于最优解。网络的拥塞程度可以通过时延因素直观的显现出来。式(4.5)充分考虑了时延因素的影响，对缓解网络拥塞有极大的推动作用。

(2) 更新信息素的规则如下：无论行走路径和方向如何，蚁群算法的信息素一定会积累和增强，与搜索结果和搜索路径无关。早期发现的优质解更容易影响蚁群算法的搜索结果，使算法陷入局部最优解，无法找到全局最优解，因为随着时间的推移，没有搜索到更优解的信息素会挥发甚至消失，并不会增强。本文在更新方法中融入了奖罚机制，有效解决了陷入局部最优解的问题。在搜索过程中，如果没有发现更优解，则进行适当的惩罚；如果发现更优解，则进行适当的奖励。

如式(4.3)所示，为信息素更新公式， $\Delta\tau_{ij}(t)$由式(4.6)确定：

$$\Delta\tau_{ij}(t)=\begin{cases}(Q+C_{BEST}-C_K)/L, & \text{蚂蚁k在本次循环中经过节点ij}\\ 0, & \text{否则}\end{cases} \tag{4.6}$$

其中，C_K为第 k 次搜索的费用；C_{BEST}为发现的最好费用。

4.5 算法实现

经过以上的分析可知，经过优化的蚁群智能算法可以有效的减少网络拥堵，能够最大限度的利用网络中的闲置计算能力。

4.6 本章小结

本章介绍了常用的一些降载策略，分析了这些策略的优缺点，总结并分析了蚁群智能算法的优缺点和特性，提出了基于蚁群算法的降载策略并实现之。

第五章 蚁群降载算法在邮包传送系统中的应用

为了检测基于蚁群算法的降载策略在实际应用中的效果，本文考虑用这样一个实际系统-邮包传送系统:对不同方向的邮包传输数据进行在线处理，利用这个实际的应用场景检测该降载机制的有效性。

5.1 邮包传送系统

5.1.1 系统描述

本系统是基于分布式的邮包收发数据处理系统。该系统是在遵循一个特定通信协议的基础上，由四个邮包数据的子站向监测系统主站实时发送邮包信息，主站收到邮包数据后进行过滤处理并实时显示最新的邮包数据。

本系统的基本要求是：

(1) 能实时显示邮包所在地

(2) 能存储 3 天的邮包信息

(3) 能灵活打印报表

(4) 能接收邮包信号

(5) 邮包不少于 4 个方向

5.1.2 系统总体设计

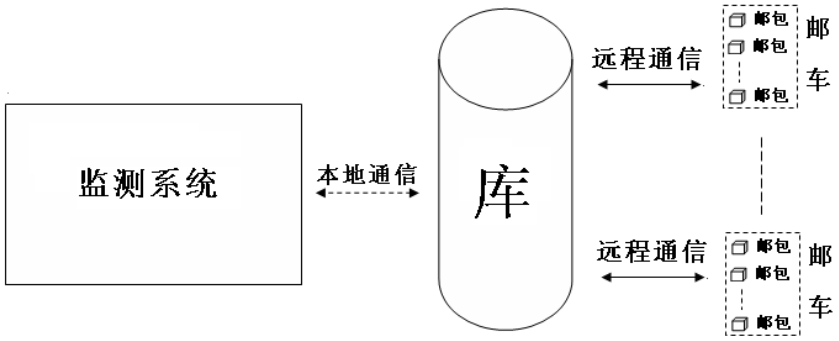

图 5.1 邮包传送系统结构

(1) 邮包是模拟信号；

(2) 库在通信协议的基础上存储邮包信息；

(3) 检测系统用来按需数据信息；

(4) 库和邮包之间采用双向通信，检测系统和库之间为本机双向通信。

5.2 主要功能模块

5.2.1 编址

表 5.1 编址结构

地址前缀	接口标识

(1) 地址前缀：地址前缀能够实现路由分配整个 IPv6 网络，表示该地址所属的子网络，定义为 IPv6 地址的前 64bit。

(2) 接口地址：接口地址用来标识子网络的节点，地址空间约为 185 亿，最多可以为同一子网定义 2^{64} 个节点，定义为 IPv6 地址的后 64bit。

在接口标识 64 位中，根据需要，设计包括如下基本信息：

表 5.2 接口信息

目的地	出发时间	出发地

目的地：邮包寄往的城市。

出发时间：邮包第一次进入邮局扫描时的时间。

出发地：邮包下单的节点邮局。

用这 3 部分即可唯一标识邮包，物物相连的网络中，出发地可以用以前的 IPv 表示，最多需要 32 位，初定用 4 个字节表示。出发时间至少需要 20 位，初定个字节(24 位)目的地用当地邮编表示，需要 1 个字节表示。

