文章编号：0253-374X(2020)02-0276-11

DOI：10.11908/j.issn.0253-374x.19054

针对具有稀疏性的流式大数据卸载方法

王　顺[1,3]，李振星[2]，连增申[2]，曾国荪[1,3]，丁春玲[4]

（1. 同济大学 电子与信息工程学院，上海 200092；2. 北京捷软世纪信息技术有限公司，北京 100085；3. 同济大学 国家高性能计算机工程技术中心同济分中心，上海 201804；4. 同济大学 化学科学与工程学院，上海 200092）

摘要：在保证实时性的前提下提高流式大数据卸载的准确性是一个重要问题。针对具有稀疏性的流式大数据开展2种典型场景下的卸载研究。对普通均匀业务的流式大数据进行空间建模，使用弹性距离对数据间的距离进行放缩，提出基于离心率的卸载方法。对异常检测业务流式大数据应用场景进行特征分析，使用预处理自动机对数据的动态处理过程进行描述，在综合考虑数据和处理行为相似度基础上，提出基于等价类划分的卸载方法。重复试验表明，所提出的卸载方法与传统卸载方法相比能明显提高卸载的有效性。

关键词：流式大数据；数据卸载；稀疏性；弹性距离；行为相似

中图分类号：TP338　　**文献标志码**：A

Load Shedding Methods for Big Data Stream with Sparsity

WANG Shun[1,3], *LI Zhenxing*[2], *LIAN Zengshen*[2], *ZENG Guosun*[1,3], *DING Chunling*[4]

(1. College of Electronics and Information Engineering, Tongji University, Shanghai 200092, China; 2. Beijing AgileCentury Information Technology Co., Ltd., 100085, China; 3. Tongji Branch National Engineering & Technology Center of High Performance Computer, Tongji University, Shanghai 201804, China; 4. College of Chemical Science and Engineering, Tongji University, Shanghai 200092, China)

Abstract: How to improve the accuracy of load shedding under the premise of ensuring real-time performance is an important problem. Sparsity is a widespread feature of the big data stream. Therefore, we propose two load-shedding methods of the big data stream with sparsity in two scenarios. In the normal business scenario, we model the big data stream with the high dimensional space. Then we propose a load shedding method based on centrifugation, which uses the elastic distance to measure the distance of data. In the anomaly-monitoring scenario, we analyze the feature of the big data stream and propose a load shedding method based on equivalence class, which uses the combined similarity to divide the data set into equivalence classes. The combined similarity was composed of processing behavior similarity and data similarity to measure the difference between data. Repeated test results show that the two load shedding methods in this paper can significantly improve the accuracy compared with the conventional load shedding methods.

Key words: big data stream; load shedding; sparsity; elastic distance; behavior similarity

随着大数据技术的深入发展，传统的批量计算越来越难以满足大数据处理的及时性要求。与此同时，以海量数据持续到达、在线计算、实时响应为计算模式的应用变得越来越广泛。一般地，持续到达的数据称为流数据，对流数据的计算称为流式计算，简称流计算。在短时间内产生海量的流数据，称为流式大数据，也称为流大数据。在网络数据监控、金融分析、社交网络、互联网用户行为在线分析、交通监控等领域中都需要对流大数据进行实时处理和响应。由于流大数据是在线到达，且到达速度和数据特征是不确定的，这就使得流大数据的实时流量往往不可预知，从而可能导致因瞬时流量增大，无法被系统及时处理的过载现象。一旦发生严重过载，会导致计算系统性能的急剧下降。因此，如何解决过载问题，是流大数据处理的一个重要问题。

为了加快流大数据的处理速度，解决其中的过

收稿日期：2019-02-21

基金项目：国家社科基金(17BQT086)；国家海底科学观测系统子项目(2970000001/001/016)；CCF 信息系统开放课题(CCFIS2018-01-03)

第一作者：王　顺(1989—)，男，博士生，主要研究方向为流式大数据计算. E-mail：wangshunflyh@126.com

通信作者：曾国荪(1964—)，男，教授，博士生导师，主要研究方向为并行分布计算、大数据处理、信息安全. E-mail：gszeng@tongji.edu.cn

论文拓展介绍

载问题,研究人员从计算任务、体系结构、数据本身等多方面进行了大量研究。在计算任务方面,在云计算环境中通过优化任务调度提高资源利用效率。在体系结构方面,通过对计算系统的横向或者纵向扩展,提高计算能力。在数据方面,主要是通过对数据本身进行近似处理,以便减少不必要的计算,从而加快计算速度。尽管通过任务-资源优化调度可以提高资源利用效率,从而加快计算速度,但硬件计算能力总会有极限,因此不能从根本上解决过载问题。并行分布式计算技术的发展提高了大数据计算的处理速度,但随着节点的增加通信开销也会相应增加,所以并不一定能够提高计算速度,且在实际应用中增加计算资源不仅会带来巨大的成本开销,而且会造成计算资源的浪费。

卸载技术[1-3]是其中的典型方法,即当系统的处理能力小于当前的计算负载时,通过抛弃一部分数据来降低系统负载。在实际应用环境中,一方面因噪声、异常、重复、冗余等因素存在,会产生大量无用数据。另一方面,在流计算中往往只有部分数据是计算任务所需的数据,例如在物联网监测预警应用中,较少出现的数据是重要数据,在正常范围内则可能产生大量重复或无用的数据。因此,通过丢弃部分无用数据,从而加快处理速度的卸载技术是解决资源有限条件下流大数据过载问题的有效方法。

自流计算出现以来,研究人员已对卸载方法进行大量研究,这些卸载方法主要分为2类,分别是随机卸载和语义卸载。

随机卸载是以随机的方式丢弃部分数据来加快整个流大数据的处理速度。Babcock等[4]提出以置信度为前提的随机卸载策略。Tatbul等[5]提出了向流大数据查询任务中插入或移除随机卸载操作符的方法来处理负载波动。闫莺等[6]提出改进的随机卸载方法:基于概率的均匀降载策略和小窗口准确降载策略,并结合查询任务调度和卸载提高流大数据处理效率。随机卸载方法虽然提高了流计算系统的吞吐量,显著加快了流大数据的整体处理速度,但是随机卸载可能会降低计算精度和服务质量。

语义卸载是在一定条件下根据数据在应用中的重要程度来选择要卸载数据的方法。由于流大数据产生方式不同,数据结构多样,应用场景各异,因此出现许多具体方法。文献[7]针对固定结构的元组流大数据,提出基于频率的卸载方法,该方法根据数据出现的频率判断数据重要程度,通过在内存中维护一个T-tree数据字典,卸载出现频率较低的数据,实验结果表明该方法能支持高速率到达的流大数据卸载。Zhang等[8]提出一种基于反馈控制的自适应卸载策略,首先对流大数据建立一个包括负载监视器、反馈控制器、卸载器等模块的卸载系统,然后使用根轨迹法设计反馈控制器,以便接收来自监视器的反馈信息,进行动态自适应卸载。文献[9]提出基于执行开销和数值分布优先级的卸载算法,首先将数据在其值域上划分成若干区域,然后根据不同区域数据在多个串联或并联操作符上的执行路径计算出执行开销,在内存中维护数值和开销二维表,根据数据优先级进行卸载。Maison等[10]提出基于预测的流大数据卸载框架,通过对高斯分布的流大数据进行流量和数值分布的预测,结合不同的查询操作符,给出相应丢弃最大、最小等的不同的卸载策略,实验表明该方法能有效提高服从高斯分布数据源的卸载精度和数据吞吐量。文献[11]提出基于直方图的卸载方法,该方法针对缓存的数据构建一种塔形矩阵的数据存储结构,每桶提取一个代表数据并删除该桶中其余数据,将每桶的代表数据组成新的数据参与任务处理,该方法在一定程度上提高卸载精度,但海量数据处理时存在性能不高的问题。文献[12]针对一段时间内接收的以资源描述框架(RDF)表示的流大数据,根据数据间的图结构关联关系,分析其语义信息从而卸载语义上的无效数据,但该方法仅适用于数据间关联关系已知的流大数据卸载。

