中图分类号：TP393　　论文编号：183131371
学科分类号：520.60　　密　　级：

天津理工大学研究生学位论文

数据流聚类算法在 Web 访问日志分析中的应用研究

（申请硕士学位）

专业学位类别：工程硕士

专业(领域)：计算机技术

研究方向：网络数据分析与挖掘

作者姓名：张倩

校内指导教师：王法玉 副研究员

企业指导教师：朱玉来 高级工程师

2021 年 3 月

Thesis Submitted to Tianjin University of Technology for the Master's Degree

Research and Application of Data Stream Clustering Algorithm in the Analysis Web Access Log

By

Qian Zhang

Supervisor

Fayu Wang

Yulai Zhu

March 2021

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，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 天津理工大学 或其他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。

学位论文作者签名：[illegible] 签字日期：2021 年 3 月 1 日

学位论文版权使用授权书

本学位论文作者完全了解 天津理工大学 有关保留、使用学位论文的规定。特授权 天津理工大学 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，并采用影印、缩印或扫描等复制手段保存、汇编，以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复本和电子文件。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：[illegible] 导师签名：[illegible]

签字日期：2021 年 3 月 1 日 签字日期：2021 年 3 月 1 日

摘要

近年来互联网技术取得了惊人的发展成绩，各类使用也日益普及，在使用互联网的过程中产生了大量的 Web 数据，如何从这些海量数据中统计和分析出有价值的信息，挖掘日志数据中蕴藏的访问行为是一个值得研究和关注的热点问题。对这一问题的研究有助于网站管理员及时发现网站安全风险，修复网站漏洞，不断提升网站运维人员的网络安全意识，也有助于管理员及时了解用户关注的网站内容，及时更新维护相关内容，更好发挥网站的作用。

本文针对 Web 访问日志分析的问题，对数据流相关的概念及算法进行研究，提出数据流聚类算法优化方法，对现有的基于密度网格的数据流聚类算法进行改进。然后在充分研究 Storm 框架基本原理的前提下，将改进算法基于 Storm 进行分布式并行化设计，提高改进数据流聚类算法的处理效率。最后将改进的分布式数据流聚类算法在实际采集的 Web 访问日志的分析上加以应用。

本文的主要研究内容如下：

（1）选择基于密度网格的数据流聚类算法作为 Web 访问日志分析的基础算法。针对现有算法在阈值参数设置和簇边界判定两个方面的不足，对现有算法进行改进，以提高现有算法聚类性能为目的，提出改进的基于密度网格的数据流聚类算法。

（2）针对无法在单机环境下高效处理 Web 访问日志这类实时、海量数据的问题，搭建分布式流处理平台 Storm，以并行化分布式的方式来设计数据流聚类算法，并且基于 Storm 来实现该算法。

（3）将基于 Storm 实现的分布式数据流聚类算法在 Web 访问日志分析中应用，设计 Web 访问日志分析模型。基于实际采集的校园网站 Web 访问日志进行实验，结果表明，改进的算法聚类效果更优，并行化的计算更好地适应了 Web 访问日志的数据特点，算法具有分布性、实时性和准确性。得出的分析结果对网站的管理具有一定的参考价值，可以为网站管理中的类似问题提供参考依据。

关键词：Web 访问日志，数据流，Storm 计算框架，实时聚类，密度网格

Abstract

In recent years, Internet technology has made striking achievements, and its usage has become increasingly popular. In the process of using the Internet, there must be generated a large number of Web data. Therefore, how to mine valuable information from those big data, mine access behavior from log data is a hot issue worthy of research and attention. Research on this issue will help website administrators discover website security risks timely, fix website vulnerabilities, and continuously improve the network security awareness of website operation and maintenance personnel. It also helps administrators understand the content of the website that users are concerned about and update maintenance content timely, to better play the role of the website.

In view of the problem of Web access log analysis, this paper makes in-depth research on the related concepts of data stream, and develops the optimizing methods for further improving the data stream clustering algorithm. Then, on the premise of comprehensively studying the basic principles of the Storm framework, designs distributed and parallel algorithm based on Storm to improve the processing efficiency of the improved data stream clustering algorithm. Finally, applies the improved distributed data stream clustering algorithm to the analysis of actual collected Web access logs.

The main research contents included of this paper are as follows:

(1) In this paper, the data stream clustering algorithm based on density and grid is used as the basic algorithm, and then the shortcomings of the basic algorithm (including threshold parameter setting and cluster boundary determination) are fully considered to further optimize the basic algorithm. Based on the density and grid data stream clustering algorithm designs an improved algorithm to improve algorithm's cluster performance.

(2) Aiming at the problem that it is unable to efficiently handle the real-time and massive data such as web access logs in a single-machine environment, the distributed stream processing platform Storm is built, Data stream clustering algorithm is designed in parallel and distributed way, and implemented based on Storm.

(3) In terms of web access log analysis, apply the distributed algorithm based on Storm, and the relevant model is designed. Experiments are carried out based on the actual collected campus website Web access logs. It indicates that this algorithm has

much better clustering effect, and the parallelized calculation better adapts to the data characteristics of Web access logs. The algorithm is distributed, real-time and accurate. For website management, the results have important reference significance. It can provide the necessary reference scheme for the effective solution of related problems.

Key words: Web access log, Data stream, Storm computing framework, Real-time clustering, Density and grid

目　录

第一章 绪论

1.1 研究背景与意义

目前，互联网取得了惊人的发展及普及成果，自万维网（WWW，World Wide Web）于 1991 年诞生以来，现在已经发展成为拥有亿万活跃用户和运营站点的巨大信息资源库，Web 是一个具备海量潜在知识价值的信息空间[1]。根据图 1.1 所示，Netcraft 所公布的关于全世界范围内的 Web 服务器数据报告表明，2020 年 7 月 Netcraft 共收到 1234228567 个站点、260658118 个域和 10221919 个面向 Web 的计算机数据，近年来服务器站点总数量和活跃站点数量持续保持增长趋势，21 世纪是一个 Web 数据激增的时代[2]。

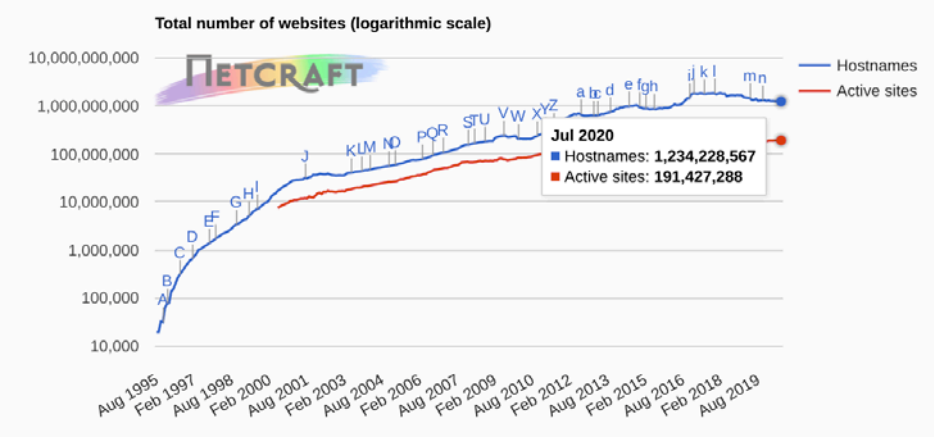


图 1.1 截至到 2020 年 7 月全球 Web 服务器统计

各种运营网站、通讯应用、社交平台等互联网工具被人们广泛使用，在日常使用的过程中 Web 数据无时无刻不在产生，大量的 Web 数据通过日志的形式记录和存储[3]。Web 日志文件包含着网站最重要的信息，从 Web 日志中既可以获得访问者感兴趣的内容，又可以分析是否存在网站漏洞攻击[4]。如何对数据量巨大且不断到达的日志数据进行分析，了解访问者在网站的访问情况，从庞大的用户信息中挖掘用户访问信息，提高 Web 站点的服务质量，对于网站的管理者来说是值得关注的问题。

将数据挖掘技术应用在 Web 上能够挖掘出一些隐含的信息，对这些信息进行分析能够更好地应用于优化网站的组织结构和空间设计[5]。运用算法来从大量数据之中将隐匿的信息搜索出来，并对各种未知模式进行捕获是数据挖掘的主要

研究内容[6]，而常规的数据挖掘技术通常面向的是结构化数据。对于 Web 挖掘来说，理论基础是数据挖掘方面的各类技术[7]，将数据挖掘的应用领域延伸至网络，使所处理的数据对象涵盖至非结构化的数据，包括存在于网页之中的图片、视频以及文本等，通过合理的算法来展开分析，对那些价值较大且潜在的相关模式进行有效地捕获。内容主要涉及到对 Web 的日志、结构以及内容进行挖掘[8]。在进行 Web 日志挖掘的过程中，主要是以 Web 服务器和客户端记录下的日志数据为研究对象，将那些具有较高价值、尚未被发现且十分丰富的相关信息从日志之中抽取出来。这些信息包括独立 IP 数、网页访问量等基础信息以及用户行为特征等深度信息。

目前 Web 日志挖掘领域的热点研究内容主要集中在对用户访问路径进行挖掘和根据用户访问内容对用户进行聚类这两个方面[9]，本文重点对校园网站 Web 访问日志进行分析，相比较挖掘用户的访问路径，更关注用户的访问行为，所以本文重点对用户聚类这方面的方法进行研究。在数据挖掘的研究范围之内，聚类分析属于十分重要的方法之一，聚类就是将某个集合中的所有数据按照某种判断数据相似度的方法划分成若干个类别，最大化类别之中的数据相似性，并且最小化各个类别之间所存在的数据相似性[10-12]。将聚类方法应用于 Web 访问日志的分析，可以从 Web 访问日志中聚类出具有相似特性的用户或数据项，统计行为相似用户的共性行为，对这些行为进行分析，得到的聚类结果能够致力于网站的开发和管理。

然而，传统聚类算法只能对数据进行静态处理，做不到随时间变化得到簇的形状，并且面对海量的日志数据，单机环境下的处理开始展现出延迟性高、扩展性不佳等缺点。针对 Web 访问日志实时到达、不断递增、数据量巨大的特点[13]来选择和完善聚类算法，并基于大数据处理工具对日志进行分析具有重要意义。

本文结合 Web 访问日志的数据特点，基于当前的密度网格数据流聚类算法，提出改进算法。而且以 Storm 为基础来开展改进算法的并行化分布式的设计工作，最终应用到对实际采集的校园网站 Web 访问日志的分析中，深入挖掘网站用户的访问行为，辅助网站的管理。

对 Web 访问日志进行分析具有以下几点的现实意义：

（1）根据 Web 访问日志的实际数据特征，选择使用数据流聚类算法对日志数据进行分析，算法处理流程与流数据特征相适应，可以为使用数据挖掘技术分析日志这类流数据提供有益的思路和方法。

（2）将数据挖掘中的数据流聚类算法和分布式流数据处理平台等前沿技术相结合，避免因数据量过大导致的算法计算灾难，更好的适应日志数据这类大数据量的计算。

（3）使用自动化分析手段辅助网站管理，避免人工方式的低效率操作，使管理者及时了解网站访问者的访问行为与习惯，对网站的建设有宏观的把握，减少网站被攻击的风险，增强网站的整体性能。

1.2 国内外研究现状

本节主要从 Web 日志分析和数据流聚类算法两个方面，对国内外研究现状进行介绍。

1.2.1Web 日志分析

它是指将那些看起来不存在规律性的日志信息从 Web 服务器之中有效地提取出来，得到具有较高价值的知识与信息，进而根据所得结果来优化网站。近年来，一些国内外学者对 Web 日志分析进行了研究，发展了一系列新型的日志分析方法，包括聚类、分类、关联规则、序列模式以及统计等分析方法[14]。

实际上，统计分析是一种较为基础性的日志分析法，主要是针对相关的属性来开展指标统计，具体包括用户停留时长、访问次数以及 IP 地址总数，等等。目前国内外有很多利用统计分析方法对日志进行分析的工具，国外的 Webalizer 可通过使用网站服务器中与网站访问和使用相关的日志来实现 HTML 格式的分析网页的生成，并且对一系列信息进行统计，包括：服务器数据的下载量，访问来源，访问量以及点击率等方面。GoogleAnalytics 则可实时统计网站访问者的相关信息并且能够显示当前的系统的运行状况。在国内，高伟伟[15]设计了基于 ELK 的 Web 日志安全分析平台，基于 Elasticsearch、Logstash 和 Kibana 工具对日志进行采集、传输和解析。曾新励等人[16]以 Hadoop 平台作为基础，设计了新型的 Web 日志分析系统，该系统为分布式的。

以统计分析为基础的 Web 日志分析方法大多通过脚本或代码的方式对日志进行筛选统计，只能对已知属性的 Web 日志进行统计，无法做到对未知类型 Web 日志的识别。并且目前大多数日志分析系统都是单机环境的，处理海量数据时性能会略显不足，为了满足大数据分析的需求，对日志的分析开始转向分布式计算的方向。

考虑统计分析方法的不足，Chen 等人[17]创新性地在该领域当中应用了数据挖掘技术，进行分析的时候，将那些无关于用户行为路径分析的主要日志文件删除掉，对有关的用户访问行为进行深入的挖掘。由此，Web 日志分析与数据挖掘技术开始结合。

在国外，Janisa Colaco 和 Jayashri Mittal [18]提出将 K-harmonic means 算法和

FP growth 算法结合，对用户接下来的访问请求进行预测。Ansari,ZA[19]及合作者发展了以模糊神经网络(缩写为 FNCN)作为基础的新型框架，它可被用来聚类从教育机构服务器相关网络日志里的访问数据。而 Guo 及合作者[20]则是结合页面及用户聚类，基于矩阵聚类与 K-means 等两种算法的分析，发展了改进型聚类算法，使有关的问题得以很好地解决，此外，还使噪声数据发生阻塞的相关缺陷也得以解决。杨勇及合作者[21]分析了经典 K-means 算法，尤其是探讨它挖掘处理庞大 Web 日志数据的过程中存在的具体问题，发现存在的核心性问题，基于 Hadoop 云平台，与粒子群优化(缩写为 PSO)进行有机地融合，对 K-means 算法进行了改进优化。通过这一算法可有效地避免初始聚类中心所带来的相关影响，此外还可基于 Hadoop 平台来实现 MapReduce 编程的相关目标。

目前在 Web 日志挖掘方面的研究多使用传统聚类方法，只能对 Web 日志进行静态处理，每次进行聚类都要对全体数据集进行操作，产生巨大的时间和空间消耗，并且做不到随时间变化得到簇的形状和位置的变化。对 Web 日志这类实时到达、不断递增的数据进行挖掘应该是一个连续的、在线的过程，本文采用数据流聚类的方法，对此类数据进行处理。

1.2.2 数据流聚类算法

数据流聚类算法主要是基于经典聚类算法而逐步地发展起来，聚类算法[22]为一种无监督算法，基本原理如下：考虑不同样本所具有的相似性来划分数据集，得到相应的簇。若类簇内部的数据对象有较高的相似性，不同的类簇间有较低的数据样本相似性，则聚类具备较好的聚类效果。对于数据流聚类算法而言，所需处理的对象为数据流。相较于传统的聚类算法，处理过程更加的简单快捷，此外还通过一次扫描来实现，因此对数据流所具有的演变性有着良好的适应性，可对簇的位置及形状改变进行有效的追踪[23]。

目前已经提出的数据流聚类算法有很多。最早的 STREAM 算法[24]由 O’Callaghan 团队所提出，它主要是基于划分的概念而发展起来。该算法通过批处理方式来有效地划分数据流，得到大小固定的一系列小块，再利用 K-means 算法聚类成 K 个类簇，然而这种批处理的处理方式实质上并不是真正的实时处理。所以而后 Aggarwal 及合作者提出了 CluStream 算法[25]，它是基于层次概念的。CluStream 算法的处理过程由在线微聚类和离线宏聚类两个阶段组成，前一个阶段获得一系列数据分布方面的概要性相关信息，也被称作微簇。后一个阶段通过在线阶段得到的概要信息来进行聚类，得到相应的聚类结果。此外，它也提出了新型的时间衰减结构（形式为金字塔形），对数据的存储过程进行优化。然而由于该算法采用 K-means 算法形成初始的微簇结构，因此它只可得到球形簇结果，