5.2.2 数据库设计

数据库的作用是存储相关数据并供用户读取。本系统的数据库设计分为三级。

三级数据库是指各邮车的数据库，存储着经过此邮车的所有邮包数据信息。

二级数据库介于监测系统和邮车之间，专门用来接收邮包信息并归类处理。采集存储各邮车的邮包数据信息，主要作用是减少监测系统的负荷，提高用户获取数据的速度，起到缓存作用。

一级数据库是监测系统的数据库。当用户查询数据时，监测系统直接到二级数据库中查询所需数据并存储供用户查找。此数据库中除了存储邮包数据信息外，还可以存储

一些用户经常需要查询的数据，比如区号对应、邮资费用等。

根据以上分析，本系统的数据库表按照以下几个主体设计：

（1）用户

表 user 用于存储注册用户的资料信息，其主要属性包括(uid，name，sex，phone，code，level，flag)。

uid 是表 user 的主键，唯一标识一个用户。

Name 是用户姓名

sex 是用户性别

phone 是用户联系号码

code 是用户所在地区的邮编

level 是用户的等级

flag 是用户的审核状态

（2）邮包数据

表 parcel 用存储邮包的数据信息，其属性包括(id, begin, destination, location, time_begin, time_receive, flag)。

id 是邮包的唯一标识符，对应的即是每个邮包的 IPV6 编址

begin 是起始地

destination 是目的地

location 是目前所在位置

time_begin 是邮包开始发出的时间

time_receive 是邮包最后被接收到的时间

flag 是接收状态

id+zipcode 构成 parcel 表的主键，用来唯一标识某个邮包

（3）区号对应

表 code 用来存储区号对应地区的数据，其主要属性包括(id,postcode,area)

Id 是本表主键，由数据库自动生成

Postcode 是区号

Area 为对应地区名称

（4）区间邮资

表 fax 用来存储各区间的邮资费用，其主要属性包(id,begin,destination,money)

Id 是本表主键，由数据库自动生成

Begin 为区间起始地名

Destination 为区间目的地名

Money 为该区间的费用

5.2.3 监测

针对邮局管理员和客户两种不同群体，本系统设计了两种具有不同查询权限的会员系统，即一般注册用户和超级用户。在查询之前，用户须在本系统注册，提交个人基本资料信息。超级用户的用户名和密码在本系统中事先给定。

一般注册用户可按邮包发出地点或者发出地点的邮编、邮包接收地点或者接受地点的邮编、邮资查询基本的数据信息。

超级用户除了可以执行一般注册用户的查询请求外，还可以执行一些高级查询，比如：可以按照区间查询经过此区间的邮包信息、查询经过某固定地点的所有邮包信息、查询某特定邮包从始发地点到目的地点所经历的所有地区列表、比较在一个相同区间，从不同路径传送所经历的时间长短和邮资费用。