目前,学者们针对多种场景对流大数据卸载技术进行了大量的研究工作,包括基于数据出现频率[7]、任务反馈控制[8,13]、维护优先级表或树[9,14]、流大数据到达规律预测[10]、数据关联图[12]等方面。尽管有些学者已提出基于离群点和基于任务内容的语义卸载方法,然而尚未发现对数据距离进行动态弹性度量的研究工作,也未发现对数据处理行为等价度量的深入研究工作。为此,本文针对2种典型的应用场景进行分析和建模,使用弹性距离机制提出基于离心率的卸载方法,使用数据处理行为相似性提出基于等价类划分的卸载方法。

1　流大数据及其稀疏性

1.1　流大数据处理的一般过程

根据流大数据的处理过程,一般可以将流大数据的生命周期分为数据到达、数据预处理、任务计算、应用挖掘、事后存储等多个阶段,如图1所示。

在数据到达阶段,数据源可分为单源、多源。从

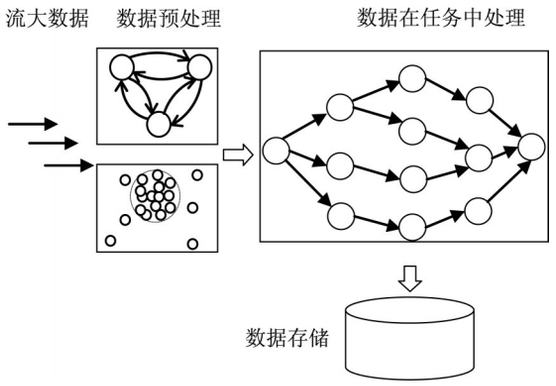

图1 流大数据处理的一般过程

Fig.1 The general process of big data stream processing

数据结构上来讲,可分为固定结构和可变结构流大数据。从数据之间的关联关系上来讲,可分为无关联流大数据和有关联流大数据,例如社交网络流是图结构流大数据,而定点监测流大数据则是无关联的。从流大数据的到达方式上来讲,可分为稳定到达、不确定到达。目前,流计算尚无对所有形态流大数据的统一计算模型。因此,不同的数据到达方式,决定着流大数据的不同形态,后续数据预处理、任务计算、应用挖掘等方法也各不相同。本文假设所讨论的流大数据,以元组为数据的最小单位,它是相互无关联的、结构相同的数据实体,由现实应用场合源源不断地产生,并依次交由流大数据计算系统处理。

定义1 流数据。指随时间变化,持续产生的大量无关联、固定结构的元组数据序列。假设一条流数据记为$D=\{d_1, d_2, \cdots, d_n\}$,其中$d_i$表示第$i$个元组,每个元组都有$k$个属性组成,$d_i$带属性的记法为$d_i(a_{i,1}, a_{i,2}, \cdots, a_{i,k})$,其中$a_{i,j}$表示第$i$个数据的第$j$个属性,$1\leqslant j\leqslant k$。当流数据在短时内海量到达时称为流大数据。

根据应用场合的实际含义,元组数据的各个属性往往具有不同的数据类型和取值范围,为了后续处理方便,采用极差标准化方法对到达的数据进行预处理。对于元组$d_i(a_{i,1}, a_{i,2}, \cdots, a_{i,k})$,令$a_{i,j}$标准化后的属性$a_{i,j}'=\dfrac{a_{i,j}-\min(a_{i,j})}{\max(a_{i,j})-\min(a_{i,j})}$,因此每个属性取值范围为[0,1],实现所有数据的归一化操作。

如图1所示,在流大数据处理的整个过程中,数据预处理阶段主要任务是对实时到达的流大数据进行在线检测、清洗、去噪、补全和卸载等预处理,以便保证流计算系统的正常运行,并且为后续流大数据的处理做准备工作。本文就是在该阶段找出使用价值较低的数据并卸载。任务计算和应用挖掘阶段则是对上一阶段输出的数据进行流计算任务处理。根据不同的应用场景,流大数据系统所执行的计算任务也各不相同。为不失一般性,如图1所示,用无环有向图(directed acyclic graph,DAG)来描述流大数据挖掘处理的计算任务。海量流大数据在DAG任务处理下,任务程序的动态执行路径可能十分复杂,不同的执行路径其开销也可能不同,因此数据的功能行为也是对数据进行分类的一个有效方法。为此研究基于预处理自动机的行为相似性度量方法。

本文流大数据处理模式为每隔一个固定的时间周期T_0对到达的数据实施集中卸载,这似乎和流大数据的实时处理的要求不相符,但事实上实时处理是相对的,只要T_0设置足够小,应该能够达到流大数据及时处理的目标。假设计算负载和数据量正相关,数据量越大,则计算负载也越大。如果过载,要求在数据处理前卸载一定数量的数据即可。

1.2 流大数据的稀疏性

定义2 流大数据的稀疏性。流大数据的稀疏性指流大数据具有2个方面的特性,一是数据的各个属性值呈现出稠密不均的非均匀分布的特性;二是流大数据的流量随时间不断变化。

定义3 数据距离。把数据看作空间中的点,2个点间的距离称为数据距离。假设2个数据为$d_1(a_{1,1}, a_{1,2}, \cdots, a_{1,k})$和$d_2(a_{2,1}, a_{2,2}, \cdots, a_{2,k})$,则这2个数据间距离为$l(d_1, d_2)=\sqrt{\sum_{j=1}^{k}(a_{1,j}-a_{2,j})^2}$。

定义4 数据半径。在一个大数据集合中,所有数据点到中心点的平均距离称为半径。假设大数据集为$D=\{d_1, d_2, \cdots, d_n\}$,$D$的中心点为$d_c(a_{c,1}, a_{c,2}, \cdots, a_{c,k})$,其中$a_{c,j}=1/n\sum_{i=1}^{n}a_{i,j}$,则半径$r=1/n\sum_{i=1}^{n}l(d_i, d_c)$。

定义5 数据离心率。在一个大数据集合中,单个数据点偏离中心数据点的程度称为该数据点的离心率。假设大数据集为$D=\{d_1, d_2, \cdots, d_n\}$,$D$的中心点为$d_c(a_{c,1}, a_{c,2}, \cdots, a_{c,k})$,半径为$r$,$D$中任意一点$d_i$到$d_c$的距离为$l(d_i, d_c)$,那么$d_i$的离心率$\xi(d_i, d_c)=\dfrac{l(d_i, d_c)}{r+l(d_i, d_c)}$。