此外如果在数据集中含有高维数据以及在数据集之中存在有噪音的情况下，稳定性较低。

为解决 CluStream 所存在的缺陷，我国研究人员进一步改进了数据流聚类算法。Cao 及合作者以密度为基础发展了 DenStream 算法[26]，它同样使用二阶段框架的形式，然而相较于 CluStream，其中的离线阶段以 DBSCAN 算法作为基础，因此可进行各种不同形状的簇的生成。Chen 及合作者则是以密度网格结构作为基础发展了 D-Stream 算法[27]。将网格结构相关基本概念引入进来，在网格中映射数据流，在离线阶段，通过合并高密度网格单元来实现簇的生成，D-Stream 算法是目前聚类速度较快且聚类效果比较稳定的一种数据流聚类算法，然而由于需要以网格信息的方式存储数据记录，所以容易造成数据的丢失，尤其是处于聚类类簇边界的数据。

目前尽管国内外的专家们不断地对数据流聚类算法进行研究与改进，但是面对实时、海量的数据流，如何提高数据流聚类算法的聚类精度仍然是一项挑战。另外现有的对数据流聚类算法的研究都未从数据流聚类的分布性方面考虑，面对当今数据量爆炸性增长的情况，实现算法的分布式并行化计算是非常有必要的。

1.3 研究内容

本文主要研究在分布式环境下，通过数据流聚类算法来深入全面地分析 Web 访问日志，以 D-Stream 算法为基础，提出改进的密度网格数据流聚类算法，基于分布式实时大数据处理框架 Storm 对改进算法进行并行化设计，并且应用于实际采集的校园网站 Web 访问日志进行分析，挖掘网站访问行为，辅助网站的管理。

首先，通过查阅大量的现有的研究成果，对 Web 日志挖掘、数据流聚类算法、Storm 计算框架等方面进行理论准备。对传统的数据流聚类算法的特点进行分析，在密度阈值的确定和簇边界识别部分，进一步改进及完善 D-Stream 算法，使性能得以提升。通过对比实验，验证改进算法的聚类能力和精确度。

其次，为解决在非集群环境下不能有效应对大规模数据的问题，搭建 Storm 分布式框架，详细设计基于 Storm 的分布式并行化方案，并提出与平台相适应的分布式改进算法。通过轮廓系数对分布式算法的聚类效果进行验证，通过对比算法运行时间说明分布式算法的高效率性。

最后，基于上述研究内容，设计并实现基于 Storm 的 Web 访问日志分析模型，通过采集校园网站的 Web 访问日志真实数据进行实验，对聚类结果进行分析，得出最终的分析结论以望对校园网站的管理与建设提供一定的思路。

1.4 论文结构

针对提出的通过改进数据流聚类算法并分布式实现算法的方式分析 Web 访问日志的想法，当前的论文主要阐述了下述内容：

第一章，绪论。本章对 Web 访问日志分析目前主要的研究进展进行全面的介绍，进而分析目前国内外学者对这一领域的研究现状，再概括本文的拟开展工作内容，并介绍了本文的主体结构。

第二章，相关理论与技术。本章对论文相关领域的基本概念进行详细介绍。概述数据流模型以及数据流等主要概念，对数据流聚类算法进行深入研究。最后针对可以实现算法并行化的分布式流处理技术进行介绍，对 Storm 框架进行深入研究。

第三章，改进的密度网格数据流聚类算法研究。首先对 D-Stream 算法基本原理进行了全面的阐述，进而结合 D-Stream 算法的主要不足之处，提出改进算法。开展相应的对比实验，对改进算法的性能进行验证。

第四章，算法并行化设计及基于 Storm 的实现。本章将改进的数据流聚类算法和分布式并行化的思想相结合，对分布式并行化数据流聚类算法进行全面的设计，同时基于 Storm 来实现该算法。通过设计单机环境与分布式环境下的对比实验，验证分布式数据流聚类算法的处理效率。

第五章，基于 Storm 的改进算法在 Web 日志分析中的应用。本章对 Web 访问日志的格式和内容进行说明，建立基于 Storm 的 Web 访问日志分析模型，使用基于 Storm 的改进算法对采集的校园网站 Web 访问日志进行分析，并对分析结果结合实际情况进行分析，得出结论。

第六章，总结与展望。本章全面总结本文的工作内容和主要结论，同时指出未来的探索方向。

第二章 相关理论与技术

本章对进行Web访问日志分析的相关背景知识进行介绍，首先对Web挖掘、Web日志挖掘的相关技术进行概述，然后对数据流的相关定义、与数据流处理相关的技术和现有的典型的数据流聚类算法进行介绍，最后对分布式流处理技术进行介绍，对其中的Storm技术体系、Storm框架的组成结构和基本原理进行详细分析。

2.1Web日志挖掘

2.1.1Web挖掘概述

Web挖掘即将传统的数据挖掘技术应用于Web领域之上，借助传统的数据挖掘技术由相关于万维网的资源及行为里筛选有趣味的、有用的模式及隐含数据，涉及到Web技术及数据挖掘等诸多领域的知识，属于一项综合类技术[28]。此外Web挖掘还会用到数据库技术及信息检索技术等诸多相关领域的知识，众多研究者立足于自身领域，在不同的视角对Web挖掘的含义有着不同的理解及定义。依照Web数据的各个类型对Web挖掘进行了三类划分：内容挖掘、结构挖掘及日志挖掘（又将其称为使用挖掘）。

所谓Web内容挖掘即挖掘Web页面中的内容亦或挖掘底层的数据库数据,立足于其文档的内容信息获取相关知识[29]。根据研究内容不同，能够分为文本数据及音频视频数据。Web内容挖掘里较为关键的技术领域即Web文本挖掘，其同样为具备最广泛应用的技术，几乎所有存在的网页信息都可以利用文本挖掘的技术来处理。

Web结构挖掘主要是对Web各页面间的关联进行挖掘[30]，借助分析诸多页面间的关联，比如页面内部链接及页面被链接的数量级，获悉Web局部构造及Web整体构造的特征，发掘其隐藏的结果信息。

Web日志挖掘又被叫做Web使用挖掘[31]，是本文主要关注的研究内容。以Web日志为研究对象，Web日志里记录了用户与网站进行交互的信息，比如用户的IP地址、用户访问的页面上的内容或者用户使用的访问工具等，通过对这些信息的分析，能够全面的掌握访问网站的用户的情况，也可以把握网站的运营情况。

2.1.2Web 日志挖掘概述

Web 挖掘领域最为关键的部分之一为 Web 日志挖掘。Web 日志挖掘可借助强大的数据挖掘算法，系统的挖掘 Web 服务器里的日志，进而获悉用户的访问模式亦或偏爱兴趣，接着依照发现的用户模式对用户的使用偏好及行为模式等等进行一定程度上的分析[32]，整调网站拓扑构造、强化系统性能、对网站的交互体验进行改进、给予智能化的个性服务，使用户的粘性获得强化。

网络中的通信一般是通过 Web 客户端-代理服务器-Web 服务器的结构实现的[33]。每个节点的服务器都会产生日志，Web 客户端会产生该客户端用户访问多个网站的日志文件，代理服务器会产生使用该代理服务器的多个用户访问多个网站的日志文件，Web 服务器会产生多个访问该网站的用户的日志文件。所以，在通过日志对用户访问行为进行挖掘时，也可以从 Web 客户端、代理服务器端及 Web 服务器端三个方向进行挖掘。由于三个节点的日志数据不同，所以能够挖掘出的用户信息也不同。

现下 Web 日志领域算法的研究焦点主要有两个方面：序列模式挖掘算法及聚类算法。挖掘客户端及代理服务器端的访问模式需要从整个 Web 出发，一般选择序列模式挖掘算法，可以用来研究用户的偏好的访问路径，检索用户感兴趣的内容，完善智能搜索引擎功能等。使用聚类算法挖掘 Web 日志一般是在服务器端，对访问该网站的全部用户产生的日志数据进行聚类，通过聚类结果对用户访问行为进行分析。本文最终对校园网站日志进行分析，对访问用户的访问行为进行认定，故选择使用聚类算法挖掘服务器端的访问模式。

2.1.3Web 日志挖掘过程

通常情况下，将 Web 日志挖掘过程划分成日志采集、日志预处理、模式发现、模式分析四个阶段[34]。

（1）日志采集阶段

Web 日志原始数据一般来自于 Web 服务器日志目录，本文也是采用这种日志展开分析工作，也有通过 Web 页面 JavaScript 脚本收集日志的方式。

（2）日志预处理阶段

按照数据挖掘的一般步骤，这个阶段对原始数据执行预处理。原始日志数据包含不规则、不干净、无效的数据，原始的日志为数据预处理的核心操作对象，向适合数据挖掘的格式进行转换，依据统一的格式存储，然后继续分析。

（3）模式发现阶段

这一阶段根据不同的业务需求采用不同的挖掘算法，主要涵盖统计分析、分

类分析、聚类分析、关联规则、路径分析及序列数据挖掘[35]等，对上一阶段的数据进行挖掘，发现有价值的信息。

（4）模式分析阶段

这一阶段就是数据挖掘的解释及评价过程，将得到的各种模式转化成易于理解的知识，为制定更好的策略提供理论依据。

2.2 数据流聚类算法

2.2.1 数据流

数据流 $X=(x_1, x_2, \cdots, x_t)$是一个包含一系列按时间到达的数据点的集合，其中任一数据点 x_i 均为一个 d 维数据记录，集合中有标记数据到达时间的时间戳 t（Time Stamp）。下面为数据流的几个特点[36]：

（1）数据流一般是实时到达的；

（2）一般不能自定义处理数据流的顺序；

（3）数据流的数据量一般是巨大且无法预知的；

（4）一般较难对全部数据流进行完整的存储。

2.2.2 数据流模型

数据流模型存在形式较为繁多，立足于数据流模型自身特性的视角，能够由下述两个方面来类别划分数据流模型[37]。

（1）根据数据点表达数据流的方式进行分类[38]。

a）时序模型：数据流里的各个数据点于此模型里均为一个独立个体信号；

b）现金登记模型：数据流里的各个数据点于此模型里某种程度的增量的表达数据流；

c） 十字转门模型：数据流里的各个数据点于此模型里变量的表达数据流。

（2）根据对时序的界定范围进行分类[39]。

a）快照模型：该模型选择两个事先定义的时间戳来对数据的处理范围进行定义；

b）界标模型：该模型选择某个已知时间与目前时间两个时间点对数据的处理范围进行定义；

c）滑动窗口模型：该模型选择一个终点为目前时间的大小固定的窗口对数据的处理范围进行定义。

2.2.3 数据流挖掘

因数据流自身的特性，诸多领域开始着手进行有关数据流实时处理的工作，但是现下数据流挖掘的技术研究面对着一些难题。数据流挖掘即基于数据流获悉提取数据流内含信息及知识的过程[40]。

数据流挖掘的过程应该与数据流模型的特点相适应，依照数据流模型的特性，数据流处理技术有单次查询、时间复杂度低及空间复杂度低三方面的要求[41]。Gaber M M 等人[42]对数据流挖掘的核心技术进行了划分：一类是数据驱动（Data-based）的技术，一类是任务驱动（Task-based）的技术。

（1）数据驱动的技术

在数据挖掘方面，对越是大数据量的进行数据挖掘越能发掘出更全面细致的问题，然而虑及进行数据挖掘的系统内存的容量及面对大数据量时的处理速度，一般难以完整存储和处理所有的数据，可以借助保存核心数据结构的方式来存储数据。保存核心数据结构的方法主要有大纲数据结构、梗概及抽样等[43]。

a）大纲数据结构：大纲数据框架借助概要技术，如频率矩、小波分析等，得到当前数据流的概要信息。

b）梗概：梗概借助随机投影的技术，将数据流中的数据投影汇总到一个低维度的空间。

c）抽样：抽样借助水库抽样的技术，从“水库”中抽取部分样本进行处理。

（2）任务驱动的技术

任务驱动技术从算法的应用方面进行研究，目前有滑动窗口、衰减函数及倾斜时间框架等技术[44]。

a）滑动窗口：滑动窗口是基于“距离当前时间愈近的数据愈加能够影响到整体数据”的思想，对历史数据概要的存储，保留少量近期数据进行细节分析。

b）衰减函数：衰减函数设定衰减因子随时间的增长而减少，将衰减因子引入到数据的分析中，计算数据点对当前结果的影响。

c）倾斜时间框架：滑动窗口及衰减函数仅能操作同一时间维的窗口。倾斜时间框架可以在多个时间维的窗口展开挖掘与分析，又称为多窗口技术。此框架选择细粒度窗口来存放近期数据，选择粗粒度窗口来存储较为久远的数据。

孙玉芬等人[45]也对除上述已总结的两类技术外的其他有关数据流挖掘技术进行了归纳，如以算法为基础的自适应技术及近似技术等。以上提出的完善和改进数据流挖掘的技术都在一定意义上解决了数据流挖掘方面的一些问题，一定程度上提高了算法的处理效率和准确性，本文对数据流进行挖掘的过程中也采用了相关的技术和概念。

2.2.4 数据流聚类算法

STREAM 算法及 D-Stream 算法等有关算法均属于数据流聚类里的典型算法，从这些经典算法里筛选出了相关于本文算法的部分进行了简介。

（1）CluStream 算法[46]。此算法是最早提出使用在线和离线两个阶段这样的框架来对数据流进行聚类的算法，此后大部分的数据流聚类算法都继续使用了这个处理框架。在 CluStream 算法在线阶段，首先使用 K-means 算法先对部分数据进行聚类，得到 K 个初始微簇，然后对于新到达的数据，计算数据与各微簇中心点的距离，通过比较距离将新到达数据分配到相应的微簇。离线阶段根据在线阶段产生的各微簇中心点信息，使用 K-means 算法进行聚类。由于 CluStream 算法基于 K-means 算法进行聚类，需要提前设置类簇数 K，并且通过计算距离来衡量各数据点之间的相似程度，只能产生球形的类簇。

（2）DenStream 算法[47]。此算法主要是对 CluStream 算法的不足之处进行改进，引入核心微簇、潜在核心微簇和离群微簇的概念优化 CluStream 算法中对微簇的定义，引入时间衰减函数定义数据点的权重，使数据点的权重随时间增加而减少。同样使用在线和离线两个阶段对数据流进行聚类，在线阶段对各微簇的状态进行动态调整，通过基于密度的 DBSCAN 算法对潜在核心微簇进行聚类。离线阶段对各微簇进行合并，完成聚类。由于 DenStream 算法是基于密度进行聚类的，所以可以得到任意形状类簇的聚类结果。

（3）D-Stream 算法[48]。此算法在 DenStream 算法基于密度进行聚类的思想之上引入网格结构的概念，将数据空间按照一定的规则划分成网格，在网格中放置数据记录，对网格进行聚类。D-Stream 算法同样使用在线和离线两个阶段进行处理。在线阶段，将数据流按照时间顺序映射到网格当中，根据网格中数据记录数量的多少将网格分为密集网格、过渡网格和稀疏网格。离线阶段删除网格内数据记录极少并没有可能变为密集网格的网格，使用其余的过渡网格和密集网格进行聚类。

2.3 分布式流处理技术

2.3.1 分布式流处理技术

流处理属于核心大数据处理手段之一，此手段的关键特点是处理主体为流数据，流数据实时更新且具备庞大的数据量。分布式流处理属于一类围绕动态数据展开的细粒度处理方法，以分布式的存储当作基础，对不断产生的动态数据进行处理[49]。在数据处理方面有快速，高效，低延迟等特性，逐渐在大数据处理中发

挥越来越重要的作用。

通过分布式并行化方式对流数据进行聚类分析时，一般是把数据记录分配至各个局部节点上，借助各节点本地聚类算法展开初步的聚类处理，局部聚类就这样被完成，中间的临时结果形成后，对临时结果进行收集与合并操作，并借助全局的聚类算法对合并的结果展开二次聚类，形成最终的聚类结果，这样就分布式的对流数据进行了聚类分析。

现下具备较宽泛应用的分布式流处理平台有 Storm、Spark Streaming、Samza、Flink 等。各平台都有着其各自的特点。

（1）Storm

Storm 是一款由 Back Type 公司开发的分布式流数据平台，并且在 2011 年被正式开源，使用主从系统架构，主节点和工作节点同时存在于 Storm 集群，Zookeeper 在 Storm 集群当中负责各节点之间的协调安排工作[50]。主节点操作 Nimbus 进程，于集群中进行代码发放，借助 Zookeeper 协调任务的分配并对集群进行监测；工作节点操作 Supervisor 进程，执行工作任务。依靠任务拓扑（topology）进行具体操作任务的完成，拓扑中有 Spout 组件和 Bolt 组件， Spout 组件负责接收数据，Bolt 组件负责处理数据。