现根据系统要求，设计主要界面如图 5.2 所示：

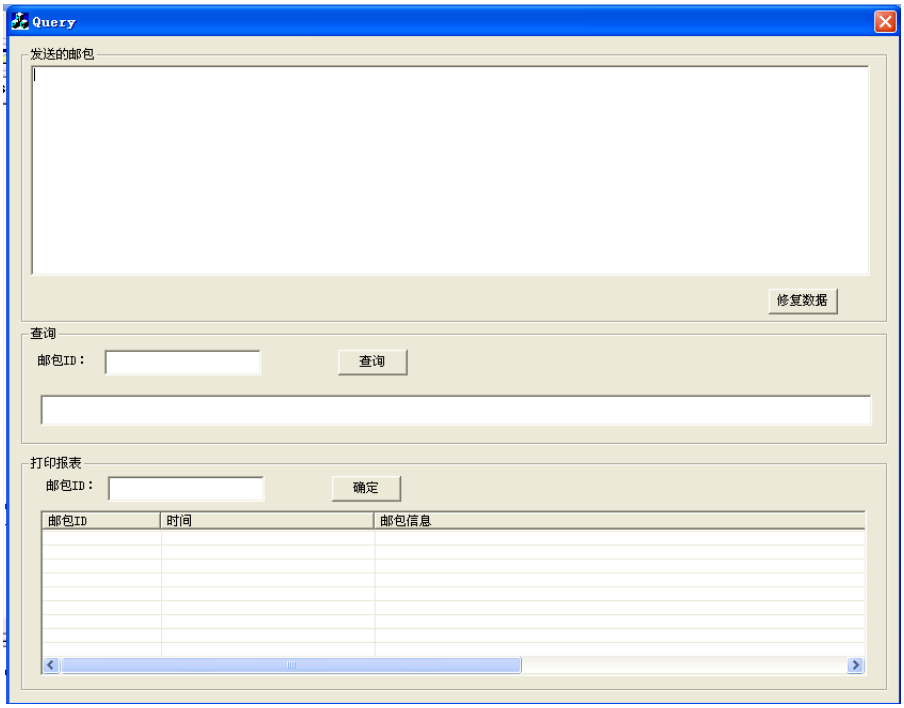

图 5.2 邮包发送界面

5.3 实验分析

实验主要针对网络中 DDSMS 的负载处理能力进行测试。

通过调整分配具有闲置 DDSMS 处理能力的节点来改变系统的负载:单位时间内数据输入量为 τ 时，系统刚好不过载，即系统单位时间内能处理的数据量为 τ，此时系统的负载量为 M;如果将单位时间内输入数据量提高到 2τ，则此时的系统负载量为 2M，依次类推。

对于该系统的蚁群降载策略稳定的评价标准是：

$$\text{负载变化率} = \frac{\text{系统实际处理的数据量}}{\text{进入系统的总数据量}}$$

当负载变化率的数据波动比较稳定的时候，说明系统能够很好的处理增加的数据量，证明了系统降载策略的有效性。

经过实验反复测试，随着系统负载量的变化，系统负载的变动情况如图 5.3 所示(横向坐标 X 表示进入系统的总数据量，纵向坐标 Y 标识负载变化率)。从图 5.3 中可以看出，随系统负载量增加负载波动率增长缓慢，说明系统具有较好的负载处理能力。

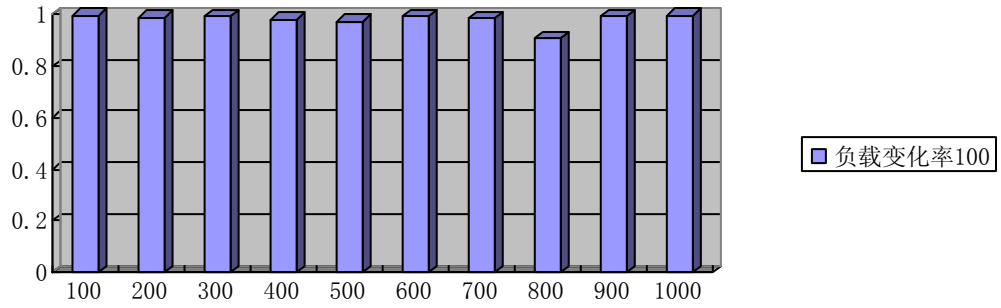

图 5.3 系统负载变化图

5.4 本章小结

本章简单介绍了一个具体的分布式数据处理系统－邮包发送系统，并将蚁群降载算法应用于该系统，模拟测试出了该降载策略使得处理系统的负载波动率平稳，说明基于蚁群降载算法的邮包发送系统具有良好的负载处理能力。

第六章 总结与展望

6.1 工作总结

本文研究了分布式数据处理系统的几个重要方面，主要工作如下：

(1) 分析了国内外相关处理系统的原型。例如：斯坦福大学研制的 STREAM 系统，麻省理工和布朗大学研制的专门用于数据流监控的 Aurora 系统，美国加州大学研制的专门处理连续数据流的 TelegraphCQ 系统等。