定义6 数据差异度。指某数据在数据集合中的离散程度,用该数据到中心点数据的距离与中心点数据的模的比值表示。设数据集为$D=\{d_1, d_2, \cdots, d_n\}$,$d_c$为数据集$D$上的中心点,则数据$d_i\in D$的差异

度为$\varphi(d_i,d_c)=\left|\frac{d_i-d_c}{d_c}\right|$,$i=1,2,\cdots,n$。

2　流大数据卸载方法

流大数据分布稀疏性的现象是客观存在的,这也造成每个数据的价值各不相同。显然,低价值数据的处理将占用不必要的计算资源。为了应对流大数据稀疏性中的流量波动特征导致的过载问题,针对2种应用场景进行分析和建模,并针对这2种场景提出相应的卸载算法,旨在保证计算系统实时性前提下提高卸载准确性,提高服务质量。

2.1　普通均匀业务流大数据分析应用场景

在许多普通应用中,数据源产生的数据呈现高斯分布,即数据在靠近聚集中心的周围出现概率较大,分布较密集;离聚集中心远的数据出现概率较小,分布较稀疏。如电子商务交易中,希望在线分析用户购买偏好,大部分正常业务数据的消费金额、购买数量等数据都在某个区间内,但也有一小部分客户会购买金额、数量、种类等极大或者极小的情况存在。在这类应用中,那些分布集中的数据是重要的数据,那些稀疏的离群数据往往被认为是无用数据。因此,流计算时,卸载应该丢弃掉那些离群数据。

据定义5,计算离心率的关键是度量2个点之间的距离,不同的距离计算方法对离心率计算的准确性有直接影响,其中最典型的是欧氏距离计算方法。由于数据处于临界边缘时,使用传统的距离计算方法无法明显区分异常的离群点和正常的聚集点。为此提出弹性距离的计算方法:对于那些距离中心点越近的数据,缩小它们的差异,使它们具有更明显的聚集特征;而对于那些距离中心点越远的数据,放大它们的差异,使它们具有更明显的稀疏和离群特征,从而更有利于对异常的离群点进行甄别。

定义7　弹性距离。指数据点到中心点的距离进行缩放后的距离。假设数据集的中心为$d_c(a_{c,1},a_{c,2},\cdots,a_{c,k})$,对任意数据点$d_i(a_{i,1},a_{i,2},\cdots,a_{i,k})$与$d_c$的距离进行弹性度量,记$d_i$与$d_c$的弹性距离为$L(d_i,d_c)$,则$L(d_i,d_c)=f(d_i,d_c)\times l(d_i,d_c)$。其中,$f(d_i,d_c)$表示以$d_c$为中心点的弹性系数,令$f(d_i,d_c)=e^{\alpha(l(d_i,d_c))}+\beta$,它是以欧氏距离为自变量的指数函数。由于指数函数的单调递增特性,当d_i与d_c的距离越近时,弹性系数$f(d_i,d_c)$越小。当d_i与d_c的距离越远时,弹性系数越大。通过将弹性系数与欧氏距离相乘,对点d_i与d_c的距离进行弹性度量,用于对实时到达流大数据的离心率进行计算,从而对远偏离的数据进行卸载,显然更具有合理性。

根据普通均匀业务的流大数据分析应用场景的数据特点可知,在固定时间周期T_0内,到达的数据为内部稠密、外部稀疏的"超球"分布,卸载时应丢弃偏离中心点较远的数据。根据定义5,以T_0周期内到达的所有数据为点集、以该点集的平均值作为中心点、以所有点到中心的平均距离为半径计算每个数据的离心率,并根据离心率预先设定的阈值决定该数据是否被卸载。但是,由于流大数据的海量性,可能有大量数据的离心率处于阈值边缘。为了更好地区分稀疏和稠密数据,尤其是在阈值附近的数据,使用弹性距离计算每个数据的离心率。

定义8　弹性离心率。指以弹性距离为数据间测度的离心率。已知在T_0内到达的流数据为$D=\{d_1,d_2,\cdots d_n\}$,d_i表示D中第i个数据,$d_c(a_{c,1},a_{c,2},\cdots,a_{c,k})$为$D$的中心点,$L(d_i,d_c)$表示$d_i$与中心点的弹性距离,记$d_i$的弹性离心率为$\psi(d_i,d_c)$,则

$$\psi(d_i,d_c)=\frac{L(d_i,d_c)}{r+L(d_i,d_c)}=\frac{1}{\frac{r}{L(d_i,d_c)}+1}\quad(1)$$

由定义8可知,离心率值域为[0,1),弹性系数$f(d_i,d_c)=e^{\alpha(l(d_i,d_c))}+\beta$值域为$[1+\beta,e^{\alpha}+\beta)$,当$l(d_i,d_c)=1/\alpha\times\ln(1-\beta)$为弹性缩放的分界点,其中参数$\alpha$、$\beta$由实际应用需求确定。同时,根据实验和历史经验,给定该T_0时间周期内的离心率分割阈值ε_0,那么离心率大于阈值ε_0视为无用点全部抛弃。根据上述过程,基于离心率卸载算法的伪代码如下。

算法1 基于离心率的卸载算法(CBLS),输入:$D=\{d_1,d_2,\cdots d_n\}$,ε_0,其中D为预定周期T_0到达的流大数据,ε_0为卸载阈值;输出:D',其中D'为D中要卸载的数据。

```
CBLS()
    1. { D'←∅, r←0; // r为半径
    2. d_c←find_center_point(D);
    //d_c为中心点
    3. for ∀d_i∈D
    4. { dis(d_i,d_c)←get_distance(d_i,d_c);
    5. df_i←(e^(α(dis(d_i,d_c)))+β)×dis(d_i,d_c);
    // df_i为d_i和中心点的弹性距离
    6. r←r+dis(d_i,d_c);
    7 }
    8. r←r/n;
```

9. while ($D\neq\varnothing$)

10. $\{d_i\leftarrow$get_one_data(D) ; //任取一个元素

11. $D\leftarrow D-\{d_j\}$;

12. $\psi(d_i,d_c)\leftarrow\dfrac{1}{\dfrac{r}{L(d_i,d_c)}+1}$;

//根据式(1)计算每个数据的弹性离心率

13 if ($\psi(d_i,d_c)>\varepsilon_0$) //卸载离心率大于$\varepsilon_0$

14. $D'\leftarrow D'+\{d_i\}$;

15. }

16. return D';

17. }

其中1、2行是初始化并计算中心点,3~8行是计算半径r,第12行是计算每个数据的弹性离心率。13~14行根据弹性离心率的大小判断是否需要卸载。其中,时间复杂度的关键在于弹性离心率的计算。D中数据点的个数为n,整个算法求中心点和半径分别需要n次计算,之后每个数据都只需根据公式计算一次弹性离心率,因此算法1的时间复杂度为$O(n)$。

该算法是对一个T_0时间周期到达的数据进行卸载处理。对于持续到达的流大数据,可以通过计时,每隔T_0周期,循环调用该算法,即可实现对该应用场景下流大数据的持续卸载处理。

2.2 异常检测业务流大数据应用场景

在异常检测类应用场景中,用户主要对那些较少出现的数据特别关注,而对经常出现的数据则不太关心。例如,公安犯罪监测、海关信息监测等应用中,违法信息往往在数额、数量、频率等属性上与通常数据明显不同,因此这些数据属于重要数据。又如,在有些物联网应用中,一旦检测到不常见的温度、湿度或者异常的化学物质等信息,应及时作出反应。因此,在这类应用中,当流数据新到达时,应该特别关注异常数据。

