基于 EMD 距离的数据流分布式相似性连接技术

许 嘉[1),2),3)] 宋 超[1)] 吕 品[1),2),3)] 李陶深[1),2)]

[1)](广西大学计算机与电子信息学院 南宁 530004)

[2)](广西高校并行分布计算技术重点实验室 南宁 530004)

[3)](广西多媒体通信与网络技术重点实验室 南宁 530004)

摘 要 随着数据获取设备的不断进步和数据获取技术的快速发展,如何分析和挖掘应用中快速产生的数据流成为亟待解决的问题.数据流的相似性连接返回两个数据流上相似的数据对,是分析和挖掘数据流的重要操作.相比于 L_p 范式距离,例如曼哈顿距离和欧氏距离,EMD 距离(Earth Mover's Distance)因其可以更准确地量化直方图元组之间的相似性而受到广泛关注,被广泛应用于解决基于内容的图像检索、冗余图像识别以及视频对象跟踪等重要应用问题.然而 EMD 距离的计算复杂度却高达三次方,阻碍了 EMD 距离在数据流相似性连接问题中的应用.该文基于开源的 Apache Storm 数据流分布式并行处理框架,设计并实现了基于 EMD 距离的数据流分布式相似性连接技术,命名为 EMD-DDSJ 技术.该技术在数据分发时维护了连接计算节点上的数据局部性,并基于该数据局部性增强了连接算法对不相似直方图元组对间 EMD 计算的过滤性能,提高了各个连接计算节点的执行效率.同时基于连接计算节点的代价模型,提出了基于反馈的负载均衡策略,有效提升 EMD-DDSJ 技术的整体执行性能.在真实数据集上的实验结果展示了该文提出的 EMD-DDSJ 技术的高效性和可扩展性,比相关最好的技术在处理吞吐率上最高提升了 1.4 倍,在元组平均处理延迟上最多降低了 44%,并且随着相似性阈值或滑动窗口大小的增大该提升比率还会进一步增大.

关键词 EMD 距离;相似性连接;数据流;Apache Storm 框架;数据局部性

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2019.01779

Distributed Similarity Join Over Data Streams Based on Earth Mover's Distance

XU Jia[1),2),3)] SONG Chao[1)] LV Pin[1),2),3)] LI Tao-Shen[1),2)]

[1)](*College of Computer, Electronics and Information, Guangxi University, Nanning* 530004)

[2)](*Guangxi Colleges and Universities Key Laboratory of Parallel and Distributed Computing, Nanning* 530004)

[3)](*Guangxi Key Laboratory of Multimedia Communications Network Technology, Nanning* 530004)

Abstract With the continuous progress of data acquisition equipment and the rapid development of data acquisition technologies, analyzing and mining quick-generated data streams from practical applications become an urgent problem to solve. Similarity join over data streams, which returns all similar pairs of tuples from two data streams, is an important operation for analyzing and mining data streams. Compared with L_p norm distance functions, such as Manhattan distance and Euclidean distance, Earth Mover's Distance (EMD) is more robust in quantifying the similarities of histogram-representative tuples and thus receives widespread attentions. EMD is widely applied

收稿日期:2017-09-27;在线出版日期:2018-04-21.本课题得到国家自然科学基金(61402494)、广西自然科学基金青年基金(2015GXNSFBA139243)、"广西八桂学者"专项经费、广西大学科研基金资助项目(XGZ141182,XGZ150322)和广西高等教育本科教学改革工程项目重点项目(2017JGZ103)资助.**许 嘉**,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为数据库理论与技术、图数据分析、数据隐私保护和大数据分布式并行计算.E-mail:xujia@gxu.edu.cn.**宋 超**,硕士研究生,主要研究方向为数据流管理技术.**吕 品**(通信作者),博士,副研究员,中国计算机学会(CCF)会员,主要研究方向为无线网络与移动计算、物联网、网络虚拟化和网络数据分析.E-mail:lvpin@gxu.edu.cn.**李陶深**,博士,教授,中国计算机学会(CCF)会员,主要研究领域为无线 Mesh 网络、云计算与大数据、网络计算与信息安全、分布式数据库等.

to solve many important application problems, such as content-based image retrieval, duplicated image recognition and video object tracking. However, the computation of EMD owns cubic time complexity, which hampers the application of EMD in solving similarity join problem over data streams. Based on Apache Storm, which is an open source distributed parallel data stream processing framework, we design and implement a distributed similarity join technique for data streams based on EMD, named EMD-DDSJ. This technique partitions histogram-representative tuples in data streams based on their similarities in terms of EMD and thus maintains good data locality on each join-computing node. It is the good data locality that enhances the filtering effect of joining algorithm to unpromising EMD calculations between dissimilar histogram-representative tuple pairs, and therefore improves the execution efficiency of join-computing nodes. On the basis of the cost model of join-computing nodes, we also propose an effective load balancing strategyon the strength of feedback messages sent by join-computing nodes, which improves the overall performance of the EMD-DDSJ technique. Experimental results on physical-world datasets show the efficiency and good scalability of the proposed EMD-DDSJ technique. Particularly, the processing throughput of EMD-DDSJ technique is at most 1.4 times higher than that of the best related technique and the average tuple processing delay of EMD-DDSJ technique decreases up to 44% compared with the best related technique. Besides, with the growth of the similarity threshold or the size of sliding window, the throughput improvement ratio of EMD-DDSJ technique compared to the related technique will be further increased.

Keywords earth mover's distance; similarity join; data stream; Apache Storm framework; data locality

1 引 言

随着数据获取设备的不断进步和数据获取技术的快速发展,如何对源源不断、快速产生的数据流进行高质量的数据分析成为工业界和学术界普遍关注的问题.数据流相似性连接返回两个数据流上相似的元组对,是分析和挖掘数据流的重要操作,广泛应用于事件检测和数据去重等实际应用中.本文讨论由直方图数据模型表征的数据元组(简称为直方图元组)构成的数据流间的相似性连接问题.每个直方图元组可以形式化表示为 $P=\{p_1,\cdots,p_n\}$,其中 $p_i\in[0,1]$表示数据对象出现在直方图第 i 个数据桶的概率值.直方图因为提取过程简单,被广泛应用于对复杂数据对象特征的概率分布进行建模.例如在计算机视觉领域中常使用直方图来表征图像的内容特征(例如颜色、明暗和纹理等)的概率分布[1];又如在无线传感器网络中通常会把多个临近传感器节点同一时间段的采样数据构建成直方图来表征采样数据的不确定性[2];再如在基于位置的服务中,通常将不同位置采集装置所获取的目标对象在同一时刻的位置信息整合成直方图概率分布的形式实现对目标对象某时刻位置状态的建模[3].直方图相似性度量函数的选择直接影响直方图数据流相似性连接的结果质量.相比于以欧式距离为代表的 L_p 范式距离函数,EMD 距离(Earth Mover's Distance,EMD)[4]不仅量化了直方图对应数据桶概率值之间的相似性,还量化了直方图临近数据桶概率值之间的相似性,因而对概率分布之间的微小偏移不敏感,具有更强的鲁棒性.基于 EMD 距离的相似性搜索因而被广泛应用在基于内容的图像检索[4-5]或视频监控[6]、冗余图像识别[7]、对象跟踪[8]和传感器网络数据监控[2,9]等重要领域.

基于 EMD 距离的数据流相似性连接问题颇具挑战.一方面,数据流源源不断、快速到达的特性要求连接算法必须快速高效.另一方面,计算两个直方图元组之间的 EMD 距离需要求解一个时间复杂度高达 $O(n^3\log n)$(n 为直方图所包含的数据桶的数目)的线性规划问题,这无疑给设计与实现高效的基于 EMD 距离的数据流相似性连接技术带来巨大挑战.面对 EMD 距离的高计算复杂度,业界有三种解决思路.第一种思路是尽可能减少 EMD 距离的计

算次数，因为其是主要的计算瓶颈，包括两类方法："扫描—求精"法[1,10]和"索引—求精"法[7,11-13]. 这两类方法都是通过设计计算复杂度相对较低的 EMD 距离的上界或下界计算函数，然后基于这些计算函数或直接扫描过滤数据集("扫描—求精"法)，或构建索引后基于索引过滤数据集("索引—求精"法)，尽可能避免对数据集中不相似直方图元组对之间 EMD 距离的计算，从而提升相似性查询算法的处理效率. 其中，文献[13]针对滑动窗口语义下基于 EMD 距离的数据流相似性连接问题设计了面向 EMD 距离的轻量级索引方案，能够在数据被频繁插入和删除的情形下实现对索引的高效更新和维护. 本文运用了该研究工作提出的索引方案在各个分布式计算节点上构建和维护索引. 然而，该研究工作只是针对集中式计算环境设计的基于 EMD 距离的数据流相似性连接解决方案，不能解决分布式计算环境下的数据划分问题和各个分布式计算节点的负载均衡问题. 第二种解决思路是研究 EMD 距离在某些特殊设定下的快速计算策略[14-15]或是研究如何将 EMD 距离的计算过程用其它线性时间复杂度的计算过程进行近似[16]，目的是通过牺牲问题求解的普适性或是求解结果的计算精度来提升 EMD 距离的计算速度. 第三种解决思路是运用分布式并行计算策略缩减算法的整体执行时间. 例如基于流行的分布式并行批处理框架 MapReduce[17]，研究人员先后提出了 MELODY-JOIN[9]算法、Head-join[18]算法、EMD-MPJ 算法[7]和 Top-k DLPJ 算法[19]，有效提升了基于 EMD 距离的相似性连接算法在处理大规模数据集时的可扩展性. 可见，分布式并行计算是解决基于 EMD 距离相似性连接这类计算密集型应用的高效可行手段. 然而，这些研究工作只针对基于 EMD 距离的相似性连接问题提供了批处理解决方案. 批处理情况下数据是静态完整的，算法可以通过预先扫描数据集获得数据集的分布特性，并基于该分布特性提出有针对性的优化策略. 而本文研究的基于 EMD 距离的数据流相似性连接问题涉及动态变化的数据流，一方面算法很难提前获得数据从而对数据流有足够认识，另一方面数据流上的查询要求快速返回连接结果，增大了算法优化的难度.

在分布式系统负载均衡方面，也有不少研究工作针对如何划分数据集实现各分布式计算节点的负载均衡展开了讨论，主要有两种解决思路. 第一种解决思路是在发现各个分布式计算节点的计算负载出现不平衡的时候将计算负载从高计算负载的节点向低计算负载的节点进行迁移. 例如文献[20]以数据流的数据分布变化导致的系统负载分配不均为问题背景提出了有效的负载迁移的策略，然而数据迁移本身的代价不容小视. 第二种解决思路是研究负载划分的优化策略. 一方面基于哈希的数据划分策略虽然能维护各个分布式计算节点上的数据局部性(Data locality)，有利于利用数据局部性实现进一步查询优化，但是当对静态数据集数据分布的掌握不精准或是数据分布动态变化容易造成分布式计算节点间负载不均衡的现象. 另一方面随机的数据划分策略能够严格保证各个计算节点间的负载均衡性，却丢失了计算节点上数据的局部性. 基于以上观察，许多研究工作将基于哈希的数据划分策略和随机的数据划分策略相结合，优化数据划分过程以期实现计算节点间的负载近似均衡，典型的代表工作有面向静态数据集的 EMD-MPJ 算法[7]和面向数据流的 Cont-Rand[21]算法. 为了进一步在数据流环境下优化负载均衡策略，研究人员还先后提出 PKG 算法[22]和 PKGE 算法[23]. 区别于 Cont-Rand 算法，PKG 算法和 PKGE 算法对分布式计算框架中每个下游节点的数据分发量进行统计，并依据此优化下游计算节点的数据分发策略. 然而，以上提出的面向分布式数据流处理的负载均衡策略均是针对数据流上计算复杂度较低的等值连接操作或是针对基于简单距离函数的相似性连接操作，因此连接代价模型关注的是计算节点所获得的数据量而非实际发生的计算量. 由于 EMD 距离具有高计算复杂度，节点执行相似性连接时实际发生的 EMD 距离计算次数将是连接代价模型关注的重点，因此以上研究工作均不适用于解决数据流上基于 EMD 距离的分布式相似性连接问题. 虽然 EMD-MPJ 算法[7]和 Top-k DLPJ 算法[19]基于 EMD 距离的特性设计了分布式计算环境下的数据相似性连接方案，但是它们处理的均是静态数据集，其负载均衡策略不适用于动态变化的数据流环境.