（2）Spark Streaming

Spark 是一款由 UC Berkeley AMP 实验室开发的分布式处理框架。其中的 Spark Streaming 模块用来进行流计算。Spark Streaming 跟 Spark 类似均采用弹性分布式数据集（RDD）的核心处理机制[51]。Spark Streaming 引入微批次的概念处理流数据，对数据流按照秒级的时间间隔进行划分，将完整的数据流划分为许多的时间块数据，接着借助批处理的方法对这些时间块数据进行处理。但是由于 RDD 转换仅具备有限的操作能力，此外使用微批次处理的方式处理数据流会造成延迟，因此 Spark Streaming 在流数据处理方面的能力有限。

（3）Samza

Samza 是一款 2013 年于 Linked In 公司被开源的流处理平台。Samza 将单一消息视作数据流处理的单位[52]。Samza 将完整的数据流分解成一个或多个分区，每个分区由一组只读消息的有序数列组成的小数据流组成，每条小数据流均有一个自身的 ID。此系统同样能进行批处理，简单讲就是对同一个数据流分区的多个小数据流进行逐次处理。Samza 一般依靠 Kafka 和 Yarn 进行数据的传输，其中 Kafka 进行数据发送，Yarn 对资源进行分配，对各类任务进行调度。

（4）Flink

Flink 是一款由 Apache 软件基金会开发的开源流处理架构。Flink 结合流处理和批处理技术进行流数据的分布式计算，既可以对无界的实时流数据进行处理，

还可以将批数据作为流数据的极限特例，对有界的批数据进行处理[53]。流计算优先原则为 Flink 流数据处理的基石，按条进行流数据处理，具有低延迟和高吞吐率的特点。

上述四类分布式流处理平台于诸多领域中获取了宽泛应用。本文选择使用 Storm 平台分布式处理 Web 访问日志的工具，因为 Storm 的语言无关性，Storm 中的拓扑结构和各种组件支持使用多种编程语言进行定义，编码灵活性较高；其次，Storm 进行流数据的处理能够达到毫秒级的响应，其他平台微批处理的方式具有一定的时间延迟；此外，相较于其他工具，Storm 的项目成熟度更高，目前被使用的范围也更广泛，系统容错性更好，也更健壮，数据丢失的情况较少。综合上述几个方面的考虑，本文采用 Storm 平台进行分布式处理。

2.3.2 分布式流处理平台 Storm

作为一类处理流式数据的框架，Storm 具备分布式、可容错等特性[54]。Storm 对数据流进行处理时，会将处理任务分配给各个组件，各组件针对性的负责自身的处理任务，各组件之间互不干扰。

Storm 选用如图 2.1 所示的主从系统架构， Nimbus 守护进程为主节点，将代码发放至集群内，将处理任务指派至每个工作结点。Supervisor 守护进程为工作节点，执行主节点指派的任务[55]。Zookeeper 进行 Supervisor 与 Nimbus 之间的协调服务。

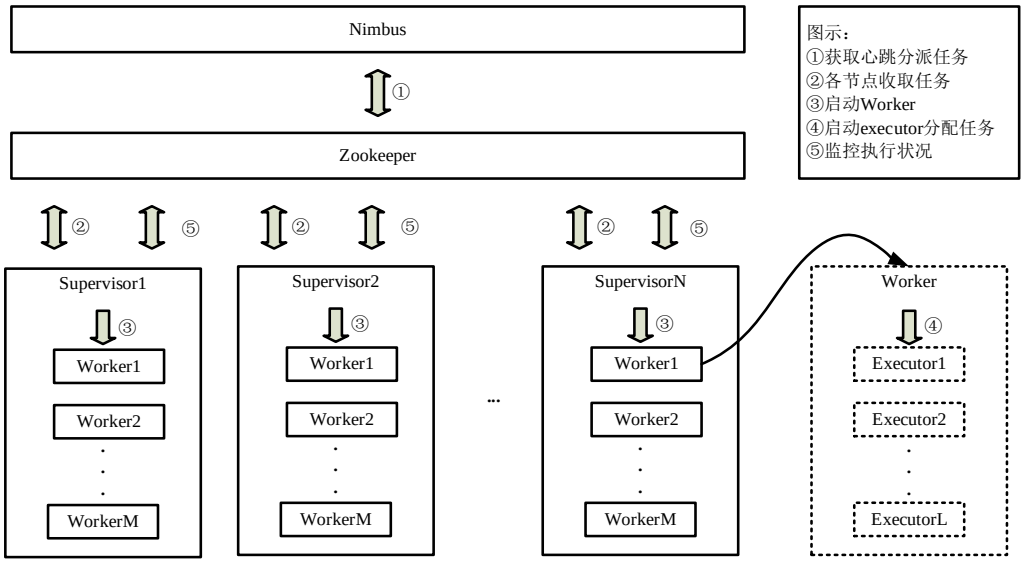


图 2.1 Storm 集群架构与执行流程

客户端将负责计算任务的任务拓扑提交至主节点 Nimbus，Nimbus 对所获的计算任务开展针对性的调度，指派至各工作节点 Supervisor，此外 Nimbus 会借助 Zookeeper 对指派至各 Supervisor 上的计算任务的完成情况进行实时监控，假

设某个 Supervisor 有问题产生，Nimbus 会检测故障，对该 Supervisor 进行重启，并对资源和计算任务展开新的分配。

工作节点 Supervisor 负责 Storm 系统中的分布式处理的任务，各个 Supervisor 之间不具备关联性，由主节点 Nimbus 对 Supervisor 开展调度管理，确保其对 Nimbus 分配任务的监管成效，Nimbus 任务指派完成后，各结点会对自身的工作进程 Worker 进行启动，每个 Worker 内有多个 Executor，每个 Executor 与一个 Task 对应，每个 Task 与任务拓扑中的某个 Spout 或 Bolt 线程一一对应。

Zookeeper 是一款分布式的应用程序协调服务系统，在 Storm 的主节点 Nimbus 和工作节点 Supervisor 之间进行协调服务，其能够对指派到各 Supervisor 的工作信息外加各 Supervisor 自身任务执行现状进行监控，同时向主节点 Nimbus 进行反馈，确保其对于 Storm 系统整体监控的高效性；倘若 Supervisor 亦或 Worker 遭受故障不再工作时，其同样会对有关故障部件的情况进行监控，并反馈至 Nimbus，以保证故障检测的效率。

Storm 系统 Tuple（元组）的形式进行信息的传递，大量的、不断产生的 Tuple 形成流 Stream，Stream 以不同的分组方式在各节点之间传送，分组方式包括 shuffle 分组（于不同线程里平均划分元组）、field 分组（于各线程里根据关键字划分元组）、all 分组（于全部线程里传送全部元组）。后文在将 Storm 平台当作基础的算法实现中，对 shuffle 分组及 all 分组这两种分组方式进行了使用。

Storm 中最为核心的特性即其有向无环的拓扑结构，Storm 使用任务拓扑结构（topology）作为处理任务的逻辑单元，下图 2.2 为 Storm 的拓扑结构，在 Storm 对数据进行实时处理的过程中，将计算任务以 topology 的形式进行发布，topology 的构成组件主要有：Spout 组件和 Bolt 组件[56]。假设用户希望借助 Storm 开展流式计算，首先要对计算任务进行分解，分解成接收数据部分及执行数据部分，用户需要对每个 Spout 组件及 Bolt 组件的详细功能进行分配，Spout 组件负责对任务中使用的数据进行接收，从外部数据源中进行数据的拉取，并将接收到的数据打包成数据流传送到 Bolt 组件，Bolt 组件负责对接收到的数据进行处理，Bolt 组件能够实现用户实现定义的一系列的具体的操作任务。Spout 组件与 Bolt 组件之间通过传送数据流的方式进行联系[57]，诸多 Spout 组件与 Bolt 组件级联成一个有向无环图的拓扑结构，任务拓扑在 Storm 系统中发布并执行的过程当中，如果没有被人为地终止，就会在 Storm 启动后一直运行下去。

Spout 组件与 Bolt 组件之间、不同的 Bolt 组件与 Bolt 组件之间的信息传递是通过 Tuple（元组）作为信息载体而实现的。Tuple 作为数据流中的一个基本的处理单元，在 topology 里的每一个节点中，都要设置需要发送的 Tuple 的内容。Storm 会追踪 Tuple 的树形结构是否被成功的建立，并且监控树形结构中每一个

节点的 Tuple 是否被成功的处理。

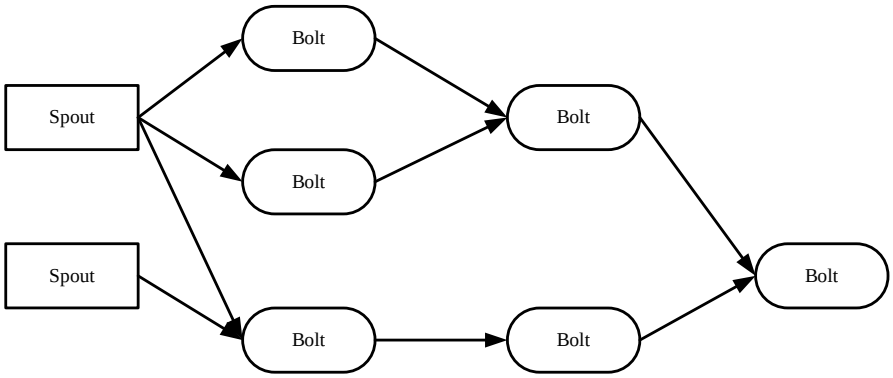


图 2.2 Storm 拓扑结构

2.4 本章总结

本章介绍了跟本文研究内容有关的理论及技术。先是围绕 Web 日志挖掘的诸多概念开展了介绍，接着对数据流、数据流模型及数据流挖掘的有关概念特征展开了分析，介绍了一些经典的数据流聚类算法。最后介绍了数据流分布式计算的关键技术分布式流计算平台，对其中的 Storm 技术体系、Storm 框架的基本组成进行详细论述。

第三章 改进的密度网格数据流聚类算法研究

基于上一章节对本文研究领域相关理论与技术的分析，本章重点研究数据流聚类算法，先是对以密度网格作基础的数据流聚类算法 D-Stream 的核心定义及相关概念进行介绍，接着总结 D-Stream 算法的缺陷，围绕 D-Stream 算法的不足，对密度网格数据流聚类算法进行改进设计，接着进行对比实验对改进算法及 D-Stream 算法进行比较。

3.1 基于密度网格的数据流聚类算法 D-Stream

3.1.1D-Stream 算法概述

D-Stream 算法于 2007 年被 Chen Yixin 及 Tu Li 二人提出，该算法以密度网格作基础展开数据流聚类，引入密度网格的概念对数据流进行聚类，避免了以距离作基础的传统聚类算法仅可生成球形类簇的不足，同时规避了以密度作基础的聚类算法对全部数据记录进行处理，处理效率低下的缺点。D-Stream 算法将对数据的处理转化为对网格的处理，通过划分数据空间对数据进行压缩表示。既可以识别任意形状类簇又可以高效率处理流数据。

D-Stream 算法首先将数据空间按照一定的规则划分成互不相交的网格，于在线部分读取连续到达的数据记录，接着将读取到的每个数据记录按照定义映射至划分好的网格里，根据网格内的数据信息对网格的特征向量进行更新，于离线部分对各个网格的密度进行计算，依照网格的密度对网格进行三类划分：稀疏网格、密集网格及过渡网格，按照网格密度划分和一定的规则对网格进行聚类，设置时间周期 *gap*，在每个时间周期 *gap* 内动态调整聚类，在第一个时间周期 *gap* 之后，算法将生成初始聚类，然后算法根据时间周期定期检查并删除零星网格以调节聚类。并且借助密度衰减技术动态计算数据记录权重，从而动态计算网格密度，不断的获取和调整类簇。图 3.1 即为 D-Stream 算法的整体流程图。

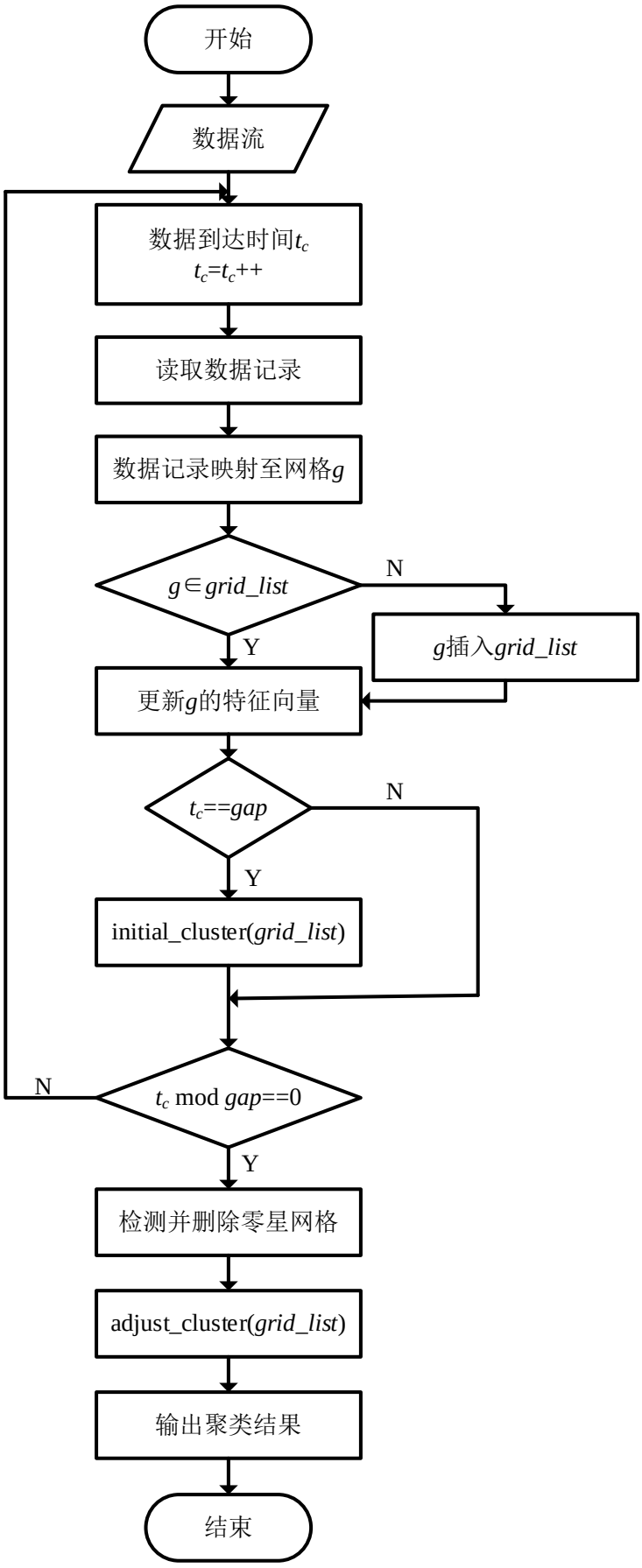


图 3.1 D-Stream 算法流程图

3.1.2D-Stream 算法基本定义

假设输入的数据记录为$\mathrm{x}=(x_1,x_2,\cdots,x_d)$，共有 d 维，在数据空间中定义所有输入的数据记录，数据空间的定义如公式 3.1 所示，S_i 代表第 i 维数据空间。

$$\mathrm{S}=S_1\times S_2\times\cdots\times S_d \tag{3.1}$$

D-Stream 算法对 d 维的数据空间 S 进行了均匀划分，将数据空间划分为多个网格。假设把各维数据空间$S_i,i=1,\ldots,d$划分成 p_i 个部分，那么划分后的数据空间可以表示为公式 3.2 所示。

$$S_i=S_{i,1}\cup S_{i,2}\cup\cdots\cup S_{i,p_i}\ \ (1\le i\le d) \tag{3.2}$$

这样数据空间 S 就被划分成了$\mathrm{N}(\mathrm{N}=\prod_{i=1}^{d}p_i)$个网格。公式 3.3 为各网格 g 的表示方式。

$$\mathrm{g}=S_{1,j_1\times}S_{2,j_2}\times\cdots\times S_{d,j_d}\quad j_i=1,2,\cdots,p_i \tag{3.3}$$