(2) 设计了一种通用的 DDSMS 方案，并对基本功能模块做出简要分析。

(3) 设计了控制滑动窗口规模的动态查询算法，以限制每个输入流存储下来的元组数量，达到最小的内存空间和处理时间的消耗。

(4) 分析和比较了数据处理中常用的降载策略，设计了基于蚁群的降载算法。

(5) 在邮包发送系统的项目上应用了基于蚁群的降载算法，并模拟验证了该算法的有效性。

6.2 展望

虽然分布式数据处理系统已经有了一点进展，还没有形成一整套完整的方法和规范，还需要在如下方面做进一步的研究改进：

(1) 本文以滑动窗口作为概要数据构建的方法，而且滑动窗口可以动态调整规模，可以很好的提高查询的效率，但是尚未可虑对历史数据的应用，因此如何提供可靠的历史数据以满足金融等领域的应用，是一个重要的研究方向。

(2) 在设计中，我们假设了数据流到达是依照时间或元组有序的，但在现实中，由于网络的不确定因素，数据到达常常是无序的，这就需要一种措施能够对无序的数据流进行排序或在数据流无序的情况下得出准确结果。

(3) 继续完善数据流管理系统的功能，不断提高该系统的性能。还需要继续深入研究该系统的体系结构和接口、优化查询模式、合理分配内存资源、数据挖掘的实际应用等问题。

因笔者知识储备和实际经验有限，无法全面深入了解问题本质，还请各位专家批评指正，我会继续努力，不会停止前进的步伐。

参考文献

[1]Andrew S.Tanenbaum,Maarten Van Steen著,辛春生等译.分布式系统原理与范型[M].清华大学出版社,2009.

[2]宋国杰,唐世渭,杨冬青等.数据流中异常模式的提取与趋势监测[J].计算机研究与发展,2004,41(10):1754-1759.

[3]郭龙江,李建中,王伟平等.数据流上的连续预测聚集查询[J].计算机研究与发展,2004,41(10):1690-1695.

[4]王栩,李建中,王伟平.基于滑动窗口的数据流压缩技术及连续查询处理方法[J].计算机研究与发展,2004,41(10):1639-1644.

[5]张冬冬,李建中,王伟平等.分布式复式数据流的处理[J].计算机研究与发展,2004,41(10):1780-1785.

[6]金澈清,钱卫宁,周傲英.流数据分析与管理综述[J].软件学报,2004,15(8):172-181.

[7]Motwani R, Widom J, Arasu A. Query processing, resource management, and approximation in a data stream management system [J].CIDR,2003,42(23):132-136.

[8]Arasu A, Babu S, Widom J. The CQL continuous query language: semantic foundations and query execution [M].Stanford University Press,2003.

[9]Srivastava U, Widom J. Flexible time management in data stream systems [C]. Proc. of the 2004 ACM Symp On Principles of Database Systems, New York ACM Press,2004: 263-274.

[10]Eduardo F, Mario P. Designing secure databases [J]. Information and Software Technology, 2005,57(47):463-477.

[11]Babcock M. Monitoring streams-a new class of data management applications [C]. In Proc. Int. Conf. On Very Large Data Bases, Hong Kong,2003:245-256.

[12]Chandrasekaran S, Cooper O, Deshpande A .Telegraphy:continuous dataflow Processing for an uncertain world[C]. ln Proc. Conf. on lnnovative Data syst.Res, Asilomar,2003:269-280.

[13]Avnur R, Hellerstein J . Eddies: continuously adaptive query processing[C].In Proceedings of the 2000 ACM SIGMOD international conference on Management of data,Dallas,2000:261-272.

[14]Madden S R, Shah M, Hellerstein J M. Continuously adaptive continuous queries over streams[C]. In Proceedings of the 2002 ACM SIGMOD international conference on Management of data,2002:49-60.

[15]Chandrasekaran S,Franklin M. A system for streaming queries over streaming data [J]. The VLDB Journal, 2003, 12(2):140-156.