2.2.1 流大数据的处理过程和行为

数据的作用和价值往往通过分析和处理来获得。数据分析和处理往往具有复杂的计算过程,一般包含多个计算子任务,它们互为因果,交互作用,它们的动态行为可抽象为有限状态自动机[15]。不同的输入数据,在检测预处理任务程序中可能产生不同的状态转移过程,即不同类别的输入数据在分析处理时,对应的任务程序有不同处理过程或者行为。每个数据的处理过程或者行为体现了数据的价值和功能,因而记录并比较每个数据的动态处理行为,可以判断出每个数据在该任务下的功能相似度。首先使用行为自动机对流大数据的检测预处理过程进行建模,并根据数据预处理的行为进行等价类划分,为流大数据卸载提供决策依据。将流大数据检测预处理任务的计算过程抽象为一个5元组,即行为自动机$M=<S,S_0,C,E,F>$,其中S是有穷状态集合;S_0是初始状态集;C是状态转移约束和开销权重的集合,用$<c_i,w_i>$表示;$E=S\times(\Phi(C)\times 2^C\times S)$是状态转换关系的集合,边$e=<s,a,\delta,s',w>$表示当输入为$a$时从状态$s$到$s'$的转换,$\delta$表示一个条件约束,$w$表示权重;$F$是终止状态集合。如图2所示给出了一个举例。

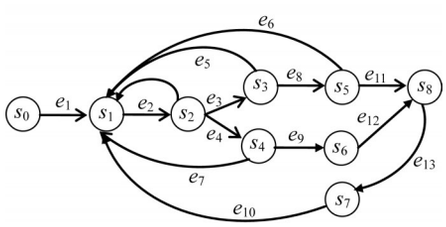

图2 任务预处理自动机

Fig.2 Task automata machine

定义9 数据处理行为。假设流大数据处理的过程用自动机$M=<S,S_0,C,E,F>$描述,那么数据d_i的处理行为是指d_i在自动机M中的状态转移路径,由状态节点和边的交替序组成,记为$P_i=s_0e_x\cdots s_ve_y\cdots e_zs_{if}$,其中$s_0$为初始状态,$s_0\in S_0$,$s_{if}$为结束状态且$s_{if}\in F$,$s_v$和$e_y$为$P_i$的中间状态,且$s_v\in S$,$e_y\in E$。

例如,如图2所示,该自动机有9个状态,数据处理的初始状态为s_0,结束状态为s_8,s_2~s_7为中间状态,s_0到s_8之间的不同路径对应不同数据的处理行为。特别地,假设图2箭头路径为数据d_1的处理过程,则d_1的处理行为$P_1=s_0e_1s_1e_2s_2e_4s_4e_7s_1e_2s_2e_3s_3e_8s_5e_{11}s_8$。

为了得到数据处理行为序列,构建预处理自动机程序。首先,对数据任务进行抽象,将一个流大数据处理任务划分成多个子任务,根据子任务之间的相互关系构建任务DAG图。然后,根据DAG任务每个节点的状态转移条件编写预处理自动机,记录每个数据到达时的状态转移路径,此即为通过预处理获取数据处理行为的基本方法。

2.2.2 流大数据处理行为相似性度量

一般地,如果2个处理行为的开始状态和结束状态相同,且中间处理过程也存在一定程度的相同,那么我们认为这2个处理行为具有相似性。

定义10 行为相似。设数据d_i的处理行为$P_i=$

$s_{i0}e_{iu}\cdots e_xs_x\cdots e_{iv}s_{if}$,数据$d_j$的处理行为$P_j=s_{j0}e_{ju}\cdots e_ys_y\cdots e_{jv}s_{jf}$,其中$s_{i0}$和$s_{j0}$分别为$P_i$和$P_j$的起始节点,$s_{if}$和$s_{jf}$分别为$P_i$和$P_j$的终止节点。如果有$s_{i0}=s_{j0}$,$s_i=s_{jf}$,且$P_i$的边集$E_i$与$P_j$的边集$E_j$有$E_i\cap E_j\neq\varnothing$,或$P_i$的状态集$S_i$与$P_j$的状态集$S_j$有$S_i\cap S_j\neq\varnothing$,则称数据$d_i$与数据$d_j$处理行为相似。

如果2个数据存在处理行为相似,那么它们处理过程的状态转移路径有3种情形:①完全相同;②有若干交点,但不存在循环;③有若干交点,且存在循环。如果将每次循环的边都看作不同的边,那么无论在某个节点或多个节点之间是否存在循环,它们可以统一归纳为情形②。例如,假设2个数据处理行为分别为$P_i=s_0e_1s_1e_3s_3e_5s_4e_6s_5e_8s_8$和$P_j=s_0e_2s_2e_4s_3e_5s_4e_7s_6e_9s_7e_{10}s_8$,如图3所示,除了初始节点$s_0$和结束节点$s_8$之外有2个交点$s_3$、$s_4$。由于$P_i$与$P_j$节点和边个数可能不同,无法一一对应比较,因此根据处理行为交点的个数将P_i与P_j分为若干个相似阶段,分别从节点和边的相似程度来度量。

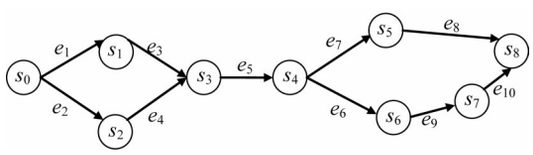

图3　数据处理行为相似的情况

Fig.3　Similarity of data processing behavior

设$S_i\cap S_j=S_c$,且$|S_c|=m$,则P_i与P_j可分为$m+1$个相似阶段。设P_i的第p个相似阶段有$M_{i,p}$个节点。P_i的任意节点s_r的后继节点记为$V(P_i,s_r)$,若$V(P_i,s_r)\notin P_j$,令$V(P_i,s_r)=1$;如果$V(P_i,s_r)\in P_j$,令$V(P_i,s_r)=0$;路径P_i中S_r节点的后继边记为$E(P_i,s_r)$,且令$E(P_i,s_r)=w_r$,其中w_r为$E(P_i,s_r)$的开销权重。记数据d_i、d_j处理行为P_i与P_j相似度为$B(d_i,d_j)$,则

$$B(d_i,d_j)=1/m\cdot\sum_{p=1}^{m}\left(\sigma\frac{2}{\sum_{r=1}^{M_{i,p}}V(P_i,s_r)+\sum_{t=1}^{M_{j,p}}V(P_j,s_t)+2}+\rho\frac{\min\left\{\sum_{r=1}^{M_{i,p}}E(P_i,s_r)\times w_r,\sum_{t=1}^{M_{j,p}}E(P_j,s_t)\times w_t\right\}}{\max\left\{\sum_{r=1}^{M_{i,p}}E(P_i,s_r)\times w_r,\sum_{t=1}^{M_{j,p}}E(P_j,s_t)\times w_t\right\}}\right)$$