当数据记录映射到网格 g 中时形成密度网格 $g(x)$中可以表示为公式 3.4 所示。

$$\mathrm{g(x)}=(j_1,j_2,\cdots,j_d)\quad where\ \ x_i\in S_{i,j_i} \tag{3.4}$$

对于每一个数据记录 x 引入衰减因子$\lambda\in(0,1)$，衰减因子随 x 到达时间的增长而减少。假设数据记录 x 在t_c时刻到达，那么定义数据记录 x 的时间戳：$T(x)=t_c$，这样 x 在时刻 t 的密度系数$D(x,t)$的计算方式如公式 3.5 所示。

$$\mathrm{D(x,t)}=\lambda^{t-T(x)}=\lambda^{t-t_c} \tag{3.5}$$

定义 1 网格密度。对于每个密度网格 $g(x)$，在 t 时刻的网格密度$D(g_{(x)},t)$定义为 t 时刻之前密度网格 $g(x)$内所有数据记录 x 的密度系数$D(x,t)$的和，计算方式如公式 3.6 所示。其中$E(g_{(x)},t)$代表在 t 时刻，密度网格 $g(x)$内全部数据记录的集合。

$$D\left(g_{(x)},t\right)=\sum_{x\in E(g_{(x)},t)}D(x,t) \tag{3.6}$$

假设密度网格 $g(x)$上一次在 t_l 时刻接收到一个数据记录，在 $t_n(t_n>t_l)$时刻接收到一个新的数据记录，经计算，密度网格 $g(x)$在 t_n 时刻的网格密度可以表示为公式 3.7 所示。

$$\mathrm{D}\left(g_{(x)},t_n\right)=\lambda^{t_n-t_1}D\left(g_{(x)},t_1\right)+1 \tag{3.7}$$

定义 2 密度网格性质。由上述定义可以证明，从 0 到 t 时刻，对于所有到达的数据记录集合 $\mathrm{X}(t)$，有$\sum_{x\in X(t)}D(x,t)\le\frac{1}{1-\lambda}$，$for\ \ any\ \ t=1,2,\ldots$，并且$\lim_{t\to\infty}\sum_{x\in X(t)}D(x,t)=\frac{1}{1-\lambda}$。所以每个密度网格的平均密度接近于$\frac{1}{N(1-\lambda)}$。定义网格密度阈值 D_l和 D_m，令$D_l=\frac{C_l}{N(1-\lambda)}$，$D_m=\frac{C_m}{N(1-\lambda)}$。其中，$C_l(0<C_l<1)$和$C_m(C_m>1)$是控制阈值的参数，需要预先设置。

根据网格密度阈值 D_l和 D_m，对密度网格进行三类划分：密集网格、过渡网格及稀疏网格。在时刻 t，密度网格 g 的网格密度$\mathrm{D}(g_{(x)},\mathrm{t})>D_m$时为密集网格，密度网格 g 的网格密度$D_l\le\mathrm{D}(g_{(x)},\mathrm{t})\le D_m$时为过渡网格，密度网格 $g(x)$的网格密度$\mathrm{D}(g_{(x)},\mathrm{t})<D_l$时为稀疏网格。

定义 3 邻居网格。对于两个密度网格 $g_1=（j_1^1,j_2^1,\cdots,j_d^1）$ 和 $g_2=（j_1^2,j_2^2,\cdots,j_d^2）$，如果存在 $k\in[1,d]$，使得 k 满足以下两个条件：

（1） $j_i^1=j_i^2,\quad i=1,2,\cdots,k-1,k+1,\cdots,d$

（2） $|j_k^1-j_k^2|=1$

那么在第 k 维空间，g_1和g_2就是邻居网格。表示为：$g_1\sim g_2$。密度网格 g 的邻居网格记为 $\mathrm{NB}(g)$。

定义 4 网格组。网格组是一个密度网格的集合$\mathrm{G}=(g_1,g_2,\cdots,g_m)$，如果集合 G 内任意两个密度网格$g_i,g_j\in G$，存在一系列密度网格 $g_{k_1},g_{k_2},\cdots,g_{k_l}$使得$g_{k_1}=g_i$,$g_{k_l}=g_j$并且$g_{k_1}\sim g_{k_2}$，$g_{k_2}\sim g_{k_3}$,…，$g_{k_{l-1}}\sim g_{k_l}$，那么 G 就是一个网格组。

定义 5 内部网格和外部网格。在一个网格组 G 当中，对于所有密度网格 g，$\mathrm{g}=(j_1,j_2,\cdots,j_d)(\mathrm{g}\in\mathrm{G})$，在每一个维度都存在邻居网格，那么 g 就是 G 中的一个内部网格，否则 g 就是 G 中的一个外部网格。

定义 6 网格簇。如果网格组 G 的每一个内部网格都是密集网格，每一个外部网格都是一个密集网格或过渡网格，那么 G 就是一个网格簇。

定义 7 特征向量。以五元组的形式定义特征向量，以特征向量的方式概要的存储密度网格的数据结构。定义密度网格 g 的特征向量$(t_g,t_m,D,label,status)$，其中的 t_g 代表密度网格 g 最近一次更新的时间，如果判断密度网格 g 为零星网格，那么 t_m 代表密度网格 g 从网格列表 *grid_list* 中删除的时间，D 代表密度网格 g 最近一次更新后的网格密度，*label* 代表密度网格 g 所属的聚类簇的簇号，*status*

代表删除零星网格操作中的一个标签，*status* 有 SPORADIC 和 NORMAL 两种状态。

3.1.3 时间周期 gap 与网格检查

（1）时间周期 *gap* 值的确定

由于衰减因子的引入，随着时间的改变，网格密度也是不断变化的，密集网格长时间不接收新数据记录，就会退化成过渡网格亦或是稀疏网格，稀疏网格获取新的数据记录后，其能够向过渡网格亦或密集网格进行升级。所以聚类结果也应该相应的改变。定义密度网格从密集网格向着稀疏网格进行降级，亦或从稀疏网格向着密集网格进行升级所用的最少时间为时间周期*gap*，时间周期如果太大，将无法充分识别数据流的动态变化，时间周期如果太小，则会增加工作量，减慢处理速度。时间周期 *gap* 的计算如公式 3.8 所示。

$$\text{gap} = \left\lfloor \log_{\lambda}（\max\{\frac{C_l}{C_m}, \frac{N-C_m}{N-C_l}\}） \right\rfloor \tag{3.8}$$

（2）零星网格的检查和删除

基于密度网格的数据流聚类面临一个严峻的挑战，就是网格数量庞大，如果将所有的网格都存储起来将占用大量的空间，因此算法只存储含有数据的密度网格。但是根据观察，数据空间中大部分的密度网格内都没有数据记录亦或获取的数据记录较少，定义这样的密度网格为零星网格，为了对算法时效性及聚类精度进行提升，在数据记录的处理过程中必须不断地检查和删除零星网格，进而使存储空间得到节省，计算速率得到提高。

网格密度低于稀疏网格密度阈值的密度网格均可能为零星网格，零星网格一般有两种，一种是密度网格本身接收到的数据记录少，第二种是长时间未接收新的数据记录，从而导致网格密度随时间衰减，第一种是需要删除的对象。算法通过定义密度阈值来区分稀疏网格和零星网格。

定义 8 零星网格密度阈值。零星网格密度阈值定义为公式 3.9 所示，t_g 是该密度网格最后一次更新的时间，t 是当前时刻，其中 $t>t_g$。当稀疏网格的网格密度 $D(g_{(x)},t) < \pi(t_g,t)$时，将该稀疏网格判定成零星网格，并且在下一次周期性检查时删除该密度网格。

$$\pi(t_g,t) = \frac{C_l}{N}\sum_{i=0}^{t-t_g}\lambda^i = \frac{C_l(1-\lambda^{t-t_g+1})}{N(1-\lambda)} = D_l(1-\lambda^{t-t_g+1}) \tag{3.9}$$

3.1.4 算法描述

本节描述用于生成初始化聚类的算法 initial_cluster(*grid_list*)和用于周期调整簇结构的算法 adjust_cluster(*grid_list*)。生成初始化聚类的过程如算法 1 所示，首先更新网格列表中所有非空密度网格的密度，后续的更新密度网格、搜索邻居网格等操作都是对网格列表进行维护的过程，这样的操作可以在不损失聚类质量的前提下提高处理速度。

算法 1 生成初始化聚类算法 initial_cluster(*grid_list*)

输入：所有密度网格

输出：初始网格簇集合

步骤：

1. 更新 *grid_list* 中密度网格的密度；
2. 对密集网格进行聚类；
3. 其他密度网格 *label* 设为 NO_CLASS；
4. **repeat**
5. **for each** 类簇 *c*
6. **for each** *c* 中的外部网格 *g*
7. **for each** *g* 的邻居网格 *h*
8. **if**（$h \in c'$）
9. **if** （$|c| > |c'|$）c'中密度网格 lab*el* 设为 *c*；
10. **else** *c* 中密度网格 *label* 设为c'；
11. **else if**（*h* 是过渡网格）*h* 的 *label* 设为 *c*；
12. **until** 类簇 *label* 不再发生变化

周期调整簇结构算法具体描述如算法 2 所示。簇结构的调整只对上一次调整以来网格性质发生变化的密度网格进行操作，对簇结构的调整可以反映数据流的演化过程。

算法 2 周期调整簇结构算法 adjust_cluster(*grid_list*)

输入：所有密度网格

输出：网格簇集合

步骤：

1. 更新 *grid_list* 中密度网格的密度；
2. **for each** 经过时间周期 *gap* 后状态发生变化的密度网格 g
3. **if**（g 变为稀疏网格）
4. g 从所属簇 c 中删除，*label* 设为 NO_CLASS；
5. **if**（若簇 c 不连通）将 c 分为两个簇；
6. **else if**（g 变为密集网格）
7. 在密度网格 g 的邻居网格中寻找密度网格 h，其所属簇 c_h 的规模在 g 的所有邻居网格中最大；
8. **if**（h 是密集网格）
9. **if**（g 的 *label* 为 NO_CLASS）将 g 的 *label* 标记为 c_h；
10. **else if**（$g \in c\ and\ |c| > |c_h|$）簇 c 合并簇 c_h；
11. **else if**（$g \in c\ and\ |c| \leq |c_h|$）簇 c_h 合并簇 c；
12. **else if**（h 是过渡网格）
13. **if**（g 的 *label* 为 NO_CLASS）**and**（若 g 加入簇 c_h，h 变为外部网格）将 g 的 *label* 标记为 c_h；
14. **else if**（$g \in c\ and\ |c| > |c_h|$）$h$ 的 *label* 标记为 c；
15. **else if**（g 变为过渡网格）
16. 在 g 的所有邻居网格中查找到能满足 g 加入该簇可以成为外部网格的最大的簇c'；
17. **end for**

3.1.5D-Stream 算法的不足

D-Stream 算法是以密度网格作基础展开聚类的数据流聚类算法，一般情况下，D-Stream 算法会根据一定规则将数据空间划分成若干个网格，然后接收持续到达的数据记录，将数据记录映射到各个网格，接着计算各密度网格的密度，最后基于网格的密度进行聚类。网格结构的使用减少了内存的消耗，算法仅需要对密度网格进行操作，但是网格结构的缺陷也在算法中有所体现，主要包括以下两个方面：

（1）D-Stream 算法进行聚类之前需要通过人工设定的密度阈值参数 C_l 及

C_m 来对网格密度阈值 D_l 及 D_m 进行计算，网格密度阈值能够对密度网格的网格性质进行区别，在手动设置密度阈值参数的过程中如果缺乏相关的阈值设置知识会致使的阈值参数设置不当，从而影响密度阈值对网格性质的判断能力，降低算法的聚类质量。

（2）D-Stream 算法是以密度网格作为聚类基础的，基于网格的数据存储方式实质上是对数据进行压缩存储，使用特征向量表示一个密度网格的方式是一种粗分辨率的表示方式。这种方法会造成数据记录的丢失，尤其是处于簇边界的密度网格中的数据记录，聚类结果的簇边界缺失现象严重。

3.2 改进算法设计

3.2.1 改进算法基本思想

本节围绕 3.1.5 节中总结的以密度网格作基础的数据流聚类算法 D-Stream 的不足之处，研究相应的优化策略，提出改进的密度网格数据流聚类算法（下文中称之为改进算法），改进算法仍采用在线部分和离线部分相结合的处理方式。

在线部分对到达的数据记录进行处理，将到达的数据记录映射到划分好的网格当中，对存在数据记录的密度网格的特征向量进行更新，在此部分，改进算法借助各密度网格数据量、网格密度以及存在的网格数量来确定网格的密度阈值 D_m、D_l 和定期检查的时间周期 gap，接着生成初始的簇结构同时基于此依据周期对零星网格进行检查删除。

离线部分每隔一个时间周期 gap，调整一次簇结构，将网格引力概念引进簇边界判定方法，对稀疏网格根据密度网格间的引力进行聚类，将稀疏网格归并到相应的簇中，输出最终的聚类结果，在一定程度上避免簇边界数据的误删，提高聚类精度。

3.2.2 改进算法基本定义及相关概念

改进算法大部分基础定义与 D-Stream 算法相同，本节将介绍改进算法区别于 D-Stream 算法的定义与概念。改进算法相关定义如下：

定义 8 网格引力。假设网格 g 在第 i 维的宽度为 $2r_i$，网格的中心点为 c_i。对于映射到网格 g 中的数据记录 x，构造 d 维的，以 x 为中心，宽度为 $2e_i$，$0 \leq e_i \leq r_i$的立方体 $Cube$(x)。定义 V(x,h)为 $Cube$(x)与网格 h 的相交体积，$Cube$(x)与网格 h 的初始网格引力$attr_{ini}(x,h)$定义为 V(x,h)与 $Cube$(x)体积的商，并且经过计算得出如公式 3.10 所示计算结果，将其定义为数据记录 x 与密度网格 h 之间

的引力。当 *Cube*(x)完全处于网格 g 中，且网格 h 与网格 g 对应第 i 维中的相同分区时，设置 $b_i(x,h)=1$，当网格 h 与网格 g 对应第 i 维中的不同分区时，设置 $b_i(x,h)=0$。

$$attr_{ini}(x,h)=\frac{V(x,h)}{\prod_{i=1}^{d}2e_i}=b_1(x,h)b_2(x,h)\dots b_i(x,h) \tag{3.10}$$

对于数据记录 x，假设 x 在 t_c 时刻到达，那么数据记录 x 在当前时刻 t 与密度网格 h 之间的引力定义为公式 3.11 所示。

$$attr(x,h,t)=\lambda^{t-t_c}attr_{ini}(x,h) \tag{3.11}$$

若密度网格 g 和 h 是邻居网格，那么密度网格 g 在 t 时刻对 h 的网格引力的定义如公式 3.12 所示。其中 $E(g_{(x)},t)$代表密度网格 g 在 t 时刻之内所映射的所有数据记录的集合。网格引力是非对称的，也就是说网格 g 对 h 的引力不等于网格 h 对 g 的引力。

$$attr(g,h,t)=\sum_{x\in E(g_{(x)},t)}attr(x,h,t) \tag{3.12}$$

定义 9 改进算法的特征向量。改进算法的特征向量相比较 D-Stream 算法增加了两项内容，如公式 3.13 所示。其中 C 为一个二维向量，记录密度网格 g 对其前两个邻居网格的引力值，count 记录密度网格的数据记录的数量。

$$（t_g,t_m,\mathrm{C},\mathrm{D},\mathrm{label},\mathrm{count},\mathrm{status}） \tag{3.13}$$

改进算法涉及到的相关概念如下：

（1）网格密度阈值的动态设定

密度网格的密度阈值能够对网格性质进行区分，阈值的选择能够对算法的质量生成直接影响[58]。密度阈值参数于 D-Stream 算法里需要被人工设置，以该参数作基础对网格的密度阈值进行计算，划分密度网格属性，若密度阈值过低，会混淆数据空间中的密集网格和稀疏网格，聚类结果中含有大量噪音，若密度阈值过高，则会将分布比较平均的簇识别为稀疏网格，这样只能发现少量高密度的区域。为了避免以上问题，本文设计的改进算法采用平均密度的思想，收集一部分数据，对其分布进行统计，然后基于统计信息实现网格密度阈值的动态设置。

改进算法先对一定数据量 K（$K<v\times 1s$，v 是数据流速）的数据记录进行

处理。统计所有非空网格，最大的网格密度记为 D_{max}，最小的网格密度记为 D_{min}，平均网格密度记为 D_{ave}，D_{ave} 的计算如公式 3.14 所示。

$$D_{ave} = \frac{\sum_{i=1}^{K} D_i}{K} \tag{3.14}$$

其中 D_i 为非空密度网格 g_i 的网格密度，K 为非空密度网格的数据量。那么，网格的密度阈值 D_m 和 D_l 的计算如公式 3.15 和 3.16 所示。

$$D_m = \frac{D_{max}+D_{ave}}{2} \tag{3.15}$$

$$D_l = \frac{D_{min}+D_{ave}}{2} \tag{3.16}$$

（2）时间周期 *gap* 值的确定

密度阈值调整后，相应的定期检查密度网格操作的时间周期也做出调整，调整后的时间周期 *gap* 的计算如公式 3.17 所示。

$$\text{gap} = \left\lfloor \log_{\lambda} \left(\max\left\{ \frac{D_l}{D_m}, \frac{1-D_m(1-\lambda)}{1-D_l(1-\lambda)} \right\} \right) \right\rfloor \tag{3.17}$$