为了保证数据能够得到快速的计算，本文选取开源 Apache Storm① 作为直方图数据流的分布式处理框架平台. 集中探讨如何基于 Apache Storm 框架实现滑动窗口语义下基于 EMD 距离的数据流分布式相似性连接问题，有效支持但不限于以下应用：

应用 1.　在基于内容的视频检索和监控应用中，每个视频帧的颜色、灰度和纹理等重要内容特征

① http://storm.apache.org

通常可以用直方图的形式进行提取和表征.因而两个视频流可以转换成两条直方图数据流.在滑动窗口语义下基于EMD距离对这两条数据流进行相似性连接可以返回这两条数据流上时间临近且颜色、灰度或纹理相似的视频帧对,为视频流间的相关性分析或视频流中的对象跟踪提供支撑[6].具体而言,可以应用于对视频分享网站(例如YouTube或优酷)两个用户实时上传的视频流内容进行相似性连接比对,及时发现用户上传视频流中的类似场景,从而及时发现潜在的热点事件.又如,还可以应用于基于监控视频的对象跟踪:对邻近的两个监控摄像头拍摄的视频流进行相似性连接比对,从而得到同一对象的运动轨迹.

应用2. 在无线传感器监控应用中,由于硬件故障、信号干扰和电量损耗等原因,单个传感器反馈的采样数据存在一定的不精确性.通常会基于同一区域中的多个传感器节点在同一时段内的采样数据构建采样数据分布直方图来表征区域采样值的不确定性[2,11].这样从同一区域传回的传感器采样数据流可以转化为直方图形式表征的采样数据分布数据流.在滑动窗口语义下基于EMD距离对不同区域传回的直方图数据流进行相似性连接可以挖掘临近时间范围中采样读数具有关联性的监控区域对,便于后续的监控和分析[2,11].

综上,本文的主要贡献包括:

(1)基于Apache Storm框架设计并实现了滑动窗口语义下基于EMD距离的数据流分布式相似性连接技术,简称为EMD-DDSJ技术.EMD-DDSJ技术在数据分发时维护了连接计算节点上的数据局部性,并基于该数据局部性增强了连接算法对不相似直方图元组对间EMD计算的过滤性能,有效减少了分布式计算框架中各个连接计算节点上的计算负载;

(2)给出了连接计算节点的代价模型,并基于该代价模型设计与实现了基于反馈的负载均衡策略,有效提升了EMD-DDSJ技术的整体执行效率;

(3)采用真实的流数据集进行了大量实验,证明了本文提出的EMD-DDSJ技术的有效性和可扩展性.实验证明,本文提出的EMD-DDSJ技术比相关最好的技术在处理吞吐率上最多提高了1.4倍,在元组平均处理延迟上最多降低了44%,并且随着相似性阈值或滑动窗口大小的增大该提升比率还会进一步增大.

本文第2节给出问题定义并对两个相关技术ShuffleJ和Head-SJ进行介绍和分析;第3节重点阐述本文所提出的基于EMD距离的数据流分布式相似性连接技术(EMD-DDSJ技术)的整体设计思路;第4节着重介绍EMD-DDSJ技术中基于反馈的负载均衡策略;第5节对EMD-DDSJ技术、ShuffleJ技术和Head-SJ技术进行代价分析和比较,从理论上证明EMD-DDSJ技术的优越性;第6节对EMD-DDSJ技术和相关技术ShuffleJ和Head-SJ进行实验评估并对实验结果进行分析;第7节对全文进行总结.

2 预备知识

2.1 问题定义

定义1. EMD距离.已知各包含n个数据桶的直方图元组$P=\{p_1,\cdots,p_n\}$和$Q=\{q_1,\cdots,q_n\}$,以及地面距离矩阵$\boldsymbol{C}=[c_{ij}]$,则P和Q之间的EMD距离,记为$EMD(P,Q)$,是将直方图P搬运转换为直方图Q的最小搬运代价,即等于以式(1)所对应线性规划问题的最优解.

$$\begin{aligned}&\text{Minimize: } \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}\times c_{ij},\\&\text{s.t. } \forall i: \sum_{j=1}^{n} f_{ij}=p_i,\\&\forall j: \sum_{i=1}^{n} f_{ij}=q_j,\\&\forall i,j: f_{ij}\geq 0\end{aligned}\tag{1}$$

其中,已知的地面距离$c_{ij}\in C$表示从直方图P的第i个数据桶到直方图Q的第j个数据桶的搬运代价.可见,该线性规划问题共有n^2个变量,记为$F=\{f_{ij}\}$.其中,f_{ij}表示从直方图P的第i个数据桶搬运至直方图Q的第j个数据桶的概率值.EMD距离可以基于单纯形算法进行求解,且求解的时间复杂度高达$O(n^3\log n)$.本文假设EMD距离定义中使用的地面移动距离是一种几何距离(Metric distance),例如常见的曼哈顿距离和欧几里德距离,此时EMD距离也是一种几何距离[4],满足几何距离的三大特性:非负性($EMD(P,Q)\geq 0$)、对称性($EMD(P,Q)=EMD(Q,P)$)和三角不等性($EMD(P,Q)+EMD(P,U)\geq EMD(Q,U)$).

定义2. 滑动窗口语义下基于EMD距离的数据流相似性连接.已知直方图数据流$R=\{r_1,\cdots,r_i,\cdots\}$和$S=\{s_1,\cdots,s_j,\cdots\}$,滑动窗口的大小为$|W|$,相似性阈值为$\theta$,则滑动窗口语义下基于EMD距离的数

据流 R 和 S 的相似性连接返回以下直方图元组对的集合：

$$\{\langle r_i, s_j\rangle \mid r_i \in R, s_j \in S, EMD(r_i, s_i) \leq \theta, \\ |r_i.ts - s_j.ts| \leq |W|\} \quad (2)$$

其中，$x.ts$ 表示直方图元组 x 的时间戳.

本文涉及的重要符号及其含义说明详见表 1.

表 1　重要符号及其含义

符号	含义说明
R/S	数据流 R 和 S
r_i	数据流 R 上的第 i 个直方图元组
s_i	数据流 S 上的第 j 个直方图元组
n	直方图数据桶的个数
k	Join-bolt Task 的并行度
θ	相似性阈值
$\|W\|$	滑动窗口的大小
$Z=\{\Phi,\Pi\}$	一组 EMD 距离对偶线性规划问题的可行解
$\Omega(Z)$	基于 EMD 距离对偶线性规划问题的一组可行解 Z 得到的一维实数映射空间
$key(P,\Pi)$	直方图 P 在 $\Omega(Z)$ 上的键值
$ckey(P,\Pi)$	直方图 P 在 $\Omega(Z)$ 上的反键值
m_k	第 k 个 Join-bolt Task 所负责的 $\Omega(Z)$ 中子空间的划分点

2.2　相关技术

2.2.1　Apache Storm 框架

由于编程模型简洁，执行效率、可靠性和可扩展性高等优点，开源的分布式数据流实时计算框架 Apache Storm（简称为 Storm）被广泛应用于实时数据分析、联机学习和 ETL 等应用中. 本文基于 Storm 框架设计与实现了滑动窗口语义下基于 EMD 距离的数据流分布式相似性技术，下面对 Storm 框架做简要介绍.

Storm 框架基于 Topology 实现对数据流的分布式实时处理. Topology 中数据以元组（Tuple）的形式进行处理和转发. Topology 一经启动将永久运行，不断处理实时到达的数据元组. 如图 1 所示，一个 Topology 由 Spout 和 Bolt 两类组件构成. Spout 作为数据产生者，从一个外部源读取数据并向 Topology 里喷射数据元组. Bolt 作为数据消费者，对所接收的数据元组进行处理和转发. 一个复杂的 Topology 可由多个 Spout 和多个 Bolt 组成，且可以

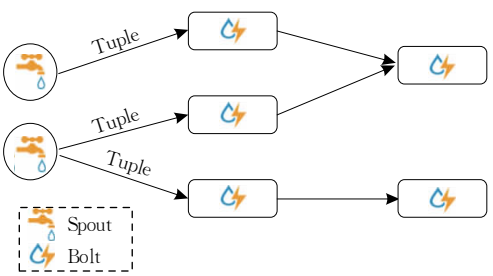


图 1　Apache Storm 框架的一个典型 Topology 结构

为每个 Spout 或 Bolt 设置 Task 并行度，由多个 Task 并行完成其处理逻辑. Storm 提供多种组件间的数据分组（划分）策略，例如随机分组（Shuffle grouping）、字段分组（Field grouping）和全局分组（All grouping），完成 Topology 中上游组件的各个 Task 向下游组件的各个 Task 的数据分发.

2.2.2　ShuffleJ 技术

运用 Storm 提供的随机分组策略和全局分组策略可以将文献[13]所述的集中式计算环境下面向滑动窗口语义的基于 EMD 距离的数据流相似性连接解决方案应用于分布式计算环境（下文简称为 ShuffleJ 技术）. 具体而言，Spout 组件上的每个 Task 每当收到一条来自 S 数据流的元组时即基于全局分组（All grouping）策略向下游负责执行相似性连接的 Bolt 组件的各个 Task（简称为 Join-bolt Task）广播该元组；Spout 组件每当收到一条来自 R 数据流的元组时即基于随机分组（Shuffle grouping）策略将该元组随机划分给下游负责执行相似性连接的 Bolt 组件的一个 Join-bolt Task. 每个 Join-bolt Task 获得当前滑动窗口内完整的 R 数据分片和 S 数据分片后，按照 3.3 节中的过滤链执行这两个数据分片之间基于 EMD 距离的相似性连接，并将连接结果写入外部数据存储设备. 这种对数据流 R 进行随机划分的分布式相似性连接技术易于实现，由于各个 Join-bolt Task 得到的数据量和数据分布近似一致，各个 Join-bolt Task 的连接负载是均衡的. 然而，由于随机分发 R 数据流上的元组，ShuffleJ 技术破坏了 Join-bolt Task 所获得的 R 数据分片的数据局部性（Data locality），即不能把 EMD 距离相近的 R 数据流元组分发到同一个 Join-bolt Task 上进行连接处理，因而 Join-bolt Task 就不能利用数据局部性过滤大量不相似元组对之间的 EMD 距离计算，制约了 Join-bolt Task 上基于 EMD 距离的相似性连接处理效率的提升.

2.2.3　Head-SJ 技术

文献[18]基于 Hadoop 框架提出了基于 EMD 距离的大规模数据相似性连接的分布式批处理技术 Head-Join. 本文给 Head-Join 技术加上了滑动窗口语义并在 Apache Storm 框架下进行了实现，命名为 Head-SJ 技术. Head-SJ 技术的 Spout 组件同样基于全局分组策略向下游的各个 Join-bolt Task 广播 S 数据流的每一个元组. 同时，Head-SJ 技术的 Spout 组件利用 EMD 距离的正态下界函数（Normal lower bound）[12] 将每个到达的 R 数据流元组映射为霍夫

正态空间(Hough normal space)内的一个二维向量.将霍夫正态空间划分为 k 个子空间,则二维映射向量落在第 i 个子空间内的 R 数据流元组被划分给第 i 个 Join-bolt Task 进行相似性连接处理[18].当一个 Join-bolt Task 获得当前滑动窗口内完整的 R 数据分片和 S 数据分片后,其按照包含了 EMD 距离的上界/下界函数的过滤链,即"正态下界[12]→对偶下界[11]→质心下界[4]→引用上界[18]",执行这两个数据分片之间基于 EMD 距离的相似性连接,并将连接结果写入外部数据存储设备.Head-SJ 技术将二维映射向量相近的 R 数据流元组划分给同一个 Join-bolt Task,由于该二维映射向量反映了 EMD 距离的正态下界,因而也能够保证每个 Join-bolt Task 所获得的 R 数据分片针对 EMD 距离的数据局部性.然而,Head-SJ 技术却没有充分利用这种数据局部性对 EMD 距离的求精计算进行过滤.本文提出的 EMD-DDSJ 技术通过在过滤链中嵌入新可行解过滤机制和三角不等式过滤机制,可以利用数据局部性过滤掉更多的不相似元组对之间的 EMD 距离计算,详见 3.3 节.同时,由于 Head-SJ 技术是由基于 EMD 距离的大规模数据相似性连接的批处理技术 Head-Join 变化实现的,因而没有基于 R 数据流元组数据分布的动态变化来调整对 R 数据流的划分策略,没法保证 Join-bolt Task 间的负载均衡.