其中σ、ρ分别表示顶点和边在实际应用中开销的重要程度,且$\sigma+\rho=1$。由$B(d_i,d_j)$的计算式可知其取值范围为[0,1]。当P_i和P_j完全相同时,$B(d_i,d_j)=1$,当d_i和d_j不存在行为相似即$E_i\cap E_j=\varnothing$,或者$S_i\cap S_j=\varnothing$时,$B(d_i,d_j)=0$。

2.2.3 流大数据综合相似度计算

2个数据处理行为的相似度反映了2个数据的价值的相似程度。但是,即使具有相同的处理行为,因数据本身的差异,在进行异常数据分析检测时,不同数据应该区别对待。为此,在行为相似的基础上,结合纯数据的相似性,进行综合相似性度量。根据文献[16]的相似性定义,2个数据d_1、d_2的纯数据相似度表示$F(d_1,d_2)=1/(1+l(d_1,d_2))$,其中$l(d_1,d_2)$为数据$d_1$、$d_2$间的距离,参见定义3。数据$d_1$、$d_2$的综合相似度用纯数据相似度与数据处理行为相似度的积表示,记为$H(d_1,d_2)$,则

$$H(d_1,d_2)=F(d_1,d_2)\times B(d_i,d_j)\tag{2}$$

显然,$H(d_1,d_2)$是取值为[0,1]内的实数,取值越大,表示数据d_1和d_2在处理过程和数值上具有越高的相似程度。反之,表示d_1和d_2在处理过程和数值上差异越大。当$H(d_1,d_2)=1$时表示这2个数据完全相同。该公式综合考虑了数据的纯数据相似性与功能相似性,分别从静态和动态2个方面来度量数据间的相似性,为后文等价类划分提供了依据。

定义11　数据等价类。在一个大数据集合中,如果2个数据的综合相似度大于某个给定的阈值,则称它们之间具有等价关系。满足上述等价关系的数据构成的子集称为一个数据等价类。

2.2.4 基于等价类划分的卸载算法

基于等价类划分的卸载方法的主要思想是:在时间周期T_0内到达的数据中,找出差异度最大的那个数据,以此作为基点,并且根据数据综合相似度划分成2个等价类,一个是异常等价类,另一个是正常等价类。在异常检测业务流大数据应用场景下,特别需要关注异常数据,理应保留异常等价类,因此可以卸载正常等价类。同时,根据实验和历史经验,可以设定一个综合相似度的阈值η_0,那么当流大数据中的单个数据的综合相似度大于阈值η_0时,归为异常等价类。反之,归为正常等价类。根据上述算法思想,下面给出基于等价类划分卸载算法的伪代码。

算法2 基于等价类划分的卸载算法(equivalence based load shedding,EBLS)输入:$D=\{d_1,d_2,\cdots,d_n\}$,η_0,其中D为流大数据,η_0为综合相似度阈值。输出:D',D''。其中D'为异常等价类,D''为正常等价类,即要卸载的数据。

```
EBLS()
1. {D'←∅, D''←∅;
//初始化
2. d_c←find_center_point(D); //求中心点
3. forall ∀d_i∈D
```

// 计算每个数据的差异度

4. $\varphi(d_i,d_c)\leftarrow\left|\frac{d_i-d_c}{d_c}\right|$;

//求数据差异度

//求d^*为差异度最大的数据

5. $d^*\leftarrow arg_{d_i}max\{\varphi(d_1),\varphi(d_2),\ldots,\varphi(d_n)\}$;

//计算每个数据与d^*的综合相似度

6. while($D\neq\varnothing$)

7. {$d_i\leftarrow$get_one_data(D); //逐个取D中数据

8. $D\leftarrow D-\{d_j\}$;

9. $H(d_j,d^*)\leftarrow$calculate_simulation(d_i,d_j); //根据式(2)计算d_j与d^*的综合相似度

10. if($H(dj,d^*)>\eta_0$)

11. $D'\leftarrow D'+\{d_j\}$;

//异常等价类

12. else

13. $D''\leftarrow D''+\{d_j\}$;

//正常等价类

14. }

15. return D', D'';

16. }

其中,第1行是初始化,第2~4行是计算每个数据的差异度,第5行是求有最大差异度的数据d^*,第7~13行是计算每个数据与d^*的相似度,并根据阈值η_0判断等价类。第2、4、5行均需要计算n次,6~14行为一层循环,也为n次,因此算法的时间复杂度为$O(n)$。

EBLS算法是对单个T_0时间周期到达的数据进行的处理,应将正常等价类D''卸载。对于持续到达的流大数据,通过计时,每隔T_0周期循环调用该算法,即可实现对异常监测应用场景下流大数据持续的卸载处理。

3 实验分析

为了分析所提出的卸载算法的有效性,设计了一系列对比实验。实验使用的计算机为Dell T330,内含Intel Xeon E3-1220 V5CPU @ 3.0GHz,16.00 GB内存,2TB主硬盘,CentOS 6.5 (64位)操作系统。算法CBLS、EBLS以及其他传统卸载算法均采用Storm框架+JavaSE8.0编程实现。由于算法在重复执行时,每次结果可能会有微小差异,因此每组实验重复进行5次,结果取其平均值分别进行算法有效性评价。

实验采用Storm流大数据处理框架,实现流大数据的产生、到达、卸载处理。通过编写读取程序,实现将静态数据集转换为不确定方式到达的动态流大数据,经标准化后将数据发送到卸载处理程序。实验共分为2组,分别利用算法CBLS和算法EBLS,与传统的直接卸载、随机卸载、基于频率的卸载[7]方法进行比较。所谓直接卸载,是指当到达的数据个数超过系统所能处理的极限时直接卸载后续新到达的过载数据。所谓随机卸载,是指当到达的流大数据个数超过系统所能处理的极限时随机选择一批数据丢弃的卸载方法。基于频率的卸载方法的关键在于它构造了基于内存的T-tree数据结构记录数据的频率,根据数据出现的频率选择卸载数据的方法。对其加以改造,与本文其他对比算法相同,每个固定时间T_0进行一次卸载。实验算法均通过对Storm框架中任务节点过载处理逻辑改造实现。为了评估数据卸载效果,最常用的数据质量的评价标准是平方距离和(SSQ),为了便于比较,这里把平均平方距离和作为数据卸载质量的度量指标。

3.1 普通均匀业务分析流大数据应用实验

实验使用Yahoo时间序列数据集[17]以及人工合成数据,2种数据按1:1混合作为实验数据。人工合成数据按照文献[18]生成服从高斯分布的数据。获取了672 000条数据,经过处理后每个元数据含8个属性,经过极差标准化方法对数据进行规范化处理后,所有属性取值映射到[0,1]区间。根据应用场景的要求,模拟该数据集大部分数据相对均匀集中,小部分数据较为稀疏,符合普通均匀业务应用场景。

为了对比4种算法的卸载效果,实验利用4种卸载算法都处理同一个数据集,分别卸载不同比例的数据,统计卸载后数据的平均SSQ。实验中,卸载比例从5%个增加到25%。其中,直接卸载算法选择最后到达的那部分数据卸载;随机卸载算法从数据集中随机选择相应比例数据卸载;基于频率的卸载算法根据比例卸载T-tree上频率较低的数据;本文CBLS算法根据经验确定相应卸载比例下的离心率阈值,从而进行卸载处理。利用统计程序分别计算在使用每个算法处理后的平均SSQ。