（3）稀疏网格的簇边界判定方法

D-Stream 算法选用的是以密度网格作基础的聚类策略，围绕数据展开压缩存储，会致使数据记录的丢失，尤其是处于簇边界的密度网格。如图 3.2 所示，密度网格 a 由于网格密度过小，所以在聚类过程中不会被加入到簇中，最终造成该簇的边界数据缺失。

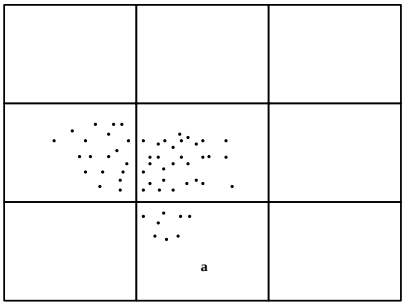


图 3.2 簇边界的缺失

改进算法针对 D-Stream 算法的这一缺陷，采用新的簇边界判定策略，对于稀疏网格 g，搜索其邻居密集网格 h，计算网格 g 与网格 h 之间的双向引力，当两个邻居网格强相关时将它们合并。定义两个邻居网格之间的双向引力均高于阈

值θ（$\theta > 0$）时，两个密度网格为强关联关系。定义$\theta = \frac{C_m}{|\rho|(1-\lambda)}$，其中$\rho = \{（g,h）|g \in S, h \in NB(g)\}$，以此有效地解决簇边界缺失的问题。该判定算法的具体描述如算法 3 所示。

算法 3 稀疏网格的簇边界判定方法 boundary_cluster(*grid_list*)

输入：全部稀疏网格 g

输出：更新稀疏网格 g 的 *label* 值

步骤：

1. **for each** 稀疏网格 g，搜索其所有邻居网格 h
2. **if**（q 为密集网格&$q \in c$）
3. 计算 q 与 h 之间的双向引力$attr(g,h,t)$和$attr(h,g,t)$；
4. **end if**
5. **if**（$attr(g,h,t) > \theta$、$attr(h,g,t) > \theta$）将 g 的 *label* 记为 c；
6. **else** 将 g 的 *label* 记为 NO_CLASS；
7. **end if**

3. 2. 3 算法描述

本节首先对改进算法的总体流程以流程图的形式进行介绍，并用伪代码的形式围绕改进算法展开详细描述。

（1）总体流程

改进算法的总体流程如图 3.3 所示。改进算法仍然分为在线阶段和离线阶段两个部分。改进算法在线部分处理快速到达的数据记录，将数据记录映射到划分好的网格当中，然后根据网格中数据对象的信息更新网格的特征向量，当数据空间中的数据量达到事先设置的 K 值时，统计含有数据的密度网格的密度值，从而计算密度网格的密度阈值，即 D_m 和 D_l，和定期检查的时间周期 *gap*，形成初始的簇结构并且对零星网格进行周期性的检测和删除。改进算法离线部分周期性地调整簇结构，在处理簇边界的稀疏网格时，引入网格引力概念对稀疏网格进行判定，以避免簇边界缺失的问题，最终完成聚类结果的输出。

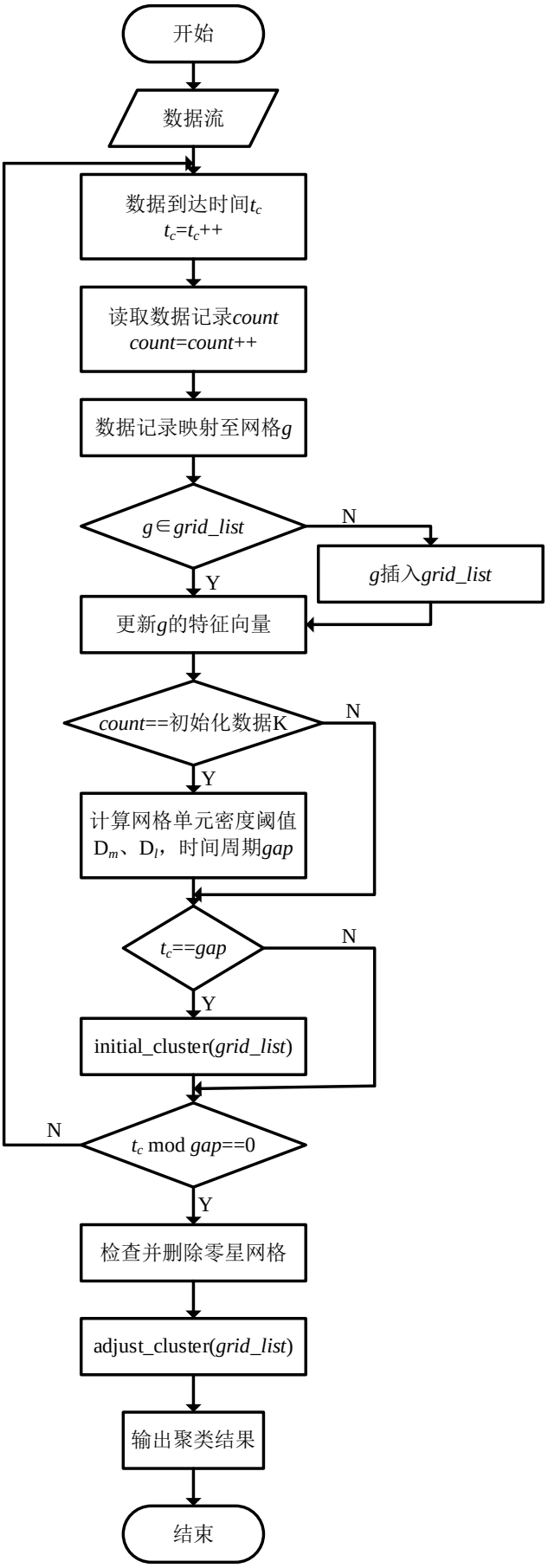


图 3.3 改进算法总体流程图

（2）改进算法在线部分算法描述

参考对 D-Stream 算法阈值设定时的缺陷的分析，本文改进了 D-Stream 算法，通过引入平均密度概念，自动设置网格密度阈值。改进算法在线部分算法描述如算法 4 所示。

算法 4 改进算法在线部分算法

输入：数据流 DS，密度网格划分参数 p，衰减系数 λ，初始化用的数据量 K

输出：网格簇集合

步骤：

```
1.  count=0，tc=0；
2.  对数据空间 S 进行划分，建立网格结构；
3.  while 数据流未结束 do
4.      读取数据记录 x = (x1, x2, … , xd)；
5.      tc++，count++；
6.      数据映射至相应网格 g；
7.          if（g∉grid_list）
8.              将 g 插入 grid_list 并更新网格单元 g 的特征向量；
9.          if（count==K）
10.             确定网格单元密度阈值 Dm、Dl，以及时间周期 gap；
11.         end if
12.         if（tc==gap）
13.             call initial_cluster（grid_list）；
14.         end if
15.         if（tc mod gap==0）
16.             检查和删除松散网格；
17.         end if
18. end while
```

（3）改进算法离线部分算法描述

改进算法离线部分主要包含簇结构的调整内容，密度网格为稀疏网格时，使用基于网格引力的簇边界判定算法对处于簇边界的稀疏网格进行判定，改进算法离线部分具体描述如算法 5 所示。

算法 5 改进算法离线部分算法

输入：所有网格单元

输出：聚类结果

步骤：

1. **for each** 经过时间周期 *gap* 后状态发生变化的网格单元 *g*
2. **if**（*g* 变为稀疏单元）
3. call boundary_cluster（*g*∈*grid_list*）；
4. **if**（*g* 的 *label* 为空）
5. *g* 从所属簇 *c* 中删除，若簇 *c* 不连通，则将 *c* 分为两个簇；
6. **end if**
7. **else if**（*g* 变为稠密网格）
8. **for each** 网格 *g* 的邻接网格中所属簇 *c1* 核心单元最大的网格 *h*
9. **if**（*h* 是稠密网格）
10. **if**（*g* 的 *label* 为空） 将 *g* 的 *label* 标记为 *c1*；
11. **else if**（*g*∈*c* and |*c*|>|*c1*|） 簇 *c* 合并簇 *c1*；
12. **else if**（*g*∈*c* and |*c*|≤|*c1*|） 簇 *c1* 合并簇 *c*；
13. **else if**（*h* 是过渡网格）
14. **if**（*g* 的 *label* 为空 and 若加入簇 *c*，*h* 变为外部网格）
15. 将 *g* 的 *label* 标记为 *c1*；
16. **else if**（*g*∈*c* and |*c*|>|*c1*|） *h* 的 *label* 标记为 *c*；
17. **else if**（*g* 变为过渡网格）
18. 在 *g* 的所有邻接网格中查找到能满足 *g* 加入该簇可以成为外部网格的最大的簇 *c2*；
19. **end if**
20. 输出聚类结果；

3.3 实验与结果分析

为了检验改进算法的性能，对算法聚类能力进行直观评估，本节设计改进算法与 D-Stream 算法的对比实验，对改进算法的聚类结果进行评估。

3.3.1 实验数据与实验环境

实验软件环境是 64 位 Windows10 处理系统，硬件环境是 Intel Core i7，Intel

Q270 系列芯片组，16GB 内存。算法均使用 Java 语言实现。

第一部分实验选用仿真数据集，数据集为 50K 的二维任意形状数据集，涵盖 5 类数据，每个数据有 2 维属性。第二部分实验采用 KDD-CUP-99 数据集和 Forest CoverType 数据集。KDD-CUP-99 数据集包括 5 类数据，每个数据有 42 维属性。Forest CoverType 数据集包括 7 类数据，各数据具备 54 维属性。实验选用 KDD-CUP-99 数据集里 34 维的连续属性，Forest CoverType 数据集里 10 维的连续属性。所有的实验数据都做了标准化处理。

3.3.2 算法参数的设置

聚类算法的参数设定在很大程度上决定着算法的聚类品质，在进行实验前，调整算法的参数是非常必要的，本节对改进算法中参数的设置进行讨论。

衰减因子λ体现了历史数据在整个聚类结果中所占的比重，可以根据具体情况进行设置，参考D-Stream算法原始论文，统一各算法衰减因子λ =0.998。

在改进算法网格密度阈值设置阶段，计算网格平均密度需要统计数据记录的数量K（$K < v \times 1s$，v是数据流速），K的值不应太小，否则确定的密度阈值不能充分体现数据记录的平均密度，所以设置v=1000条/s。

在以网格为基础的聚类算法中，数据空间 S 被划分成了$N(N = \prod_{i=1}^{d} p_i)$个密度网格，密度网格的划分比较细时算法的聚类质量较好，但是过细的划分会增加密度网格的数量，从而降低算法的处理速率，因此网格划分参数 p 的取值应该折中考虑。图 3.4 为改进算法对 KDD-CUP-99 数据集进行聚类时，将数据空间划分成不同数量的网格时的算法聚类精确度对比，能够得知，p 取较高数值时的聚类精确度高于 p 取较低数值时。考虑到网格数量过多对内存消耗增大的问题，设置 p=50。

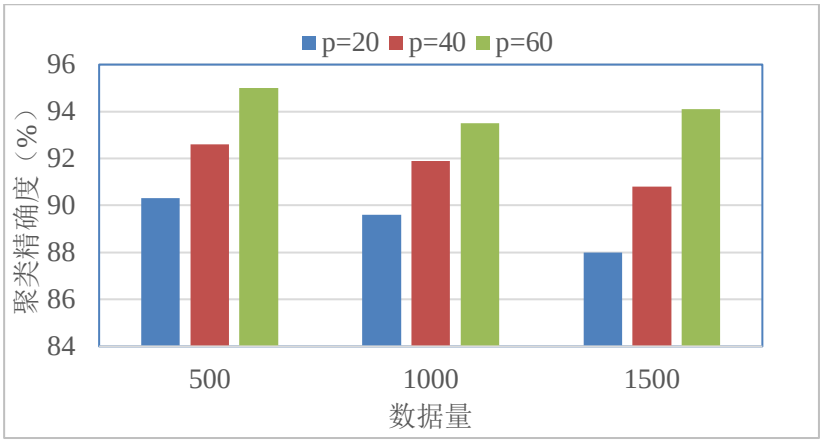
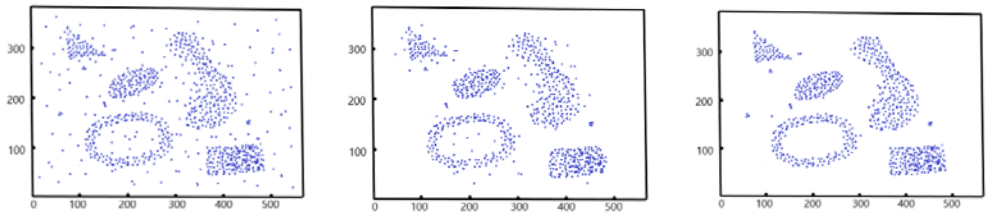


图3.4 不同网格划分参数改进算法的聚类精确度

3.3.3 实验结果分析

为检测改进算法对数据集的处理能力，第一部分实验使用改进算法和 D-Stream 算法对仿真数据集进行处理。根据经验设置 D-Stream 算法 C_m=3.0，C_l=0.8。原始数据集总体分布如图 3.5（a）所示，经过 D-Stream 算法聚类之后的类簇分布如图 3.5（b）所示，经过改进算法聚类之后的类簇分布如图 3.5（c）所示，实验结果表明，改进算法能够识别出任意形状的类簇，对簇边界的处理效果也优于 D-Stream 算法，说明改进算法在簇边界处理方面的改进对聚类结果有着正向的影响。



（a）原始数据集　　（b）D-Stream 算法聚类后　　（c）改进算法聚类后

图 3.5 聚类结果展示

为验证改进算法的聚类效果，第二部分实验选用改进算法及 D-Stream 算法围绕 KDD-CUP-99 及 Forest CoverType 两类数据集展开聚类，采用聚类精确度作为聚类结果的评估指标，对聚类结果进行评估。

聚类精确度（Accuracy AC）为当下的一个流行聚类评价指标，可以基于真实标签已知的情况，对聚类结果及实际标签间的吻合水平进行分析。其定义如公式 3.18 所示。

$$\mathrm{AC}=\frac{\sum_{i=1}^{n}\delta(S_i,map(r_i))}{N} \tag{3.18}$$

其中，r_i为聚类后的标签，S_i为真实标签，n 为数据总的个数，δ为指示函数，指示函数定义如公式 3.19 所示。

$$\delta(\mathrm{x},\mathrm{y})=\begin{cases}1 & if\ \ x=y\\ 0 & otherwise\end{cases} \tag{3.19}$$

根据以往经验，D-Stream 算法 C_m=3.0，C_l=0.8 时聚类效果最为稳定，故本实验算法设置参数与第一部分实验相同，实验结果如图 3.6、图 3.7 所示，图 3.6

为两种算法对 KDD-CUP-99 数据集进行聚类的聚类精确度对比结果，图 3.7 为两种算法对 Forest CoverType 数据集进行聚类的聚类精确度对比结果。从两种数据集上的实验结果可以得出，改进算法在不同数据量下的聚类精确度均高于 D-Stream 算法，虽然 D-Stream 算法根据以往经验选择了能够使聚类结果相对稳定的密度阈值，但是随着数据量的增多，D-Stream 算法的聚类精确度明显变低，改进算法动态设置密度阈值的策略使得聚类精确度保持稳定状态，并且面对不同的数据集时处理效果基本稳定。说明改进算法使用动态设定网格密度阈值策略进行聚类的效果是优于根据现有经验设定阈值方法的。改进算法的聚类精确度也可以说明改进算法具有较高的聚类质量。