[16]Krishnamurthy S, Chandrasekaran S, Cooper O. Telegraphy:an architectural status report [J]. IEEE Data Engineering Bulletin, 2003, 26(1):11-18.

[17]Liu L, Pu C, Tang W. Continual queries for internet-scale event-driven information delivery [J]. In IEEE Trans.Knowledge and Data Eng, 1999, 11(4):610-628.

[18]Chen J, DeWitt D. A scalable continuous query system for internet databases[C]. In proceedings of the 2000 ACM SIGMOD international conference on Management of data, Dallas, 2000:379-390.

[19]Charikar Lukasz. Continuous queries over append-only databases [C]. In Proceedings of the 1992 ACM SIGMOD international conference on Management of data, San Diego, 1992:321-330.

[20]Muthukrishnan S, Kotidis Y. An architecture for queries over streaming sensor data [C].In Proceedings of the 2002 international conference on data enerneering, 2002:555-566.

[21]Yao Y, Gehrke J. Query processing for sensor networks[C]. In Proc. Conf.on lnnovative Data system.Res, Asilomar, 2003:233-244.

[22]Cranor C, Gao Y, Johnson T. High performance network monitoring with an SQL interface [C]. In Proc. ACM Int. Conf. on Management of Data, Madison, 2002:623-623.

[23]Zhu Y, Shasha D. Statistical monitoring of thousands of data streams in real time [C].In Proc. Int. Conf on Very Large Data Bases, Hong Kong, 2002:358-369.

[24] 张健沛. 数据库原理及应用系统开发[M].电子工业出版社,1999.

[25]Matias Y, Vitter S, Wang M. Wavelet-based histograms for selectivity estimation [C]. In Proc.of SIGMOD,1998:127-131.

[26]Matias Y, Vitter S, Wang M. Dynamic maintenance of wavelet-based histograms [C]. In Proc.of VLDB, 2000:241-249.

[27]Chandrasekaran S, Franklin M J. Streaming Queries over Streaming Data [C]. Proc. International. Conf. on VLDB, 2002:203-214.

[28]周明中,龚俭.数据流管理系统综述[J].计算机工程,2006, 32(2):10-12.

[29]Guha S, Koudas N. Approximating a data stream for querying and estimation: Algorithms and performance evaluation [J]. IEEE Computer Society, 2002, 10(2):567-576.

[30]Golab L, Ozsu T. Processing sliding window multi-joins in continuous queries over data streams[C]. Proc. of the 29th Int'l Conf. on Very Large Data Bases Berlin: Morgan Kaufmann Publishers, 2003:500-511.

[31]Viglas S, Naughton J, Burger J. Maximizing the output rate of multi-way join queries over streaming information sources[C]. Proc. of the 29th Int'l Conf. on Very Large Data Bases Berlin: Morgan Kaufmann Publishers, 2003:285-296.

[32]金澈清,钱卫宁,周傲英. 流数据分析与管理综述仁[J]. 软件学报,2004,15(8):1172-1182.

[33]陈慧南. 算法设计与分析-C++语言描述[M]. 电子工业出版社,2010.

[34]Huaguang Zhang, Dedong Yang, Tianyou Chai. Guaranteed cost networked control for T-S fuzzy systems with time delays [J]. IEEE Trans on Systems, Man and Cybernetics: Part C, 2007, 37(2): 160-172.

[35]Zhanzhi Qiu, Qingling Zhang, Zhiwu Zhao. Stability of singular networked control systems with control constraint [J]. J of Systems Engineering and Electronics, 2007, 18(2):290-296.

[36]Rabello A, Bhaya A. Stability of asynchronous dynamical systems with rate constraints and applications[J]. IEEE Proc of Control Theory Application, 2003, 150(5): 546-550.

[37]Ayad A, Naughton J , Wright S. Approximating streaming window joins under CPU limitations[C]. ICDE, 2006:43-56.

[38]李建中,张冬冬. 滑动窗口规模的动态调整算法[J]. 软件学报,2004,15(12):1800-1806.

[39]Ayad M, Naughton J, Wright S. Approximating streaming window joins under CPU limitations technical report #1542[R]. University of Wisconsin-Madison Computer Sciences Department, 2005.