为了直观体现4种算法的卸载效果,特意选取当卸载比例为5%时采用4种方法处理后的数据,取前2维作数据分布示意图。如图4所示,其中灰色星号点为要卸载的点,黑色点为保留点。图4为直接卸载、随机卸载、基于频率卸载和CBLS算法的卸载

示意图。其中,图 4a、图 4b 灰色点和黑色点混在一起,说明这 2 个算法无法区分哪些是离群点。图 4c 的周围为灰色,而中间为黑色,说明基于频率卸载能够较好理离群数据,但边界不清。图 4d CBLS 算法能够比直接卸载和随机卸载算法更明显区分离心点。这从直观上说明 CBLS 算法有更高的卸载准确性。

为了更加客观地对比 4 种算法在卸载准确性上

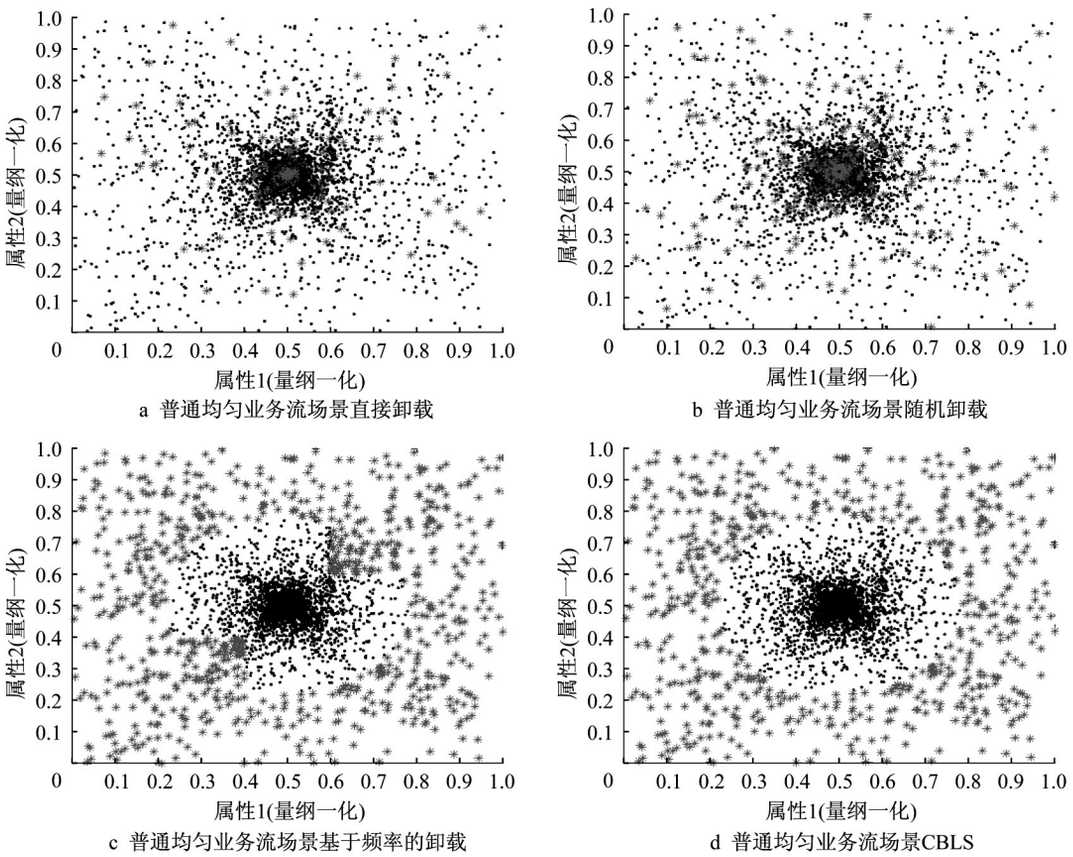

图 4　CBLS 和传统方法卸载质量对比

Fig.4　Quality comparison of load shedding between CBLS and traditional methods

的差异,表 1 展示了 4 种卸载算法在卸载比例从 5% 到 25% 时相应的平均 SSQ 值,并据此绘制成如图 5 所示的数据质量对比折线图。在该场景下平均 SSQ 值越小,表示越能识别出无用数据,算法效果越好。

由图 5 可以看出,在不同的卸载比例下,采用随机卸载算法处理后的数据的平均 SSQ 值较为平稳,

表 1　不同卸载方法的平均 SSQ

Tab.1　Average SSQ of CBLS and traditional shedding methods

卸载比例/%	直接卸载	随机卸载	基于频率卸载	CBLS
5	3.838 8	3.831 8	3.802 8	3.702 8
6	3.837 6	3.831 6	3.800 6	3.700 6
7	3.838 2	3.825 2	3.795 2	3.695 2
8	3.840 8	3.826 8	3.790 8	3.690 8
9	3.844 8	3.824 5	3.784 5	3.684 8
10	3.836 8	3.824 1	3.779 8	3.681 8
11	3.824 8	3.823 6	3.770 8	3.674 8
…	…	…	…	…
25	3.831 2	3.816 2	3.699 5	3.545 1

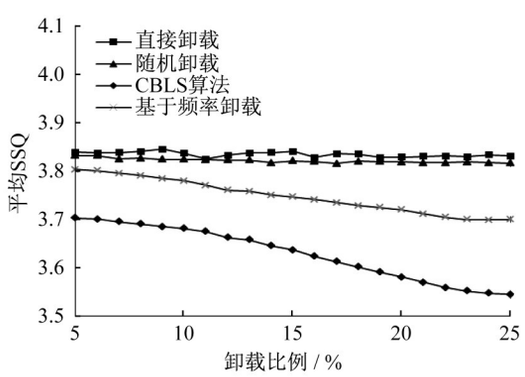

图 5　CBLS 和传统方法卸载不同比例质量对比

Fig. 5　Quality comparison between CBLS and traditional methods when shedding different proportion

采用直接卸载算法处理的平均 SSQ 值上下有波动,且始终保持较高。而基于频率的卸载方法和 CBLS

算法的平均SSQ值随着卸载比例的增大卸载后不断减小,但CBLS算法减小更快。这说明在该场景下,随着卸载比例的增大,CBLS算法卸载了更多的无用数据,相比其他算法具有更高的卸载准确性。

3.2 异常检测业务流大数据应用实验

在实验中,根据该应用场景下数据分布特点,人工合成实验所需数据集。使用文献[19]中的方法共生成613 450条数据,每个元数据包含10个数值属性,经规范化后所有属性的取值均在[0,1]区间。根据应用场景的要求,模拟的该数据集大部分数据在较小的取值区间范围均匀分布,部分异常数据则较为分散,符合异常检测业务流大数据应用场景。

同样地,为了分析EBLS算法在异常检测应用场景下的卸载效果,实验对比了该算法与直接卸载、随机卸载和基于频率的卸载算法,在不同卸载比例下处理同一个数据集的平均SSQ值。其中,基于频率的卸载算法根据比例卸载T-tree上频率较高的数据。实验中,卸载比例从5%个增加到25%。其中,EBLS算法在不同卸载比例下的综合相似度阈值η_0通过经验确定。在每个卸载算法处理后,都通过统计程序计算平均SSQ判断卸载的有效性。