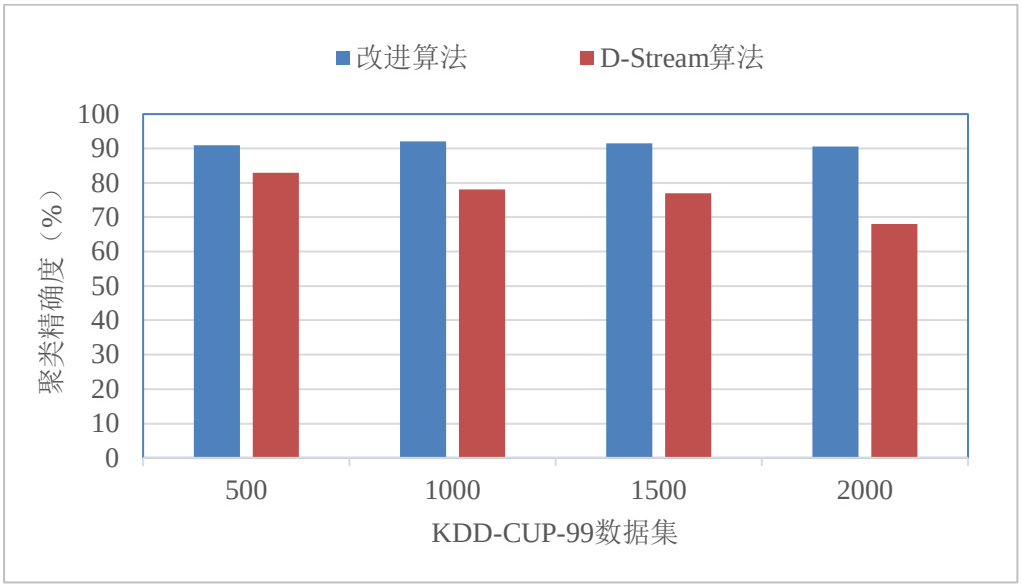


图 3.6 KDD-CUP-99 数据集上聚类精确度对比图

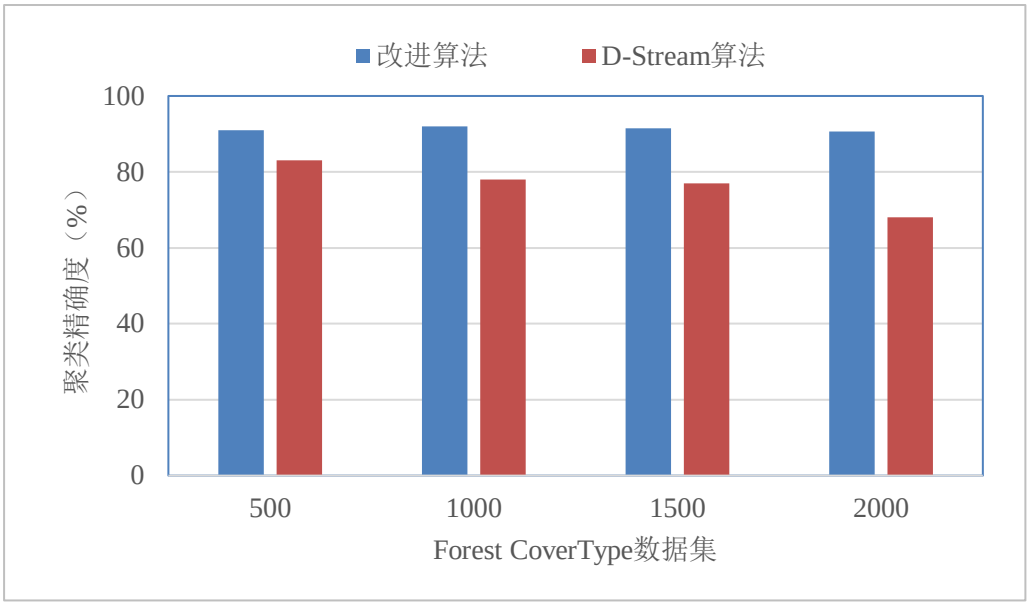


图 3.7 Forest CoverType 数据集上聚类精确度对比图

3.4 本章总结

本章对数据流聚类算法展开了详细研究，分析 D-Stream 算法，围绕 D-Stream 算法网格密度阈值设定和簇边界判定两方面的问题，提出改进算法。于改进算法里能够对网格密度阈值进行动态设置，使得因先验知识不足致使的初始参数设定失误的问题得到解决。并且引进网格引力概念，对处于簇边界的稀疏网格进行判定，进一步精确对聚类簇边界数据的确定，减少簇边界的缺失，提高聚类的精确度。此外还对改进算法及 D-Stream 算法开展了对比实验，对改进算法的特性优势进行了检验。

第四章 算法并行化设计及其基于 Storm 的实现

基于上一个章节对数据流聚类算法提出的改进，本章将改进算法与流处理平台 Storm 的结构逻辑相结合，设计并提出分布式的数据流聚类算法，并且搭建 Storm 环境，基于实际采集的数据对分布式数据流聚类算法进行实现，通过设计对比实验测试并行化设计对算法聚类效果和处理时间带来的影响。

4.1 算法并行化设计

4.1.1 算法并行化设计基本思想

本节从改进算法的在线部分和离线部分两个方面分别对改进算法进行并行化设计。算法在线部分将网格结构的生成分为局部生成和全局生成两个部分。离线部分对在线部分生成的各局部网格结构进行全局聚类，得到最终的聚类结果。改进算法的总流程如图 4.1 所示。

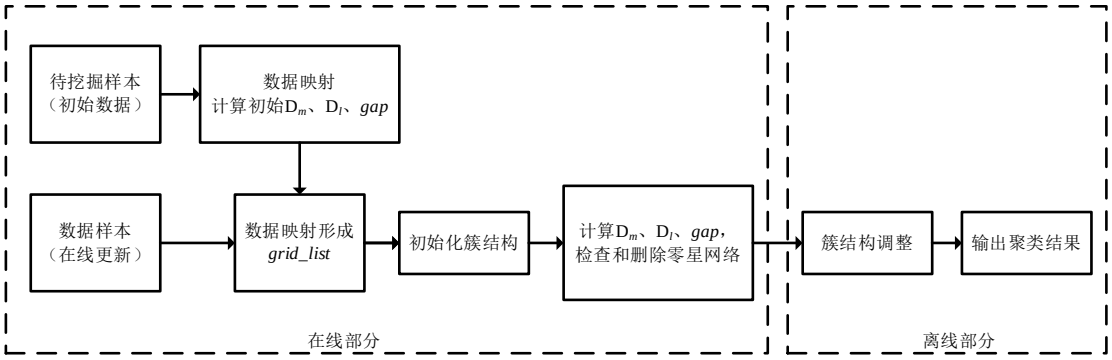


图 4.1 改进算法总流程

在线部分首先接收一部分的数据，参数生成节点计算算法的初始参数，数据接收节点获取数据流，局部网格生成节点生成网格结构，全局网格生成节点合并所有局部网格生成节点的网格结构，完成网格结构的整体生成。在时间到达第一个时间周期 *gap* 后，生成的整体网格结构被发送至聚类初始化节点和参数计算节点，聚类初始化节点根据网格结构生成初始类簇，参数计算节点更新算法参数，将生成的初始类簇和更新的算法参数发送至整体聚类节点，整体聚类节点合并现有的所有簇结构，形成聚类结果。在对算法进行并行化设计的总体流程中，各节点之间的操作任务相对独立，互不干预。

4.1.2 分布式数据流聚类算法概述

基于 4.1.1 节中对改进算法进行并行化设计的基本思想，本节设计分布式数据流聚类算法，算法分布式的思想体现在以下两个方面：

（1）局部网格结构的生成

网格结构的生成工作被分派到多个局部网格生成节点，各局部网格生成节点接收被平均分配的数据流 DS，分别将数据流 DS 映射到划分好的网格中，更新网格的特征向量，从而生成局部的网格结构，当局部网格生成节点中处理的数据流 DS 的数量到达阈值 w（$w=(gap/2)\times v\times 1s$，v 为数据流的流速）时，局部网格生成节点将生成的局部网格结构 *grid_list* 发送到全局网格生成节点。局部网格结构生成部分流程如图 4.2 所示。

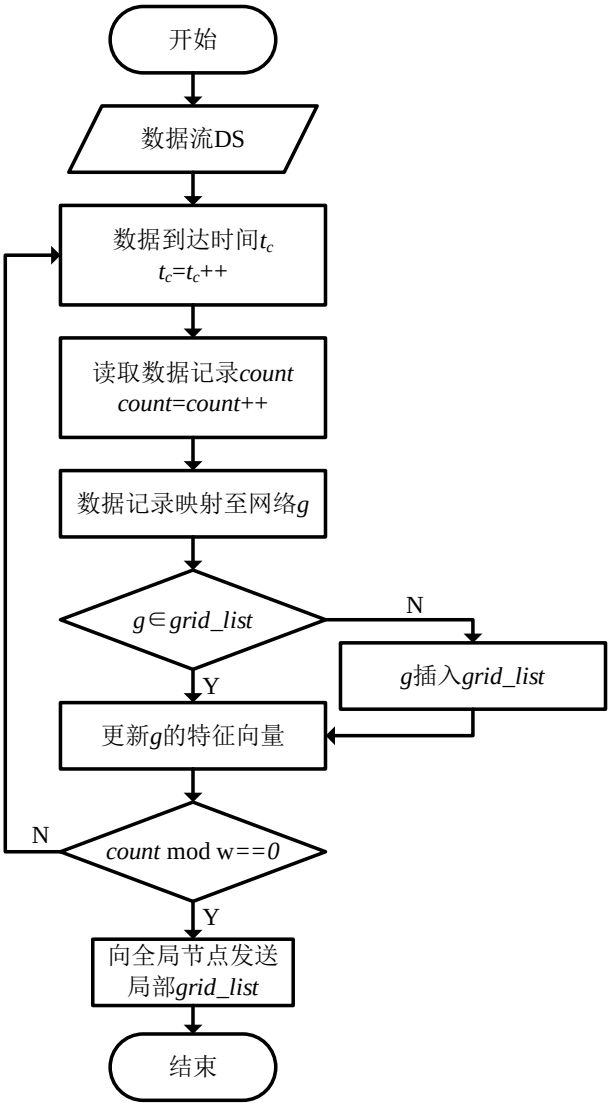


图 4.2 局部网格结构生成流程

（2）全局网格结构的生成

全局网格生成节点接收各局部网格生成节点发送的网格结构，根据各网格 *id* 对网格结构进行合并，将存在相同 *id* 网格组的网格加入到该网格组，不存在相同 *id* 网格组的网格加入到 *grid_list* 中，生成全局网格结构 *grid_list*，并且每隔一个时间周期 gap 发送全局网格结构 *grid_list* 至下一个节点。全局网格结构生成部分流程如图 4.3 所示。

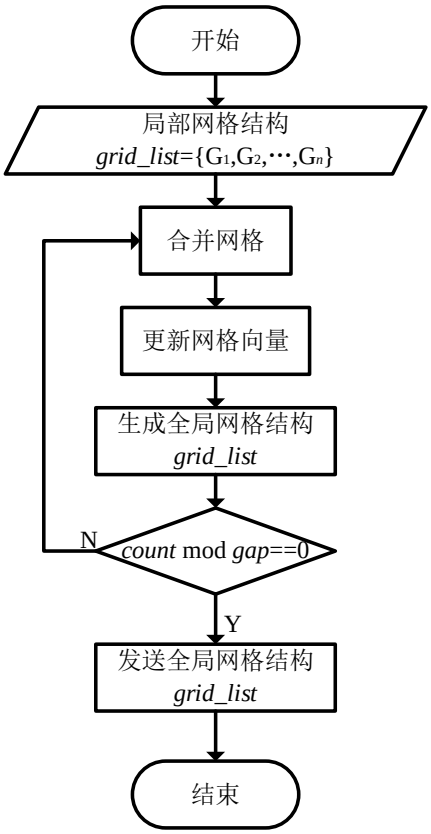


图 4.3 全局网格结构生成流程

4.2 算法基于 Storm 的实现方案

分布式数据流聚类算法各部分的任务分配是依据 Storm 的逻辑单元任务拓扑（topology）进行设计的，在 Storm 集群中，计算任务被打包成任务拓扑进行发布，每个任务拓扑之中包含若干个 Spout 组件和 Bolt 组件[59]，Spout 组件负责接收外部数据，Bolt 组件负责处理数据。分布式数据流聚类算法的拓扑结构如图 4.4 所示。

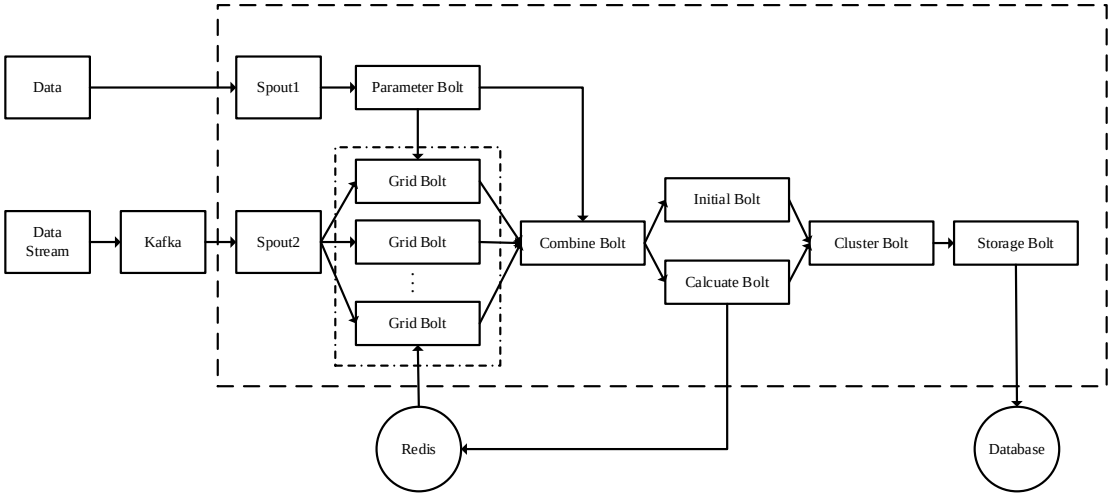


图 4.4 分布式数据流聚类算法拓扑结构

在算法拓扑结构中，Spout1 负责接收衰减因子 λ、网格划分参数 p 和数据流速 v 等算法参数数据，并且将这些数据传送到 Parameter Bolt 和 Combine Bolt。Spout2 负责接收需要处理的数据流，并且将数据传送到 Grid Bolt。

由于原始数据流的流速是不确定的，数据流流速的突然变化会导致数据计算的失败，或者会导致整个拓扑结构的瘫痪。为了可以让算法正常运行，在数据流的获取过程中使用分布式发布订阅消息系统 Kafka 作为原始数据流与 Spout2 之间的消息中间件，以容错的方式获取数据流并再次发布，以达到调节数据流流速的效果。

Parameter Bolt 接收 Spout1 传送的数据，并完成计算算法初始参数的任务，然后将参数传送到各 Grid Bolt 和 Combine Bolt。

Grid Bolt 接收 Spout2 传送的需要处理的数据流、Parameter Bolt 传送的参数和 Redis 传送的更新的算法参数和网格结构，根据这些数据，将数据流映射到相应的密度网格，同时更新密度网格的特征向量，实现网格结构的局部生成，并且按周期将局部网格结构传送到 Combine Bolt。

Combine Bolt 接收各 Grid Bolt 传送的局部网格结构和 Parameter Bolt 传送的初始参数，根据这些数据，合并各 Grid Bolt 传送的局部网格结构，生成全局网格结构，并且每隔一个时间周期 gap，传送全局网格结构到 Initial Bolt 和 Calculate Bolt。

Initial Bolt 接收 Combine Bolt 传送的全局网格结构，在第一个时间周期 gap 后完成聚类簇结构的初始化，并将初始聚类簇结构传送到 Cluster Bolt。

Calculate Bolt 接收 Combine Bolt 传送的全局网格结构，根据全局网格结构中的数据，计算更新的算法参数，并且检查和删除零星网格，生成更新的网格结构。

算法在局部生成网格结构的部分，除了参考初始参数之外，还需要参考更新全局网格结构后的更新参数，这样就形成了算法内的一个环形结构，然而 Storm 的拓扑结构是有向无环的结构，所以在 Calculate Bolt 和 Grid Bolt 之间使用分布式的键值对数据库 Redis 作为中间存储，以打破算法中的环形结构[60]。

Cluster Bolt 接收 Calculate Bolt 传送的更新的网格结构和 Initial Bolt 传送的初始聚类簇结构，根据这些数据进行聚类，更新聚类簇结构，并将最终的聚类结果传送到 Storage Bolt。

Storage Bolt 存储最终的聚类簇结构。

4.3 实验与结果分析

为验证分布式数据流聚类算法的聚类质量和分布式设计的时间效率优势，本节设计并行化处理实验，测试算法的并行化对数据处理时间和算法聚类结果的影响。

4.3.1 实验数据与实验环境

实验硬件环境方面使用 VMware Workstation 15.5 pro 模拟 4 台虚拟机，设置 1 个 Nimbus 节点、3 个 Supervisor 节点。各节点信息如表 4.1 所示。