3 基于 EMD 距离的数据流分布式相似性连接

从维护数据分发时的数据局部性和利用数据局部性增强对不相似直方图元组对间 EMD 计算的过滤性能这两个角度出发,本节讨论如何基于 Apache Storm 框架设计高效的基于 EMD 距离的分布式数据流相似性连接技术,简称为 EMD-DDSJ 技术.下面首先阐述 EMD-DDSJ 技术的 Storm Topology 结构.

3.1 EMD-DDSJ 技术的 Storm Topology 结构

EMD-DDSJ 技术的 Storm Topology 结构如图 2 所示.首先,数据流 R 和 S 通过 Spout 节点将原始数据转换成直方图元组的形式并喷射给下游的 Partition-bolt.假设 Partition-bolt 下游的 Join-bolt 上有 k 个并行执行的 Join-bolt Task(图 2 中简单表示为 Task).Partition-bolt 一方面基于 Storm 框架的全局分组(All grouping)策略将每个来自数据流 S 的直方图元组广播至 Join-bolt 的每个 Join-bolt Task 上,然后每个 Join-bolt Task 为其所收到的 S 数据流分片构建轻量级的索引;Partition-bolt 另一方面将来自数据流 R 的相似的直方图元组发送给同一个 Join-bolt Task(详见 3.2 节).相比于 2.2 节介绍的 ShuffleJ 技术,EMD-DDSJ 技术对数据流 R 上元组进行划分时维护了各个 Join-bolt Task 得到的 R 数据流分片的数据局部性,为 Join-bolt Task 上的查询优化提供可能.每个 Join-bolt Task 在获得当前滑动窗口内 R 和 S 的数据分片之后,对 R 和 S 的数据分片执行基于 EMD 距离的相似性连接(详见 3.3 节).连接过程中利用 R 数据流分片的数据局部性过滤大量不相似直方图元组对之间的 EMD 距离计算,最终把连接结果写入外部数据存储系统,例如 Apache HDFS 分布式文件系统.各 Join-bolt Task 在执行连接计算的过程中记录自身在当前反馈周期的连接负载并在当前反馈周期结束时以负载反馈元组的形式发送给上游的 Partition-bolt(如图中虚线箭头所示).Partition-bolt 收集到上一反馈周期所有 Join-bolt Task 提交的负载反馈元组之后,基于代价模型调整当前反馈周期对数据流 R 的划分策略,均衡各个 Join-bolt Task 上的连接计算负载(详见第 4 节).

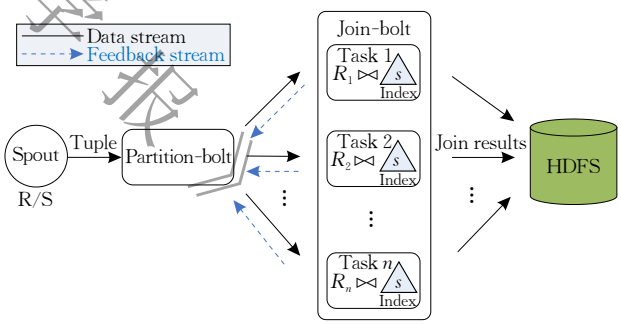


图 2 EMD-DDSJ 技术的 Storm Topology 结构

3.2 基于于数据局部性的数据流划分策略

基于线性规划的对偶理论,文献[11]提出了直方图元组至一维实数空间的映射机制.假设每个直方图元组包含 n 个数据桶,根据对偶理论可知式(1)所示的求解 EMD 距离的线性规划问题存在且仅存在一个对偶线性规划问题,即

$$\begin{aligned} &\text{Maximize}: \sum_{i=1}^{n} \varphi_i \times p_i + \sum_{j=1}^{n} \pi_j \times q_j, \\ &\text{s.t.} \quad \forall i,j: \varphi_i + \pi_j \le c_{ij}, \\ &\qquad \forall i: \varphi_i \in R, \\ &\qquad \forall j: \pi_j \in R \end{aligned} \tag{3}$$

以上对偶线性规划问题的一组可行解 $Z=\{\Phi,\Pi\}$ 包含 $2n$ 个变量，表示为 $\Phi=\{\varphi_1,\cdots,\varphi_n\}$ 和 $\Pi=\{\pi_1,\cdots,\pi_n\}$. 给定直方图元组 $P=\{p_1,\cdots,p_n\}$，则该元组在 EMD 距离对偶线性规划问题的可行解 $Z=\{\Phi,\Pi\}$ 下的一维实数映射空间（表示为 $\Omega(Z)$）中的映射值，即 $key(P,\Phi)$ 可由式(4)计算得到. 下文将该映射值简称为键值.

$$key(P,\Phi)=\sum_{i=1}^{n}(\phi_i\times p_i) \tag{4}$$

给定直方图元组 $Q\{q_1,\cdots,q_n\}$ 作为查询对象，则所有和 Q 之间的 EMD 距离不大于相似性阈值 θ 的直方图元组的键值必然落在式(5)所示的一维实数映射空间 $\Omega(Z)$ 的子空间中[8]. 其中 $ckey(Q,\Pi)$ 为查询对象 Q 的反键值，定义为 $ckey(Q,\Pi)=\sum_{i=1}^{n}(q_i\times\pi_i)$.

$$[\min_{i=1}^{n}(\phi_i+\pi_i)+key(Q,\Phi)-\theta,\theta=ckey(Q,\Pi)] \tag{5}$$

式(5)说明基于式(4)的映射函数可将 EMD 距离相近的直方图元组映射至一维实数映射空间 $\Omega(Z)$ 的临近区域，即该一维实数映射空间维护了直方图元组基于 EMD 距离的数据局部性.

给定 EMD 距离对偶线性规划问题的一组可行解 $Z=\{\Phi,\Pi\}$，且假设下游的 Join-bolt 上有 k 个并行执行的 Join-bolt Task，基于上述直方图元组的映射机制，Partition-bolt 首先将直方图的一维实数映射空间 $\Omega(Z)$ 划分为 k 个子空间，并规定映射至第 i 个子空间上的直方图元组由第 i 个 Join-bolt Task 负责处理. 每当接收到数据流 R 的元组，Partition-bolt 将基于式(4)计算该元组的键值，若该键值落在 $\Omega(Z)$ 的第 i 个子空间，则将该元组划分给第 i 个 Join-bolt Task. 由于式(4)所示的直方图元组映射函数维护了直方图元组基于 EMD 距离的数据局部性，因而 Partition-bolt 能够将 R 数据流上彼此相似的直方图元组分配给同一个 Join-bolt Task，维护了 Join-bolt Task 上 R 数据流分片的数据局部性. 可以通过实现 Storm 框架中的自定义分组（Custom stream grouping）接口实现 Partition-bolt 对数据流 R 的上述数据划分逻辑.

3.3　相似性连接策略

在获得当前滑动窗口内数据流 R 和 S 的完整数据分片之后，每个 Join-bolt Task 采用“索引—求精”的方式完成 R 数据流分片和 S 数据流分片之间面向滑动窗口语义且基于 EMD 距离的相似性连接. 首先在 S 数据流分片上构建轻量的面向流数据的 B^+ 森林索引[13]. 给定查询对象，该索引机制利用式(5)的结论过滤 S 数据流分片中和查询对象之间 EMD 距离大于相似性阈值的无关直方图元组. 其次以 R 数据分片中的每个元组 r 为查询对象，利用 S 数据流分片上的 B^+ 森林索引完成元组 r 在 S 的数据分片上面向滑动窗口语义且基于 EMD 距离的相似性连接，得到 r 与 S 数据流分片的相似性连接结果. 具体过程如图 3 所示：首先访问 S 数据流分片上的 B^+ 森林索引过滤所有与查询对象 r 不相似的 S 数据流分片上的直方图元组，得到 S 数据流分片上的约减查询候选集；其次先后基于新可行解过滤机制[24]、EMD 距离的下界函数 LB_{IM} 过滤机制[25]、EMD 距离的上界函数 UB_p 过滤机制[11]和三角不等式过滤机制[24]逐步约减 S 数据流分片上的查询候选集最终得到约减后的查询候选集 S'；最后求精计算 r 和 S' 中每个直方图元组之间的 EMD 距离，若距离值不大于相似性阈值 θ，则将该结果对输出至外部存储系统.

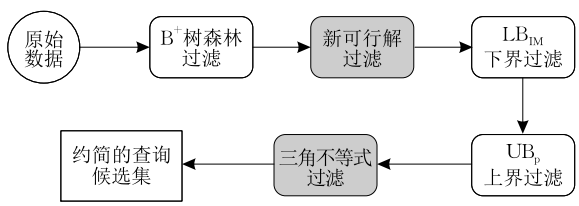


图 3　过滤链设置

图 3 所示的过滤链中，灰色背景标注的新可行解过滤机制和三角不等式过滤机制均利用了 R 数据流分片上前后查询对象之间的相似性（即良好的数据局部性）对 S 数据分片中的无关直方图元组进行过滤. R 数据流分片所维护的数据局部性越好，这两个过滤机制的过滤性能就越高[7]. 具体而言，新可行解过滤机制利用不可避免的 EMD 计算过程中顺带产生的 EMD 距离对偶线性规划问题的新可行解计算 EMD 距离的下界，并基于该下界值对候选集进行过滤. 前后查询对象之间越相似，基于新产生的可行解计算得到的 EMD 距离的下界越紧，则基于该下界值进行过滤的效果越好. 三角不等式过滤机制利用前后查询对象（例如 Q_i 和 Q_{i+1}）之间的相似性，利用 EMD 距离所满足的三角不等特性（即 $EMD(Q_i,P)+EMD(Q_i,Q_{i+1})\leq EMD(Q_{i+1},P)$），计算 Q_{i+1} 和 Q_i 的连接结果 P 之间 EMD 距离的下界值，并基于该下界值判断 P 是否也是 Q_{i+1} 的连接结果，从而避免 Q_{i+1} 和 P 之间的 EMD 距离计算. 可见，前后查询对象之间越相似，三角不等式过滤机制的对 EMD 计算的过滤效果越好. 图 4 展示了同一

数据集在经过图 3 所示的过滤链处理后，Join-bolt Task 对具有数据局部性的 R 数据流分片进行相似性连接时和 Join-bolt Task 对没有数据局部性的 R 数据流分片进行相似性连接时新可行解过滤机制和三角不等式过滤机制各自过滤掉的 EMD 计算次数占总 EMD 计算次数的比率. 可见，利用数据局部性设计的新可行解过滤机制和三角不等式过滤机制在 R 数据流分片具有数据局部性时的过滤性能比在 R 数据流分片不具有数据局部性时的过滤性能提高了至少 1 倍. 考虑到 EMD 距离的高计算复杂度，在数据划分时维护 Join-bolt Task 上 R 数据流分片的数据局部性可以有效提升各个 Join-bolt Task 处理基于 EMD 距离的相似性连接的执行效率.