选取当卸载比例为5%时,对采用4种方法处理后的数据取前2维作数据分布示意图,如图6所示,其中灰色星号点为要卸载的点,黑色点为保留点。图6为直接卸载、随机卸载和基于频率的卸载和EBLS算法卸载效果示意图,其中图6a和图6b灰色点分布在整个数据集,这显示出这2种卸载算法会丢弃一部分异常数据。图6c在中心部分能够卸载更多的重复数据,但仍会卸载分布在边缘的异常数据。图6d能更多保留边缘的异常数据,卸载正常数据。这4张图从直观上体现了EBLS卸载算法有较好的卸载准确性。

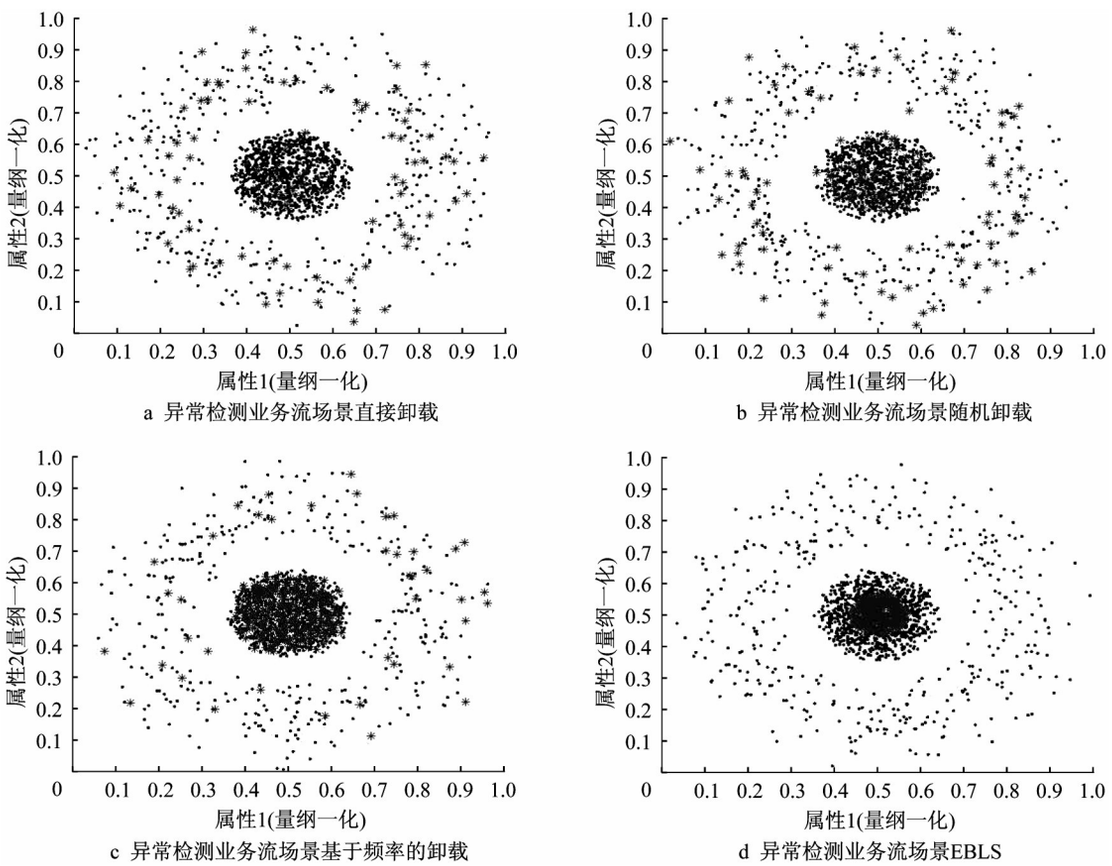

图6 EBLS和传统方法卸载质量对比

Fig.6 Quality comparison of load shedding between EBLS and traditional methods

为了客观对比4种算法的卸载效果,表2展示了4种卸载算法在卸载比例从5%到25%时,4种算法卸载处理后的平均SSQ值,并据此绘制成如图7所示的数据质量对比折线图。在这种场景下平均SSQ值越大,表示越能识别出异常数据,也就是说算法效果越好。

由图7可见,利用4种卸载算法处理数据时,随着卸载比例的增加直接卸载算法的平均SSQ值有

表2　EBLS和传统卸载方法的平均SSQ

Tab.2　Average SSQ of EBLS and traditional shedding methods

卸载比例/%	直接卸载	随机卸载	基于频率卸载	EBLS
5	4.180 6	4.070 1	4.321 1	4.621 1
6	4.152 2	4.032 1	4.325 1	4.625 1
7	4.142 8	4.051 8	4.354 2	4.654 2
8	4.190 5	4.087 4	4.364 3	4.714 3
9	4.245 3	4.025 6	4.406 5	4.786 5
10	4.361 2	4.056 4	4.415 8	4.735 8
11	4.181 3	4.022 3	4.417 7	4.757 5
…	…	…	…	…
25	4.185 1	4.023 4	5.008 5	5.775 1

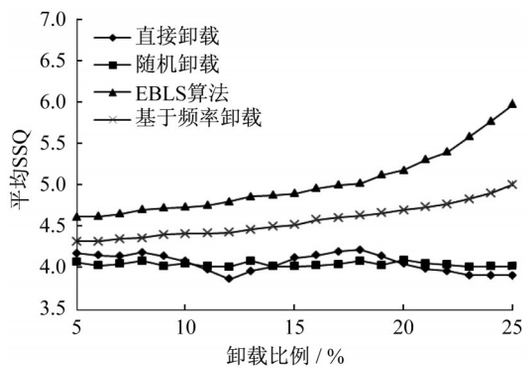

图7　EBLS和传统方法卸载不同比例质量对比

Fig. 7　Quality comparison between EBLS and traditional methods when shedding different proportion

一定的波动,而随机卸载算法总体较为平稳,且直接卸载和随机卸载算法的平均SSQ值明显低于基于频率的卸载算法和EBLS算法。但随着卸载比例的增加,EBLS算法的平均SSQ值增长明显快于基于频率的卸载算法,说明在该场景下EBLS算法更具有效性。

综上可知,在发生过载时使用了算法CBLS和算法EBLS卸载的准确性明显高于直接卸载和随机卸载,能够有效提高卸载准确性。

4　结语

针对普通均匀业务流大数据应用场景进行分析,把数据看成空间中的点,将该场景下无效数据视为离群点,给出了数据间弹性距离度量方法对不同价值的数据进行缩放,结合该应用场景下流大数据的超球分布特点,提出了以数据离心率为依据的CBLS卸载算法。针对异常检测业务流大数据应用场景下,希望卸载行为相似差异度不大的数据的要求,采用预处理自动机对数据处理过程进行建模,给出了数据处理行为相似性度量方法,结合数据差异度给出了综合相似性度量方法,并以此提出了基于等价类划分的EBLS卸载算法。最后,开展了实验测试,分别在2种应用场景的数据分布条件下,使用直接卸载、随机卸载和本文提出的2种卸载算法,在处理过载流大数据时进行对比实验,结果表明使用本文算法明显提高了流大数据卸载的准确性。

本文不足之处是未考虑流大数据之间有关联的情况,也未考虑概念漂移情况下的数据卸载。为此,下一步工作将研究在多源流大数据之间存在相互关联的情况下,如何有效地进行流大数据的卸载处理。

参考文献:

[1] ASSUNÇÃO M, VEITH A, BUYYA R. Distributed data stream processing and edge computing: A survey on resource elasticity and future directions [J]. Journal of Network and Computer Applications, 2017, 103(1): 1.