表 4.1 各节点信息

序号	主机名	IP 地址
1	Nimbus	59.67.152.20
2	Supervisor1	59.67.152.21
3	Supervisor2	59.67.152.22
4	Supervisor3	59.67.152.23

实验软件环境方面使用的相关软件及版本如表 4.2 中所示。

表 4.2 软件环境信息

软件	版本
Linux	Centos7
JDK	jdk1.8.0_231
Zookeeper	Zookeeper3.6.1
Storm	Storm2.1.0
Kafka	Kafka2.4.1
Flume	Flume1.9.0
Redis	redis2.4.5

实验数据使用采集的本学院官方网站 2020 年 5 月 10 日至 16 日连续一周的 Web 访问日志数据，数据共 182605 条，选取其中 8 维属性，进行数据预处理。

4.3.2 实验结果分析

为验证分布式数据流聚类算法的聚类质量，改变 Storm 集群环境的线程数进行实验，分别选择单线程、线程数为 2 和线程数为 4 时进行算法测试，对比不同线程数时聚类结果的轮廓系数。其中，不同线程针对的是 Grid Bolt 的线程数。图 4.5 为不同线程数时聚类结果的轮廓系数对比。轮廓系数是一种通过计算数据对象之间的相似度来判定聚类质量的评价指标，轮廓系数的大小可以体现各类簇内数据的密集程度，适用于无标签数据集的聚类结果评估[61]。其计算方式如公式 4.1 所示。

$$S = \frac{b-a}{\max(a,b)} \tag{4.1}$$

a 为该数据对象与其所在簇内其他数据对象的平均距离；b 为该数据对象与距离它最近的另一个簇内数据对象的平均距离。聚类结果的整体轮廓系数如公式 4.2 所示。

$$S_k = \frac{1}{n}\sum_{i=1}^{n} S_i \tag{4.2}$$

轮廓系数的取值范围为[-1,1]，轮廓系数的值越大，表示各类簇内数据之间越密集，聚类结果的准确率越高。由图 4.5 可见，算法在不同线程数时的整体轮廓系数值相差不多，且都高于 0.9，说明算法的聚类质量稳定且不受分布式处理环境影响。

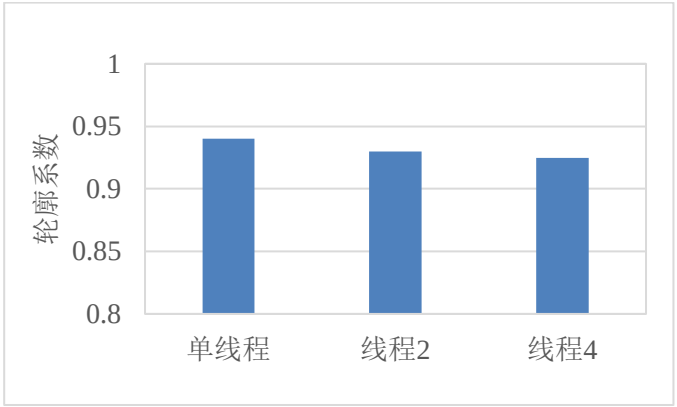


图 4.5 不同线程轮廓系数对比结果

为验证分布式数据流聚类算法的时效性，对算法的时间效率进行测试，根据图 4.5 所示，将 Grid Bolt 的线程数设置为 4 时聚类效果最好，所以本次实验将集群环境下算法线程数设置为 4，对比在处理不同数据量的数据时，算法在集群环境与单机环境下所需运行时间。实验结果如图 4.6 所示，在数据量为 50000、10000 和 150000 时，集群环境下的算法运行时间均低于单机环境，并且数据量增加时，单机环境下的算法运行时间开始大幅度增长，集群环境下的运行时间增长缓慢。由此可以分析得出，基于 Storm 的算法并行化可以提高算法的处理效率，降低算法运行时间，算法的并行化设计是可行的。

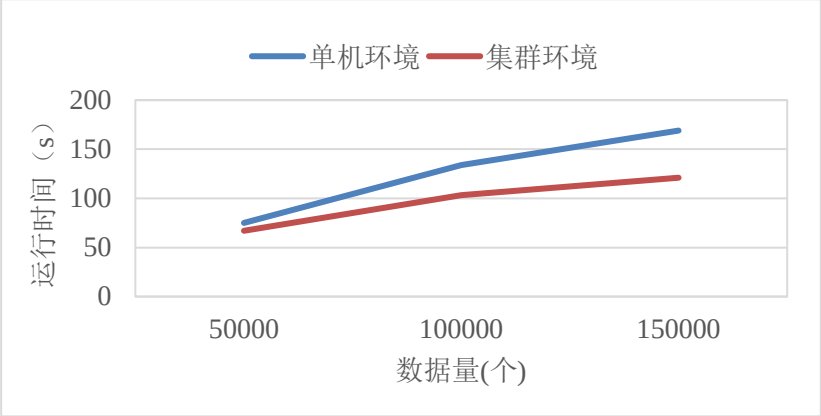


图 4.6 算法聚类时间测试结果

4.4 本章总结

为提高改进算法的处理效率，本章基于上一章节提出的改进算法，结合流处理平台 Storm 的计算逻辑，设计并提出了分布式的数据流聚类算法，并且搭建了 Storm 集群环境，基于实际采集的数据对分布式数据流聚类算法进行实现，通过设计对比实验验证了并行化设计下的改进算法具有稳定的聚类效果，并行化的设计提高了改进算法的处理效率。

第五章 基于 Storm 的改进算法在 Web 日志分析中的应用

基于上一个章节对改进算法的并行化设计，本章将分布式数据流聚类算法应用于校园网站 Web 访问日志的分析，首先对 Web 访问日志的格式和内容进行详细说明，建立基于 Storm 的 Web 访问日志分析模型。使用基于 Storm 的改进算法对采集的学院官方网站 Web 访问日志进行分析。

5.1Web 访问日志格式说明

本文以本学院官方网站 Web 访问日志为研究对象进行分析。学院网站使用 IIS 网站服务器，日志服务选用 W3C 日志格式。IIS 日志样例及各字段说明如表 5.1 所示。

表 5.1 IIS 日志样例及各字段说明

序号	属性名称	属性说明	日志样例
1	date	客户端发出请求的日期	2020-08-23
2	time	客户端发出请求的时间	00:00:10
3	s-sitename	服务名	W3SVC
4	s-ip	服务端 IP 地址	59.67.152.3
5	cs-method	请求方法	GET
6	cs-uri-stem	请求 URI 资源	/images/txwgg.gif
7	cs-uri-query	URI 查询	-
8	s-port	服务器端口	80
9	cs-username	通过验证的域或用户名	-
10	c-ip	客户端 IP 地址	42.***.**.75
11	cs(User-Agent)	用户代理	Baiduspider+(+http://www.baidu.com)
12	sc-status	协议状态码	200
13	sc-substatus	协议子状态码	0
14	sc-win32-status	win32 状态	64
15	sc-bytes	服务器发送的字节数	185173
16	cs-bytes	服务器接受的字节数	296
17	time-taken	操作所花时间	2

5.2 基于 Storm 的 Web 访问日志分析模型设计

为了满足 Web 访问日志分析的各项需求，结合 Web 访问日志实际情况，从分布式数据流聚类算法的流程和 Storm 平台的特性考虑，本节设计基于 Storm 的 Web 访问日志分析模型。模型分为数据处理模块和后台学习模块。数据处理模块主要负责对 Web 日志数据进行采集，然后对采集到的数据进行标准化处理产生训练集。后台学习模块使用设计好的改进算法对经过处理的训练集进行聚类，对得到的聚类结果进行存储和标记，得到关于日志信息的知识库。模型结构如图 5.1 所示。

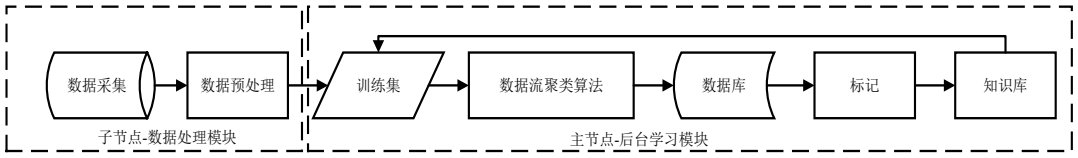


图 5.1 分布式 Web 访问日志分析模型结构

各模块作用如下：

（1）数据处理模块

数据处理模块由两部分构成，分别是数据采集和数据预处理。数据采集部分使用工具监测数据源的生成并进行采集，生成原始数据集。实验过程使用 Flume 进行 Web 访问日志的采集，Flume 配置完成后可以实现自动检测，自动采集。并且可以通过配置 source，扩展成同时收集几个日志服务器的日志。数据预处理部分首先对原始数据集进行去躁和降维的数据清洗，然后提取数据特征值，对特征值进行标准化处理。

（2）后台学习模块

后台学习模块由四部分构成，分别是生成训练集、运行数据流聚类算法、聚类结果标记和形成知识库。生成训练集部分将预处理后的日志数据作为训练集，当训练集中数据达到一定的数量时，开始运行数据流聚类算法部分，该部分对训练集中的数据进行聚类，得出聚类簇，并将其存储在数据库中。聚类结果标记部分将生成的各类簇进行分析并标记类簇的类型。形成知识库部分使用标记好的各类簇生成规则库。

5.3 模型应用结果及分析

实验采集本学院官方网站真实的 Web 访问日志数据，以 5 月 10 日至 16 日

连续一周为例绘制图 5.2 所示折线图。

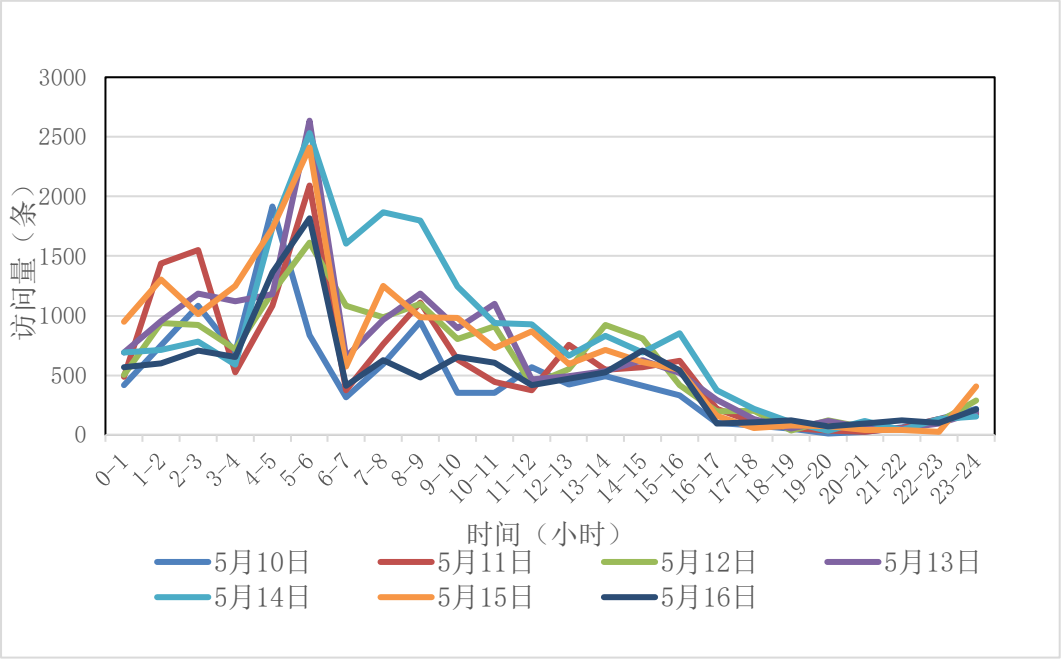


图 5.2　2020 年 5 月 10 日至 16 日网站访问量折线图

从该图中可以初步得出，每天 5:00 到 6:00 时间段，访问量普遍到达最高点，这个时间段本应是休息时间，却产生了大量的访问日志，非工作时间高频访问大多为异常访问行为，所以相应的在这段时间应该加强网站的防护，以防有异常访问或者攻击行为的出现。每天 20:00 到 21:00 时间段，访问量普遍到达最低点，这个时间段用户对网站的访问行为最少，网站管理者可以选择在这段时间对网站进行维护及更新。

在对 Web 访问日志的聚类分析中，同样采用本学院官方网站 2020 年 5 月 10 日至 16 日的 Web 访问日志数据。经过参数设定和并行度设定之后，达到如下聚类结果：数据集一种被分成了 13 个类簇，其中数据对象数量超过 5000 个的类簇一共有 6 个，这 6 个类簇包含的典型数据对象如表 5.2 所示：

表 5.2 类簇内典型数据对象

簇号	date	time	s-ip	cs-method	cs-uri-stem	c-ip	sc-status
1	2020-05-16	11:15:02	59.67.152.3	GET	/images/mttszy.gif	36.**.**.51	200
2	2020-05-13	05:13:22	59.67.152.3	POST	/moon.php	43.***.**.31	404
3	2020-05-17	03:24:49	59.67.152.3	GET	/favicon.ico	117.***.**.146	404
4	2020-05-16	02:29:45	59.67.152.3	GET	/robots.txt	203.***.**.67	404
5	2020-05-16	01:30:11	59.67.152.3	HEAD	/webadmin/login.php	121.**.***.188	404
6	2020-05-14	13:29:38	59.67.152.3	GET	/dbke/etc/passwd	211.**.**.22	404

对以上 6 个类簇的聚类结果进行分析，分析结果如下：

1 号类簇数据对象数量最多，一共包含 121442 个数据节点。对该簇内数据对象进行分析，发现簇内数据对象 cs-method 属性为 GET，sc-status 属性为 200，cs-uri-stem 属性内容基本相似，大多是对学院网站上已有内容的请求。可以说明此类访问用户以 GET 方法访问了网站中的资源，并且都访问成功，作为网站管理者可以将此类簇内 Web 访问认定为正常的用户访问行为。

2 号类簇共包含 19054 个数据节点。对该簇内数据对象进行分析，发现该簇内数据对象 c-ip 属性基本一致，sc-status 属性为 404，time 属性多集中在 00:00-07:00。可以说明此类访问多为同一用户的非工作时间高频访问。用户对网站的正常访问一般都集中在工作时间，非工作时间的高频访问很大概率是异常访问行为[62]。这类行为虽然对网站工作时间内的服务影响不大，但也是在消耗网站资源，作为网站管理者应该关注此类访问行为，非工作时间也不能放松对网站的管理和维护，也应对有此类访问行为的 IP 进行判定和标记。

3 号类簇共包含 10354 个数据节点。对该簇内数据对象进行分析，发现该簇内数据对象 sc-status 属性为 404，cs-uri-stem 属性都为/favicon.ico。可以说明此类访问用户都访问了/favicon.ico 资源，并且未找到该文件。favicon 的全称是 favorites icon，/favicon.ico 是网站在浏览器标题和收藏夹标签中的精简图标。如果网站的目录中没有这个图标，在访问请求中会报 404 的错误。这类问题虽不会影响网站的正常使用，但是因此而产生的大量访问错误日志会对网站的管理带来负担，在网站的页面设计方面可以针对这个问题做出改进。

4 号类簇共包含 8406 个数据节点。对该簇内数据对象进行分析，发现该簇内数据对象 sc-status 属性为 404，cs-uri-stem 属性都为/robots.txt。可以说明此类访问用户都访问了/robots.txt 资源，并且未找到该文件。搜索引擎爬虫对网站进行爬取时，会根据网站根目录下设定的 robots.txt 来决定爬取哪些内容、不爬取哪些内容。由表 5.2 中簇 4 包含的典型数据对象可以看出，某搜索引擎在 5 月 16 日对本网站进行了一次爬取。对本次访问方 IP 地址进行查询得出，该 IP 地址对应的是“北京市谷歌(中国)公司”，那么可以将此次访问行为看成谷歌中国搜索引擎对本网站进行了爬取。

robots.txt 是网站与网络爬虫之间的协议，网站使用一个 txt 文本定义网站允许网络爬虫访问的权限。当一个网络爬虫访问某个网站时，网络爬虫首先检查该网站的根目录中 robots.txt 文件是否存在。如果存在 robots.txt 文件，网络爬虫则按照文件中规定的访问范围来进行爬取，如果不存在 robots.txt 文件，那么网络爬虫便可以爬取网站中没有被口令保护的全部内容。鉴于网站的 Web 访问日志中出现了此类访问行为，网站的开发人员可以在搭建网站时，在网站的根目录中