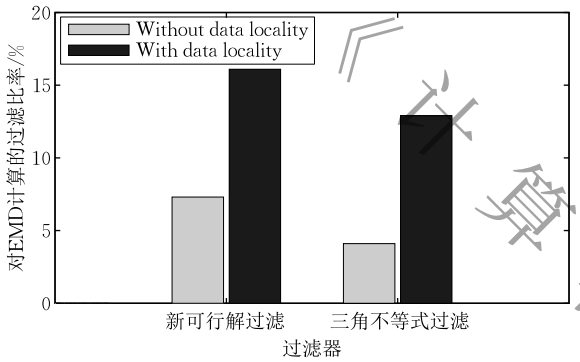


图 4 不同过滤器的过滤效果

3.4 基于 EMD 距离的数据流分布式相似性连接算法

算法 1 展示了 EMD-DDSJ 技术在 Partition-bolt 和 Join-bolt 上的具体处理流程. 在 Partition-bolt 端，对于每个到达的直方图元组 x_i，如果 x_i 属于数据流 S，则 Partition-bolt 基于 Storm 系统提供的全局分组(All grouping)策略将 x_i 广播至下游 Join-bolt 的所有 Task 上(行 1～3)；反之如果 x_i 属于数据流 R，则 Partition-bolt 首先计算 x_i 在直方图一维实数映射空间 $\Omega(Z)$ 内的键值 $Key(x_i,\Phi)$(详见式(4))，然后基于维护数据局部性的分组策略，将 x_i 发送至负责其键值 $Key(x_i,\Phi)$ 所在子空间的 Join-bolt Task 上(行 4～6). 每当收到上一反馈周期所有下游 Join-bolt Task 的连接负载反馈元组 F_M，Partition-bolt 将基于第 3 节提出的基于反馈的负载调整策略(详见算法 2)重新调整一维实数映射空间 $\Omega(Z)$ 内各个子空间的划分边界，从而调整当前周期对数据流 R 的划分策略，均衡各个 Join-bolt Task 上的连接负载(行 7～8). 在 Join-bolt 端，如果到达的元组 x_i 属于数据流 S，则 Join-bolt Task 将 x_i 插入 B^+ 森林索引 I[19](行 11～12)；反之如果 x_i 属于数据流 R，Join-bolt Task 将基于 2.3 节所述的相似性连接策略对不相似的元组对间的 EMD 计算进行过滤，最终得到元组 x_i 与 S 数据流分片的相似性连接的结果集 RS(行 14)，并在删除元组 x_i(行 15)之后将结果集 RS 存储至外部存储系统，例如 Apache HDFS(行 16). 如果当前反馈周期期满，则 Join-bolt Task 将其在当前反馈周期统计得到的负载反馈元组 F_M 发送给 Partition-bolt(行 17～18).

算法 1. EMD-DDSJ based on Apache Storm.

输入：直方图数据流 R 和 S；EMD 距离对偶线性规划问题的一组可行解 $Z=\{\Phi,\Pi\}$；滑动窗口 W；相似性阈值 θ

输出：相似性连接结果集

$RS=\{(r_i,s_j)|r_i\in R,s_j\in S,EMD(r_i,s_j)\leq\theta,|r_i.ts-s_j.ts|\leq|W|\}$

```
/* Partition-bolt */
1.  FOR EACH incoming tuple x_i on Partition-bolt DO
2.    IF x_i ∈ S THEN
3.      emit x_i by allGrouping();
4.    ELSE IF x_i ∈ R THEN
5.      calculate Key(x_i, Φ);
6.      emit x_i to a Join-bolt Task based on Key(x_i, Φ)
        and the partition rule got by dataLocalityGrouping
        (Key(x_i, Φ));
7.    IF receive the set of feedback workload tuples {F_M}
      from all Join-bolt Tasks in period P_l THEN
8.      dataLocalityGrouping({F_M});
9.      l++;
/* Join-bolt */
10. FOR EACH incoming tuple x_i on Join-bolt DO
11.   IF x_i ∈ S THEN
12.     insert x_i into B+ forest index I;
13.   ELSE IF x_i ∈ R THEN
14.     RS = WindowBasedEMDJoin(x_i, W, θ, I);
15.     delete x_i;
16.     store RS into HDFS;
17.   IF P_l.expire() == TRUE THEN
18.     send feedback workload tuple F_M to Partition bolt
```

4 基于反馈的负载均衡策略

给定 EMD 距离对偶线性规划问题(如式(3)所示)的任一可行解 $Z=\{\Phi,\Pi\}$，EMD-DDSJ 技术将直方图元组的一维实数映射空间 $\Omega(Z)$ 划分为若干个子空间，并将映射至第 i 个子空间的 R 数据流元

组分派给第 i 个 Join-bolt Task 进行连接处理. 这样做维护了 Join-bolt Task 上 R 数据流分片的数据局部性，可以利用数据局部性过滤不相似直方图元组对之间的 EMD 距离计算. 由于数据流上的数据分布随着时间推移会发生改变，如果不能在数据分布变化时及时调整各个子空间边界划分，将导致各个 Join-bolt Task 的负载不均衡，从而降低 EMD-DDSJ 技术的整体处理吞吐率. 实际应用中产生的数据流普遍具有的时间关联性，即同一数据流上时间戳相近的元组的取值也相近，例如传感器采样数据流和监控视频流都具有这样的性质. 本文假设直方图数据流具有时间关联性. 首先给出 Join-bolt 上的连接负载代价模型(4.1 节)，并基于该代价模型和数据流的时间关联性提出基于反馈的负载均衡策略(4.2 节)，该策略利用基于反馈的数据划分规则获取算法(4.3 节)根据各个连接计算节点的负载反馈信息阶段性调整 Partition-bolt 的数据划分策略，实现各个连接计算节点间的负载均衡.

4.1　系统负载代价模型

EMD-DDSJ 技术中每个 Join-bolt Task 负责完成 R 数据流分片和 S 数据流分片之间基于 EMD 距离相似性连接，因而在一个执行周期内一个 Join-bolt Task 的连接代价 O 可量化为

$$O = |R'|O_{\text{filter}} + N_{\text{emd}}O_{\text{emd}} \tag{6}$$

其中，R'表示 Join-bolt Task 所获得的 R 数据流分片，$|R'|$表示 R'的元组基数，O_{filter}表示每个 R 数据流元组和 S 数据流分片进行相似性连接时的平均过滤代价，O_{emd}表示 EMD 距离的平均计算代价，N_{emd}表示 Join-bolt 上实际发生的 EMD 距离的计算次数. 因为 EMD 距离的计算复杂度远比其它 EMD 距离的过滤机制要高，即 $O_{\text{emd}} \gg O_{\text{filter}}$，而 EMD 距离的平均计算代价 O_{emd}的取值稳定，故 Join-bolt Task 的连接代价 O 可以用该 Join-bolt Task 上发生的 EMD 距离的计算次数，即 N_{emd}来近似量化，即 $O \approx N_{\text{emd}}$.

4.2　基于反馈的负载均衡策略

基于以上代价模型，本文提出基于反馈的负载均衡策略. 该策略将数据流的时间域划分为连续等长不相交的反馈周期(Feedback period). 要求每个 Join-bolt Task 在执行相似性连接时统计每个反馈周期内所发生的 EMD 计算次数作为其在该反馈周期的计算负载代价，并在反馈周期末将该统计信息以负载反馈元组的形式发送给上游的 Partition-bolt(如图 2 中虚线箭头所示). 基于数据流的时间关联性可认为每个 Join-bolt Task 在时间域上相邻的两个反馈周期上统计得到的计算负载代价具有相似性. 因此，Partition-bolt 在汇总所有 Join-bolt Task 在上一反馈周期统计的负载反馈元组之后，可基于该汇总信息调整直方图一维实数映射空间 $\Omega(Z)$ 内各子空间的划分边界，从而调整当前反馈周期 Partition-bolt 对数据流 R 的划分策略，实现当前反馈周期内各 Join-bolt Task 间的负载均衡. 可见，基于反馈的负载均衡策略需要 Join-bolt 和 Partition-bolt 共同参与完成.

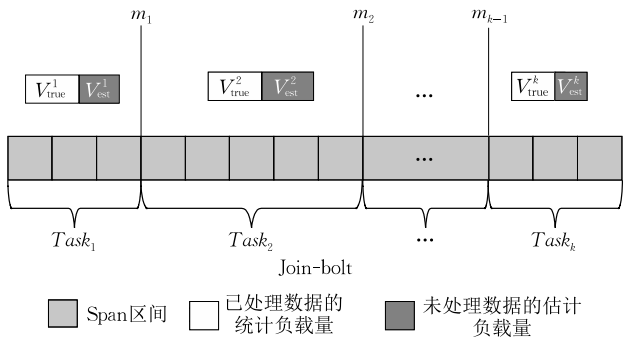


图 5　$\Omega(Z)$中的各个子空间和各个 Join-bolt Task 之间的映射关系

为了降低统计开销，如图 5 所示，首先将直方图的一维实数映射空间 $\Omega(Z)$划分为等长不重叠的区间(称之为 Span 区间)，因而每个 Span 区间涵盖了一个等长且连续的 key 值空间. 根据具体的负载划分策略，每个 Join-bolt Task 负责处理落在一个或多个连续 Span 区间内的 R 数据流元组，这样就保证了每个 Join-bolt Task 上 R 数据流分片的数据局部性. 如图 5 所示，m_1，…，m_{k-1}分别表示 k 个 Join-bolt Task 所负责的 $\Omega(Z)$中子空间的划分点，可见第 1 个 Join-bolt Task(对应图中 $Task_1$)负责处理落在前 3 个 Span 区间内的 R 数据流元组，第 2 个 Task(对应图中 $Task_2$)负责处理落在第 4 个至第 8 个 Span 区间内的 R 数据流元组. 在 Span 语义下，每个 Join-bolt Task 就能够基于一定的反馈周期 P 以 Span 为粒度统计本地的 EMD 距离计算负载信息. 下面依次介绍每个 Join-bolt Task 上统计 EMD 距离计算负载的具体过程和 Partition-bolt 上基于负载反馈元组调整数据流 R 的划分策略的过程.

4.2.1　Join-bolt 上的负载信息统计

假设直方图的一维实数映射空间 $\Omega(Z)$被划分为 u 个互相不重叠的 Span 区间，记为 $Span_x$，$1 \leq x \leq u$，基于反馈的负载均衡策略在 Join-bolt 上的主要工作流程如下：每个 Join-bolt Task 在执行基于 EMD 距离的相似性连接时以每个 Span 区间为粒

度统计当前反馈周期内落在每个 Span 区间内的 R 数据流元组所触发的 EMD 距离总计算次数.此处记第 x 个区间 $Span_x$ 上统计得到的 EMD 距离总计算次数为 $N_{\mathrm{EMD}}(x)$,则第 i 个 Join-bolt Task 在当前反馈周期结束时即可得到一个 u 维计算负载向量,记为 $\boldsymbol{V}^i_{\mathrm{true}}=\langle N_{\mathrm{EMD}}(1),\cdots,N_{\mathrm{EMD}}(u)\rangle$,对应图 5 中灰色斜线阴影方框.由于某些 Join-bolt Task 在当期反馈周期结束时可能仍有未处理完的 R 数据流元组,导致负载向量 $\boldsymbol{V}^i_{\mathrm{true}}$ 的统计信息不完整.为了得到完备的计算负载信息,需要每个 Join-bolt Task 基于当前反馈周期结束时已处理的 R 数据流元组所产生的 EMD 计算次数对当前反馈周期结束时未处理的 R 数据流元组即将在下一反馈周期产生的 EMD 计算次数进行估计.记第 i 个 Join-bolt Task 对当期未处理的 R 数据流元组在下一反馈周期产生的 EMD 计算次数的估计值为 V^i_{est},V^i_{est} 对应图 5 中的黑色竖线阴影方框,其计算方法如式(7)所示.

$$V^i_{\mathrm{est}}=(|R_i|-|R'_i|)\times\frac{N^i_{\mathrm{EMD}}}{|R'_i|}(1\leqslant i\leqslant k) \quad (7)$$

其中,$|R_i|$ 表示第 i 个 Join-bolt Task 在当前反馈周期开始时所获得的未处理的 R 数据流元组的基数,$|R'_i|$ 表示第 i 个 Join-bolt Task 在当前反馈周期结束时已处理的 R 数据流元组的基数,则 $|R_i|-|R'_i|$ 表示第 i 个 Join-bolt Task 在当前反馈周期结束时尚未处理的 R 数据流元组的基数.$N^i_{\mathrm{EMD}}=\sum_x N_{\mathrm{EMD}}(x)$(其中,$N_{\mathrm{EMD}}(x)\in\boldsymbol{V}^i_{\mathrm{true}}$)表示第 i 个 Join-bolt Task 在当前反馈周期结束时在所有 Span 区间上统计得到的 EMD 距离计算次数的和,则 $N^i_{\mathrm{EMD}}/|R'_i|$ 表示第 i 个 Join-bolt Task 在当前反馈周期内处理每个 R 数据流元组的过程中产生的平均 EMD 计算次数.当前反馈周期结束时,每个 Join-bolt Task 将统计得到的本周期已处理的 EMD 距离计算负载 $\boldsymbol{V}^i_{\mathrm{true}}$ 和估算得到的下一反馈周期将要处理的 EMD 距离计算负载 V^i_{est} 打包成负载反馈元组,形式为 $F^i_M=\langle\boldsymbol{V}^i_{\mathrm{true}},V^i_{\mathrm{est}}\rangle$,发送给上游的 Partition-bolt 组件.