[2] JANI R, BHATT N, SHAH C. A survey on issues of data stream mining in classification [C] // International Conference on Information and Communication Technology for Intelligent Systems. Ahmedabad: Springer Verlag, 2017: 137 -143.

[3] KITHULGODA C I, PEARS R, NAEEM M A. The incremental Fourier classifier: Leveraging the discrete Fourier transform for classifying high speed data streams [J]. Expert Systems with Applications, 2018, 97(1): 1.

[4] BABCOCK B, DATAR M, MOTWANI R. Load shedding for aggregation queries over data streams [C] // International Conference on Data Engineering. Washington D C: IEEE Press, 2004:350-361.

[5] TATBUL N, CETINTEMEL U, ZDONIK S. Load shedding in a data stream manager [C] // International Conference on Very Large Data Bases. Berlin: ACM Press, 2003: 309-320.

[6] 闫莺,金澈清,曹锋,等. 多数据流上共享窗口连接查询的降载策略[J]. 计算机研究与发展,2004, 41(10):1836.
YAN Ying, JIN Cheqing, CAO Feng, *et al*. Load Shedding for shared window joins over data streams [J]. Journal of Computer Research and Development, 2004, 41(10): 1836.

[7] CHANG J H, KUM H C. Frequency-based load shedding over a data stream of tuples [J]. Information Sciences, 2009, 179 (21): 3733.

[8] ZHANG Y, HUANG C, ZHANG D. A novel adaptive load shedding scheme for data stream processing [C] // Future Generation Communication and Networking. Jeju: IEEE Press, 2007: 378-384.

[9] MA L, ZHANG Q, SHI N. A semantic load shedding algorithm based on priority Tab. in data stream system [C] // Seventh International Conference on Fuzzy Systems and Knowledge Discovery. Yantai: IEEE Press, 2010: 1167-1172.

[10] MAISON R, ZAKRZEWICZ M. Prediction-based load shedding for burst data streams [J]. Bell Labs Technical Journal, 2011, 16(1): 121.

[11] 魏霞, 李国徽. 基于直方图的数据流降载策略研究[J]. 华中科技大学学报(自然科学版), 2014, 42(9): 24.
WEI Xia, LI Guohui. Histogram strategy for load shedding over data streams[J]. Journal of Huazhong University of Science and Technology(Natural Science Edition), 2014, 42(9): 24.

[12] BELGHAOUTI F, BOUZEGHOUB A, AOUL Z K. Graphoriented load-shedding for semantic data stream processing [C]// International Workshop on Computational Intelligence for Multimedia Understanding. Prague: IEEE Press, 2015: 1-5.

[13] TU Y C, PRABHAKAR S. Control-based load shedding in data stream management systems [C]// International Conference on Data Engineering Workshops. Atlanta: IEEE Press, 2006: 144-148.

[14] DESAI D, JOSHI A. A deviant load shedding system for data stream mining [J]. Procedia Computer Science, 2015, 45(1): 118.

[15] 杨晓明, 张翔, 王佳昊, 等. 基于有限自动机的RFID入侵检测[J]. 电子科技大学学报, 2014, 43(5): 775.
YANG Xiaoming, ZHANG Xiang, WANG Jiahao, *et al*. RFID intrusion detection with finite automation [J]. Journal of University of Electronic Science and Technology of China, 2014, 43(5): 775.

[16] 李昆仑, 万品哲, 张德智. 基于改进用户相似性度量和评分预测的协同过滤推荐算法[J]. 小型微型计算机系统, 2018, 39(3): 567.
LI Kunlun, WAN Pinzhe, ZHANG Dezhi. Collaborative filtering recommendation algorithm based on improved user similarity measure and scoring forecast[J]. Journal of Chinese Computer Systems, 2018, 39(3): 567.

[17] Yahoo! Inc. Webscope dataset ydata labeled time series anomalies v1.0 [EB/OL]. [2015-03-24]. https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70.

[18] AGGARWAL C C, HAN J. A framework for clustering evolving data streams[C]// International Conference on Very Large Data Bases. Berlin: Elsevier, 2003: 81-92.

[19] 吕艳霞, 王翠容, 王聪, 等. 一种基于数据不确定性的概念漂移数据流分类算法[J]. 应用科学学报, 2017, 35(5): 559.
LV Yanxia, WANG Cuirong, WANG Cong, *et al*. Data stream classification with data uncertainty and concept drift[J]. Journal of Applied Sciences, 2017, 35(5): 559.

(上接第164页)

ZHENG Yang, ZOU Daoqin. Energy evaluating method for critical structural elements based on Neumann series [J]. Structural Engineers, 2012, 28(4): 63.

[7] JOSHI D, KUMAR S. Interval-valued intuitionistic hesitant fuzzy Choquet integral based TOPSIS method for multi-criteria group decision making [J]. European Journal of Operational Research, 2015, 248 (1), 183. DOI: 10.1016/j.ejor.2015.06.047.

[8] 石永久, 王萌, 王元清. 钢框架焊接节点损伤破坏行为分析[J]. 工程力学, 2013, 30(4): 73.
SHI Yongjiu, WANG Meng, WANG Yuanqing. Analysis on damage and failure behavior of steel frame welded connections [J]. Engineering Mechanics, 2013, 30(4): 73.

[9] 吴迪, 武岳, 杨庆山, 等. 大跨屋盖结构风振响应参数灵敏度分析[J]. 工程力学, 2015, 32(2): 171.
WU Di, WU Yue, YANG Qingshan, *et al*. Parameter sensitivity analysis of wind-induced response of long-span roofs [J]. Engineering Mechanics, 2015, 32(2): 171.

[10] 丁洁民, 沈祖炎. 一种半刚性节点的实用计算模型[J]. 工业建筑, 1992(11): 29.
DING Jiemin, SHEN Zuyan. A practical calculating model for semi-rigid connections [J]. Industrial Construction, 1992(11): 29.

[11] European Committee for Standardization. Eurocode 3: Design of steel structures - Part 1-8: Design of Joints [S]. Brussels: CEN, 2005.

[12] 中华人民共和国住房和城乡建设部. 建筑抗震设计规范: GB 50011—2010[S]. 北京: 中国建筑工业出版社, 2010.
Ministry of Housing and Urban-Rural Development of the People's Republic of China. Code for seismic design of buildings: GB 50011—2010 [S]. Beijing: China Architecture Industry Press, 2010.

[13] 欧进萍, 段宇博, 刘会仪. 结构随机地震作用及其统计参数[J]. 哈尔滨建筑工程学院学报, 1994, 27(5): 1.
OU Jinping, DUAN Yubo, LIU Huiyi. Structural random earthquake action and its statistical parameters [J]. Journal of Harbin Architectural & Civil Engineering Institution, 1994, 27(5): 1.

[14] 刘海鑫. 建筑钢结构适用性能的可靠性分析与蒙特卡罗实现[D]. 重庆: 重庆大学, 2003.
LIU Haixin. The reliability analysis of applicable performance for building steel structure and its realization with Monte Carlo [D]. Chongqing: Chongqing University, 2003.