创建一个 robots.txt 文件，声明该网站中不想被网络爬虫访问的部分，这样就可以在一定程度上保护网站内容不完全被网络爬虫收录。

5 号类簇共包含 7175 个数据节点。对该簇内数据对象进行分析，发现该簇内数据对象 sc-status 属性为 404，cs-method 属性为 HEAD，cs-uri-stem 属性多包含 login.php。可以说明此类访问用户试图使用 HEAD 方式请求 login.php 资源，但都请求失败。HEAD 请求可以检查页面中超链接的有效性，也可以检查页面是否有被修改过的痕迹，多用于网络爬虫收录网页的标志性信息，获取网站的更新内容，传输网站的安全认证信息等。此类访问行为需要引起注意，可以在服务器配置中对 HEAD 请求进行限制。

6 号类簇共包含 6313 个数据节点。对该簇内数据对象进行分析，发现该簇内数据对象 sc-status 属性为 404，cs-uri-stem 属性都包含/etc/passwd。可以说明此类访问用户试图访问/etc/passwd 的资源，且访问失败。这类操作很明显是本地文件包含[63]尝试，一般在网站的开发过程中，会把网站经常使用的一些函数和内容写到一个文件当中，在网站需要时调用并加载相关内容，以节省开发时间，这个过程称为文件包含，但是当客户端可以调用这些文件时，会造成文件包含漏洞。/etc/passwd 是文件包含漏洞中常见的敏感信息路径，/etc/passwd 文件内保存着网站的用户信息，如果/etc/passwd 文件被破坏，系统便无法读取用户信息，用户便无法正常登录网站。为应对此类问题，网站开发人员在对网站进行建设时应从代码层和 Web 服务器安全配置两方面进行防范，在对网站进行防护时须对此类 IP 多加注意，对于存在此类攻击行为的访问者 IP，可以在防火墙进行设置拦截。

除了上述数据对象数量较多的类簇之外，还有一些数据量小的类簇，比如 cs-uri-stem 属性中含有 1=1 的内容，此类数据表示网站有被 SQL 注入攻击的风险，攻击者在正常的查询语句后添加相应的 SQL 语句，用于获得网页未授权的可查询数据；还有 cs-uri-stem 属性中含有<script> </script>等内容，表示有访问者试图在请求地址中包含测试脚本进行跨站脚本攻击，通过在网站中添加虚假内容，欺骗用户进行访问从而盗取用户信息。

表 5.3 聚类结果的应用范围

簇号	应用范围
1、2	网站的日常维护
3	网站的开发设计
4、5、6	网站的安全防护

通过上述对 Web 访问日志的聚类分析，可以总结出很多用户访问行为，这些访问行为的分析结果主要可以应用于表 5.3 所示三个方面，分别是可以应用于

网站的日常维护方面、网站的开发设计方面和网站的安全防护方面。这些筛选出的信息与网站建设息息相关，充分利用这些信息可以对网站建设做出贡献，特别是校园网站，校园网站属于综合性质的网站类型[64]，多用于发布消息，提供师生服务等，但是校园网站多疏于管理，会存在一些安全隐患，对于校园网站 Web 访问日志的分析不同于经营性网站，不过多的关注用户的访问习惯偏好，对校园网站 Web 访问日志的分析更偏向于安全性和实用性的问题。近年来，众多的高校正投入大量的资金和人力来进行校园网站的建设工作，不断提高校园网站的各方面性能，以达到更好地为高校师生提供服务的效果，将自动化手段应用于校园网站的管理与建设对校园网站的发展有着积极意义，本文得出的分析结果希望可以对校园网站的日常监管和防护提供有益思路。

5.4 本章总结

本章研究改进算法在基于 Storm 的 Web 日志分析中的应用。对 Web 访问日志的格式和内容进行了详细说明，根据 Web 访问日志分析的各项需求，建立了基于 Storm 的 Web 访问日志分析模型，并介绍了各模块的功能。使用基于 Storm 的改进算法对采集的高校校园网站 Web 访问日志进行了分析，分析得出的结论与现实相结合，对高校网站建设提供一定的参考。

第六章 总结与展望

6.1 总结

互联网技术迅猛发展，互联网应用高度普及，人们在使用互联网访问网站的过程中，海量的日志文件存储在网站的服务器上，对日志进行分析有助于网站管理员及时发现网站安全性问题，充分了解用户访问行为，以更新维护相关内容，更好地建设网站，发挥网站的作用。

本文针对 Web 访问日志的分析问题，重点对数据流聚类算法和分布式流处理技术进行了研究，对基于密度网格的数据流聚类算法进行了改进，并基于分布式实时大数据处理框架 Storm 对改进算法进行了并行化设计，最终将分布式数据流聚类算法在 Web 访问日志分析的实际场景中加以应用。

本文的主要工作如下：

（1）为对 Web 访问日志进行分析，对 Web 日志挖掘、数据流聚类算法的相关研究和 Storm 计算框架的相关技术进行了综述和分析。选择使用基于网格密度的数据流聚类算法作为 Web 访问日志分析的基础算法，从算法阈值的设定和簇边界处理策略两个方面对原始算法进行了改进，设计了改进算法。并设计对比实验验证了改进算法的聚类效果和聚类精度。

（2）为应对单机环境不能有效应对海量数据处理的问题，搭建了 Storm 分布式框架，基于改进的数据流聚类算法和 Storm 流计算框架的特点，设计了分布式数据流聚类算法。并通过对比实验验证了分布式数据流聚类算法的时效性。

（3）基于上述研究，实现了分布式数据流聚类算法在 Web 访问日志分析中的应用，设计了分布式的基于 Storm 的 Web 访问日志分析模型，对实际采集到的校园网站 Web 访问日志进行了实验，得出了聚类结果并进行了分析，得到的聚类结果与现实情况相符合，结果分析对校园网站的管理建设可以起到一定的参考价值。

6.2 展望

本文所做工作还可以做进一步的研究与拓展，之后的研究可在如下几个方面进行：

（1）本文沿用 D-Stream 算法中平均划分的策略对数据空间中的密度网格进行划分，可以考虑采用不同的网格划分策略对改进算法进一步优化。

（2）本文改进算法基于 D-Stream 算法，D-Stream 算法不能高效率的处理高维数据，可以考虑采用投影概念对数据进行降维，保证高维数据可以被有效处理；

（3）本文使用基于 Storm 的改进算法对 Web 日志进行了聚类，得出聚类结果，可以结合实际情况，将聚类结果应用到网站的建设和防护工作之中。

参考文献

[1]李学龙,龚海刚.大数据系统综述[J].中国科学:信息科学,2015,45(01):1-44.

[2]米加宁,章昌平,李大宇,等.“数字空间”政府及其研究纲领——第四次工业革命引致的政府形态变革[J].公共管理学报,2020,17(01):1-17+168.

[3]冯秋燕.基于 Web 应用的日志异常检测与用户行为分析研究[D].广州:华南理工大学,2019.

[4]陈洲.一种改进 K-Means 算法的 Web 日志挖掘技术的研究[D].镇江:江苏科技大学,2019.

[5]柴伟.基于信息构建理论的MOOC网站优化研究[D].郑州:郑州大学,2018.

[6]袁汉宁,王树良,程永,等.数据仓库与数据挖掘[M].北京:人民邮电出版社,2015:07.

[7]徐璐.基于Web挖掘的视频推荐系统分析与实现[D].南京:南京邮电大学,2016.

[8]韩家炜,孟小峰,王静,等.Web挖掘研究[J].计算机研究与发展,2001,04:405-414.

[9]黄昊翔.基于分布式计算平台的 Web 日志挖掘技术的研究与应用[D].北京:北京邮电大学,2018.

[10]汤效琴,戴汝源.数据挖掘中聚类分析的技术方法[J].微计算机信息,2003,01:3-4.

[11]Veselina Bureva,Stanislav Popov,Velichka Traneva,et al.Generalized Net Model of Cluster Analysis Using CLIQUE:Clustering in Quest[J].International Journal Bioautomation,2019,23(2).

[12]Wang Guiyan,Bu Changjiang,Luo Yuesheng.Modified FDP cluster algorithm and its application in protein conformation clustering analysis[J].Digital Signal Processing,2019,92.

[13]Paresh Solanki,Sanjay Garg,Hitesh Chhinkaniwala.A Comprehensive Review of Privacy Preserving Techniques in Data Mining and Data Stream Mining[J].International Journal of Data Mining And Emerging Technologies,2018,8(1).

[14]Kim Jungkee.Web Server Log Visualization[J].International journal of advanced smart convergence,2018,7(4).

[15]高伟伟.基于ELK的WEB日志安全分析平台实现及分析[D].兰州:兰州大学,2020.

[16]曾新励.基于 Hadoop 平台的分布式 web 日志分析系统的研究与实现[D].成都:西南石油大学,2017.

[17]Chen M S,Park J S,Yu P S,et al.Data mining for path traversal patterns in a Web environment[C].Proceedings of the 16th International

Conference on Distributed Computing System,Hong Kong,1996:385-392.

[18]Colaco J,Mittal J.An efficient approach to user navigation pattern prediction[C].International Conference on Advances in Electrical.IEEE, 2016.

[19]Ansari Z A,Sattar S A,Badu A V.A fuzzy neural network based framework to discover user access patterns from web log data[J].Advances in Data Analysis & Classification, 2015.

[20]Guo J,Zhang S,Qiu Z.A Kind of Improved Data Clustering Algorithm in Web Log Mining[C].2015 International Conference on Intelligent Systems Research and Mechatronics Engineering.2015.

[21]杨勇,任淑霞,冉娟,等.基于粒子群优化的 k-means 改进算法实现 Web 日志挖掘[J].计算机应用,2016，S1:29-32.

[22]章永来,周耀鉴.聚类算法综述[J].计算机应用,2019,39(07):1869-1882.

[23]Jonathan A,Silva,Elaine R,et al.Data stream clustering[J].ACM Computing Surveys(CSUR),2013,46(1).

[24]O'Callaghan L,Mishra N,Meyerson A,et al.Streaming-Data Algorithms for High-Quality Clustering[C].International Conference on Data Engineering.IEEE,2002.

[25]Aggarwal C C,Han J,Wang J,et al.A Framework for Clustering Evolving Data Streams[J].Proceedings 2003 VLDB Conference,2003:81-92.

[26]Cao F,Ester M,Qian W,et al.Density-based clustering over an evolving data stream with noise[C].Proceedings of the Sixth SIAM International Conference on Data Mining,April 20-22,2006,Bethesda,MD,USA.DBLP,2006.

[27]Chen Y,Tu L.Density-based clustering for real-time stream data[C].Acm Sigkdd International Conference on Knowledge Discovery & Data Mining. ACM,2007.

[28]刘齐.PageRank 算法在 Web 挖掘中的研究与应用[D].镇江:江苏科技大学,2020.

[29]吕亚兵.WEB 站点日志数据挖掘的研究与实现[D].武汉:武汉理工大学,2006.

[30]夏烈阳.大数据背景下基于 Web 日志的用户访问模式挖掘研究[D].昆明:云南财经大学,2019.

[31]王春玲,李川,李想.基于 Apriori 算法的高校 Web 日志挖掘系统构建[J].中国林业教育,2019,37(02):22-26.

[32]黄昊翔.基于分布式计算平台的 Web 日志挖掘技术的研究与应用[D].北京:北京邮电大学,2018.

[33]盛昀瑶.数据挖掘对 Web 访问日志的深度分析[D].上海:华东师范大学,2007.

[34]李建，曾新励.基于 Hadoop 的民航日志分析系统及应用[J].软件导刊,2017,16(01):100-103.

[35]陆勰.基于Hadoop的云安全日志分析技术研究[D].北京:北京邮电大学,2019.

[36]Guha S,Mishra N,Motwani R,et al.Clustering data streams[C].Foundations of computer science.IEEE,2000:359-366.

[37]万新贵.分布式数据流挖掘技术综述[J].微型机与应用,2016,35(21):8-10.

[38]申飞.基于数据流挖掘的入侵检测系统研究与应用[D].邯郸:河北工程大学,2019.

[39]庞景月.滑动窗口模型下的数据流自适应异常检测方法研究[D].哈尔滨:哈尔滨工业大学,2013.

[40]Mahmood Shakir Hammoodi,Frederic Stahl,Atta Badii.Real-Time Feature Selection Technique with Concept Drift Detection using Adaptive Micro-Clusters for Data Stream Mining[J].Knowledge-Based Systems,2018.

[41]窦勇,王嘉伦,苏华友,等.从计算机体系结构发展历程看数据流计算思想[J].中国科学:信息科学,2020,50(11):1697-1713.

[42]Gaber M M,Zaslavsky A,Krishnaswamy S.Mining data streams:a review[J].ACM Sigmod Record,2005,34(2):18-26.

[43]张丹丹.面向数据流挖掘的分类和聚类算法研究[D].北京:北京交通大学,2014.

[44]王少峰,韩萌,贾涛,等.数据流高效用模式挖掘综述[J].计算机应用研究,2020,37(09):2571-2578.

[45]孙玉芬,卢炎生.流数据挖掘综述[J].计算机科学,2007,34(1):1-5.

[46]张政淇.面向流数据的聚类算法改进及其服务化实现[D].北京:北方工业大学,2020.

[47]Arian Baer,Pedro Casas,Alessandro D’Alconzo,et al.DBStream:A holistic approach to large-scale network traffic monitoring and analysis[J].Computer Networks,2016,107.

[48]王琳琳.基于 Storm 的流数据聚类算法的研究与实现[D].济南:齐鲁工业大学,2019.

[49]龚芳海.大数据分析下分布式数据流处理技术研究[J].电子元器件与信息技术,2020,4(02):92-93.

[50]鲁亮,于炯,卞琛,等.大数据流式计算框架 Storm 的任务迁移策略[J].计算机研究与发展,2018,55(01):71-92.

[51]陈剑.基于Spark计算的实时数据分析的应用研究[D].青海:青海师范大学,2020.

[52]钱石磊.分布式流数据处理平台基准评测与性能优化[D].上海:上海交通大学,2017.

[53]蔡鲲鹏,马莉娟.基于Flink on YARN平台的应用研究[J].科技创新与应用,2020,16:173-175+178.

[54]戚红雨.流式处理框架发展综述[J].信息化研究,2019,45(06):1-8.

[55]杜小勇,卢卫,张峰.大数据管理系统的历史、现状与未来[J].软件学报,2019,30(01):127-141.

[56]王铭坤,袁少光,朱永利,等.基于 Storm 的海量数据实时聚类[J].计算机应用,2014,34(11):3078-3081.

[57]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[J].软件学报,2014,25(4):839-862.

[58]杨善红,梁金明,李静雯.基于网格密度影响因子的多密度聚类算法[J].计算机应用研究,2015,32(3):743-747.

[59]马可.基于 Storm 的流数据聚类挖掘算法的研究[D].南京:南京邮电大学,2016.

[60]Lewis B W.rredis:"Redis" Key/Value Database Client[J].2015.

[61]尹世庄,王韬,谢方方,等.基于互信息和轮廓系数的聚类结果评估方法[J].兵器装备工程学报,2020,41(08):207-213.

[62]王冬.基于跨域行为分析的内部威胁检测研究[D].重庆:重庆邮电大学,2018.

[63]赵星.Web 漏洞挖掘与安全防护研究[D].太原:中北大学,2016.

[64]李川.基于 Apriori 算法的高校 Web 日志挖掘系统研建[D].北京:北京林业大学,2018.

在学期间取得的科研成果和科研情况说明

取得的科研成果：

[1] 第一作者.改进的密度网格流聚类算法应用于 Web 访问日志分析.计算机应用研究.增刊录用

[2] 第一作者.基于模糊聚类的 Web 访问日志分析识别系统 V1.0，授权号：2020SR0851373（软件著作权）

致　　谢

流光瞬息，旦夕之间两年半的研究生生活就要结束了，在这两年半的时光里既有努力科研的艰辛与辛苦，也有收获的成长与喜悦，这两年半的时间必将成为我一生的纪念。行文至此，我想向所有在这段时间曾与我有过共同经历的人表达最诚挚的感谢。

首先，由衷的感谢我的导师王法玉老师，感谢您在学习中的督促与指导，感谢您在生活中的关怀与指引。在论文的完成过程中，从论文开始的选题、实验平台的搭建到最终论文的撰写，王老师给了我许多的帮助和意见。学贵得师，亦贵得友，王老师对待科研和生活的态度，时刻影响和指导着我。

其次，感谢班级同学和实验室同学，感谢大家在生活上和学习上提供的帮助。

最后，感谢我的家人和朋友，感谢你们的陪伴和鼓励。谢谢！