4.2.2 Partition-bolt 上的负载划分调整策略

当 Partition-bolt 组件收集到所有下游 Join-bolt Task 发送的上一周期的负载反馈元组之后,需要根据这些反馈负载信息重新确定 k 个 Join-bolt Task 各自负责的直方图一维实数映射空间 $\Omega(Z)$ 的子空间的划分点,即图 5 所示的 $m_1,\cdots,m_{k-1}$,计算过程如下:

(1)根据所收集的已处理元组的统计信息得到负载分布信息.将 k 个 Join-bolt Task 反馈发回的 k 条负载反馈元组,即 $\{F^1_M,\cdots,F^k_M\}$ 的 $V^i_{\mathrm{true}}(1\leqslant i\leqslant k)$ 分量进行累加得到以 Span 区间为统计粒度的 EMD 计算次数在直方图一维实数映射空间 $\Omega(Z)$ 中的分布信息,记为 $\boldsymbol{D}_{\Omega(Z)}$.

(2)基于负载分布信息计算下一反馈周期给每个 Join-bolt Task 分配的 EMD 计算负载量的估计值.考虑到第 i 个 Join-bolt Task 发回的反馈元组 $F^i_M=\langle\boldsymbol{V}^i_{\mathrm{true}},V^i_{\mathrm{est}}\rangle$ 的 V^i_{est} 分量相当于下一反馈周期已经划分到第 i 个 Join-bolt Task 的 EMD 计算负载量,在决定下一反馈周期各个 Join-bolt Task 在 $\Omega(Z)$ 中的子空间的划分边界时应该考虑到 V^i_{est} 带来的影响,可以基于式(8)计算下一反馈周期 Partition-bolt 组件需要给第 i 个 Join-bolt Task 新分配的 EMD 计算负载量,记为 L_i.

$$L_i=\frac{\sum_{i=1}^{k}F^i_M\cdot V^i_{\mathrm{est}}+\boldsymbol{D}_{\Omega(Z)}}{k}-V^i_{\mathrm{est}} \quad (8)$$

其中,$\left(\sum_{i=1}^{k}F^i_M\cdot V^i_{\mathrm{est}}+\boldsymbol{D}_{\Omega(Z)}\right)/k$ 表示下一反馈周期每个 Join-bolt Task 需要处理的平均 EMD 计算负载量的估计值,该值减去 V^i_{est} 即可得到 Partition-bolt 组件在下一反馈周期需要给第 i 个 Join-bolt Task 新分配的 EMD 计算负载量的估计值.

(3)得到下一反馈周期的数据划分策略.基于 $L_i(1\leqslant i\leqslant k)$,即可调用 3.3 节提出的基于反馈的数据划分规则获取算法(算法 2)在 $\boldsymbol{D}_{\Omega(Z)}$ 分布上以 Span 区间为粒度求解各个 Join-bolt Task 在直方图一维实数映射空间 $\Omega(Z)$ 中所负责的子空间的边界值,即求得 $m_1,\cdots,m_{k-1}$,从而得到 $\Omega(Z)$ 中 k 个 Join-bolt Task 和 k 个子空间的新映射关系.

在下一反馈周期中,Partition-bolt 组件将基于所得到的新的数据划分策略(即 $\Omega(Z)$ 中子空间的边界值 $m_1,\cdots,m_{k-1}$)决定各个 Join-bolt Task 的数据划分.

需要注意,当 $\boldsymbol{D}_{\Omega(Z)}$ 所示的 EMD 距离计算次数在直方图一维实数映射空间 $\Omega(Z)$ 上的分布存在倾斜时,某个 Span 区间所对应的 EMD 距离计算次数较别的 Span 区间显著较大(称为热度 Span 区间),导致该 Span 区间无论在下一反馈周期分给哪个 Join-bolt Task 都可能使该 Join-bolt Task 的 EMD 计算负载量激增,导致各个 Join-bolt Task 间负载不均衡.此处认为当区间 $Span_i$ 的 EMD 距离计算次数超过下一反馈周期每个 Join-bolt Task 需要处理的平均 EMD 计算负载量的估计值 $\left(\sum_{i=1}^{k}F^i_M\cdot V^i_{\mathrm{est}}+\right.$

$\boldsymbol{D}_{\Omega(Z)}\big)/k$ 时，该 Span 区间为热度 Span 区间，记为 $Span_i^{\text{hot}}$. 针对 $\boldsymbol{D}_{\Omega(Z)}$ 内存在热度区间的情况，负载划分策略会首先求解 $\boldsymbol{D}_{\Omega(Z)}$ 中包含的所有热度区间，得到热度区间集合，记为 $\{Span_i^{\text{hot}}\}$，并在下一反馈周期中将落在热度区间 $Span_i^{\text{hot}}$ 内的 R 数据流元组随机发送给所负责的 Span 区间和 $Span_i^{\text{hot}}$ 区间前后临近的两个 Join-bolt Task 上.

4.3　基于反馈的数据划分规则获取算法

算法 2 展示了基于反馈的数据划分规则获取算法. Partition-bolt 每当收到下游某个 Join-bolt Task（设为 $Task_i$）所发送的上一反馈周期的负载反馈元组 $F_M^i=\langle \boldsymbol{V}_{\text{true}}^i, V_{\text{est}}^i\rangle$，即将该 Join-bolt Task 的接收标志位 T_i 设置为 TRUE（行 1～2）. 当 Partition-bolt 接收到所有 Join-bolt Task 的负载反馈元组之后（行 3），将基于这些负载反馈元组生成下一反馈周期 Partition-bolt 对 R 数据流的数据划分规则（行 4～17）. 具体过程为：首先基于这些计算负载反馈元组计算得到以 Span 区间为统计粒度的 EMD 计算次数在直方图一维实数映射空间 $\Omega(Z)$ 中的分布信息 $\boldsymbol{D}_{\Omega(Z)}$（行 4～5）；其次基于式（8）计算出下一反馈周期需要给每个 Join-bolt Task 新分配的 EMD 计算负载量 L_i（行 6～7）；紧接着获取 $\boldsymbol{D}_{\Omega(Z)}$ 中的热度区间集合 hotSpanSet（行 8～10）；然后遍历 $\boldsymbol{D}_{\Omega(Z)}$ 中所有非热度区间，基于每个 Join-bolt Task 在下一反馈周期的估计计算负载值 L_i 得到下一反馈周期每个 Join-bolt Task 所负责的 $\Omega(Z)$ 子空间的划分边界点集合 $\{m_1,\cdots,m_{k-1}\}$，并将该集合添加到 R 数据流划分规则集变量 *partitionRule* 中（行 11～16）；接着将热度区间集合 *hotSpanSet* 中每个热度区间内 R 数据流元组的划分规则同样添加到 R 数据流划分规则集变量 *partitionRule* 中（行 17～19）；最后重置各个 Join-bolt Task 的接收标志位（行 20～21），将向量 $\boldsymbol{D}_{\Omega(Z)}$ 中的各维数据置 0（行 22），并返回 R 数据流划分规则集变量 *partitionRule*（行 23）.

算法 2.　DataLocalityGrouping.

输入：Join-bolt 组件的 Task 并行度 k；$\Omega(Z)$ 中 Span 区间的个数 l；Span 区间的设定宽度 w_{span}；每 Join-bolt Task 所发送的上一反馈周期的负载反馈元组集合 $\{F_M^i\}$

输出：R 数据流的新数据划分规则集 *partitionRule*

1. FOR EACH received feedback workload tuple $F_M^i=\langle \boldsymbol{V}_{\text{true}}^i, V_{\text{est}}^i\rangle$ DO
2. T_i=TRUE;
3. IF (T_1 && T_2 && … && T_k) THEN
4. FOR i from 1 to k DO
5. $\boldsymbol{D}_{\Omega(Z)}$ += $F_M^i.\boldsymbol{V}_{\text{true}}^i$;
6. FOR i from 1 to k DO
7. calculate L_i based on Eq. (8);
8. FOR EACH j from 1 to l DO
/*$Span_j$ 为 $\boldsymbol{D}_{\Omega(Z)}$ 中第 j 个 Span 对应的计算负载量*/
9. IF $\left(Span_j>\left(\sum_{i=1}^{k}F_M^i.V_{\text{est}}^i+\boldsymbol{D}_{\Omega(Z)}\right)/k\right)$ THEN
10. *hotSpanSet.insert*($Span_j$)
11. FOR i from 1 to k DO
12. FOR EACH j from 1 to l DO
13. IF(*hotSpanSet.exist*($Span_j$)==FALSE)THEN
14. $L_i=L_i-Span_j$;
15. IF L_i<=0 THEN
16. *partitionRule.add*($m_i=j*w_{span}$);
17. FOR EACH $Span_j$ in *hotSpanSet* DO
18. IF($j*w_{\text{span}}\in[m_{i-1},m_i]$) THEN
19. *partitionRule.add*(R stream tuples in $Span_j$ will be randomly assigned to Join-bolt $Task_{i-1}$ and $Task_i$);
20. FOR i from 1 to k DO
21. T_i=FALSE;
22. set each dimension in $\boldsymbol{D}_{\Omega(Z)}$ to 0;
23. RETURN *partitionRule*

5　代价分析

本节对本文提出的 EMD-DDSJ 技术、2.2.2 节所述的 ShuffleJ 技术和 2.2.3 节所述的 Head-SJ 技术的算法复杂度进行对比分析. 让 $|R|$ 和 $|S|$ 分别表示数据流 R 和数据流 S 的元组基数. 并假设 Join-bolt 上有 k 个并行执行的 Task. 因为无论是 EMD-DDSJ 技术、ShuffleJ 技术还是 Head-SJ 技术均将数据流 S 上的每个直方图元组广播给 k 个 Join-bolt Task，同时将数据流 R 上的每个元组发送给一个特定的Join-bolt Task，因而 EMD-DDSJ 技术和 ShuffleJ 技术的网络传输代价均为 $O(|R|+k\times|S|)$.

让 $Avg_{\text{EMD-DDSJ}}$、Avg_{ShuffleJ} 和 $Avg_{\text{Head-SJ}}$ 分别表示 EMD-DDSJ 技术、ShuffleJ 技术和 Head-SJ 技术处理每个 R 数据流元组和 S 数据流分片之间相似性连接时的平均计算代价，则 EMD-DDSJ 技术、ShuffleJ 技术和 Head-SJ 技术在所有 Join-bolt Task 上的总计算代价分别为 $O(|R|Avg_{\text{EMD-DDSJ}})$、$O(|R|Avg_{\text{ShuffleJ}})$ 和 $O(|R|Avg_{\text{Head-SJ}})$. 因为 EMD-DDSJ 技术能够利用数据局部性过滤掉更多不相似直方图元组对之间的 EMD 距离求精计算，所以相似性连接的平均计算代价 $Avg_{\text{EMD-DDSJ}}$ 要小于 Avg_{ShuffleJ} 和

$Avg_{\text{Head-SJ}}$. 因此 EMD-DDSJ 技术在所有 Join-bolt Task 上的总计算代价小于 ShuffleJ 技术或 Head-SJ 技术在所有 Join-bolt Task 上的总计算代价.