[40]Das A, Gehrke J, Riedewald M. Semantic approximation of datastream joins [J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 7(1):44-59.

[41]Adegoke O, Qiang,Hou Zhu,en-Chi. Window join approximation over data streams with importance semantics [C]. International Conference on Information and Knowledge Management, Arlington, VA, United States, 2006:112-121.

[42]Xiaochun Yang, Lin Li. Associated load shedding strategies for computing multi-joins in sensor networks [C].The 11th international conference on database systems for advanced applications,2006:11-15.

[43]Dorigo M, Maniezzo V, Colorni A. Ant System: Optimization by a Colony of Cooperating Agents [J]. IEEE Transactions on System,Man,and Cybernetics-Part B,1996,26(1):29-41.

[44]Ganbardella L, Taillard E, Dorigo M. Ant colonies for the QAP [R]. Technical report, IDSIA,Lugano, Switzerland, 1997.

[45]Nemes L, Roska T. A CNN model of oscillation and chaos in ant colonies: a case study [J]. IEEE transactions on circuits and systems: fundamental theory and applications, 1995, 12(10):741-745.

[46]Dorigo M. Learning by probabilistic boolean networks [C]. In Proceedings of the IEEE International Conference on Neural Networks, 1994:887-891.

附录 1 攻读硕士学位期间撰写的论文

(1) 朱沙沙. 基于 XML 的异构数据库复制方案[C]. 全国计算机继续教育研究会,2012(5):109-112.

附录 2 攻读硕士学位期间参加的科研项目

(1)软件开发环境国家重点实验室开放课题：基于中介逻辑的潜无限系统理论与计算机应用基础研究(批准号：SKLSDE-2011KF-04)；

(2)国家自然科学基金面上项目：过渡现象的处理方法及其逻辑基础的研究(批准号：61170322)。

致谢

幸福的时光总是短暂的，我的研究生生涯也已接近尾声。此时的心情十分复杂。首先，我要感谢我的导师洪龙教授，洪龙教授对我的帮助和指导让我终身受益。在学习上，洪龙教授孜孜不倦的传授我科学文化知识，为我指引正确的前进方向，培养我的自学能力和钻研能力。在生活上，洪龙教授无微不至的关怀让我感受到了人间的真情和温暖，让树立了面对和克服困难的勇气。在思想上，洪龙教授教会我淡然的心境、认真的态度、严于律己的作风，让我养成了良好的行为习惯。

其次，我要感谢我的朋友们，在我遇到困难的时候伸出帮助之手，在我心情低落的时候让我开怀大笑，在我成功的时候跟我一起分享胜利的果实。

再次，我要感谢实验室的兄弟姐妹，陪伴我一起度过了研究生的生活，一起建设了属于我们的“家”，共同分享生活中的酸甜苦辣。

最后，我要感谢我的父母家人，全力支持我的学习，为我分担生活上的压力，帮我解决人生道路上遇到的困难。有你们的支持，我才能更加专心的学习。

分布式数据处理系统的研究与应用

作者： 朱沙沙
学位授予单位： 南京邮电大学

参考文献(8条)

1. 宋国杰,唐世渭,杨冬青,王腾蛟 数据流中异常模式的提取与趋势监测[期刊论文]-计算机研究与发展 2004(10)

2. 郭龙江,李建中,王伟平,张冬冬 数据流上的连续预测聚集查询[期刊论文]-计算机研究与发展 2004(10)

3. 王栩,李建中,王伟平 基于滑动窗口的数据流压缩技术及连续查询处理方法[期刊论文]-计算机研究与发展 2004(10)

4. 张冬冬,李建中,王伟平,郭龙江 分布式复式数据流的处理[期刊论文]-计算机研究与发展 2004(10)

5. 金澈清,钱卫宁,周傲英 流数据分析与管理综述[期刊论文]-软件学报 2004(08)

6. 周明中,龚俭 数据流管理系统综述[期刊论文]-计算机工程 2006(02)

7. 金澈清,钱卫宁,周傲英 流数据分析与管理综述[期刊论文]-软件学报 2004(08)

8. 李建中,张冬冬 滑动窗口规模的动态调整算法[期刊论文]-软件学报 2004(12)