6 实验结果与分析

6.1 实验设置

本节使用真实的流数据集对本文提出的 EMD-DDSJ 技术、2.2.2 节提到的 ShuffleJ 技术、2.2.3 节提到的 Head-SJ 技术进行比较评估. 同时还将基于 EMD 距离的大规模数据相似性连接的分布式批处理技术 Head-Join 在 Apache Storm 框架下进行实现,命名为 Head-SJ 技术,并将 Head-SJ 技术与本文提出的 EMD-DDSJ 技术进行实验比较. 实验使用的集群由 5 台计算节点构成. 每个计算节点的配置是 16 核 CPU、8 GB 内存,运行 64 位 Linux 操作系统. 每个节点拥有 50 GB 的硬盘容量,节点间用百兆交换机相连,集群上部署的 Apache Storm 框架的版本为 0.9.1,系统源代码的实现语言为 Java.

如表 2 所示,实验基于 3 部电影视频产生真实流数据集. 具体而言,抽取每个电影的视频关键帧,并从抽取的每个视频帧中提取出包含 256 个数据桶的灰度直方图即得到 1 个直方图元组,依此每个电影视频均可转换为 1 条直方图数据流(元组基数为 100 万). 这里将基于 3 部电影视频分别得到的 3 条直方图数据流并分别记为 S、SS 和 M. 如表 2 所示,由于第 2 部电影视频是将第 1 部电影视频每一帧的亮度降低 58f 得到的,因而基于第 2 部电影视频得到的直方图数据流 SS 和基于第 1 部电影视频得到的直方图数据流 S 很相似性. 用符号 $\bowtie$ 表示面向滑动窗口语义且基于 EMD 距离的数据流分布式相似性连接操作. 实验中基于欧几里德距离和直方图每个数据桶的索引号计算数据桶之间的地面距离矩阵 $\boldsymbol{C}=[c_{ij}]$,并将该地面距离矩阵用于 EMD 距离的计算.

表 2 三部电影视频的截图

电影 1(对应数据流 S) 电影 2(对应数据 SS) 电影 3(对应数据流 M)

表 3 给出了实验中相关参数的设置情况,加粗的数字表示该参数的默认值. 其中滑动窗口的步长是指每次窗口在时间域上向前滑动的距离. 每个实验数据都是运行 5 次实验得到的平均值.

表 3 实验参数设置(粗体为默认值)

参数	数值
相似性阈值 θ	0.1, 0.3, **0.5**, 0.7, 0.9
Spout 的元组发射速率	**1000** tuples/s
滑动窗口大小$\lvert W\rvert$	1s, 3s, **5s**, 7s, 9s, 11s
滑动窗口滑动步长	1s
Join-bolt Task 的并行度	2, **5**, 8, 11, 14, 17
数据流元组基数	**100** 万

6.2 实验结果与分析

首先测试了反馈周期 P 和 Span 区间的大小(详见第 3 节)对 EMD-DDSJ 技术的影响. 定义 Storm 中 Join-bolt 组件的负载不均衡率=(Join-bolt Task 的最大负载量-所有 Join-bolt Task 的平均负载量)/所有 Join-bolt Task 的平均负载量. 图 6 展示了 EMD-DDSJ 技术中反馈周期的大小和 Join-bolt 组件的负载不均衡率之间的关系. 可见随着反馈周期 P 的增大,无论是在处理数据流 S 和 SS 之间的连接还是在处理数据流 S 和 M 之间的连接,Join-bolt 组件的负载不均衡率呈现先递减后递增的趋势,且反馈周期 $P=2.5$ s 时,Join-bolt 组件的负载不均衡率达到最低. 这是因为反馈周期设置越短,反馈周期内收集到的各 Join-bolt Task 对 EMD 距离计算负载的统计信息越少,基于有限的 EMD 距离计算负载统计信息不能准确指导下一反馈周期中 R 数据流元组的划分,造成下一反馈周期中 Join-bolt 组件的负载不均衡率偏高. 当反馈周期 $P>2.5$ s 时,随着反馈周期 P 变长,Join-bolt 组件的负载不均衡率会不断增加,这是因为反馈周期 P 增大则会弱化相邻反馈周期之间计算负载统计信息的时间关联性,从而降低当前的负载反馈信息对下一反馈周期数据划分策略的指导准确性. 在接下来的实验中,统一设置反馈周期 $P=2.5$ s.

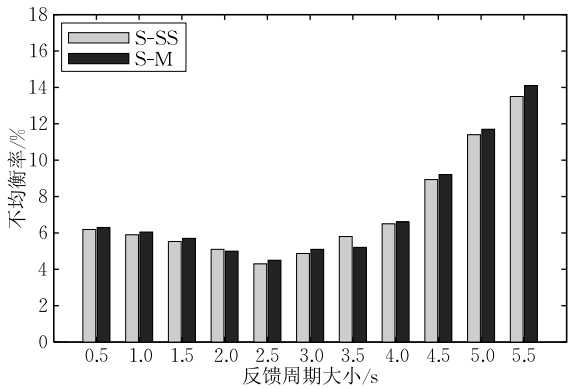


图 6 反馈周期 P 大小对负载均衡性的影响

图 7 展示了 Span 区间的大小对 EMD-DDSJ 技术执行时间的影响. 可见,不论是执行 S$\bowtie$SS 连

接还是 S ⋈ M 连接，EMD-DDSJ 技术的执行时间均呈现随 Span 增大先降低后增大的趋势，且 Span 区间大小为 15 时算法的执行时间最短. 这是因为 Span 区间设置越小，虽然对直方图一维实数映射空间内各键值对应的计算负载的统计越精细，但考虑到下一反馈周期的 EMD 距离计算负载在映射空间内的分布可能存在一定的偏移，因而统计得越精细反倒会降低统计信息对下一反馈周期 EMD 距离计算负载划分的指导意义. 反之，如果 Span 区间设置得过大则统计信息粒度过粗糙也不能准确指导下一反馈周期 EMD 距离计算负载的划分. 在接下来的实验中设置 Span 为 15.

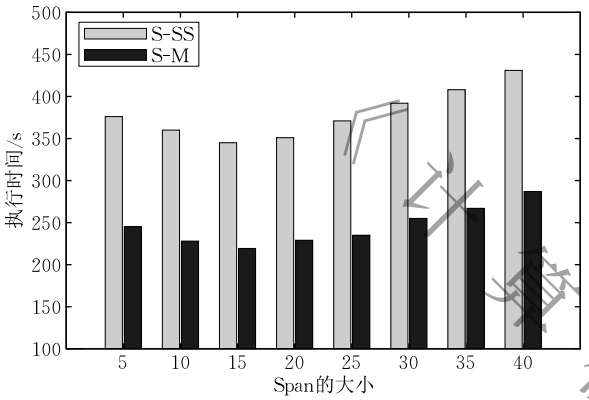


图 7　Span 大小对运行时间的影响

图 8 展示了不同技术在处理不同数据流连接时 Join-bolt 组件上并行执行的各 Task 间的负载均衡性. ShuffleJ 技术随机给每个 Join-bolt Task 分配了近似等量且同分布的数据，负载均衡性最好. 值得肯定是，EMD-DDSJ 技术所提出的基于反馈的负载均衡策略在均衡各 Join-bolt Task 的计算负载方面取得了显著成效，使 EMD-DDSJ 技术的负载不均衡率仅略高于 ShuffleJ 技术的负载不均衡率. Head-SJ 技术的负载不均衡率比 EMD-DDSJ 技术高 2 倍多，这是因为 Head-SJ 技术不能基于数据流数据分布的变化自适应地调整其数据划分策略，导致数据流数据分布变化时部分 Join-bolt Task 上的负载过高.

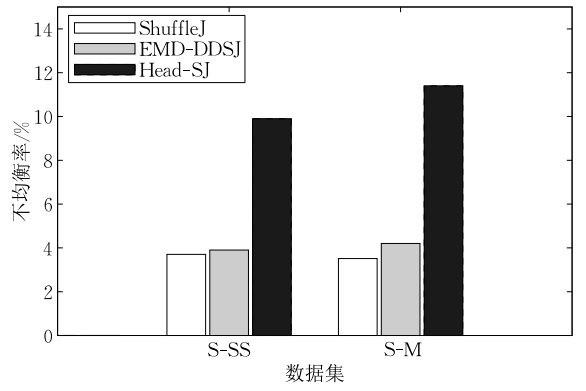
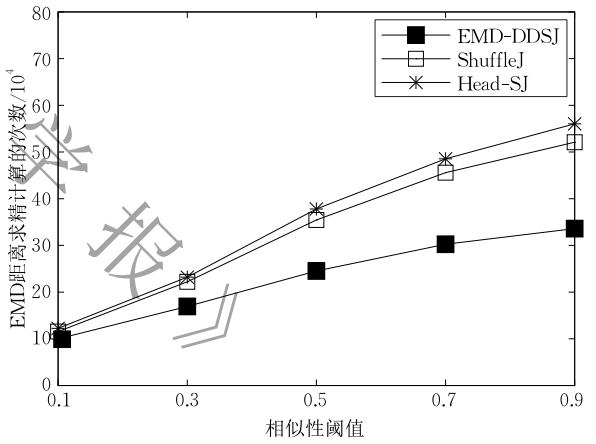
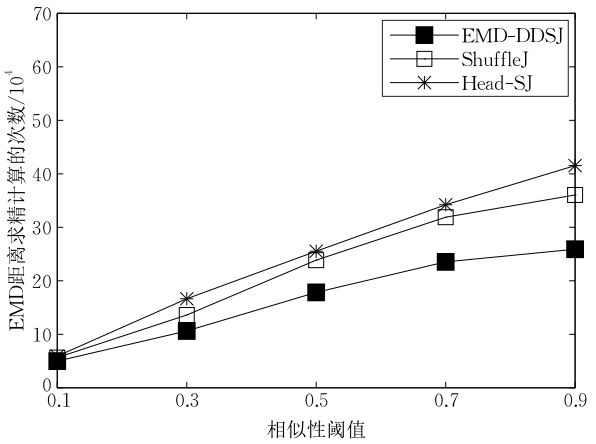


图 8　Join-bolt 上的负载均衡性测试

图 9 测试了不同数据集上相似性阈值的大小对各个技术中产生的 EMD 距离求精计算总次数的影响. 其 EMD 距离求精计算总次数是指不同技术在处理两条直方图数据流间基于 EMD 距离的相似性连接时经过图 3 所示的过滤链过滤后仍需要进行的 EMD 距离计算的总次数. 如图 9 所示，随着相似性阈值的增大，ShuffleJ、EMD-DDSJ 和 Head-SJ 技术各自的 EMD 距离求精计算总次数都呈现出上升的趋势. 这是因为阈值越大则相似性连接的结果候选集越大，因而 EMD 距离求精计算的次数也随之增加. 从图 9 中还可观察到，无论是执行 SSS 连接还是执行 SM 连接，EMD-DDSJ 技术比 ShuffleJ 技术的 EMD 距离求精计算总次数降低了 12%～36%，比 Head-SJ 技术降低了 15%～40%. 这是因为 EMD-DDSJ 技术使用了基于数据局部性的数据流 R 划分策略，每个 Join-bolt Task 可以利用 R 数据流分片良好的数据局部性来增强三角不等式过滤



(a) S⋈SS

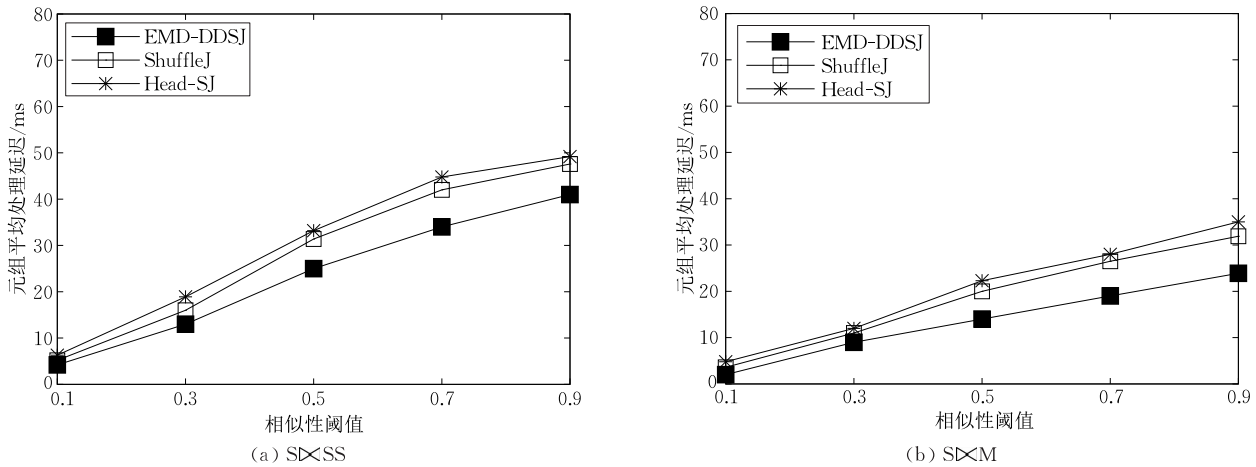


(b) S⋈M

图 9　相似性阈值的大小对 EMD 计算次数的影响

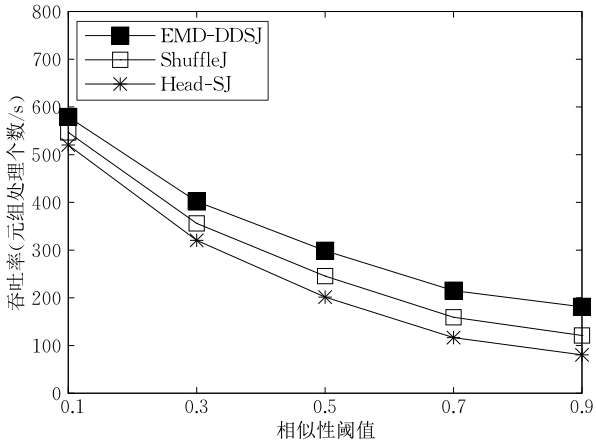
机制和新可行解过滤机制对不相似直方图元组对之间 EMD 距离计算的过滤性能.同时也说明 Head-SJ 技术使用的过滤链的总体过滤性能比 ShuffleJ 技术使用的过滤链的总体过滤性能略弱.这是因为 Head-SJ 技术没有充分利用 Join-bolt Task 的数据局部性过滤不相似元组对之间的 EMD 距离计算.还可观察到随着阈值增大,EMD-DDSJ 技术的 EMD 距离计算次数和 ShuffleJ 技术的 EMD 距离计算次数之间的差值增大,这是因为阈值越大则查询候选集越多,EMD-DDSJ 技术利用查询候选集的数据局部性对不相似直方图元组对之间 EMD 距离计算进行过滤的性能就越好.同时,由于数据流 S 和 SS 比 S 和 M 更相似,在执行 S⋈SS 连接时每个 Join-bolt Task 上 R 数据流分片的数据局部性更好,因而在执行 S⋈SS 连接时 EMD-DDSJ 技术对 EMD 距离计算的过滤性能也就更好.

图 10 和图 11 基于不同数据集分别展示了三种技术在相似性阈值变化时的元组平均处理延迟和系统处理吞吐率变化情况.考虑到 Join-bolt 是整个系统的处理瓶颈,元组平均处理延迟定义为元组到达 Join-bolt Task 的时刻和该元组被 Join-bolt Task 处理完的时刻之间的平均时间差.处理吞吐率定义为系统每分钟处理的 R 数据流元组的个数.从图 10 和图 11 可看出,在不同的数据集上,随着相似性阈值的增大三种技术的元组平均处理延迟均呈现上升趋势,同时它们的处理吞吐率都呈现下降的趋势.这是因为相似性阈值越大则相似性连接处理中的结果侯选集增大,导致 EMD 距离的求精计算次数增加从而增加了元组的平均处理延迟,降低了系统的处理吞吐率.还可观察到在执行 S⋈SS 时 EMD-DDSJ 比 ShuffleJ 的元组的平均处理延迟降低了 14%～20%(图 10(a)),处理吞吐率提高了 6%～49%(图 11(a));EMD-DDSJ 比 Head-SJ 的元组的平均处理延迟降低了 17%～33%(图 10(a)),处理吞吐率提高了 11%～104%(图 11(a));在执行 SM 时,EMD-DDSJ 比 ShuffleJ 的元组的平均处理延迟降低了 18%～44%(图 10(b)),处理吞吐率提高了 8%～33%(图 11(b)),EMD-DDSJ 比 Head-SJ 的元组的平均处理延迟降低了 25%～58%(图 10(b)),处理吞吐率提高了 13%～81%(图 11(b)).这一方面是因为 EMD-DDSJ 技术使用了基于数据局部性的数据流 R 划分策略,并利用该数据局部性强化了三角不等式过滤机制和新可行解过滤机制的过滤性能,因此能比 ShuffleJ 技术和 Head-SJ 技术过滤掉更多的 EMD 距离的计算(详见图 9).另一方面是因为 EMD-DDSJ 技术基于 R 数据流上数据分布的动态变化调整各个 Join-bolt Task 的数据划分,使得各个 Join-bolt Task 的负载均衡性较好,而 Head-SJ 技术由于没有负载均衡策略导致其部分 Join-bolt Task 过载,元组的平均处理延迟和处理吞吐率都显著差于 EMD-DDSJ 技术.同时从图 11 还可观察到,三种技术在处理 S⋈SS 连接时比在处理 S⋈M 连接时在处理吞吐率方面有显著提高,这是因为数据流 S 和 SS 之间比数据流 S 和 M 之间更相似,因而执行 S⋈SS 的元组平均处理延时比执行 S⋈M 的更高(图 10).最后还可观察到,随着相似性阈值的增大,EMD-DDSJ 技术比 ShuffleJ 技术和 Head-SJ 技术在处理吞吐率方面的优势会进一步增强,正是由于 EMD-DDSJ 技术在阈值增大时对 EMD 距离计算的过滤性能更好(详见图 9).

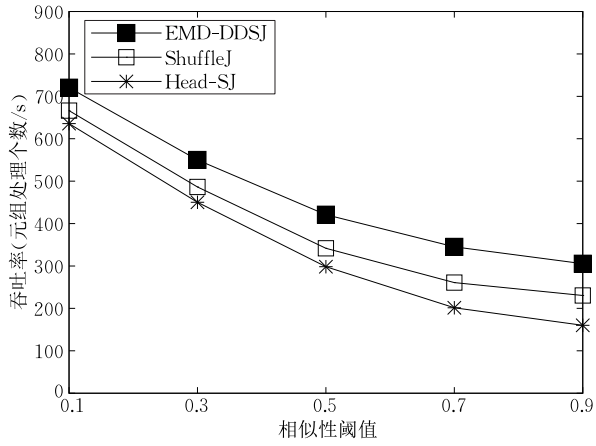


(a) S⋈SS (b) S⋈M

图 10 相似性阈值的大小对元组平均处理延迟的影响



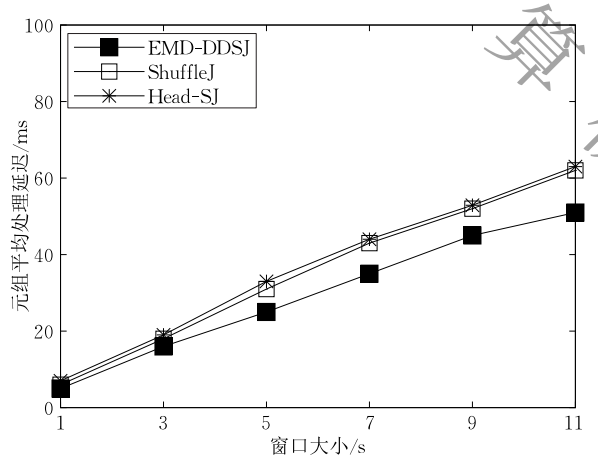
(a) S⋈SS



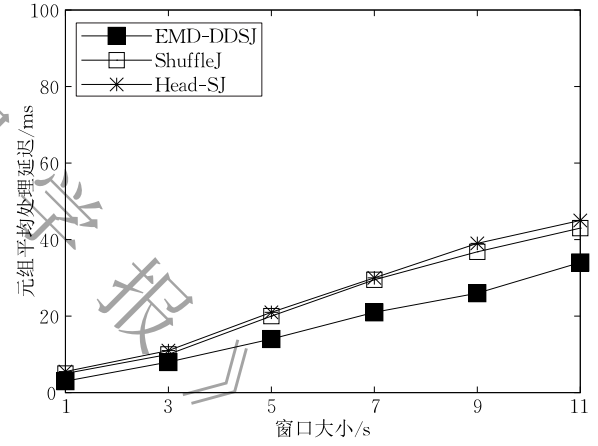
(b) S⋈M

图 11　相似性阈值的大小对处理吞吐率影响

图 12 和图 13 基于不同数据集分别展示了三种技术在滑动窗口大小变化时元组的平均处理延迟和系统处理吞吐率变化情况. 可见,随着滑动窗口大小的增大,三种技术的元组平均处理延迟均呈现上升趋势,同时它们的处理吞吐率都呈现出下降的趋势. 这是因为随着滑动窗口的增大,相似性连接处理中的结果后选集也增大,导致 EMD 距离的求精计算次数增加从而影响了元组平均处理延迟和整个系统的处理吞吐率. 同时从图 13 中还可观察到,当滑动窗口的大小增大时,无论在处理 S⋈SS 连接时还是

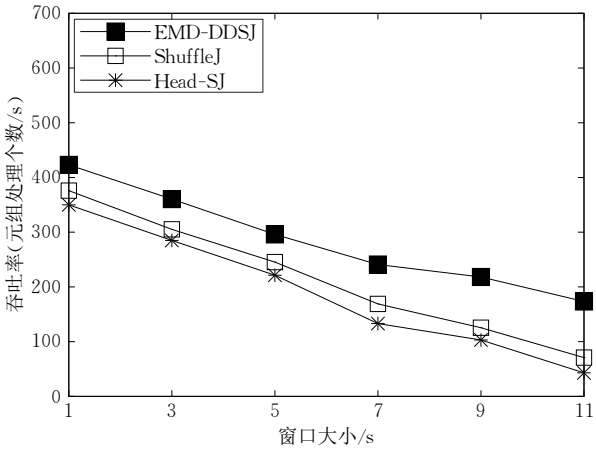


(a) S⋈SS

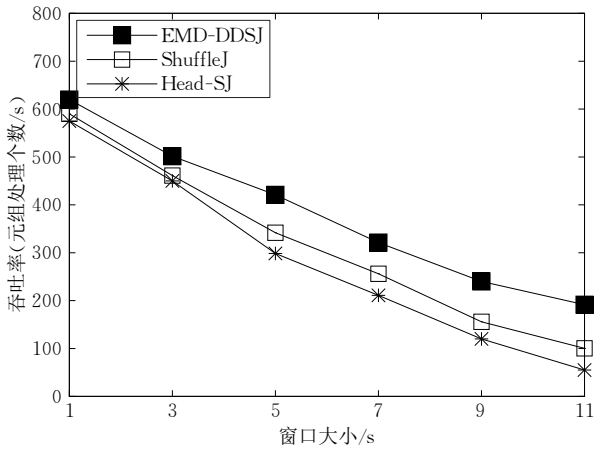


(b) S⋈M

图 12　窗口大小对元组平均处理延迟的影响



(a) S⋈SS



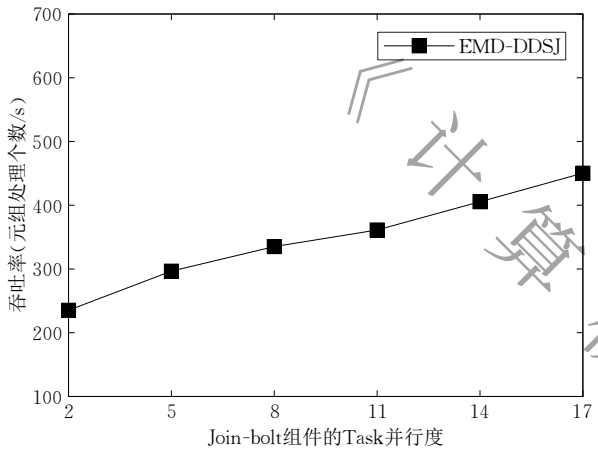
(b) S⋈M

图 13　窗口大小对处理吞吐率的影响

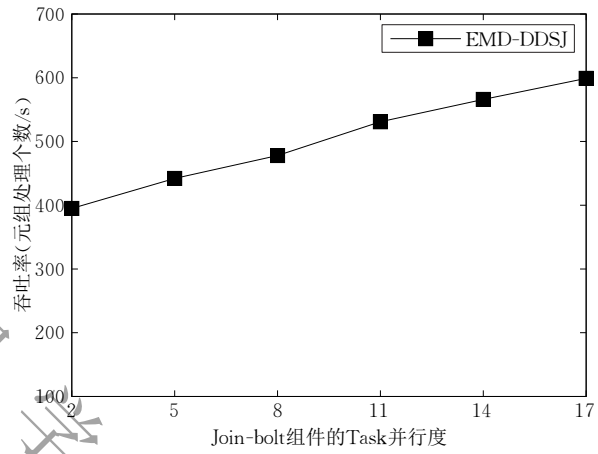
在处理 S⋈M 连接时，EMD-DDSJ 技术比 ShuffleJ 技术和 Head-SJ 技术在处理吞吐率上的优势都呈现出逐渐增强的趋势. 尤其是当滑动窗口大小为 11 s 时，EMD-DDSJ 技术的系统处理吞吐率比 ShuffleJ 技术的系统处理吞吐率提高了 1.4 倍，更是比 Head-SJ 技术的系统处理吞吐率提高了 3 倍. 这是因为窗口越大参与计算的元组就越多，相似性连接过程中所产生的结果候选集也就越大，在这种情况下 EMD-DDSJ 技术利用 Join-bolt Task 上的数据局部性对结果候选集进行过滤所带来的优势就更明显，因而 EMD-DDSJ 技术在处理吞吐率上的提升优势就越显著.

综合图 11 和图 13 的结论，当相似性阈值或滑动窗口大小增大的时候 EMD-DDSJ 技术更有优势，可以很好地支持对相似性阈值和滑动窗口大小有不同需求的应用.

图 14 展示了 EMD-DDSJ 技术的系统可扩展性. 通过在不同数据集上改变 Join-bolt 组件中并行执行的 Task 的数目测试不同技术的处理吞吐率情况. 从图 14 中可以看出，随着 Join-bolt 组件中并行执行的 Task 数目的增加，系统的处理吞吐率呈线性递增趋势，表明 EMD-DDSJ 技术具有良好的可扩展性. 且当 Join-bolt 的 Task 并行度设置为 2 时，EMD-DDSJ 技术每秒已经可以处理 235～395 个 R 数据流元组，考虑到视频流或传感器数据流每秒钟一般产生不超过 40 个流元组，因此本文提出的 EMD-DDSJ 技术可以满足应用的实时性分析要求.



(a) S⋈SS



(b) S⋈M

图 14 可扩展性测试

7 结束语

大数据时代数据流源源不断地快速产生，数据流之间的相似性连接是分析和挖掘数据流的重要操作. 分布式并行计算是提高数据流相似性连接处理效率的有效手段. EMD 距离可以更准确地量化直方图元组之间的相似性却具有高达三次方的计算复杂度，阻碍了其在数据流相似性连接问题中的应用. 基于开源的 Apache Storm 数据流分布式计算框架，本文提出了滑动窗口语义下基于 EMD 距离的数据流分布式相似性连接技术 EMD-DDSJ. 该技术基于直方图至一维实数空间的映射机制设计了数据划分策略，维护了各个分布式连接计算节点上面向 EMD 距离的数据局部性，并利用该数据局部性有效降低了各个连接计算节点上的 EMD 距离求精计算次数. 同时，该技术还提出了基于反馈的负载均衡策略，有效提升了系统对数据流的整体处理吞吐率. 真实流数据集上的实验表明了 EMD-DDSJ 技术的有效性和可扩展性，可以很好地支持实时数据分析应用. 未来将着眼于增强系统在处理数据流爆发时的系统弹性.

参 考 文 献

[1] Liu P, Guo J M, Chamnongthai K, Prasetyo H. Fusion of color histogram and LBP-based features for texture image retrieval and classification. Information Sciences, 2017, 390: 95-111

[2] Deshpande A, Guestrin C, Madden S R, et al. Model-based approximate querying in sensor networks. The VLDB Journal, 2005, 14(4): 417-443

[3] Lian X, Chen L. Efficient join processing on uncertain data streams. IEEE Transactions on Knowledge & Data Engineering, 2009, 23(11): 1718-1734

[4] Rubner Y, Tomasi C, Guibas L J. The earth mover's distance as a metric for image retrieval. International Journal of Computer Vision, 2000, 40(2): 99-121

[5] Nirmala S G, Kiran K P. Earth mover's distance-based CBIR using adaptive regularized Kernel fuzzy C-means method of liver cirrhosis histopathological segmentation. International Journal of Signal and Imaging Systems Engineering, 2017, 10(1/2): 39

[6] Xu J, Bai Q, Gu Y, Tung A K. Eudemon: A system for online video frame copy detection by earth mover's distance// Proceedings of the 28th International Conference on Data Engineering. Washington, USA, 2012: 1233-1236

[7] Xu J, Lei B, Gu Y, Winslett M, et al. Efficient similarity join based on earth mover's distance using MapReduce. IEEE Transactions on Knowledge & Data Engineering, 2015, 27(8): 2148-2162

[8] Zhao Q, Yang Z, Tao H. Differential earth mover's distance with its applications to visual tracking. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2010, 32(2): 274-287

[9] Huang J, Zhang R, Buyya R, Chen J. MELODY-JOIN: Efficient earth mover's distance similarity joins using MapReduce //Proceedings of the 28th International Conference on Data Engineering. Chicago, USA, 2014: 808-819

[10] Wichterich M, Assent I, Kranen P, et al. Efficient EMD-based similarity search in multimedia databases via flexible dimensionality reduction//Proceedings of the 2008 ACM SIGMOD/PODS Conference. Vancouver, Canada, 2008: 199-212

[11] Xu J, Zhang Z J, Tung A K, Yu G. Efficient and effective similarity search over probabilistic data based on earth mover's distance//Proceedings of the 36th International Conference on Very Large Data Bases. Singapore, 2010, 21(4): 535-559

[12] Ruttenberg B E, Singh A K. Indexing the earth mover's distance using normal distributions//Proceedings of the 36th International Conference on Very Large Data Bases. Seattle, USA, 2011, 5: 205-216

[13] Xu J, Zhang J Z, Song C, et al. EMD-DSJoin: Efficient similarity join over probabilistic data streams based on earth mover's distance//Proceedings of the 18th Asia Pacific Web Conference. Suzhou, China, 2016: 9-20

[14] Li L J, Ma M, Lei P, et al. A linear approximate algorithm for earth mover's distance with thresholded ground distance. Mathematical Problems in Engineering, 2014, 2014(3): 1-9

[15] Shirdhonkar S, Jacobs D W. Approximate earth mover's distance in linear time//Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. Columbus, USA, 2014: 1-8

[16] Li W, Osher S, Gangbo W. A fast algorithm for earth mover's distance based on optimal transport and L1 type regularization. 2016. arXiv:1609.07092

[17] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters//Proceedings of the Symposium on Operating Systems Design & Implementation. San Diego, USA, 2008, 51(1): 107-113

[18] Huang J, Zhang R, Buyya R, et al. HEADS-JOIN: Efficient earth mover's distance similarity joins on hadoop. IEEE Transactions on Parallel, Distributed Systems, 2016, 27(6): 1660-1673

[19] Lei Bin, Xu Jia, Gu Yu, Yu Ge. Parallel Top-k similarity join algorithm on probabilistic data based on earth mover's distance. Journal of Software, 2013, 24(Suppl.(2)): 188-199(in Chinese)
(雷斌，许嘉，谷峪，于戈. 概率数据上基于 EMD 距离的并行 Top-k 相似性连接算法. 软件学报，2013，24(Suppl.(2))：188-199)

[20] Fang J, Wang X, Zhang R, Zhou A Y. Flexible and adaptive stream join algorithm//Proceedings of the 18th Asia Pacific Web Conference. Suzhou, China, 2016: 3-16

[21] Lin Q, Ooi B C, Wang Z, Yu C. Scalable distributed stream join processing//Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. Melbourne, VIC, Australia. 2015: 811-825

[22] Nasir M A U, Morales G D F, Garciasoriano D, et al. Partial key grouping: Load-balanced partitioning of distributed streams. Computer Science, 2015, 106(2): 181-203

[23] Nasir M A U, Morales G D F, Kourtellis N, Serafini M. When two choices are not enough: Balancing at scale in distributed stream processing//Proceedings of the 32nd International Conference on Data Engineering. Helsinki, Finland, 2016: 137-148

[24] Xu J, Zhang Z J, Tung A K, Yu G. Efficient and effective similarity search over probabilistic data based on earth mover's distance. The VLDB Journal, 2012, 3(1): 758-769

[25] Assent I, Wenning A, Seidl T. Approximation Techniques for indexing the earth mover's distance in multimedia databases// Proceedings of the 32nd International Conference on Data Engineering. Helsinki, Finland, 2006: 11

XU Jia, Ph. D., associate professor. Her research interests include database theory and technology, graph data analysis, data privacy protection and distributed parallel computing of big data.

SONG Chao, M. S. candidate. His research interest is data stream management technology.

LV Pin, Ph. D., associate researcher. His research interests include wireless network and mobile computing, Internet of Things, network virtualization and network data analysis.

LI Tao-Shen, professor, His research interests include wireless mesh network, cloud computing and big data, network computing and information security and distributed database.

Background

The Earth Mover's Distance (EMD) is a powerful tool for data similarity measurement, because the similarity quantified by the EMD is in accordance with people's perception to similarities. However, EMD has cubic computational complexity, which hinders its wide application in real-time analysis of data streams. Plenty of efforts have been made in this field to reduce the computational complexity of the EMD. In this paper, an EMD-based distributed data stream similarity join technique is designed and implemented, which is more efficient and scalable compared with the related technique. This method is implemented based on Apache Storm, which is a well-known open-source distributed and parallel data stream processing framework. According to experimental results on physical-world data sets, the proposed technique is at most 1.4 times higher than the best related technique in terms of tuple processing throughput and decreases up to 44% of average tuple processing delay compared with the best related technique. Besides, with the growth of the similarity threshold or the size of sliding window, the throughput improvement ratio of EMD-DDSJ technique compared to the related technique will be further increased. This research work has considerable theoretical significance and practical value. It can serve as a precedent for other computationally intensive data stream analysis applications.

This work is supported by the National Natural Science Foundation of China under Grant No. 61402494, the Guangxi Natural Science Foundation under Grant No. 2015GXNSFBA139243, Special Funding for Guangxi Bagui Young Scholars, and partially supported by the Scientific Research Foundation of Guangxi University under Grant Nos. XGZ141182 and XGZ150322, and the Key Projects of Higher Education Undergraduate Teaching Reform Project in Guangxi under Grant No. 2017JGZ103. The aim of these research projects are to study efficient distributed similarity analysis techniques on fast-generated data streams based on EMD, including data stream distributed processing model, query optimization strategy, as well as system optimization mechanism. Research works related to these projects have been published in international top conferences and top journals such as VLDB 2010, ICDE 2012 (demo paper), ICDE 2016 (TKDE poster session), *The VLDB Journal*, *IEEE Transactions on Knowledge and Data Engineering*, etc.